目录

Android逆向：静态分析

- 最新版本：`v1.0.2`
- 更新时间：`20230914`

简介

介绍安卓逆向之静态分析。包括先是概览；然后是相关子项，包括apk文件，以及其中的AndroidMenifest.xml，以及反编译，包括反编译的典型流程，以及如何从apk破解出java源码；常见反编译工具，包括jad、JEB、GDA等；以及常见的二进制编辑器，比如010editor、EverEdit等；接着是综合类工具，包括AndroidKiller、Android-Crack-Tool For Mac、ByteCode Viewer、Android Decompiler、decompile-apk、Android逆向助手、Androguard等；接着是按层级去介绍安卓静态分析所涉及的内容；以及再按文件格式去分相关内容；包括针对apk的，查看apk信息的aapt、APK Analyzer、ClassyShark等，apk解包的apktool，apk转java的jadx、JEB、GDA等；针对dex的，如何砸壳导出dex文件，包括判断哪家加固方案，砸壳工具FDex2以及如何用FDex2砸壳导出dex、DumpDex、DexExtractor、drizzleDumper、InDroid、DexHunter、FART等；接着是反编译dex，包括dex转jar、dex转smali、dex转java等。其中dex转jar包括dex2jar以及用法、Enjarify、Dedexer，dex转smali包括baksmali、dex转java包括jadx、GDA；针对jar的，jar转java，包括各种java反编译器的对比，以及常用的java反编译器，包括Procyon/Luyten、CFR、JD-GUI、Krakatau、Fernflower、Dare、JAD等；针对so的，ELF文件格式，以及如何从ELF的so导出静态资源供分析，常用工具有readelf、objdump、razbin2；以及如何反编译查看代码逻辑，常用工具有IDA、Hopper、radare2、Ghidra等；以及涉及到的子领域，包括反代码混淆，以及相关系列的子教程。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

HonKit源码

- crifan/android_re_static_analysis: Android逆向：静态分析

如何使用此HonKit源码去生成发布为电子书

详见：crifan/honkit_template: demo how to use crifan honkit template and demo

在线浏览

- Android逆向：静态分析 book.crifan.org
- Android逆向：静态分析 crifan.github.io

离线下载阅读

- Android逆向：静态分析 PDF
- Android逆向：静态分析 ePub
- Android逆向：静态分析 Mobi

版权和用途说明

鸣谢

感谢我的老婆**陈雪**的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

其他

作者的其他电子书

本人 `crifan` 还写了其他 `150+` 本电子书教程，感兴趣可移步至：

crifan/crifan_ebook_readme: Crifan的电子书的使用说明

关于作者

关于作者更多介绍，详见：

关于CrifanLi李茂 – 在路上

Android逆向静态分析概览

- 安卓的逆向破解 静态分析
  - 按文件格式类型分
    - 针对 `apk`
      - 查看apk信息
        - `aapt`
      - 解包工具：输出 `dex` 、 `so` 、 `smali`
        - `apktool`
      - 反编译工具
        - 直接apk转java
          - `jadx`
          - `JEB`
          - `GDA`
    - 针对 `dex`
      - 各种导出dex的工具=砸壳工具=脱壳工具
        - `FDex2`
        - `DumpDex`
        - `DexExtractor`
      - 各种反编译dex的工具
        - dex转jar
          - `dex2jar`
        - dex转smali
          - `baksmali`
        - dex直接转java
          - `jadx`
          - `GDA`
    - 针对 `jar`
      - jar转java=各种反编译工具
        - `Procyon`
        - `CFR`
        - `JD-GUI`
    - 针对 `so` 库文件=（往往是ARM64架构）的ELF文件
      - 导出静态资源
        - `readelf`
        - `objdump`
        - `razbin2`
      - 反编译代码逻辑
        - `IDA`
        - `Hopper`
        - `Radare2`
        - `Ghidra`
  - 涉及子领域
    - 反代码混淆

按层级分

- 按层级分
  - Java 层
    - 主要是dex文件
    - Dex文件
      - 先要有(未加壳的)dex
        - 如果apk加壳：要先砸壳
          - 壳的类型：360加固保，腾讯乐固等
          - 砸壳工具：FDex2等
          - 得到：脱壳后的dex =未加壳的dex
        - 如果未加壳：Apk解压直接得到 未加壳的dex
          - apktool即可解压和反编译
            - 内部从解压apk得到dex
      - 继续处理：(未加壳的)dex
        - 反编译
          - Dex to java
            - 工具：jadx, JEB, GDA等
          - 先Dex to jar 再jar to Java
            - Dex to jar
              - 工具：dex2jar等
            - Jar to Java
              - 各种java反编译器
  - Native层
    - 主要是so库文件
      - （安卓中的）so文件主要是ELF格式
        - 查看ELF文件的信息
          - 工具：readelf、objdump、rabin2等
      - 反编译so库文件
        - 主要是ARM汇编）查看代码逻辑
          - 反汇编工具：IDA、Hopper、radare2、Ghidra等

按文件格式

- 安卓的逆向破解之静态分析 按文件格式类型分
  - 针对 `apk`
    - 查看apk信息
      - `aapt`
      - `APK Analyzer`
      - `ClassyShark`
    - 解包工具：输出 `dex` 、 `so` 、 `smali`
      - `apktool`
    - 反编译工具
      - 直接apk转java
        - `jadx`
        - `GDA`
  - 针对 `dex`
    - 各种导出dex的工具=砸壳工具=脱壳工具
      - `FDex2`
      - `DumpDex`
      - `DexExtractor`
    - 各种反编译dex的工具
      - dex转jar
        - `dex2jar`
      - dex转smali
        - `baksmali`
      - dex直接转java
        - `jadx`
        - `GDA`
  - 针对 `jar`
    - jar转java=各种反编译工具
      - `Procyon`
      - `CFR`
      - `JD-GUI`
  - 针对 `so` 库文件=（往往是ARM64架构）的ELF文件
    - 导出静态资源
      - `readelf`
      - `objdump`
      - `razbin2`
    - 反编译代码逻辑
      - `IDA`
      - `Hopper`
      - `Radare2`
      - `Ghidra`

针对apk

查看apk信息

aapt

APK Analyzer

- `Android Studio` 在 `v2.2` 之后自带 `APK Analyzer`
  - 另外还有个命令行版本：apkanalyzer
- 作用和功能
  - (重要)直观的看到apk中各个文件的大小(比如DEX，resource等等)
    - 我们可根据文件大小信息，减小apk的大小
  - (重要)学习大企业app的命名规范和目录架构规范，还可以查看大公司app使用了什么技术和第三方框架
  - 了解DEX文件的组成
  - 快速查看APK的版本信息（例如androidmanifest.xml等也可以看到）
  - 直接比较两个APK的信息，有对比才有伤害
- 下载
  - Android Studio
    - Download Android Studio and SDK tools
- 截图
  -

-
-
-

- 。

ClassyShark

- 主页
  - google/android-classyshark: Binary analysis of any Android/Java based app/APK/game
- 功能
  - 单独的二进制程序
    - 用来浏览和查看apk信息
      - 类的接口和成员
      - dex个数和依赖
    - 支持输入格式
      - 库
        - `dex`
        - `aar`
        - `so`
      - 可执行文件
        - `apk`
        - `jar`
        - `class`
      - 安卓二进制XML
        - `AndroidManifest`
        - 资源文件
        - 布局文件
- 截图
  -

-
-

-
-

-

apk解包

apktool

apk转java

前提

- apk没有被加固
  - 怎么看出是否加固?
    - 如果你用jadx打开看到的和导出的代码和目录结构
      - 有 qihoo奇虎360 tencent腾讯乐固legu 等字样的
        - 那么说明就是被加固了的
          - 用的是 360加固保 腾讯乐固legu 的加固方案
          - 这样导出的代码也只是加固后的代码
            - 不是app的自己的业务逻辑的java代码
      - 没有 qihoo奇虎360 tencent腾讯乐固legu 等字样的
        - 那说明没有加固
          - jadx等工具，可以直接转换

直接从apk反编译得到java代码，可以用的工具有：

- `jadx`
- `JEB`
- `GDA`

jadx

详见：jadx

JEB

GDA

详见：GDA

针对dex

砸壳导出dex

- `去壳` = `砸壳` = `脱壳` = `去掉加固的壳`
  - 名词来源
    - 安卓的app的加固手段之一是：加壳
      - 对应的破解加壳所以叫：`去壳` = `脱壳` = `反加固`
  - 涉及内容
    - 主要涉及到的是dex文件
      - 所以主要内容是：砸壳导出dex文件

脱壳机制原理

- `smali` 层：只做了一些简单的混淆
- `native` 层：
  - 和如下内容相关
    - 各种so库
      - 比如 `libdvm.so`
    - 对应着内部函数调用
      - `.init`
      - `.init_array`
      - `JNI_Onload`
    - 分析修改ELF头信息
    - `sub_xxx` 函数
      - 比如：`sub_78614CD0`
    - R2寄存器
    - 最后分析出：
      - `ClassLoader`
      - `loadDex`
      - `multidex`
  - （作者）QEver
    - 写的一个IDA的脚本=一个dex脱壳脚本
    - 配合 `kill` 方法，可以实现脱绝大部分运行于 `dalvik` 上的 `dex` 壳
      - 可以dump导出正确的dex文件

脱壳注意事项和说明

不是所有的加固的安卓apk都能成功脱壳的。

比如，康美通的安卓apk：

- 老版本 `v2.0.7`：没有加固，可以直接用 `Jadx` 反编译得到源码的
- 新版本 `v4.4.0`：是 `360加固保` 加固的，用FDex2也无法导出 `dex`，无法破解

总结出来就是：

- `没有加固` 的：直接用Jadx即可导出源码
  - 比如老旧的Android的apk，很多都没有加固
    - 不管你怎么 `混淆` 都很容易被人分析得干干净净
- 部分加密不强的：可以脱壳
  - 包括
    - **老一代**或**免费**的 `360加固保`
    - 爱加密（收费）

        - 娜迦加固（收费）
    - 用 `FDex2` 可以脱壳
        - 可以hook导出 `dex` ，再dex转 `jar` ，jar转 `java` 源码
- `腾讯乐固` ，**新一代**的 `360加固保` ：没法脱壳
    - 即使用FDex2也无法脱壳无法破解，无法得到dex文件
        - 其中新一代的360加固保：用 `art` 模式+ `dex2oat` 相关机制，或许可以破解
            - 后续研究
                - 【未解决】用ART，oat，dex2oat相关机制去破解新一代360、腾讯等安卓apk的加固

对于使用加固方案的建议和结论

- 免费版的加固可以防止大多数只会反编译的小白
    - 对于普通攻击者还是很有效果的
    - 对于会用工具脱壳的，还是没太大用途的
- 除非用更加高级的，收费版的加固服务
    - 估计就很难破解，很难脱壳了
- 如果真的想要彻底防止别人破解
    - 除了考虑（用更高级的）加固方式
    - 还要花精力在app的业务逻辑层面，权限校验等方面，防止被破解

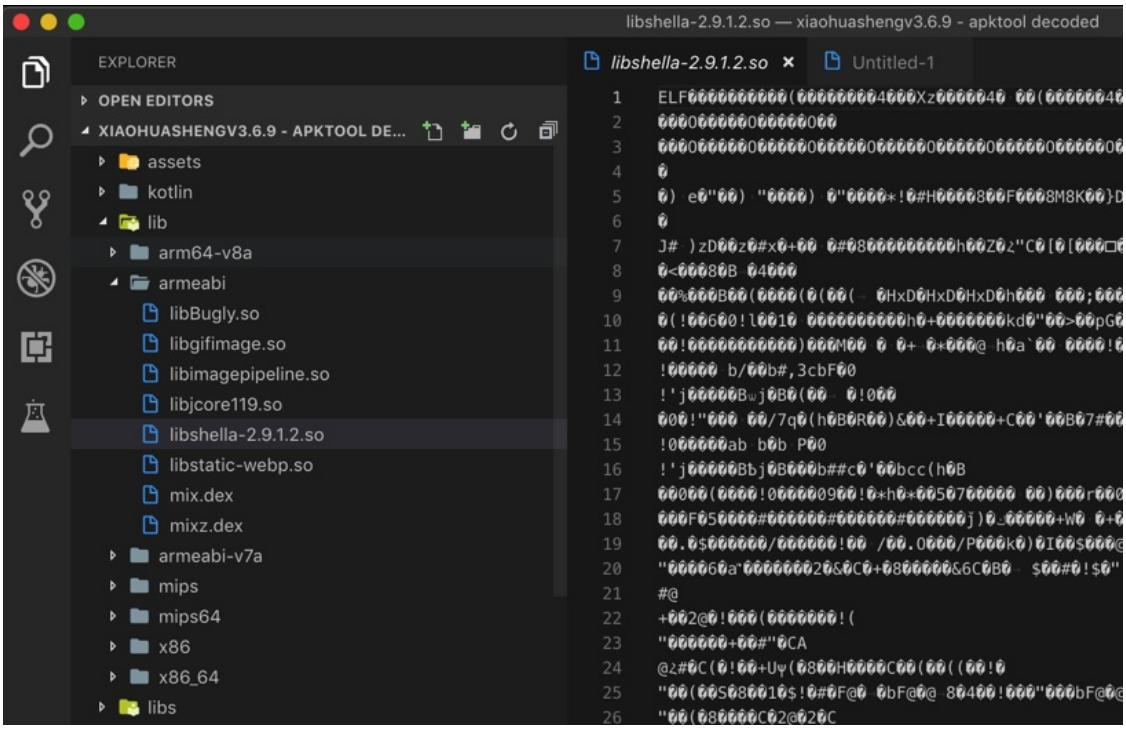
如何判断哪家加固方案

通过反编译工具后，从 `dex` 或 `jar` 包的目录结构，以及相关的文件（比如 `AndroidManifest.xml` ）的内容，往往可以看出是哪家的加密方案：

腾讯乐固加密后的目录结构和典型内容

腾讯的 `乐固legu` 加密加壳后的apk，（去用 `apktool` ）反编译后得到的 `jar` 包的典型目录结构是：

- 图

- 

- 文字
  - 核心文件夹
    - com
      - tencent
        - bugly
          - legu
            - crashreport

        - proguard
    - StubShell
        - TxAppEntry
- 详细的目录结构和文件

```
➜  tencent ll
total 0
drwxr-xr-x  12 crifan  staff   384B  3 14 13:39 StubShell
drwxr-xr-x   3 crifan  staff    96B  3 14 13:39 bugly
➜  tencent tree .
.
├── StubShell
│   ├── SystemClassLoaderInjector.smali
│   ├── SystemInfoException.smali
│   ├── TxAppEntry.smali
│   ├── TxReceiver.smali
│   ├── XposedCheck.smali
│   ├── ZipUtil.smali
│   ├── a.smali
│   ├── b.smali
│   ├── c.smali
│   └── d.smali
└── bugly
    └── legu
        ├── Bugly.smali
        ├── BuglyStrategy$a.smali
        ├── BuglyStrategy.smali
        ├── CrashModule.smali
        ├── a.smali
        ├── b.smali
        ├── crashreport
        │   ├── BuglyHintException.smali
        │   ├── BuglyLog.smali
        │   ├── CrashReport$CrashHandleCallback.smali
        │   ├── CrashReport$UserStrategy.smali
        │   ├── CrashReport.smali
        │   ...
        │   └── inner
        │       └── InnerAPI.smali
        └── proguard
            ├── a.smali
            ├── ...
            └── z.smali
14 directories, 123 files
➜  lib tree .
.
├── arm64-v8a
│   ├── libBugly.so
│   ├── libgifimage.so
│   ├── libimagepipeline.so
│   ├── libjcore119.so
│   ├── libshella-2.9.1.2.so
│   └── libstatic-webp.so
├── armeabi
│   ├── libBugly.so
│   ├── libgifimage.so
│   ├── libimagepipeline.so
│   ├── libjcore119.so
│   ├── libshella-2.9.1.2.so
│   ├── libstatic-webp.so
│   ├── mix.dex
│   └── mixz.dex
├── armeabi-v7a
│   ├── libBugly.so
│   ├── libgifimage.so
│   ├── libimagepipeline.so
│   ├── libjcore119.so
│   ├── libshella-2.9.1.2.so
│   └── libstatic-webp.so
├── ...
7 directories, 36 files
```

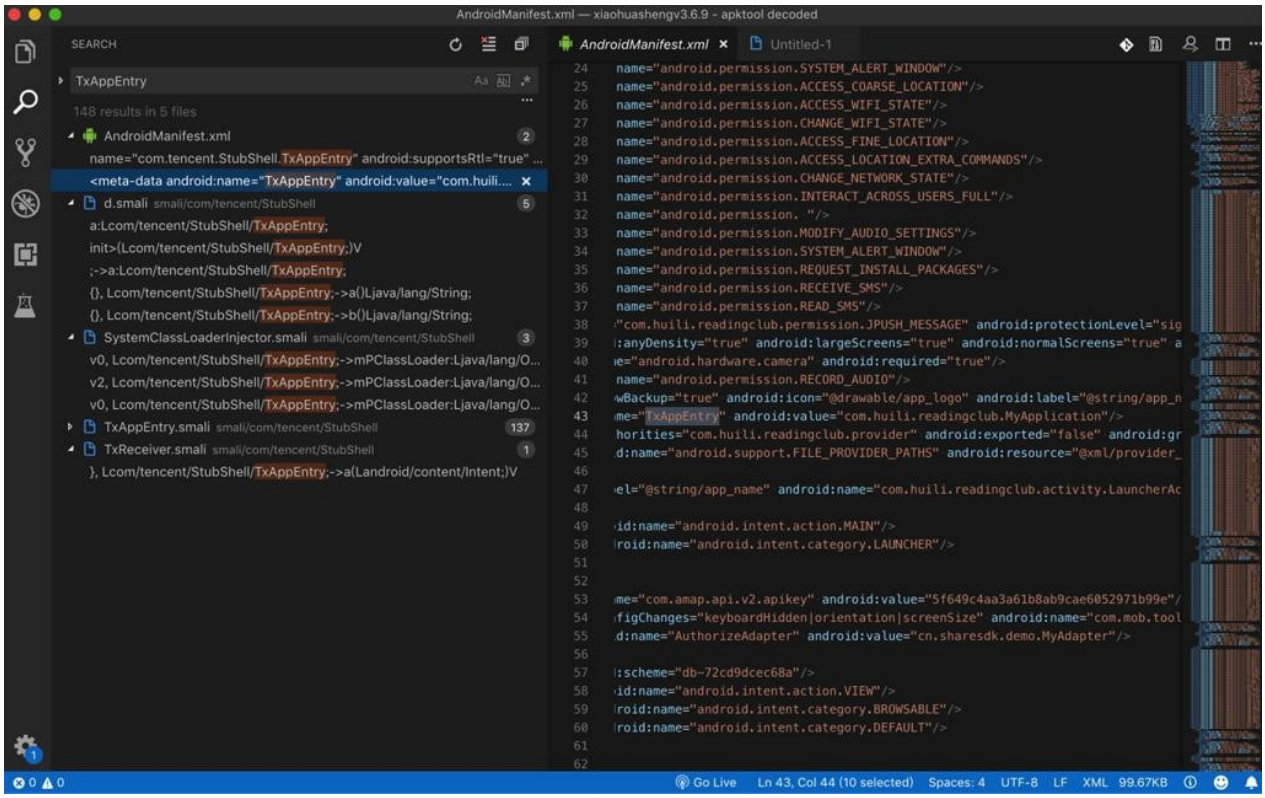
反编译出的 `AndroidManifest.xml` 内容：

```
<application android:allowBackup="true" android:icon="@drawable/app_logo" android:label="@string/app_name" android:name="com.tencent.StubShell.TxAppEntry" android:supportsRtl="true" android:theme="@style/AppTheme">
<meta-data android:name="TxAppEntry" android:value="com.huili.readingclub.MyApplication"/>
```

中有：

- `android:name="com.tencent.StubShell.TxAppEntry"`
  - 其中有：
    - `com.tencent.StubShell.TxAppEntry`
- `<meta-data android:name="TxAppEntry"`

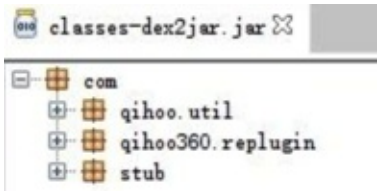
以及，多处都可以搜到：`TxAppEntry`

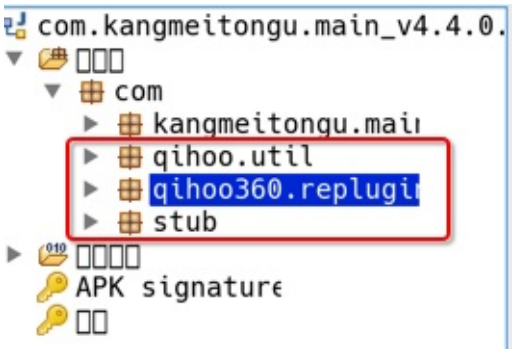


都是典型的腾讯乐固的相关内容。

360加固保的加固的目录结构

360加固后的 `apk` 经过 `dex2jar` 反编译后的目录结构是：





- `com.qihoo.util`
- `com.qihoo360.replugin`
- `com.stub`

这种结构就说明是360加固保加固的。

砸壳工具

此处整理常见的：

- 砸壳导出dex文件 == 作为XPosed辅助插件，从 `运行中的安卓app` 导出dex

的砸壳工具=hook工具

从app导出dex的文件的命名

折腾 `FDex2` 等工具，从运行中的app导出 `dex` 文件时，看到很多文件名都是类似于

`com.huili.readingclub8825612.dex`

对这类文件名，并没注意到有何特别。

后来从别人的帖子中，突然意识到：

这些hook工具的生成dex的代码中，对应的命名规则应该是：

`packageName` + `fileSize` .dex

所以此处的

`com.huili.readingclub8825612.dex`

对应着就是：

- `packageName` ： `com.huili.readingclub`
- `fileSize` ： `8825612`
  - = `8.4MB`
    - 就是dex文件本身的大小

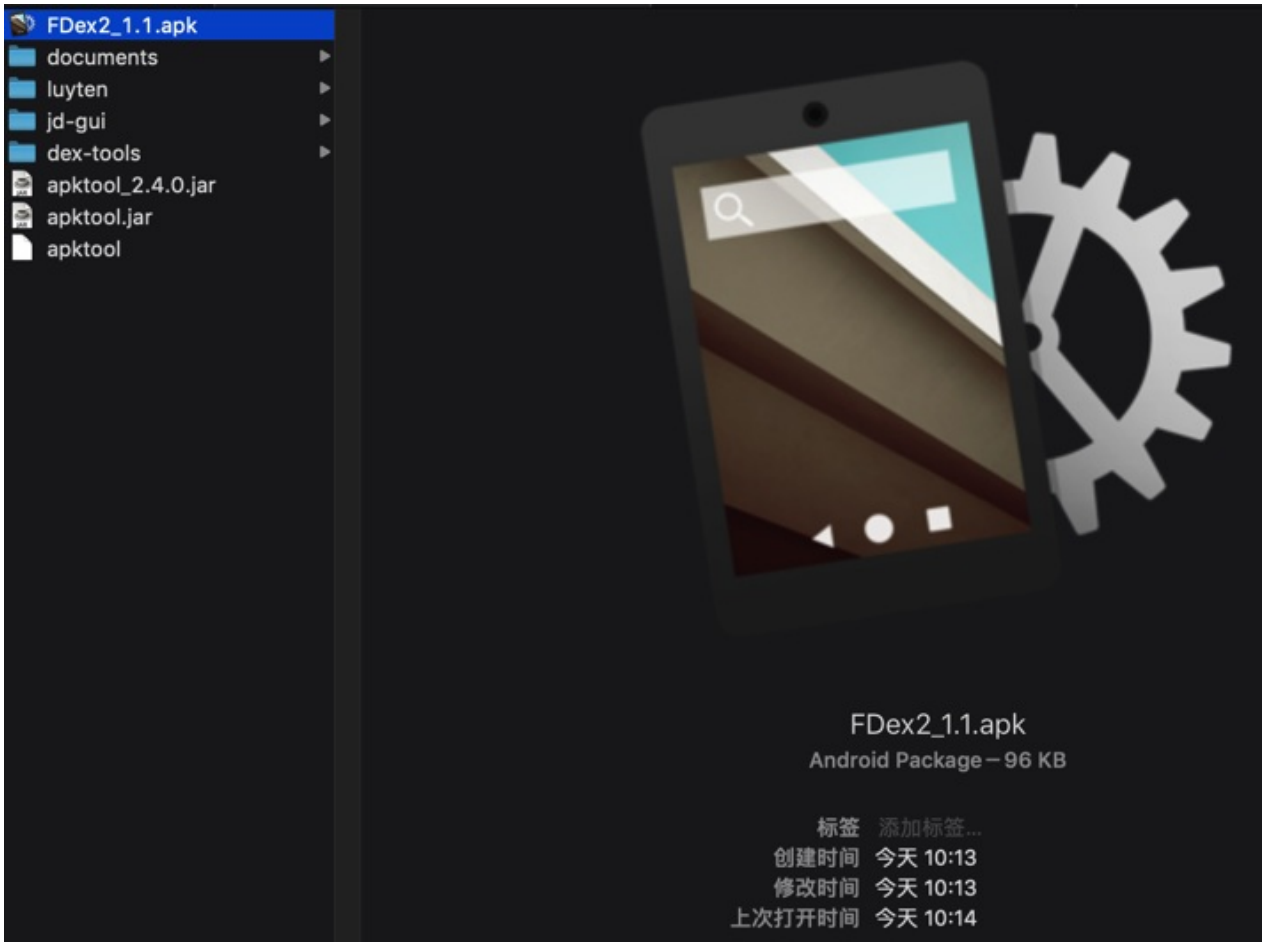
FDex2

- `FDex2`
  - 是什么：XPosed的一个插件
  - 功能：用来从 `运行中的安卓app` 中导出 `dex` 文件
  - 下载
    - 百度网盘
      - 链接: https://pan.baidu.com/s/1ITF8CN96bxWpFwv7J174lg 提取码: 3e3t
    - CSDN
      - 脱壳工具 FDex2-CSDN下载

如何使用FDex2

安装FDex2（这个安卓apk）

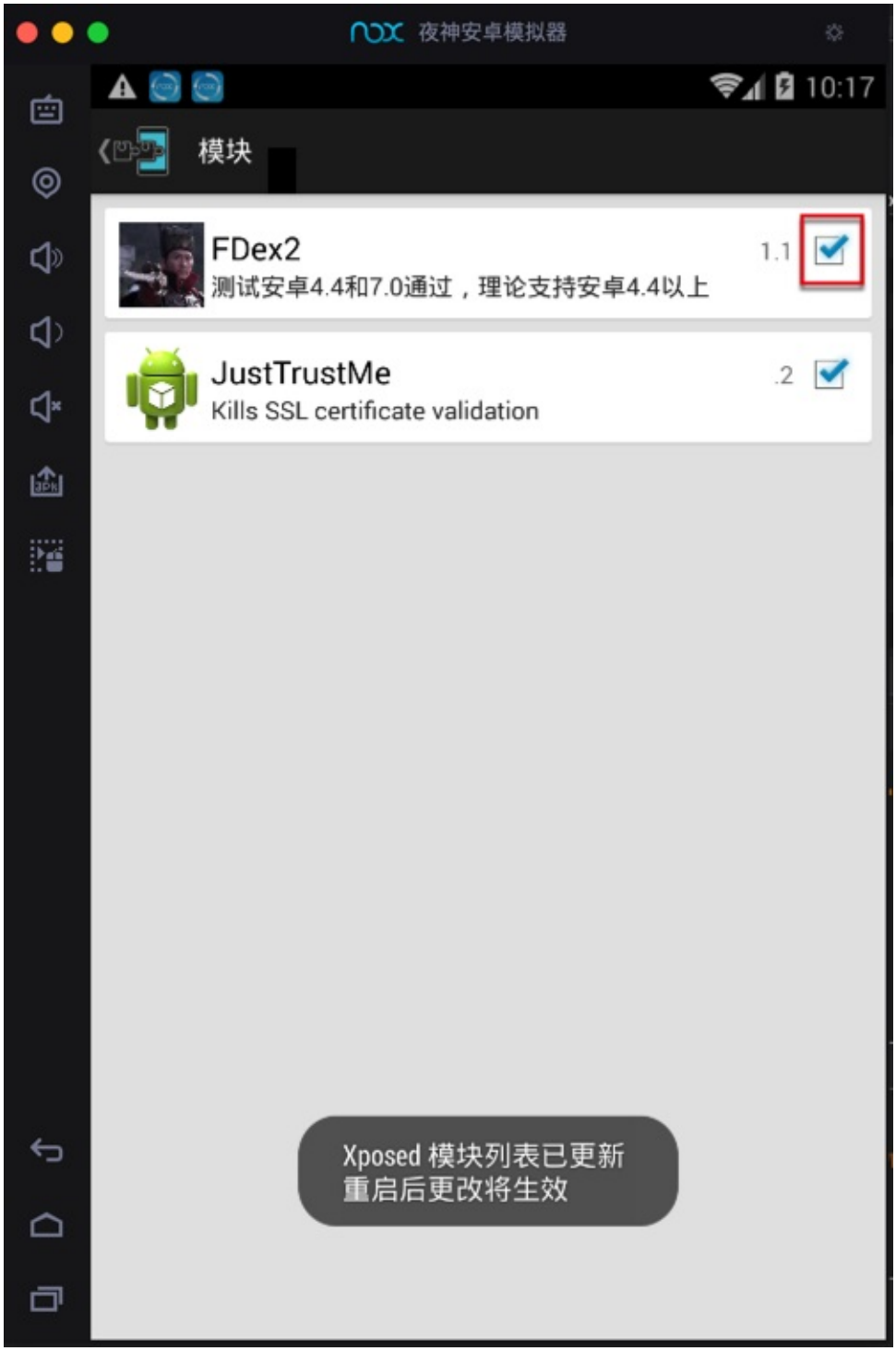
下载得到 `96KB` 的apk文件：`FDex2_1.1.apk`



先安装到安卓设备中，比如此处的Nox夜神模拟器：

Xposed中激活FDex2

再去打开 `Xposed` ，勾选=激活 `FDex2` ：



- 注意
  - 其会提示：`XPosed模块列表已更新，重启后更改将生效`
  - 所以为了使`FDex2生效`，记得去重启`XPosed`

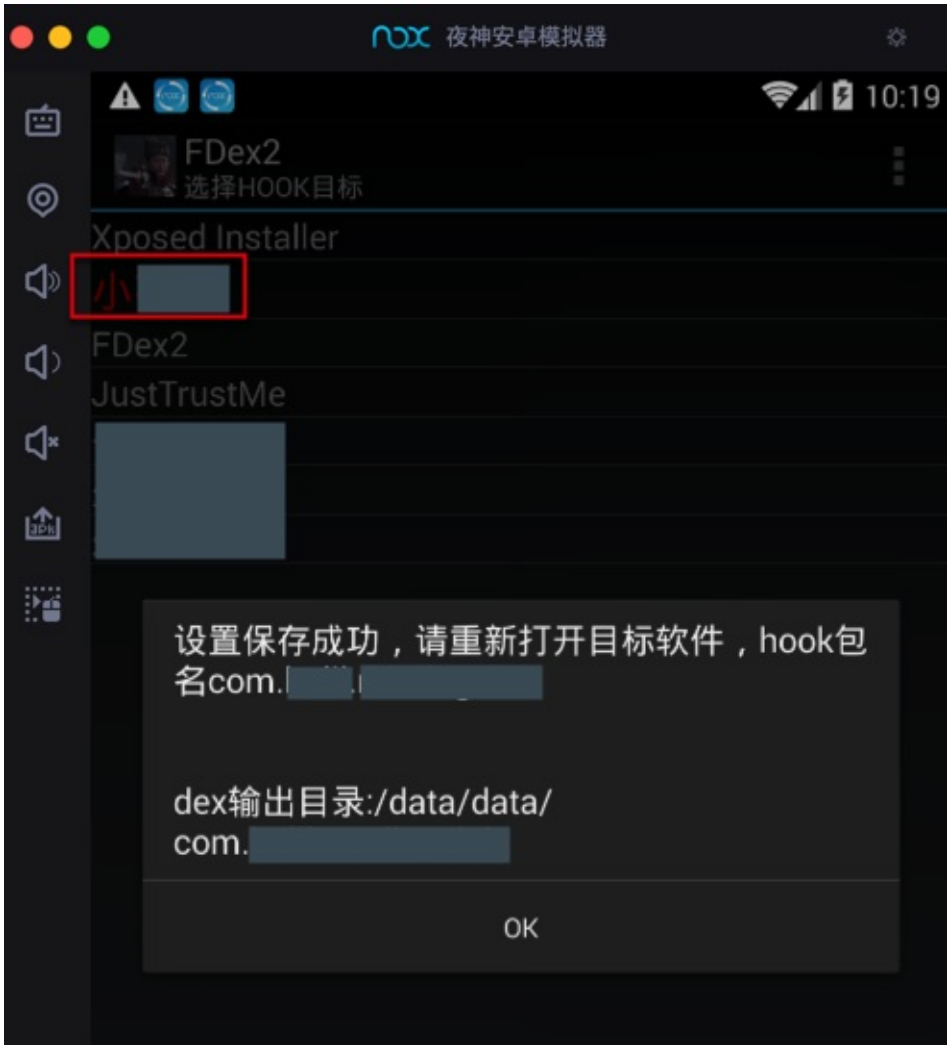
运行FDex2，点选要破解的app

再去打开`FDex2`，会看到一个（当前安卓系统已安装的）app的列表：

然后点击对应的，你要破解导出dex的app，比如此处的：`小花生`。

被选中的app名字会变红色，且会弹框提示你包名和保存（导出dex文件的）路径：

> 设置保存成功，请重新打开目标软件，hook包名：com.huili.readingclub
>
> dex输出目录：/data/data/com.huili.readingclub

