现在已经有很多方法可以反编译apk文件，常用的有dex2jar和JD-Gui的组合，可以看到近似java的代码，非常容易理解，使得破解一个apk的难度大大降低。

但有时候，静态看代码也非常复杂，参数传来传去，尤其是那些不是直接调用，而是通过类似intent传过来的，要找到调用的源并了解每个参数的含义，还是比较复杂的。

那么有没有一种方法，可以方便的在没有源码的情况下，对一个apk进行调试，从而更加简单的了解调用逻辑呢？答案当然是肯定的，下面我就将介绍一种方法，使用ApkTool和Eclipse ADT的DDMS。

首先，当然是准备工具，ADT就不解释了，很容易。至于ApkTool，我们使用目前最新的2.0.0b9版本。可以从这里下载到：[http://connortumbleson.com/2014/02/apktool-2-0-0-beta-9-released/](http://connortumbleson.com/2014/02/apktool-2-0-0-beta-9-released/" \t "_blank)。特别注意的是，新的版本不像1.x的版本，必须使用java 7，否则无法使用。

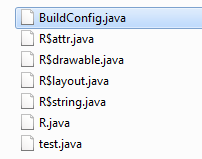
好了，工具准备好了，我们正式开始动手了，主要分以下几个步骤：

1）使用ApkTool反编译，注意和以往不同的是，此时需要打开debug功能（加上-d选项）

**[plain]** [view plain](http://blog.csdn.net/roland_sun/article/details/26399669) [copy](http://blog.csdn.net/roland_sun/article/details/26399669)

1. java -jar apktool\_2.0.0b9.jar d -d xxx.apk -o out

那么加上这个-d选项和不加有什么区别吗？我们来看看解出来的代码，进入out\smali目录看看：



看见了没有，后缀名变成了.java，而不是普通的.smali了。这样做是为了方便后面在Eclipse里面建立一个对应的Java项目来调试。

那么具体的代码有什么变化吗？



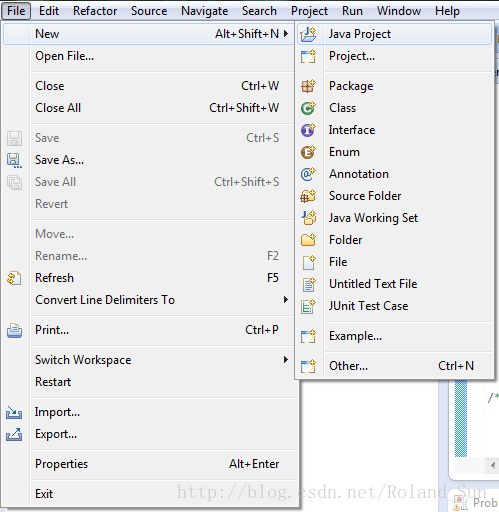
加了-d选项后，原有的smali代码变成了在Java的注释里，同时在外面还“套”了一个伪造的类，还有每行注释前面的费代码“a=0”。这么做的目的就是"欺骗"Eclipse，我是一个Java项目呀，其实根本不是。

2）完成了上一步，下一步要把程序的debug选项打开。打开此项的目的是为了告诉Android系统，本程序允许调试。否则，调试器是连接不到这个程序上去的。进入out目录，看到AndroidManifest.xml文件，随便用一个文本编辑器打开，找到<application>，加一个属性android::debuggable="true"进去：

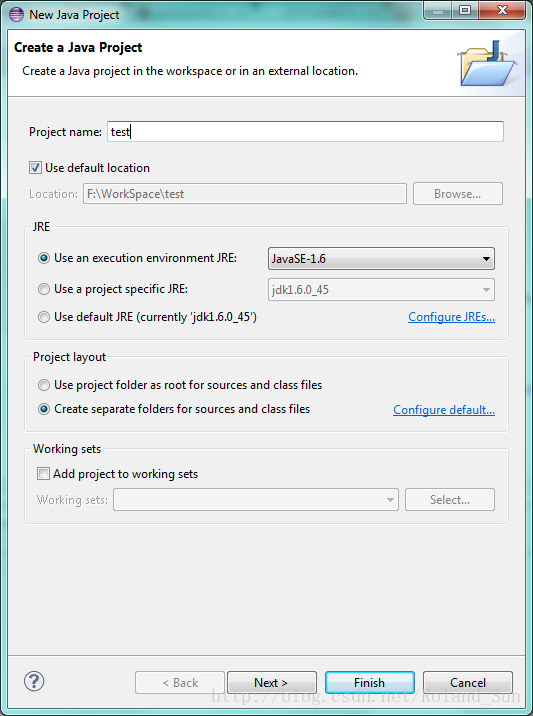
http://img.blog.csdn.net/20140520174408031?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvUm9sYW5kX1N1bg==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast

