1. **Apk dex 加固**

   目前存在对 apk 中的 class.dex 进行加密的技术，即加壳。通过对 dex 文件的加壳，可以达到减小文件大小，隐藏真实代码的效果。

2. **运行时解密**

   与 windows 上的 shell 一样，Android 加壳程序在程序运行时，先到达壳的入口点，运行解壳代码，然后再到达程序入口点，运行代码。

   而要脱壳则要在程序解码完毕后，到达程序真实入口点中间某个位置，把原始的 dex 代码给 dump 出来，还原到 apk 文件中。

3. **一个简单 demo**

   入口点改变：

   壳入口：

   ```
   <application android:name="com.ali.mobisecenhance.StubApplication" />
   ```

   程序入口：

   ```
   <activity android:name="com.ali.encryption.MainActivity"/>
   ```

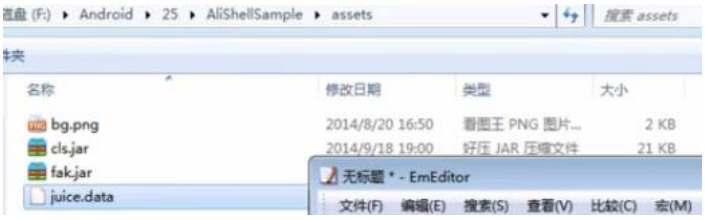
   壳代码中 Java 层转 jni 层

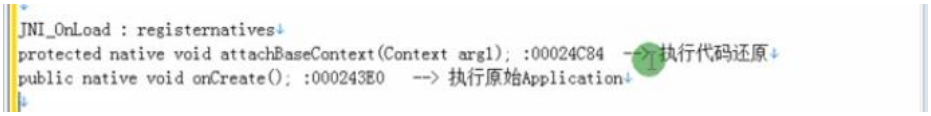
   ```
   protected native void attachBaseContext(Context arg1) {
   }

   public native void onCreate() {
   }
   ```

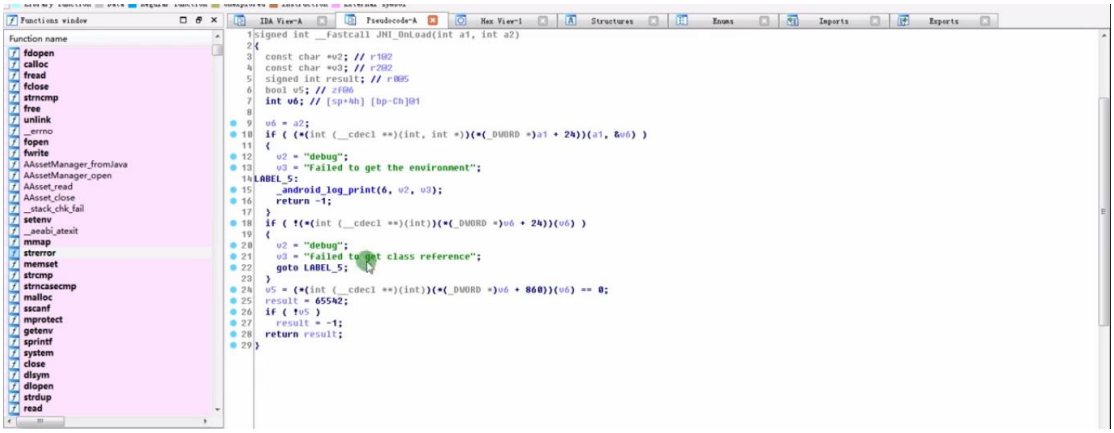
   jni 层中解密 dex 数据，还原并替换为原始 Application 数据，内存中获取到原始数据后，即可直接放到原始的 apk 文件中。