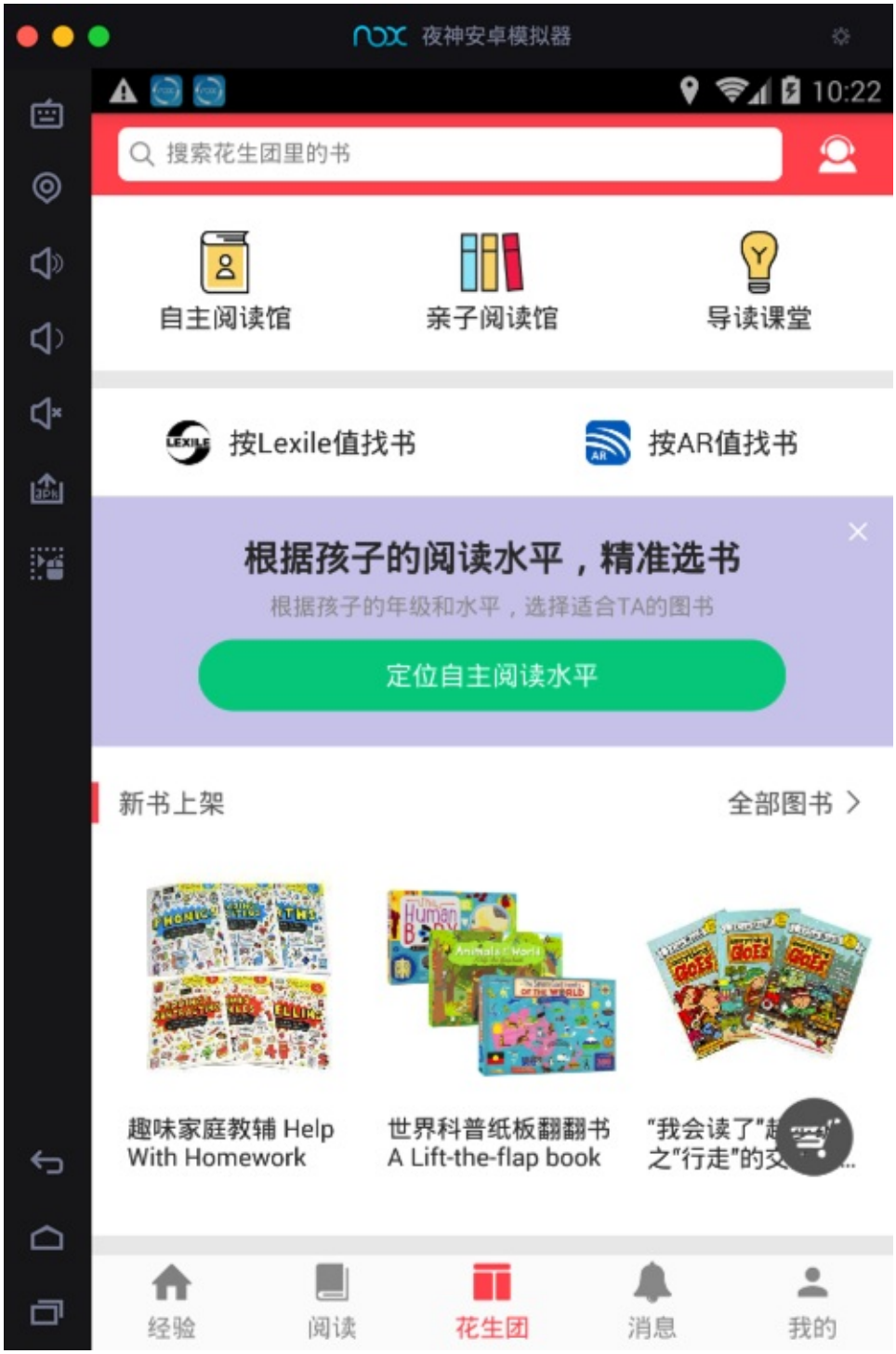


运行要破解的app

再去点击运行被破解的app（此处的 小花生）

正常来说：只要打开了app，稍等几秒（等待FDex2内部的计算和导出保存dex文件的操作），即可完成dex的导出

比如进入了小花生的主页：



稍等几秒，即可。

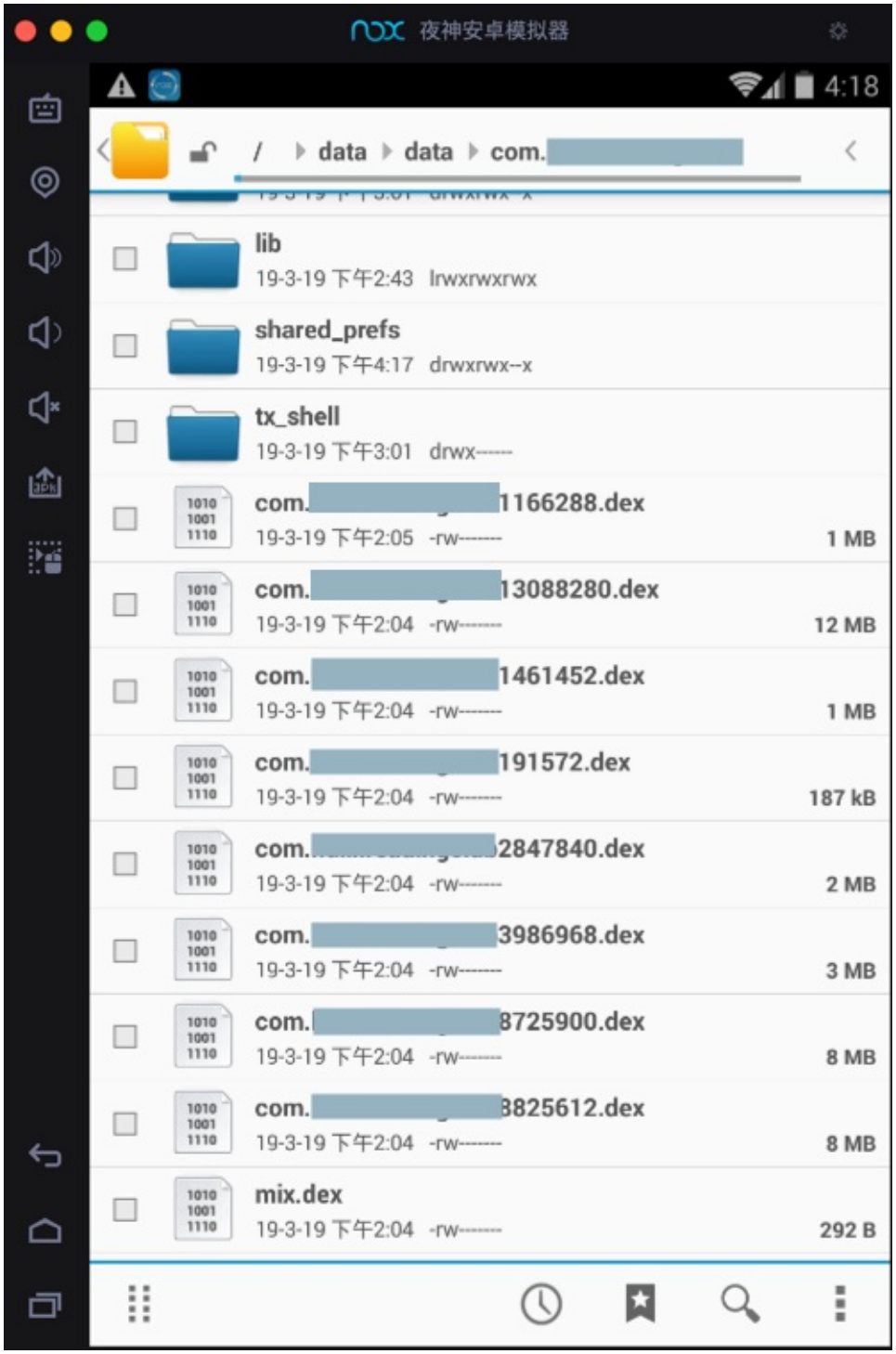
但是为了更保险，此处可以去，随意点击和切换页面，感觉会更好。

去对应目录找导出dex文件

此处内部逻辑是：

`FDex2` 正在导出app的所有dex文件到对应的目录了： `/data/data/com.huili.readingclub`

后续（Nox中用 `文件管理器` ）去打开对应目录，即可看到希望得到的（多个）dex文件：



用FDex2砸壳从安卓app中导出dex

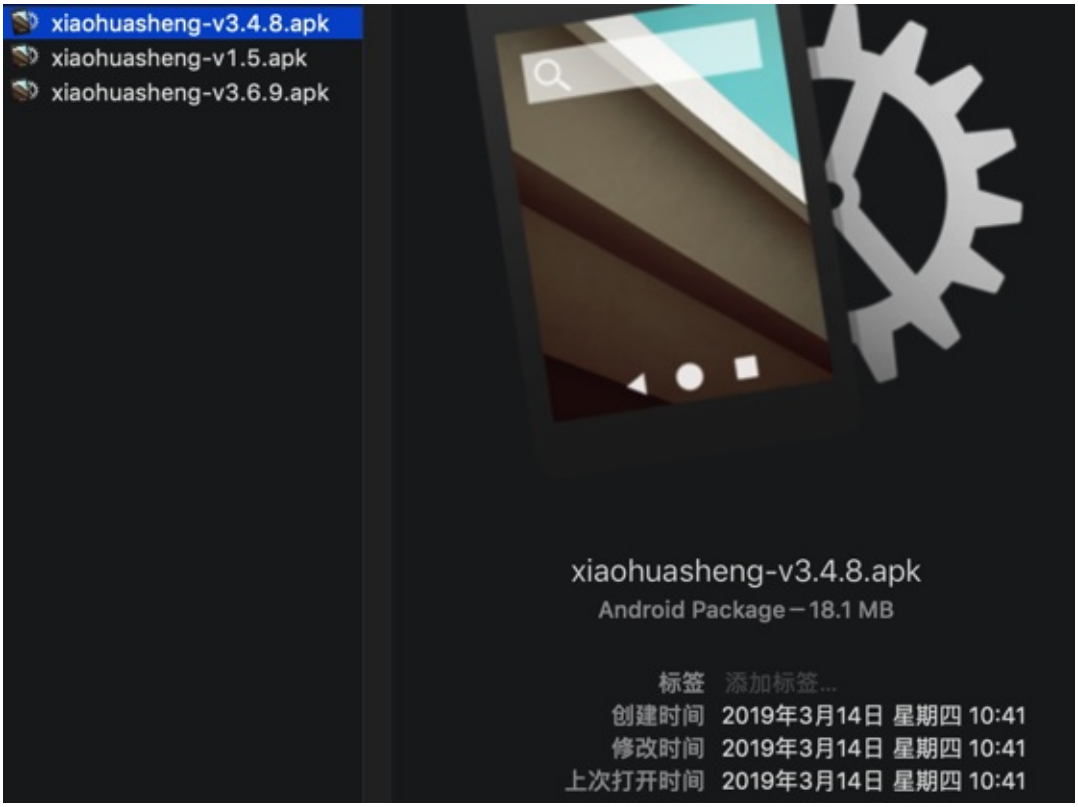
环境初始化

- 已 `root` 了的安卓 `4.4+` 的 `真机` 或 `模拟器`
  - 因为后续的 `FDex2` 要求 `Android 4.4+`
  - 此处用的是： `夜神模拟器` = `Nox App Player`
    - 

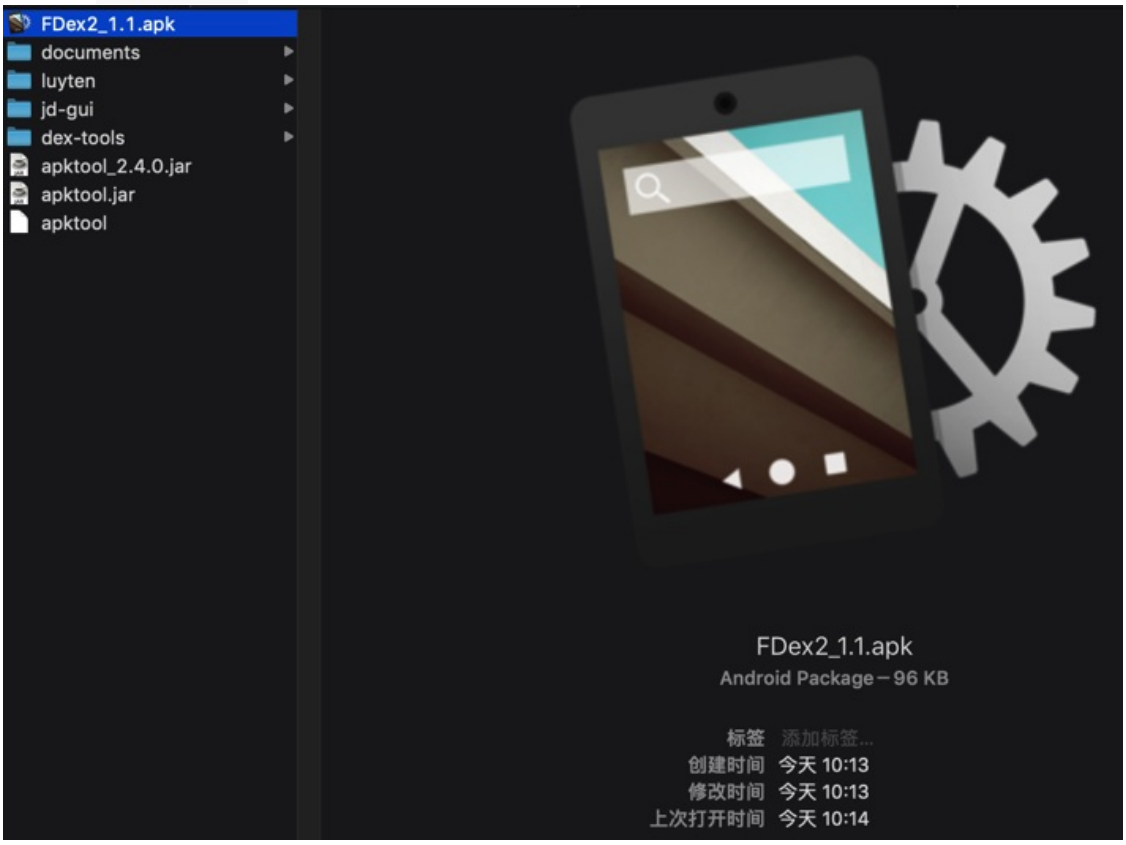
    - 去模拟安卓 `4.4.2`
    - `夜神模拟器` 已自带 `root`
- `Hook框架`

    - 用于后续安装插件去导出 `dex` 文件
    - 此处选用： `XPosed框架`
        - 在安卓中安装 `XPosed` 框架
            - 之前已安装 `XPosed框架`
            - 

- 要破解的安卓apk
    - 比如 `v3.4.8` 的小花生的apk：



- 下载好 `FDex2` 的 `apk`
  - 从这里下载到 `FDex2_1.1.apk`



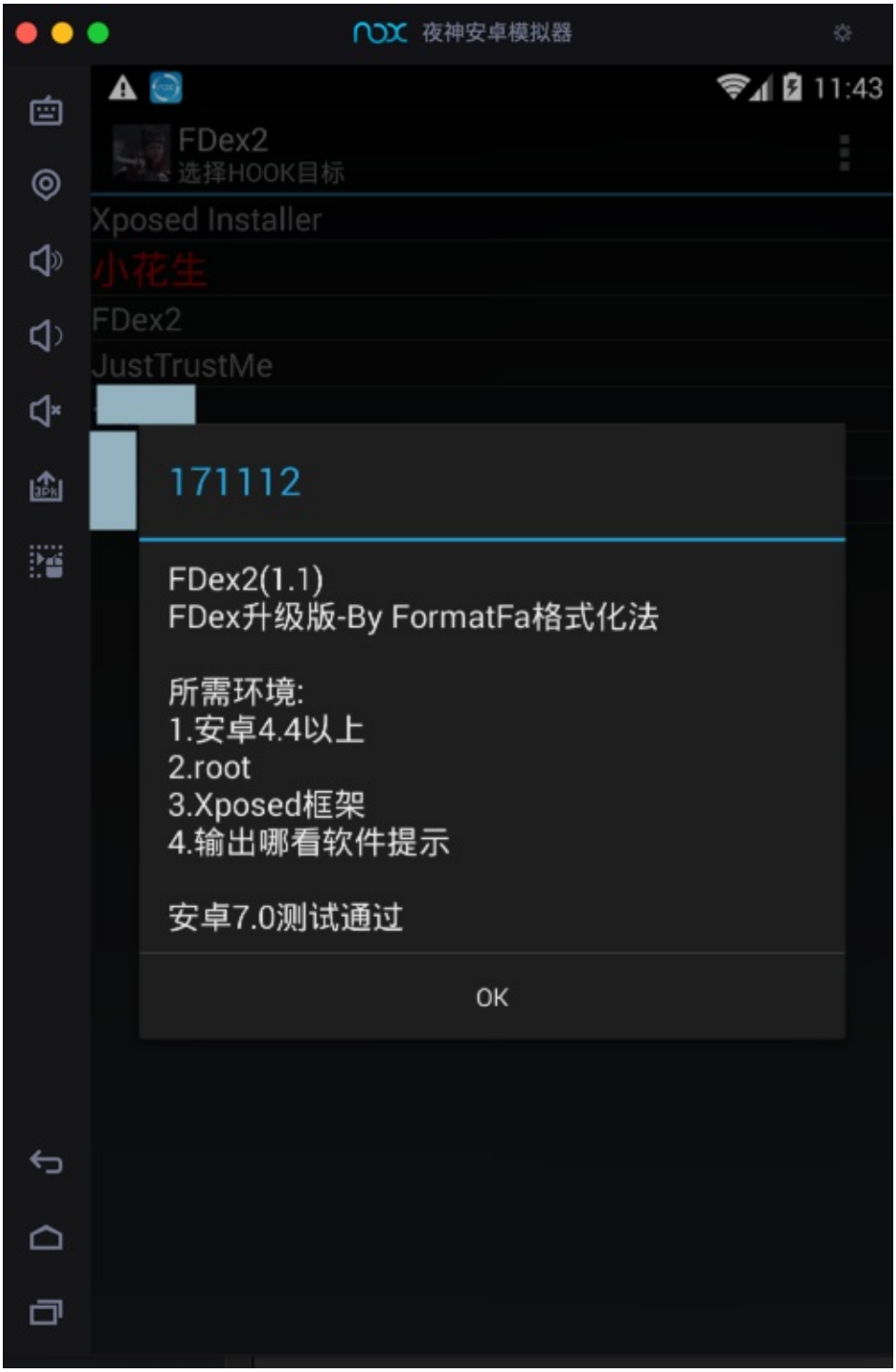
下面以 `夜神模拟器` + `XPosed框架` 为例解释如何操作。

砸壳导出dex的详细步骤

- 安装要破解的apk
  - 把要破解的 `apk` 安装到 `夜神安卓模拟器` 中：

  -
- 安装和激活 `FDex2`
  - 安装 `FDex2` 安装到夜神模拟器中
  - 去XPosed里勾选 `激活` FDex2：

-
- 注意：
  - 其会提示：`XPosed模块列表已更新，重启后更改将生效`
  - 所以为了使 `FDex2生效` ，记得去重启 `XPosed` 。
- 此处的版本是 `1.1` ：

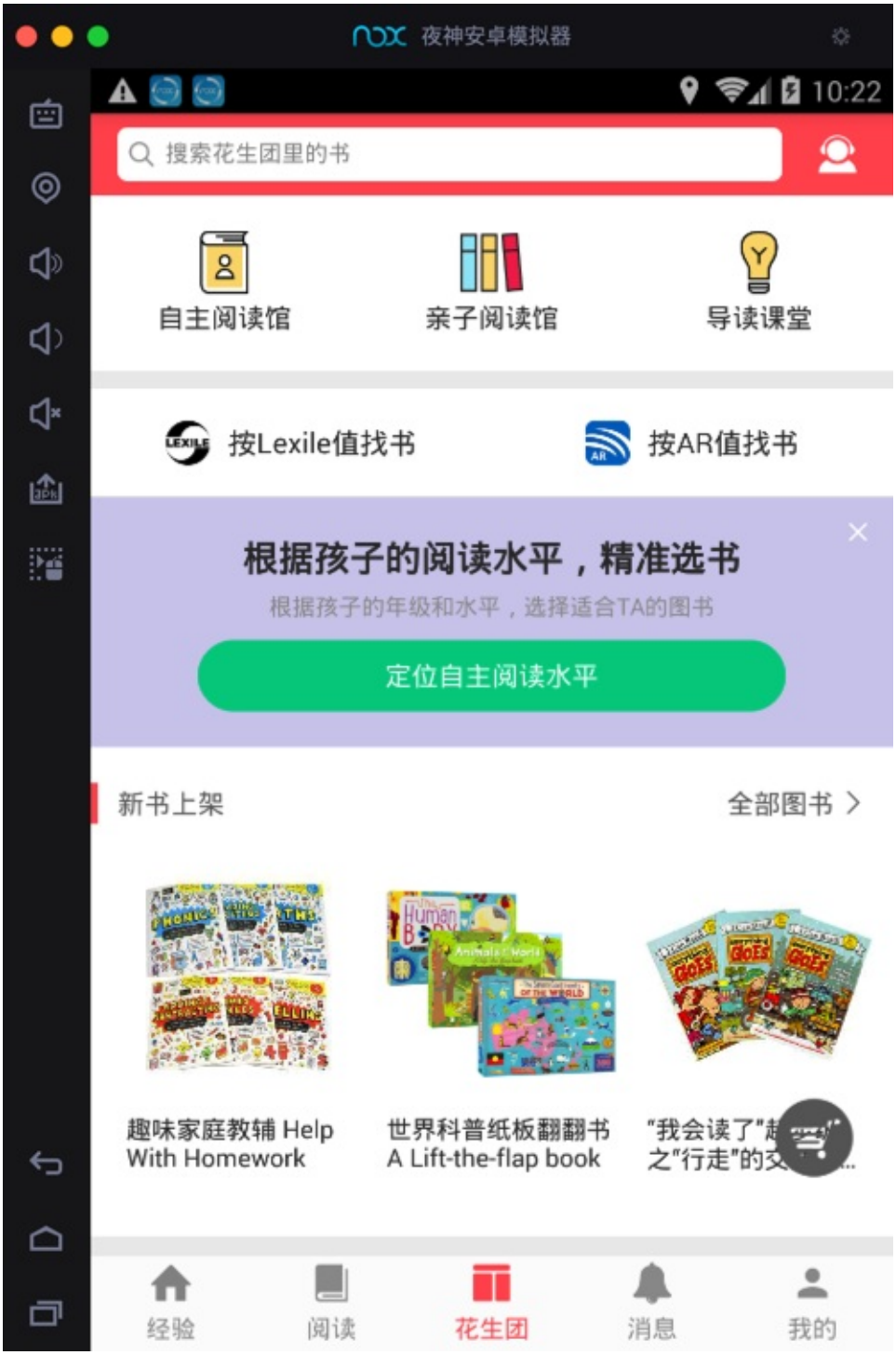


- `FDex2` 中设置要处理的app
  - 然后再去打开FDex2，点击此处要破解的app：小花生

-
- 会提示设置成功：
- > 设置保存成功，请重新打开目标软件，hook包名：`com.huili.readingclub`
- > dex输出目录：`/data/data/com.huili.readingclub`
-

- 运行要破解的app
  - 正常去打开和运行要破解的app

-
- 注意：
  - 其实只要打开了：



- 稍等几秒，即可
  - 此处内部 `FDex2` 已经去导出app的所有文件到对应的目录了：`/data/data/com.huili.readingclub`
- 但是为了更保险，此处再去：随意点击和切换页面，也点击到了要破解的页面，感觉会更好

- 拷贝出已导出的 `dex` 等文件
  - 所以接着去对应路径
    - 之前 `FDex2` 设置app时已提示的输出路径
      - `/data/data/com.huili.readingclub`
  - 找dex文件（和其他相关项目文件）：

-
- 并拷贝出来即可

最终拷贝出我们要的dex文件：

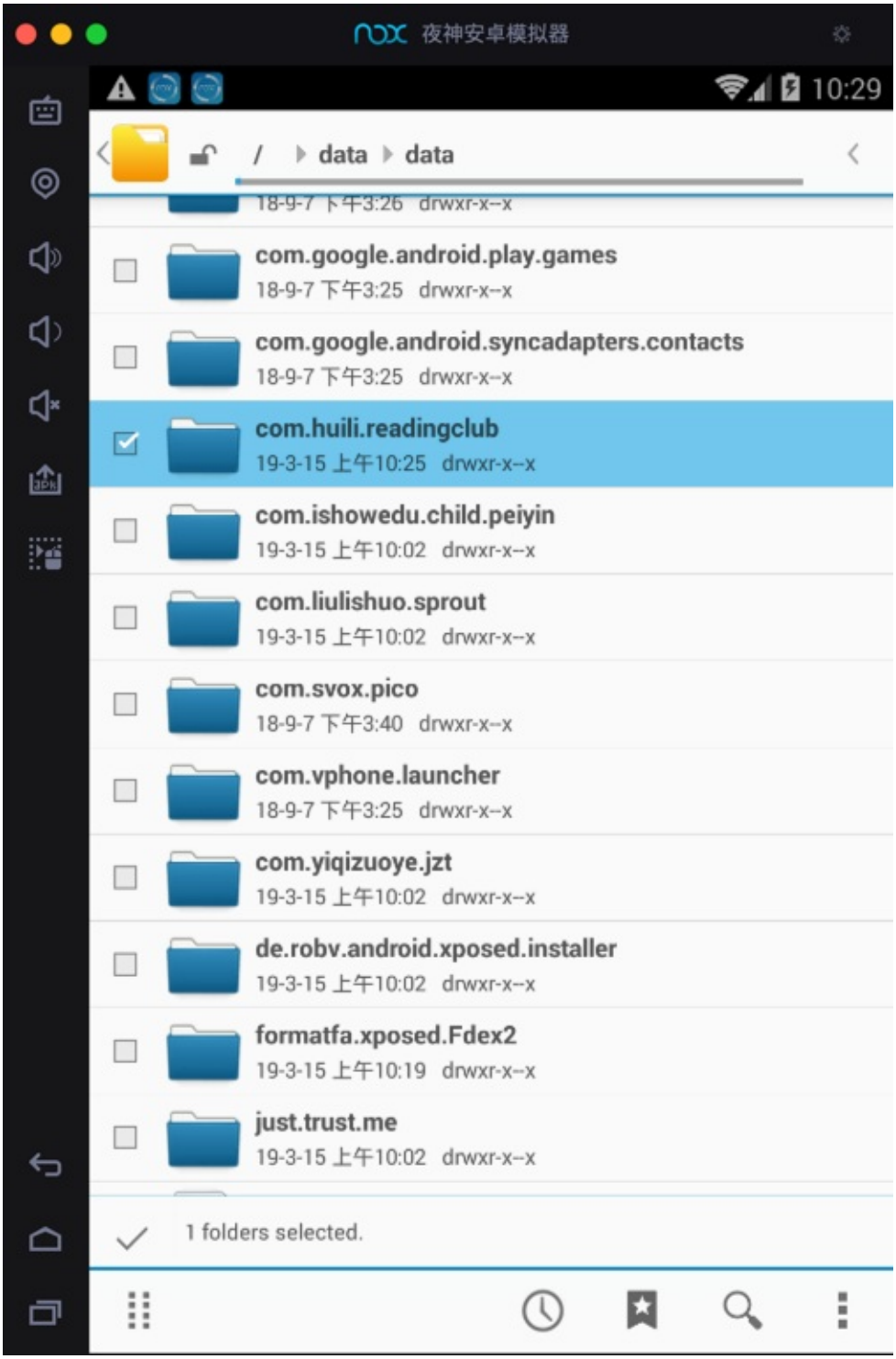
```
➜  v3.4.8 ll
total 81656
-rw-------  1 crifan  staff   1.1M  3 19 14:05 com.huili.readingclub1166288.dex
-rw-------  1 crifan  staff    12M  3 19 14:04 com.huili.readingclub13088280.dex
-rw-------  1 crifan  staff   1.4M  3 19 14:04 com.huili.readingclub1461452.dex
-rw-------  1 crifan  staff   107K  3 19 14:04 com.huili.readingclub191572.dex
-rw-------  1 crifan  staff   2.7M  3 19 14:04 com.huili.readingclub2847840.dex
-rw-------  1 crifan  staff   3.8M  3 19 14:04 com.huili.readingclub3986968.dex
-rw-------  1 crifan  staff   8.3M  3 19 14:04 com.huili.readingclub8725900.dex
-rw-------  1 crifan  staff   8.4M  3 19 14:04 com.huili.readingclub8825612.dex
```

注意和说明

如何从夜神模拟器中导出文件到Mac中

此处使用夜神模拟器自带的 文件管理器 ：

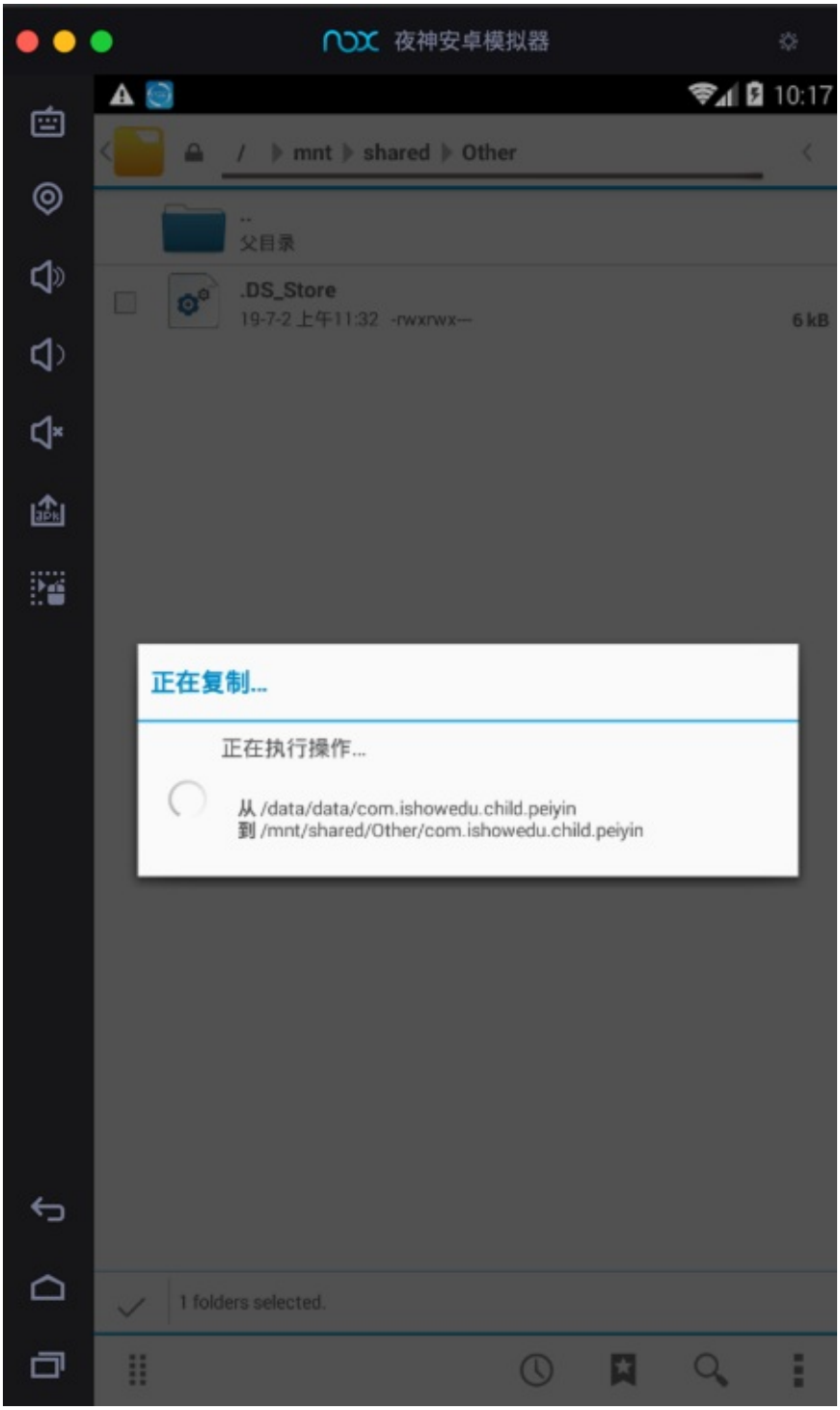
- 详细步骤：
  - 先把导出的文件拷贝到共享目录
    1. 勾选 `/data/data/` 下面的 `com.huili.readingclub`



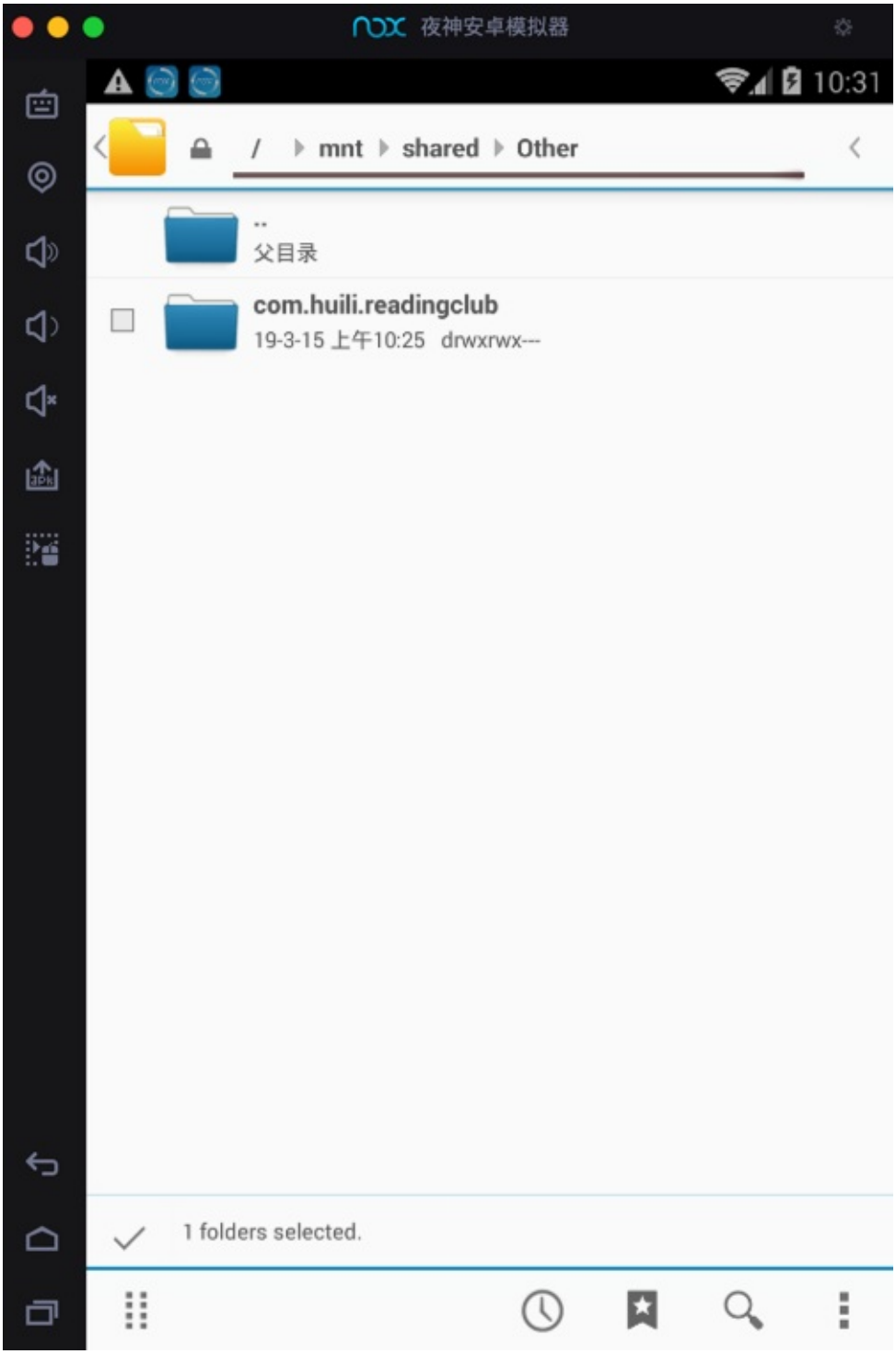
2. 切换到夜神用于和电脑共享的文件夹 `/mnt/shared/Other` 中去 `操作` -> `粘贴选择项`


夜神安卓模拟器
10:15
/ mnt shared Other
..
父目录
.DS_Store
19-7-2 上午11:32 -rwxrwx---
6 kB
操作
属性
刷新
新建文件夹
新建文件
全部选择
取消全选
粘贴选择项
移动选择项
压缩选择项
添加到书签
取消
1 folders selected.

