Android 逆向工具篇—反编译工具的选择与使用

2019-12-04 阅读 3.2K

本文被 1 个清单收录，推荐清单

Python爬虫安卓APP逆向

作者 | 天天记小本子上的lilac
来源 | CSDN

今天给大家介绍一下Android App 在Java层的逆向工具。

逆向工具的介绍

在过去，当我们想要了解一个 app 内部运作细节时，往往先通过 ApkTool 反编译 APK，生成 smali 格式的反汇编代码[1]，然后大佬和老手直接阅读 smali 代码，适当的进行修改、插桩、调试，经过一定的经验和猜想，理解程序的运行逻辑和加解密细节，比如如下的 smali 代码。

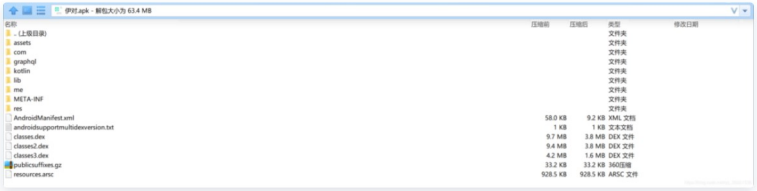
```
# direct methods
.method public constructor <init>(Lcom/xingin/skynet/b/a;)V
    .registers 6

    const-string v0, "argumentsLoader"

    invoke-static {p1, v0}, Lkotlin/jvm/internal/k;->b(Ljava/lang/Object;Ljava/lang/String;)V

    .line 0
    invoke-direct {p0}, Ljava/lang/Object;-><init>()V

    .line 19
    iput-object p1, p0, Lcom/xingin/skynet/f/b;->r:Lcom/xingin/skynet/b/a;

    const-string p1, "platform"

    .line 20
    iput-object p1, p0, Lcom/xingin/skynet/f/b;->a:Ljava/lang/String;

    const-string p1, "deviceId"

    .line 21
    iput-object p1, p0, Lcom/xingin/skynet/f/b;->b:Ljava/lang/String;

    const-string p1, "device_fingerprint"

    .line 22
    iput-object p1, p0, Lcom/xingin/skynet/f/b;->c:Ljava/lang/String;

    const-string p1, "device_fingerprint1"

    .line 23
    iput-object p1, p0, Lcom/xingin/skynet/f/b;->d:Ljava/lang/String;

    const-string p1, "versionName"
```

smail

我们只要先这样，再那样，最后再这样，对对对，就这样，一个程序的加密就被破解出来了。
是不是迫不及待想来一次App的逆向之旅了?

事实上，这种方式对小白实在不友好，有没有更加友好的方式呢？当然是有的，如果你百度或者 google 搜索逆向相关的教程和分享，很容易就会发现下面这三个工具。在介绍工具之前，我们先补充一下APK结构的知识，我们以伊对这个社交 Apk 为例。

APK 文件其实是一种特殊的 zip 格式，我们可以直接用 360 压缩或者别的压缩工具打开。



为了满足自身的功能和设计，几乎每一个都会在基础的文件结构上添加不少东西，但有六个部分是不变的，我们罗列和称述一下。

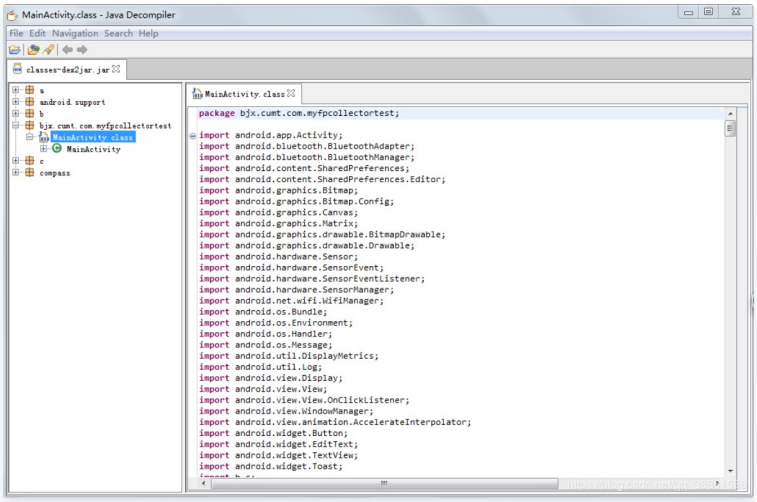
文件或目录	作用
META-INF/	描述apk包信息的目录，主要存放了签名信息，配置信息，service注册信息
res/	存放apk资源文件的目录，比如图片、图标、字符串、样式、颜色
assets/	同样是存放apk资源文件的目录，但和res有差异，和我们关系不大
resources.arsc	资源索引，包含不同语言环境中res目录下所有资源的类型、名称与ID所对应的信息
lib/	存放so文件，越来越多的应用由C/C++编写核心代码，以SO文件的形式供上层JAVA代码调用，以保证安全性，这个目录是逆向解密关注的重点
classes.dex(一个或数个)	Android程序运行在Dalvik虚拟机上，而dex就是Dalvik虚拟机的可执行文件, 相当于Windows平台中的exe文件，通过反编译dex，可以获得apk源码（这个说法不很准确，但方便理解）
AndroidManifest.xml	清单文件，包含了App大量的的配置信息，比如包名、应用需要拥有的权限（打电话/录音/网络通信等等）、以及所有的界面和程序组件的信息,无法解压apk时直接打开，因为清单文件在apk打包过程中被编译成了二进制格式文件

