**反编译防护测试**

**测试描述：**

若客户端进行了加壳保护，无风险

若没有进行加壳保护，进行反编译测试。若大部分代码（核心代码）经过混淆，可认为低风险。若部分代码混淆，可以获知其关键代码（加密或通信等），中风险。

**测试方法：**

1、客户端是否进行加壳保护

* 什么是apk的加壳？

apk的加壳意思就是在程序外面包裹上另外一段代码，保护里面的代码不被非法修改或者反编译。这段代码会在程序运行时优先取得程序的控制权，然后完成它们保护软件的任务。

* 加壳的利弊

优势：保护自己核心代码算法，提高破解、盗版、二次打包的难度；还可以缓解代码注入、动态调试、内存注入攻击。

劣势：影响兼容性；影响程序运行效率

加壳也有一些不容易解决的问题：所有服务端漏洞都不解决。还有文件和logcat中的敏感信息泄露、安装包签名、组件导出，有时附带截屏和Activity劫持。

* 如何进行apk加壳？

现在企业需要对自己的apk进行加壳时，大多会购买第三方加固平台的加固服务，最有名的应当属于：爱加密和梆梆加固。如何自己进行加壳和加壳的原理我还没有深入研究，以后可继续分享，需要了解可以参考网上的资料。

* 如何测试apk是否已进行了加壳？

工具

过程

2、**是否能进行反编译，java代码是否经过混淆**

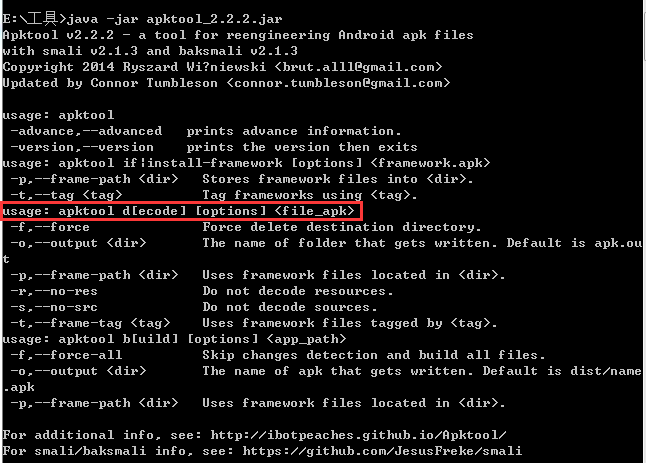
* 安全性描述：对apk进行成功的反编译之后，未被保护的java代码可以使得攻击者能够完整地分析APP的运行逻辑，尤其是相关业务接口协议、和通信加密的实现，并利用代码中的缺陷对客户端或服务器端进行攻击。
* 使用工具：apktool

下载地址<https://bitbucket.org/iBotPeaches/apktool/downloads>

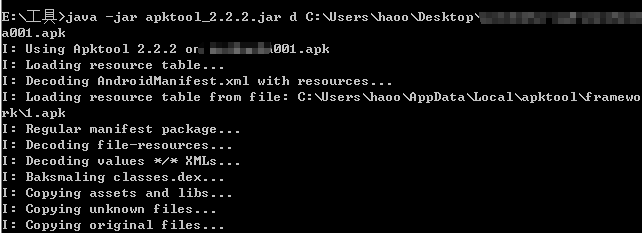
* 测试方法：

a、在apktool所在的文件夹下打开cmd（按住shift键的同时在文件夹空白处右键，选择“在此处打开命令窗口”）

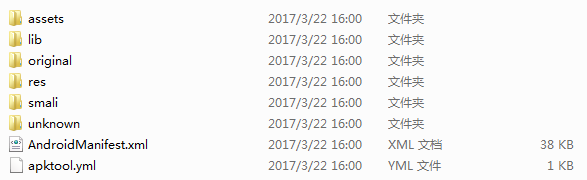
b、输入java -jar apktool.jar，可以看到相关的使用命令的提示，d是反编译指令（注意：你的apktool.jar的名字是什么就写什么，比如我的命令是java -jar apktool1\_2.2.2.jar；也可以把名字改成apktool.jar）



c、再输入**java -jar apktool.jar d xxx.apk**，即可完成反编译。(注意：apk如果不在当前目录下，要把apk所在的路径写全)



d、反编译完成后就会在当前目录下输出一个文件夹，该文件夹包含manifest文件和res下的资源文件，还有smali文件）。smali文件夹是反编译出来的代码。注意，smali文件夹下面，结构和我们的源代码的package一模一样，只不过换成了smali语言，它有点类似于汇编的语法，是Android虚拟机所使用的寄存器语言



e、查看smail文件夹和“lib\armeabi”文件夹下是否有so文件。

客户端程序可以把关键代码以JNI的方式放在so库中，so库是经过编译的arm汇编代码，可以对其进行加壳保护，以防止逆向分析。如果客户端程序使用了JNI技术，在“lib\armeabi”文件夹下会有相应的so库文件。

参考：

https://my.oschina.net/u/2323218/blog/393372

http://blog.csdn.net/jiangwei0910410003/article/details/48415225

http://www.jianshu.com/p/e0661ff7d955 android反编译:apktool和dex2jar使用

https://unclechen.github.io/2016/09/07/Android%E5%8F%8D%E7%BC%96%E8%AF%91%E6%8A%80%E6%9C%AF%E6%80%BB%E7%BB%93/ Android反编译技术总结

https://bitbucket.org/pxb1988/dex2jar/downloads/ dex2jar下载地址

http://jd.benow.ca/

http://download.csdn.net/download/flyingsir\_zw/7712571 smali2java下载地址