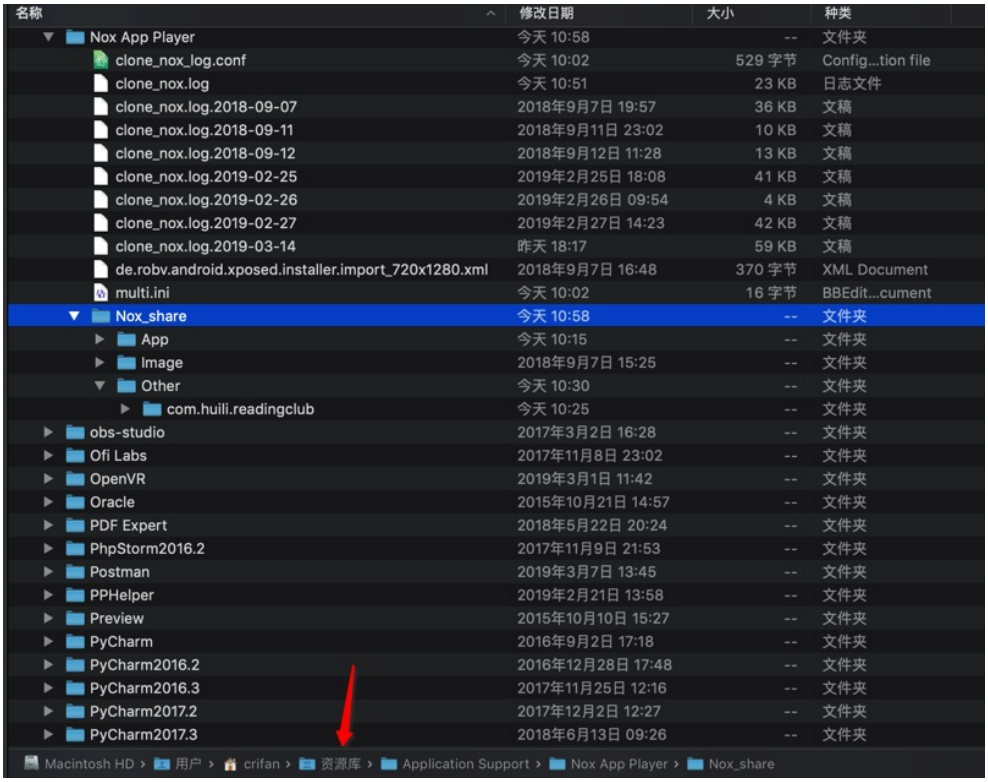
- 然后会开始复制和粘贴：



3. 复制粘贴后，无法立即看到已拷贝的文件。回到上一级目录，再重新进来即可看到：

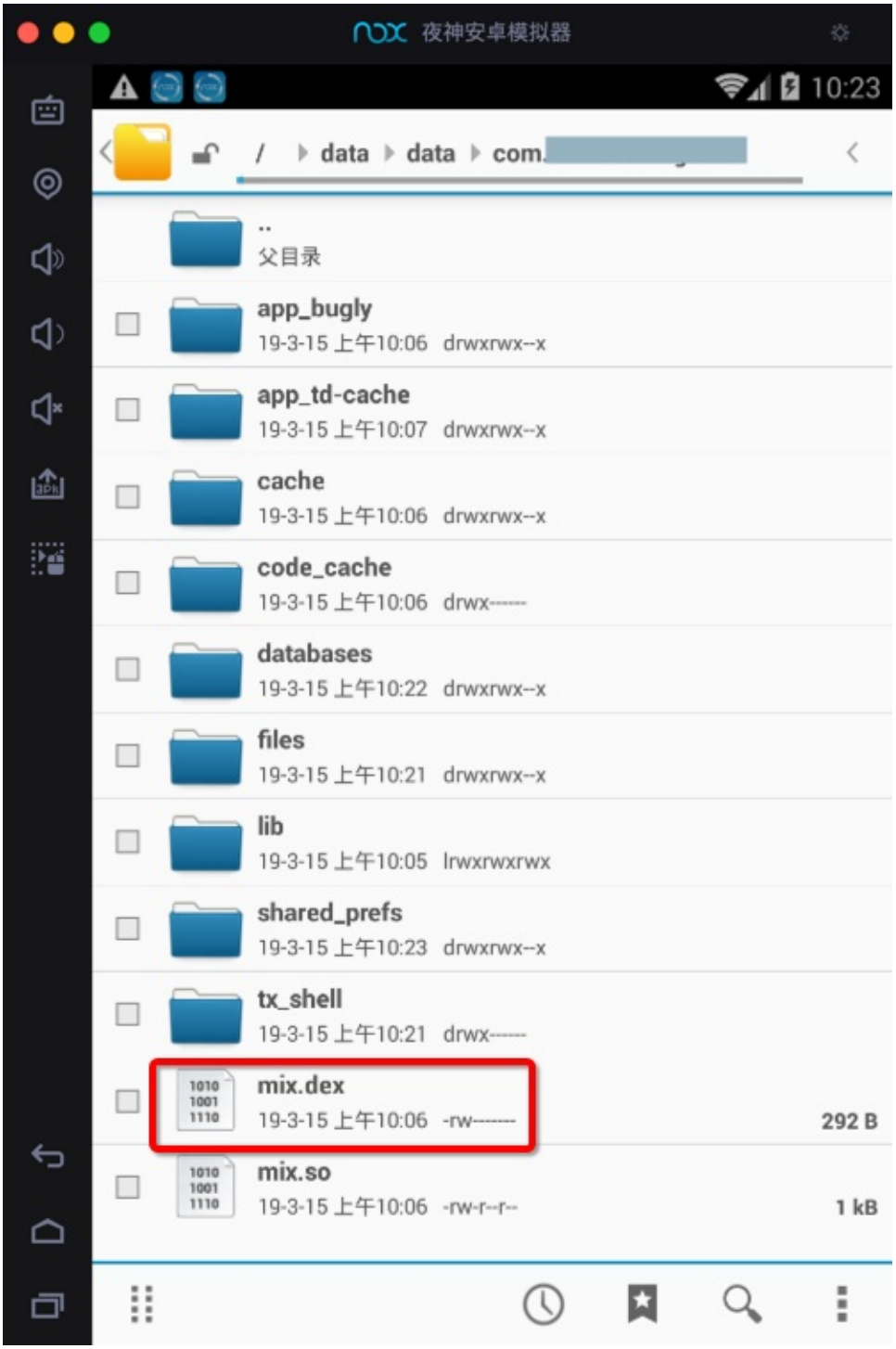


- 再去到PC（此处Mac）中找到共享根目录，并拷贝文件
  - 夜神模拟器和PC的共享根目录
    - `Windows`：
      - 据说是 `C:\Users\{USERNAME}/Nox_share/Other`
    - `Mac`：
      - `/Users/{用户名}/Library/Application Support/Nox App Player/Nox_share`
      - 举例：
        - `/Users/crifan/Library/Application\ Support/Nox\ App\ Player/Nox_share`
          - 说明：其中空格需要 `\` 转义

- 

没看到我们要导出的dex文件

1. 有时候没有看到导出文件，则可以重新多试试几次，即可。
2. 有些app（的有些版本），导出的文件中，没有我们希望的（多个dex文件中的某个）包含了app业务逻辑的dex文件
   - 说明该apk采用了更加高级的加固，无法导出我们要的dex文件
   - 举例：
     - 比如小花生的v3.6.9导出的只有一个无效的292B的dex文件：mix.dex



DumpDex

用来从 `运行中的安卓app` 中导出 `dex` 文件的工具。

- 官网地址
  - WrBug/dumpDex:　一款Android脱壳工具，需要xposed支持, 易开发已集成该项目

不要从DeveloperHelper下载

对于想要去下载dumpDex的话，建议去：

https://github.com/WrBug/dumpDex/releases

而不要去作者所说的，去下载

集成了DumpDex的DeveloperHelper

-> 否则我此处会导致夜神模拟器中的app运行时产生崩溃，从而无法导出想要的dex文件

具体用法详见：

配合XPosed插件导出dex · 好用的安卓模拟器：夜神Nox (crifan.org)

DexExtractor

- 功能
  - 用于破解邦邦加密的安卓 `dex` 文件提取器
- github主页
  - lambdalang/DexExtractor
- 使用说明
  - 4.4的虚拟机 编译好了libdvm。
  - 代码github上，脱梆梆的壳，别的没测试
  - 把dexdump出来，然后base64解码下，然后odex2dex，没了
  - system.img 有空上传
    - 作者编译好的镜像文件 `system.img` 的下载地址
      - system-arm_md5_6395c2f1451dbbed027d7293ab39a6e7.img.tar.gz
  - 启动模拟器加上sdcard
- 注意
  - apk没有写权限的反编译了加上write就好了
- 支持
  - 梆梆加固
  - 爱加密（新版本没事）
  - 其他，暂时没测试

drizzleDumper

听别人提到过，自己没用过。

- 功能
  - 一款基于内存搜索的Android脱壳工具
    - 可以从运行中的安卓app中，利用 `ptrace机制` ，导出dex文件
- github主页
  - DrizzleRisk/drizzleDumper: drizzleDumper是一款基于内存搜索的Android脱壳工具
- 机制和原理
  - root设备之后，通过ptrace附加需要脱壳的apk进程，然后在脱壳的apk进程的内存中进行dex文件的特征搜索，当搜索到dex文件时，进行dex文件的内存dump
- 使用步骤
  - 将 `\armeabi` 下的 `drizzleDumper` 去 `push` 进手机
  - 进入 `shell` ，赋给可执行权限
  - 运行 `drizzleDumper [包名] [等待时间,默认为0]`
  - 运行需要脱壳程序
- 使用举例

```
$ adb push F:\drizzleDumper /data/local/tmp
$ chmod 755 drizzleDumper
$ ./drizzleDumper xyz.sysorem.crackme
```

InDroid

- 主页
  - romangol/InDroid: Dalvik vm Instrumentation OS
- 作者
  - GoSSIP小组
- 功能
  - 基于Dalvik VM的插桩分析框架
- 原理
  - 直接修改AOSP上的Dalvik VM解释器，在解释器解释执行Dalvik字节码时，插入监控的代码，这样就可以获取到所有程序运行于Dalvik上的动态信息，如执行的指令、调用的方法信息、参数返回值、各种Java对象的数据等等。InDroid只需要修改AOSP的dalvik vm部分代码，编译之后，可直接将编译生成的新libdvm.so刷入任何AOSP支持的真机设备上

DexHunter

- `DexHunter`
  - 主页
    - zyq8709/DexHunter: General Automatic Unpacking Tool for Android Dex Files
      - https://github.com/zyq8709/DexHunter
  - 是什么：一个Android dex的通用脱壳器
  - 作用：导出dex文件
  - 主要思想：以AOP的模式对运行时ART和DVM进行定制
  - 详细介绍
    - [原创]Android dex文件通用自动脱壳器-Android安全-看雪论坛-安全社区|安全招聘|bbs.pediy.com
      - https://bbs.pediy.com/thread-203776.htm

FART

- FART
  - 是什么：ART环境下基于主动调用的自动化脱壳方案
  - 特点：支持ART（新安卓的VM虚拟机）
    - 基于Android 6.0实现，理论上可以移植到任何ART系统上
  - FART脱壳的步骤
    - 1.内存中DexFile结构体完整dex的dump
    - 2.主动调用类中的每一个方法，并实现对应CodeItem的dump
    - 3.通过主动调用dump下来的方法的CodeItem进行dex中被抽取的方法的修复
  - 详解
    - hanbinglengyue/FART: ART环境下自动化脱壳方案
      - https://github.com/hanbinglengyue/FART
    - [原创]FART：ART环境下基于主动调用的自动化脱壳方案-Android安全-看雪论坛-安全社区|安全招聘|bbs.pediy.com
      - https://bbs.pediy.com/thread-252630.htm

反编译dex

dex转jar

注意事项

我们从砸壳导出dex文件等方式，得到（一个或）多个dex文件

接下来，就是想办法，如何从dex导出我们要的java源代码了

不过在转换之前，需要了解一个细节：

- 到底要转换那个dex（为java）
  - 全部都转换吗？
  - 如果不是，那需要转换那个dex呢？
- 尤其是这么多 `dex` ，哪个dex或哪些dex，才是真正包含了安卓app的业务逻辑的 `dex` 文件？
  - 往往是dex中最大文件的那个
  - 当前具体是哪个，还需要根据实际情况去确认
- 当然，转换一下也不麻烦，所以可以挨个转换一遍
  - 只不过，dex转jar再jar转java，或dex直接转java，往往期间会出现各种错误
  - 总之如果尽量早知道哪个是具体所关注的包含安卓app业务逻辑的dex后，其他dex的转换期间，即使报错，也可以忽略了，可以省去一些麻烦

我们后续会（经过转换和尝试而）找到该 `dex` ，然后用 `dex2jar` 等工具从 `dex` 转换出 `jar` 文件

dex2jar

- `dex2jar`
  - 功能
    - 用于处理 `安卓` 的 `.dex` 文件和 `java` 的 `.class` 文件的一系列的工具
      - 核心和常用功能
        - 从 `dex` 文件导出 `jar` 文件
      - 一系列的工具，包括
        - `dex-reader/writer`：读写 `dex` （Dalvik Executable）文件
          - 具有和 `ASM` 类似的轻量级的API接口
        - `d2j-dex2jar`：把 `dex` 文件转换为 `class` 文件（=jar压缩包文件= `jar` 包= `jar` 文件）
        - `smali/baksmali`：反汇编 `dex` 转换出 `smali` 文件，`从smali` 文件中汇编出 `dex` 文件
          - 和smali/baksmali虽然语法相同，但不太一样的是，此处支持描述中包含 `"Lcom/dex2jar\t\u1234;"` 这类文字描述
        - 其他一些工具
          - d2j-decrypt-string
  - 资料
    - 官网
      - dex2jar | Penetration Testing Tools
    - 其他主页/镜像
      - github
        - pxb1988/dex2jar: Tools to work with android .dex and java .class files
      - SourceForge
        - https://sourceforge.net/p/dex2jar

下载dex2jar

- 从dex2jar的下载页面下载到最新版本的 `dex2jar`
  - 比如：dex-tools-2.1-SNAPSHOT.zip
    - 解压后得到：`d2j-dex2jar.sh`

使用

- 概述

```
d2j-dex2jar.sh -f apk_file.apk
d2j-dex2jar.sh -f dex_file.dex
```

  - 举例

```
sh dex-tools/dex-tools-2.1-SNAPSHOT/d2j-dex2jar.sh -f com.huili.readingclub3986968.dex
```

用dex2jar从dex导出jar

对于前面的 `v3.4.8` 版本的某安卓apk，经过前面一步导出了多个 `dex` 文件后，此处接着去用 `dex2jar` 去分别转换出 `jar`：

转换期间，很多个 `dex` 转换 `jar` 的期间都会报错。

经过尝试最终找到了，转换不仅不报错且转换出来的 `jar` 还是我们希望的有效的包含了app业务逻辑的 `jar`：

```
➜  v3.4.8 /Users/crifan/dev/dev_tool/android/reverse_engineering/dex-tools/dex-tools-2.1-SNAPSHOT/d2j-dex2jar.sh -f c
om.huili.readingclub3986968.dex
dex2jar com.huili.readingclub3986968.dex -  ./com.huili.readingclub3986968-dex2jar.jar
➜  v3.4.8 /Users/crifan/dev/dev_tool/android/reverse_engineering/dex-tools/dex-tools-2.1-SNAPSHOT/d2j-dex2jar.sh -f c
om.huili.readingclub8825612.dex
dex2jar com.huili.readingclub8825612.dex -  ./com.huili.readingclub8825612-dex2jar.jar
```

经后续确定，此处：

- 从：`8.4MB` 的 `com.huili.readingclub8825612.dex`
- 转换出的：`10MB` 的 `com.huili.readingclub8825612-dex2jar.jar`

就是我们要的，包含了app的业务逻辑的代码。

说明和提示

通过apk改名zip解压得到的 `classes.dex` 是无效的，无法转换处 `jar`

比如之前的 `v3.6.9` 版本的某安卓apk，改名为 `zip` 再解压得到的 `classes.dex`，此处用 `dex2jar` 去转换：

```
➜  classes.dex pwd
/Users/crifan/dev/dev_root/company/naturling/projects/crawl_data/小花生app/xiaohuasheng/decoded dex/v3.6.9/classes.dex
➜  classes.dex /Users/crifan/dev/dev_tool/android/reverse_engineering/dex-tools/dex-tools-2.1-SNAPSHOT/d2j-dex2jar.sh
-f classes.dex
dex2jar classes.dex -  ./classes-dex2jar.jar
➜  classes.dex ll
total 26000
-rw-------  1 crifan  staff   212K  3 25 10:20 classes-dex2jar.jar
-rwxr-xr-x@ 1 crifan  staff    12M  1 25 17:43 classes.dex
```

虽然转换过程并没有报错，但是才得到200多KB的jar

-》最终经确认，其中没有包含业务逻辑代码，不是我们需要的jar，是无效的jar

多个 `dex` 转换 `jar` 的期间会报错

比如：

```
➜  v3.4.8 /Users/crifan/dev/dev_tool/android/reverse_engineering/dex-tools/dex-tools-2.1-SNAPSHOT/d2j-dex2jar.sh -f c
om.huili.readingclub1166288.dex
'''
GLITCH: 0000 Lcom/android/internal/telephony/uicc/VoiceMailConstants;.getVoiceMailTag(Ljava/lang/String;)Ljava/lang/S
tring;  zero-width instruction op 0xf4
Detail Error Information in File ./com.huili.readingclub1166288-error.zip
Please report this file to one of following link if possible (any one).
    https://sourceforge.net/p/dex2jar/tickets/
    https://bitbucket.org/pxb1988/dex2jar/issues
    https://github.com/pxb1988/dex2jar/issues
    dex2jar@googlegroups.com

➜  v3.4.8 /Users/crifan/dev/dev_tool/android/reverse_engineering/dex-tools/dex-tools-2.1-SNAPSHOT/d2j-dex2jar.sh -f c
om.huili.readingclub13088280.dex
'''
```

```
GLITCH: 009f Lcom/tencent/bugly/legu/proguard/z;.a(Ljava/lang/Thread;Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;)V | zero-width instruction op 0xf8
Detail Error Information in File ./com.huili.readingclub13088280-error.zip


➜  v3.4.8 /Users/crifan/dev/dev_tool/android/reverse_engineering/dex-tools/dex-tools-2.1-SNAPSHOT/d2j-dex2jar.sh -f com.huili.readingclub1461452.dex
...
GLITCH: 0000 Lcom/google/android/util/SmileyResources;.getSmileys()Lcom/google/android/util/AbstractMessageParser$TrieNode; | zero-width instruction op 0xf4
WARN: can't get operand(s) for sub-double/2addr, out-of-range or not initialized ?
WARN: can't get operand(s) for int-to-float, out-of-range or not initialized ?
WARN: can't get operand(s) for return-wide, out-of-range or not initialized ?
WARN: can't get operand(s) for move-exception, out-of-range or not initialized ?
WARN: can't get operand(s) for move-exception, out-of-range or not initialized ?
Detail Error Information in File ./com.huili.readingclub1461452-error.zip


➜  v3.4.8 /Users/crifan/dev/dev_tool/android/reverse_engineering/dex-tools/dex-tools-2.1-SNAPSHOT/d2j-dex2jar.sh -f com.huili.readingclub191572.dex
...
GLITCH: 0006 Lcom/android/okhttp/internal/tls/OkHostnameVerifier;.verifyHostName(Ljava/lang/String;Ljava/lang/String;)Z | zero-width instruction op=0xee
Detail Error Information in File ./com.huili.readingclub191572-error.zip

➜  v3.4.8 /Users/crifan/dev/dev_tool/android/reverse_engineering/dex-tools/dex-tools-2.1-SNAPSHOT/d2j-dex2jar.sh -f com.huili.readingclub2847840.dex
...
GLITCH: 0006 Lsun/misc/Unsafe;.unpark(Ljava/lang/Object;)V | zero-width instruction op=0xf8
Detail Error Information in File ./com.huili.readingclub2847840-error.zip
➜  v3.4.8 /Users/crifan/dev/dev_tool/android/reverse_engineering/dex-tools/dex-tools-2.1-SNAPSHOT/d2j-dex2jar.sh -f com.huili.readingclub8725900.dex
...
GLITCH: 0000 Landroid/widget/ZoomControls;.setOnZoomOutClickListener(Landroid/view/View$OnClickListener;)V | zero-width instruction op=0xf4
GLITCH: 0000 Landroid/widget/ZoomControls;.setZoomSpeed(J)V | zero-width instruction op=0xf4
WARN: can't get operand(s) for move-result-object, wrong position ?
WARN: can't get operand(s) for move-result-object, wrong position ?
WARN: can't get operand(s) for move-result-object, wrong position ?
WARN: can't get operand(s) for move-object/16, out-of-range or not initialized ?
WARN: can't get operand(s) for shr-int/2addr, out-of-range or not initialized ?
WARN: can't get operand(s) for move/16, out-of-range or not initialized ?
WARN: can't get operand(s) for move-result-object, wrong position ?
WARN: can't get operand(s) for move/16, out-of-range or not initialized ?
WARN: can't get operand(s) for move-result, wrong position ?
WARN: can't get operand(s) for cmpl-float, out-of-range or not initialized ?
WARN: can't get operand(s) for move-result-object, wrong position ?
WARN: can't get operand(s) for sput-boolean, out-of-range or not initialized ?
WARN: can't get operand(s) for move-result-object, wrong position ?
WARN: can't get operand(s) for move-result-object, wrong position ?
WARN: can't get operand(s) for move-result-object, wrong position ?
WARN: can't get operand(s) for move-object/from16, out-of-range or not initialized ?
WARN: can't get operand(s) for move-object/from16, out-of-range or not initialized ?
WARN: can't get operand(s) for move-object/from16, out-of-range or not initialized ?
WARN: can't get operand(s) for move-object/from16, out-of-range or not initialized ?
WARN: can't get operand(s) for move-object/from16, out-of-range or not initialized ?
WARN: can't get operand(s) for sput-boolean, out-of-range or not initialized ?
WARN: can't get operand(s) for sput-boolean, out-of-range or not initialized ?
WARN: can't get operand(s) for move-result-object, wrong position ?
WARN: can't get operand(s) for aput-char, out-of-range or not initialized ?
WARN: can't get operand(s) for mul-float, out-of-range or not initialized ?
WARN: can't get operand(s) for move-result-object, wrong position ?
WARN: can't get operand(s) for move-result-object, wrong position ?
WARN: can't get operand(s) for move-result-object, wrong position ?
WARN: can't get operand(s) for move-wide/16, out-of-range or not initialized ?
WARN: can't get operand(s) for move-result-object, wrong position ?
WARN: can't get operand(s) for mul-int/2addr, out-of-range or not initialized ?
WARN: can't get operand(s) for aput-char, out-of-range or not initialized ?
WARN: can't get operand(s) for aput-char, out-of-range or not initialized ?
WARN: can't get operand(s) for move-result-object, wrong position ?
WARN: can't get operand(s) for sput-byte, out-of-range or not initialized ?
WARN: can't get operand(s) for aget-byte, out-of-range or not initialized ?
WARN: can't get operand(s) for and-int/2addr, out-of-range or not initialized ?
WARN: can't get operand(s) for move/from16, out-of-range or not initialized ?
WARN: can't get operand(s) for iput-boolean, out-of-range or not initialized ?
WARN: can't get operand(s) for iput-boolean, out-of-range or not initialized ?
WARN: can't get operand(s) for move-result-object, wrong position ?
WARN: can't get operand(s) for cmpg-float, out-of-range or not initialized ?
```

```
Detail Error Information in File ./com.huili.readingclub8725900-error.zip
Please report this file to one of following link if possible (any one).
    https://sourceforge.net/p/dex2jar/tickets/
    https://bitbucket.org/pxb1988/dex2jar/issues
    https://github.com/pxb1988/dex2jar/issues
    dex2jar@googlegroups.com
java.util.IllegalFormatConversionException: d != java.lang.String
    at java.util.Formatter$FormatSpecifier.failConversion(Formatter.java:4302)
    at java.util.Formatter$FormatSpecifier.printInteger(Formatter.java:2793)
    at java.util.Formatter$FormatSpecifier.print(Formatter.java:2747)
    at java.util.Formatter.format(Formatter.java:2520)
    at java.util.Formatter.format(Formatter.java:2455)
    at java.lang.String.format(String.java:2940)
    at com.googlecode.d2j.smali.BaksmaliDumpOut.s(BaksmaliDumpOut.java:68)
    at com.googlecode.d2j.smali.BaksmaliCodeDumper.visitFilledNewArrayStmt(BaksmaliCodeDumper.java:248)
    at com.googlecode.d2j.node.insn.FilledNewArrayStmtNode.accept(FilledNewArrayStmtNode.java:19)
    at com.googlecode.d2j.smali.BaksmaliDumper.accept(BaksmaliDumper.java:569)
    at com.googlecode.d2j.smali.BaksmaliDumper.baksmaliCode(BaksmaliDumper.java:544)
    at com.googlecode.d2j.smali.BaksmaliDumper.baksmaliMethod(BaksmaliDumper.java:482)
    at com.googlecode.d2j.smali.BaksmaliDumper.baksmaliMethod(BaksmaliDumper.java:428)
    at com.googlecode.dex2jar.tools.BaksmaliBaseDexExceptionHandler.dumpMethod(BaksmaliBaseDexExceptionHandler.java:1
48)
    at com.googlecode.dex2jar.tools.BaksmaliBaseDexExceptionHandler.dumpTxt0(BaksmaliBaseDexExceptionHandler.java:126)

    at com.googlecode.dex2jar.tools.BaksmaliBaseDexExceptionHandler.dumpZip(BaksmaliBaseDexExceptionHandler.java:135)
    at com.googlecode.dex2jar.tools.BaksmaliBaseDexExceptionHandler.dump(BaksmaliBaseDexExceptionHandler.java:92)
    at com.googlecode.dex2jar.tools.Dex2jarCmd.doCommandLine(Dex2jarCmd.java:120)
    at com.googlecode.dex2jar.tools.BaseCmd.doMain(BaseCmd.java:290)
    at com.googlecode.dex2jar.tools.Dex2jarCmd.main(Dex2jarCmd.java:33)
```

其中注意到：

- 如果报错，会有把错误信息导出到名为xxx-error.zip的压缩文件中。
  - 比如：`com.huili.readingclub2847840-error.zip`
    - 可供后续分析使用
- 后续经过确认，这些报错的，往往是没有包含app业务逻辑的，不是我们要的dex，所以可以忽略这些错误

上述所有的`dex`转换为`jar`之后：

```
➜  v3.4.8 ll
total 125288
-rw-------  1 crifan  staff   469K  3 21 09:55 com.huili.readingclub1166288-dex2jar.jar
-rw-r--r--  1 crifan  staff    14K  3 21 09:55 com.huili.readingclub1166288-error.zip
-rw-------  1 crifan  staff   1.1M  3 19 14:05 com.huili.readingclub1166288.dex
-rw-------  1 crifan  staff   121K  3 21 09:56 com.huili.readingclub13088280-dex2jar.jar
-rw-r--r--  1 crifan  staff    16K  3 21 09:56 com.huili.readingclub13088280-error.zip
-rw-------  1 crifan  staff    12M  3 19 14:04 com.huili.readingclub13088280.dex
-rw-------  1 crifan  staff   669K  3 21 09:56 com.huili.readingclub1461452-dex2jar.jar
-rw-r--r--  1 crifan  staff    25K  3 21 09:56 com.huili.readingclub1461452-error.zip
-rw-------  1 crifan  staff   1.4M  3 19 14:04 com.huili.readingclub1461452.dex
-rw-------  1 crifan  staff   103K  3 21 09:57 com.huili.readingclub191572-dex2jar.jar
-rw-r--r--  1 crifan  staff   7.0K  3 21 09:57 com.huili.readingclub191572-error.zip
-rw-------  1 crifan  staff   187K  3 19 14:04 com.huili.readingclub191572.dex
-rw-------  1 crifan  staff   1.6M  3 21 09:58 com.huili.readingclub2847840-dex2jar.jar
-rw-r--r--  1 crifan  staff    47K  3 21 09:58 com.huili.readingclub2847840-error.zip
-rw-------  1 crifan  staff   2.7M  3 19 14:04 com.huili.readingclub2847840.dex
-rw-------  1 crifan  staff   3.5M  3 21 09:59 com.huili.readingclub3986968-dex2jar.jar
-rw-------  1 crifan  staff   3.8M  3 19 14:04 com.huili.readingclub3986968.dex
-rw-------  1 crifan  staff   5.1M  3 21 10:00 com.huili.readingclub8725900-dex2jar.jar
-rw-r--r--  1 crifan  staff    68K  3 21 10:00 com.huili.readingclub8725900-error.zip
-rw-------  1 crifan  staff   8.3M  3 19 14:04 com.huili.readingclub8725900.dex
-rw-------  1 crifan  staff   9.5M  3 21 10:00 com.huili.readingclub8825612-dex2jar.jar
-rw-------  1 crifan  staff   8.4M  3 19 14:04 com.huili.readingclub8825612.dex
```