接下来我们介绍以下反编译工具，看一下反编译工具的作用

工具	作用

工具	作用
ApkTool	解析resources.arsc，AndroidManifest.xml等文件，反编译dex文件为smali源码
Dex2jar	将dex文件转化为jar文件
Jd-gui	反编译jar，查看java源码

比如使用 Dex2jar+Jd-gui, 最终得到这样的结果。



是不是感觉友好很多？只需要cmd敲七八行命令就可以得到java源代码。[2]

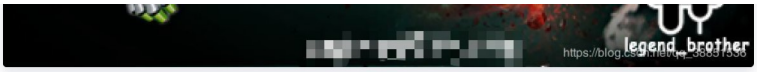
这样做肯定没问题，但能不能更加简单一些呢？能不能直接将Apk拖到什么软件里，然后电脑屏幕发亮，蓝底黑字，日志和指令不停流淌，过一会儿完整的java代码和apk结构就显现出来？

前人种树后人乘凉，真的有不少这样的工具，通过这一类高集成度的逆向工具，我们可以方便快捷对Apk进行逆向分析。

这里介绍几款：**JADX，JEB，Android Killer，GDA。**

1.1 Android Killer

首先说一下Android killer，这也是我接触的第一个反编译工具，开场界面非常酷。



它集成了Apktool，Jd-Gui等工具实现了拖拽式反编译，功能强大，并且可以安装插件，使用android killer进行smali一键插桩非常的畅爽。但由于更新慢，逐渐老旧，使用它反编译apk越来越力不从心，而且它只可以在windows平台使用，所以我们这边不做过多介绍，但不可否认它有一些非常棒的功能。

1.2 GDA

GDA是国人制作的一款反编译神器，功能强大，灵活至极。

我们先说一下它的优点，仅2.6M大小，简直不可思议，而且它不需要JDK环境，测试时反编译七八十兆大的apk也不会卡死，除此之外，它还附带有反混淆，查壳等功能……

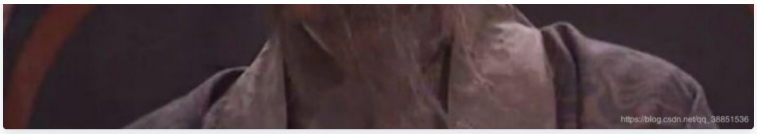
接下来我们说一下它的缺点：

一、反编译出来的java代码展示性不够友好，变量名不够友好，大多是v0, v1, p1等（更像是原生寄存器的命名法）。

二、工具的文档和文章不算充分，在搜索逆向工具教程时，比较难找到其相关介绍。

三、只能在windows平台运行和使用，不支持mac等其他平台，这很遗憾。

四、单论反编译效果，JADX太好用了……



1.3 JADX vs JEB

JADX：免费，开源，强大，更新快

优点：反编译能力强，代码结构好，变量名合理，支持多平台，完全就是个和我一样完美的靓仔，是我心中逆向分析APK的第一工具。[3]

