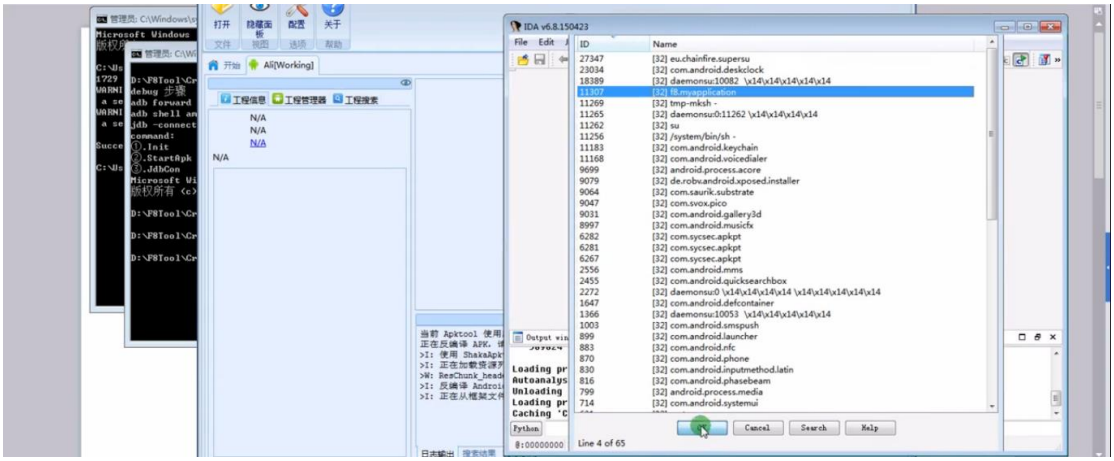
1. **Odex 转 dex**

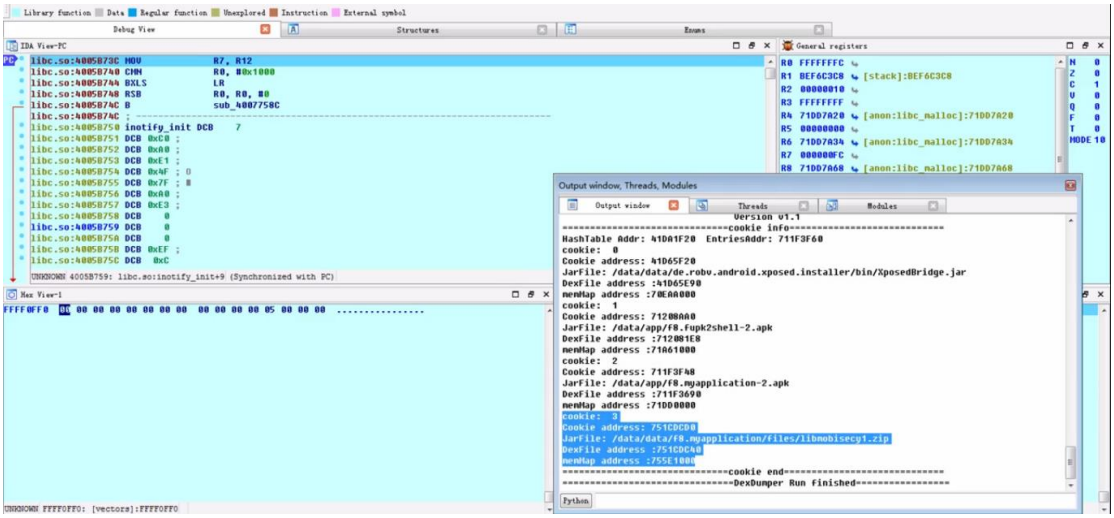
   参考 Android 源代码，完成逆转换

2. **Demo 演示（阿里壳）**

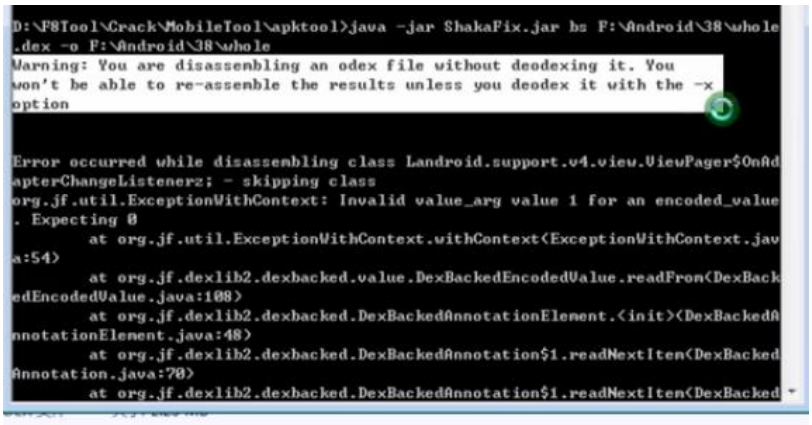
   安装、开启调试端口、端口转发、反编译、挂接

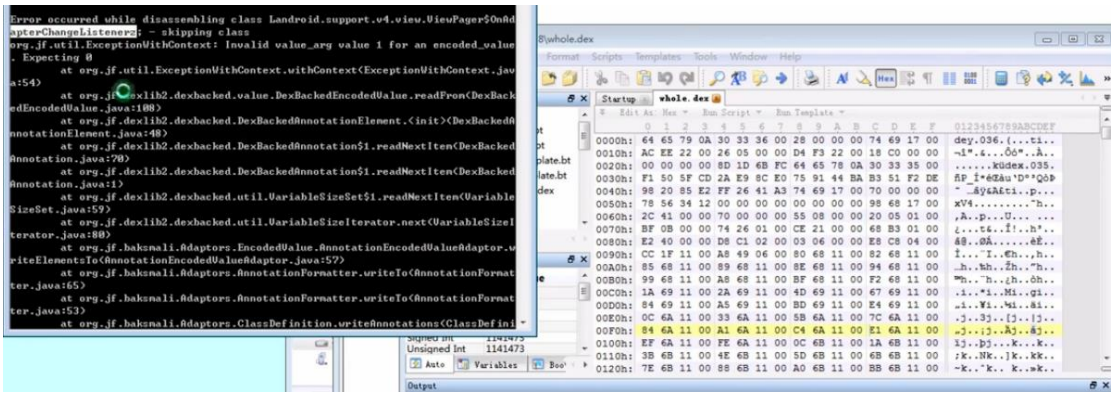


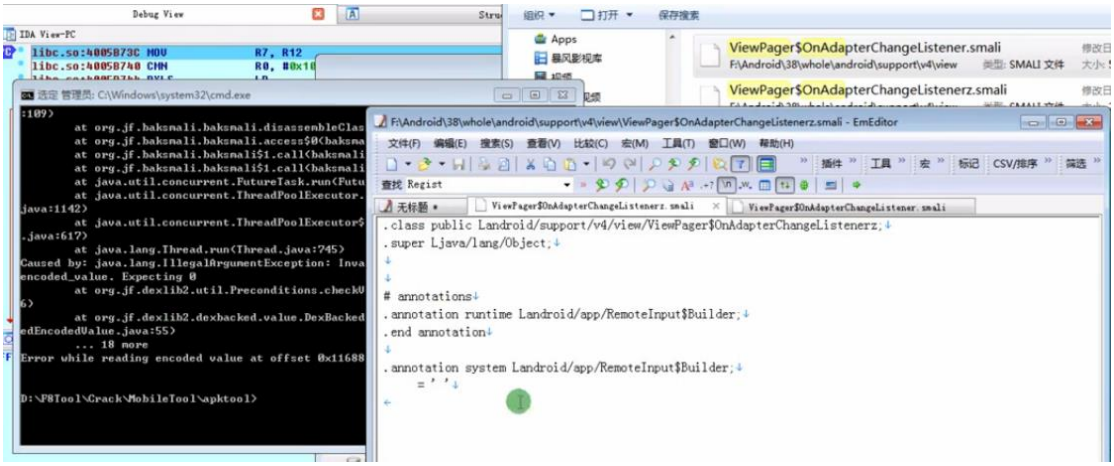
运行脚本，识别出了 4 个 dex 数据，前两个为自行输入的，第三个是壳的入口点，最后一个是内存中解码获取的数据



