目录

安卓逆向利器：JEB

- 最新版本：`v1.0.2`
- 更新时间：`20240717`

简介

介绍安卓逆向中的好用工具JEB，主要用于静态分析中的反编译apk/dex，和动态调试安卓程序。先是JEB概览；然后是如何下载和运行JEB；然后介绍JEB的功能和界面，包括Bytecode/Hierarchy、单个文件多个显示模式、底部多个Tab页、保存项目为jdb2文件、多主题显示效果、支持的反编译器、脚本功能、插件机制；然后介绍JEB的静态分析，包括反编译安卓apk，其中包含反编译出java和解析so库文件；反编译出java包括单个反编译、全部反编译、反编译效果对比，其下包括JEB和jadx对比；接着是JEB动态调试；以及JEB使用心得，包括页面显示方面和反编译java代码方面；以及JEB常见问题，包括页面显示问题和反编译java代码问题。

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_tool_jeb: 安卓逆向利器：JEB

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

- 安卓逆向利器：JEB book.crifan.org
- 安卓逆向利器：JEB crifan.github.io

离线下载阅读

- 安卓逆向利器：JEB PDF
- 安卓逆向利器：JEB ePub
- 安卓逆向利器：JEB 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

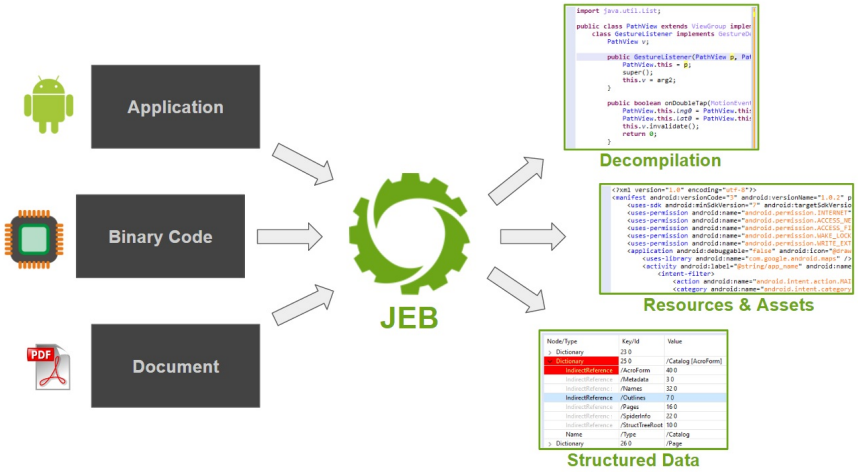
关于作者

关于作者更多介绍，详见：

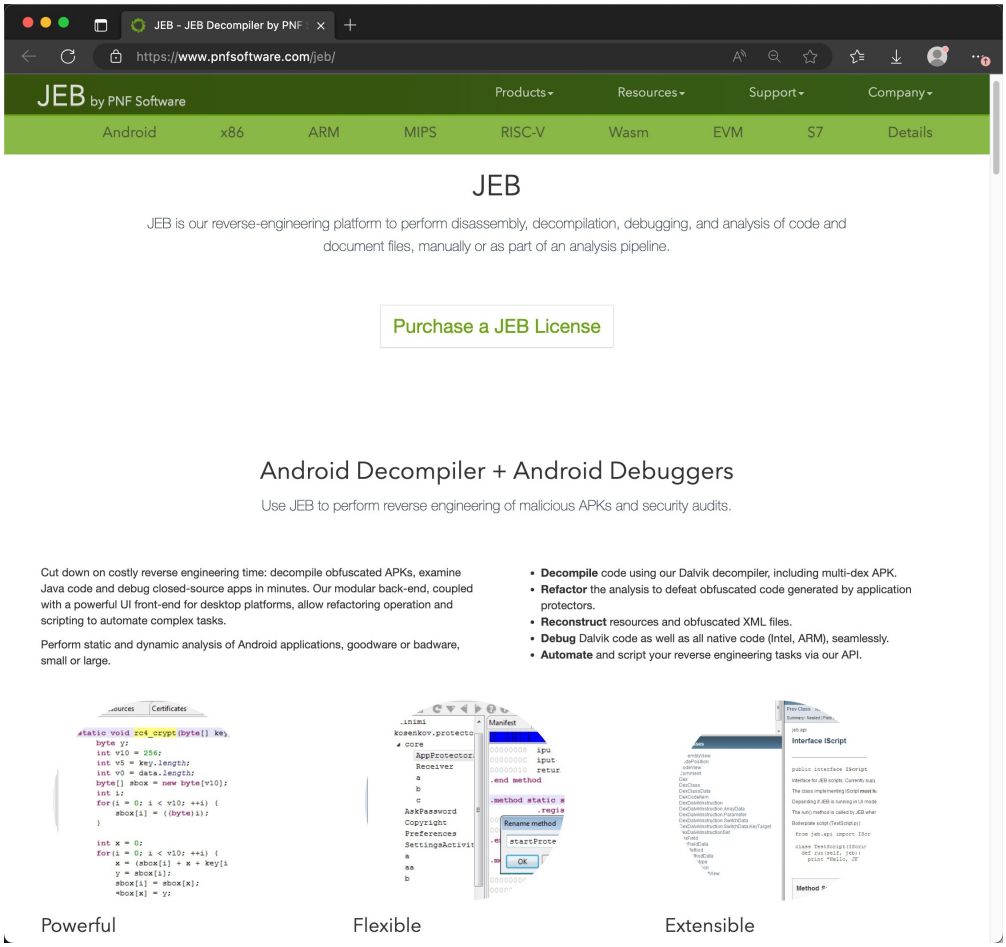
关于CrifanLi李茂 – 在路上

JEB概览

- JEB
  - 是什么：安卓逆向工具
    - 一款专业逆向工具，主要用于安卓逆向
  - 一句话描述
    - Android Decompiler + Android Debuggers
  - 主要用途
    - 安卓逆向领域的
      - 静态分析中的：反编译 apk / dex
        - 最大特点：反编译效果，比jadx还好
      - 动态调试中的：调试安卓app
  - 特点
    - 模块化
      - 安卓模块=Android modules
        - The Android modules for JEB provide static and dynamic analysis capabilities to analyze Android applications, goodware or badware, small or large
      - 原生代码分析模块=native code analysis modules