3）在Eclipse中创建一个Java项目，并把out\smali下的所有代码拷贝到新创建项目的src目录下去。

点击File->New->Java Project



在接下来的“New Java Project”页面里，随便写一个项目名，然后点击Finish



最后直接把out\smali下的所有代码拷贝到新创建项目的src目录下去，按F5刷新一下就行了。

4）用ApkTool把修改过AndroidManifest.xml的程序打包回apk文件（同样要加上-d的选项）

**[plain]** [view plain](http://blog.csdn.net/roland_sun/article/details/26399669) [copy](http://blog.csdn.net/roland_sun/article/details/26399669)

1. java -jar apktool\_2.0.0b9.jar b -d out -o debug.apk

这里记住一定要加上-d选项。加上后，ApkTool会自动去除在前面反编译时加上的类和无用Java代码，只解释在Java注释中的smali代码，并且还会在dex文件中加上相应的debug信息。

5）打包回去的文件此时还无法安装，原因是还没有签名，没有签名的程序是无法安装到手机上的。可以用自己生成的证书对apk文件签名，现在有很多工具可以签名，也可以使用Java自带的命令进行签名

**[cpp]** [view plain](http://blog.csdn.net/roland_sun/article/details/26399669) [copy](http://blog.csdn.net/roland_sun/article/details/26399669)

1. jarsigner -verbose -keystore roland.keystore -signedjar debug\_signed.apk debug.apk roland.keystore

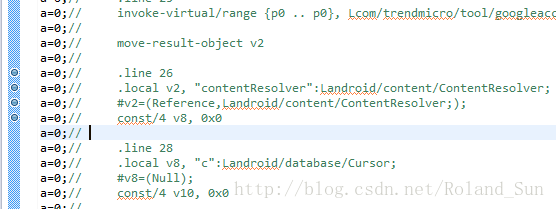
不清楚的可以看[这里](http://blog.csdn.net/roland_sun/article/details/26452219" \t "_blank)。

6）连接上设备，将安装程序推送上去

**[cpp]** [view plain](http://blog.csdn.net/roland_sun/article/details/26399669) [copy](http://blog.csdn.net/roland_sun/article/details/26399669)

1. adb install debug\_signed.apk

7）在你想要程序停止的地方设置断点，可以在代码大致的位置附近多设置几个断点，以免断不下来

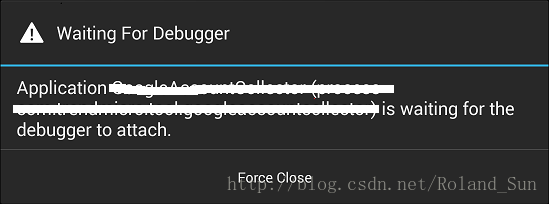


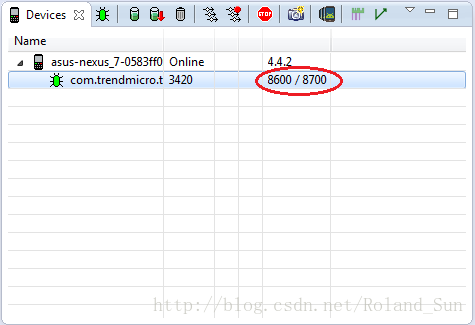
8）通过adb以调试模式启动该应用程序

**[cpp]** [view plain](http://blog.csdn.net/roland_sun/article/details/26399669) [copy](http://blog.csdn.net/roland_sun/article/details/26399669)

1. adb shell am start -D -n ｛Package Name}/.{Activity}

此时在调试设备上会显示等待调试程序接入

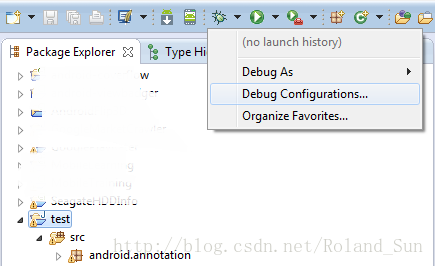
  
9）切换到DDMS页面，可以看到需要调试的程序已经显示在列表里面了



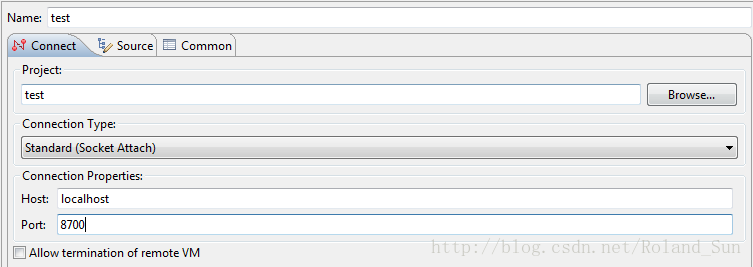
在列表里，找到对应程序的调试端口。如图所示，8600或者8700都可以。

10）新建Java远程调试

先选择到第3步建立的那个Java项目，点击Debug按钮，在下拉菜单中选择“Debug Configurations...”



在新弹出的页面中，双击左边菜单中的“Remote Java Application”，在右边新弹出的页面中特别注意的是，要将调试端口改到前面第9步中查到的端口。



接着点击页面右下角的Debug按钮，大功告成！

下面的调试方法就和普通的Java程序调试一样了，F5、F6啦~~~