

【爱加密】 Android APP 利用无效字节码防止工具逆向破解（一）

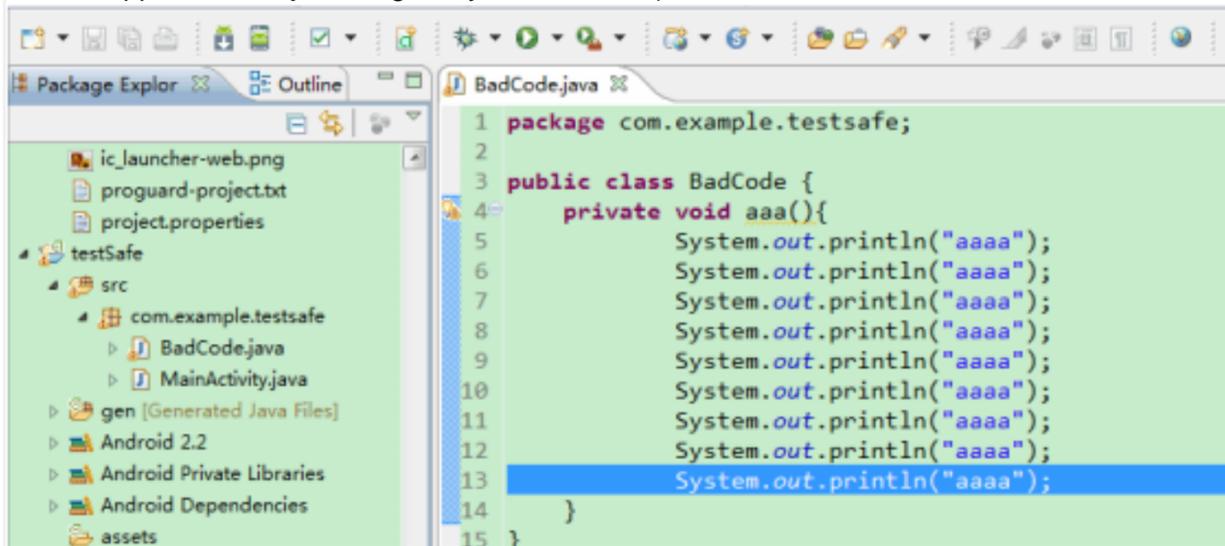
由于大部分逆向工具都是线性读取字节码并解析，当遇到无效字节码时，就会引起反编译工具字节码解析失败。我们可以插入无效字节码到 DEX 文件，但要保证该无效字节码永远不会被执行（否则您的程序就会崩溃了！）。

用到的工具：

IDA、C32Asm、DexFixer、Ijiami signer（爱加密签名工具）、
（由于百度文档限制，使用的工具无法添加连接。请自行到网上下载，带来的不便请谅解！）