        - The native code analysis modules provide advanced code analyzers and decompilers for Intel x86, ARM, MIPS, RISC-V processors, as well as WebAssembly modules and Ethereum smart contracts
      - PDF模块=PDF module
        - Our PDF module for JEB can be used to manually or automatically reverse engineer and assess PDF documents
  - 功能
    - 静态分析
      - 反编译器
        - Android Decompiler
          - 反编译decompile安卓app
            - 支持反混淆（的代码的重构）
            - 重构资源和反混淆XML文件
          - 核心流程



        - 也支持其他反编译器
            - Intel x86 Decompiler
            - ARM Decompiler
            - MIPS Decompiler
            - RISC-V Decompiler
            - WebAssembly Decompiler
            - Ethereum Decompiler
            - Simatic S7 PLC Program Decompiler
        - PDF Document Analyzer
        - 安全审计恶意安卓app
    - 动态调试
        - Android Debuggers
            - 调试debug安卓app
                - 调试Dalvik代码和native原生代码（Intel和ARM）
    - 支持（调用API实现）脚本自动化处理（安卓逆向）
- 主页
    - JEB Decompiler by PNF Software
        - JEB Android Decompiler - JEB Decompiler by PNF Software



crifan.org，使用署名4.0国际(CC BY 4.0)协议发布 all right reserved，powered by Gitbook最后更新：2024-07-17 22:55:47

下载运行JEB

- JEB版本概述
  - 收费的：`JEB Pro`、`JEB Android`
    - 用于演示功能的，可以免费下载的：`JEB Demo`
  - 免费的：`JEB CE` = `JEB Community Edition`
- JEB的不同版本功能对比
  -

Features Matrix

This table presents which analysis modules and core features ship with our various license types.

