ZJDroid脱 爱加密 的壳的经过

原创

火山之父 于 2014-09-05 17:43:43 发布
分类专栏： Android逆向 文章标签： android 脱壳 ZJDroid

用ZJDroid给某爱加密加壳的apk脱壳的经过。

本菜鸟虽然编码的经验还是有的，但是破解可以说是第一次。这次在用ZJDroid给某个APK脱壳碰到了一些问题，所以记录下来，如果能给碰到同样问题的童靴提个醒，就很高兴了。所以 高手请绕道，别笑话菜鸟啊 ：）

背景

这个APK包名为com.xxx.client包含了一个so文件，叫libxxx.so
整个包用爱加密加了壳。 典型特征就是apk的assert目录下有ijiami.dat文件

原理

一句话来说的话，就是用ZJdroid的"backsmali"的命令直接dump出来已经脱过壳的dex的内容。Android 安全加固行业分析和破解这个文章对爱加密有个很关键的一句话分析

> 在C层对Dex进行解密并加载,替换内存中的Dex 为原Dex,替换自定义的Application的运行环境 为原Application

因此，最简单的方法就是等爱加密把Dex的内容替换完了，再直接dump出来就可以了。

原理貌似很简单，可是我这种菜鸟还是花了两个星期啊，:(

环境准备

已经root的手机一个我开始是想在模拟器上测试的，但是后面碰到root问题，和分区加载顺序问题等，太麻烦了，耽误时间。所以还是搞个真机吧。
我的是小米1s，这里要注意了，虽然Xposed针对MIUI做了特殊处理，但是在我这个小米1s上还是出了问题，加载资源异常，
所以我给我的小米1s刷了小米提供的原生android版本.参见线刷MIUI及原生4.1教程

安装Xposed和ZJdroid一切顺利，就不特别说明了，记得要激化组件哦。
可以用 adb logcat "Xposed:V *:S"查看Xposed的log，确认一切正常

使用ZJDroid开始破壳这个可以参见Android动态逆向分析工具ZjDroid--脱壳神器
我开始先用“dump*dex*” *命令 dump ，结果得到的dex还是爱加密加固后的stub的dex。*
*但是用"dump*class" 可以发现dump出来的类是我感兴趣的类。那应该内存里确实有已经脱过壳的dex信息了。
后面看到了”fanqiliang“在看雪的“Android 安全加固行业分析和破解”帖子里对爱加密的分析，就更坚定了我的猜测:
"爱加密是把桩dex加载到内存后，在第一次load真正的class时候，把内存里桩dex的内容替换成脱壳后的clas内容"
所以直接dump_dex不行，那就用”backsmali“命令直接反编译dex的class数据吧。
这个过程中碰到了两个问题，记录如下：

坑和规避方法

dvm heap size不够导致dvm-heap-alloc xxx object failed .
由于我的这个apk有快2000多个类，比较大，
所以"backsmali" 在反汇编成smali文件时，一切顺利，但是再把smali文件组成dex时，由于都是在内存里完成，估计需要内存比较大，因此后面在分配内存时失败了。
解决方法：加大vm hap的内存，修改/system/build.prop中的dalvik.vm.heapsize=512m//之前的256m改成512m
重启，就没有分配内存的错误了。

```
diff --git a/src/com/android/reverse/smali/DexFileBuilder.java b/src/com/android
/reverse/smali/DexFileBuilder.java
index f1d3dfd..793561e 100644
--- a/src/com/android/reverse/smali/DexFileBuilder.java
+++ b/src/com/android/reverse/smali/DexFileBuilder.java
@@ -33,7 +33,7 @@ public class DexFileBuilder {

        public static boolean buildDexFile(String smaliPath,String dexFileName)
{

-               int jobs = 8;
+               int jobs = 4;
                boolean allowOdex = false;
                boolean verboseErrors = false;
                boolean printTokens = false;
@@ -172,7 +172,10 @@ public class DexFileBuilder {
                dexGen.setVerboseErrors(verboseErrors);
                dexGen.setDexBuilder(dexBuilder);
                dexGen.smali_file();
-
+               //ljb add
+               reader.close();
+               fis.close();
+
                return dexGen.getNumberOfSyntaxErrors() == 0;
        }
```


报java.lang.File.finalaze() timeout.
这个问题看起来很奇怪，但是猜测到小文件这么多，有可能是文件句柄没有及时释放导致的。
所以，修改ZJdroid的src/com/android/reverse/smali/DexFileBuilder.java文件，及时的关闭打开的句柄 ，编译，重装就OK了。

到此终于看到了dexfile.dex了。
现在就要把xxx重新打包，看我们脱壳后的dex是否正常
1. apktools2 d -s xxx.apk, 输出目录为xxx
2. 把dexfile.dex替换xxx里的classes.dex
3. 把xxx/assert里的ijiami.dat删除
4. 把xxx/lib/armeabi里的libexec.so libexecmain.so删除
5. 修改xxx/AndroidManifest.xml 把application里的android:name替换回来为"com.xxx.client.MyApplication"。
6. apktools2 b xxx
7. autosing 签名新生成的apk
安装，运行，一切正常。

总结

1. ZJDroid不愧为脱壳神器啊
2. 自己还要多补充系统知识啊。争取能够全面理解加壳脱壳的原理 ：）