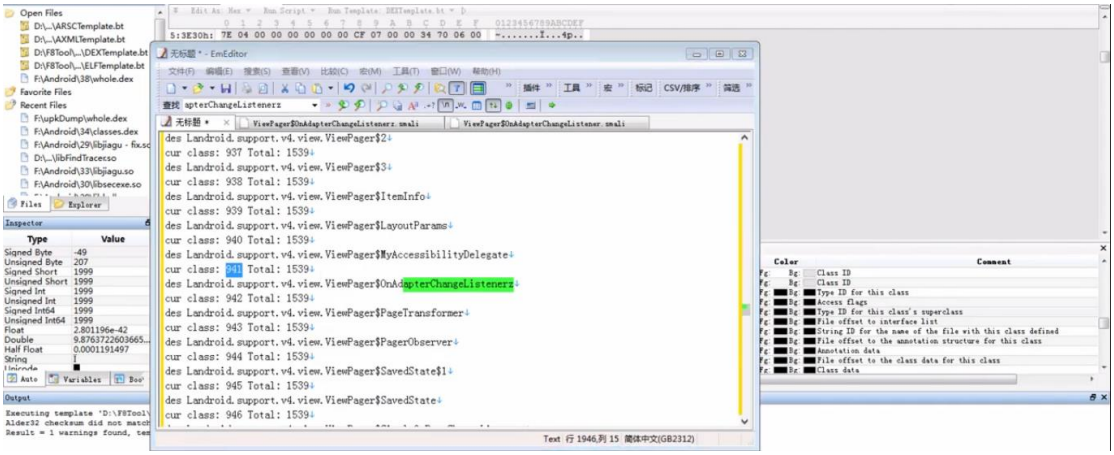
将最后的一组数据 dump 出来，然后将 dump 出来的数据进行反编译

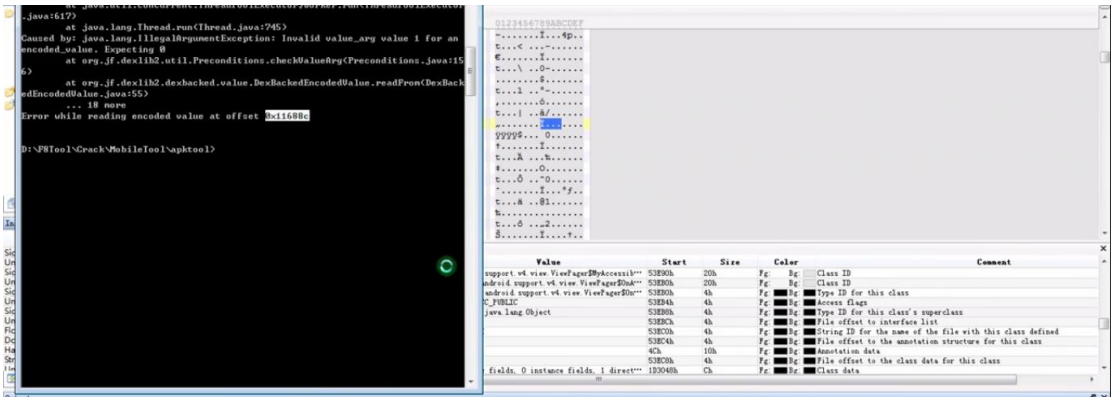




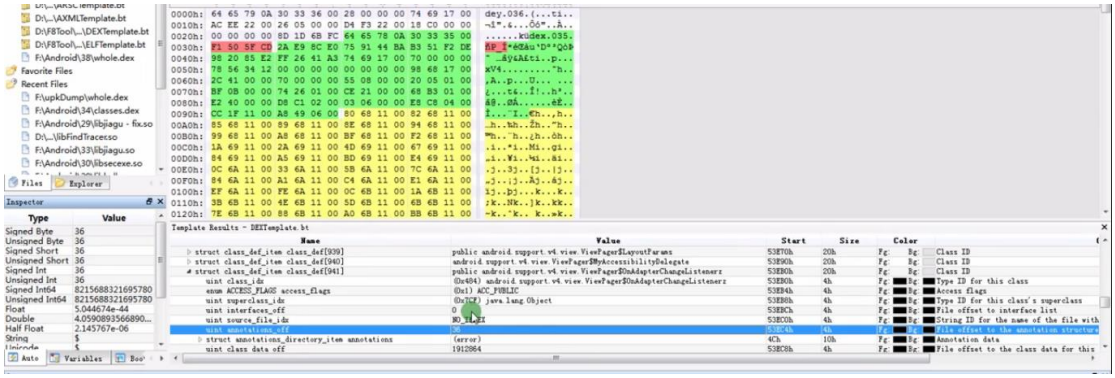


反编译因为阿里壳生成的一个类而出现异常，需要进行手动修复

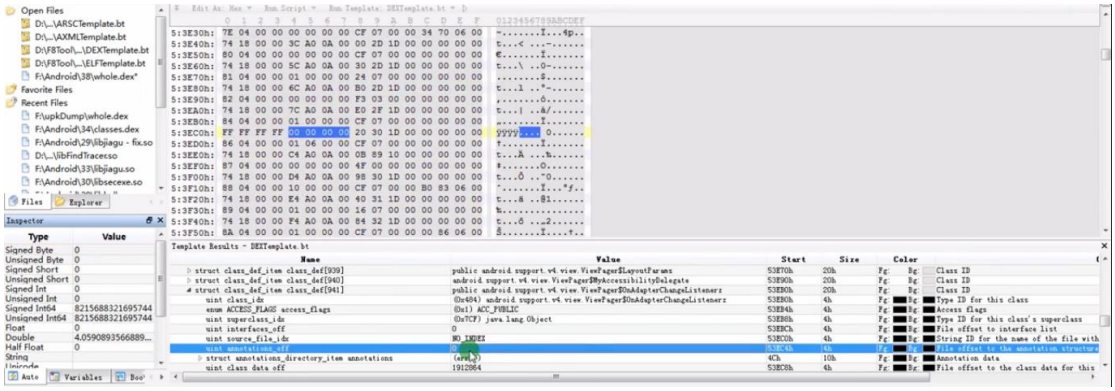




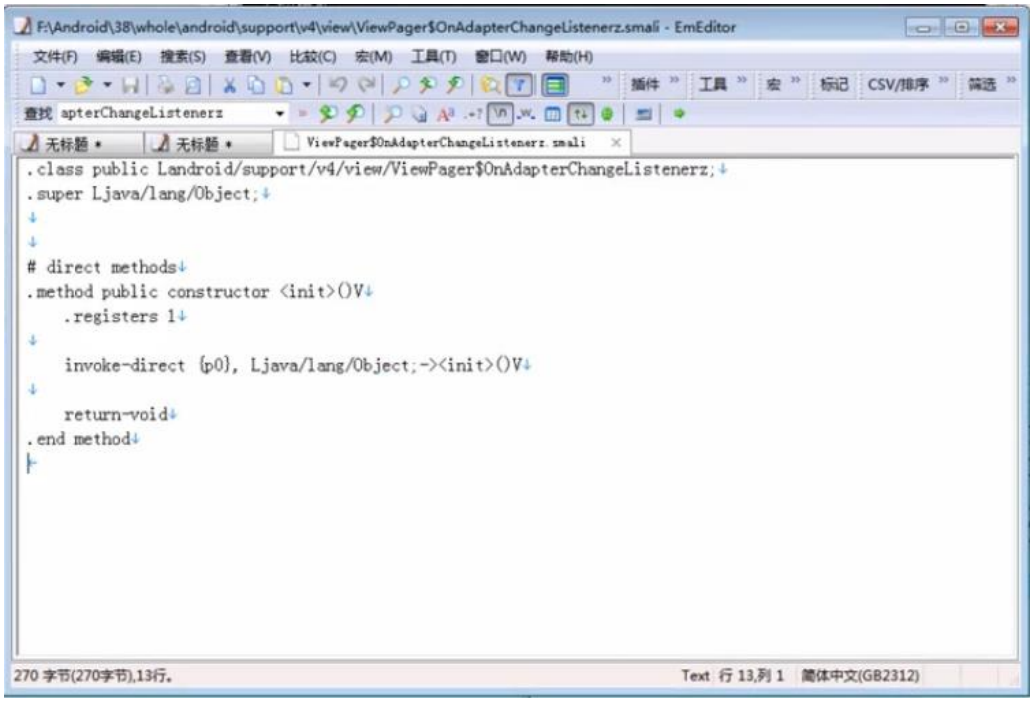
这个错误是由于解析位于该地址的内存数据时出现异常，这部分数据不正确



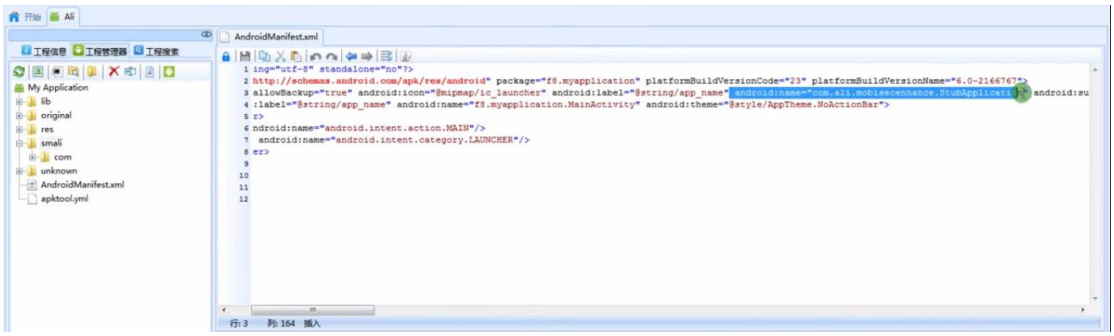
发现了阿里填充的非法数据，将其改为 0

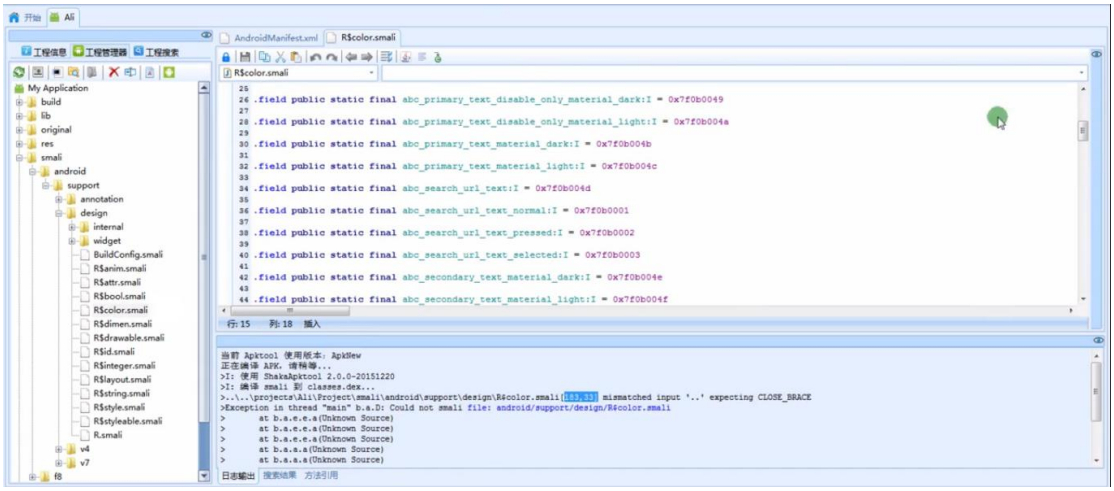


然后重新进行反编译，反编译成功，打开之前报错的类，里面的函数能够正常识别

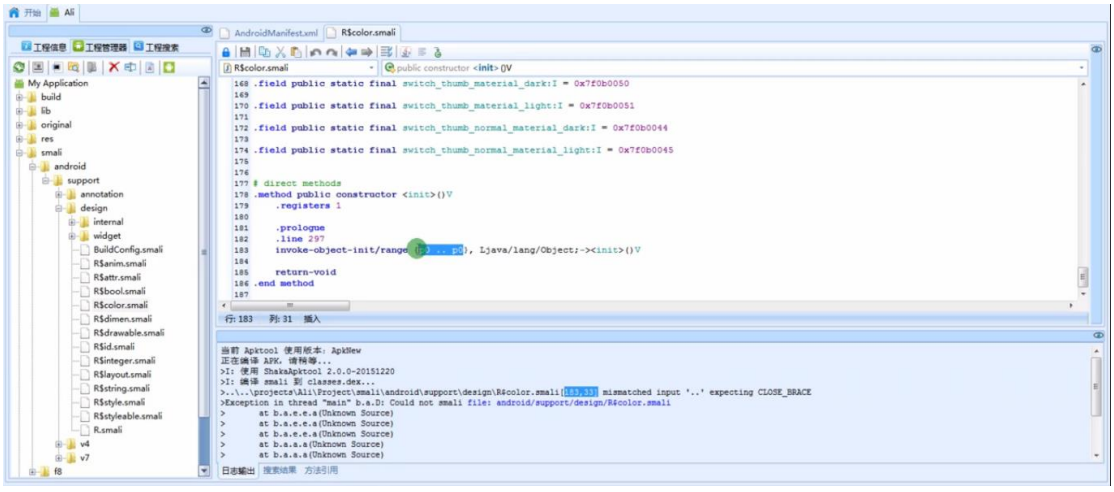


将反编译后的 smali 代码替换原代码，删掉 android:name 后重新编译



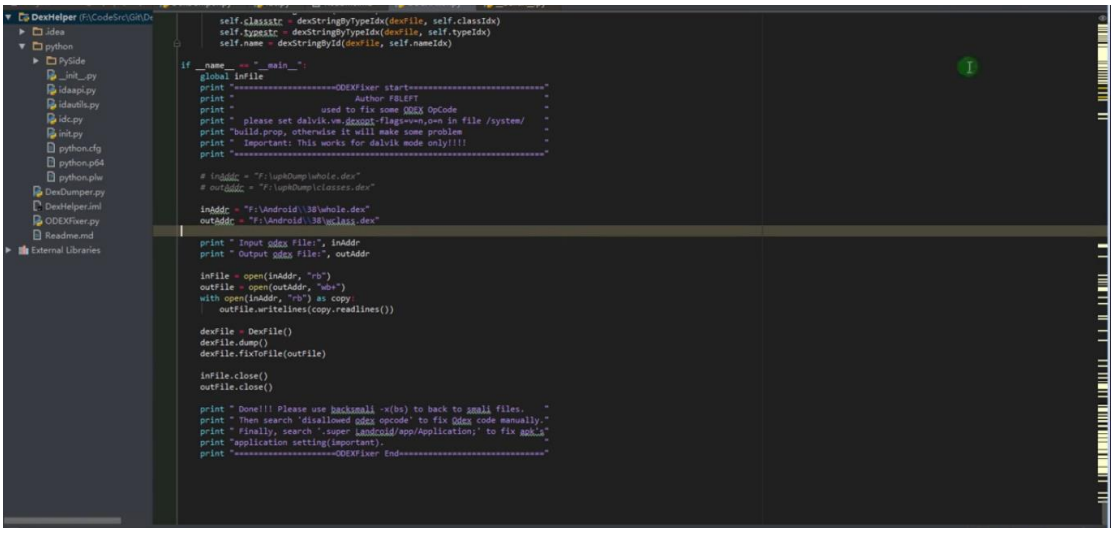


编译出错，根据错误提示定位到出错代码，发现错误是由于优化后的代码不能够正常编译导致的。

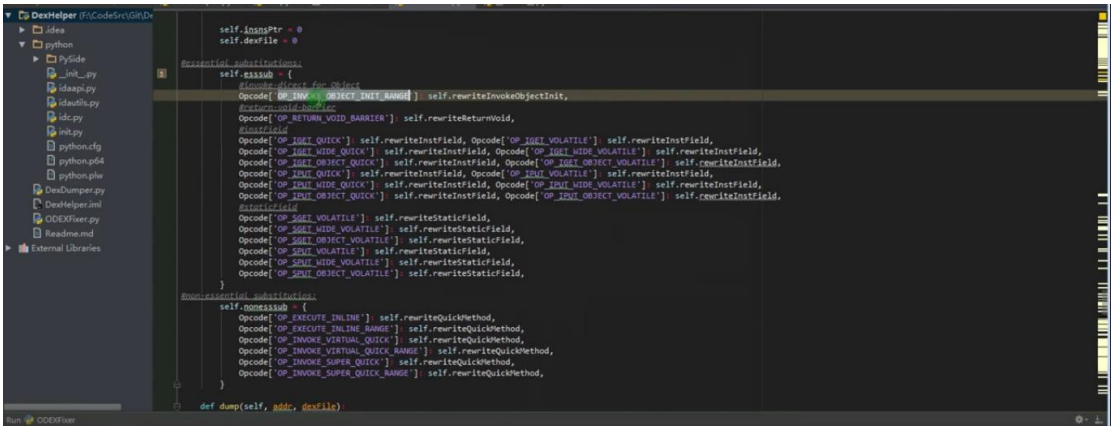


需要对优化后的 Odex 进行修复

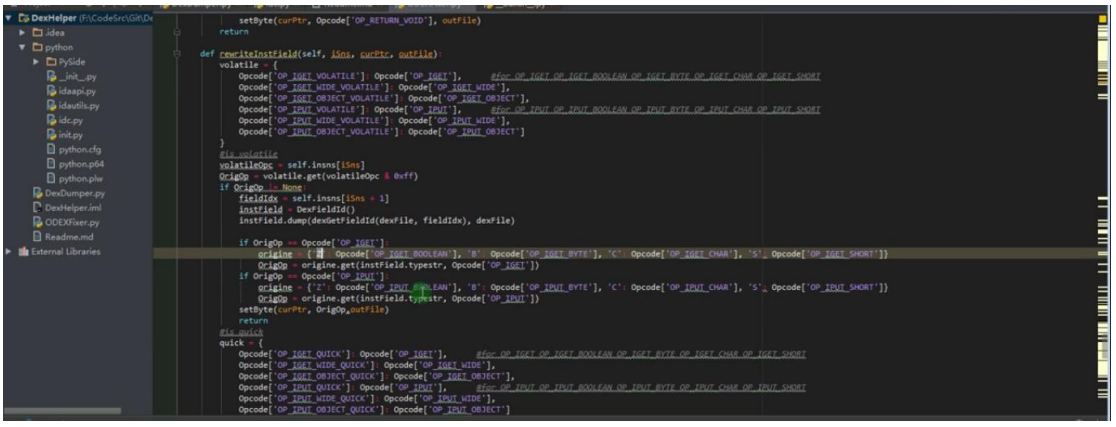
Odex 修复脚本



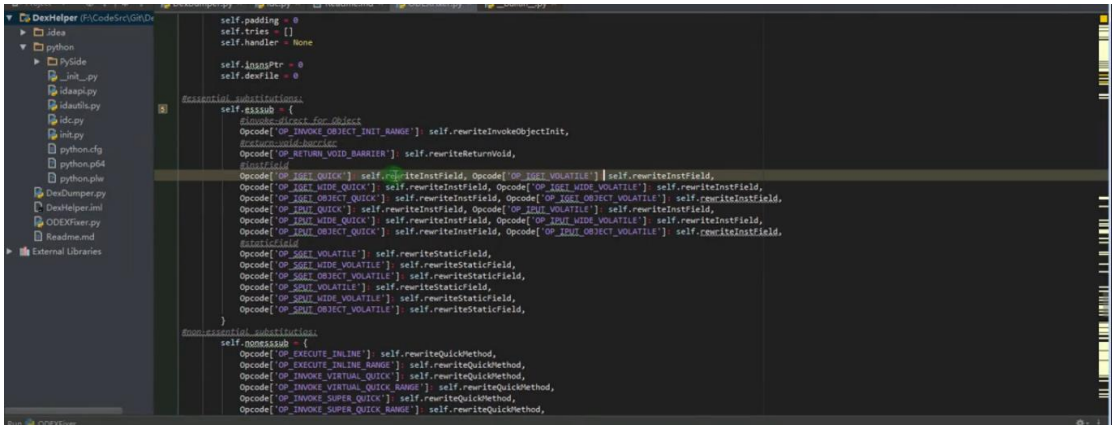
```
        self.classstr = dexStringByTypeIdx(dexFile, self.classIdx)
        self.typestr = dexStringByTypeIdx(dexFile, self.typeIdx)
