

# Apk加固之代码混淆

## 实验概述

Java代码是很容易反编码的，且Android开发的应用程序是用Java代码写的，为了更好的保护Java源代码，需要对编译好后的class文件进行混淆，本实验使用ProGuard混淆apk的代码。

## 实验目的

- 1、熟悉eclipse开发工具
- 2、学习代码混淆的原理
- 3、了解混淆代码的作用
- 4、了解反编译过程
- 5、学习ProGuard如何进行代码混淆

## 实验原理

代码混淆是指将计算机程序的代码，转换成一种功能上等价且让人很难看懂的代码，所谓功能上的等价是指其在变换前后功能相同或相近。其解释如下：程序P经过混淆变换为P'，若P没有结束或错误结束，那么P'也不能结束或错误结束；而且P'程序的结果应与程序P具有相同的输出。否则P'不是P的有效混淆。

目前对于混淆的分类，普遍是以Collberg的理论为基础，分为布局混淆、数据混淆、控制混淆和预防混淆这四种类型。

### 布局混淆

布局混淆是指删除或者混淆软件源代码或者中间代码中与执行无关的辅助文本信息，增加攻击者阅读和理解代码的难度。软件源代码中的注释文本、调试信息可以直接删除，用不到的方法和类等代码或数据结构也可以删除，这样即可以使攻击者难以理解代码的语义，也可以减小软件体积，提高软件装载和执行的效率。软件代码中的常量名、变量名、类名和方法名等标识符的命名规则和字面意义有利于攻击者对代码的理解，布局混淆通过混淆这些标识符增加攻击者对软件代码理解的难度。标识符混淆的方法有多种，例如哈希函数命名、标识符交换和重载归纳等。哈希函数命名是简单地将原来标识符的字符串替换成该字符串的哈希值，这样标识符的字符串就与软件代码不相关了；标识符交换是指先收集软件代码中所有的标识符字符串，然后再随机地分配给不同的标识符，该方法不易被攻击者察觉；重载归纳是指利用高级编程语言命名规则中的一些特点，例如在不同的命名空间中变量名可以相同，使软件中不同的标识符尽量使用相同的字符串，增加攻击者对软件源代码的理解难度。布局混淆是最简单的混淆方法，它不改变软件的代码和执行过程。

### 数据混淆

数据混淆是修改程序中的数据域，而对代码段不作处理。常用的数据混淆方式有合并变量、分割变量、数组重组、字符串加密等。

合并变量是将几个变量合并为一个数据，原来的每个变量占据其中一个区域，类似于一个大的数据结构。分割变量则是将一个变量分割为两个变量，对分割前后提供一种映射关系，将对一个变量的操作转化为对分割后两个变量的操作。

数组重组有数组的分割、合并、折叠和平滑等几种方式。分割是将一个数组分成2个或多个相同维度的数组；合并则相反；折叠是增加数组的维数；平滑则是相反。

在ELF文件中，全局变量和常量字符串存放在数据段中，反汇编工具可以轻易查找到字符串与代码之间的引用关系。在软件破解中，通过一些字符串提示可以很方便的找到代码关键语句，从而破解软件。字符串加密则可以对这些明显的字符串进行加密存储，在需要时再进行解密。

## 控制混淆

控制混淆也称流程混淆，它是改变程序的执行流程，从而打断逆向分析人员的跟踪思路，达到保护软件的目的。一般采用的技术有插入指令、伪装条件语句、断点等。伪装条件语句是当程序顺序执行从A到B，混淆后在A和B之间加入条件判断，使A执行完后输出TRUE或FALSE，但不论怎么输出，B一定会执行。

## 预防混淆

预防混淆一般是针对专用的反编译器设计的，目的就是预防被这类反编译器反编译。他是利用特定的反编译器或反混淆器的弱点进行专门设计。预防混淆对于特定的反编译器非常有效，所以在使用时要综合利用各种反编译器的特点进行设计。

ProGuard是一个混淆代码的开源项目，它的主要作用是混淆代码，ProGuard还包括以下4个功能。

压缩(Shrink): 检测并移除代码中无用的类、字段、方法和特性 (Attribute)。

优化(Optimize): 对字节码进行优化，移除无用的指令。

混淆(Obfuscate): 使用a, b, c这样简短而无意义的名称，对类、字段和方法进行重命名。

预检(Preverify): 在Java平台上对处理后的代码进行预检，确保加载的class文件是可执行的。

根据官网的翻译: Proguard是一个Java类文件压缩器、优化器、混淆器、预校验器。压缩环节会检测以及移除没有用到的类、字段、方法以及属性。优化环节会分析以及优化方法的字节码。混淆环节会用无意义的短变量去重命名类、变量、方法。这些步骤让代码更精简，更高效，也更难被逆向（破解）。

## 实验环境

虚拟机: kali linux


apk: pictureviewer.apk

工具: jd-gui、d2jdex2jar、eclipse

## 实验步骤

1、进入到apk目录，复制实验用的pictureviewer.zip到root目录下并改后缀为zip。

操作：“cd apk/”>“ls”>“cp pictureviewer.apk /root/pictureviewer.zip”>“cd”如图1



```
root@kali: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
root@kali:~# cd apk/
root@kali:~/apk# ls
AdobeReader.apk      game.apk             sieve.apk
ContentProvider.zip  pictureviewer.apk    test.apk
drozer-agent-2.3.4.apk QQ_410.apk           vulnsqlite.apk
root@kali:~/apk# cp pictureviewer.apk /root/pictureviewer.zip
root@kali:~/apk# cd
root@kali:~# ls
android-sdk-linux  bluedon.png  src      workspace  模板  图片  下载  桌面
apk                pictureviewer.zip  tools    公共      视频  文档  音乐
root@kali:~#
```

图 1修改后缀

2、解压pictureviewer.apk。

操作：“mkdir test1”>“unzip pictureviewer.zip -d test1”>“ls”>“cd test1”>“ls”如图 2

```
root@kali:~# mkdir test1
root@kali:~# unzip pictureviewer.zip -d test1/
Archive: pictureviewer.zip
  inflating: test1/AndroidManifest.xml
  inflating: test1/res/drawable/grid_background.xml
  extracting: test1/res/drawable/icon.png
  extracting: test1/res/drawable/imageborderfocus.png
  extracting: test1/res/drawable/imagebordernormal.png
  extracting: test1/res/drawable/imageborderpressed.png
  extracting: test1/res/drawable/noimage.png
  extracting: test1/res/drawable/nothumbnail.png
  inflating: test1/res/layout/main.xml
  extracting: test1/resources.arsc
  inflating: test1/classes.dex
  inflating: test1/META-INF/MANIFEST.MF
  inflating: test1/META-INF/CERT.SF
  inflating: test1/META-INF/CERT.RSA
root@kali:~# ls
android-sdk-linux  bluedon.png      src      tools      公共  视频  文档  音乐
apk                pictureviewer.zip test1     workspace  模板  图片  下载  桌面
root@kali:~# cd test1/
root@kali:~/test1# ls
AndroidManifest.xml  classes.dex  META-INF  res  resources.arsc
root@kali:~/test1#
```

创建解压目录  
解压apk包  
代码文件

图 2解压apk

3、将classes.dex文件转化为jar文件。

操作：“d2j-dex2jar classes.dex”>“ls”如图 3

```
root@kali:~/test1# d2j-dex2jar classes.dex
dex2jar classes.dex -> classes-dex2jar.jar
root@kali:~/test1# ls
AndroidManifest.xml  classes-dex2jar.jar  res
classes.dex         META-INF             resources.arsc
root@kali:~/test1#
```