缺点：比较吃内存，一个50M大小的APK，使用JADX反编译就需要占用4G左右内存。

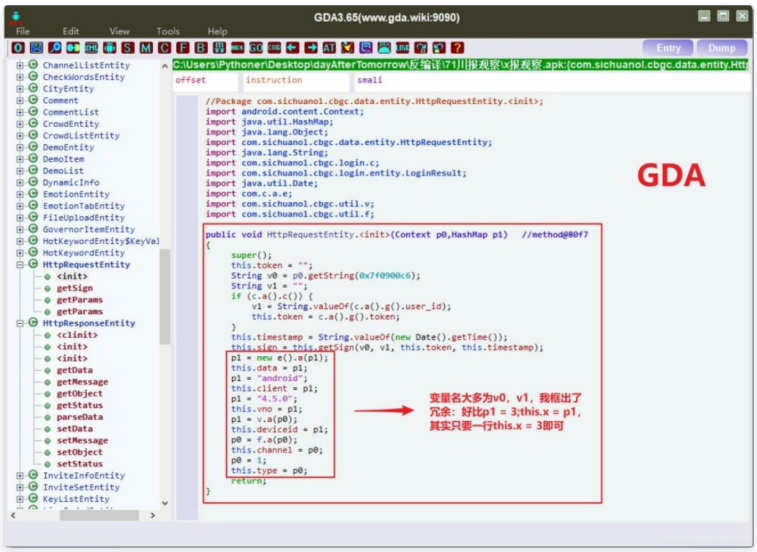
JEB：收费，可动态调试

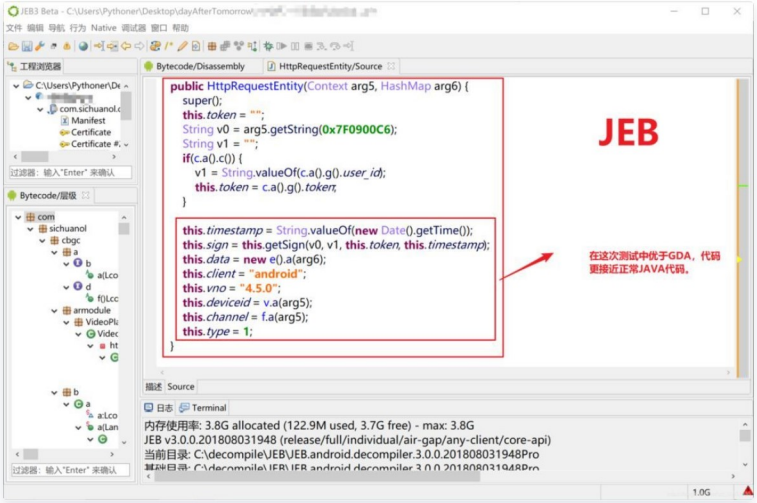
优点：可以动态调试，而且JEB吃内存比较少，反编译同等大小的APK，JEB只要Jadx一半内存就可以搞定。同时，端口自动转发带来了舒适方便的动态调试体验。

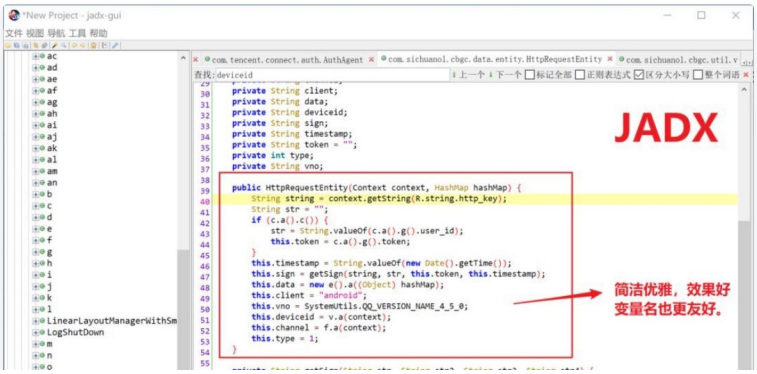
缺点：反编译出来的java伪代码展示性不够友好，变量名不够友好，大多是v0, v1, p1。

我们测试对比一下GDA，JEB，JADX三个软件的反编译效果

如图：HttpRequestEntity这个构造方法[4]反编译效果的对比







可以看出，JADX反编译的结果最为接近正常的java代码，在后续使用过程中，你还会发现它全局搜索功能的便捷和舒适。

接下来我们安装一下这三个反编译工具
链接：https://pan.baidu.com/s/1SDM9f2HxxbNzGg2XVBymPA
提取码：i1k9

你可能会困惑，上面花了不是不少时间，分析和比较了几个软件的优缺点，又用证据说明了JADX是像你一样的靓仔，那我们肯定毫不犹豫选择你啊，呸，选JADX啊。为什么要三个都装呢?