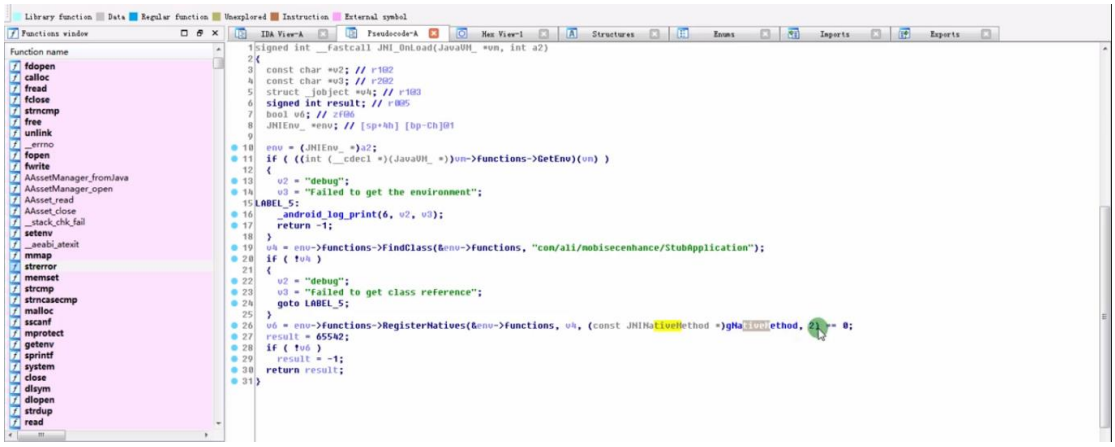
   所关注的加密文件是 cls.jar 和 juice.data



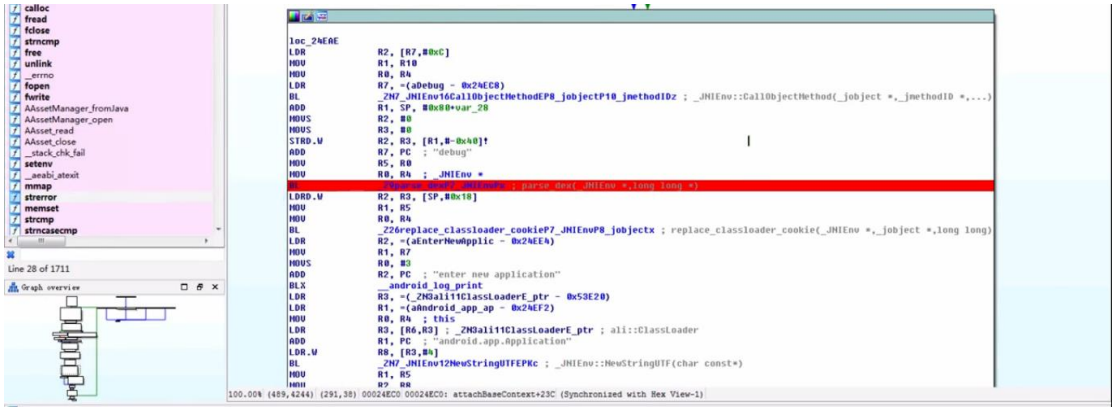

JNI_OnLoad 函数（注册 native 函数）还原：

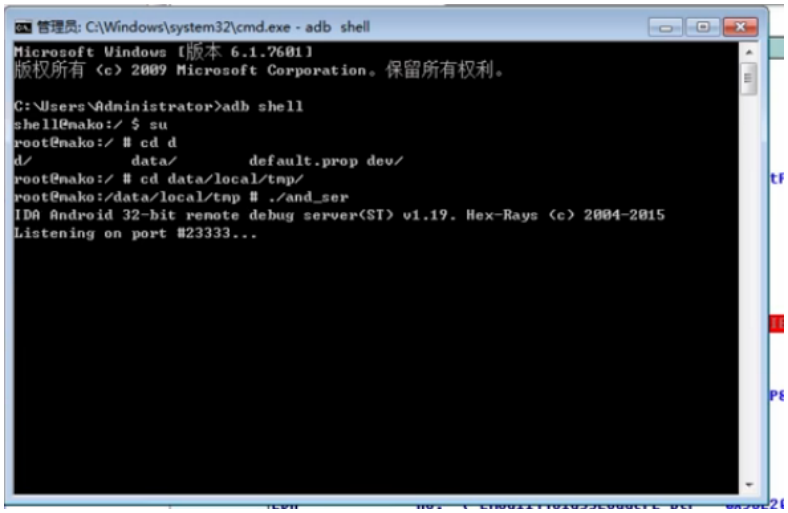


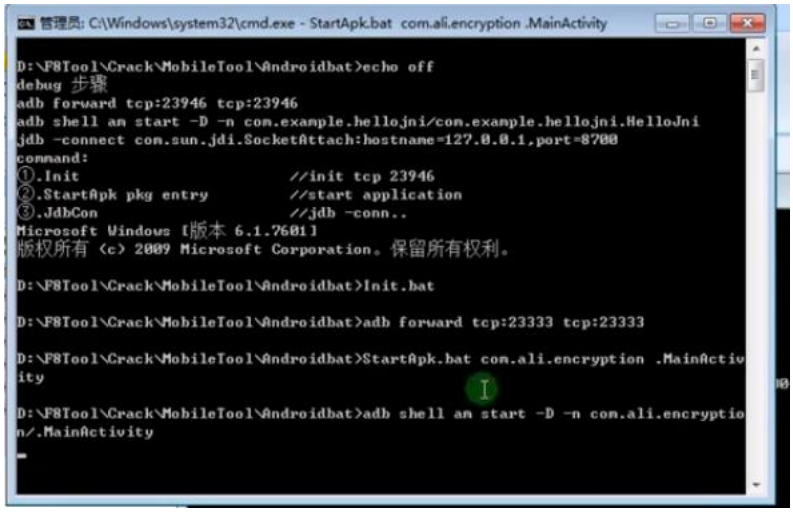


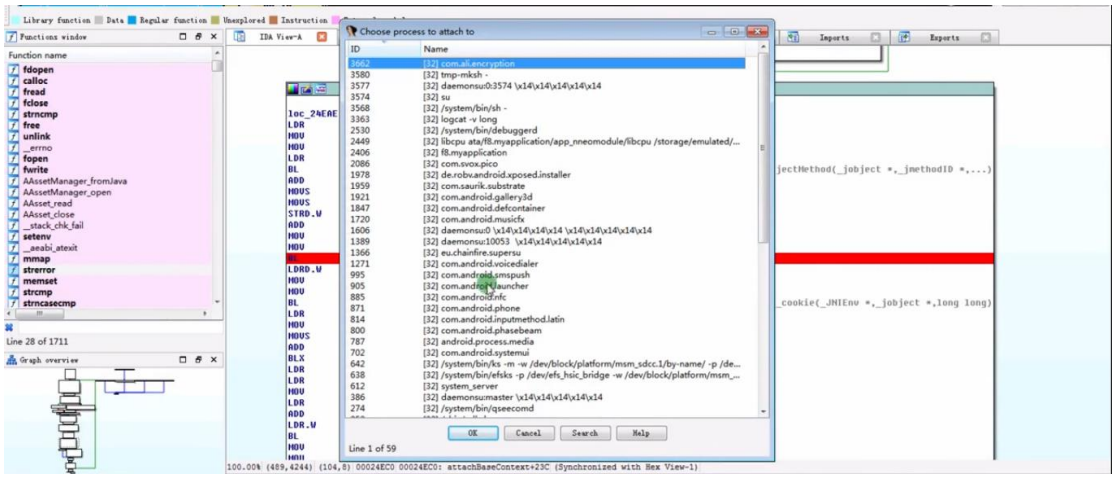
parse_dex 函数完成对 dex 文件的解密和还原

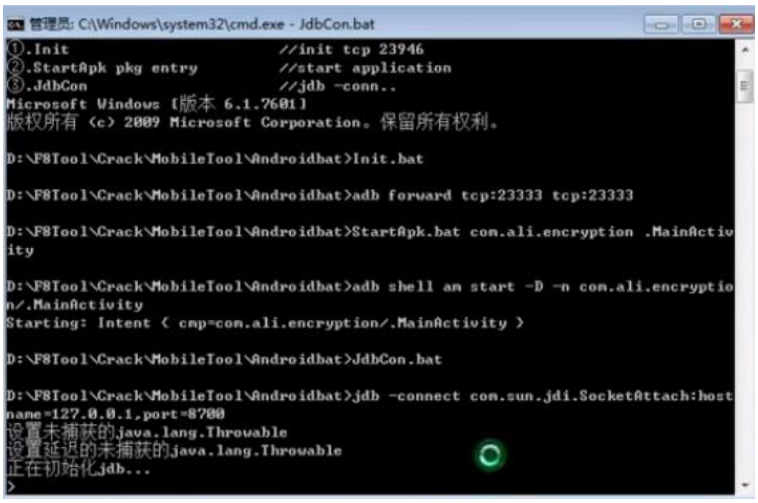
对 parse_dex 进行调试

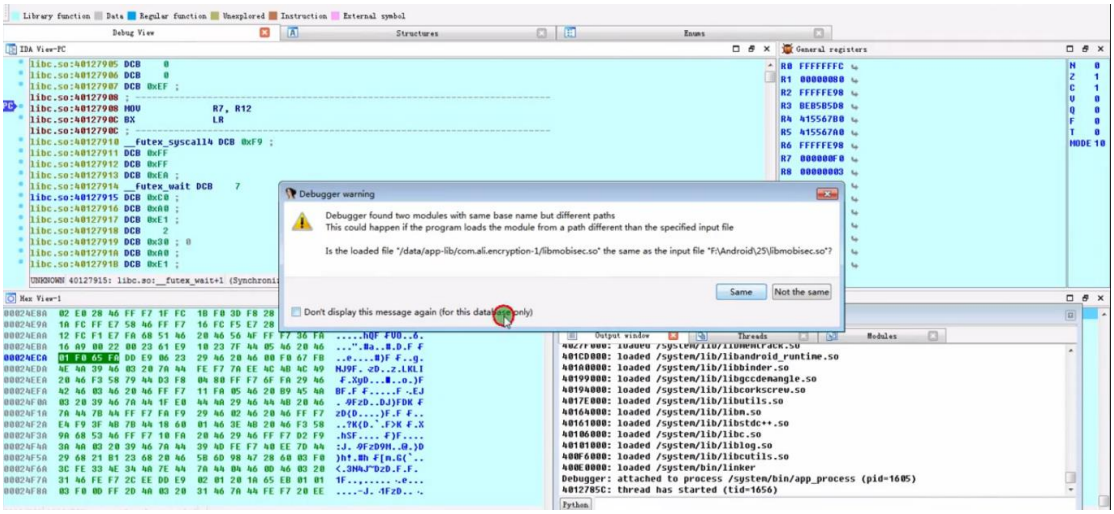



管理员: C:\Windows\system32\cmd.exe - adb shell
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。
C:\Users\Administrator>adb shell