将dex文件转化为jar  
java文件

图 3 jar文件转化

4、打开jd-gui工具。

操作：“cd /root/tools/jd-gui”>“./jd-gui.jar”如图 4

```
root@kali:~/test1# cd /root/tools/jd-gui/
root@kali:~/tools/jd-gui# ./jd-gui.jar
```

图 4开启jd-gui

5、在jd-gui中打开反编译获得的jar文件。

操作：“单击左上角位置”>“选择主文件夹”>“选择test1”>“把 classes-dex2jar.jar文件拖入jd中” 如图 5



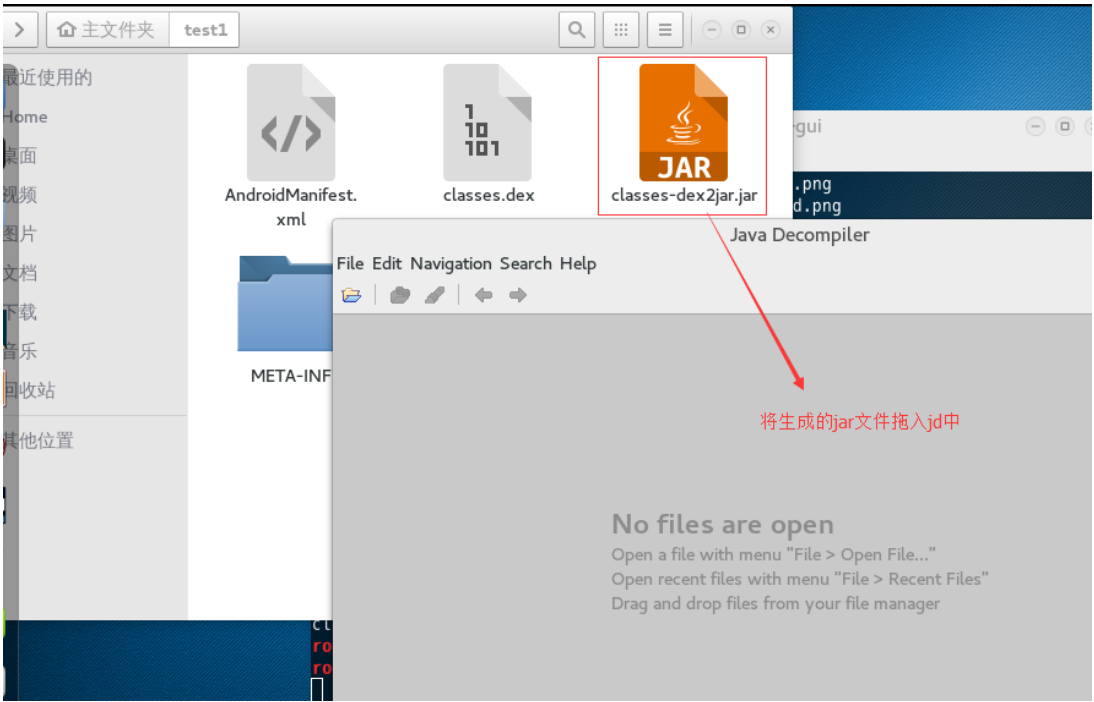


图 5打开jar文件

6、可以看到没有经过混淆的java代码。如图 6

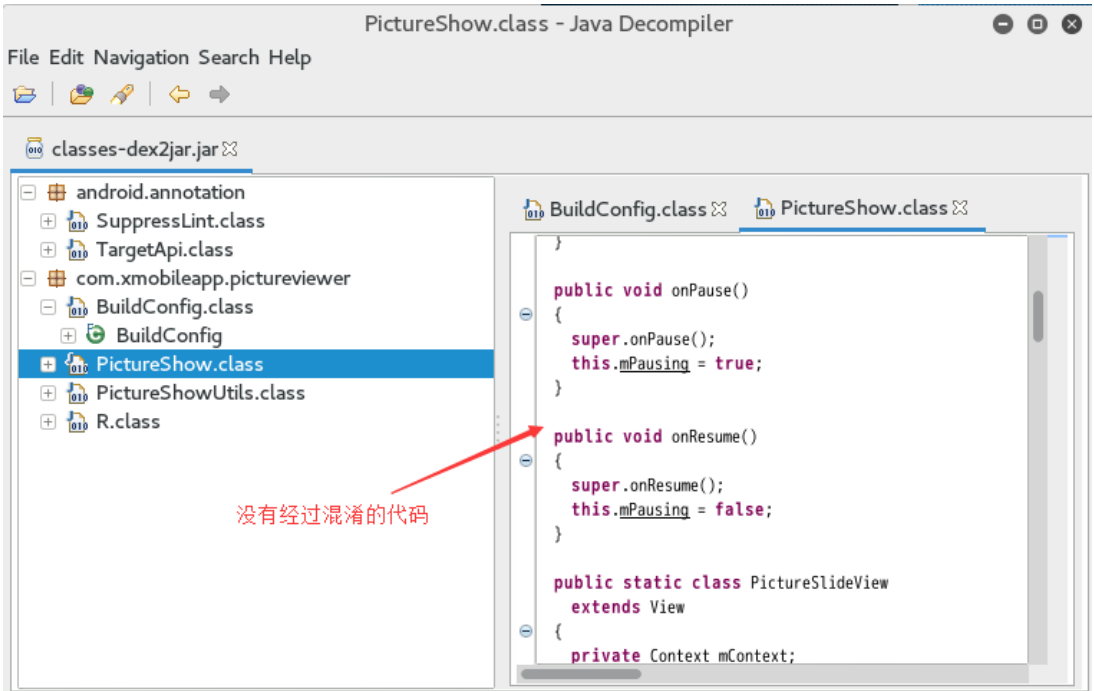


图 6 apk的java代码

7、启动eclipse。