原因很简单，在技术娴熟和精通之前，APP逆向是门不折不扣的玄学，既然是玄学，就会有很多不可控、随机、稀奇古怪的状况。当JADX中一个变量模糊不清的时候，你就需要去JEB中看一下它的反编译结果，或者使用JEB进行动态调试。当使用JEB 遇到头疼的APK混淆时，就可以试一下开启JADX的反混淆功能。因为这几个软件的逆向原理是不同的，所以在分析具体APK时各有优势，它们的功能可以互补。多个工具结合使用可以一定程度弥补个人能力的不足，只需要几百M空间放它们即可，何乐而不为呢。

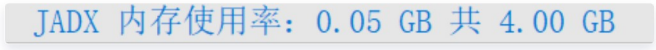
1.4 JADX的配置和使用

直接下载，找个合适的地方解压即可，按照自己的操作系统打开相应的文件即可。

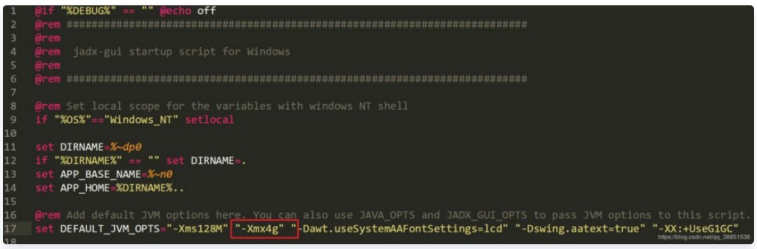
可能出现的两个问题：

为什么双击jadx-gui.bat 出现控制台一闪而过，没有正确出现界面
因为你的JAVA_HOME环境变量没有正确配置，但也不排除是JDK版本的问题。

反编译卡死、闪退
Apk超过50M就很容易出现OOM(OutOfMemoryError)，在win中，它默认使用4G内存，可以查看界面底部居中部分。



如果你的window系统是8G或者更高运行内存，我们可以修改参数进行扩容。文本模式打开jadx-gui.bat，将被框出的内容数值改为8g或者更大，如果电脑运行内存更大，也可以改成更高的数值。100M以内的App，8G内存足够了。



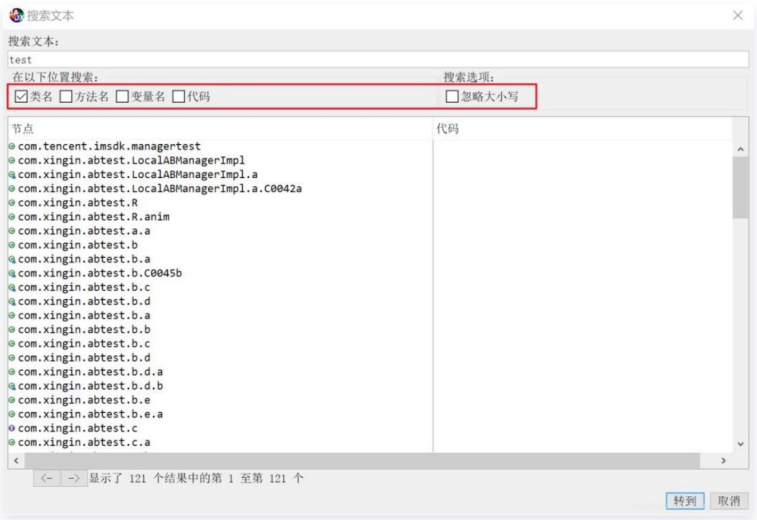
接下来重新打开JADX，内存就已经更改了。

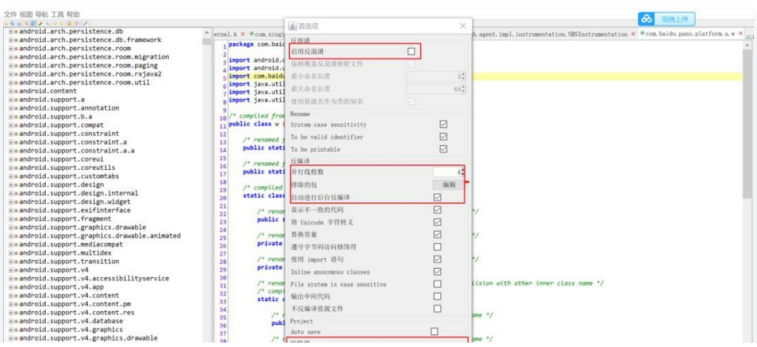
如果你的windows系统只有4G运行内存，我们依然有很多办法使用jadx，实战中再说。

