接触andorid逆向[脱壳](https://www.52pojie.cn/forum-5-1.html" \t "https://www.52pojie.cn/_blank)一段时间了，刚刚感觉开始入门，最开始时的过反调试，花费了大量的时间，但时间长了，汇编指令慢慢熟悉，学会了一些调  
试技巧，反调试方法知道了，就那么些反调试的方法，过反调就比较顺利了，所以刚开始学逆向的同学，应该也会比较头疼的，记录下我的逆向学习之  
旅，增强下记忆，记录分享也是自我提升的过程吧。轻松过掉反调试后，就能有更多的时间去研究壳本身流程和加解密操作。先前的dex加壳，不管是  
内存加载（内存dump），还是类抽取（dexHunter），从内存中dump出来后，还是比较容易重打包运行的，自从出现能解释执行smali指令的壳出现后  
，小伙伴都能把dex dump出来可是onCreate函数是native的，无法还原指令，dex的关键代码可以分析了，但是无法进行重打包运行，这tn就蛋疼了，  
没有卵用，这些天也是一有空就看，还有点成果，手里的两个apk，只是把启动相关Activity 的onCreate进行了还原，apk能运行起来，记录分享下我  
的思路。  
  
  
其实也没思路了，就是跟踪汇编，看汇编指令，函数比较复杂，而且加了混淆，刚开始头都大了。  
  
  
      1.apk 过反调试，360的反调试方法还是那么些，没有更新，可以看我前一篇文章。  
  
      2.找到正确的地方下断点，因为dump出dex的onCreate函数是native的，所以肯定是需要注册的，在so中找到给其赋值RegisterNative函数，相应的参  
数就是对应的jni函数，[IDA](https://www.52pojie.cn/thread-648089-1-1.html" \t "https://www.52pojie.cn/_blank)动态调试起来，下断点，迷糊的小伙伴可以翻看我前一篇文章，最后有介绍。  
  
  
      3.跟踪jni函数，关键在sub\_A990函数，而且函数挺长的，需要点耐心阅读

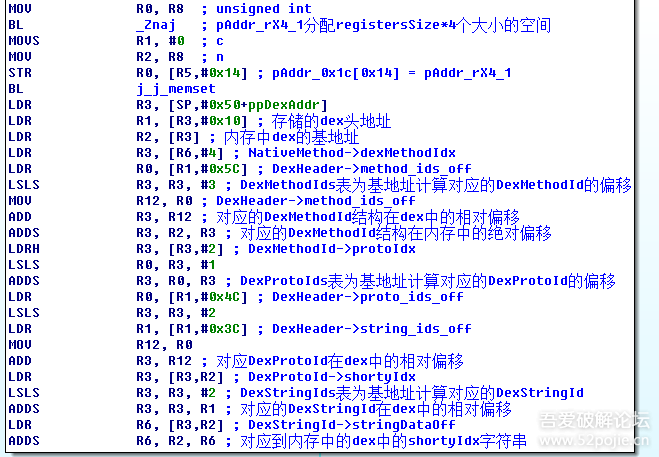
          

      4.根据方法的索引获取方法的结构体，大致结构体如下：

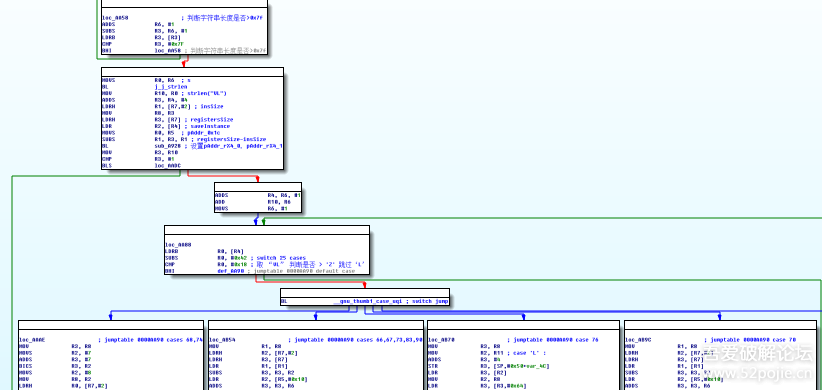
            Struct NativeMethod{  
                  int classIdx;  
                  int dexMethodIdx;  
                  int dexCodeOff;  
                  int\*\* pDexAddr;   
           }\*pNativeMethod;