操作： “cd /root/tools/eclipse”>“./eclipse”如图 7

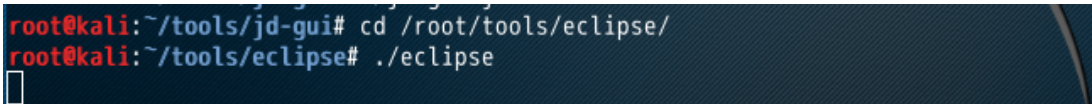


图 7启动eclipse

8、默认工作目录为/root/workspace，单击OK。如图 8

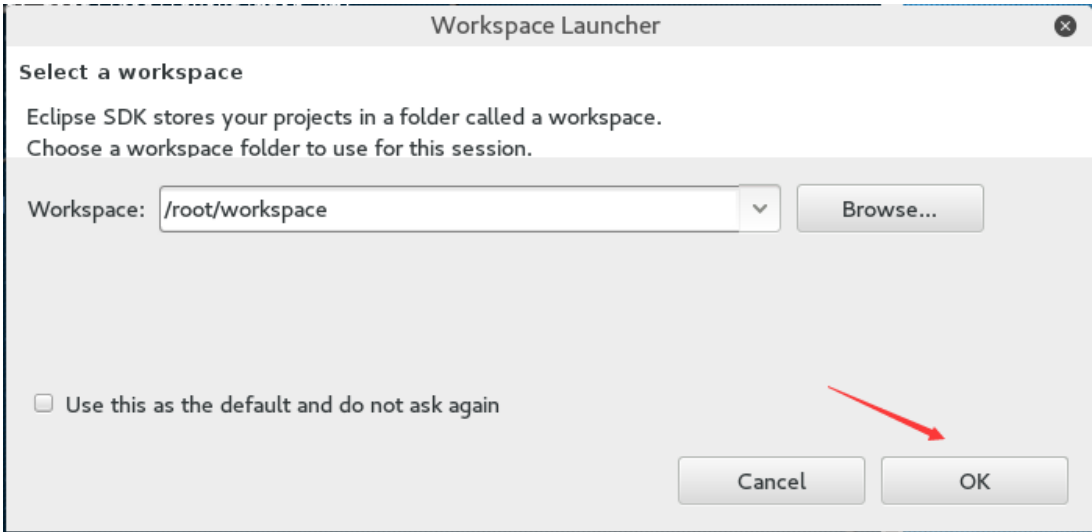


图 8默认工作目录

9、进入到eclipse主界面，如图 9

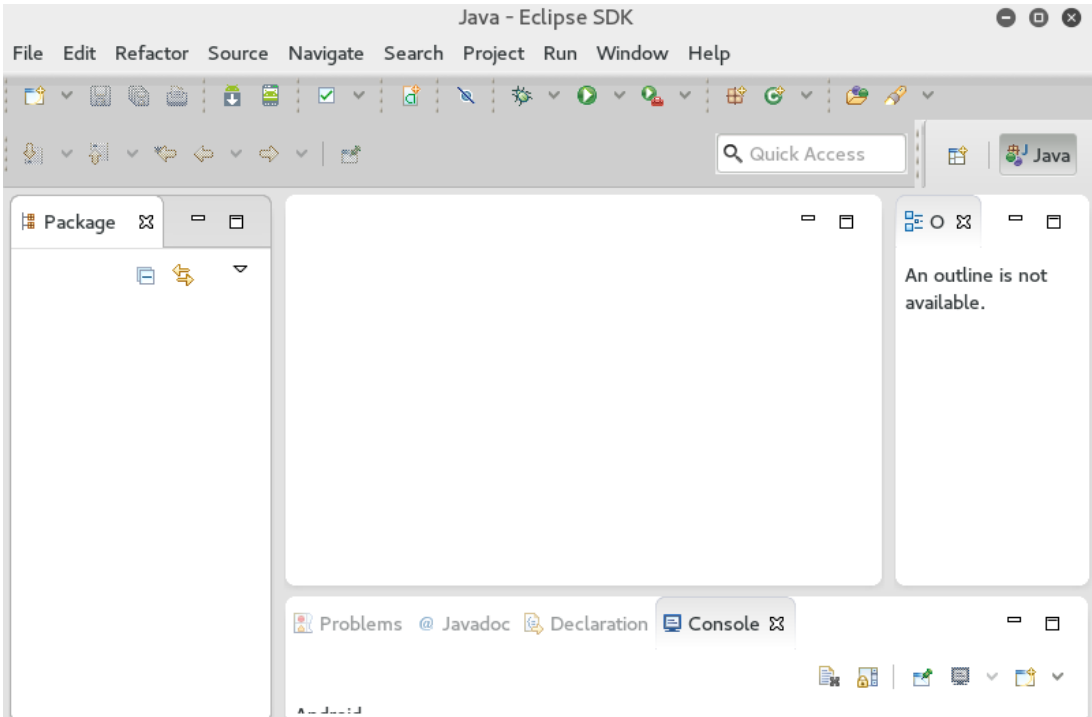


图 9 eclipse界面

10、操作：“移动到flie”>“选择import导入android项目” 如图 10

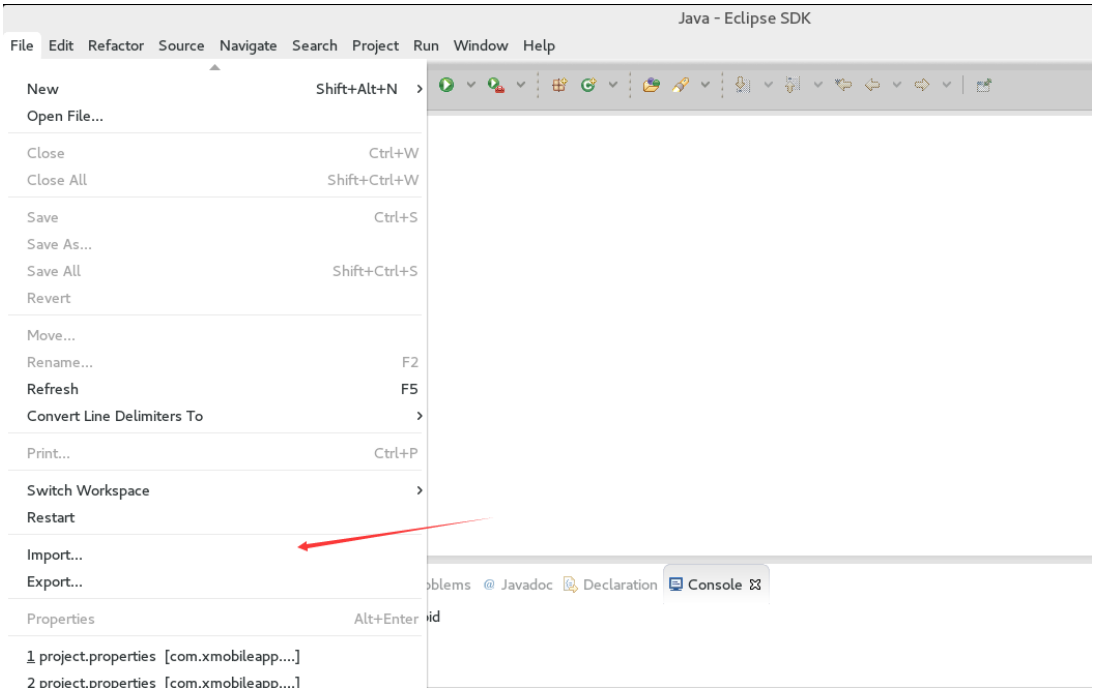


图 10导入项目

11、操作:“选择 Android”>“Existing Android Code into Workspace”>“单击 next”如图 11

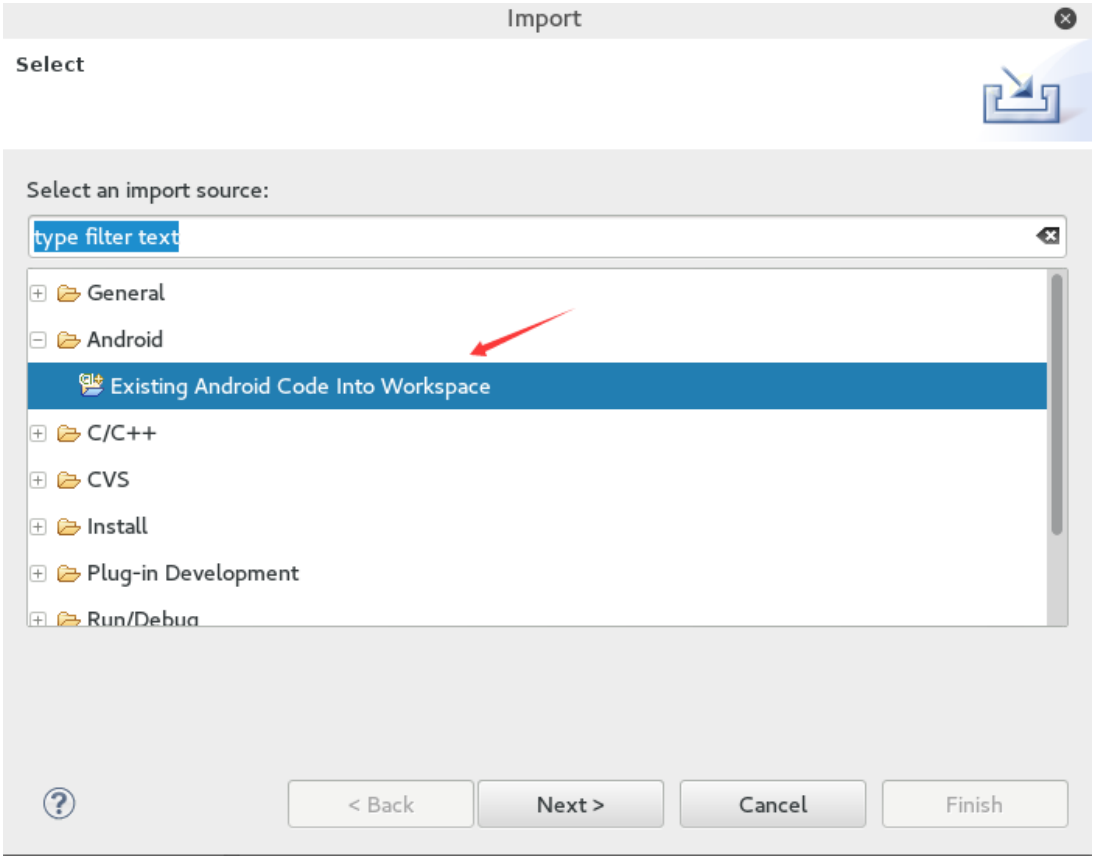


