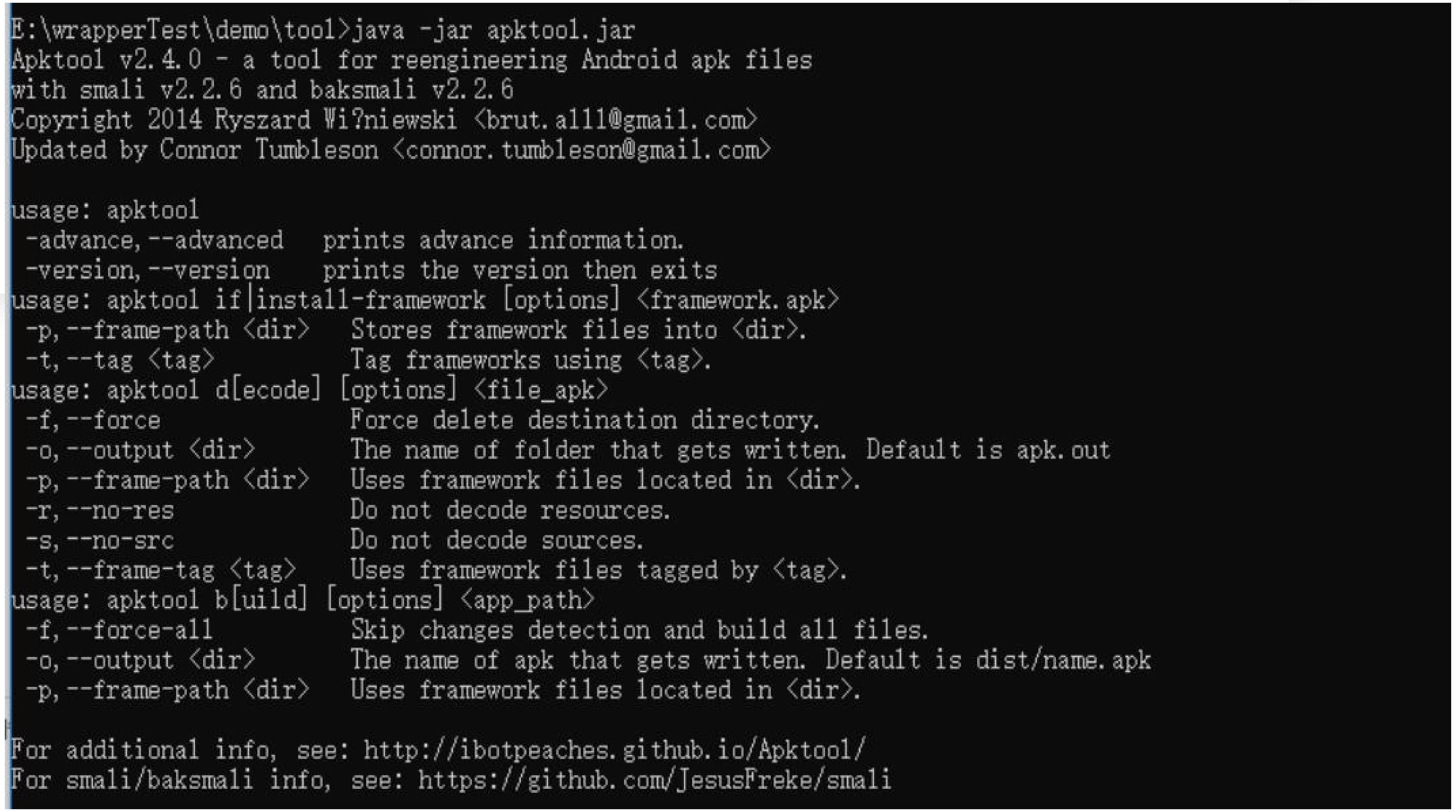
DEX加固与反编译

常见反编译工具

Apktool　　https://ibotpeaches.github.io/Apktool

- 反编译DEX为smali文件
- 反编译资源文件
- 支持重打包

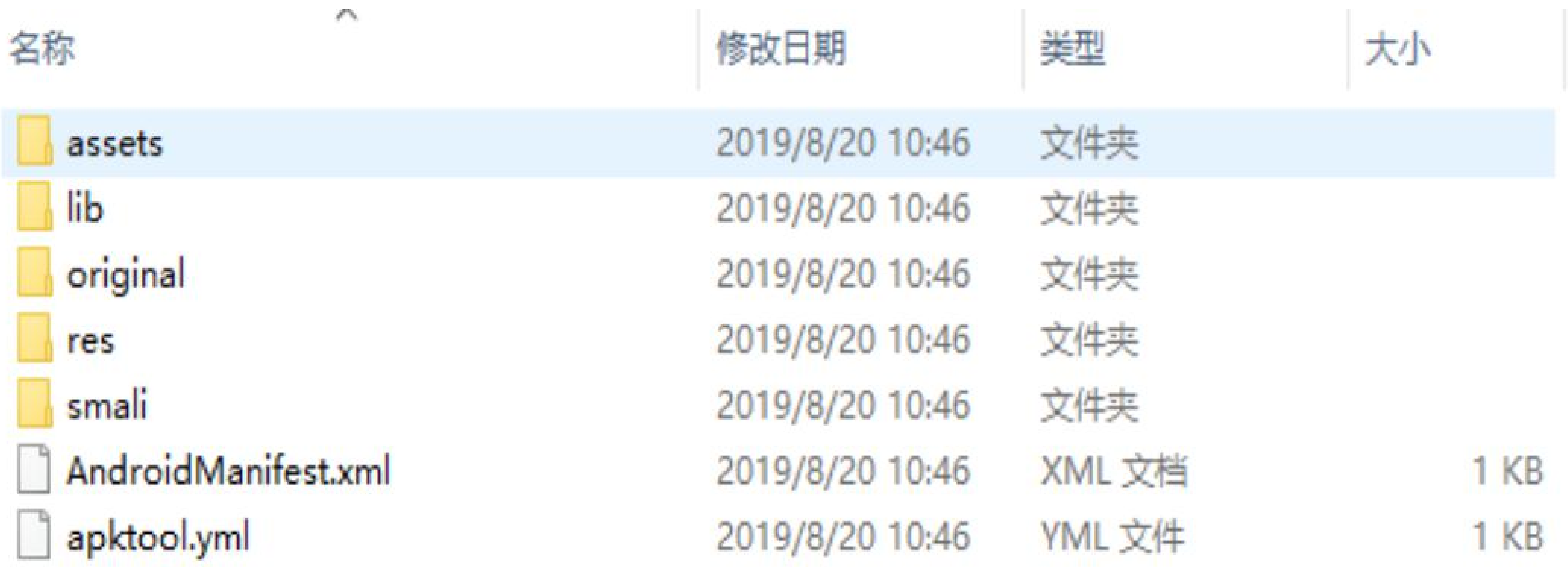


```
E:\wrapperTest\demo\tool>java -jar apktool.jar
Apktool v2.4.0 - a tool for reengineering Android apk files
with smali v2.2.6 and baksmali v2.2.6
Copyright 2014 Ryszard Wi?niewski <brut.alll@gmail.com>
Updated by Connor Tumbleson <connor.tumbleson@gmail.com>

usage: apktool
 -advance,--advanced   prints advance information.
 -version,--version    prints the version then exits
usage: apktool if|install-framework [options] <framework.apk>
 -p,--frame-path <dir>   Stores framework files into <dir>.
 -t,--tag <tag>          Tag frameworks using <tag>.
usage: apktool d[ecode] [options] <file_apk>
 -f,--force              Force delete destination directory.
 -o,--output <dir>       The name of folder that gets written. Default is apk.out
 -p,--frame-path <dir>   Uses framework files located in <dir>.
 -r,--no-res             Do not decode resources.
 -s,--no-src             Do not decode sources.
 -t,--frame-tag <tag>    Uses framework files tagged by <tag>.
usage: apktool b[uild] [options] <app_path>
 -f,--force-all          Skip changes detection and build all files.
 -o,--output <dir>       The name of apk that gets written. Default is dist/name.apk
 -p,--frame-path <dir>   Uses framework files located in <dir>.

For additional info, see: http://ibotpeaches.github.io/Apktool/
For smali/baksmali info, see: https://github.com/JesusFreke/smali
```