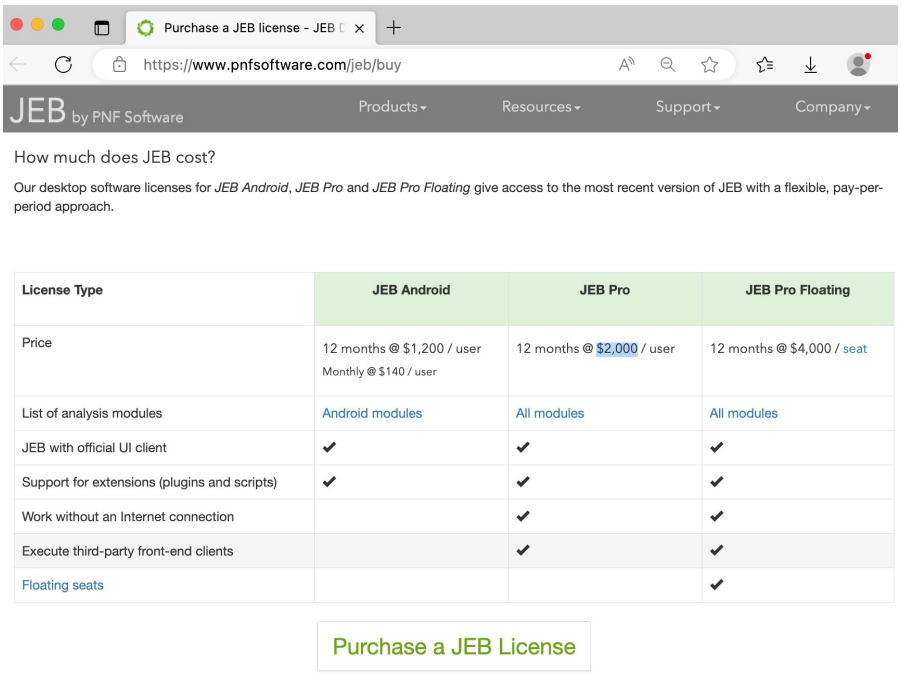
Build Type >	JEB CE Community Edition	JEB Android	JEB Pro (or Pro Floating)
Base			
Suitable for enterprise or professional use			✔
Decompilers			
Dalvik Decompiler (a.k.a. Android Decompiler)		✔	✔
Deobfuscation module for "virtualized" code (details)			✔
Advanced deobfuscators requiring native code emulation (JNI) (details, details)			✔
Java Decompiler		✔	✔
Intel x86 Decompiler	✔		✔
Intel x86-64 Decompiler	✔		✔
ARM Decompiler		✔	✔
ARM64 Decompiler		✔	✔
MIPS Decompiler			✔
MIPS64 Decompiler			✔
RISC-V Decompiler			✔
S7 PLC Block Decompiler			✔
WebAssembly Decompiler			✔
Ethereum Decompiler for Smart Contracts (EVM Decompiler)			✔
Diem (Libra) Decompiler for Move modules (open-sourced)			✔
Base Plugins			
Archive files (zip, 7z, tar)	✔	✔	✔
Image files (bmp, png, jpg, gif, ico)	✔	✔	✔
Structured text files (html, xml, json)	✔	✔	✔
Certificate files (x.509)	✔	✔	✔
Application Processors			
Android APK (incl. dex, odex, x-apk)	✔	✔	✔
Windows PE/PE64/COFF, PDB	✔	✔	✔
Linux ELF/ELF64	✔	✔	✔
Mach-O, Mach-O/FAT	✔	✔	✔
Intel HEX (ihex)	✔	✔	✔
Chrome Extension (crx)	✔	✔	✔
WebAssembly module (wasm)		✔	✔
Ethereum contracts (evm)		✔	✔
Disassemblers			
Dalvik Disassembler	✔	✔	✔
Intel x86 Disassembler	✔	✔	✔
Intel x86-64 (AMD64) Disassembler	✔	✔	✔
ARM (32, Thumb) Disassembler	✔	✔	✔
ARM64 (Aarch64) Disassembler	✔	✔	✔
MIPS Disassembler	✔	✔	✔
RISC-V Disassembler	✔	✔	✔
Atmel AVR 8-bit Disassembler	✔	✔	✔
Miscellaneous Binary Parser Plugins			
Flutter/Dart AOT Snapshot Parser (limitations)	✔	✔	✔
Debuggers			
Dalvik Debugger (Android)		✔	✔
Intel x86 Debugger (GDB/LLDB all platforms, incl. Android)	✔	✔	✔
Intel x86-64 Debugger (GDB/LLDB all platforms, incl. Android)	✔	✔	✔
ARM Debugger (GDB/LLDB all platforms, incl. Android)		✔	✔
ARM64 Debugger (GDB/LLDB all platforms, incl. Android)		✔	✔
MIPS Debugger (GDB/LLDB all platforms, incl. Android)		✔	✔
Document Parsers			
Adobe PDF			✔
FAT file systems (file access table)		✔	✔ (plugin)
Platform			
Persist JEB Projects to JDB2 Databases	✔	✔	✔
Interactivity (refactoring, renaming, commenting, etc.)	✔	✔	✔
Graphing (CFG, Callgraphs, etc.)	✔	✔	✔
Support for multiple Artifacts in a same Project		✔	✔
Extensions - Client scripts in Python	✔	✔	✔
Extensions - Back-end plugins in Java	✔	✔	✔
Automation / headless launcher			✔
Work offline			✔

官网免费版

- JEB免费版
  - 有2个
    - 社区版=JEB Community Edition = JEB Home Edition x86
      - 注：不支持Android（的arm），仅支持（32位和64位的）x86
      - 下载页面：JEB Community Edition - JEB Decompiler by PNF Software
    - 演示版=JEB Demo
      - 注：功能有很多限制，主要用于试用（功能演示）
      - 下载页面：Download a trial version of JEB - JEB Decompiler by PNF Software

JEB收费版

- JEB收费版
  - JEB Pro
    - 下载
      - 要先购买
        - Purchase a JEB license - JEB Decompiler by PNF Software
          - 价格