shell@nako:/ $ su
root@nako:/ # cd d
d/ data/ default.prop dev/
root@nako:/ # cd data/local/tmp/
root@nako:/data/local/tmp # ./and_ser
IDA Android 32-bit remote debug server(ST) v1.19. Hex-Rays (c) 2004-2015
Listening on port #23333...

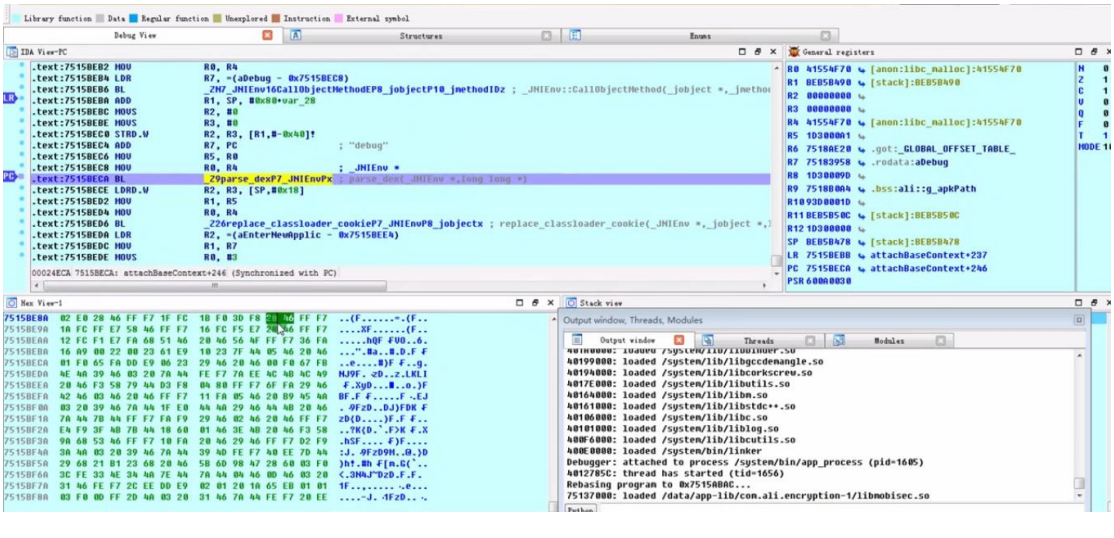

管理员: C:\Windows\system32\cmd.exe - StartApk.bat com.ali.encryption .MainActivity
D:\F8Tool\Crack\MobileTool\Androidbat>echo off
debug 步骤
adb forward tcp:23946 tcp:23946
adb shell am start -D -n com.example.hellojni/com.example.hellojni.HelloJni
jdb -connect com.sun.jdi.SocketAttach:hostname=127.0.0.1,port=8700
command:
①.Init //init tcp 23946
②.StartApk pkg entry //start application
③.JdbCon //jdb -conn..
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。
D:\F8Tool\Crack\MobileTool\Androidbat>Init.bat
D:\F8Tool\Crack\MobileTool\Androidbat>adb forward tcp:23333 tcp:23333
D:\F8Tool\Crack\MobileTool\Androidbat>StartApk.bat com.ali.encryption .MainActivity
D:\F8Tool\Crack\MobileTool\Androidbat>adb shell am start -D -n com.ali.encryption/.MainActivity