常见反编译工具

Apktool　https://ibotpeaches.github.io/Apktool

- java -jar apktool.jar d demo.apk

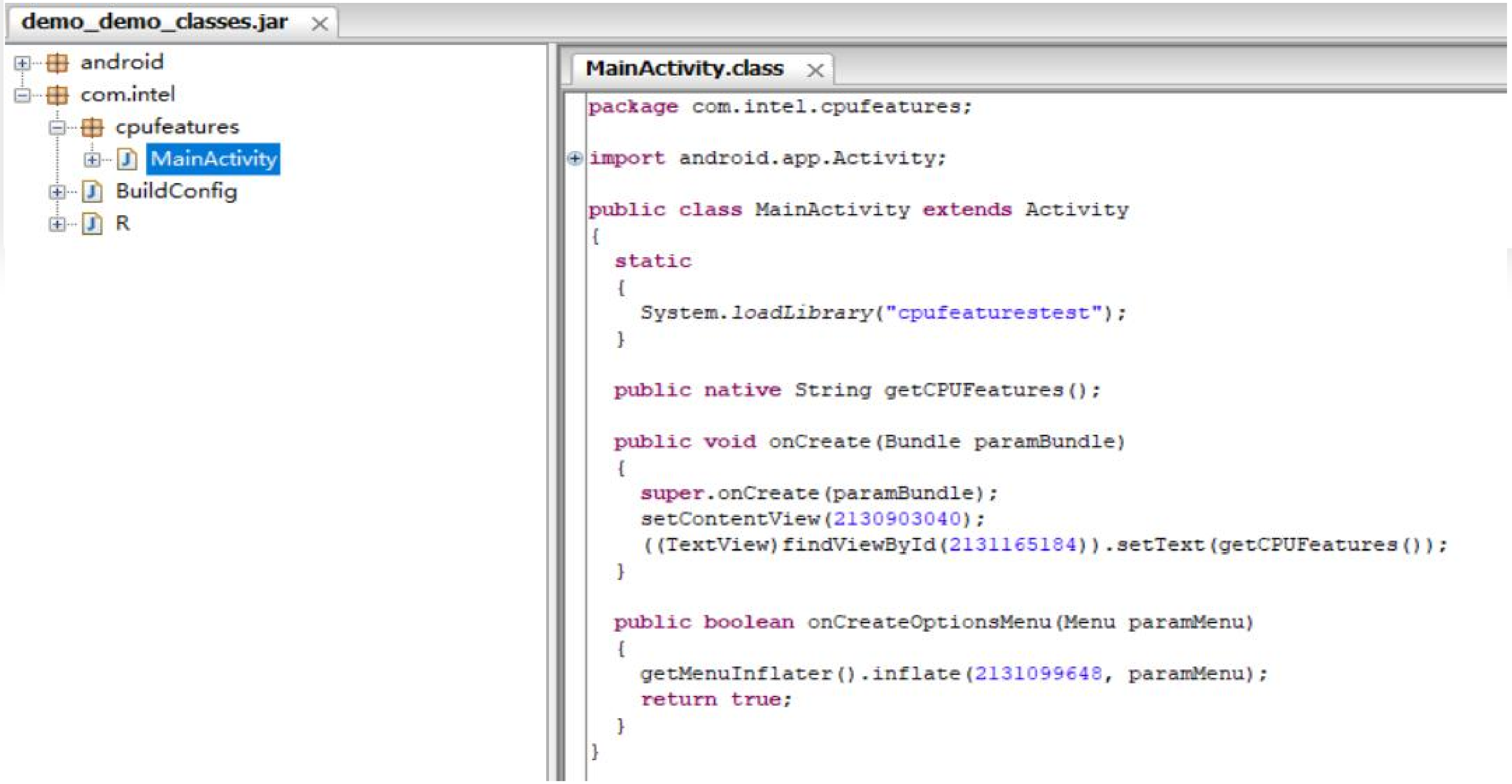


名称	修改日期	类型	大小
assets	2019/8/20 10:46	文件夹	
lib	2019/8/20 10:46	文件夹	
original	2019/8/20 10:46	文件夹	
res	2019/8/20 10:46	文件夹	
smali	2019/8/20 10:46	文件夹	
AndroidManifest.xml	2019/8/20 10:46	XML 文档	1 KB
apktool.yml	2019/8/20 10:46	YML 文件	1 KB

常见反编译工具

dex2jar   https://github.com/pxb1988/dex2jar

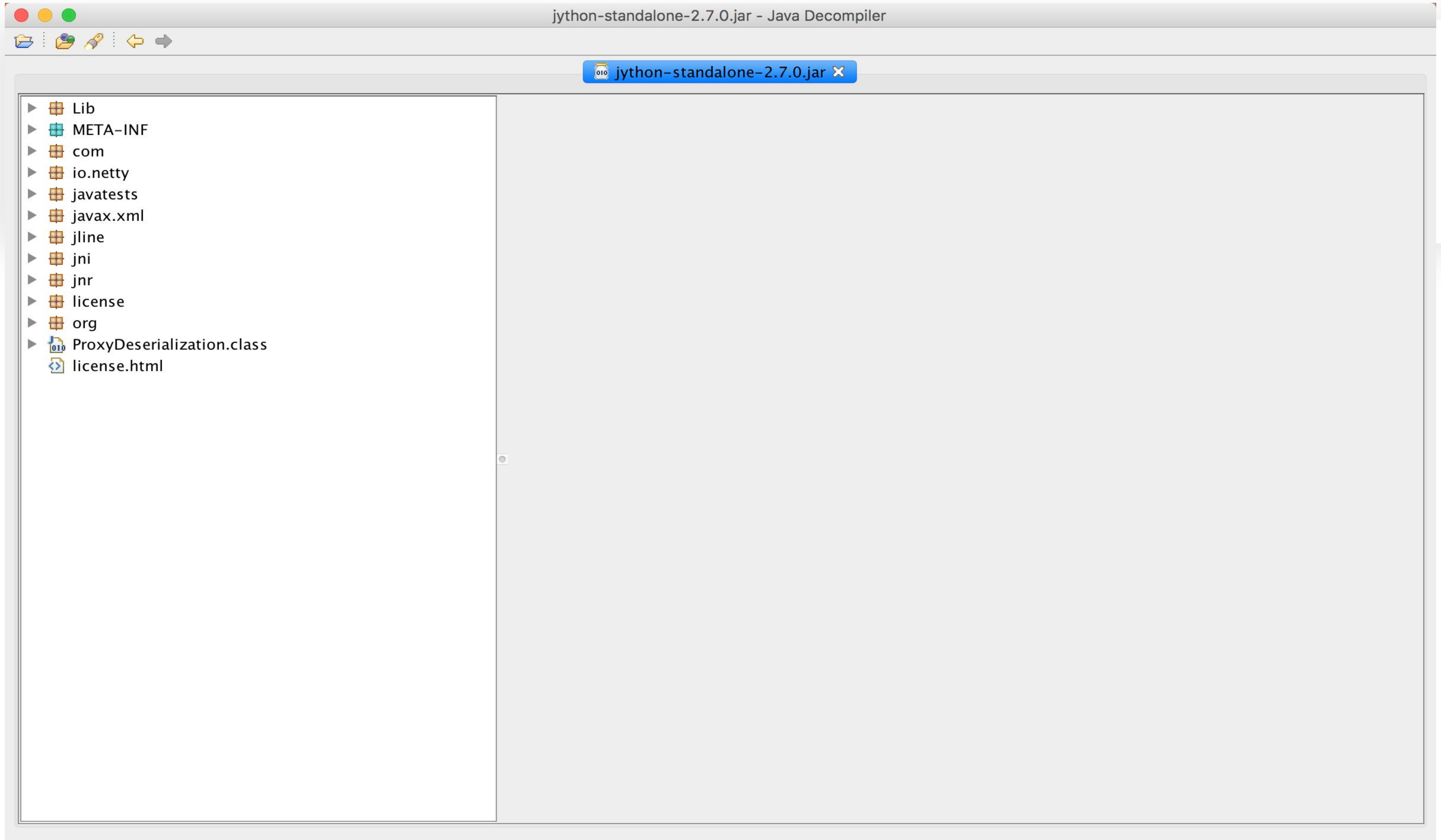
- d2j-dex2jar.bat demo.apk



```
package com.intel.cpufeatures;

import android.app.Activity;

public class MainActivity extends Activity
{
  static
  {
    System.loadLibrary("cpufeaturestest");
  }

  public native String getCPUFeatures();

  public void onCreate(Bundle paramBundle)
  {
    super.onCreate(paramBundle);
    setContentView(2130903040);
    ((TextView)findViewById(2131165184)).setText(getCPUFeatures());
  }

  public boolean onCreateOptionsMenu(Menu paramMenu)
  {
    getMenuInflater().inflate(2131099648, paramMenu);
    return true;
  }
}
```