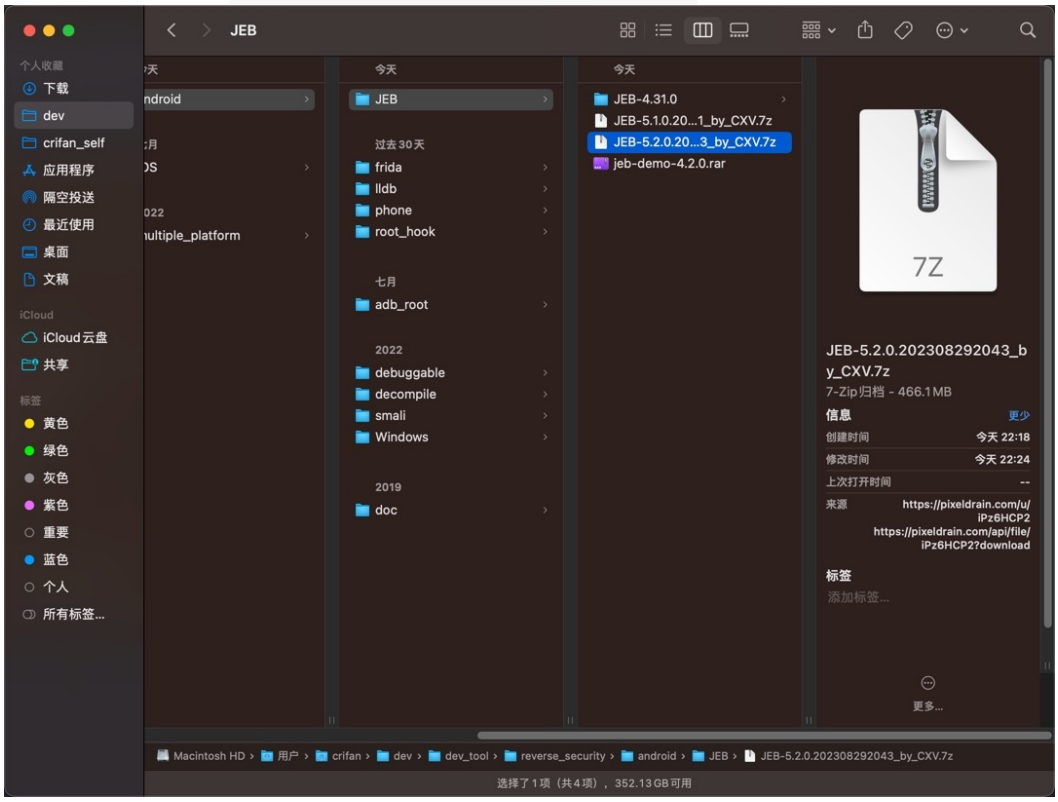
JEB收费版的破解版

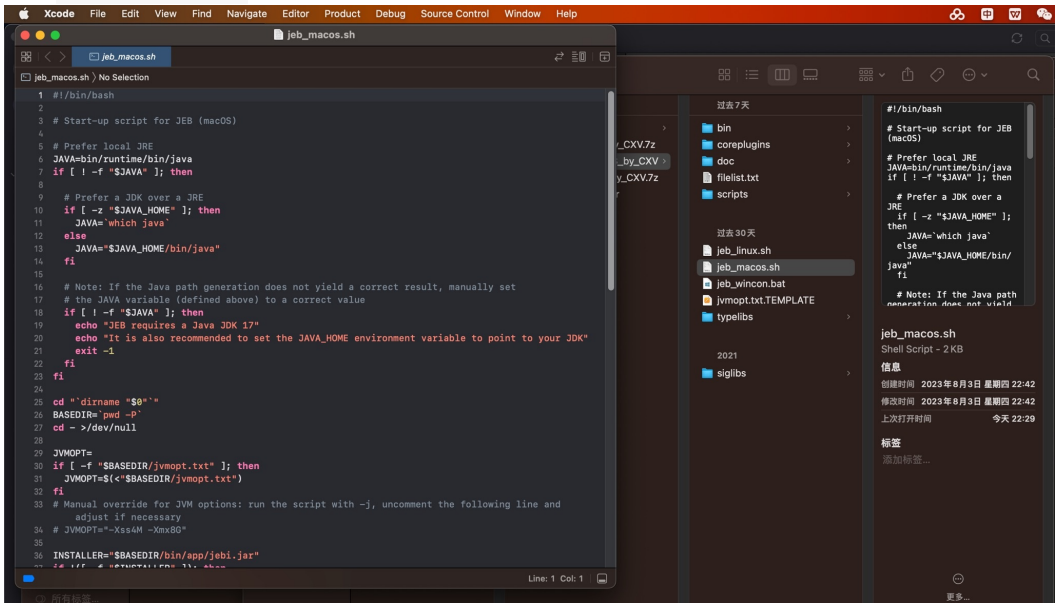
仅供学习研究技术用

此处提供破解版，仅供学习研究技术用，请勿用于其他非法和商业用途。请支持正版。

如有侵权，请联系笔者删除

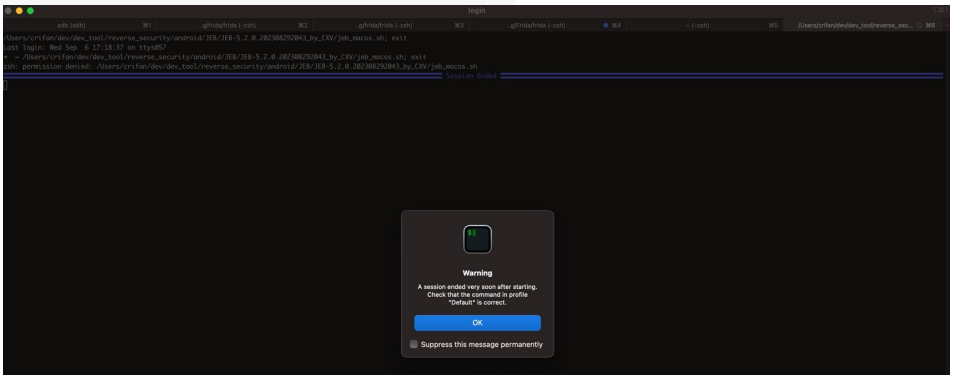
- 下载JEB

- 下载地址：JEB-5.2.0.202308292043_by_CXV.7z ~ pixeldrain
  - 下载得到 460MB+ 的：`JEB-5.2.0.202308292043_by_CXV.7z`
  - 

- 运行JEB
  - 解压后，运行对应脚本
    - Mac的：`jeb_macos.sh`
    - 

    - Win的：`jeb_wincon.bat`