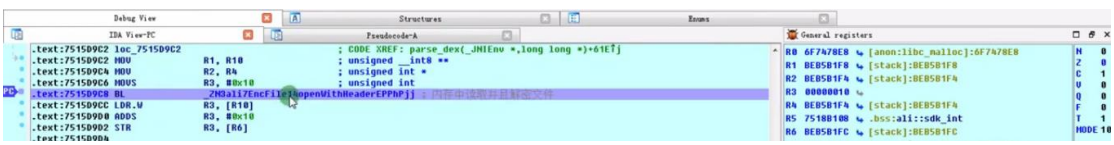

Choose process to attach to
ID
Name
3582 [32] com.ali.encryption
3580 [32] tmp-mksh -
3577 [32] daemonsu:0:3574 \x14\x14\x14\x14\x14
3574 [32] su
3568 [32] /system/bin/sh -
3363 [32] logcat -v long
2530 [32] /system/bin/debuggerd
2449 [32] libcpu_ata/f8.myapplication/app_nneomodule/libcpu /storage/emulated/...
2406 [32] f8.myapplication
2086 [32] com.svox.pico
1978 [32] de.robv.android.xposed.installer
1959 [32] com.saurik.substrate
1921 [32] com.android.gallery3d
1847 [32] com.android.defcontainer
1720 [32] com.android.musicfx
1606 [32] daemonsu:0 \x14\x14\x14\x14 \x14\x14\x14\x14\x14
1389 [32] daemonsu:10053 \x14\x14\x14\x14\x14
1366 [32] eu.chainfire.supersu
1271 [32] com.android.voicedialer
995 [32] com.android.smspush
905 [32] com.android.launcher
885 [32] com.android.nfc
871 [32] com.android.phone
814 [32] com.android.inputmethod.latin
800 [32] com.android.phasebeam
787 [32] android.process.media
702 [32] com.android.systemui
642 [32] /system/bin/ks -m -w /dev/block/platform/msm_sdcc.1/by-name/ -p /de...
638 [32] /system/bin/efsks -p /dev/efs_hsic_bridge -w /dev/block/platform/msm_...
612 [32] system_server
386 [32] daemonsu:master \x14\x14\x14\x14\x14
274 [32] /system/bin/qseecomd
OK
Cancel
Search
Help
Line 1 of 59