图 11选择存在的项目

12、操作:“选择/root/src/pritureviewer”>“确认” 如图 12

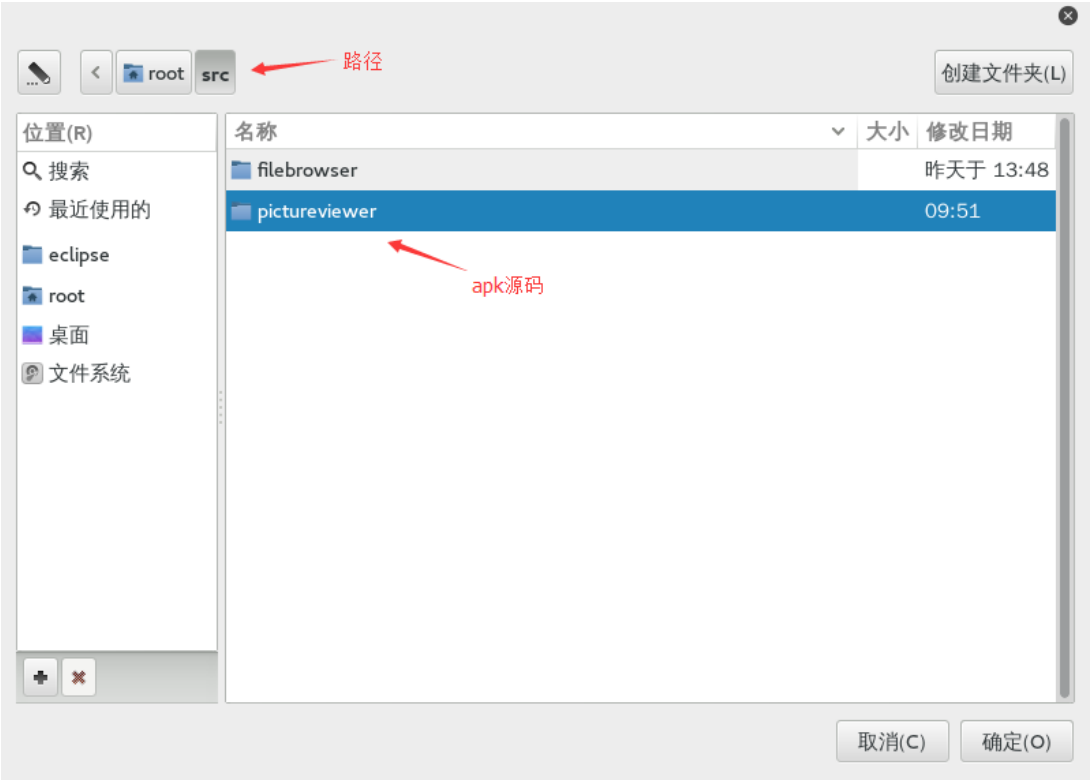


图 12选择实验apk源码

13、选择完成之后会出现如的界面，然后单击finish。图 13

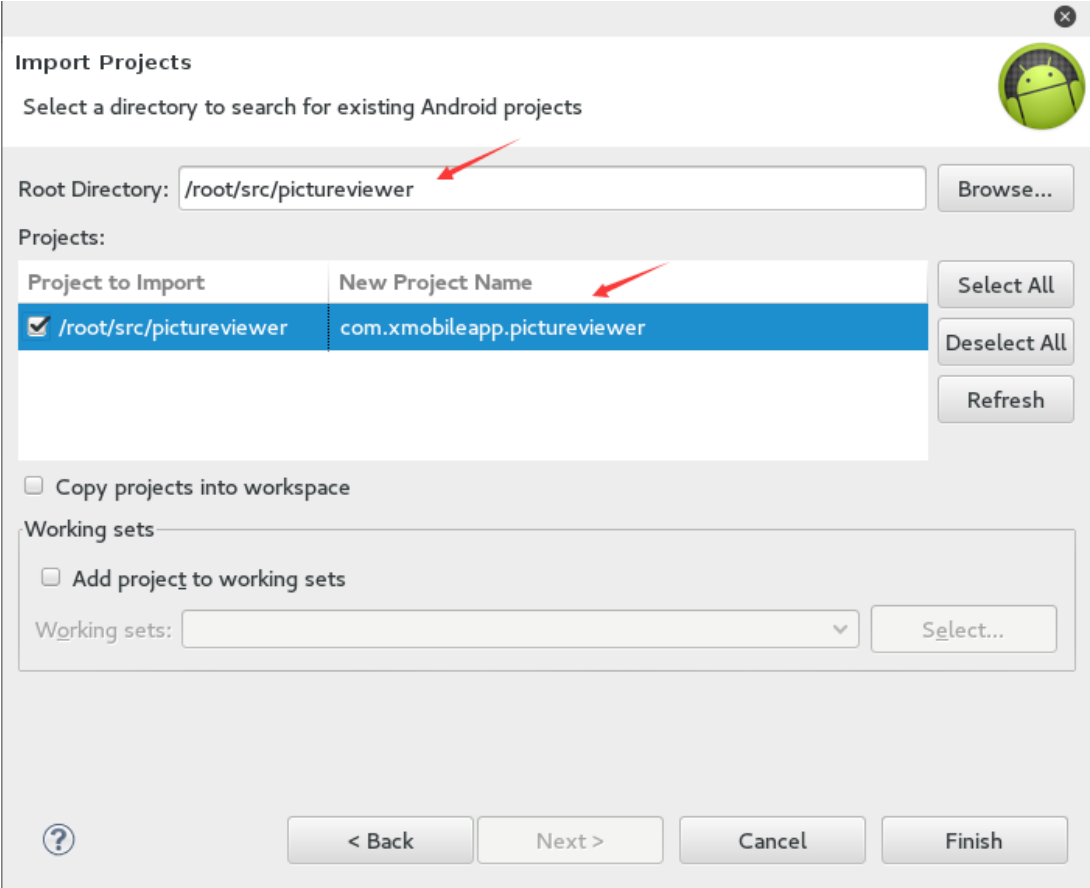




图 13选择完成

14、在左框框可以看到导入的android项目，此项目为实验用pictureviewer.apk的源码如图 14

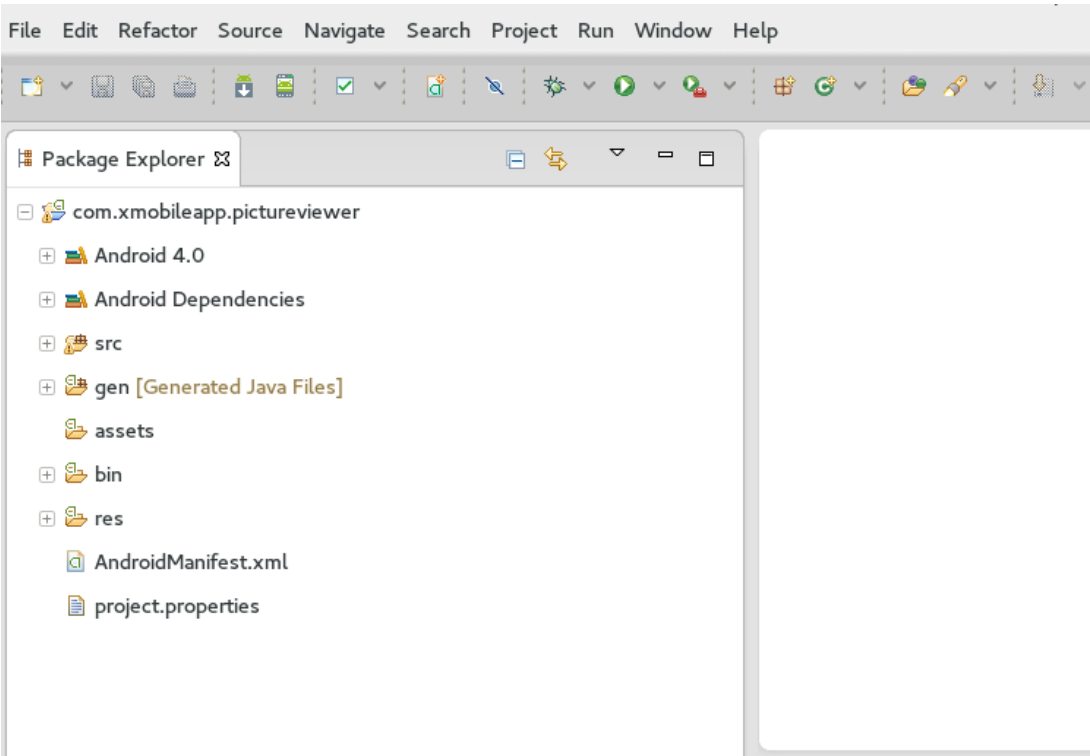


图 14 项目源码

15、修改project.properties，为apk增加代码混淆功能。

操作：“将如下的代码#号去除”>“删除此代码后面的:proguard-project.txt”>“修改\${sdk.dir}为/root/android-sdk-linux”如图 15图 16