        self.name = dexStringById(dexFile, self.nameIdx)

if __name__ == "__main__":
    global inFile
    print "====================ODEXFixer start=========================="
    print "                          Author FBLEFT                      "
    print "                  used to fix some ODEX OpCode               "
    print "  please set dalvik.vm.dexopt-flags=v=n,o=n in file /system/  "
    print "build.prop, otherwise it will make some problem              "
    print "  Important: This works for dalvik mode only!!!!             "
    print "==============================================================="

    # inAddr = "F:\upkDump\whole.dex"
    # outAddr = "F:\upkDump\classes.dex"

    inAddr = "F:\Android\\38\whole.dex"
    outAddr = "F:\Android\\38\wclass.dex"

    print " Input odex File:", inAddr
    print " Output odex File:", outAddr

    inFile = open(inAddr, "rb")
    outFile = open(outAddr, "wb+")
    with open(inAddr, "rb") as copy:
        outFile.writelines(copy.readlines())

    dexFile = DexFile()
    dexFile.dump()
    dexFile.fixToFile(outFile)

    inFile.close()
    outFile.close()

    print " Done!!! Please use backsmali -x(bs) to back to smali files.    "
    print " Then search 'disallowed odex opcode' to fix odex code manually."
    print " Finally, search '.super Landroid/app/Application;' to fix apk's"
    print "application setting(important).                                "
    print "====================ODEXFixer End============================"
```

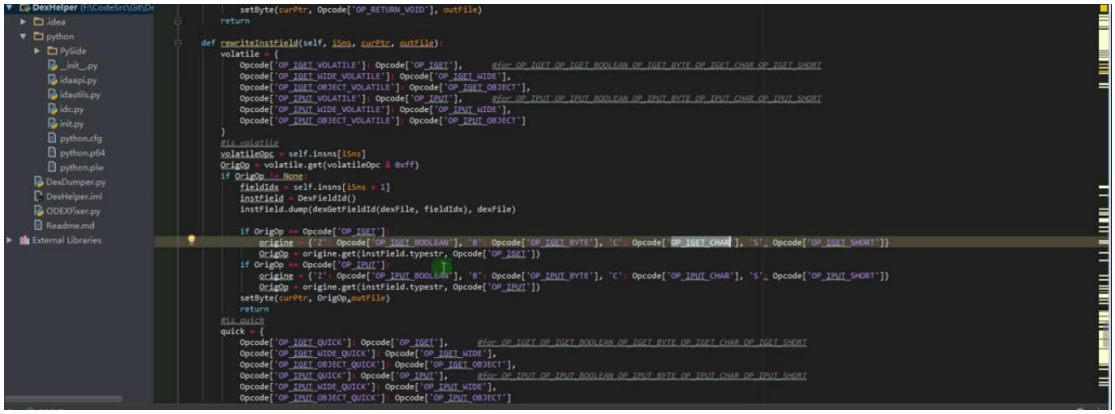


```
        self.insnsPtr = 0
        self.dexFile = 0

#essential substitutions:
        self.essSub = {
            #invoke-direct for Object
            Opcode['OP_INVOKE_OBJECT_INIT_RANGE']: self.rewriteInvokeObjectInit,