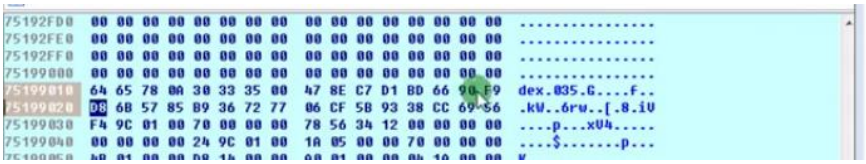




程序在断点处中断



动态调试得到原始 dex 文件被解密出来的存放地址

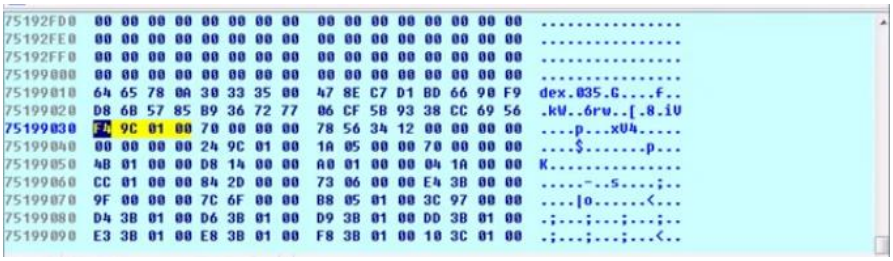



cls.jar 文件在解密后是原始的 class.dex 文件

Dump 脚本：IDC

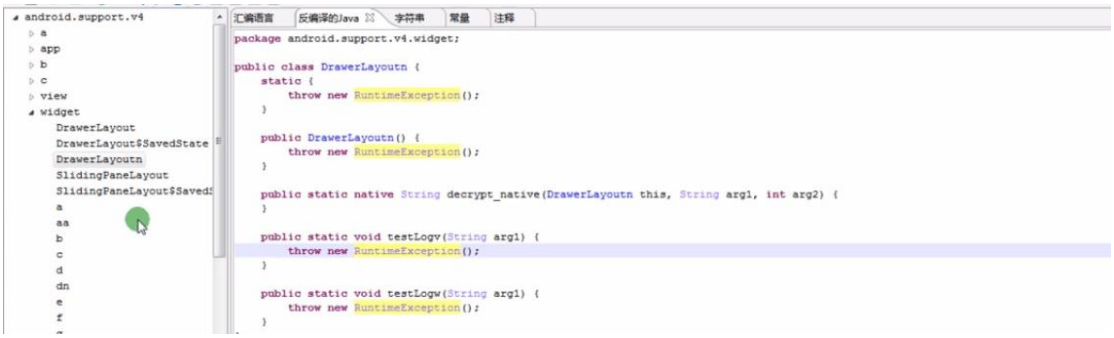
```
static main(void)
{
  auto fp, begin, end, dexbyte, len;
  fp = fopen("F:\\upkdump\\dump.dex","wb");
  begin = 0x544D2008;
  len = 0x019cf4;
  end = begin + len;
  for ( dexbyte = begin; dexbyte < end; dexbyte ++ )
      fputc(Byte(dexbyte), fp);
}
```