常见反编译工具

JD-GUI   http://jd.benow.ca/



jython-standalone-2.7.0.jar - Java Decompiler

jython-standalone-2.7.0.jar

- Lib
- META-INF
- com
- io.netty
- javatests
- javax.xml
- jline
- jni
- jnr
- license
- org
- ProxyDeserialization.class
- license.html

修改入口

修改AndroidManifest.xml入口

```
//old AndroidManifest.xml

<application
android:name=".MyApplication"
android:icon="@drawable/icon"
android:label="@string/app_name">
```

```
//new AndroidManifest.xml

<application
android:name=".MyProxyApplication"
android:icon="@drawable/icon"
android:label="@string/app_name" >
```

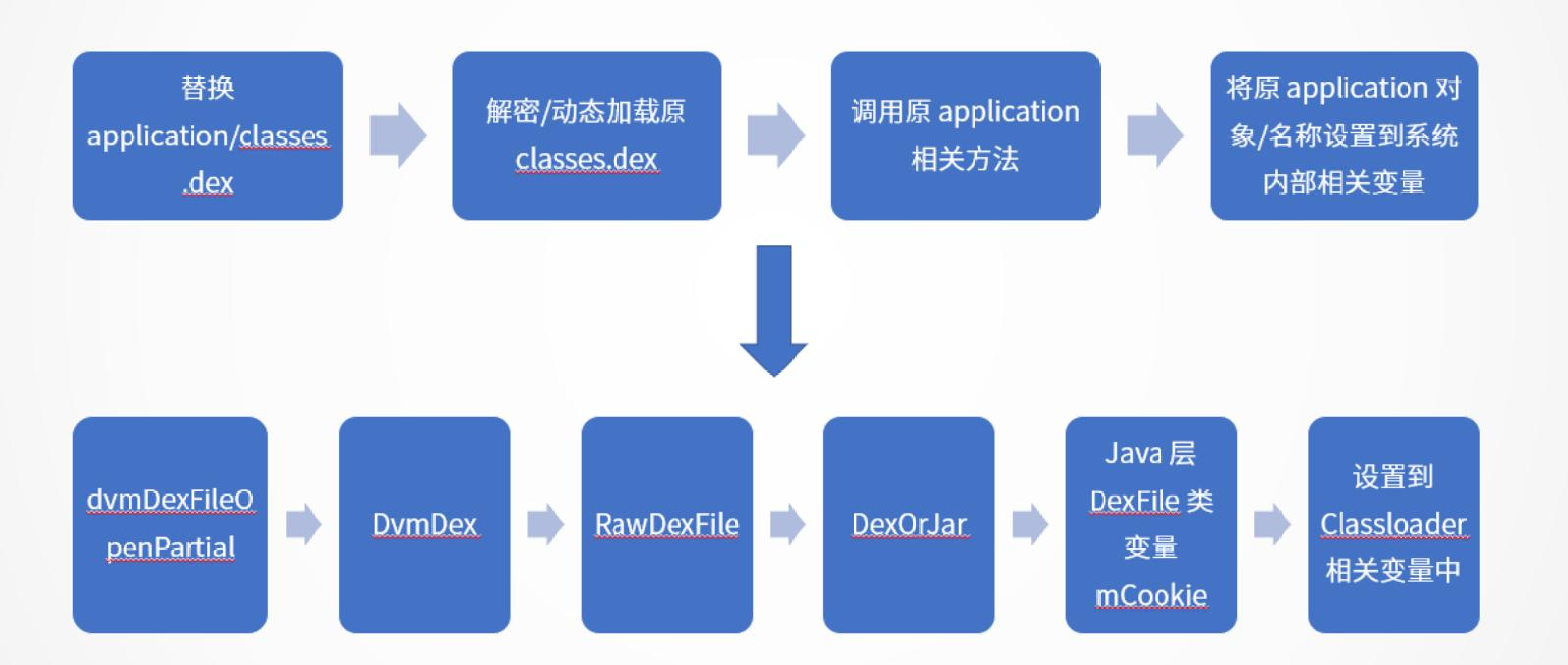
代理Application

```java
public abstract class ProxyApplication extends Application {
    protected abstract void initProxyApplication();
    @Override
    protected void attachBaseContext (Context context) {
        super.attachBaseContext(context);
        initProxyApplication();
    }
    // ……
}
```