            #return-void-barrier
            Opcode['OP_RETURN_VOID_BARRIER']: self.rewriteReturnVoid,
            #instfield
            Opcode['OP_IGET_QUICK']: self.rewriteInstField, Opcode['OP_IGET_VOLATILE']: self.rewriteInstField,
            Opcode['OP_IGET_WIDE_QUICK']: self.rewriteInstField, Opcode['OP_IGET_WIDE_VOLATILE']: self.rewriteInstField,
            Opcode['OP_IGET_OBJECT_QUICK']: self.rewriteInstField, Opcode['OP_IGET_OBJECT_VOLATILE']: self.rewriteInstField,
            Opcode['OP_IPUT_QUICK']: self.rewriteInstField, Opcode['OP_IPUT_VOLATILE']: self.rewriteInstField,
            Opcode['OP_IPUT_WIDE_QUICK']: self.rewriteInstField, Opcode['OP_IPUT_WIDE_VOLATILE']: self.rewriteInstField,
            Opcode['OP_IPUT_OBJECT_QUICK']: self.rewriteInstField, Opcode['OP_IPUT_OBJECT_VOLATILE']: self.rewriteInstField,
            #staticfield
            Opcode['OP_SGET_VOLATILE']: self.rewriteStaticField,
            Opcode['OP_SGET_WIDE_VOLATILE']: self.rewriteStaticField,
            Opcode['OP_SGET_OBJECT_VOLATILE']: self.rewriteStaticField,
            Opcode['OP_SPUT_VOLATILE']: self.rewriteStaticField,
            Opcode['OP_SPUT_WIDE_VOLATILE']: self.rewriteStaticField,
            Opcode['OP_SPUT_OBJECT_VOLATILE']: self.rewriteStaticField,
        }
#non-essential substitutions:
        self.nonessSub = {
            Opcode['OP_EXECUTE_INLINE']: self.rewriteQuickMethod,
            Opcode['OP_EXECUTE_INLINE_RANGE']: self.rewriteQuickMethod,
            Opcode['OP_INVOKE_VIRTUAL_QUICK']: self.rewriteQuickMethod,
            Opcode['OP_INVOKE_VIRTUAL_QUICK_RANGE']: self.rewriteQuickMethod,
            Opcode['OP_INVOKE_SUPER_QUICK']: self.rewriteQuickMethod,
            Opcode['OP_INVOKE_SUPER_QUICK_RANGE']: self.rewriteQuickMethod,
        }

    def dump(self, addr, dexFile):
```