Enjarify

- `Enjarify`
  - 功能
    - 把安卓的 `Dalvik` 字节码转换为Java的 `字节码`
      - 和 `dex2jar` 类似
  - 特点
    - Google 官方开源的
    - 用 `Python 3` 写的
    - 比 `dex2jar` 更新且更好
  - 用法
    - `python3 -O -m enjarify.main yourapp.apk`
    - 已设置好环境变量后
      - `enjarify yourapp.apk`
      - `enjarify classes2.dex`
      - `enjarify yourapp.apk -o yourapp.jar`
  - 主页
    - Github
      - Storyyeller/enjarify
        - 旧：google/enjarify
  - 下载
    - Releases · Storyyeller/enjarify
  - 用法

```
python3 -O -m enjarify.main yourapp.apk

enjarify yourapp.apk

enjarify classes2.dex

enjarify yourapp.apk -o yourapp.jar
```

为何不用 `dex2jar` ？

- `dex2jar`
  - 比较旧了
  - 大部分情况是转换没问题
  - 但是
    - 有时候
      - 模糊特性时
      - 边缘情况时
    - 转换
      - 会报错
      - 甚至不报错但生成的是错误的结果
- `Enjarify`
  - 最新设计的工具
  - 支持绝大多数（尽可能多的）情况
    - 包括有些dex2jar会出错的情况
    - 且额外支持
      - Unicode的类名
      - 用作多种类型的常量

- 隐式转换
- 正常执行流程中的异常处理
- 引用了非常多的常量的类
- 名字非常长的方法
- 捕获函数后的异常处理
- 错误类型的静态初始变量

Dedexer

- 主页
  - Dedexer user's manual
    - 注：此工具太老了，最后更新时间是2011.12.05
- 下载
  - dedexer - Browse Files at SourceForge.net
- 作用
  - 把dex转换为类似于汇编的格式
    - 目的：从dex文件创建类似于Jasmin的源代码
- 用法
  - `java -jar ddx.jar -o -D -d <destination directory> <source>`
  - `java -jar ddx1.5.jar -o -D -d c:\dex\gen c:\dex\classes.dex`
- 举例

```
D:\WINDOWS\system32>
java -jar ddx1.5.jar -o -D -d c:\dex\gen c:\dex\classes.dex
Processing com/eoeandroid/market/MarketActivity$2
Processing com/eoeandroid/market/MarketActivity$1
```

dex转smali

baksmali

dex转java

jadx

详见：jadx

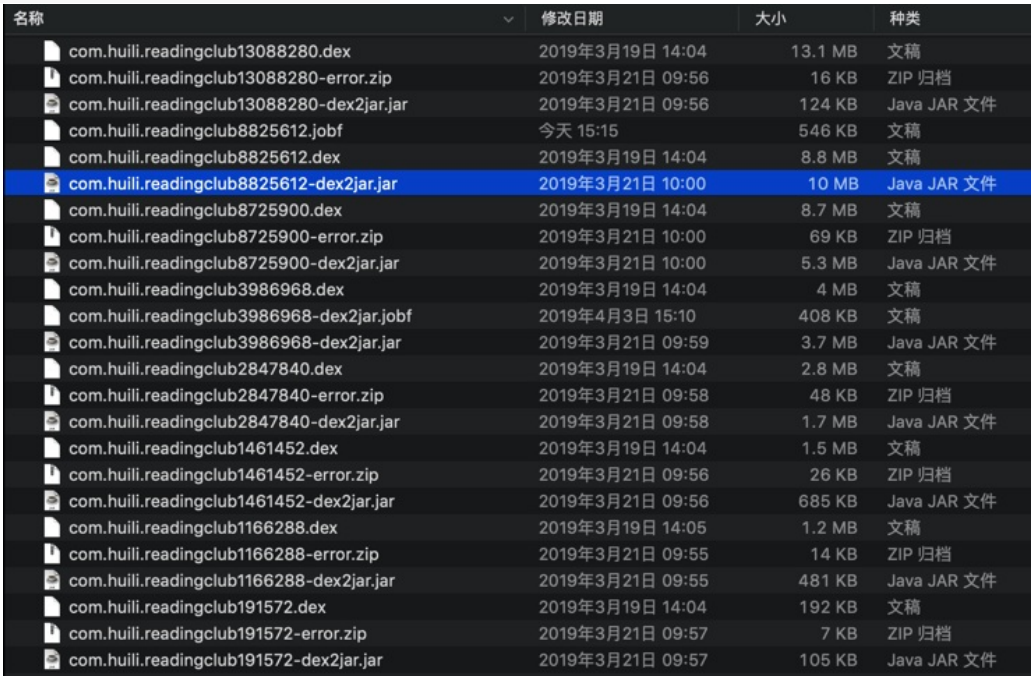
GDA

详见：GDA

针对jar

jar转java

此处主要讨论 `jar` 转 `java` 的反编译器：

- 输入： `dex` 文件
  - 前面一步从 `dex` 转换得到的 `jar` 文件
    - 举例
      - `com.huili.readingclub8825612-dex2jar.jar`
      - 

- 反编译
  - 选择合适的反编译器
  - 从 `jar` 反编译出 `java` 代码
- 输出： `java` 源代码

java反编译器对比

- 概述
  - 如果是 `jar` 转 `java` ：选 `Procyon`
    - 或基于 `Procyon` 的GUI工具： `Luyten`
  - 如果是 `apk` / `dex` 转 `java` ：选 `jadx`
- 详解
  - 关于 `JD-GUI` ：不好用
    - 网上很多人提到了用的比较广的： `JD-GUI`
      - 经过实测，基本够用，但不够完美
        - 有些文件转换会出错
        - 且代码逻辑也不够清晰
  - 后续自己测试了多个其他的反编译器
    - `Procyon` 、 `CFR` 、 `JD-GUI` 等等
  - 最终结论如下：
    - `jar` 转 `java` 的反编译效果： `Jadx` > `Procyon` > `CFR` > `JD-GUI`
      - `Procyon`
        - 特点
          - 代码转换不出错的
        - 用来
          - 查看代码：用基于Procyon的Luyten去查看代码
          - 导出代码：用Procyon去从jar反编译转换出java源代码
    - `apk` / `dex` 转 `java` ，优先选用 `jadx`
      - `jadx`
        - 特点
          - 代码转换不仅不出错（或者很少出错）
          - 关键是代码逻辑更加清晰和易懂
            - 大大提升代码质量，使得代码更加易读
        - 用来
          - 直接用jadx打开未加固的apk
            - 即可查看和导出java源代码
          - 打开（用FDex2从加固了的apk导出的）dex文件
            - 查看和导出java源代码

常用java反编译器

网上有很多安卓相关的java反编译器，大致整理如下：

- 比较老旧的
  - Jad
    - 不再维护了，源码仓库已关闭
    - 不支持Java 5+
  - Java DeObfuscator
    - https://sourceforge.net/projects/jdo/
    - JDO is a Java DeObfuscator that works on class files directly. JDO contains a simple and easy to use GUI that makes automatic deobfuscation of Java projects a one-click operation!
  - JODE
    - http://jode.sourceforge.net/
    - a java package containing a decompiler and an optimizer for Java. This package is freely available under the GNU GPL. It hasn't been updated for quite some time.
  - AndroChef
    - http://www.androiddecompiler.com/
    - =
    - http://www.neshkov.com/ac_decompiler.html
    - 只支持Windows平台
  - Candle
    - https://github.com/bradsdavis/candle-decompiler
    - by Brad Davis, developer of JBoss Cake, is an early but promising work in progress= is far away from being feature complete
- 相对新的工具
  - JD-GUI
    - 官网地址：
      - http://jd.benow.ca/
      - -》
      - http://java-decompiler.github.io
    - is an Decompiler, which comes with its own GUI. All is licensed under GPLv3. Like CFR the source for the decompiler itself, is not published, but you have the right to decompile the bionaries. And the binaries are under an OpenSource-License (CFR is under the MIT-license and JD Core is under the GPLv3 license)
    - 转换效果：经常会报错
      - 更准确的说是：对于LINQ/DLR的树编译器产生的代码，会不支持，会报错
        - -》解析后的代码中，包含/ *Error* / // Byte code 这种代码
  - CFR
    - 官网
      - CFR - yet another java decompiler.
      - http://www.benf.org/other/cfr/
    - 特点
      - 支持java 9/10/12等
    - by Lee Benfield is well on its way to becoming the premier Java Decompiler. Lee and I actually work for the same company and share regression tests. We're engaged in a friendly competition to see who can deliver a better decompiler. Based on his progress thus far, there's a very good chance he will win--at least on decompiling obfuscated code
  - Krakatau
    - https://github.com/Storyyeller/Krakatau
    - by Robert Grosse, written in Python, includes a robust verifier. It focuses on translating arbitrary bytecode into valid Java code, as opposed to reconstructing the original code.

- Fernflower
  - https://github.com/JetBrains/intellij-community/tree/master/plugins/java-decompiler/engine
  - https://github.com/fesh0r/fernflower
  - an analytical Java decompiler
- Cavaj
  - https://cavaj-java-decompiler.jaleco.com/
- Procyon
  - mstrobel / Procyon — Bitbucket
  - https://bitbucket.org/mstrobel/procyon
    - 官网使用文档
      - mstrobel / Procyon / wiki / Java Decompiler — Bitbucket
      - https://bitbucket.org/mstrobel/procyon/wiki/Java%20Decompiler
  - 重点关注Java 5之后的特性支持
    - 这些是其他很多反编译工具不支持的，会报错的
  - 更详细的解释
    - 枚举声明
    - 枚举和字符串的switch表达式
      - 目前只测试支持javac 1.7
    - 局部类Local classes
      - 匿名和带名字的都支持
    - 注解/标注Annotations
    - Java 8的Lambdas和方法引用(比如 :: 操作符)
  - -》对很多人了来说，比较关注：支持java8
  - Procyon本身是命令行工具
  - 基于Procyon的带GUI图形界面的工具
    - SecureTeam Java Decompiler
      - http://www.secureteam.net/Java-Decompiler.aspx
      - A JavaFX-based decompiler front-end with fast and convenient code navigation. Download it, or launch it directly from your browser.
    - Luyten
      - https://github.com/deathmarine/Luyten
      - An open source front-end by deathmarine
    - Bytecode Viewer
      - https://github.com/Konloch/bytecode-viewer
      - -》
      - https://bytecodeviewer.com
      - an open source Java decompilation, disassembly, and debugging suite by @Konloch. It can produce decompiled sources from several modern Java decompilers, including Procyon, CFR, and FernFlower.
    - Helios
      - https://github.com/samczsun/Helios
      - similar to Bytecode Viewer. But is a completely new and independent project, which uses SWT instead of Swing.
    - Enigma
      - http://www.cuchazinteractive.com/enigma/
      - Originally used to deobfuscate Minecraft versions. Uses Procyon internally.
      - 作者已不再维护
- Jadx
  - 也支持从jar查看java代码
    - 和导出全部代码
      - 导出方式还有2种
        - 保存全部代码
        - 和以Gradle的方式导出源码

        - 如果打开的是apk文件
        - 则导出了dex对应的java源码外，还有assets等资源和其他文件
        - -》更加利于你得到更接近apk的原始代码的项目结构
    - 同时还支持直接打开apk
        - 查看apk中的各种文件
            - 包括java源代码

`JD-GUI` VS `CFR` VS `Procyon` VS `Jadx`

对于同一个jar包：`com.huili.readingclub.jar`

用不同工具：

- JD-GUI
- CFR
- Procyon
- Jadx

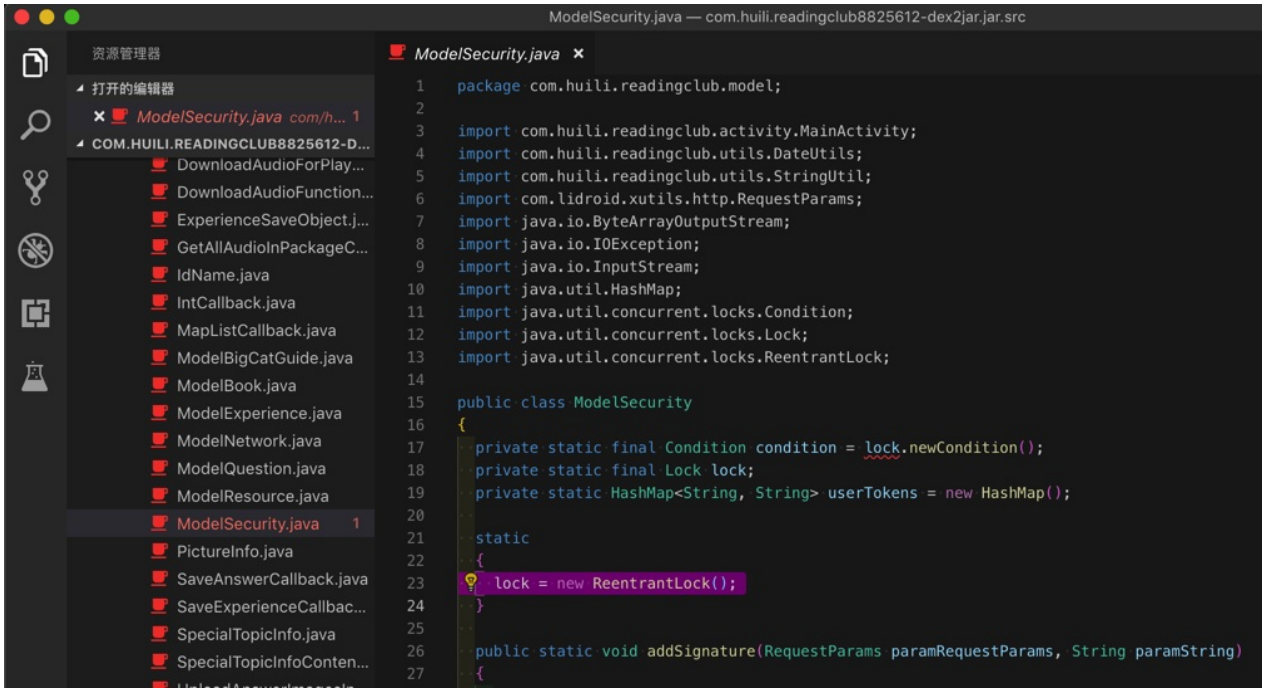
去导出java源代码后，转换导出的效果，尤其是准确性，是否出错，是不一样的。

转换的细节是否完美

jd-gui的细节不够好的地方：

static函数：

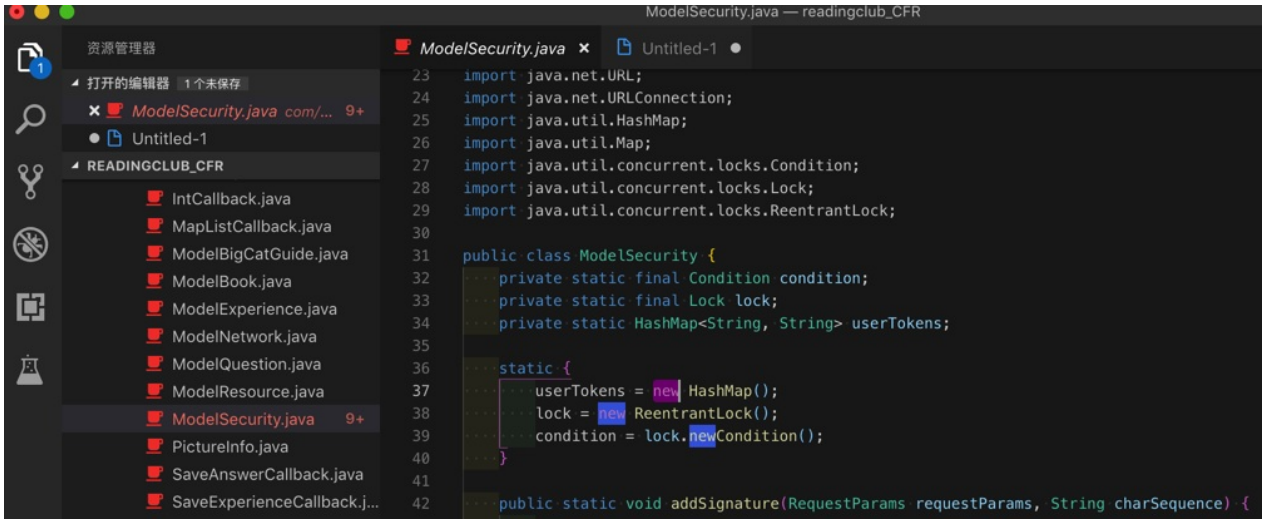
```
static
{
  lock = new ReentrantLock();
}
```



而CFR、Procyon、Jadx转换结果可以更完整：

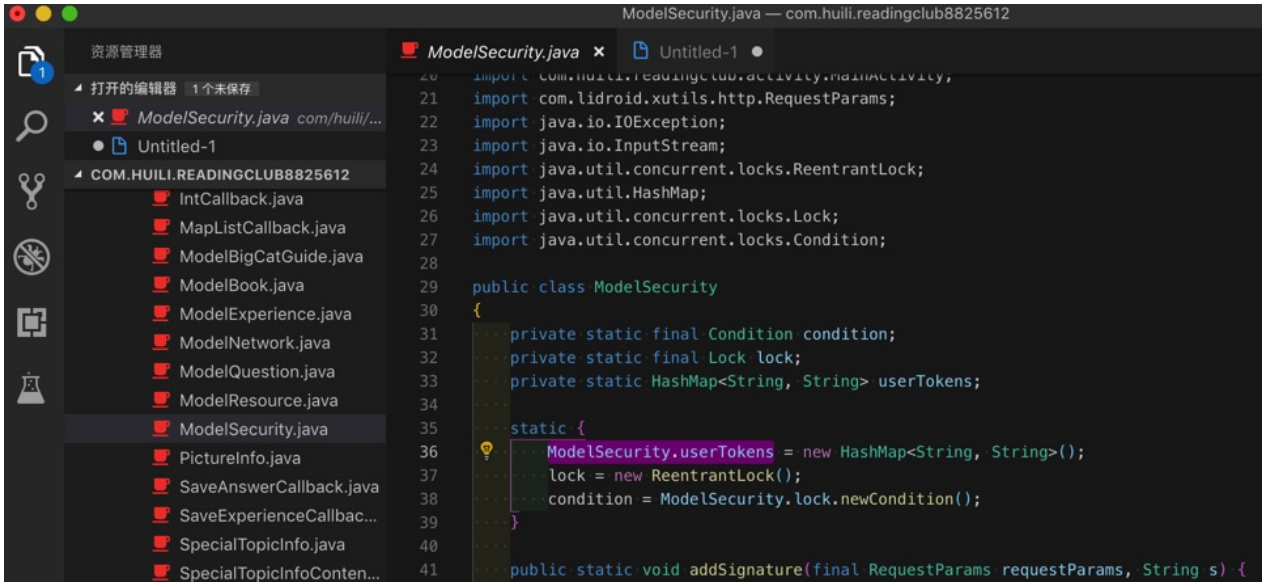
CFR的：

```
static {
    userTokens = new HashMap();
    lock = new ReentrantLock();
    condition = lock.newCondition();
}
```



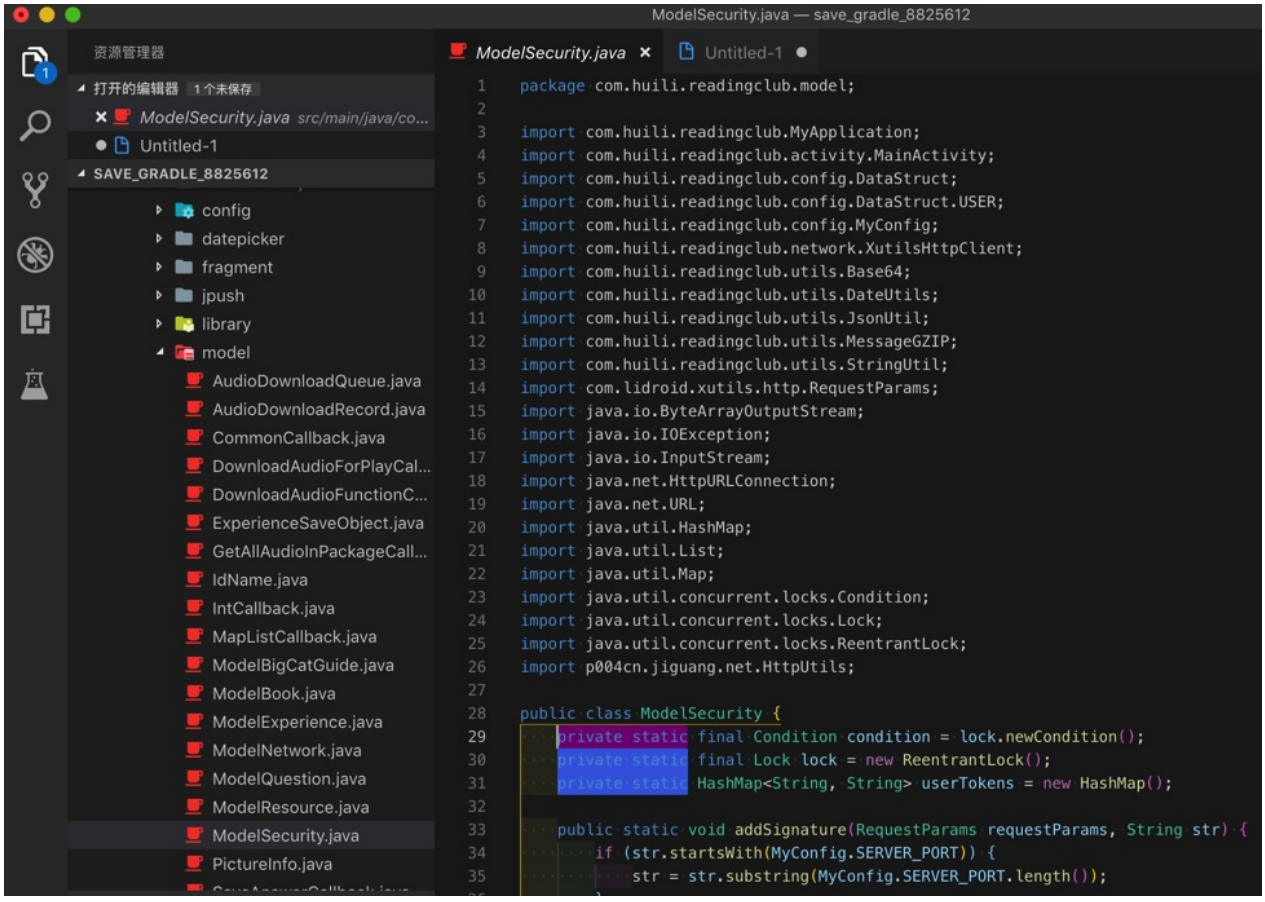
Procyon

```
static {
    ModelSecurity.userTokens = new HashMap<String, String>();
    lock = new ReentrantLock();
    condition = ModelSecurity.lock.newCondition();
}
```



Jadx

```
public class ModelSecurity {
    private static final Condition condition = lock.newCondition();
    private static final Lock lock = new ReentrantLock();
    private static HashMap<String, String> userTokens = new HashMap();
}
```



很明显从：

- CFR的：`userTokens = new HashMap();`
- Procyon的：`ModelSecurity.userTokens = new HashMap<String, String>();`
- Jadx的：`private static HashMap<String, String> userTokens = new HashMap();`

可以看出：

- `Procyon` 能识别static变量，细节转换的很完美和精确
- 而 `Jadx` ： 更进一步识别出是private的static类型的变量，更加准确。

对于细节转换的结论：

- `JD-GUI` ： 细节不够好
- `CFR` ： 细节基本满足要求
- `Procyon` ： 细节完美转换
- `Jadx` ： 不仅完美且代码变量和结构更合理

转换是否出错及代码逻辑清晰度

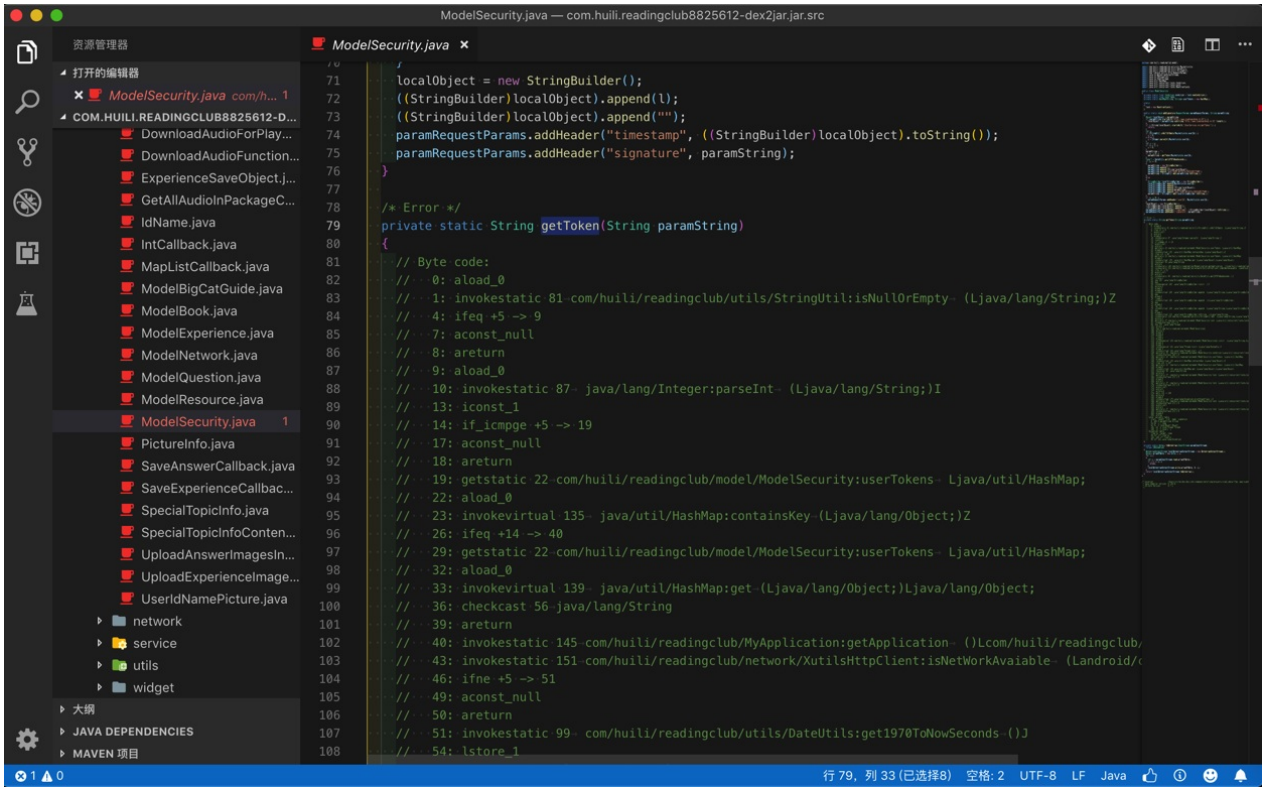
getToken函数转换效果

比如： `com.huili.readingclub.model.ModelSecurity` 的 `getToken`

JD-GUI某函数转换报错得不到源码

都是错误代码：

```
  /* Error */
  private static String getToken(String paramString)
  {
    // Byte code:
    //   0: aload_0
    //   1: invokestatic 81    com/huili/readingclub/utils/StringUtil:isNullOrEmpty    (Ljava/lang/String;)Z
    //   4: ifeq +5 -> 9
    //   7: aconst_null
    //   8: areturn
    //   9: aload_0
    //   10: invokestatic 87    java/lang/Integer:parseInt    (Ljava/lang/String;)I
    //   13: iconst_1
    //   14: if_icmpge +5 -> 19
    //   17: aconst_null
    //   18: areturn
    //   19: getstatic 22    com/huili/readingclub/model/ModelSecurity:userTokens    Ljava/util/HashMap;
    //   22: aload_0
......
```



CFR某函数转换报错但有源码

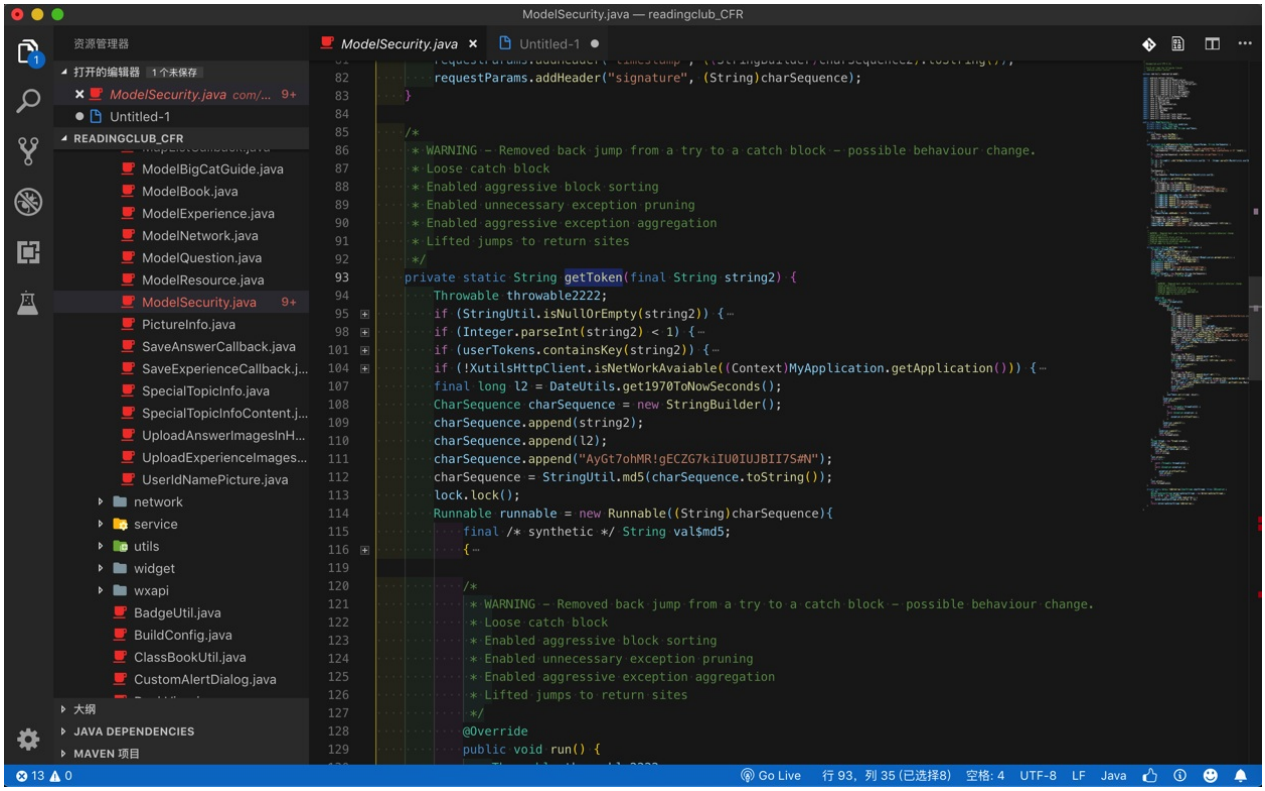
转换报错：

summary.txt

```
com.huili.readingclub.model.ModelSecurity
**************************
getToken(java.lang.String )
  Loose catch block
run()
  Loose catch block
```