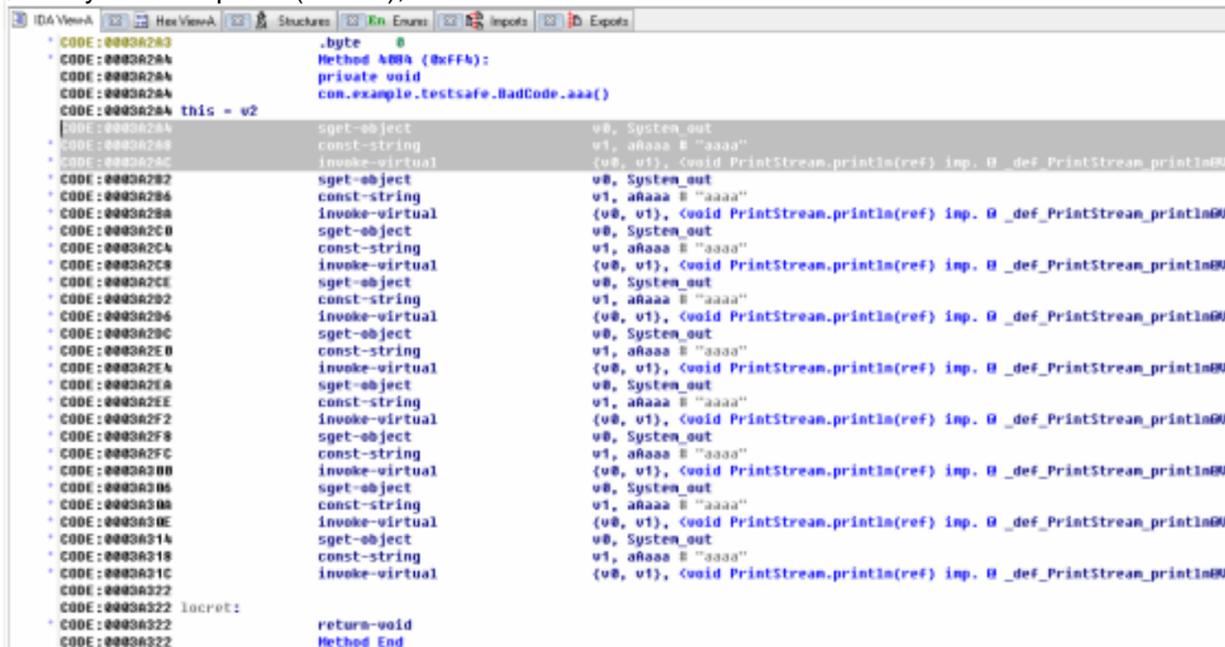


首先我们新建一个测试类。为了绕过 Dalvik 运行时代码验证，BadCode.java 要保证不被调用。（否则运行 app，会出现 java.lang.verifyerror 异常）



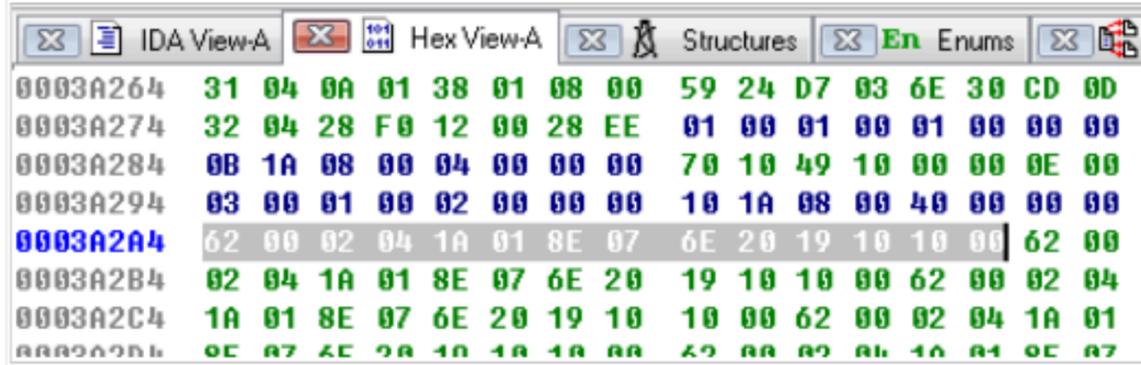
```
1 package com.example.testsafe;
2
3 public class BadCode {
4     private void aaa(){
5         System.out.println("aaaa");
6         System.out.println("aaaa");
7         System.out.println("aaaa");
8         System.out.println("aaaa");
9         System.out.println("aaaa");
10        System.out.println("aaaa");
11        System.out.println("aaaa");
12        System.out.println("aaaa");
13        System.out.println("aaaa");
14    }
15 }
```