dex 文件在内存中的大小为




```
DexSize: F4 9C 01 00 0x00019cF4
```

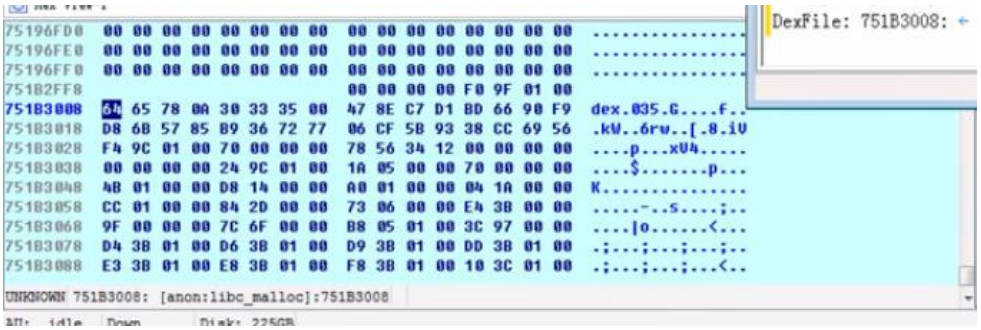
但是 dump 出来的 dex 文件打开后没有函数的真实代码



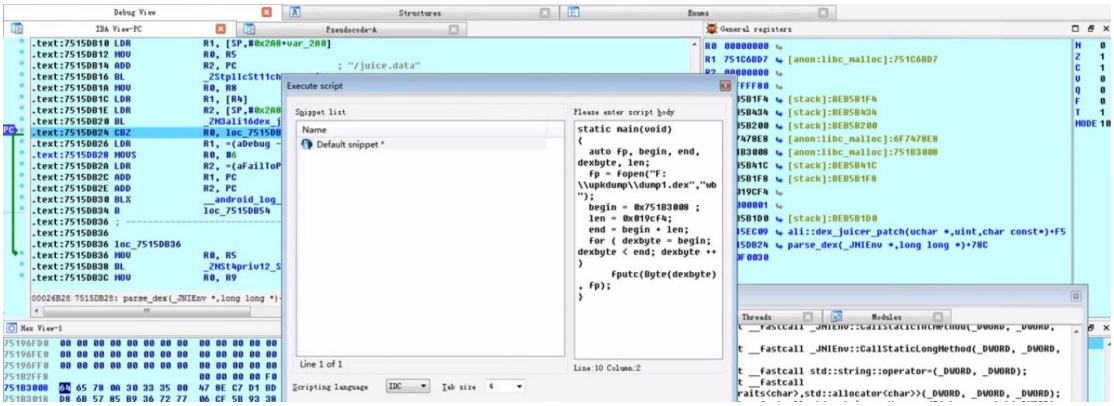

原理是程序执行之后再会对函数的真实代码进行还原

openDexFile：将内存中的 dex 文件转换为 DexOrJar 结构体

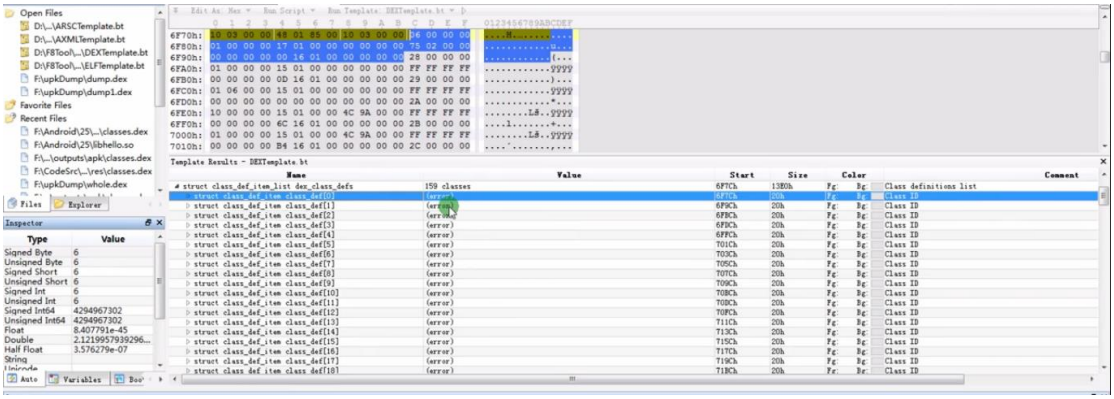
Dex 文件的新地址：



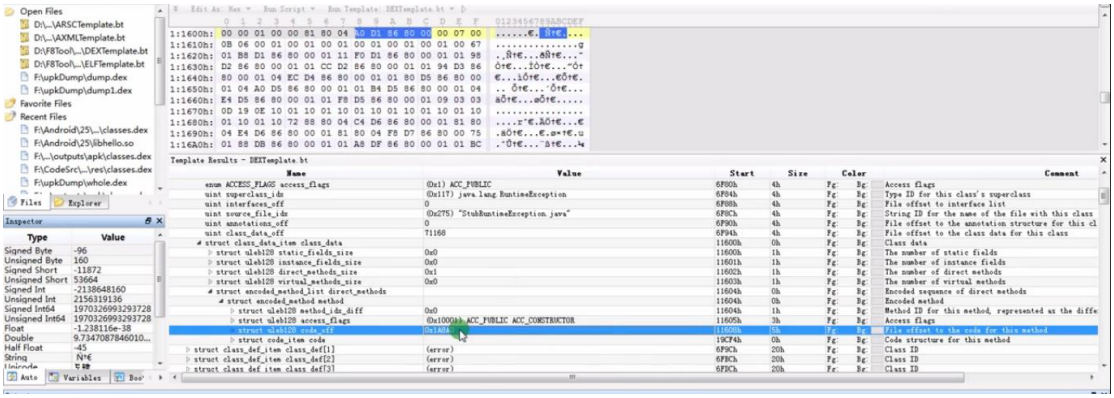

然后重新 dump 出文件



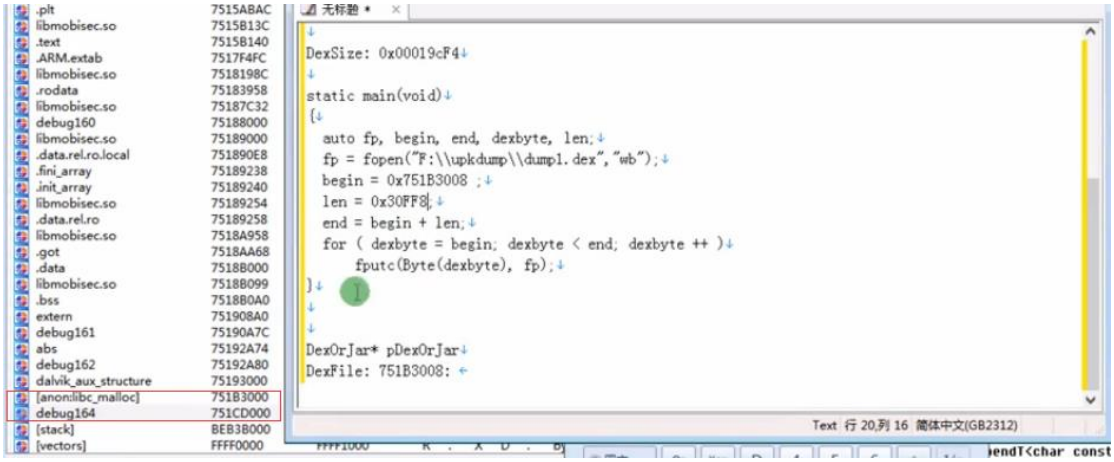