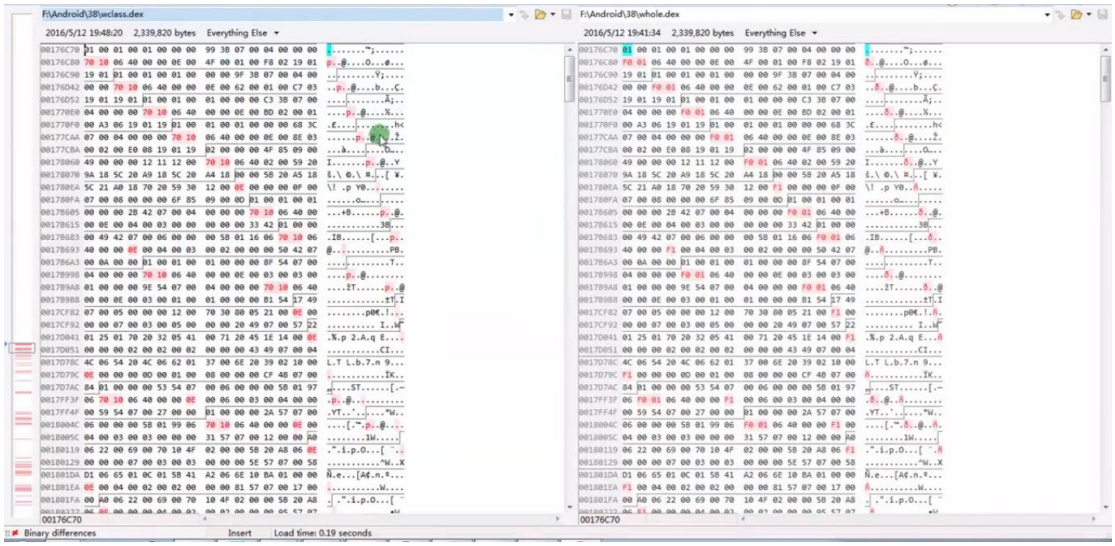


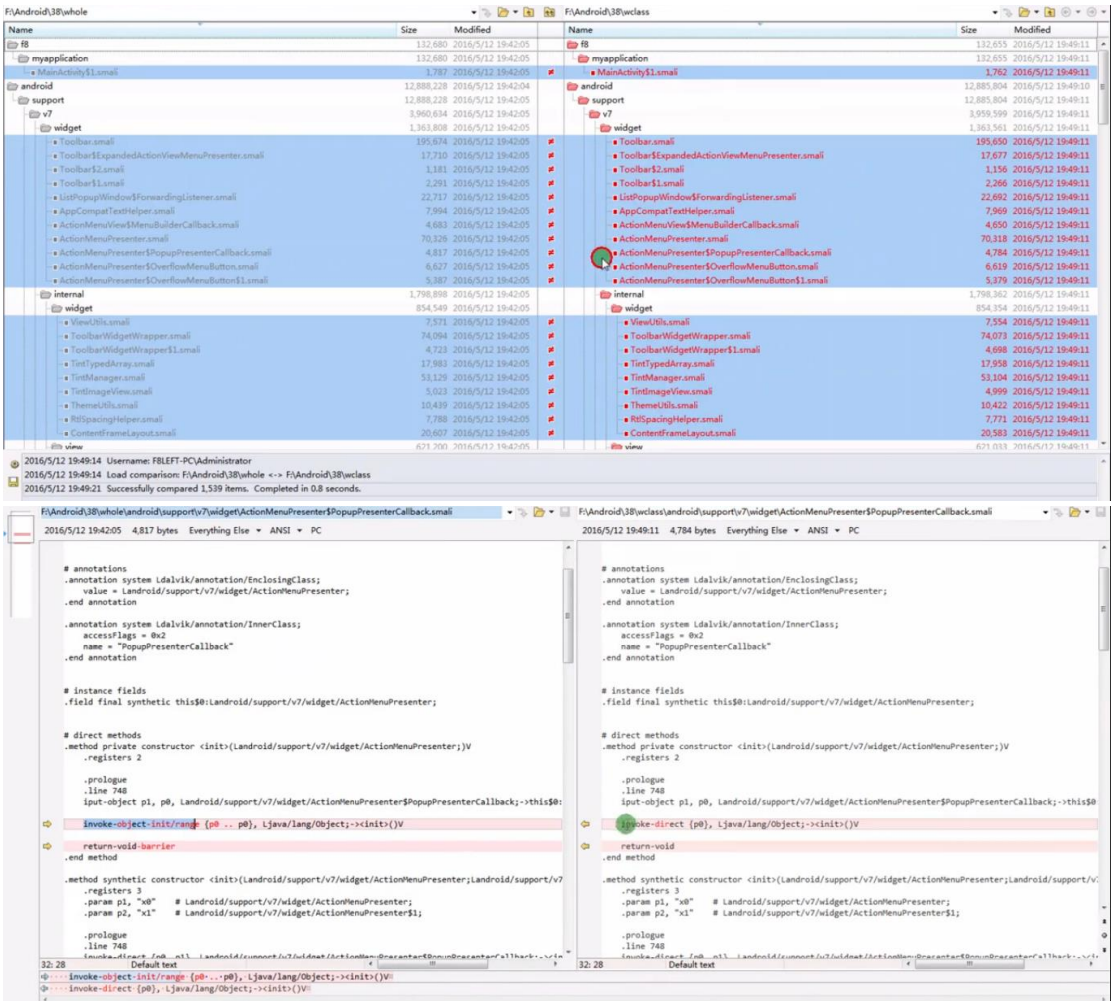
```
            setByte(curPtr, Opcode['OP_RETURN_VOID'], outFile)
        return

    def rewriteInstField(self, iSns, curPtr, outFile):
        volatile = {
            Opcode['OP_IGET_VOLATILE']: Opcode['OP_IGET'],      #for OP_IGET OP_IGET_BOOLEAN OP_IGET_BYTE OP_IGET_CHAR OP_IGET_SHORT
            Opcode['OP_IGET_WIDE_VOLATILE']: Opcode['OP_IGET_WIDE'],
            Opcode['OP_IGET_OBJECT_VOLATILE']: Opcode['OP_IGET_OBJECT'],
            Opcode['OP_IPUT_VOLATILE']: Opcode['OP_IPUT'],      #for OP_IPUT OP_IPUT_BOOLEAN OP_IPUT_BYTE OP_IPUT_CHAR OP_IPUT_SHORT
            Opcode['OP_IPUT_WIDE_VOLATILE']: Opcode['OP_IPUT_WIDE'],
            Opcode['OP_IPUT_OBJECT_VOLATILE']: Opcode['OP_IPUT_OBJECT']
        }
        #is volatile
        volatileOpc = self.insns[iSns]
        OrigOp = volatile.get(volatileOpc & 0xff)
        if OrigOp != None:
            fieldIdx = self.insns[iSns + 1]
            instField = DexFieldId()
            instField.dump(dexGetFieldId(dexFile, fieldIdx), dexFile)

            if OrigOp == Opcode['OP_IGET']:
                origine = {'Z': Opcode['OP_IGET_BOOLEAN'], 'B': Opcode['OP_IGET_BYTE'], 'C': Opcode['OP_IGET_CHAR'], 'S': Opcode['OP_IGET_SHORT']}
                OrigOp = origine.get(instField.typestr, Opcode['OP_IGET'])
            if OrigOp == Opcode['OP_IPUT']
                origine = {'Z': Opcode['OP_IPUT_BOOLEAN'], 'B': Opcode['OP_IPUT_BYTE'], 'C': Opcode['OP_IPUT_CHAR'], 'S': Opcode['OP_IPUT_SHORT']}
                OrigOp = origine.get(instField.typestr, Opcode['OP_IPUT'])
            setByte(curPtr, OrigOp,outFile)
            return
        #is quick
        quick = {
            Opcode['OP_IGET_QUICK']: Opcode['OP_IGET'],      #for OP_IGET OP_IGET_BOOLEAN OP_IGET_BYTE OP_IGET_CHAR OP_IGET_SHORT
            Opcode['OP_IGET_WIDE_QUICK']: Opcode['OP_IGET_WIDE'],
            Opcode['OP_IGET_OBJECT_QUICK']: Opcode['OP_IGET_OBJECT'],
            Opcode['OP_IPUT_QUICK']: Opcode['OP_IPUT'],      #for OP_IPUT OP_IPUT_BOOLEAN OP_IPUT_BYTE OP_IPUT_CHAR OP_IPUT_SHORT
            Opcode['OP_IPUT_WIDE_QUICK']: Opcode['OP_IPUT_WIDE'],
            Opcode['OP_IPUT_OBJECT_QUICK']: Opcode['OP_IPUT_OBJECT']
```





修复后的代码比较





对修复后的 smali 代码进行编译，编译成功，查看 smali 代码发现已经完成了修复