APK防反编译技术

罗升阳

http://weibo.com/shengyangluo

http://blog.csdn.net/luoshengyang

About Me

- 《老罗的Android之旅》博客作者
- 《Android系统源代码情景分析》书籍作者
- 博客：http://blog.csdn.net/Luoshengyang
- 微博：http://weibo.com/shengyangluo

Agenda

- 添加非法指令
- 隐藏敏感代码
- 伪APK加密技术
- 总结和思考

添加非法指令

- Step 1: 在APK包含一个无关类Bomb，该类有一个成员函数drop
- Step 2: 将APK的classes.dex解压出来，并且用dexdump进行反编译，找到Bomb.drop在classes.dex的偏移
- Step 3: 用vim以二进制方式打开classes.dex，转到Bomb.drop的偏移处，将前两个字节修改为FF FF（非法指令）
- Step 4: 重新打包和签名APK，用adb install命令安装，日志提示checksum不一致

添加非法指令

- Step 5: 用dexdump验证APK的checksum，并且将正确的checksum替换原classes.dex的checksum
- Step 6: 重新打包和签名APK，用adb install安装，DONE

隐藏敏感代码

- Step 1:在APK包含一个无关类Bomb，该类有一个成员函数drop，前面留有18个字节的垃圾指令
- Step 2:将APK的classes.dex解压出来，并且用dexdump进行反编译，找到Bomb.drop在classes.dex的偏移
- Step 3:用vim以二进制方式打开classes.dex，转到Bomb.drop的偏移处，将前18个字节修改为以下指令

```
035b0c: 3200 0900                                  |0000: if-eq v0, v0, 0009 // +0009
035b10: 2600 0300 0000                             |0002: fill-array-data v0, 00000005 // +00000003
035b16: 0003 0100 hhhh hhhh .... .... .... ...     |0005: array-data (hhhhhhhh units)
```

隐藏敏感代码

- Step 4: 用dexdump查看classes.dex的头部信息，找到class_def的偏移，以及Bomb类的class index
- Step 5: class_def的偏移，加上Bomb类的class index乘以32的积，即可得到用来描述Bomb类的class_def结构体偏移，再往前4个字节，即为其access_flags，将它的第16位设置为1，一般就是设置为0x10001，也就是将0100 0000设置为0100 0100，这样可以将Bomb类设置为已验证

隐藏敏感代码

- Step 6: 用dexdump验证APK的checksum，并且将正确的checksum替换原classes.dex的checksum
- Step 7: 重新打包和签名APK，用adb install安装，DONE

伪APK加密技术

- ZIP中的每一个文件都有一个2字节大小的全局方式位标记，其中第0位表示是否加密
- 如果ZIP中的一个文件标志为加密，那么在解压时，就需要指定解压密码
- APK默认都是不加密的，也就是APK在安装解压时，它里面的文件的加密位标志都会被忽略
- Apktool发现APK是加密的时候，会抛出一个异常出来
- 利用上述差异，就可以给APK设置一个加密标志，但不对其进行加密，从而阻止Apktool反编译

总结

- APK在安装时，会对APK进行验证，主要是checksum验证和指令合法性验证
- 安装通过验证的APK会被标记为已验证，并且会被优化
- 安装没有通过验证的APK仍然会被成功安装，但是它会被标记为未验证
- 未验证的APK的某个类在被加载时，会被验证，一旦验证失败，就会抛出异常
- 反编译工具会对APK的所有类进行验证，不管这个类安装后会不会被加载，一旦验证不通过，就会抛出异常