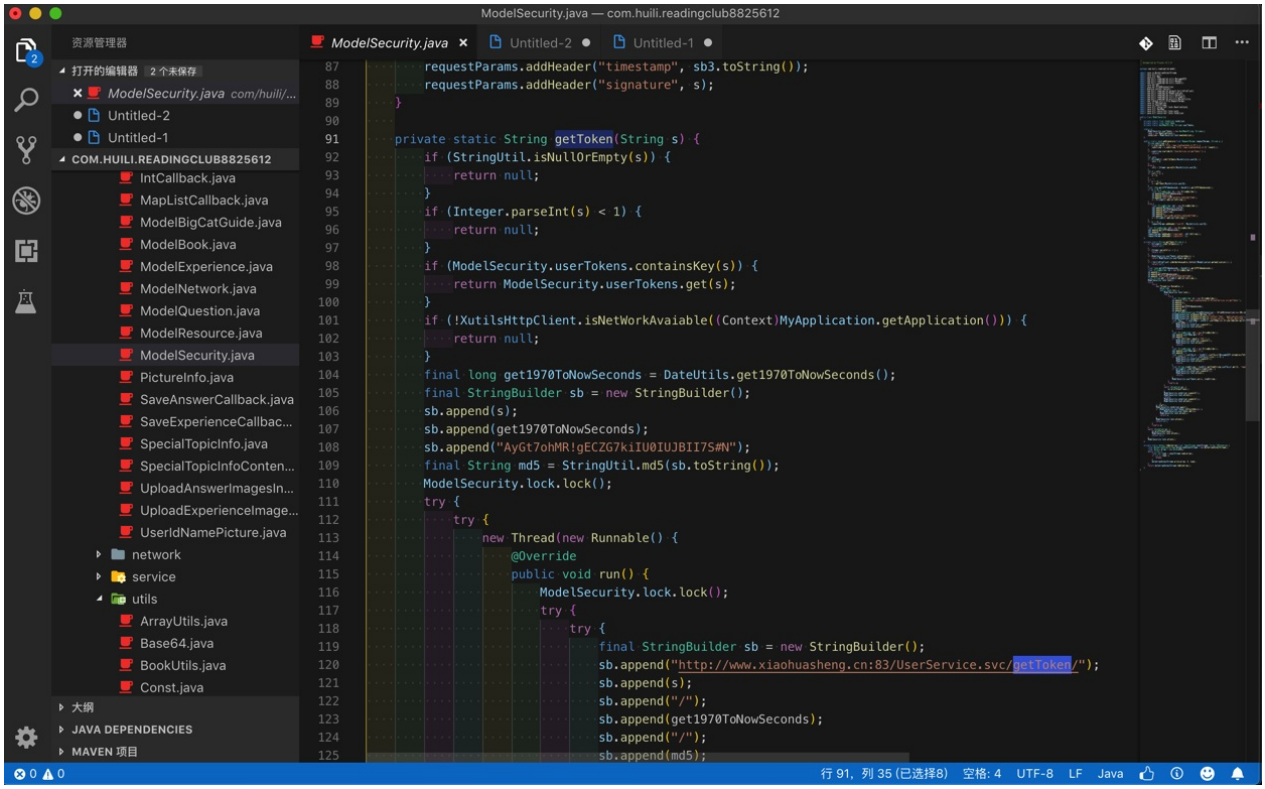
代码中有报错：

```
    /*
     * WARNING - Removed back jump from a try to a catch block - possible behaviour change.
     * Loose catch block
     * Enabled aggressive block sorting
     * Enabled unnecessary exception pruning
     * Enabled aggressive exception aggregation
     * Lifted jumps to return sites
     */
    private static String getToken(final String string2) {
        ''''''
        lock.lock();
        Runnable runnable = new Runnable((String)charSequence){
            ''''''
            /*
             * WARNING - Removed back jump from a try to a catch block - possible behaviour change.
             * Loose catch block
             * Enabled aggressive block sorting
             * Enabled unnecessary exception pruning
             * Enabled aggressive exception aggregation
             * Lifted jumps to return sites
             */
            @Override
            public void run() {
                Throwable throwable2222;
```



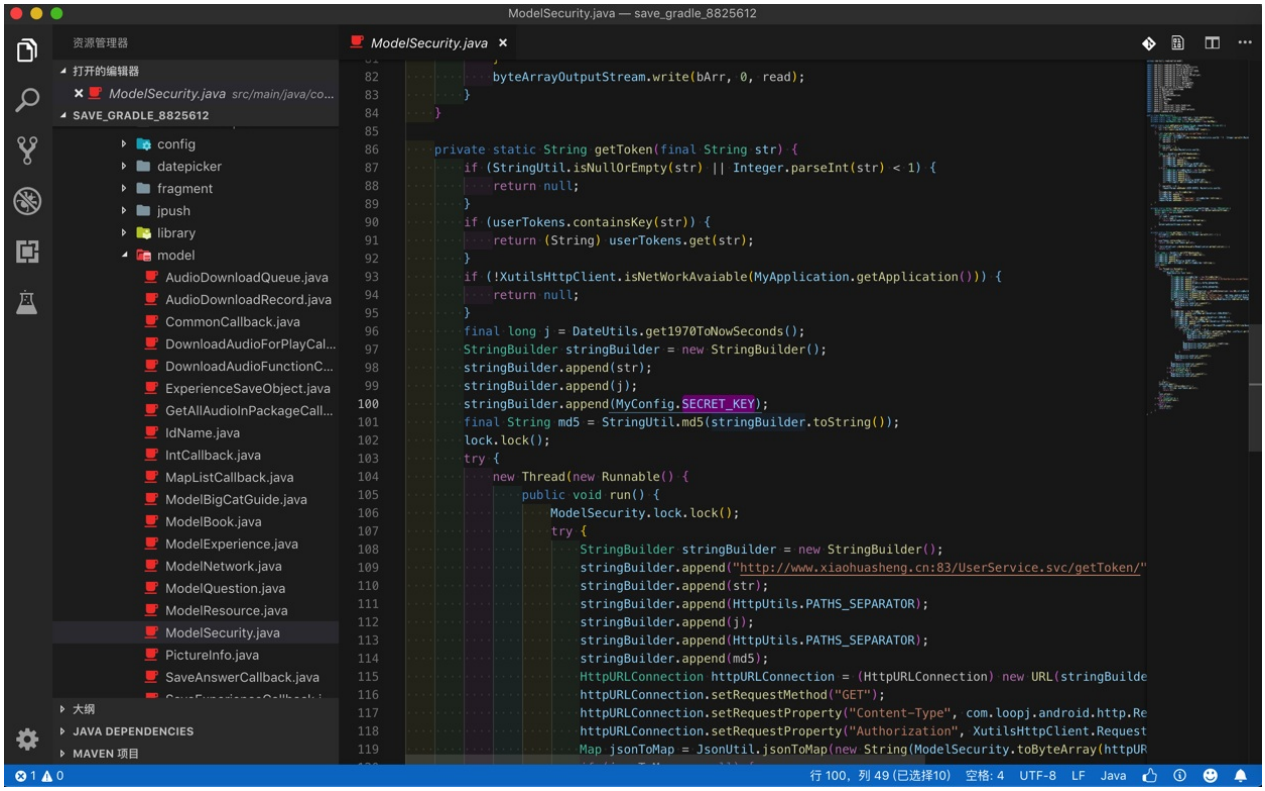
Procyon某函数完美转换无错误：

```
//
// Decompiled by Procyon v0.5.34
//
    private static String getToken(String s) {
        if (StringUtil.isNullOrEmpty(s)) {
            return null;
        }
        ...
        sb.append("AyGt7ohMR!gECZG7kiIU0IUJBII7S#N");
        ModelSecurity.lock.lock();
        try {
            try {
                new Thread(new Runnable() {
                    @Override
                    public void run() {
                        ModelSecurity.lock.lock();
                        try {
                            try {
                                final StringBuilder sb = new StringBuilder();
                                ...
                                sb.append("/");
```



Jadx某函数完美转换外，还保持代码变量和结构更合理 -> 能识别常量定义和引用

```
private static String getToken(final String str) {
    if (StringUtil.isNullOrEmpty(str) || Integer.parseInt(str) < 1) {
        return null;
    }
    ......
    stringBuilder.append(MyConfig.SECRET_KEY);
    final String md5 = StringUtil.md5(stringBuilder.toString());
    lock.lock();
    try {
        new Thread(new Runnable() {
            public void run() {
                ModelSecurity.lock.lock();
                try {
                    StringBuilder stringBuilder = new StringBuilder();
                    ...
                    stringBuilder.append(UrlUtils.PATHS_SEPARATOR);
```



其中包括 `stringBuilder.append` 的参数是

常量定义 `MyConfig.SECRET_KEY`

而不是之前代码的常量的值：

`"AyGt7ohMR!gECZG7kiIU0IUJBII7S#N"`

`getMD5Str` 的 `char` 的 `list` 转换效果

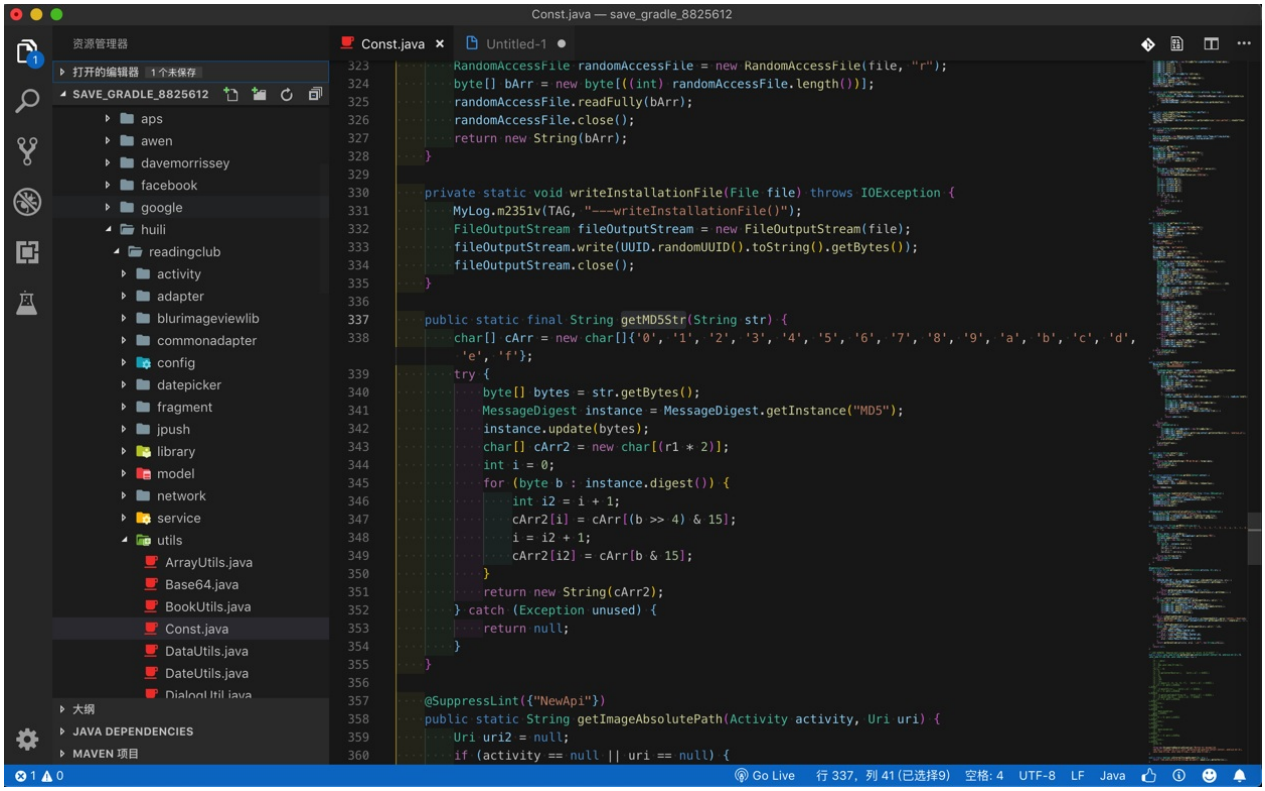
另外，再去对比：

Jadx导出的代码的逻辑和结构，非常清晰：

char的list很清楚，以及赋值语句：

```
cArr2[i] = cArr[(b >> 4) & 15];
```

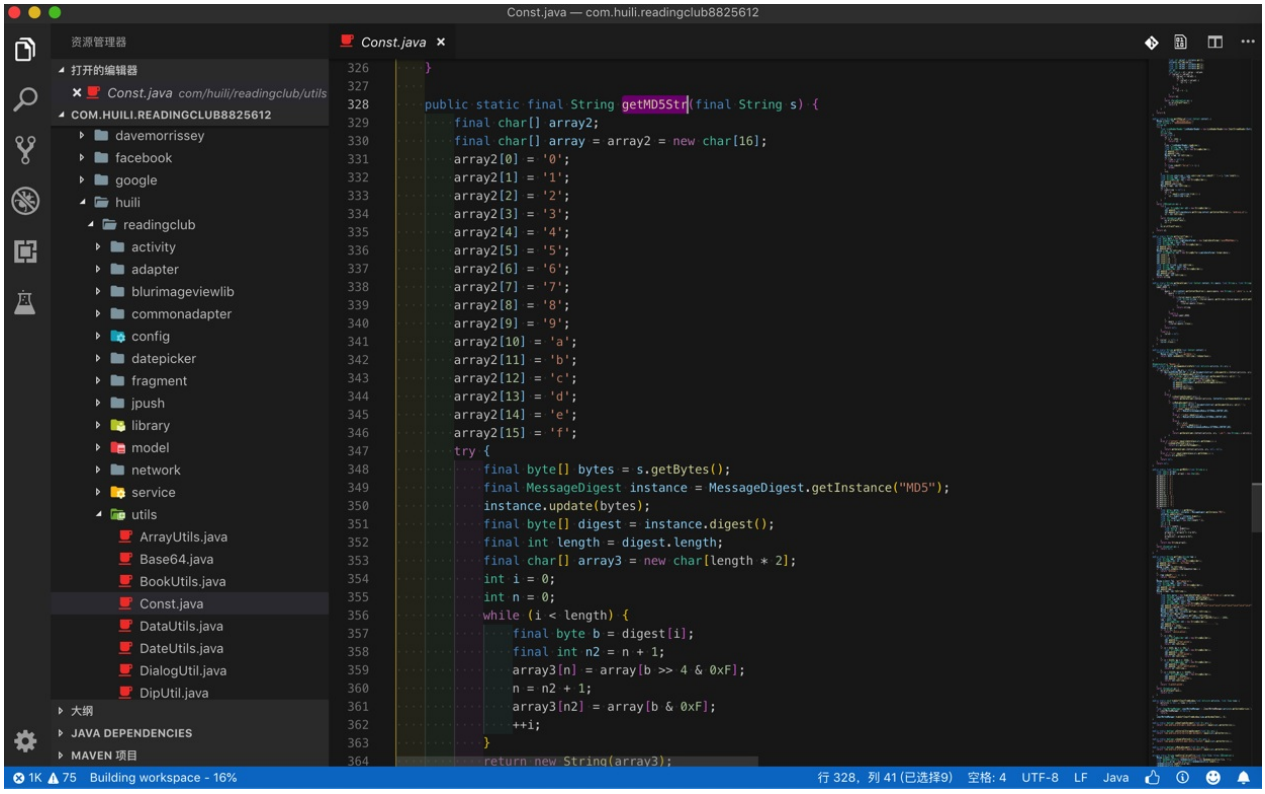
也容易看懂




```
public static final String getMD5Str(String str) {
    char[] cArr = new char[]{'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'};
    try {
        byte[] bytes = str.getBytes();
        MessageDigest instance = MessageDigest.getInstance("MD5");
        instance.update(bytes);
        char[] cArr2 = new char[(r1 * 2)];
        int i = 0;
        for (byte b : instance.digest()) {
            int i2 = i + 1;
            cArr2[i] = cArr[(b >> 4) & 15];
            i = i2 + 1;
            cArr2[i2] = cArr[b & 15];
        }
        return new String(cArr2);
    } catch (Exception unused) {
        return null;
    }
}
```

而不是像：

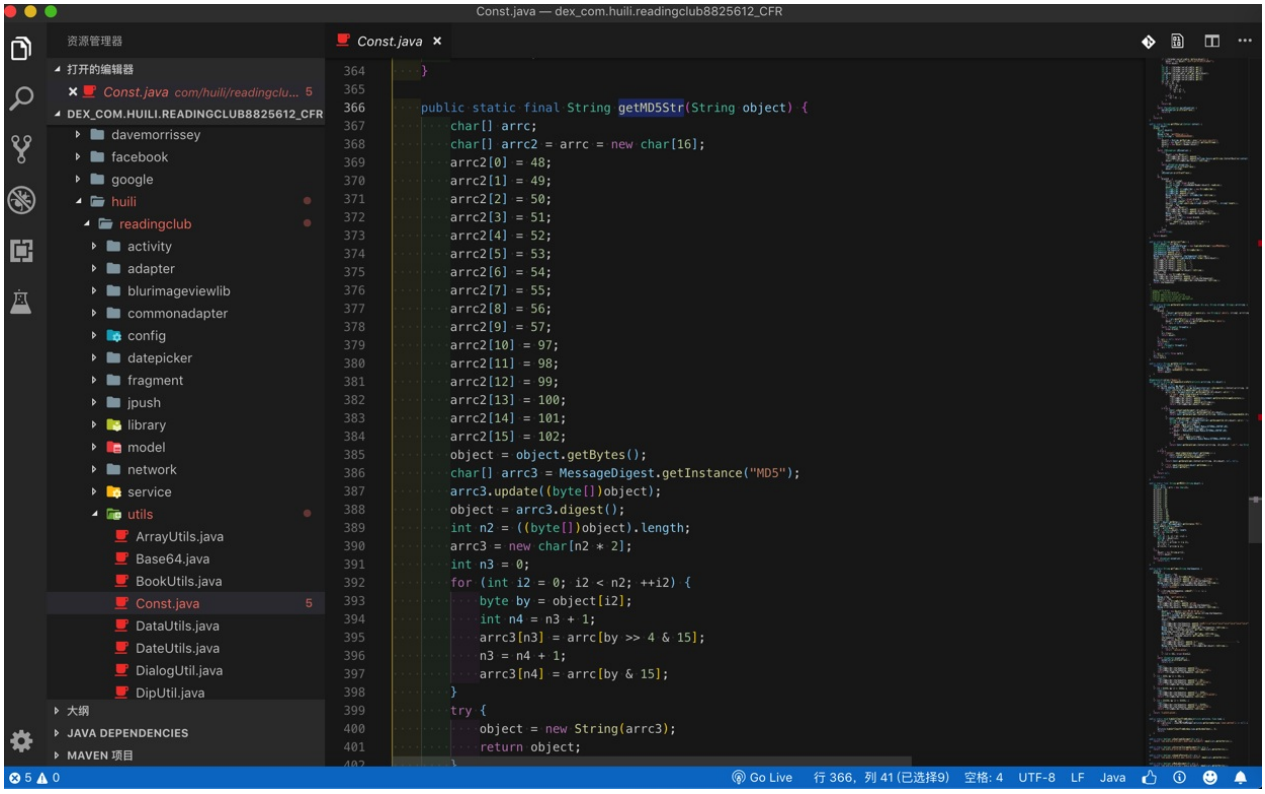
Procyon（的Luyten）的代码：




```java
public static final String getMD5Str(final String s) {
    final char[] array2;
    final char[] array = array2 = new char[16];
    array2[0] = '0';
    array2[1] = '1';
    array2[2] = '2';
    array2[3] = '3';
    array2[4] = '4';
    array2[5] = '5';
    array2[6] = '6';
    array2[7] = '7';
    array2[8] = '8';
    array2[9] = '9';
    array2[10] = 'a';
    array2[11] = 'b';
    array2[12] = 'c';
    array2[13] = 'd';
    array2[14] = 'e';
    array2[15] = 'f';
    try {
        final byte[] bytes = s.getBytes();
        final MessageDigest instance = MessageDigest.getInstance("MD5");
        instance.update(bytes);
        final byte[] digest = instance.digest();
        final int length = digest.length;
        final char[] array3 = new char[length * 2];
        int i = 0;
        int n = 0;
        while (i < length) {
            final byte b = digest[i];
            final int n2 = n + 1;
            array3[n] = array[b >> 4 & 0xF];
            n = n2 + 1;
            array3[n2] = array[b & 0xF];
            ++i;
        }
        return new String(array3);
    }
    catch (Exception ex) {
        return null;
    }
}
```

作为一个char的list，还是没有清晰的表达出来

更不像是CFR的：

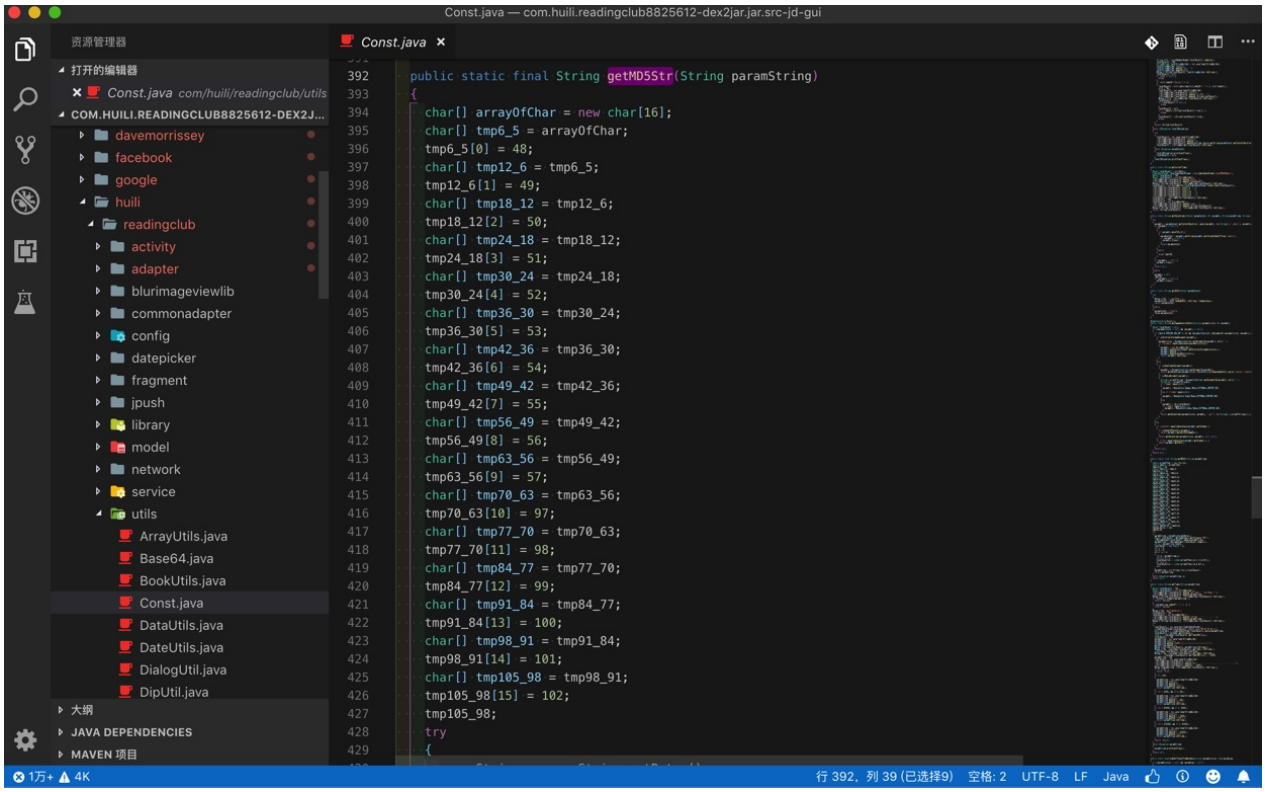



```java
public static final String getMD5Str(String object) {
    char[] arrc;
    char[] arrc2 = arrc = new char[16];
    arrc2[0] = 48;
    arrc2[1] = 49;
    arrc2[2] = 50;
    arrc2[3] = 51;
    arrc2[4] = 52;
    arrc2[5] = 53;
    arrc2[6] = 54;
    arrc2[7] = 55;
    arrc2[8] = 56;
    arrc2[9] = 57;
    arrc2[10] = 97;
    arrc2[11] = 98;
    arrc2[12] = 99;
    arrc2[13] = 100;
    arrc2[14] = 101;
    arrc2[15] = 102;
    object = object.getBytes();
    char[] arrc3 = MessageDigest.getInstance("MD5");
    arrc3.update((byte[])object);
    object = arrc3.digest();
    int n2 = ((byte[])object).length;
    arrc3 = new char[n2 * 2];
    int n3 = 0;
    for (int i2 = 0; i2 < n2; ++i2) {
        byte by = object[i2];
        int n4 = n3 + 1;
        arrc3[n3] = arrc[by >> 4 & 15];
        n3 = n4 + 1;
        arrc3[n4] = arrc[by & 15];
    }
    try {
        object = new String(arrc3);
        return object;
    }
    catch (Exception exception) {
        return null;
    }
```

```
    }
```

连char的list都不够明显，只是char的int数值。

更更不像是 `JD-GUI` 导出的源码：




```
public static final String getMD5Str(String paramString)
{
  char[] arrayOfChar = new char[16];
  char[] tmp6_5 = arrayOfChar;
  tmp6_5[0] = 48;
  char[] tmp12_6 = tmp6_5;
  tmp12_6[1] = 49;
  char[] tmp18_12 = tmp12_6;
  tmp18_12[2] = 50;
  char[] tmp24_18 = tmp18_12;
  tmp24_18[3] = 51;
  char[] tmp30_24 = tmp24_18;
  tmp30_24[4] = 52;
  char[] tmp36_30 = tmp30_24;
  tmp36_30[5] = 53;
  char[] tmp42_36 = tmp36_30;
  tmp42_36[6] = 54;
  char[] tmp49_42 = tmp42_36;
  tmp49_42[7] = 55;
  char[] tmp56_49 = tmp49_42;
  tmp56_49[8] = 56;
  char[] tmp63_56 = tmp56_49;
  tmp63_56[9] = 57;
  char[] tmp70_63 = tmp63_56;
  tmp70_63[10] = 97;
  char[] tmp77_70 = tmp70_63;
  tmp77_70[11] = 98;
  char[] tmp84_77 = tmp77_70;
  tmp84_77[12] = 99;
  char[] tmp91_84 = tmp84_77;
  tmp91_84[13] = 100;
  char[] tmp98_91 = tmp91_84;
  tmp98_91[14] = 101;
  char[] tmp105_98 = tmp98_91;
  tmp105_98[15] = 102;
  tmp105_98;
  try
```

```
    {
      paramString = paramString.getBytes();
      Object localObject = MessageDigest.getInstance("MD5");
      ((MessageDigest)localObject).update(paramString);
      paramString = ((MessageDigest)localObject).digest();
      int i = paramString.length;
      localObject = new char[i * 2];
      int j = 0;
      int k = 0;
      while (j < i)
      {
        int m = paramString[j];
        int n = k + 1;
        localObject[k] = ((char)arrayOfChar[(m >> 4 & 0xF)]);
        k = n + 1;
        localObject[n] = ((char)arrayOfChar[(m & 0xF)]);
        j++;
      }
      paramString = new String((char[])localObject);
      return paramString;
    }
    catch (Exception paramString) {}
    return null;
  }
```

连char的list不仅不够明显，不仅只是char的int数值，而且还有多余的赋值，影响代码逻辑的理解，以及对应的数据赋值：

```
localObject[k] = ((char)arrayOfChar[(m >> 4 & 0xF)]);
```

都很晦涩难懂。

对于几种反编译抓换代码的出错程度和代码逻辑是否完美的结论是

- `JD-GUI`： 某函数转换报错得不到源码
- `CFR`： 某函数转换报错但有源码
- `Procyon`： 某函数完美转换无错误
- `Jadx`： 某函数完美转换外还能识别常量定义和代码结构更清晰

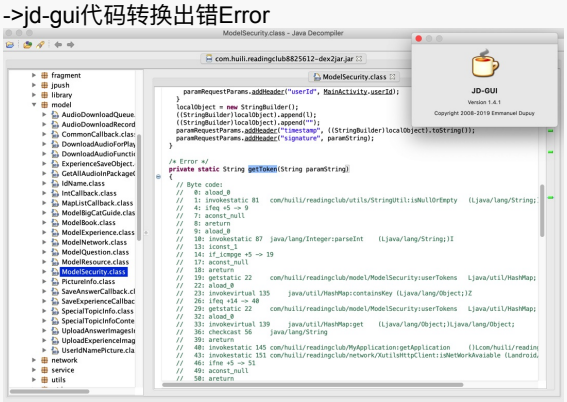
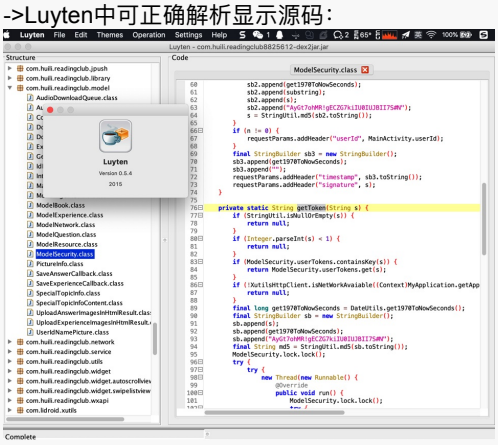
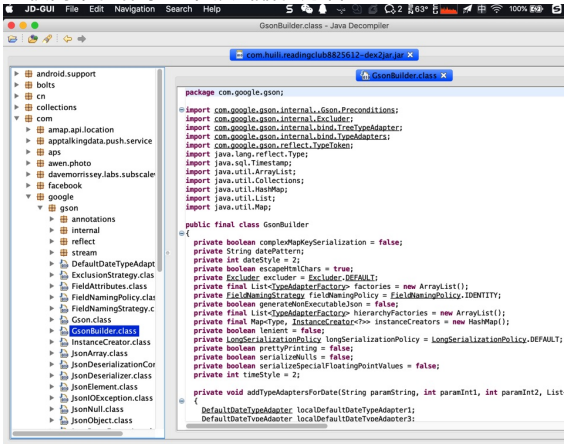
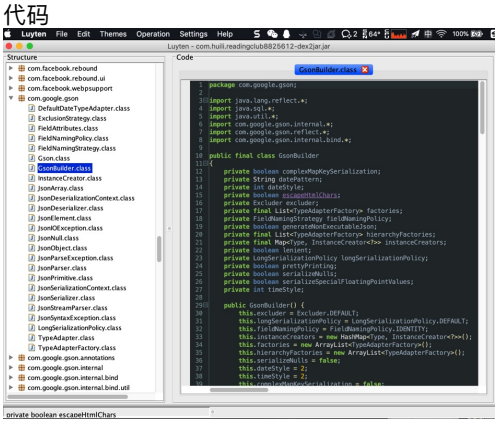
对比结果

Luyten vs Procyon

从jar包导出代码：

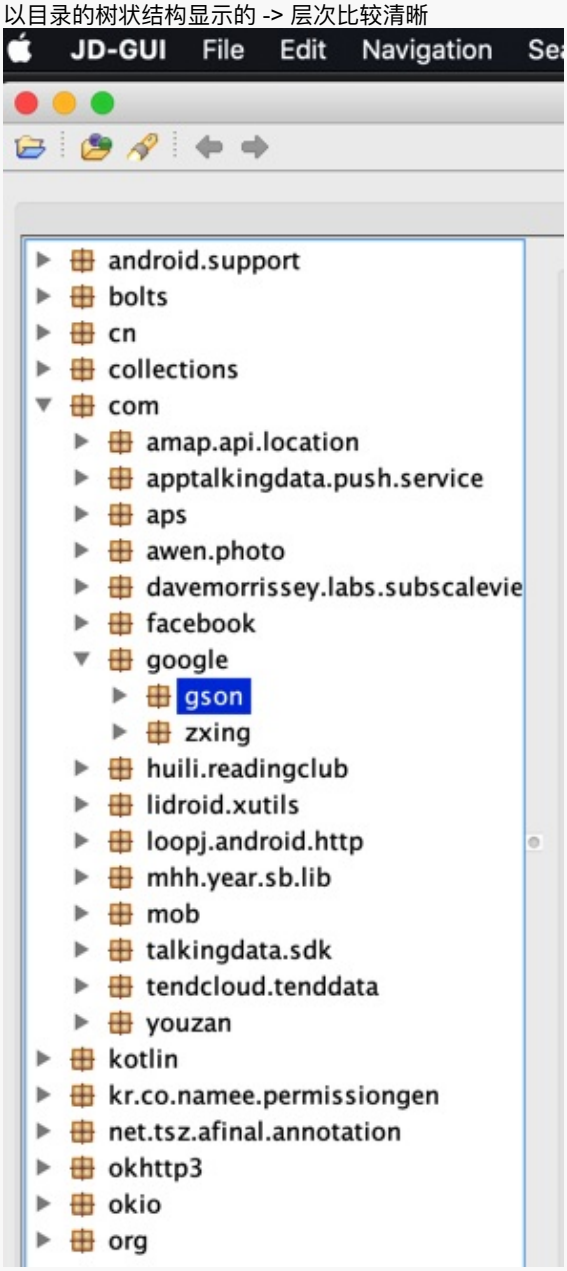
- Luyten
  - 带GUI图形界面：可直接查看代码
  - 也可用来Save all导出全部代码
    - 但是速度比较慢
      - 当代码很大时：几十分钟还没导出完毕
- Procyon
  - 不带界面，只是一个jar包：
    - `procyon-decompiler-0.5.34.jar`
  - 可直接用来从你的jar包导出源码
    - 当代码很大时：速度很快，1分钟左右即可搞定