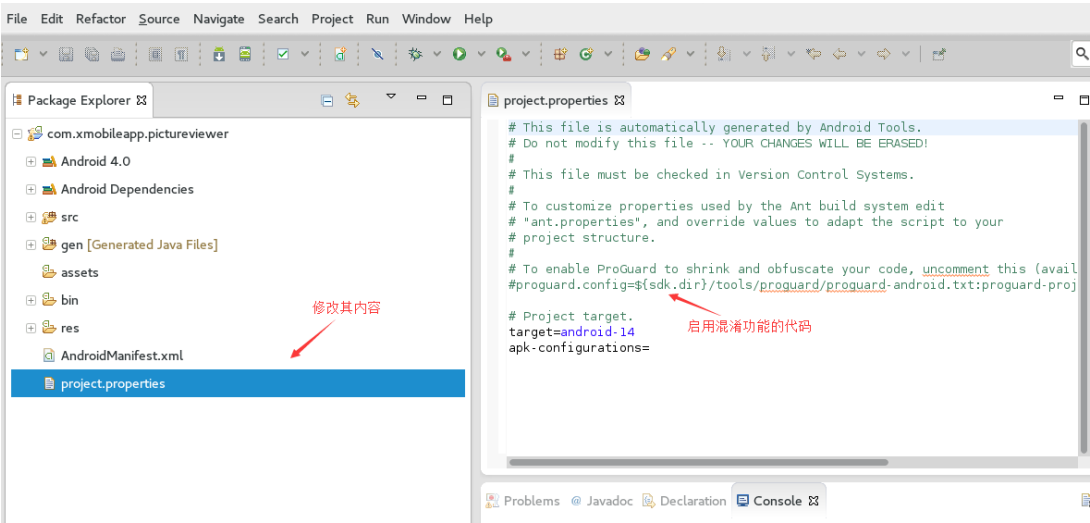


图 15修改前

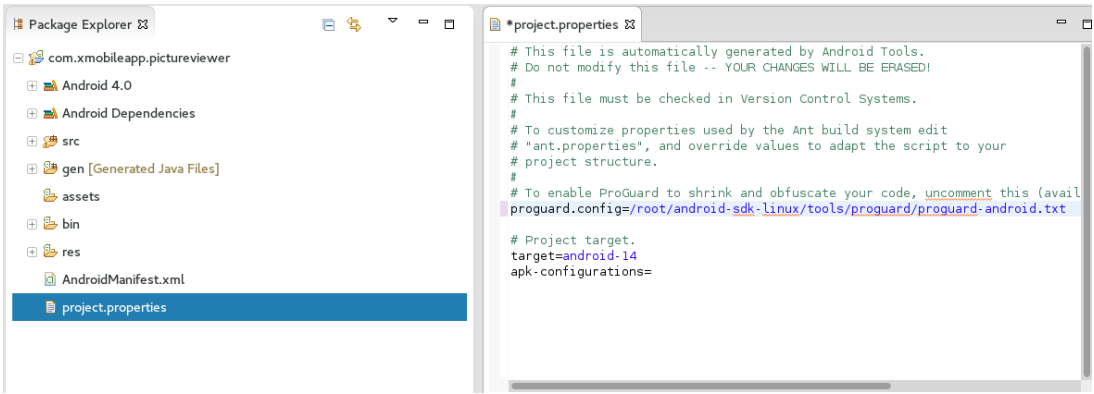


图 16修改后

16、将android项目导出为签名的apk。

操作：“选择包右键”>“选择Android Tools”>“Export Signed Application Package”如图 17

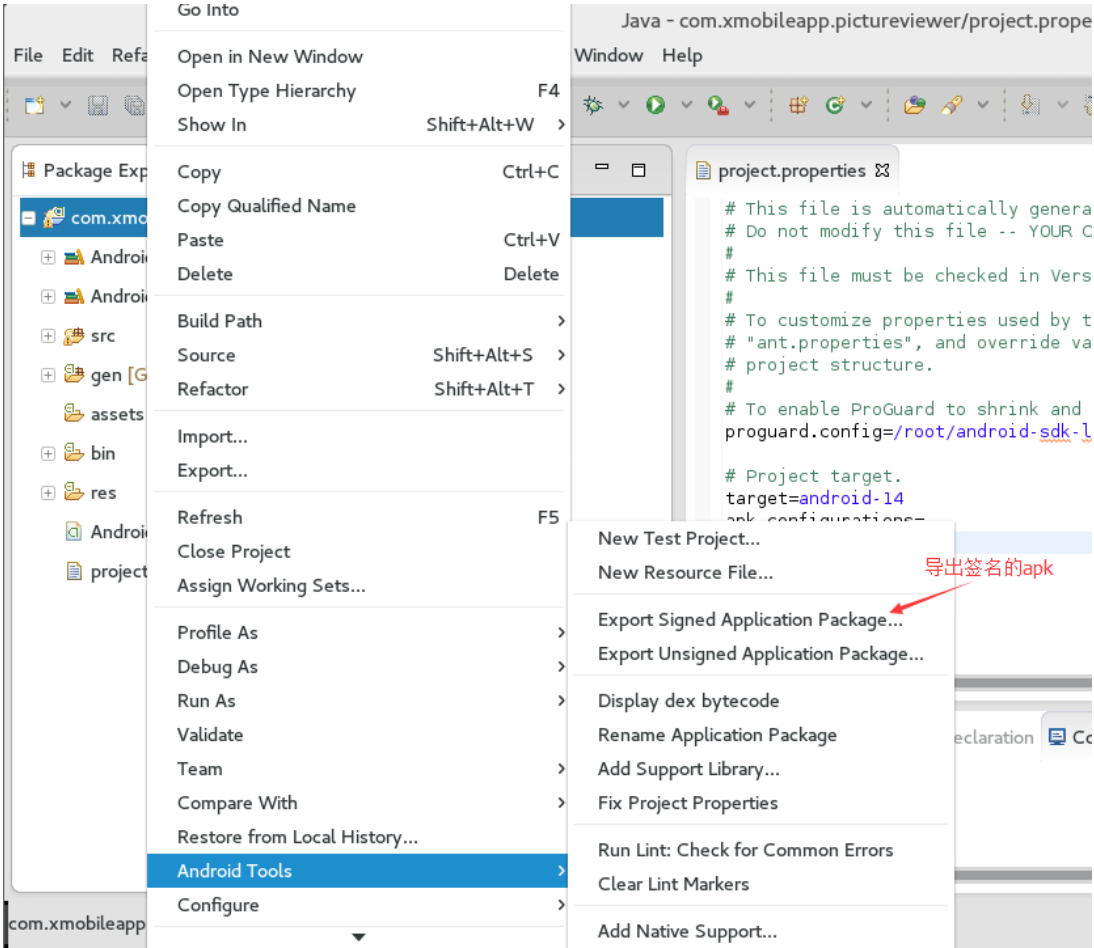


图 17导出为签名的apk

17、为apk创建一个新的签名。

操作：“Create new keystore”>“location:/root/bluedon”>“设置密码”>“单击next”如图 18

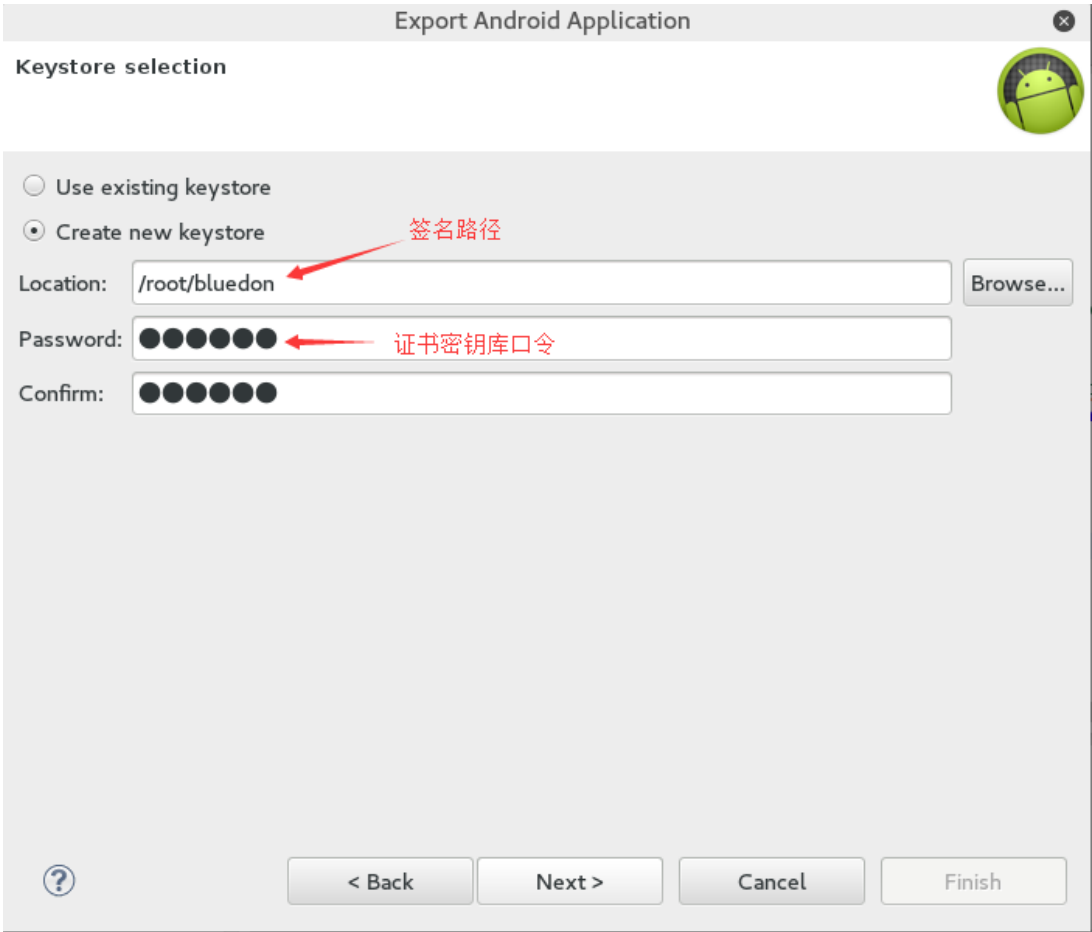


图 18创建签名

18、填写签名的内容。如图 19

Export Android Application

Key Creation

Alias:

bluedon

签名别名

Password:

●●●●●●

证书密钥口令

Confirm:

●●●●●●

Validity (years):

25

生存周期

First and Last Name:

bluedon

Organizational Unit:

bluedon

Organization:

bluedon

相关签名信息

City or Locality:

bluedon

State or Province:

bluedon

Country Code (XX):

bluedon

?