    - Linux的：`jeb_linux.sh`
  - 运行方式
    - 如果双击脚本，无法自动运行的话
      - 比如
        - 调用文本编辑器打开sh脚本
        - 报其他错误：`A session ended very soon after starting. Check that the`

`command in profile "Default" is correct.`

- 

- 则去加上可执行权限
  - Mac中

```
chmod +x jeb_macos.sh
```

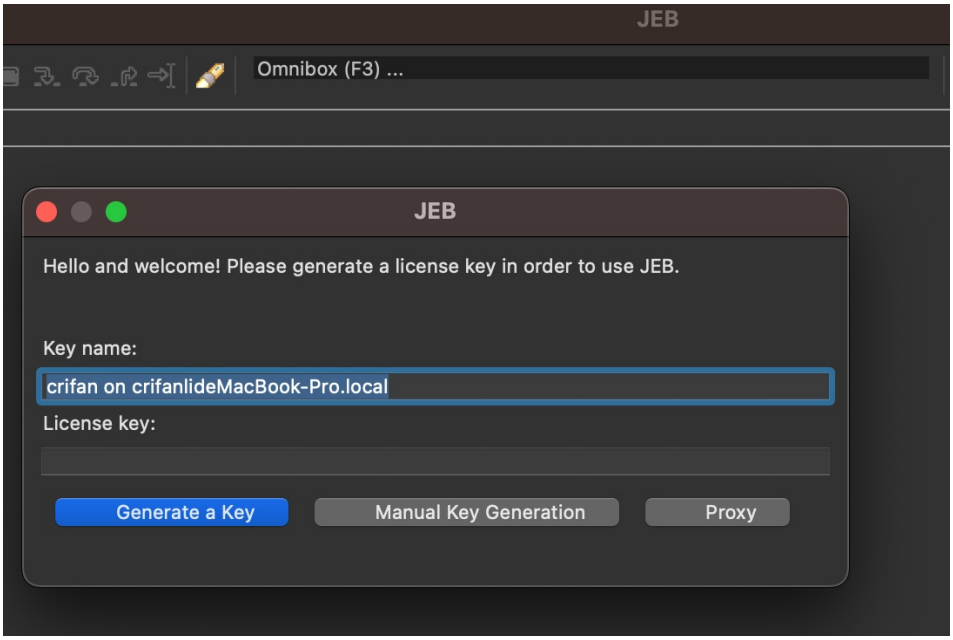
- 再放到终端terminal/shell中去运行
  - 常见终端
    - Mac：`iTerm2` / `shell`

```
./jeb_macos.sh
```