然后生成 apk，用 ida 打开 classes.dex 并找到 BadCode 类的 aaa 方法。选中的三行代码对应 "System.out.println("aaaa");"



```
CODE:0003A2A3 .byte 0
CODE:0003A2A4 Method AAA (0xFF4):
CODE:0003A2A4 private void
CODE:0003A2A4 com.example.testsafe.BadCode.aaa()
CODE:0003A2A4 this = v2
CODE:0003A2A4 sget-object v0, System.out
CODE:0003A2A4 const-string v1, aaaa # "aaaa"
CODE:0003A2A4 invoke-virtual {v0, v1}, Cvoid PrintStream.println(ref) imp. 0_def_PrintStream_printlnBUL
CODE:0003A2A2 sget-object v0, System.out
CODE:0003A2A6 const-string v1, aaaa # "aaaa"
CODE:0003A2A6 invoke-virtual {v0, v1}, Cvoid PrintStream.println(ref) imp. 0_def_PrintStream_printlnBUL
CODE:0003A2C0 sget-object v0, System.out
CODE:0003A2C4 const-string v1, aaaa # "aaaa"
CODE:0003A2C8 invoke-virtual {v0, v1}, Cvoid PrintStream.println(ref) imp. 0_def_PrintStream_printlnBUL
CODE:0003A2CE sget-object v0, System.out
CODE:0003A2D2 const-string v1, aaaa # "aaaa"
CODE:0003A2D6 invoke-virtual {v0, v1}, Cvoid PrintStream.println(ref) imp. 0_def_PrintStream_printlnBUL
CODE:0003A2DC sget-object v0, System.out
CODE:0003A2E0 const-string v1, aaaa # "aaaa"
CODE:0003A2E4 invoke-virtual {v0, v1}, Cvoid PrintStream.println(ref) imp. 0_def_PrintStream_printlnBUL
CODE:0003A2EA sget-object v0, System.out
CODE:0003A2EE const-string v1, aaaa # "aaaa"
CODE:0003A2F2 invoke-virtual {v0, v1}, Cvoid PrintStream.println(ref) imp. 0_def_PrintStream_printlnBUL
CODE:0003A2F8 sget-object v0, System.out
CODE:0003A2FC const-string v1, aaaa # "aaaa"
CODE:0003A300 invoke-virtual {v0, v1}, Cvoid PrintStream.println(ref) imp. 0_def_PrintStream_printlnBUL
CODE:0003A306 sget-object v0, System.out
CODE:0003A30A const-string v1, aaaa # "aaaa"
CODE:0003A30E invoke-virtual {v0, v1}, Cvoid PrintStream.println(ref) imp. 0_def_PrintStream_printlnBUL
CODE:0003A314 sget-object v0, System.out
CODE:0003A318 const-string v1, aaaa # "aaaa"
CODE:0003A31C invoke-virtual {v0, v1}, Cvoid PrintStream.println(ref) imp. 0_def_PrintStream_printlnBUL
CODE:0003A322 locret:
CODE:0003A322 return-void
CODE:0003A322 Method End
```

切换到 HexView-a 视图，记录下指令码 “ 62 00 02 04 1A 01 8E 07 6E 20 19 10 10 00 ” 和对
应偏移 “0003A2A4 ”



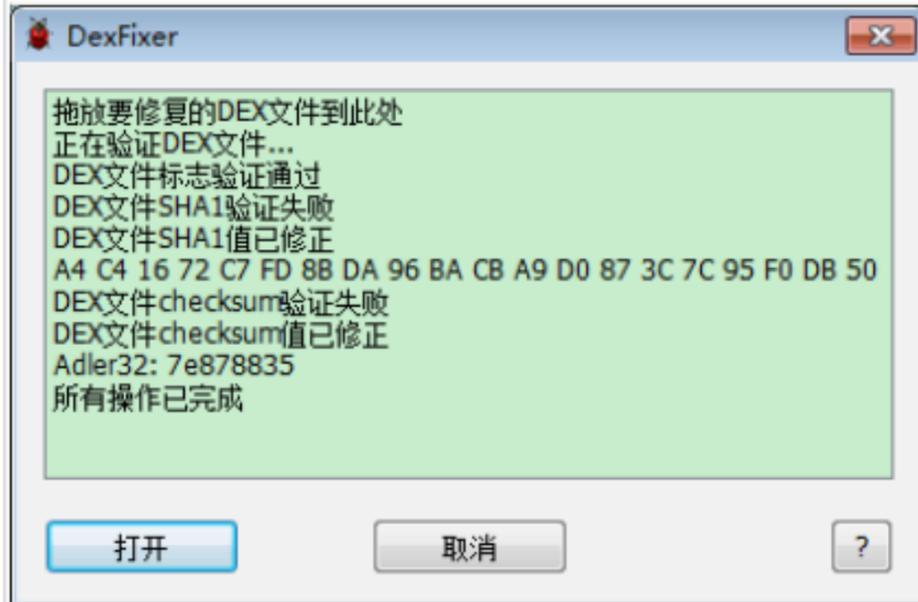
使用 C32asm，以十六进制的方式打开 dex 文件。按快捷键 “ Ctrl + G ”，定位到 “ 0003A2A4 ”
把 “ 62 00 02 04 1A 01 8E 07 6E 20 19 10 10 00 ” 改为 “ 12 01 38 01 03 00 FF FF 00 00 00 00 00 00 ”



Opcodes 解释：

12 01 // const/4 v1, 0 //v1=0
38 01 03 00 // if-eqz v1, loc_3A2AC //if(v1==0) 跳转到 loc_3A2AC:
FF FF // FFFF (Bad opcodes) // 本行代码被跳过永远不会执行
// loc_3A2AC:

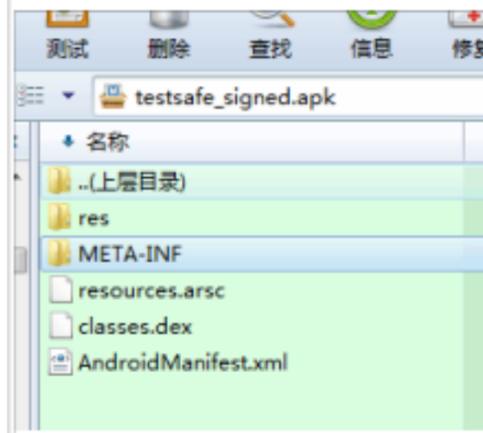
保存 dex。把修改后的 dex 文件拖入 DexFixer 进行修复。



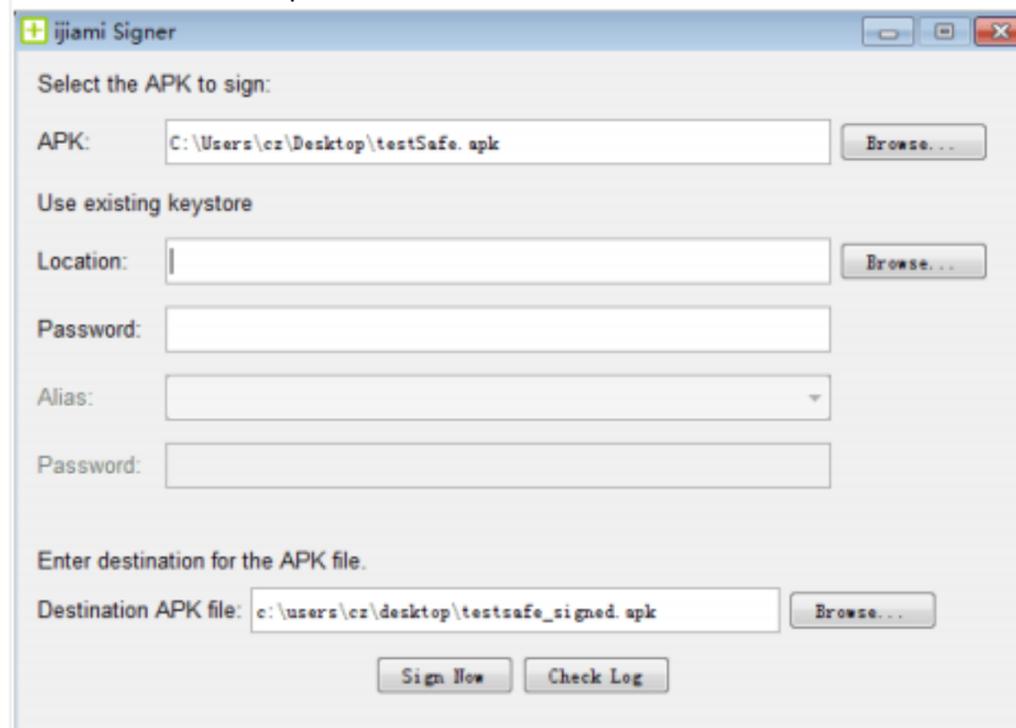
用修复后的 dex 覆盖原 apk 中的 dex 文件。



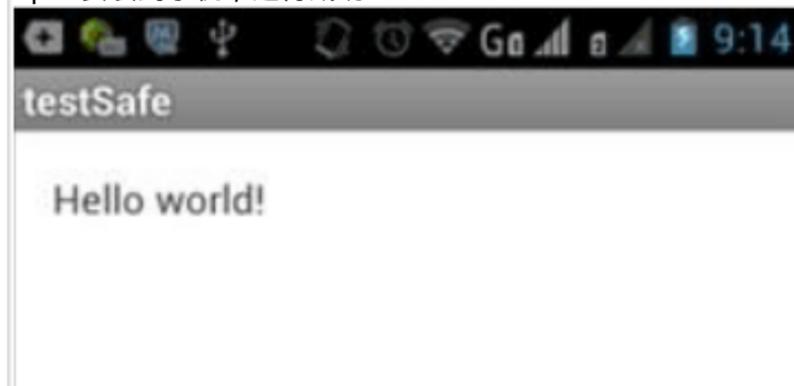
删除 META-INF 签名文件



使用签名工具，对 apk 重新签名。

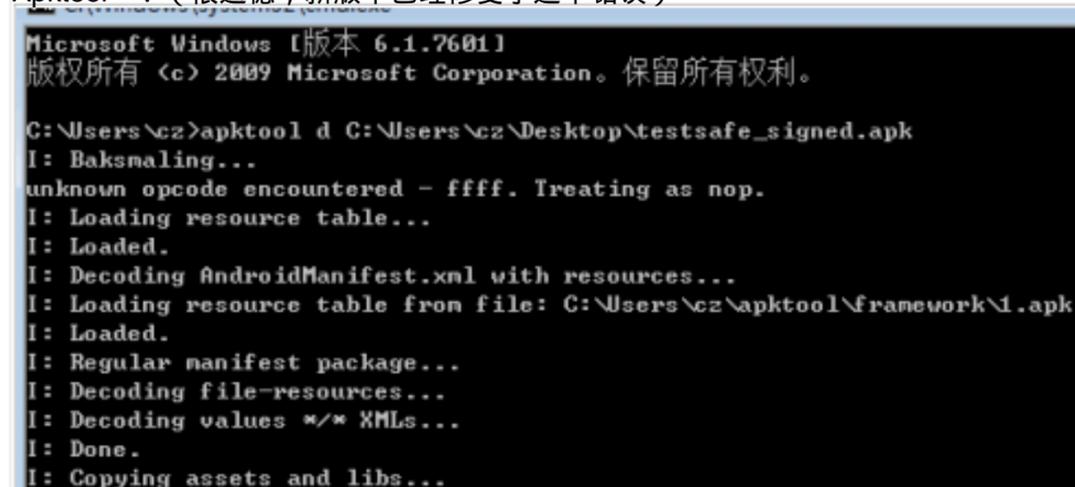


Apk 安装到手机，运行成功



下面试试反逆向工具的效果

Apktool : (很遗憾，新版本已经修复了这个错误)



Dex2jar : (反编译失败)

```
C:\Users\cz>dex2jar C:\Users\cz\Desktop\testsafe_signed.apk
this cmd is deprecated, use the d2j-dex2jar if possible
dex2jar version: translator-0.0.9.15
dex2jar C:\Users\cz\Desktop\testsafe_signed.apk -> C:\Users\cz\Desktop\testsafe_signed_dex2jar.jar
con.googlecode.dex2jar.DexException: while accept method:[Lcom/example/testsafe/BadCode;.aaa()U]
    at con.googlecode.dex2jar.reader.DexFileReader.acceptMethod(DexFileReader.java:694)
    at con.googlecode.dex2jar.reader.DexFileReader.acceptClass(DexFileReader.java:436)
    at con.googlecode.dex2jar.reader.DexFileReader.accept(DexFileReader.java:323)
    at con.googlecode.dex2jar.v3.Dex2jar.doTranslate(Dex2jar.java:85)