< Back

Next >

Cancel

Finish

图 19签名内容

19、设置输出目录为 “/root/pictureviewer.apk”如图 20

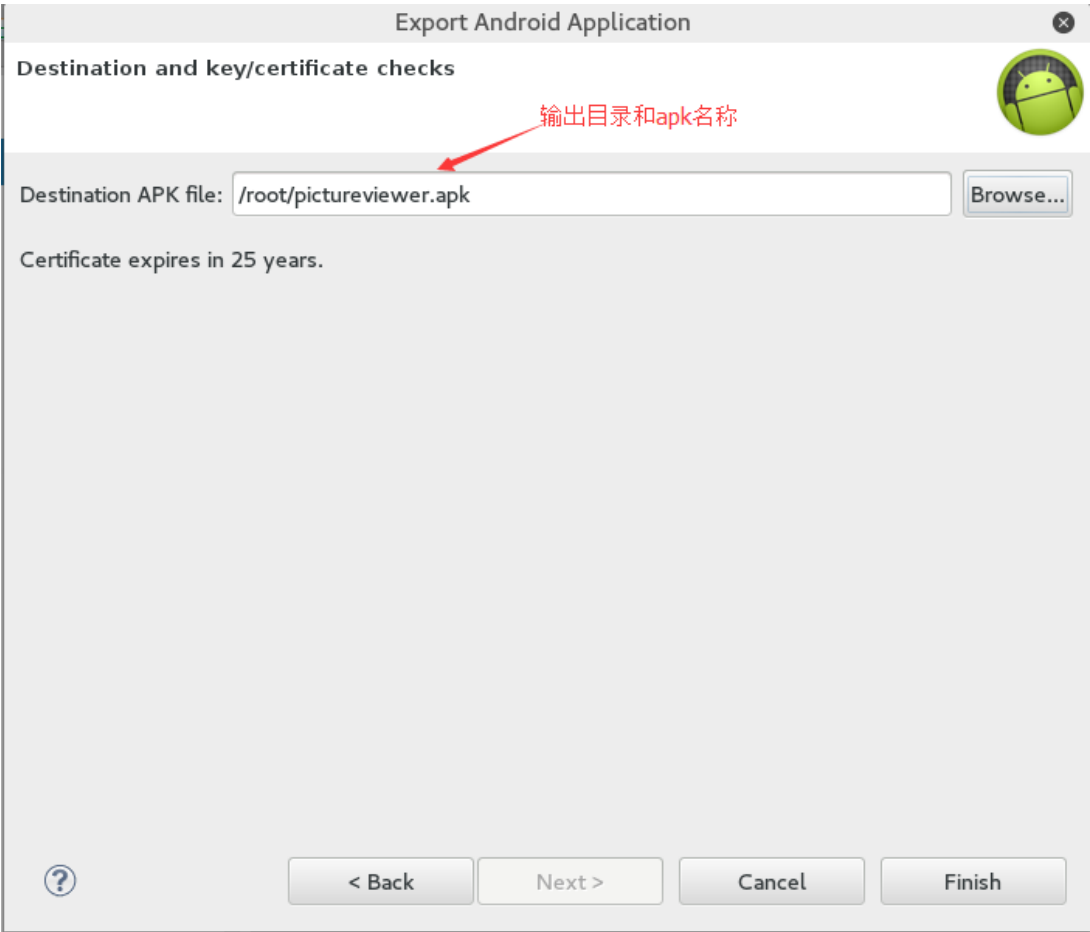


图 20输出目录和apk名称

20、查看输出的apk，并给其执行权限。

操作： “ls”>“chmod 770 pictureviewer.apk”>“ls -l”如图 21

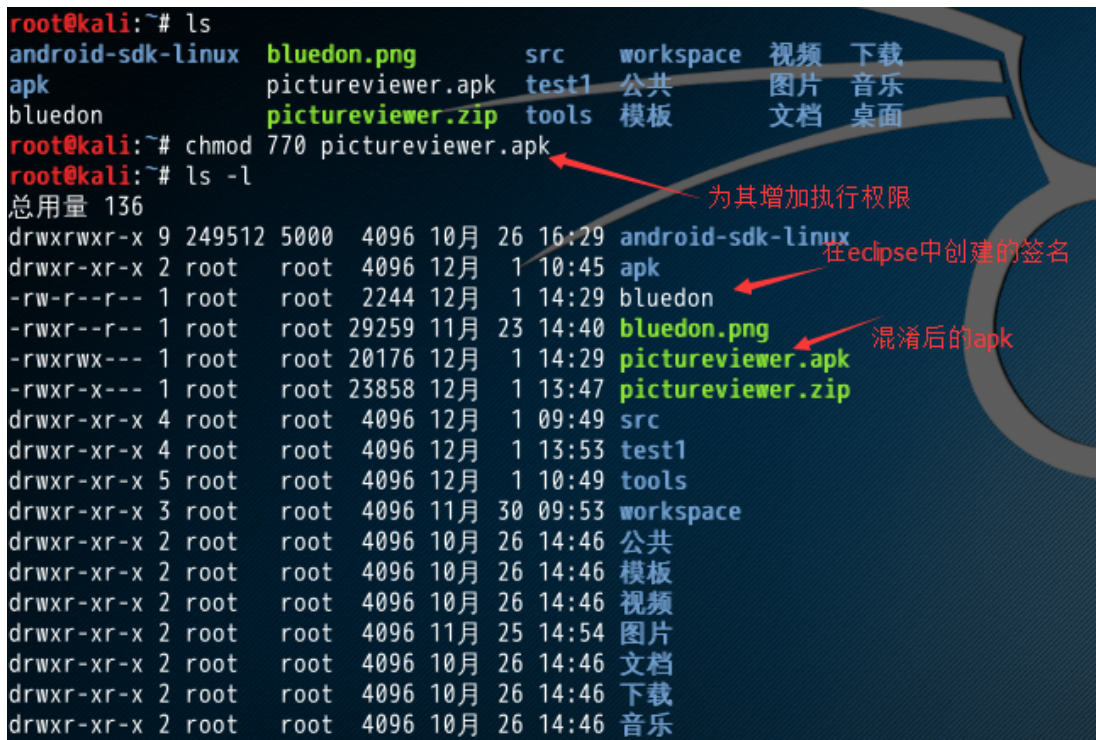


图 21赋予执行权限

21、删除未混淆的apk

操作： “rm pictureviewer.zip”>“ls”如图 22

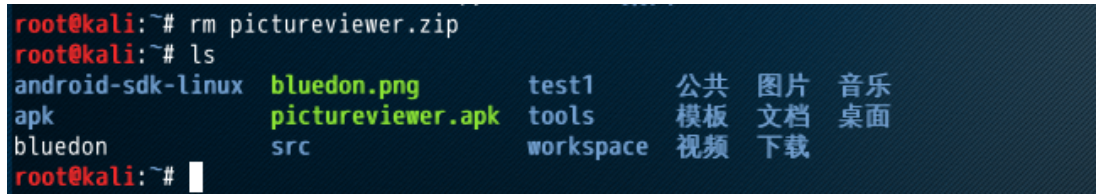


图 22删除未混淆的apk

22、解压缩刚导出的代码混淆后的apk。

操作： “mv pictureviewer.apk pictureviewer.zip”>“mkdir test2”>“unzip pictureviewer.zip -d ./test2”如图 23





```

root@kali:~# mv pictureviewer.apk pictureviewer.zip
root@kali:~# mkdir test2
root@kali:~# unzip pictureviewer.zip -d ./test2
Archive: pictureviewer.zip
  inflating: ./test2/AndroidManifest.xml
  inflating: ./test2/res/drawable/grid_background.xml
  extracting: ./test2/res/drawable/icon.png
  extracting: ./test2/res/drawable/imageborderfocus.png
  extracting: ./test2/res/drawable/imagebordernormal.png
  extracting: ./test2/res/drawable/imageborderpressed.png
  extracting: ./test2/res/drawable/noimage.png
  extracting: ./test2/res/drawable/nothumbnail.png
  inflating: ./test2/res/layout/main.xml
  extracting: ./test2/resources.arsc
  inflating: ./test2/classes.dex
  inflating: ./test2/META-INF/MANIFEST.MF
  inflating: ./test2/META-INF/CERT.SF
  inflating: ./test2/META-INF/CERT.RSA
root@kali:~#

```

图 23解压新的apk

23、把classes.dex文件转换为jar文件。

操作：“cd test2”>“d2j-dex2jar classes.dex”>“ls”如图 24



```

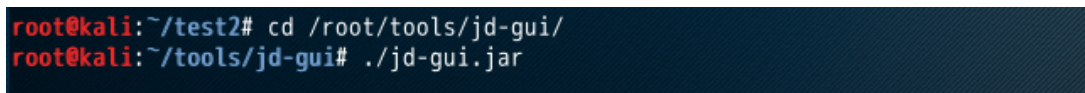
root@kali:~# cd test2
root@kali:~/test2# ls
AndroidManifest.xml  classes.dex  META-INF  res  resources.arsc
root@kali:~/test2# d2j-dex2jar classes.dex
dex2jar classes.dex -> classes-dex2jar.jar
root@kali:~/test2# ls
AndroidManifest.xml  classes-dex2jar.jar  res  resources.arsc
classes.dex          META-INF
root@kali:~/test2#

```

图 24classes.dex转换为jar文件

24、打开jd-gui工具。

操作：“cd /root/tools/jd-gui/”>“./jd-gui.jar”如图 25



```

root@kali:~/test2# cd /root/tools/jd-gui/
root@kali:~/tools/jd-gui# ./jd-gui.jar