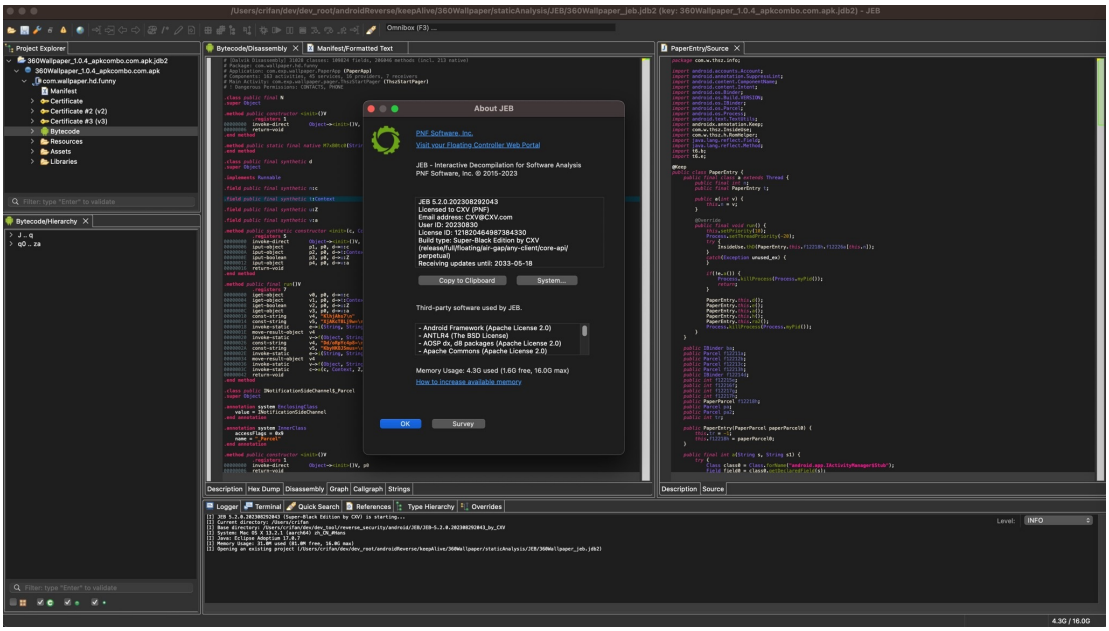
      - 附录：此处终端启动的log日志

```
➜  JEB-5.2.0.202308292043_by_CXV ./jeb_macos.sh
[I] JEB 5.2.0.202308292043 (Super-Black Edition by CXV) is starting...
[I] Current directory: /Users/crifan/dev/dev_tool/reverse_security/android/JEB/JEB-5.2.0.202308292043_by_CXV
[I] Base directory: /Users/crifan/dev/dev_tool/reverse_security/android/JEB/JEB-5.2.0.202308292043_by_CXV
[I] System: Mac OS X 13.2.1 (aarch64) zh_CN_#Hans
[I] Java: Eclipse Adoptium 17.0.7
[I] Memory Usage: 31.4M used (80.6M free, 16.0G max)
```

    - Linux：自带shell
    - Win：`cmd.exe`
- 首次启动时
  - 需要去生成key，再点击继续，即可
    - 点击 `Generate a Key` -> `Continue`