    at con.googlecode.dex2jar.v3.Dex2jar.to(Dex2jar.java:261)
    at con.googlecode.dex2jar.v3.Dex2jar.to(Dex2jar.java:252)
    at con.googlecode.dex2jar.v3.Main.doData(Main.java:43)
    at con.googlecode.dex2jar.v3.Main.doData(Main.java:35)
    at con.googlecode.dex2jar.v3.Main.doFile(Main.java:63)
    at con.googlecode.dex2jar.v3.Main.main(Main.java:86)
Caused by: con.googlecode.dex2jar.DexException: while accept code in method:[Lcom/example/testsafe/BadCode;.aaa()U]
    at con.googlecode.dex2jar.reader.DexFileReader.acceptMethod(DexFileReader.java:684)
    ... 9 more
Caused by: java.lang.RuntimeException: opcode format for 65535 not found!
    at con.googlecode.dex2jar.reader.OpcodFormat.get(OpcodFormat.java:362)
    at con.googlecode.dex2jar.reader.DexCodeReader.findLabels(DexCodeReader.java:88)
    at con.googlecode.dex2jar.reader.DexCodeReader.accept(DexCodeReader.java:332)
    at con.googlecode.dex2jar.reader.DexFileReader.acceptMethod(DexFileReader.java:682)
    ... 9 more
Done.
```