`JD-GUI` vs (基于 `Procyon` 的) `Luyten`

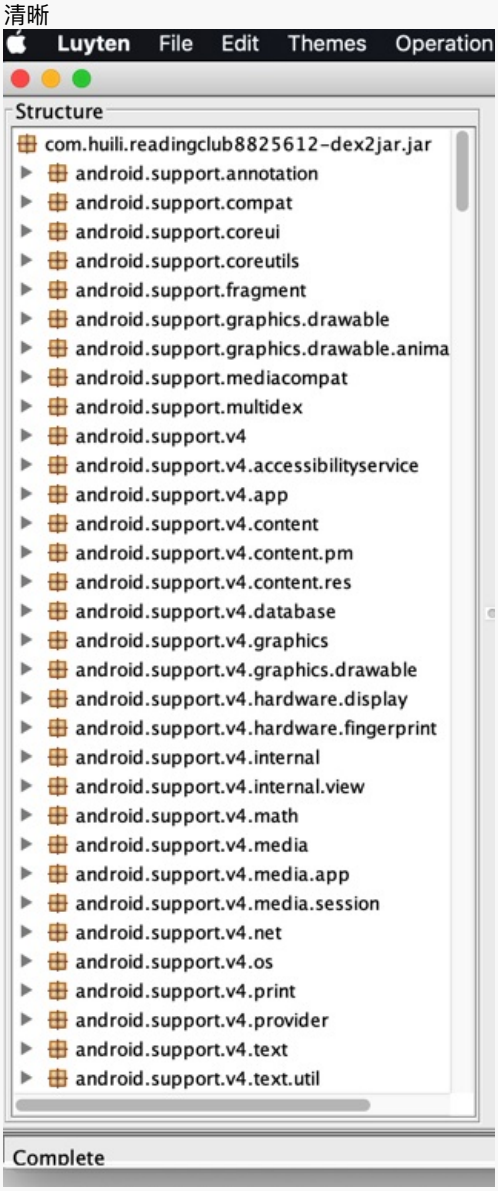
对比项	JD-GUI	Luyten
流行程度	比较广->大家用的比较多	一般->相对使用的人不是很多
jar转java的准确率	高 ->jd-gui代码转换出错Error 	很高 ->Luyten中可正确解析显示源码： 
显示：整体UI界面和代码高亮	比较简单和朴素->不够炫酷和好看 	好看，而且支持多种语法高亮效果 -> 更方便看代码 

显示：包/类的显示结构

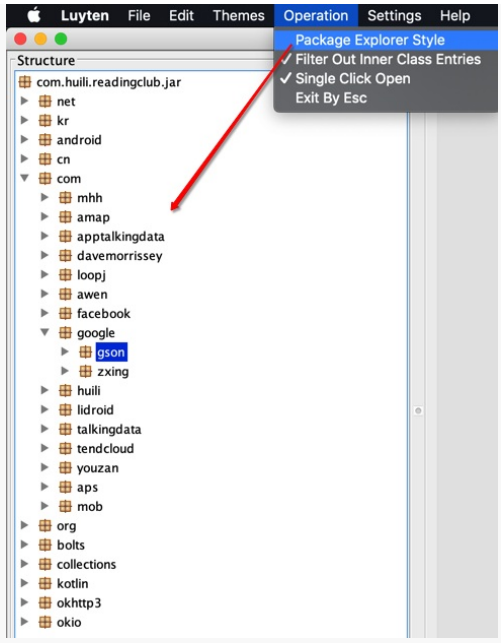
以目录的树状结构显示的 -> 层次比较清晰



（默认）平铺直叙，直接显示的 ->层次结构不够清晰



后记：
后来发现，把默认勾选的选项 `Operation -> Package Explorer Style` 取消勾选，也可以按照树状显示类和包名了：



`JD-GUI` VS `CFR` VS `Procyon` VS `Jadx` 对比总结

下面从整体总结和对比这几个反编译的反编译的效果：

java反编译器	JD-GUI	CFR	Procyon	Jadx
转换出错程度	比较多	少许	很少	极少
出错状态和相关信息	会显示： `/* Error */ // Byte code`	输出转换期间出错的地方到文件： `summary.txt`	会显示： `This method could not be decompiled. Original Bytecode`	
转换出的代码的质量	不是很好，细节不够好	部分细节转换的略有瑕疵 但是还是可以看到基本代码逻辑的	完美转换细节 代码中字符数组定义等内容可以正确转换，但是逻辑不清晰 比如只是能转换出常量解析后的字符串值，而无法识别出是常量的引用	不仅转换完美无错，而且代码逻辑更准确和完整 代码中的常量的引用都能完美还原出来 代码中字符数组定义等内容可以完美转换
转换出的文件是否有标识	无	有，顶部有标识： Decompiled with CFR 0.141 并且还能列出具体出错的类： Could not load the following classes	有，顶部有标识： Decompiled by Procyon v0.5.34	无

所以最终结论就是：

- 以后**尽量用** `Jadx`
  - 直接用jadx打开未加固的apk
    - 即可查看和导出java源代码
  - 打开（用FDex2从加固了的apk导出的）dex文件
    - 查看和导出java源代码
- 其次**考虑用** `Procyon` （或基于Procyon的GUI工具 `Luyten` ）
  - 查看代码：用基于Procyon的 `Luyten` 去查看代码
  - 导出代码：用Procyon去从jar反编译转换出java源代码

常见java反编译器

Procyon

- `Procyon`
  - 是什么：反编译器
  - 作用：从 `jar` 中反编译导出 `java` 源码
  - 语法

    ```
    java -jar /path/to/your/procyon-decompiler-0.5.36.jar -jar jav_file.jar -o output_folder
    ```

    - 举例

      ```
      java -jar  Procyon/procyon-decompiler-0.5.36.jar -jar ../../dex_to_jar/com.ishowedu.child.peiyin8392664-dex2jar.jar -o com.ishowedu.child.peiyin8392664_java
      ```

  - bitbucket官网
    - mstrobel / Procyon — Bitbucket

用Procyon从jar导出java源码

下面介绍用Procyon作为命令行工具去导出一个jar包文件为java源代码的过程：

下载Procyon的jar包

比如 procyon-decompiler-0.5.34.jar

用procyon的jar包从你的jar包转换出java代码

语法：

```
java -jar /path/to/procyon-decompiler-0.5.34.jar -jar your_to_decompile.jar -o outputFolderName
```

举例：

```
java -jar  Procyon/procyon-decompiler-0.5.34.jar -jar com.huili.readingclub8825612-dex2jar.jar -o com.huili.readingclub8825612

java -jar  Procyon/procyon-decompiler-0.5.36.jar -jar ../../dex_to_jar/com.ishowedu.child.peiyin8392664-dex2jar.jar -o com.ishowedu.child.peiyin8392664_java
```

基于 `Procyon` 的 `Luyten` 中的菜单中参数是来自 `procyon-decompiler` 命令行的参数

从 `procyon-decompiler` 的help帮助信息是：

```
procyon-decompiler -?
Usage:  main class  [options]  type names or class/jar files
  Options:
    -b, --bytecode-ast
       Output Bytecode AST instead of Java.
       Default: false
    -ci, --collapse-imports
       Collapse multiple imports from the same package into a single wildcard
       import.
       Default: false
```

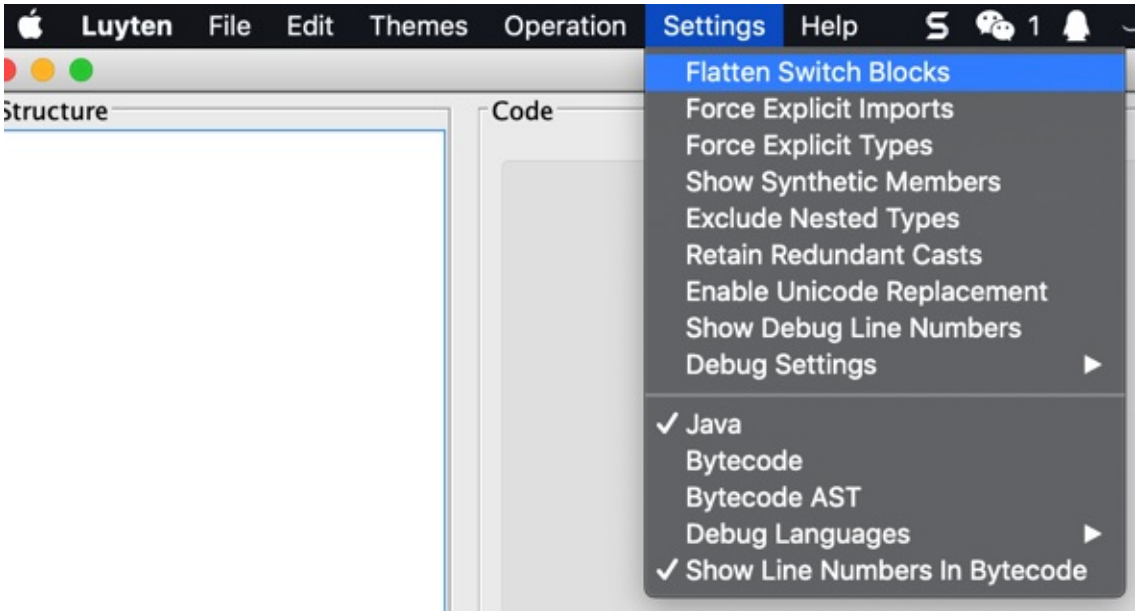
```
    -cp, --constant-pool
       Includes the constant pool when displaying raw bytecode (unnecessary with
       -v).
       Default: false
    -dl, --debug-line-numbers
       For debugging, show Java line numbers as inline comments (implies -ln;
       requires -o).
       Default: false
        --disable-foreach
       Disable 'for each' loop transforms.
       Default: false
    -eml, --eager-method-loading
       Enable eager loading of method bodies (may speed up decompilation of
       larger archives).
       Default: false
    -ent, --exclude-nested
       Exclude nested types when decompiling their enclosing types.
       Default: false
    -ei, --explicit-imports
       [DEPRECATED] Explicit imports are now enabled by default.  This option
       will be removed in a future release.
       Default: false
    -eta, --explicit-type-arguments
       Always print type arguments to generic methods.
       Default: false
    -fsb, --flatten-switch-blocks
       Drop the braces statements around switch sections when possible.
       Default: false
    -fq, --force-qualified-references
       Force fully qualified type and member references in Java output.
       Default: false
    -?, --help
       Display this usage information and exit.
       Default: false
    -jar, --jar-file
       [DEPRECATED] Decompile all classes in the specified jar file (disables
       -ent and -s).
    -lc, --light
       Use a color scheme designed for consoles with light background colors.
       Default: false
    -lv, --local-variables
       Includes the local variable tables when displaying raw bytecode
       (unnecessary with -v).
       Default: false
    -ll, --log-level
       Set the level of log verbosity (0-3).  Level 0 disables logging.
       Default: 0
    -mv, --merge-variables
       Attempt to merge as many variables as possible.  This may lead to fewer
       declarations, but at the expense of inlining and useful naming.  This feature is
       experimental and may be removed or become the standard behavior in future releases.
       Default: false
    -o, --output-directory
       Write decompiled results to specified directory instead of the console.
    -r, --raw-bytecode
       Output Raw Bytecode instead of Java (to control the level of detail, see:
       -cp, -lv, -ta, -v).
       Default: false
    -ec, --retain-explicit-casts
       Do not remove redundant explicit casts.
       Default: false
    -ps, --retain-pointless-switches
       Do not lift the contents of switches having only a default label.
       Default: false
    -ss, --show-synthetic
       Show synthetic (compiler-generated) members.
       Default: false
    -sm, --simplify-member-references
       Simplify type-qualified member references in Java output [EXPERIMENTAL].
       Default: false
    -sl, --stretch-lines
       Stretch Java lines to match original line numbers (only in combination
       with -o) [EXPERIMENTAL].
       Default: false
    -ta, --type-attributes
       Includes type attributes when displaying raw bytecode (unnecessary with
       -v).
       Default: false
```

```
      --unicode
      Enable Unicode output (printable non-ASCII characters will not be
      escaped).
      Default: false
   -u, --unoptimized
      Show unoptimized code (only in combination with -b).
      Default: false
   -v, --verbose
      Includes more detailed output depending on the output language (currently
      only supported for raw bytecode).
      Default: false
      --version
      Display the decompiler version and exit.
      Default: false
   -ln, --with-line-numbers
      Include line numbers in raw bytecode mode; supports Java mode with -o
      only.
      Default: false
```

从这些支持的参数中，可以推断：

基于 `Procyon` 的 `Luyten` 中的菜单中的选项，都是对应着这些参数的。

比如： `Settings -> Flatten Switch Blocks`



对应着此处参数 `--flatten-switch-blocks` ：

```
   -fsb, --flatten-switch-blocks
      Drop the braces statements around switch sections when possible.
      Default: false
```

其他以此类推，都是一样的逻辑：把命令行工具的参数，用菜单选项的形式展示和支持出来了。

Luyten

- 主页
  - deathmarine/Luyten: An Open Source Java Decompiler Gui for Procyon
- 功能
  - Java Decompiler Gui for Procyon
  - 基于Procyon的带图形界面的Java反编译器
- 界面
  -
- 下载
  - Release Luyten v0.5.4 · deathmarine/Luyten

Luyten的语法高亮的不同主题的效果

- 默认：Default = Default-Alt

- Dark

- Eclipse

-
- Visual Studio

-
- IntelliJ

-

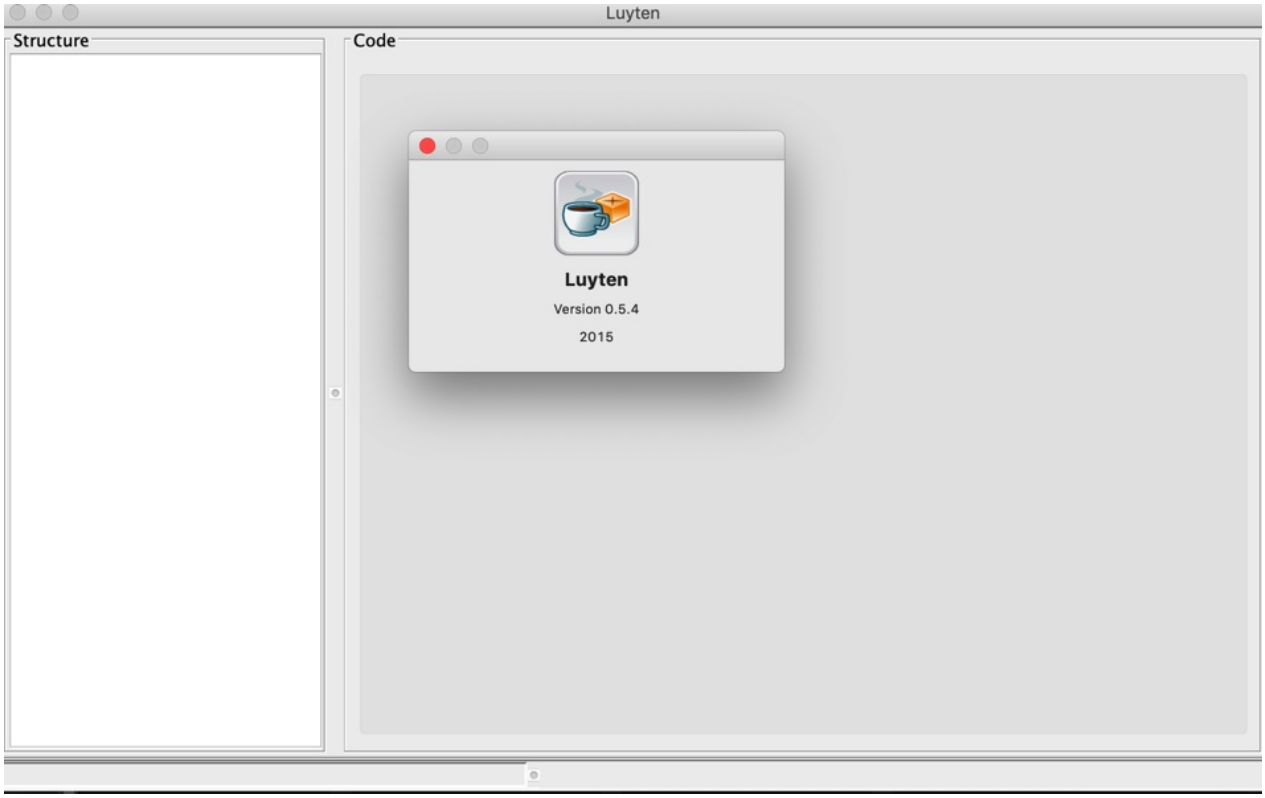
下载和使用Luyten去查看和导出java代码

去Releases · deathmarine/Luyten下载最新版本，比如Mac版的：

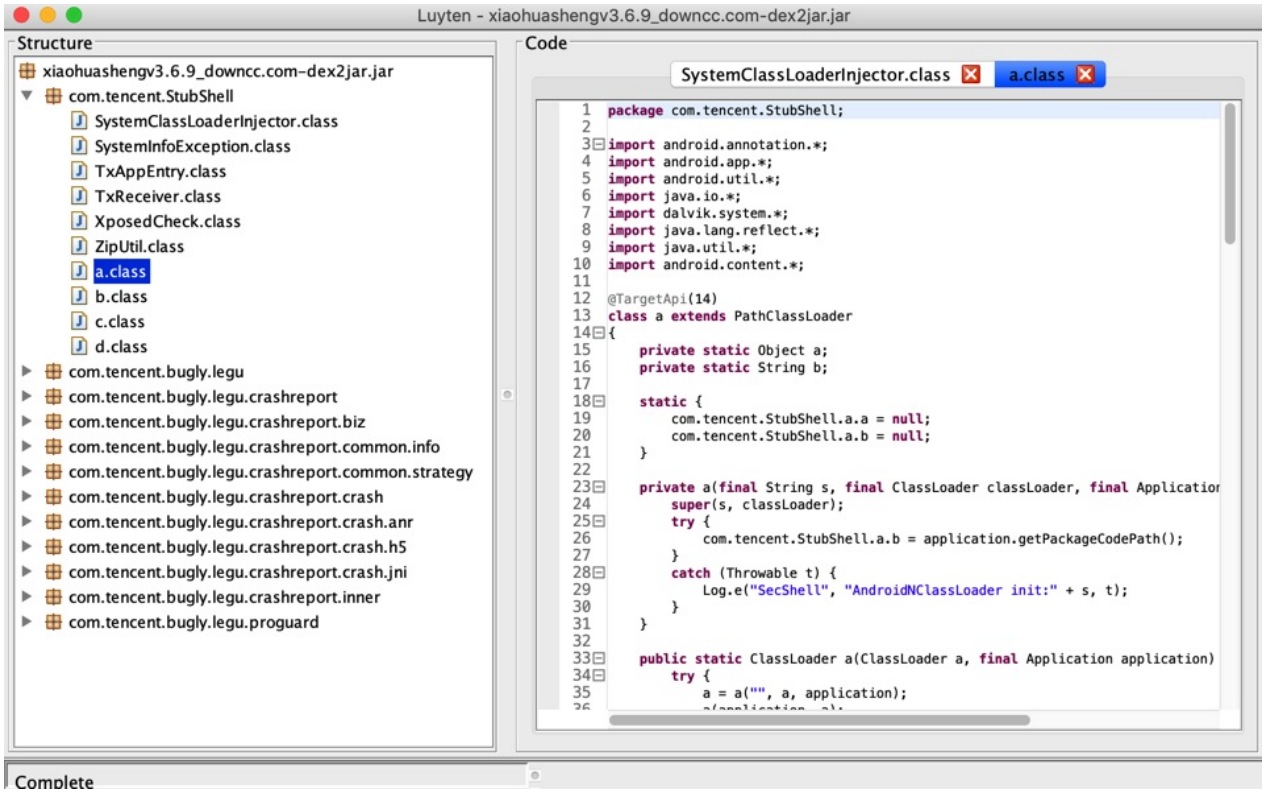
luyten-OSX-0.5.4.zip

解压得到 `Luyten.app` ：

运行后：

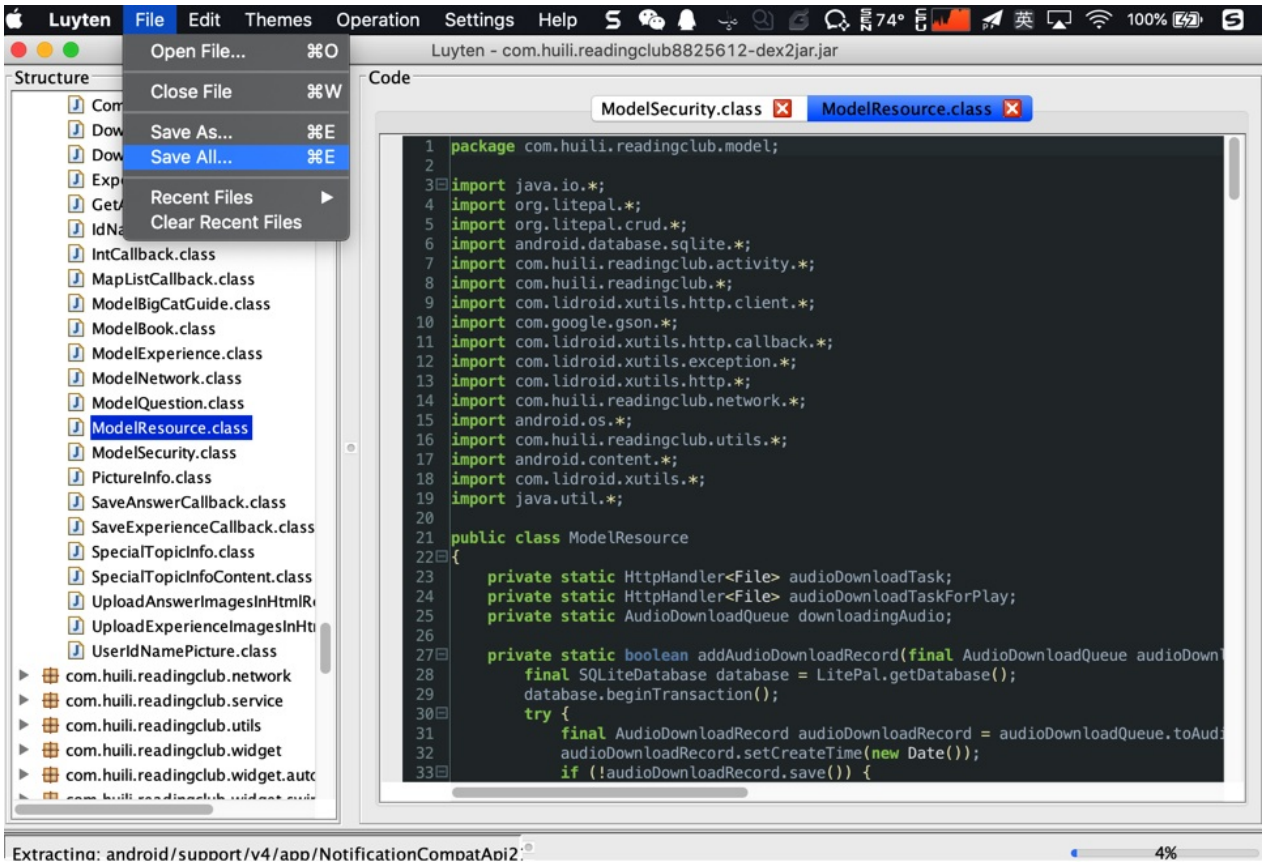


然后去把 `jar` 包拖进来即可看到代码：



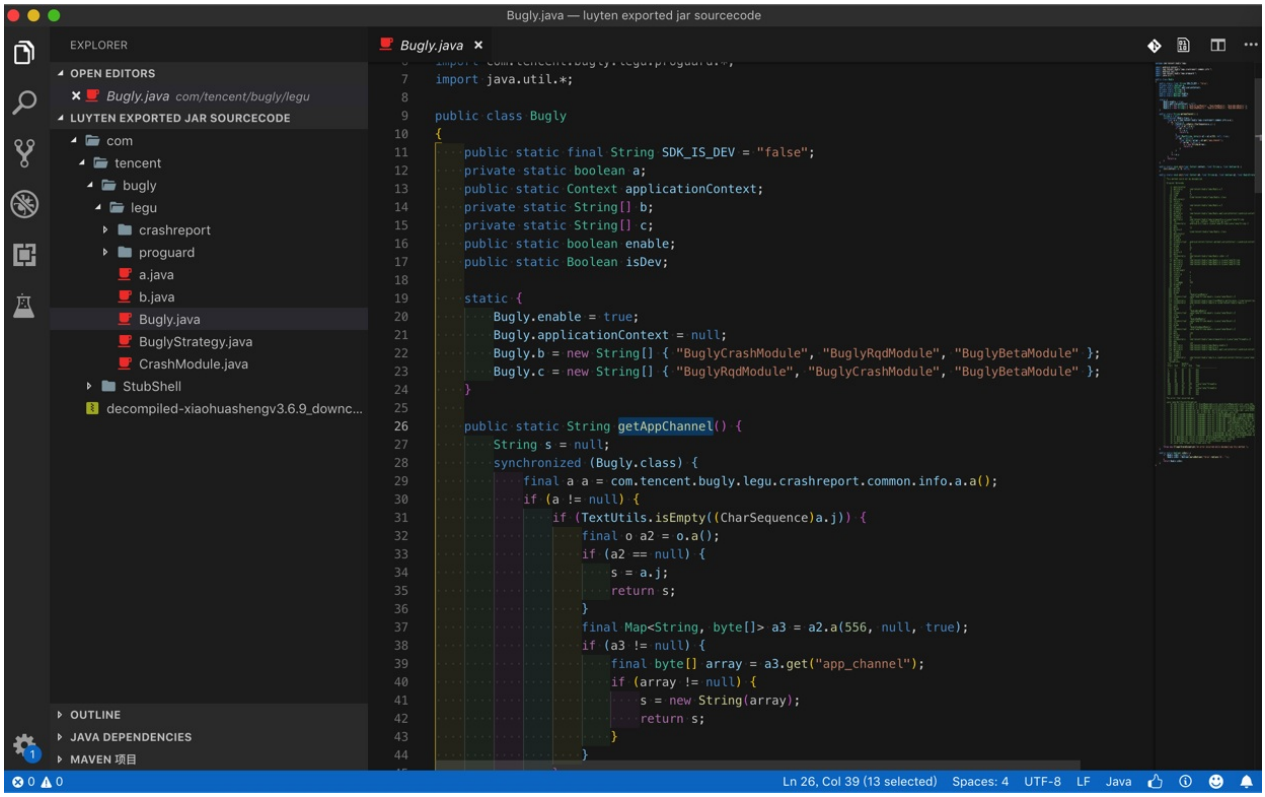
导出所有代码的步骤：

```
File -> Save All -> decompiled-xxx.zip
```

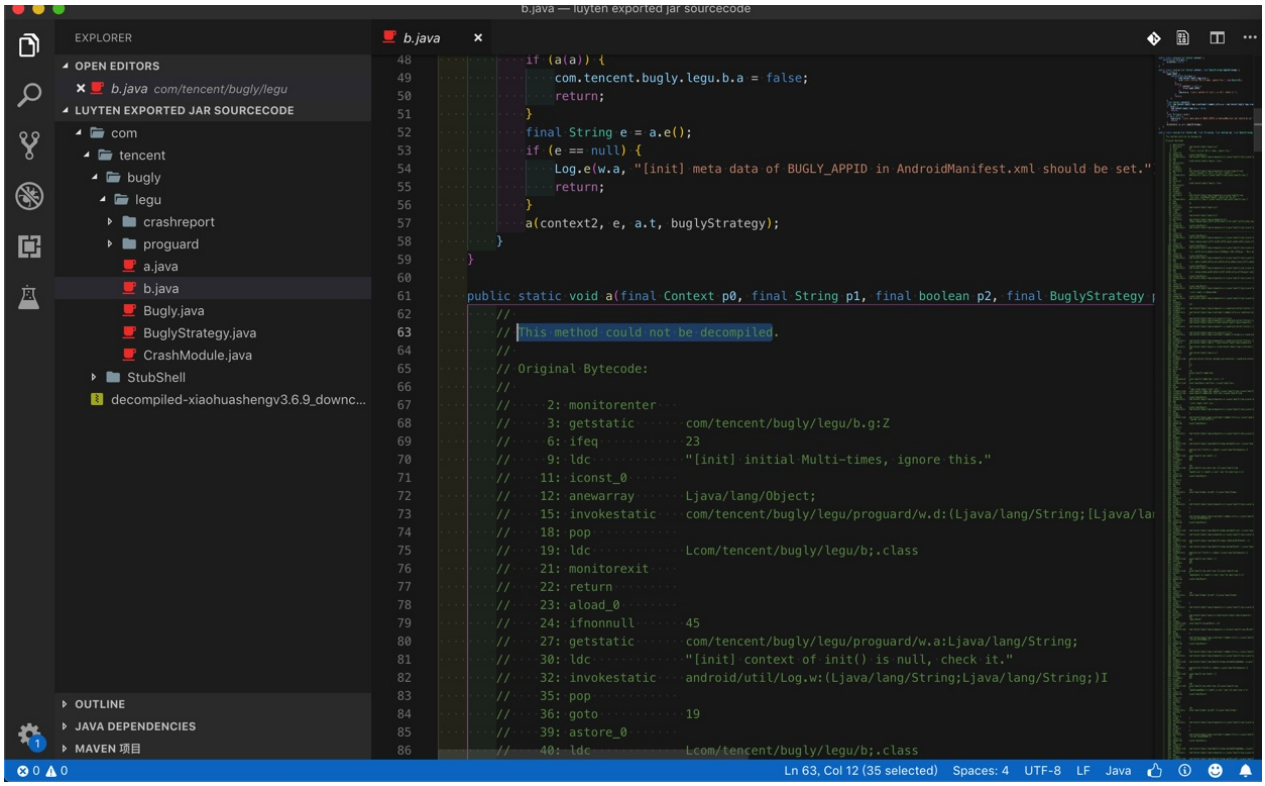


解压后，即可用VSCode等工具方便查看代码了。

之前 `JD-GUI` 、 `CFR` 等工具转换出错的代码，此处是可以正确解析的：



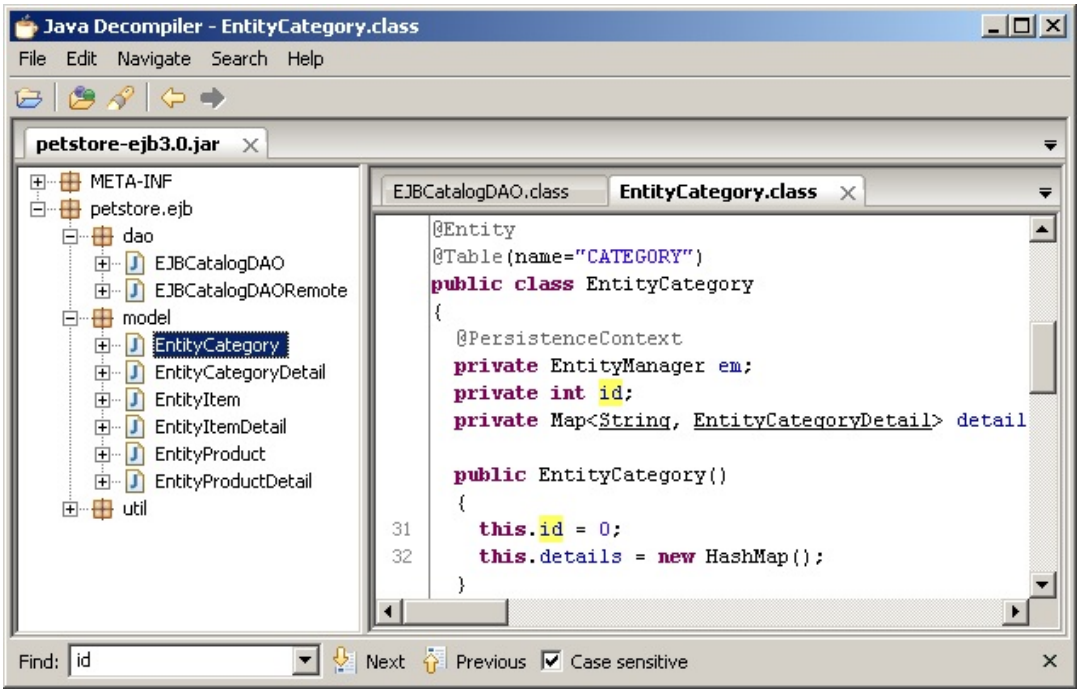
但是也还是有部分代码无法正确解析：



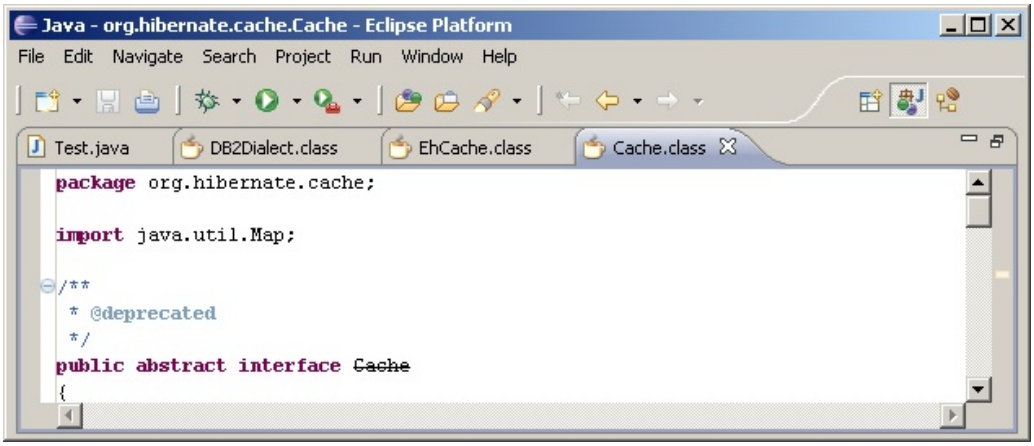
CFR

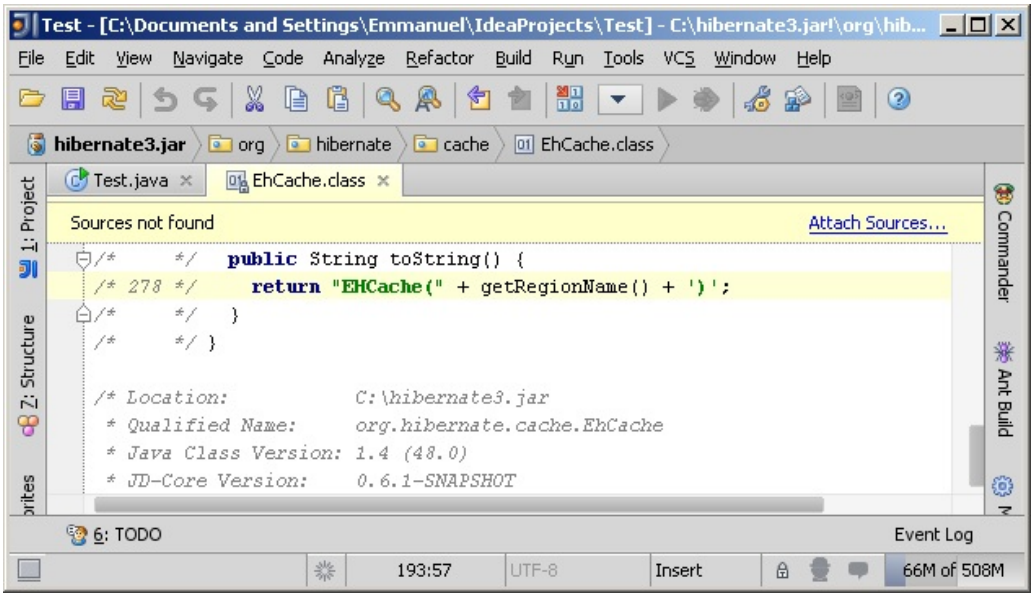
- `CFR` = `Class File Reader`

JD-GUI

- 主页
  - Java Decompiler
- JD Project=Java Decompiler Project
  - `JD-Core`
    - 是什么：`一个库`
    - 功能：从一个或多个.class文件中重构java源代码
    - 用途：
      - 可用于恢复丢失的源代码
      - 可用于查看jar包的java源码=查看JRE（Java运行时）库的源码
    - 特点：
      - java 5中最新的功能
        - 注释annotations
        - generics 或 枚举类
    - 说明：
      - JD-GUI内置包含了JD-Core
      - JD-Eclipse内置包含了JD-Core
  - `JD-GUI`
    - 是什么：一个独立的带图形界面的`程序`
    - 作用：显示查看jar包的java源代码
      - 注：`jar包`=内部包含了很多`.class`文件的，被压缩打包成`jar`
    - 举例
    - 