不过新 dump 出来的文件存在解析错误

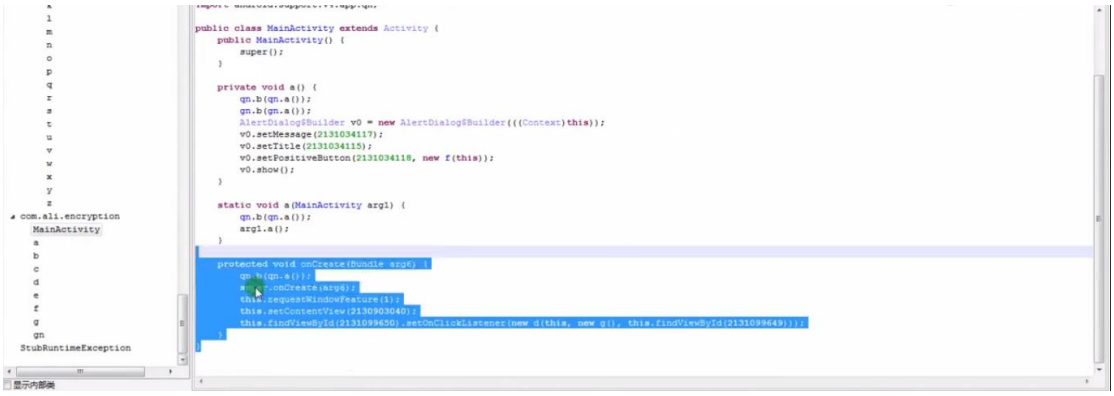


错误原因是 dump 出的 dex 文件不包括完整数据，需要将遗漏的数据补充进来



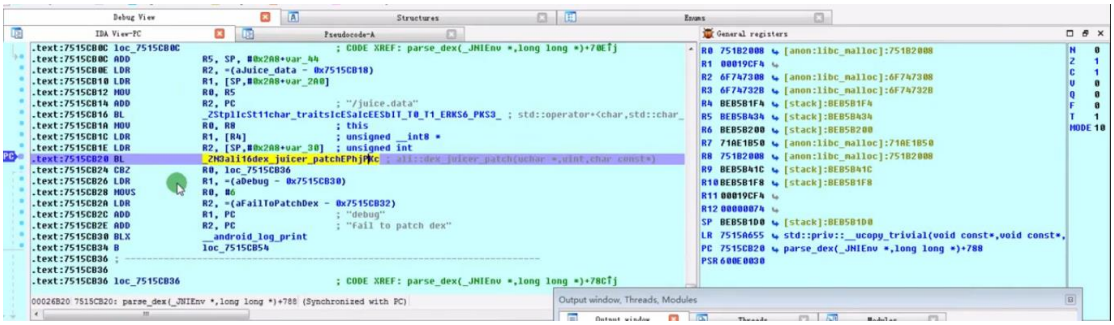
然后计算出完整文件的大小重新 dump

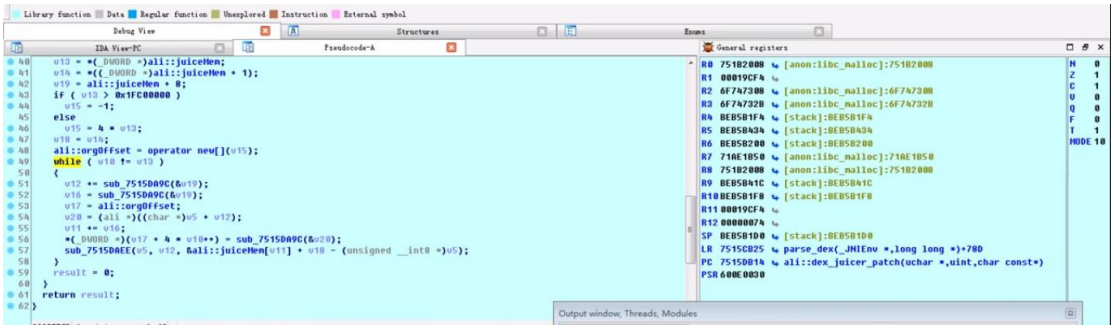



最后看到 dump 出的 dex 文件能够正常打开，脱壳步骤已经完成了

重新分析下对内存数据进行修正的函数




该循环中对 dex 数据进行修正

**将 dump 出来的数据反编译为 smali 代码，然后将该 smali 代码替换掉程序中的原有代码，然后删掉壳的入口点，重新编译之后该 app 成功脱壳。**