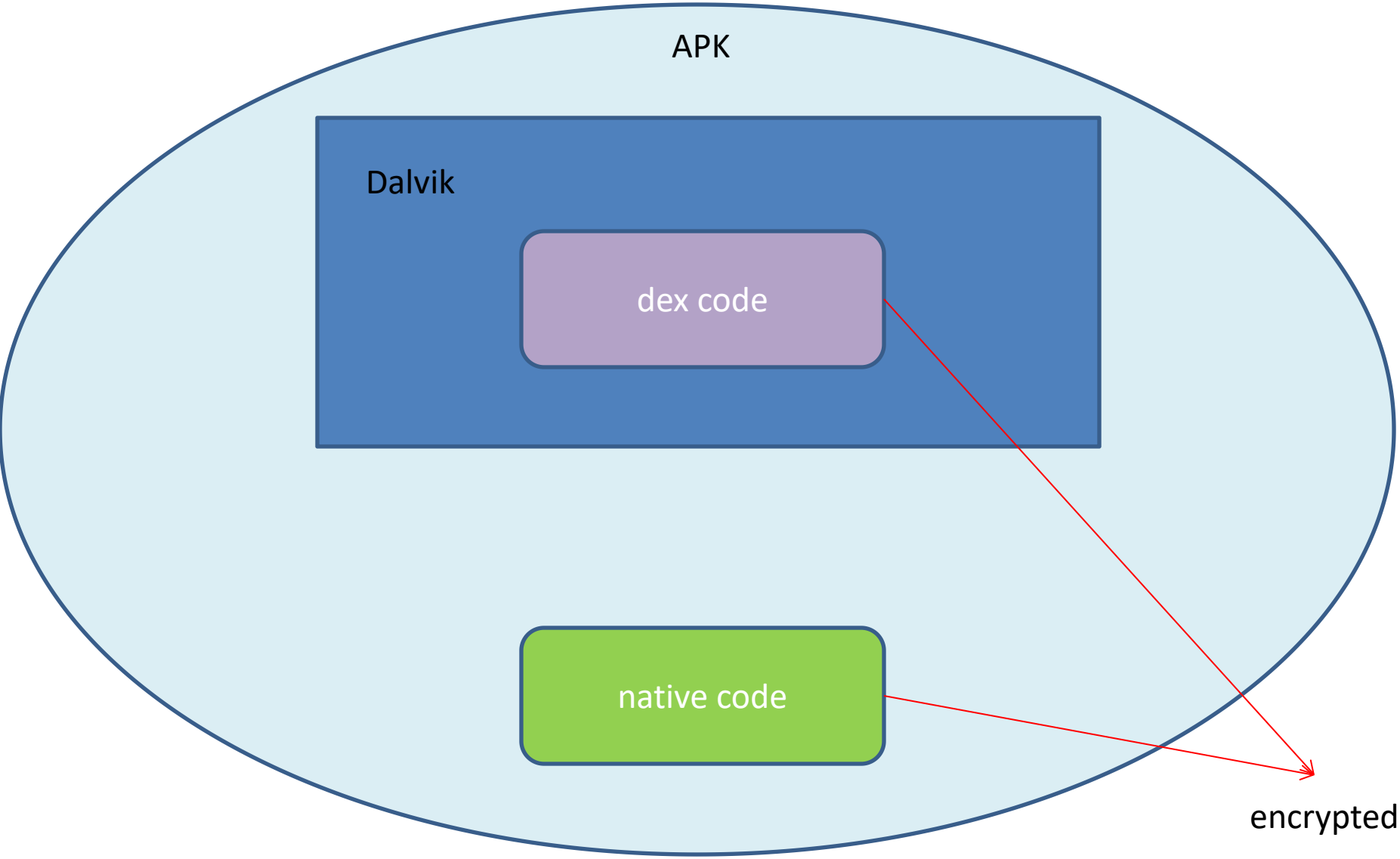
总结

- 根据以上的APK安装、验证和加载流程，就可以通过以下两个方法来阻止反编译：
  - 包含一个有非法指令（不能执行）的类，但是这个类保证永远不会被加载
  - 包含一个有非法指令（能执行）的类，同时将该类预先设置为已验证的

思考

- 反编译工具，如Apktool，采用的是线性扫描算法来分析指令流，当遇到非法指令时，就会异常退出
- 如果遇到非法指令只是简单忽略，那么就仍能正常工作
- 如果反编译工具采用的递归遍历算法来分析指令流，那么就能将fill-array-data之类的伪指令反编译出来
- 如果反编译工具忽略掉APK的加密标志，那么就能将伪加密的APK也反编译出来
- 如果反编译工具更智能一些，那么一切皆可反编译.....
- 怎么办？

思考



思考

- Dex Code加密
  - Android 4.0及以上的DexFile提供有加载内存dex文件的隐藏接口openDexFile和defineClass，通过它们就可以从内存中加载一个dex文件，在真正加载这块内存之前，可以对其进行加密
  - Android 4.0之前的版本，可以用libdvm的导出函数来实现上述相同的功能
- Native Code加密
  - 系统中的每一个so文件都是由/system/bin/linker来进行加载和解析的
  - 参考/system/bin/linker，实现一个自己的linker，这个linker可以加载和解析内存so文件，从而实现加密处理

Q&A

Thank You