initProxyApplication实现内容

- 内存加载DEX：加载原Application
- ClassLoader设置
- Application引用替换

壳启动流程



替换application/classes.dex

解密/动态加载原classes.dex

调用原 application 相关方法

将原 application 对象/名称设置到系统内部相关变量

dvmDexFileOpenPartial

DvmDex

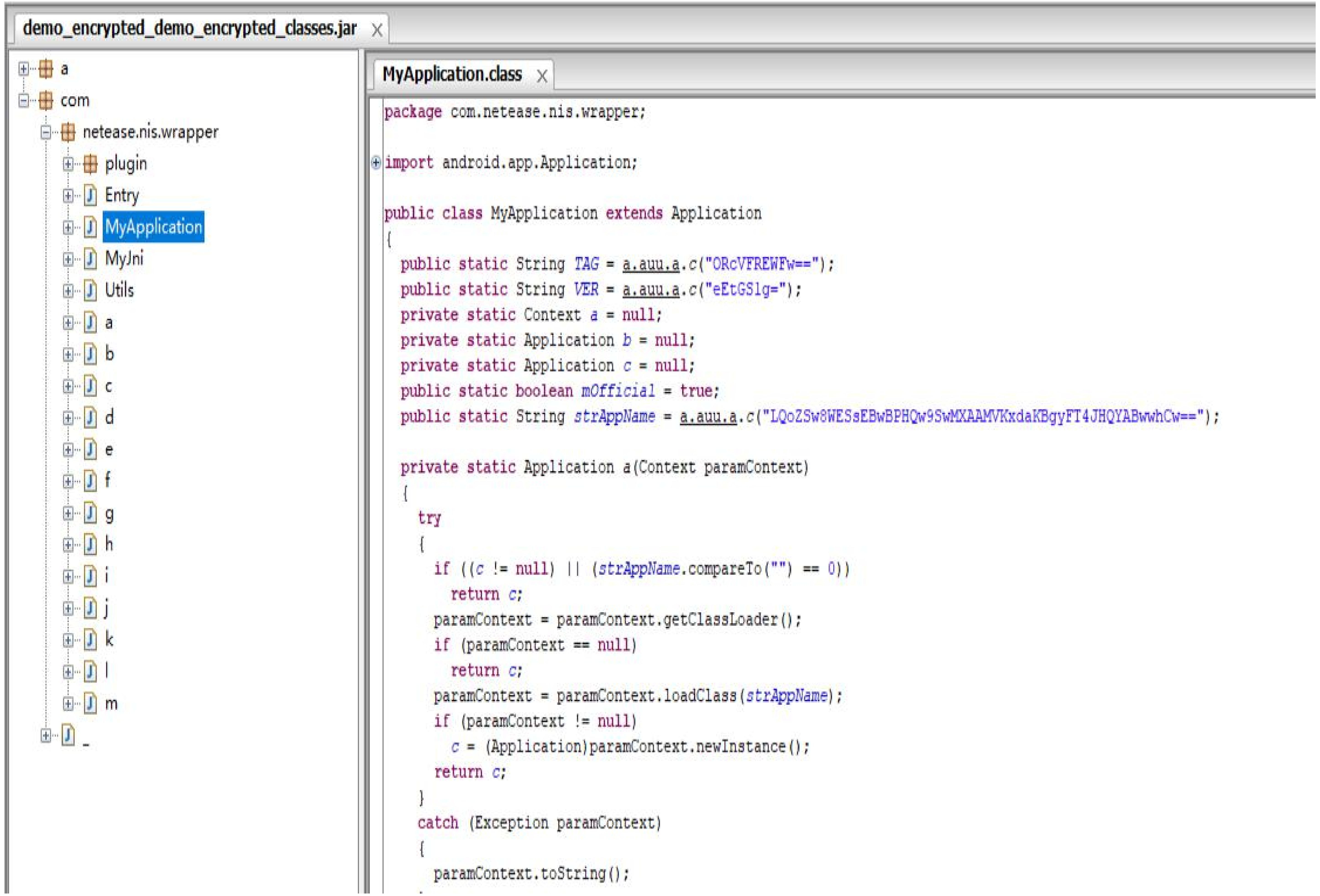
RawDexFile

DexOrJar

Java 层 DexFile 类变量 mCookie

设置到 Classloader 相关变量中

DEX加固效果（内存加载方案）



- a
- com
  - netease.nis.wrapper
    - plugin
    - Entry
    - MyApplication
    - MyJni
    - Utils
    - a
    - b
    - c
    - d
    - e
    - f
    - g
    - h
    - i
    - j
    - k
    - l
    - m
  - _

```java
package com.netease.nis.wrapper;

import android.app.Application;

public class MyApplication extends Application
{
  public static String TAG = a.auu.a.c("ORcVFREWFw==");
  public static String VER = a.auu.a.c("eEtGSlg=");
  private static Context a = null;
  private static Application b = null;
  private static Application c = null;
  public static boolean mOfficial = true;
  public static String strAppName = a.auu.a.c("LQoZSw8WESsEBwBPHQw9SwMXAAMVKxdaKBgyFT4JHQYABwwhCw==");

  private static Application a(Context paramContext)
  {
    try
    {
      if ((c != null) || (strAppName.compareTo("") == 0))
        return c;
      paramContext = paramContext.getClassLoader();
      if (paramContext == null)
        return c;
      paramContext = paramContext.loadClass(strAppName);
      if (paramContext != null)
        c = (Application)paramContext.newInstance();
      return c;
    }
    catch (Exception paramContext)
    {
      paramContext.toString();
```