    - github主页
      - java-decompiler/jd-gui: A standalone Java Decompiler GUI
    - 下载
      - http://java-decompiler.github.io
      - -》
      - Releases · java-decompiler/jd-gui
      - -》
      - 比如：
        - `Mac`：

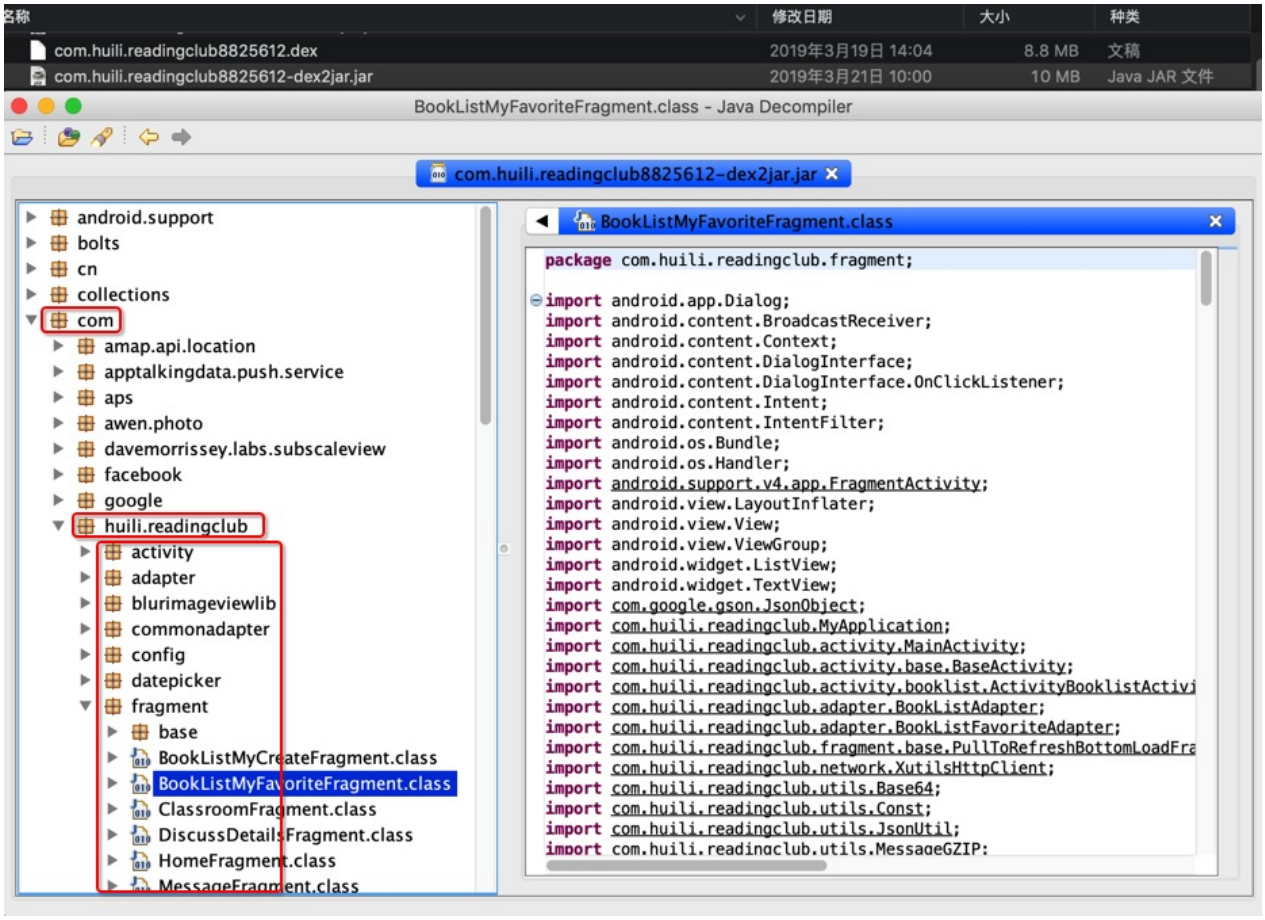
        - jd-gui-osx-1.4.1.tar
    - 插件
        - JD-Eclipse：Eclipse的插件
            - 举例：
            - 

        - JD-IntelliJ：IntelliJ IDEA的插件
            - 举例：
            - 

JD-GUI反编译jar的效果

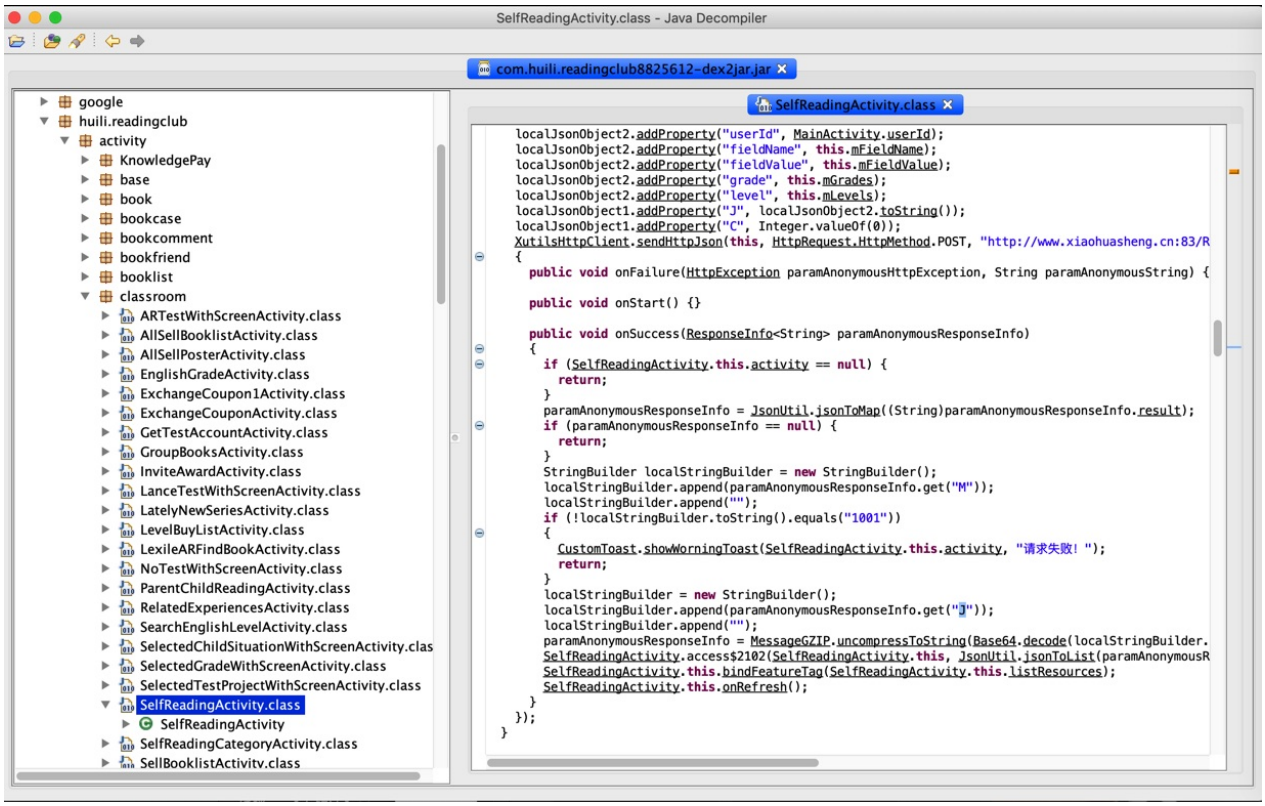
举例

用 `JD-GUI` 打开jar文件，即反编译jar文件， `com.ishowedu.child.peiyin9201516-dex2jar.jar` ，得到java代码的效果：



其中找到了我们之前需要的app相关的业务逻辑的代码：

`/com/huili/readingclub/activity/classroom/SelfReadingActivity.class`

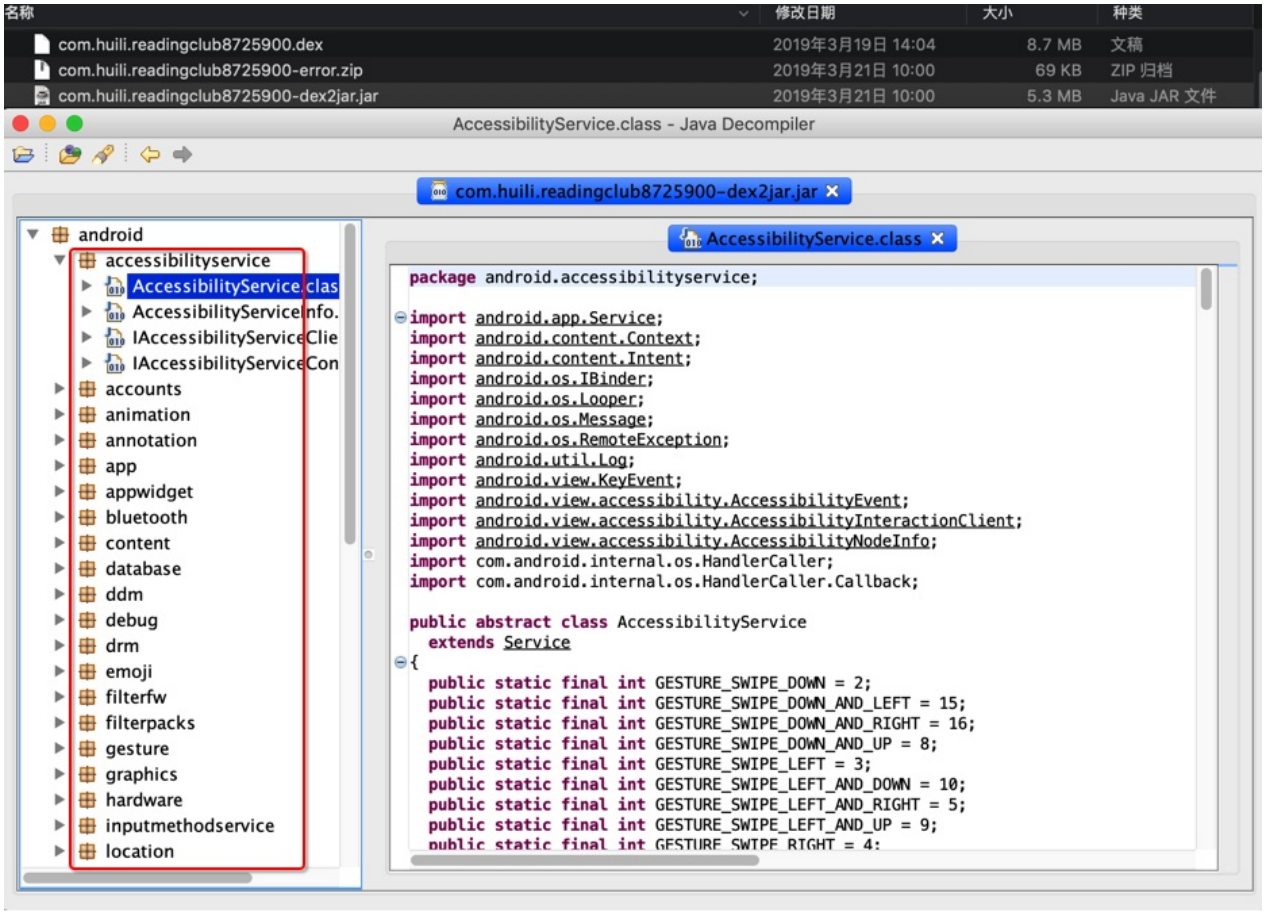


其中 `onSuccess` 中就是我们希望得到的对于 `J` 字段解密的逻辑。

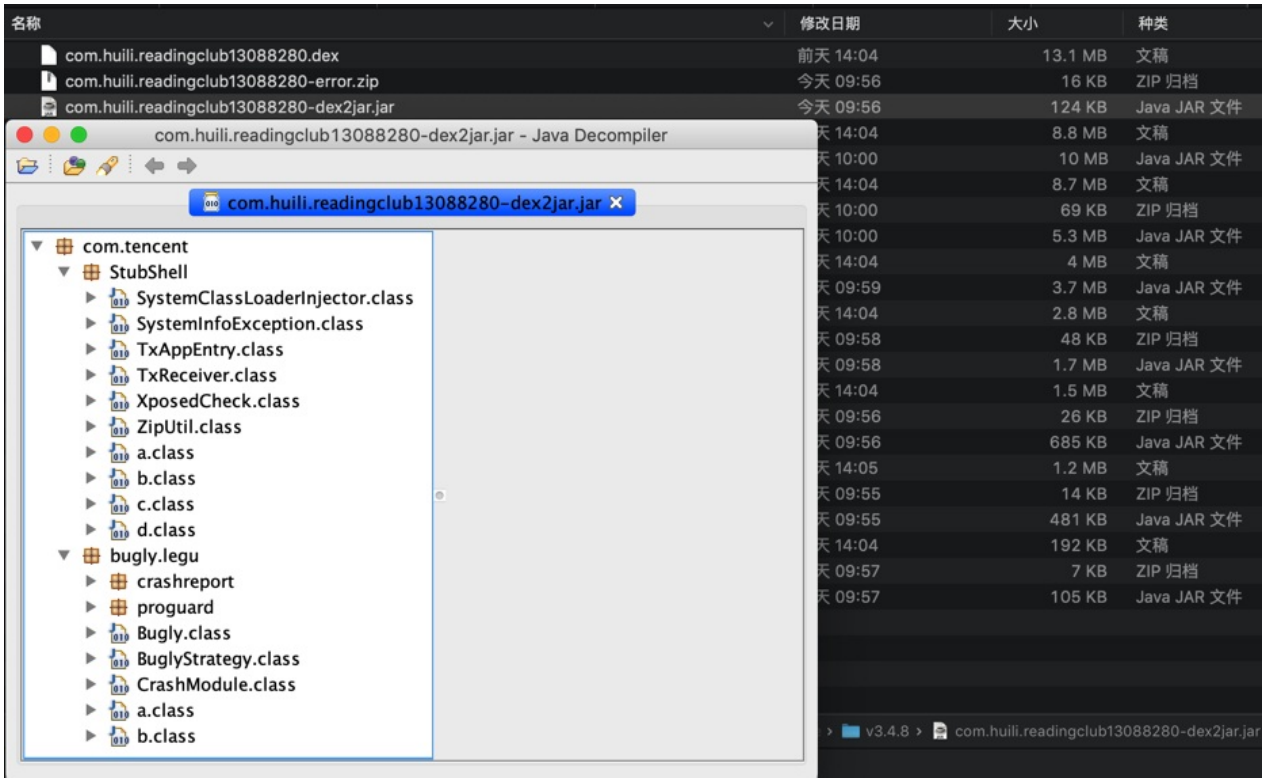
其他无效的jar转换出jar的效果

如前一步所述，从多个 `dex` 可以转换出多个 `jar`

而这些无效的、没有包含app业务逻辑的 `jar` ，去用一些反编译工具打开后的效果是：



其他的一些，比如腾讯乐固加密了的，最终转换出来的jar，去打开后只能看到腾讯乐固的代码：



Krakatau

- 主页
  - Storyyeller/Krakatau: Java decompiler, assembler, and disassembler
- 功能
  - java的反编译器、汇编器、反汇编器
    - 如果要java反编译：则需要安装Java JDK
    - 如果不需要java反编译，只是汇编和反汇编，则可以不用安装Java JDK
      - 但是为了更好的测试结果，最好安装Java JDK
- 特点
  - 全部是 `Python` 写的
  - 支持 `Java 10`

Fernflower

- 主页
  - 官网
    - intellij-community/plugins/java-decompiler/engine at master · JetBrains/intellij-community
  - 非官方 GitHub
    - fesh0r/fernflower: Unofficial mirror of FernFlower Java decompiler (All pulls should be submitted upstream)
- 功能
  - Java的反编译器
    - 从Java的 `class` 反编译出java `源代码`
- 说明
  - IntelliJ IDEA 内置
- 用法

```
java -jar fernflower.jar [- option=value ]* [ source ]+ destination
```

  - 举例

```
java -jar fernflower.jar -hes=0 -hdc=0 c:\Temp\binary\ -e=c:\Java\rt.jar c:\Temp\source\
```

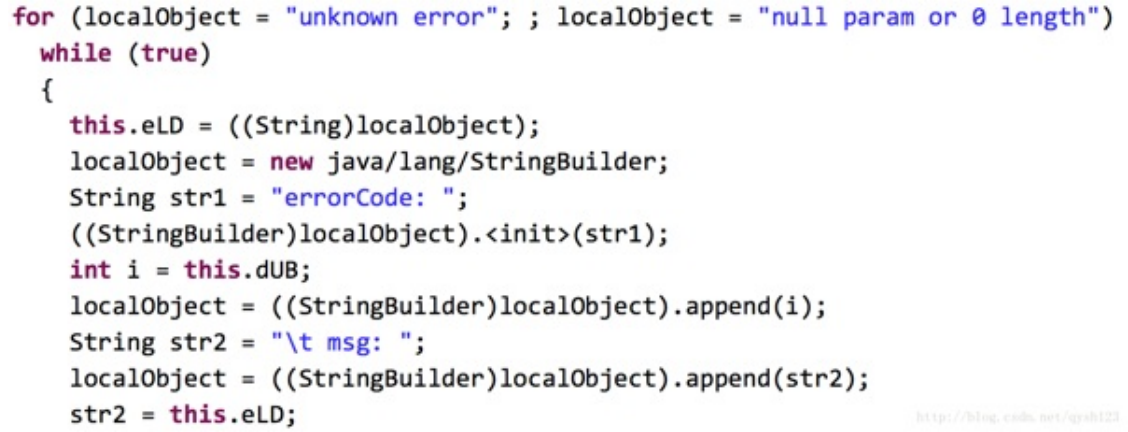
```
java -jar fernflower.jar -dgs=1 c:\Temp\binary\library.jar c:\Temp\binary\Boot.class c:\Temp\source\
```

Dare

- 主页
  - Dare Homepage
- 下载
  - Dare downloads
- 功能
  - 将 `apk` 文件反编译为Java的 `class` 文件
- 用法

```
dare -d apkoutput WeChat_462.apk
```

- 截图
  - 输出



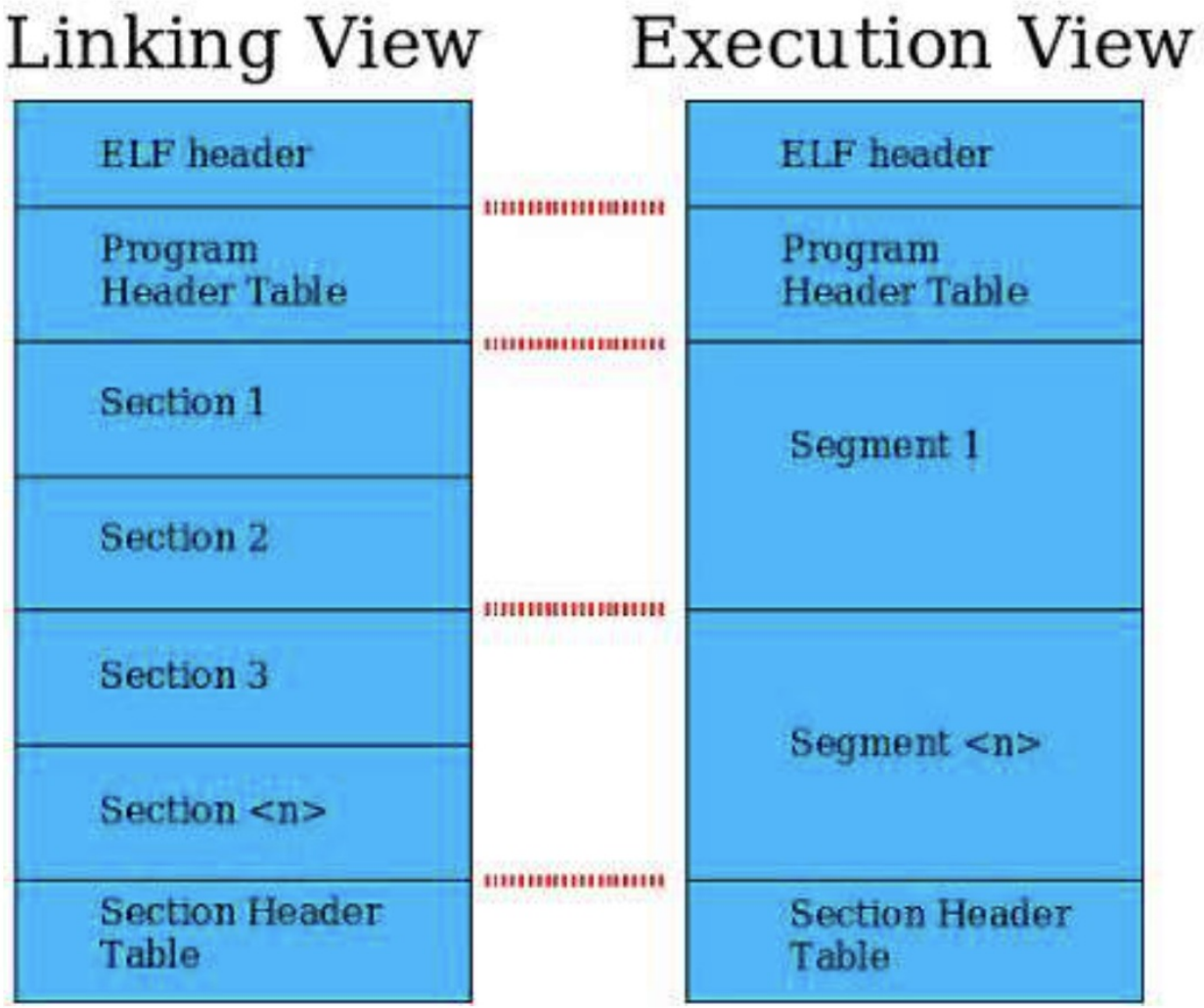
```
for (localObject = "unknown error"; ; localObject = "null param or 0 length")
  while (true)
  {
    this.eLD = ((String)localObject);
    localObject = new java/lang/StringBuilder;
    String str1 = "errorCode: ";
    ((StringBuilder)localObject).<init>(str1);
    int i = this.dUB;
    localObject = ((StringBuilder)localObject).append(i);
    String str2 = "\t msg: ";
    localObject = ((StringBuilder)localObject).append(str2);
    str2 = this.eLD;
```

JAD

- `JAD` = `the fast JAva Decompiler`
- 主页
  - Home Page of Jad - the fast Java decompiler
    - 作者个人主页？：KPD Home Page
- 下载
  - Download Jad
  - JAD Java Decompiler Download Mirror
- 说明
  - 最后更新时间：2006年
    - 比较老旧的反编译器，已不再更新和维护
      - 不建议继续使用
        - 换用最新的，比如 `jadx`

针对so

ELF文件格式



导出静态资源字符串

导出ELF格式的so文件的字符串信息

```
rabin2 -I elfFile.so > elfFile_rabin2_I_identification.coffee
rabin2 -i elfFile.so > elfFile_rabin2_i_imports.coffee
rabin2 -E elfFile.so > elfFile_rabin2_E_exports.coffee
rabin2 -l elfFile.so > elfFile_rabin2_l_libraries.coffee
rabin2 -z elfFile.so > elfFile_rabin2_z_strings.coffee
rabin2 -s elfFile.so > elfFile_rabin2_s_symbols.coffee
rabin2 -S elfFile.so > elfFile_rabin2_S_sections.coffee


strings elfFile.so > elfFile_strings.coffee


readelf -h elfFile.so > elfFile_readelf_h_header.coffee
readelf -a elfFile.so > elfFile_readelf_a_all.coffee
readelf -e elfFile.so > elfFile_readelf_e_hlS.coffee
readelf -s elfFile.so > elfFile_readelf_s_symbols.coffee
readelf -r elfFile.so > elfFile_readelf_r_relocs.coffee
readelf -l elfFile.so > elfFile_readelf_l_programHeaders_segments.coffee
readelf -S elfFile.so > elfFile_readelf_S_sections.coffee
readelf -x .dynsym elfFile.so > elfFile_readelf_x_hexDump_dynsym.coffee
readelf -p .dynsym elfFile.so > elfFile_readelf_p_stringDump_dynsym.coffee
readelf -p .dynstr elfFile.so > elfFile_readelf_p_stringDump_dynstr.coffee
readelf -n elfFile.so > elfFile_readelf_n_notes.coffee
readelf -d elfFile.so > elfFile_readelf_d_dynamic.coffee


objdump -d -j .text elfFile.so > elfFile_objdump_d_j_disassembleSection_text.coffee
objdump -d -r elfFile.so > elfFile_objdump_d_r_disassemReloc.coffee
objdump -a elfFile.so > elfFile_objdump_a_archiveHeaders.coffee
objdump -f elfFile.so > elfFile_objdump_f_fileHeaders.coffee
objdump -h elfFile.so > elfFile_objdump_h_sectionHeaders.coffee
objdump -x elfFile.so > elfFile_objdump_x_allHeaders.coffee
objdump -s elfFile.so > elfFile_objdump_s_fullContents.coffee
objdump -t elfFile.so > elfFile_objdump_t_syms.coffee
objdump -T elfFile.so > elfFile_objdump_T_dynamicSyms.coffee
objdump -r elfFile.so > elfFile_objdump_r_reloc.coffee
objdump -R elfFile.so > elfFile_objdump_R_dynamicReloc.coffee
```

举例

```
rabin2 -I libRehADGd_arm64v8a.so > libRehADGd_rabin2_I_identification.coffee
rabin2 -i libRehADGd_arm64v8a.so > libRehADGd_rabin2_i_imports.coffee
rabin2 -E libRehADGd_arm64v8a.so > libRehADGd_rabin2_E_exports.coffee
rabin2 -l libRehADGd_arm64v8a.so > libRehADGd_rabin2_l_libraries.coffee
rabin2 -z libRehADGd_arm64v8a.so > libRehADGd_rabin2_z_strings.coffee
rabin2 -s libRehADGd_arm64v8a.so > libRehADGd_rabin2_s_symbols.coffee
rabin2 -S libRehADGd_arm64v8a.so > libRehADGd_rabin2_S_sections.coffee

strings libRehADGd_arm64v8a.so > libRehADGd_strings.coffee

readelf -h libRehADGd_arm64v8a.so > libRehADGd_readelf_h_header.coffee
readelf -a libRehADGd_arm64v8a.so > libRehADGd_readelf_a_all.coffee
readelf -e libRehADGd_arm64v8a.so > libRehADGd_readelf_e_hlS.coffee
readelf -s libRehADGd_arm64v8a.so > libRehADGd_readelf_s_symbols.coffee
readelf -r libRehADGd_arm64v8a.so > libRehADGd_readelf_r_relocs.coffee
readelf -l libRehADGd_arm64v8a.so > libRehADGd_readelf_l_programHeaders_segments.coffee
readelf -S libRehADGd_arm64v8a.so > libRehADGd_readelf_S_sections.coffee
readelf -x .dynsym libRehADGd_arm64v8a.so > libRehADGd_readelf_x_hexDump_dynsym.coffee
readelf -p .dynsym libRehADGd_arm64v8a.so > libRehADGd_readelf_p_stringDump_dynsym.coffee
readelf -p .dynstr libRehADGd_arm64v8a.so > libRehADGd_readelf_p_stringDump_dynstr.coffee
readelf -n libRehADGd_arm64v8a.so > libRehADGd_readelf_n_notes.coffee
readelf -d libRehADGd_arm64v8a.so > libRehADGd_readelf_d_dynamic.coffee

objdump -d -j .text libRehADGd_arm64v8a.so > libRehADGd_objdump_d_j_disassembleSection_text.coffee
```

```
objdump -d -r libRehADGd_arm64v8a.so > libRehADGd_objdump_d_r_disassemReloc.coffee
objdump -a libRehADGd_arm64v8a.so > libRehADGd_objdump_a_archiveHeaders.coffee
objdump -f libRehADGd_arm64v8a.so > libRehADGd_objdump_f_fileHeaders.coffee
objdump -h libRehADGd_arm64v8a.so > libRehADGd_objdump_h_sectionHeaders.coffee
objdump -x libRehADGd_arm64v8a.so > libRehADGd_objdump_x_allHeaders.coffee
objdump -s libRehADGd_arm64v8a.so > libRehADGd_objdump_s_fullContents.coffee
objdump -t libRehADGd_arm64v8a.so > libRehADGd_objdump_t_syms.coffee
objdump -T libRehADGd_arm64v8a.so > libRehADGd_objdump_T_dynamicSyms.coffee
objdump -r libRehADGd_arm64v8a.so > libRehADGd_objdump_r_reloc.coffee
objdump -R libRehADGd_arm64v8a.so > libRehADGd_objdump_R_dynamicReloc.coffee
```

readelf

objdump

razbin2

反编译代码逻辑

IDA

详见：逆向利器：IDA

Hopper

radare2

- 简称：`r2`
- 主页
  - Github
    - radare/radare2: unix-like reverse engineering framework and commandline tools
  - 官网
    - radare
- 功能和特点
  - 一个开源的逆向工程和二进制分析框架
  - 功能非常强大
    - 反汇编、分析数据、打补丁、比较数据、搜索、替换、虚拟化等等
    - 强大的静态或动态分析、十六进制编辑以及溢出漏洞挖掘
    - 具备超强的脚本加载能力
  - 可以运行在几乎所有主流的平台
    - GNU/Linux, Windows, *BSD, iOS, OSX, Solaris
- 组成
  - 由一系列的组件构成
    - `rahash2`：各种密码算法，`hash算法`
    - `rabin2`：查看文件格式
    - `ragg2` / `ragg2cc`：用于更方便的生成`shellcode`
    - `rax2`：用于数值转换
    - `rasm2`：反汇编和汇编
    - `radiff2`：对文件进行`diff`
    - => `radare2`：整合了上面的工具
  - 额外还有很多**插件=**`Plugins`
    - `esilsolve`: The symbolic execution plugin, based on esil and z3
    - `r2diaphora`: Diaphora's diffing engine working on top of radare2
    - `iaito`: The official Qt graphical interface
    - `radius2`: A fast symbolic execution engine based on boolector and esil
    - `r2dec`: A decompiler based on r2 written in JS, accessed with the pdd command
    - `r2ghidra`: The native ghidra decompiler plugin, accessed with the pdg command
    - `r2frida`: The frida io plugin. Start r2 with r2 frida://0 to use it
    - `r2poke`: Integration with GNU/Poke for extended binary parsing capabilities
    - `r2pipe`: Script radare2 from any programming language
    - `r2papi`: High level api on top of r2pipe
- 典型用途
  - 参加CTF
  - 逆向工程
  - 漏洞挖掘
  - 分析恶意软件（如溯源）