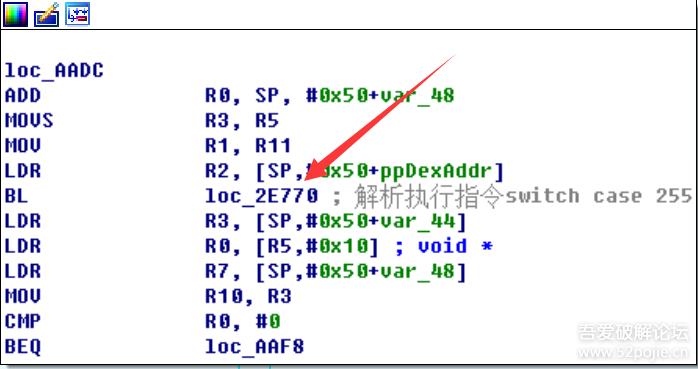
      5.根据结构体获取方法的相关信息

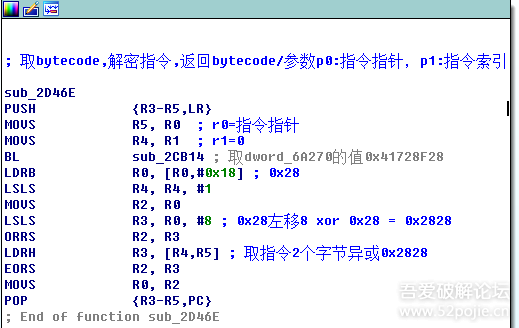
      6.根据方法的shorty，构建参数

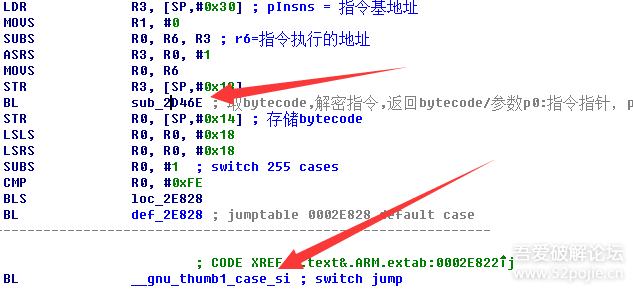
      7.解析指令

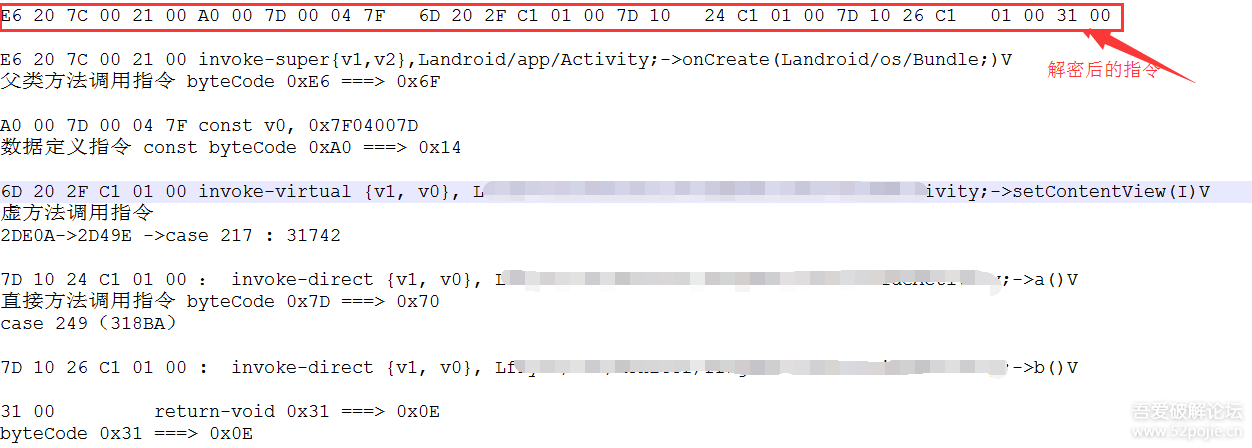
      8.解密指令，获取opCode

      9.根据opcode进行相应的case处理，比如说一条invoke指令，就会通过，FindClass，GetMethod，CallXXXMethodA方法进行翻译



      10.大体的流程就是这样的，每个case的流程，就不写了，因为没看，不会，只是看了用到的部分，以下是某个apk 简单onCreate的还原

二、总结

       360的opcode对应表不是固定的，分析了两个apk，表的对应关系是不一样的，应该是在加固过程中，动态生成的，指令加密方式也是

不一样的，尽管只是简单的异或，所以就没办法进行批处理了，只能手动分析，分析出这个apk对应的opcode表，再进行批处理操作了，反

正脱壳是越来越麻烦了，需要耐心去分析，分析用到的apk就不往论坛传了，放出去不太好，有需要的可以私我。