```

图 25开启jd-gui

25、在jd-gui中打开反编译获得的jar文件。

操作：“单击左上角位置”>“选择主文件夹”>“选择test2”>“把classes-dex2jar.jar文件拖入jd中”如图 26

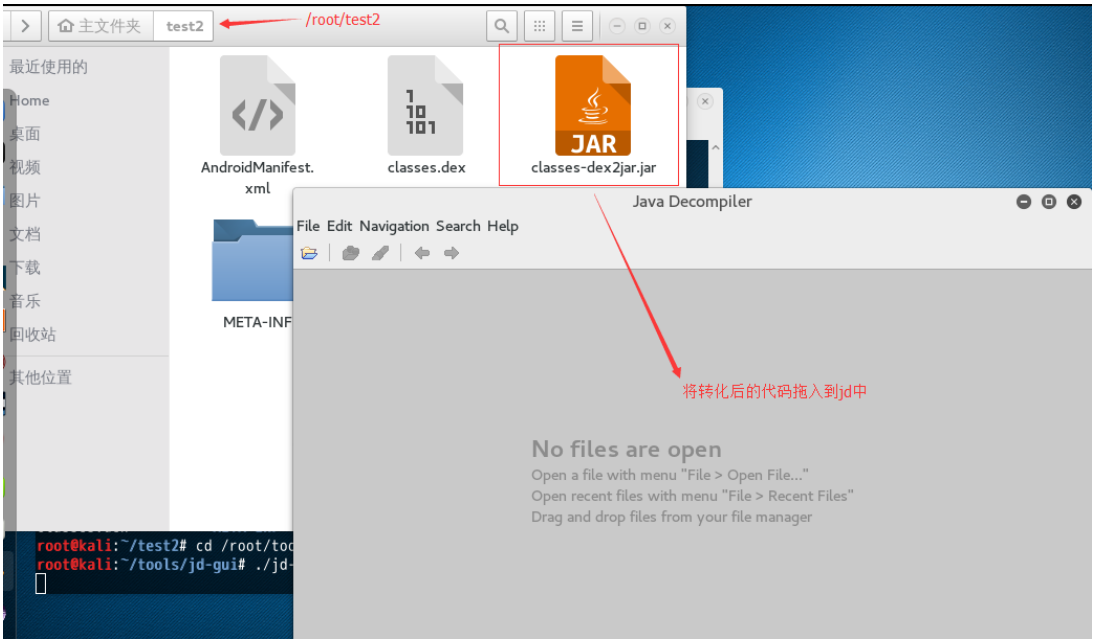


图 26打开jar文件

26、查看到混淆过的java类，如图 27

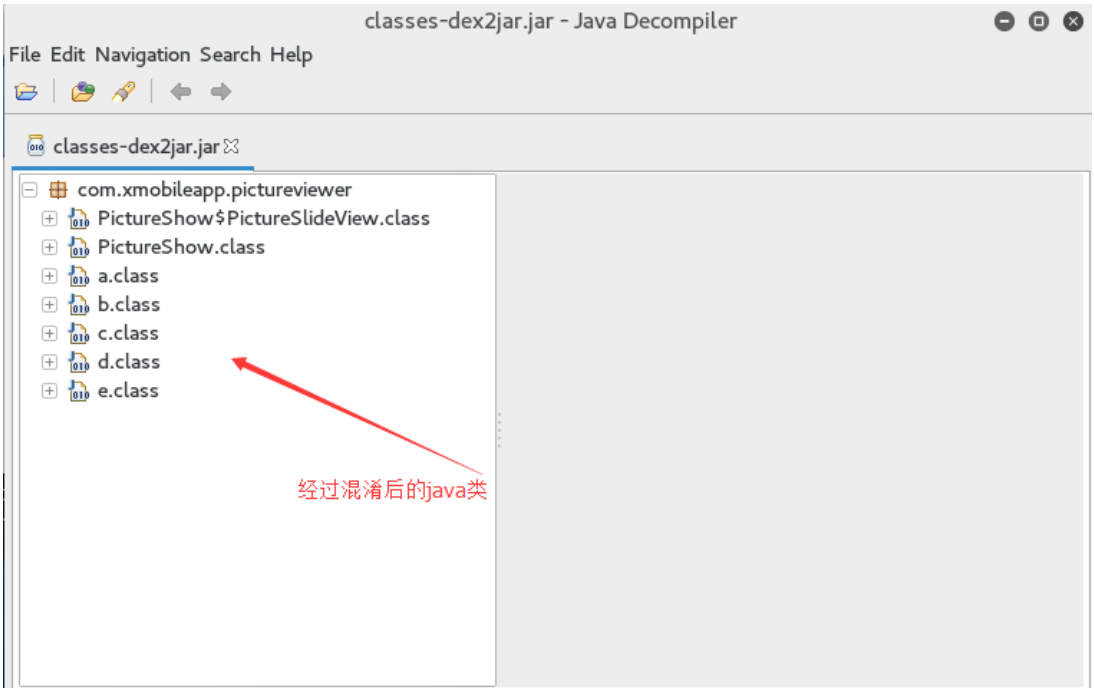


图 27混淆的java类

27、可查看到代码的变量也经过混淆，如图 28

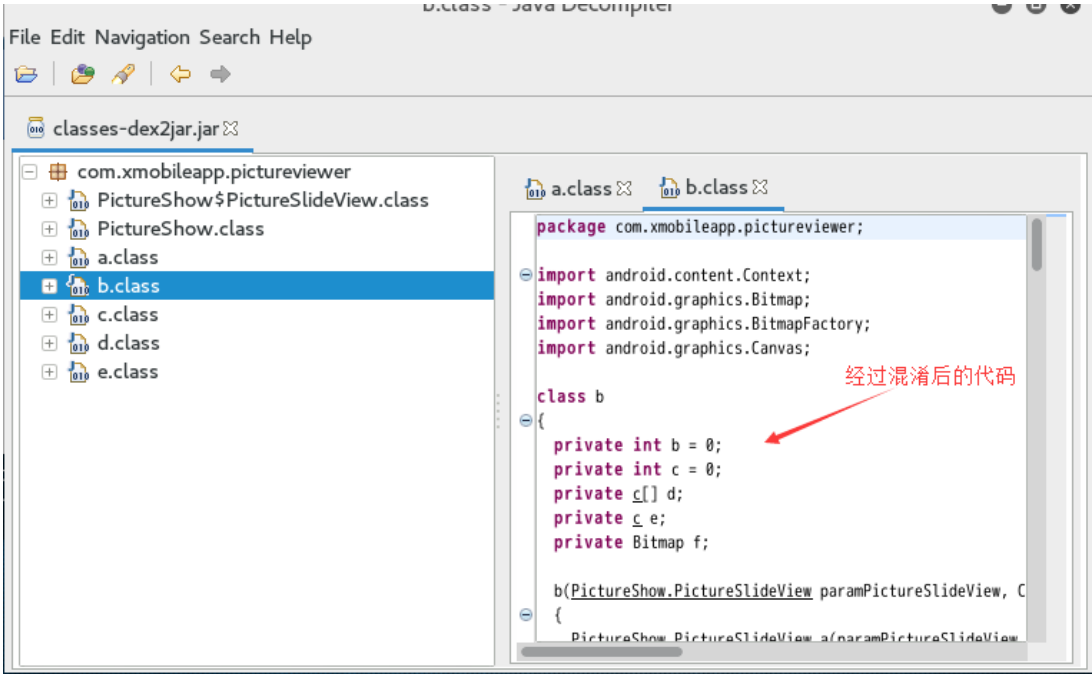


图 28混淆的java代码

### 思考总结

本实验对首先模拟黑客对pictureviewer.apk进行反编译查看其源代码，再通过对pictureviewer.apk的源码进行代码混淆获得新的pictureviewer.apk，然后通过模拟黑客对代码混淆后的apk反编译获得源代码。此时的代码的类名和变量都是经过a、b、c等字符混淆的，这样就增加了代码的阅读难度，可有效的防止apk被破解修改。

- 1、是否有办法可以选择混淆相关代码而不是全部代码？
- 2、还有什么方法可以防止apk被反编译？