Ghidra

相关子项

apk文件

AndroidManifest.xml

安卓的 `apk` 中有个 `AndroidManifest.xml` 文件，是保存了相关项目的类，资源等配置信息。

而对于安卓的 `apk` 文件：

- （由于本身就是个zip压缩包文件，所以）直接改名为 `zip` 后再解压即可得到的（加了密的）**二进制**格式的 `AndroidManifest.xml` 文件
- 用 `apktool` 等工具去反编译：得到的是**文本**格式的 `AndroidManifest.xml` 文件
  - 就可以看到xml的原始内容了
    - 注：即使 `apk` 加固了，也可以用 `apktool` 反编译

AndroidManifest.xml的作用

得到了xml源码后，可以从其中看到很多有用的信息。

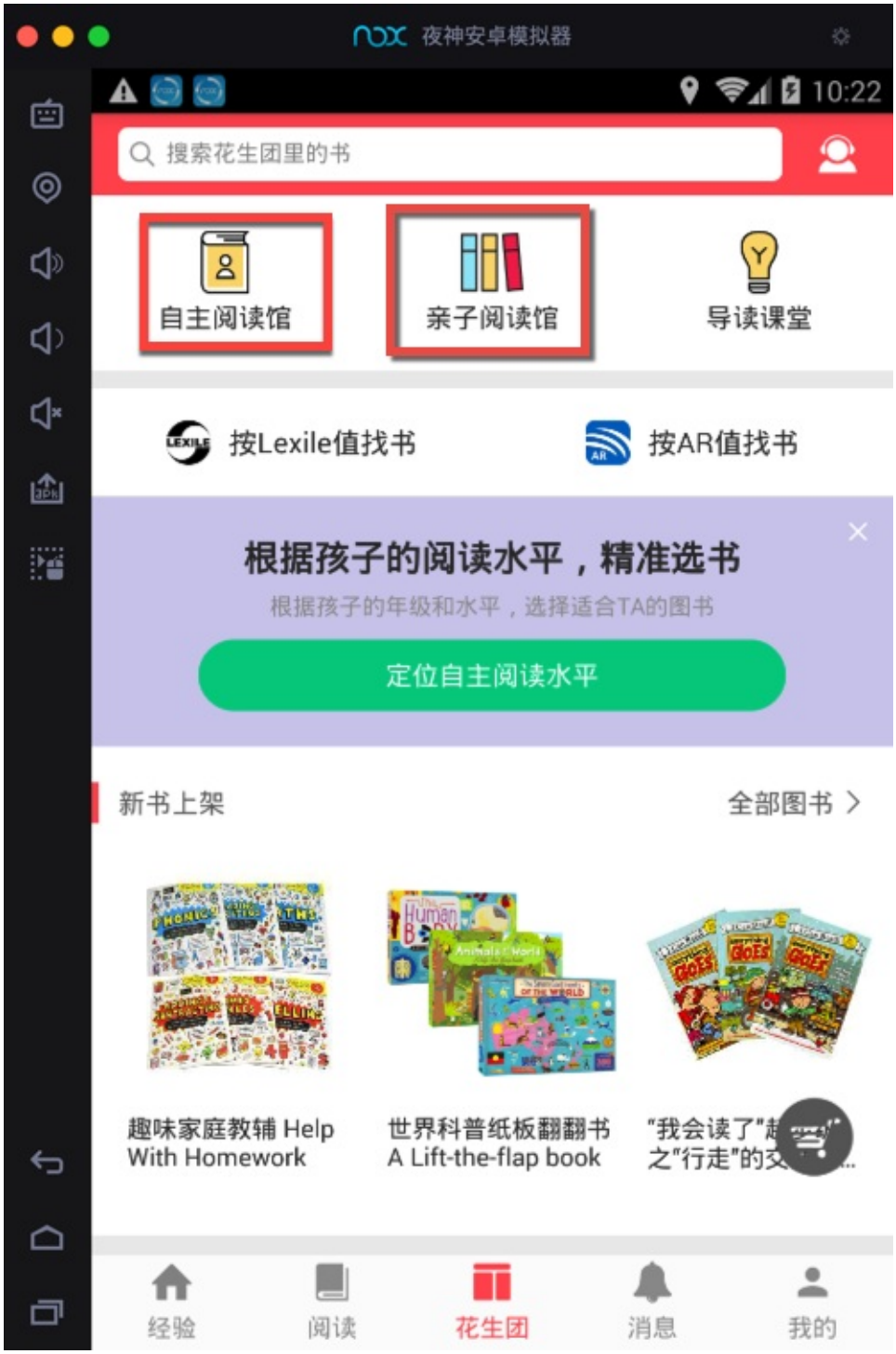
比如， 小花生安卓版 v3.6.9的 `apk` ，apktool反编译出的 `AndroidManifest.xml` 中包含：

```
<activity          name="com.huili.readingclub.activity.classroom.SelfReadingActivity"          screenOrientation="port
rait"/>
...
<activity          name="com.huili.readingclub.activity.classroom.ParentChildReadingActivity"          screenOrientation
="portrait"/>
```

其中的类名：

- `activity.classroom.SelfReadingActivity`
- `activity.classroom.ParentChildReadingActivity`

=类的文件名，对应了app界面：



中的：

- 自主阅读馆
- 亲子阅读馆

-》从而有利于后续分析内部的业务逻辑，了解到内部有哪些类和功能。

AXMLPrinter2

- 主页
  - GoogleCode
    - android4me - Google Code
  - Github
    - limpoxe/android4me: Automatically exported from code.google.com/p/android4me
    - digitalsleuth/AXMLPrinter2: AXMLPrinter jar and zip files from Google Code
- 下载
  - AXMLPrinter2.jar
- 作用
  - 将安卓的二进制的 `AXML` 转换为可读的 `xml` 文件
- 提示
  - 最近更新都是2008年，11年前的了
    - 看来是很老旧的工具了
- 用法

```
java -jar AXMLPrinter2.jar xxx.xml output.xml
```

```
java -jar ~/dev/ApkTOOL/AXMLPrinter2.jar AndroidManifest.xml > main.xml
```

反编译

- 反编译
  - 名词来源
    - 正常编译(`compile` = `compiling`)出apk的过程
      - 对于反方向的，逆向的，破解这个过程，常叫做： `反编译` = `decompile` = `decompiling`
        - 所以相关工具往往也叫做： `反编译器` = `decompiler`
  - 反编译内容
    - 主要是涉及到的：反编译dex文件
      - 其中也常涉及到到其他格式
        - 往往不严谨的说法，也叫做反编译

反编译流程

如何从apk破解出java源码

下面整理，从安卓的 `apk` 文件，如何破解，反编译，逆向工程，得到 `java` 源代码。

对于从apk破解得到java源代码，目前常见的几种思路：

- 未加固的
  - 直接一步搞定
    - 用 `jadx` 从 `apk` 导出 `java` 源码
- 已加固的
  - 前提
    - apk虽然被（旧版360加固保等）加固，但后续可以（用 `FDex2` 等砸壳工具）导出 `dex` 文件
      - 注
        - 有些新版本的加固，比如腾讯乐固legu，新版360加固保等，无法导出dex文件
          - 所以就无法继续转换出java源码
  - 两步或三步：`app` -> `dex` -> `jar` -> `java`
    - 第一步：先砸壳导出dex
      - 从 `apk` 到 `dex` ：从运行中的安卓app导出dex文件
        - 先安装安卓apk到 `已root` 的安卓 `真机` /安卓 `模拟器`
        - 在安卓中已经安装了 `Xposed` 等 `Hook框架`
        - 再去 `Xposed` 中安装 `FDex2` / `DumpDex` 等插件=砸壳工具
          - 用于后续导出 `dex` 文件
        - 运行安卓app
          - 自动导出我们要的 `dex` 文件
        - 从安卓（真机/模拟器）中拷贝出刚才导出的（往往是多个） `dex` 文件
    - 再分2种：从 `dex` 到 `java` 源码
      - 一步：
        - 第二步：直接从dex转java
          - 比如用 `jadx` 支持直接从 `dex` 转换出 `java` 源代码
      - 两步：
        - 第二步：从dex转jar
          - `dex` 转 `jar`
            - 用 `dex2jar` 等工具从（包含了我们要的、和app业务逻辑相关的那个） `dex` 文件中导出 `jar` 文件
        - 第三步：再从jar转java
          - `jar` 转 `java` 源码
            - 用反编译器（ `CFR` / `Procyon` / `JD-GUI` 等）把 `jar` 转换出 `java` 源代码

后续详细介绍每一步涉及到的内容，和如何用工具实现对应反编译。

反编译工具

- 反编译工具
  - `apk` / `dex` 转 `java`
    - `jadx`
    - `JEB`
    - `GDA`
  - `dex` 转 `jar`
  - `jar` 转 `java` = java反编译

jadx

- 概述
  - 从 `apk` / `dex` 直接一步转 `java` 的好用的反编译之一
  - 用法

    ```
    jadx -d output_folder apk_file.apk
    jadx -d output_folder dex_file.dex
    ```

  - 主页
    - skylot/jadx: Dex to Java decompiler
- 详解
  - 安卓反编译利器：jadx

JEB

GDA

- `GDA` = `GJoy Dex Analysizer` =GDA反编译器
- 主页
  - GDA主页-功能强大的交互式Android反编译分析工具
- 下载
  - GDA下载 最新版
- 功能
  - 一款强大而轻便的交互式反编译器，也是一款综合性逆向分析利器
    - 支持分析 `apk` , `dex` , `odex` , `oat` , `jar` , `class` , `aar` 类型文件
    - 支持python脚本以及方法签名制作与识别
  - 工具包含三个由作者独立完成的高速解析引擎
    - 反编译引擎
    - apk壳检测引擎
    - 恶意行为检测引擎
  - 再加上作者独创的使用了字节码直接转java伪代码的解析方式
    - 无需转换成smali汇编后再做反编译
      - 大大提升了解析速度
- 特点
  - 无需安装java环境和android环境就可以使用
    - 不是依赖其他的（很多安卓反汇编的、java的）库
  - 分析速度快、体积小、内存占用少
- 截图
  -

-
-

- GDA vs JEB

- 。

二进制编辑器

在安卓反编译期间，可能会涉及编辑一些二进制的文件，比如 `dex` 文件等。

所以也会涉及到一些二进制编辑方面的工具，整理如下：

010editor

- 主页
  - 010 Editor - Professional Text/Hex Editor with Binary Templates
- 下载
  - SweetScape Software Inc - Download 010 Editor
- 功能：
  - 二进制(16进制)编辑器
    - 查看和编辑二进制文件
      - 用来辅助破解apk：编辑dex文件
- 截图
  -

EverEdit

- 首页
  - 首页 | EverEdit
- 下载
  - 下载 | EverEdit
- 功能
  - 专为国人设计的文本编辑器
    - 身躯小巧，性能卓越，自定义功能完善，丰富的主题和脚本，完美的编码、大字符集字符显示，无论您是哪个级别的码农，EverEdit都会给您带来不一样的体验！
- 特点
  - 多点编辑
    - 只需要轻按一下Alt+F3，她就会替你选择所有的同名变量；或者按Ctrl+D逐个向下选择。
  - 超强编码
    - 准确的探测文件的编码，一软在手，万码无忧！
  - Emmet/Zencoding
    - 完美支持Emmet,支持Tab一键展开那一大坨代码！
  - 完美Markdown
    - 内置markdown的预览，实时渲染，并排放置视图，一边改一边看，智能回车键和大纲！
  - 代码片段
    - 用Tab或者Shift+Tab在多个编辑点之间来回跳转。同名位置的引用，也会在修改时同时发生相同的变更。
  - 超大文件
    - 用较少的内存异步打开巨大文件！打开文件时，不会阻塞界面而且您可以随时取消该操作！
  - 二进制编辑
    - 内置二进制编辑器，瞬间打开任意大小的文件。并且可以进行查找和替换。同时对找到的字符串和被修改的位置进行高亮显示。
  - 文档地图
    - 文档地图以缩略图的形式显示出当前文档的整体外观；您可以拖放当前区域进行滚动和定位，甚至可以完全用它替换滚动条！
  - 函数列表
    - 列出安当前文件内所有的符号(类、函数、变量、宏等)，每个符号都会用一个恰当图标进行标示！
- 支持平台
  - Windows
- 截图

-
-

综合类工具

AndroidKiller

- 功能
  - 一款可视化的安卓应用逆向工具
    - 集Apk反编译、Apk打包、Apk签名，编码互转，ADB通信（应用安装-卸载-运行-设备文件管理）等特色功能于一身，支持logcat日志输出，语法高亮，基于关键字（支持单行代码或多行代码段）项目内搜索，可自定义外部工具；吸收融汇多种工具功能与特点
    - 打造一站式逆向工具操作体验，大大简化了安卓应用/游戏修改过程中各类繁琐工作
- 下载
  - AndroidKiller v1.3.1
    - liaojack8/AndroidKiller: AndroidKiller v1.3.1
    - AndroidKiller下载|AndroidKiller(安卓APK反汇编工具)下载 v1.3.1绿色中文版_ - pc6下载站
  - 更新版本?
    - SeagullOddy/android-killer: 经典的安卓反编译工具。An android decompile tool.
- 截图
  -

-
-

Android-Crack-Tool For Mac

- 主页
  - Jermic/Android-Crack-Tool: Android crack tool For Mac
    - https://github.com/Jermic/Android-Crack-Tool
- 下载
  - https://github.com/Jermic/Android-Crack-Tool/releases
- 介绍
  - 本软件集成了Android开发中常见的一些编译/反编译工具,方便用户对Apk进行逆向分析,提供Apk信息查看功能
- 功能
  - 反编译APK
  - 重建APK
  - 签名APK
  - 优化APK
  - DEX2JAR（APK2JAR）
  - JDGUI
  - 提取DEX
  - 提取XML
  - Class to smail
  - Apk信息查看
  - Unicode转换
- 截图

-

ByteCode Viewer

- `Bytecode Viewer` = `BCV`
- 主页
  - 官网
    - Bytecode Viewer - Java & Android APK Reverse Engineering Suite/Tool
  - github
    - Konloch/bytecode-viewer: A Java 8 Jar & Android APK Reverse Engineering Suite (Decompiler, Editor, Debugger & More)
- 功能
  - 概述：一款基于Java 8的 `jar` 和 `apk` 的反编译工具包
    - 包含反编译、编辑、调试等众多工具
      - 具体包含
        - 基于图形界面的：
          - 轻量级的Java的字节码查看工具
          - Java反编译器
          - 字节码编辑器
          - Smali汇编器
          - Baksmali汇编器
          - APK编辑器
          - Dex编辑器
          - APK反编译器
          - Dex反编译器
          - Procyon的Java反编译器
          - Krakatau
          - CFR的Java反编译器
          - FernFlower的Java反编译器
          - DEX2Jar
          - Jar2DEX
          - Jar-Jar
        - 以及
          - Hex查看器
          - 代码搜索器
          - 调试器
        - 等等
    - 还支持插件系统
      - 允许你与加载的类文件进行交互
        - 举例
          - 你可以写一个字符串的反混淆工具，恶意代码搜索器，或者其他的一些你所能想到的东西
- 主要特性
  - 在Bytecode Viewer的编译／反编译工具中集成了Krakatau
  - 集成了Smali／BakSmali－现在，你可以通过smali来编辑类文件和dex文件了
  - 支持APK／DEX－使用了Dex2Jar和Jar2Dex，可以轻松加载并保存APK文件
  - Java反编译器－Bytecode Viewer的反编译工具中集成了FernFlower，Procyon和CFR
  - 字节码编译器－CFIED的修改版
  - 十六进制查看器－由JHexPane驱动
  - 每一个反编译器／编辑器／查看器都是可以进行切换的，你可以选择每一个操作面板上所显示的元素组件
  - 功能完整的搜索系统－可以搜索字符串，函数，以及变量等信息
  - 系统完全支持使用Groovy脚本
- 截图

-

Android Decompiler

- 主页
  - dirkvranckaert/AndroidDecompiler: Decompile any APK
- 功能
  - 集成了多种工具去实现反编译apk得到java代码
- 集成了哪些工具
  - `Dex2Jar` : Version 0.0.9.15
  - `android-apktool` : Version 1.5.2
  - `JD-Core-Java` : Version 1.2
  - `Artistic Style (astyle)` : Version 2.04
- 支持平台
  - Mac
  - Unix类系统

用法

```
usage: decompileAPK.sh [options] APK-file

options:
 -o,--output dir     The output directory is optional. If not set the
                        default will be used which is 'output' in the
                        root of this tool directory.
 --skipResources     Do not decompile the resource files
 --skipJava         Do not decompile the JAVA files
 -f,--format         Will format all Java files to be easier readable.
                 However, use with CAUTION  This option might change
                 line numbers
 -p,--project        Will generate a Gradle-based Android project for you
 -h,--help         Prints this help message

parameters:
 APK-file              A valid APK file is required as input
```

decompile-apk

- 主页
  - venshine/decompile-apk: Decompile APK（反编译APK）
- 功能
  - 集成了众多工具去从apk中反编译出java源码
    - 集成了哪些工具
      - `Apktool`：v2.2.4
      - `dex2jar`：v2.1
      - `jd-gui`：v1.4.0
      - `jadx`：v0.6.1
      - `android-classyshark`：v8.0

Android逆向助手

- 作者：大眼仔~旭
- 简介
  - Android 逆向助手是一款针对安卓平台的强大逆向辅助软件，功能涵盖apk反编译打包签名；dex/jar互转替换提取修复；so反编译；xml、txt加密；字符串编码等。支持直接将文件拖放到源和目标文件这，不用每次都点浏览选择。
- 支持系统：
  - WinXP、Win7，Win2003
  - 注：其它系统没有测试
- 主要功能
  - 反编译apk
  - 重新打包成apk
  - 对apk进行签名
  - 反编译dex
  - 重新打包成dex
  - dex转jar
  - dex转ddx
  - dex导出成txt
  - 反编译so
  - jar转dex
  - 提取dex
  - 替换dex
  - 修复dex
  - 加密xml转txt
  - 字符串unicode编解码
- 特别说明：
  - **源文件**处支持文件或文件夹拖放
  - 必须安装 `.Net Framework 2.0` 框架
  - 部份功能依赖 `java运行环境`，因此必须安装java
- 下载地址
  - 地址1：https://pan.baidu.com/s/1jI9L4IM
  - 地址2：Android 逆向助手 v2.2 中文版 - 大眼仔旭
- 截图

-

Androguard

- 主页
  - androguard/androguard: Reverse engineering, Malware and goodware analysis of Android applications
- 文档
  - Welcome to Androguard's documentation! — Androguard 3.4.0 documentation
- 功能
  - 主要用来进行静态分析安卓程序
  - 也可以用第三方库去反编译安卓
    - 支持的第三方库
      - `DAD`
      - `dex2jar + jad`
      - `DED`
- 包括多个模块/子功能
  - `androrisk.py`：该模块用于分析apk危险级别
  - `androapkinfo.py`：该模块分析apk列出其中的文件类型、权限、4大组件、是否NDK反射等信息
  - `androaxml.py`：该模块用于展示apk androidmanifest.xml
  - `androgexf.py`：该模块生成函数调用图
  - `apkviewer.py`：该模块生成指令级别的调用图
  - `androlyze.py`：该模块为交互分析环境
- 特点
  - 用 `Python` 写的
    - 支持多个平台：Linux/Windows/Mac
  - 支持多种模式
    - 命令行模式
    - 图形界面模式
    - 被当做库文件使用
- 截图

-

涉及子领域

反代码混淆

背景：使用反编译工具只能看到混淆之后的代码结构，看不到混淆之前的原始的代码

而如何反混淆：

- 如果有：`mapping.txt` 文件
  - 有机会反混淆，恢复和还原出原始代码
  - 背景：
    - 有安卓项目源码的开发者，在折腾ProGuard时，才有（生成）的mapping.txt文件
  - 实际上：作为要破解的人，往往没有
- 如果有：源文件和行号文件
  - 有机会反混淆，恢复和还原出原始代码
  - 背景：
    - 许多APK开发者为了在崩溃时保存源文件类名、行号等信息会在APK混淆时添加以下规则保留源文件信息
      - `-keepattributes SourceFile,LineNumberTable`
  - 实际上：作为要破解的人，往往没有

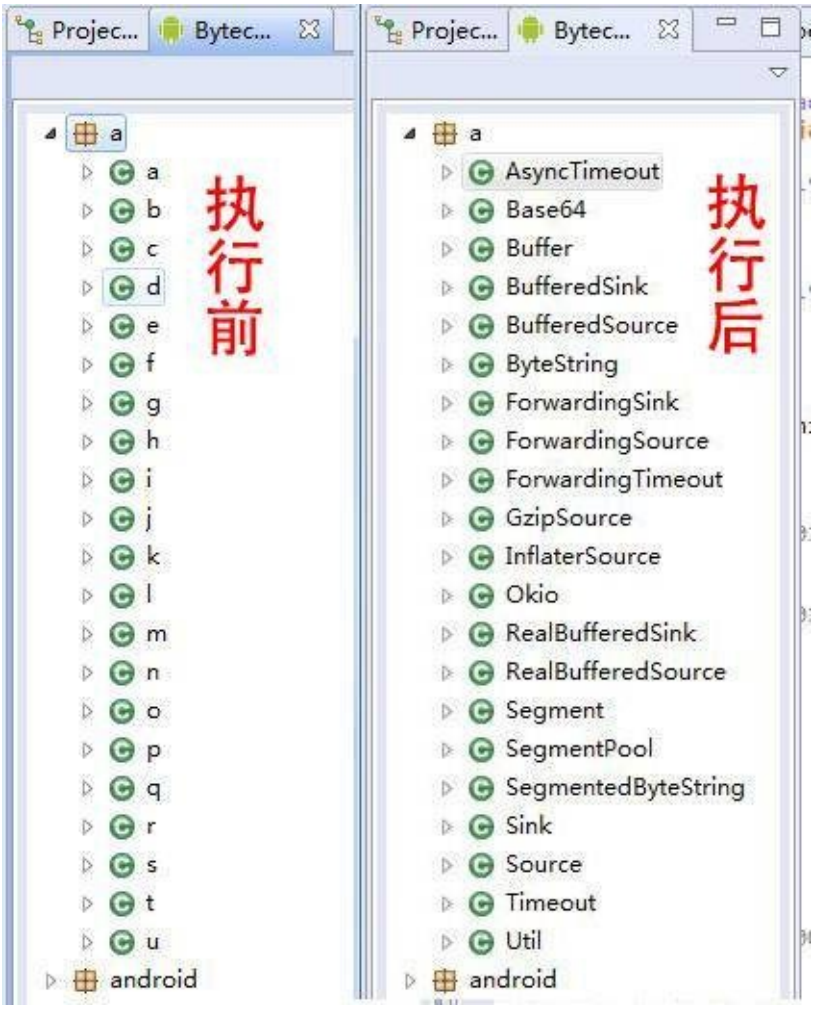
一些反混淆工具

- `JEB` = `JEB Decompiler`
  - JEB2被称为反混淆神器
  - 官网：
    - https://www.pnfsoftware.com/
  - 一些用于反混淆的插件
    - S3cuRiTy-Er1C/JebScripts: Jeb public scripts
    - flankerhqd/jebPlugins: Various Jeb plugins, including obfuscation restore
    - enovella/jebscripts: A set of JEB Python/Java scripts for reverse engineering Android obfuscated code
- `Simplify`
  - CalebFenton/simplify: Generic Android Deobfuscator

举例

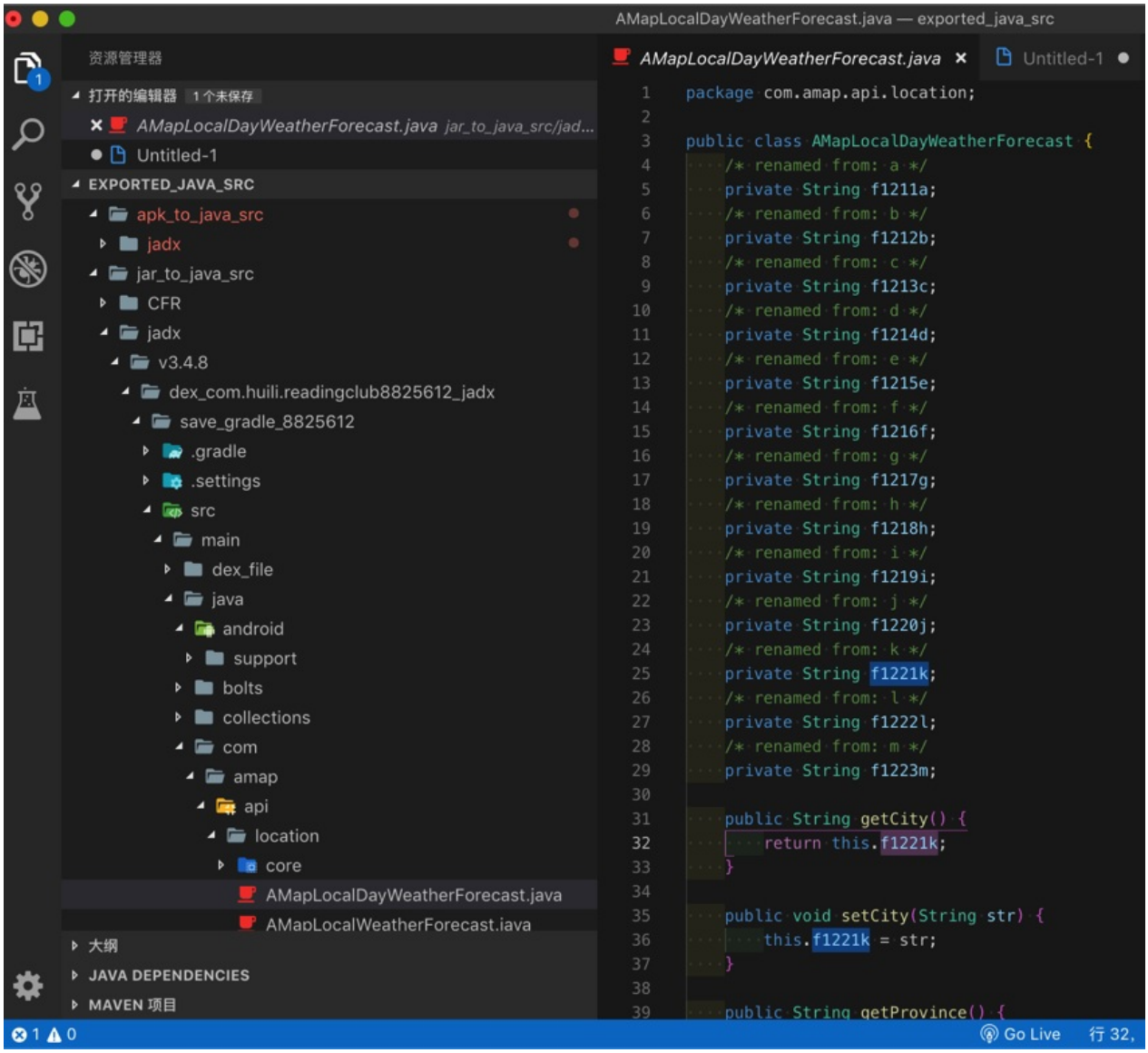
JEB反混淆效果

如果有了源文件和行号（只可惜逆向人员往往没有），则可以反混淆：



Jadx反混淆效果

比如：



详见：反混淆 · 安卓反编译利器：jadx

系列子教程

- 父教程
  - 安卓应用的安全和破解
    - Android逆向开发
- 子教程
  - 安卓反编译利器：jadx

附录

下面列出相关参考资料。

参考资料

- 【已解决】安卓逆向LiftFileManager：so库文件libRehADGd.so
-
- 【基本解决】尝试破解安卓apk马蜂窝去得到java源码 – 在路上
-
- Ghidra逆向分析工具使用与实战_ghidra 使用_Ba1_Ma0的博客-CSDN博客
- NSA开源逆向工具Ghidra入门使用教程 - 安全内参 | 决策者的网络安全知识库
- borisf/classyshark-user-guide: Everything you want to know how to use ClassyShark
- apkanalyzer | Android 开发者 | Android Developers
- Download Android Studio and SDK tools
- Android Killer V1.3.1正式版 发布信息 - 吾爱破解
- Making the most of the APK analyzer – Android Developers – Medium
- Analyze your build with APK Analyzer | Android Developers
- Android apk分析工具：APK Analyzer - 简书
- android - decompiling DEX into Java sourcecode - Stack Overflow
- decompiler - how to use DEXtoJar - Stack Overflow
- Android反编译 - 简书
- Android反编译简单实战 - 知乎
- Android APK 反编译实践 - 简书
- Android混淆（ProGuard）从0到1 - 简书
- 如何反编译Android 的apk/dex/odex，获得源码 – 码农日记
- 乐固壳分析 - bamb00 - 博客园
- 5分钟学会基于Xposed+DumpDex的apk快速脱壳方法 - 简书
- 腾讯加固纯手工简易脱壳教程 - 『移动安全区』 - 吾爱破解 - LCG - LSG |安卓破解|病毒分析|破解软件|www.52pojie.cn
- 花生日记APP邀请注册机实战（360加固脱壳） – Silkage's Blog
- Android逆向之路---脱壳360加固 - 简书
- 26款优秀的Android逆向工程工具 - 简书
- 【工具分享】Radare 2之旅：通过crackme实例讲解Radare 2在逆向中的应用（上） - 安全客，安全资讯平台
- radare/radare2: unix-like reverse engineering framework and commandline tools
- radare
- Radare2使用全解 - 先知社区
- radare2逆向笔记 - 有价值炮灰 - 博客园
- 安卓 APK 反汇编工具 AndroidKiller 1.2 中文绿色免费版 - 大眼仔旭
- AndroidKiller下载|AndroidKiller(安卓APK反汇编工具)下载 v1.3.1绿色中文版_ - pc6下载站
- GDA反编译器与其他android逆向工具对比 - 知乎
- GDA介绍-功能强大的交互式Android反编译分析工具
- EverEdit：带来惊喜的国产软件 - 简书
- 原创 使用AndBug调试Android Java Bytecode-『Android安全』-看雪安全论坛
- lambdalang/DexExtractor
- DexExtractor的原理分析和使用说明 - Fly20141201. 的专栏 - CSDN博客
- system-arm_md5_6395c2f1451dbbed027d7293ab39a6e7.img.tar.gz
- 使用drizzleDumper脱去某数字公司的壳 - suwenlai的博客 - CSDN博客
- 一种常规Android脱壳技术的拓展（附工具） - FreeBuf互联网安全新媒体平台
- DrizzleRisk/drizzleDumper: drizzleDumper是一款基于内存搜索的Android脱壳工具
- pxb1988/dex2jar: Tools to work with android .dex and java .class files
- dex2jar | Penetration Testing Tools
- JesusFreke/smali: smali/baksmali
- Java Decompiler
- java-decompiler/jd-gui: A standalone Java Decompiler GUI
- Releases · java-decompiler/jd-gui

- jd-gui-osx-1.4.1.tar
- Release Luyten v0.5.4 · deathmarine/Luyten
- deathmarine/Luyten: An Open Source Java Decompiler Gui for Procyon
- Releases · deathmarine/Luyten
- Bytecode Viewer - Java & Android APK Reverse Engineering Suite/Tool
- Konloch/bytecode-viewer: A Java 8 Jar & Android APK Reverse Engineering Suite (Decompiler, Editor, Debugger & More)
- Bytecode Viewer-一款基于Java 8的Android APK逆向工具包 - 安全客，安全资讯平台
- 010 Editor - Professional Text/Hex Editor with Binary Templates
- SweetScape Software Inc - Download 010 Editor
- IDA: About
- IDA Support: Download Center
- 安卓逆向学习笔记 (4) - 使用IDA Pro动态调试so文件 - 蜗牛 - CSDN博客
- 【手机脱壳】MT2+VXP+FDex2实现免Root脱壳 - Powered by Discuz!
- HangZhouCat/ReaverAPKTools: 逆向APK工具
- 文件管理神器 MT Manager v2.6.1 for Android-心海e站
- MT管理器2.0(bin.mt.plus) - 2.6.0 - 应用 - 酷安网
- MT管理器2.6.1公测版（第三版） - Powered by Discuz!
- 介绍 - MT管理器
- Android 逆向助手 v2.2 中文版 - 大眼仔旭
- Jermic/Android-Crack-Tool: Android crack tool For Mac
- 【已解决】用apktool，dex2jar，jd-gui去反编译安卓apk查看app源码
- Apktool - A tool for reverse engineering 3rd party, closed, binary Android apps
- Apktool - How to Install
- Android 各种脱壳工具使用 - 简书
-