接下来说一下mac中的扩容，文本模式打开jadx-gui这个文件，找到和上述类似的位置，更改为更大的值即可。

JADX的使用和快捷键

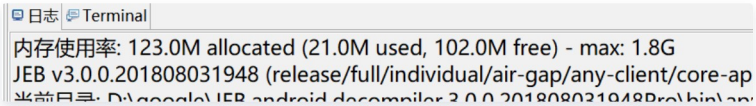
1. 搜索代码、类、方法——Ctrl+N，建议不要使用左上角的搜索类/搜索文本，因为图标太小，很容易按错，如果你第一次搜索用搜索类，第二次搜索时选择搜索文本，那第一次的搜索内容和设置是不会记录下来的。所以不如直接看一下工具栏中的快捷键，只用一个。
2. 文件-首选项中，如果内存够用，我建议勾选“自动进行后台反编译”，因为Jadx默认只有在你展开内容或者搜索内容时，才会开始反编译。不要勾选Unicode自动转义，否则代码中的中文会被转成unicode，不方便识别和搜索。
3. Ctrl+鼠标左键，可以跳转到方法内部，几乎所有的代码编辑器都是这样。
4. 别的一些可以修改的地方我放在了下图红框里，比如反编译线程数，线程数越多，反编译越快，但占用内存也越多，建议根据电脑性能调整，不调整也OK。反混淆一般不用开。



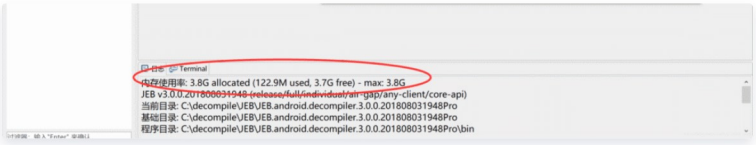


1.5 JEB的配置和使用

同样直接下载，找个合适的地方解压，按照自己的操作系统打开相应的文件即可。
我们同样要进行JEB的扩容，默认为1.8G，我们需要进行更改。



更改前



更改后

Windows中JEB的扩容

将整个红框内容替换如下

```
:startjeb
set JEB="%~dp0bin\jeb1.exe"
if exist %JEB% goto :runlauncher
%JAVA% -jar -Xmx4g -Xms4g "%~dp0bin\app\jebc.jar" %*
exit 0
:runlauncher
%JEB% %*
exit 0
```

mac如何扩容我了解不多，如果遇到问题可以和我探讨。

1.6 GDA的配置和使用

只有windows可以用，exe直接点开即可，以后会用到，到时候再说

[1]: Smali是dex文件反编译的结果，可以说，smali语言是Dalvik的反汇编语言，下文会介绍DEX

[2]: 我们这一个系列的教程针对的是无壳App，而当你自己拿到一个未知的App时，第一步要做的一定一定得是查壳。

[3]: 得到的并不是Java源代码，这是个错误的说法，但对初学者来说比较容易理解。不管你通过什么工具反编译apk，得到的java代码都和Apk开发时的源代码相差甚远。我们能得到的仅仅是一种伪代码，它可能存在错误的逻辑、奇怪的变量名、各种各样的error，但代码总体上是靠谱的。

[4]: 为了阅读和讲解的循序渐进，我们这里并没有把IDA考虑进来，IDA是神器，在后面是避免不了的。

[5]: 这条注释针对没有JAVA基础的小伙伴，构造函数在JAVA中非常常见，JAVA是一门面向对象的语言，构造函数是一种特殊的函数。其主要功能是用来在创建对象时初始化

对象， 构造函数与类名相同。如果听不懂也没关系，知道是个函数即可，后续也会慢慢补充JAVA知识。

Android Java Windows 举报

1 条评论 我来说两句

用户5034366 23 小时前

博主要不要试试 FakerAndroid https://github.com/Efaker/FakerAndroid

回复