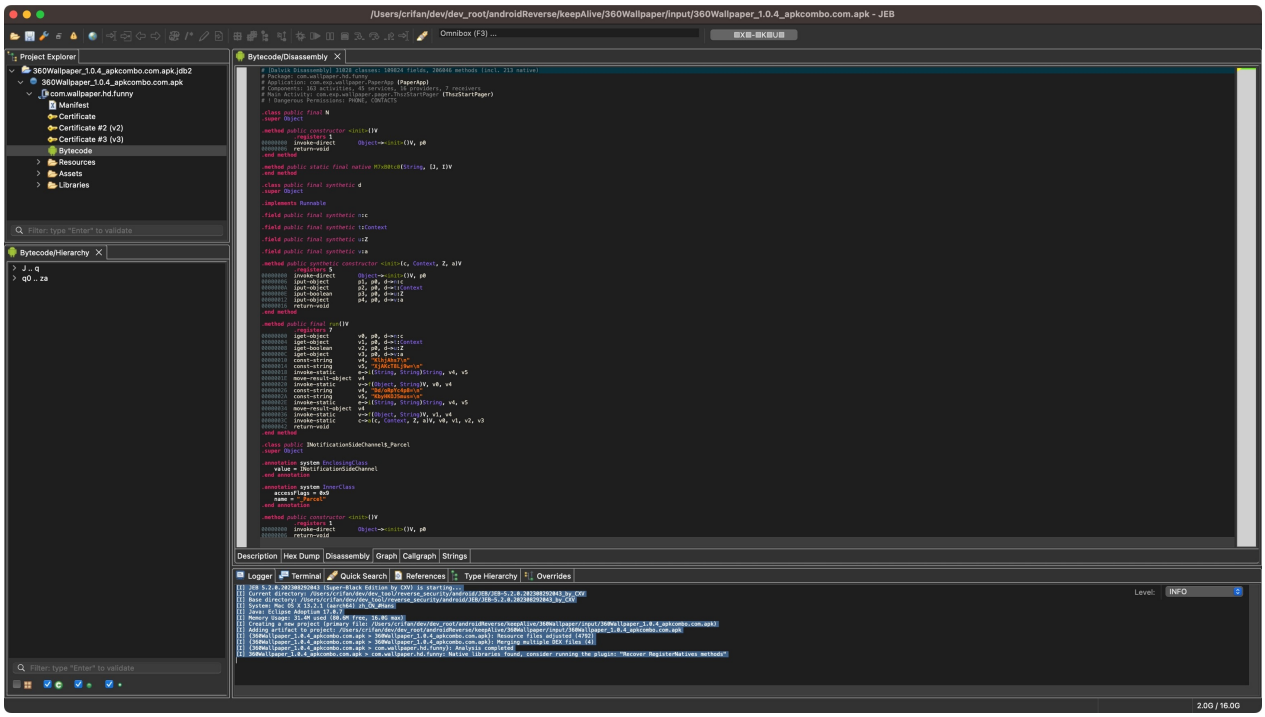
- 主界面



JEB功能和页面

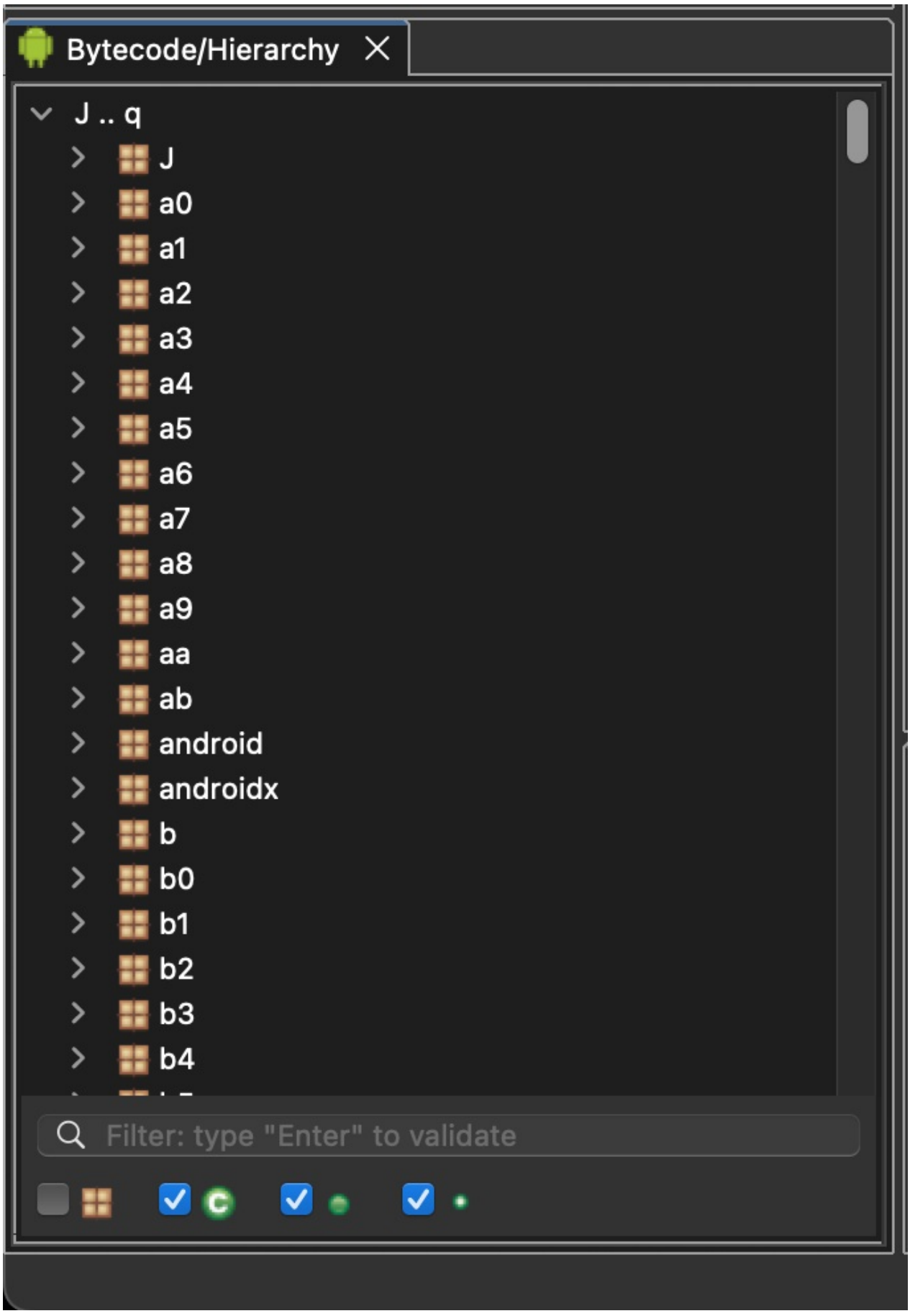
此处介绍，JEB中各个部分的页面显示效果，也就是对应的各个子功能了。

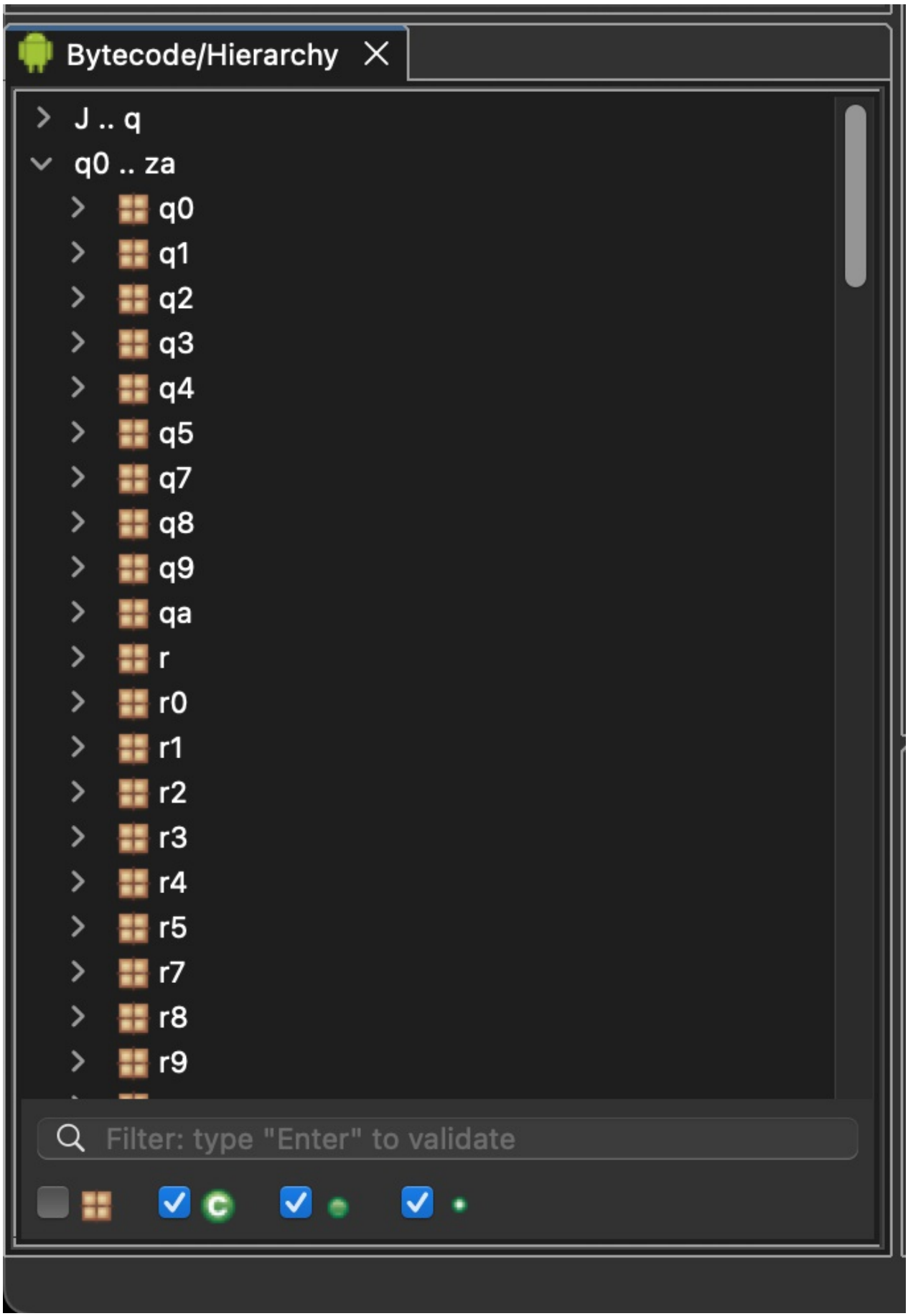
先贴出主要总体界面：



Bytecode/Hierarchy

点击展开类的节点：


Bytecode/Hierarchy
J .. q
J
a0
a1
a2
a3
a4
a5
a6
a7
a8
a9
aa
ab
android
androidx
b
b0
b1
b2
b3
b4
Filter: type "Enter" to validate



单个文件多个显示模式

此处同一个文件 `AndroidMenifest.xml` 的多个不同显示模式的效果：

- Description=描述
    -
- Text=普通文本
    -
- Formatted Text = 格式化后的（带语法高亮的）文本

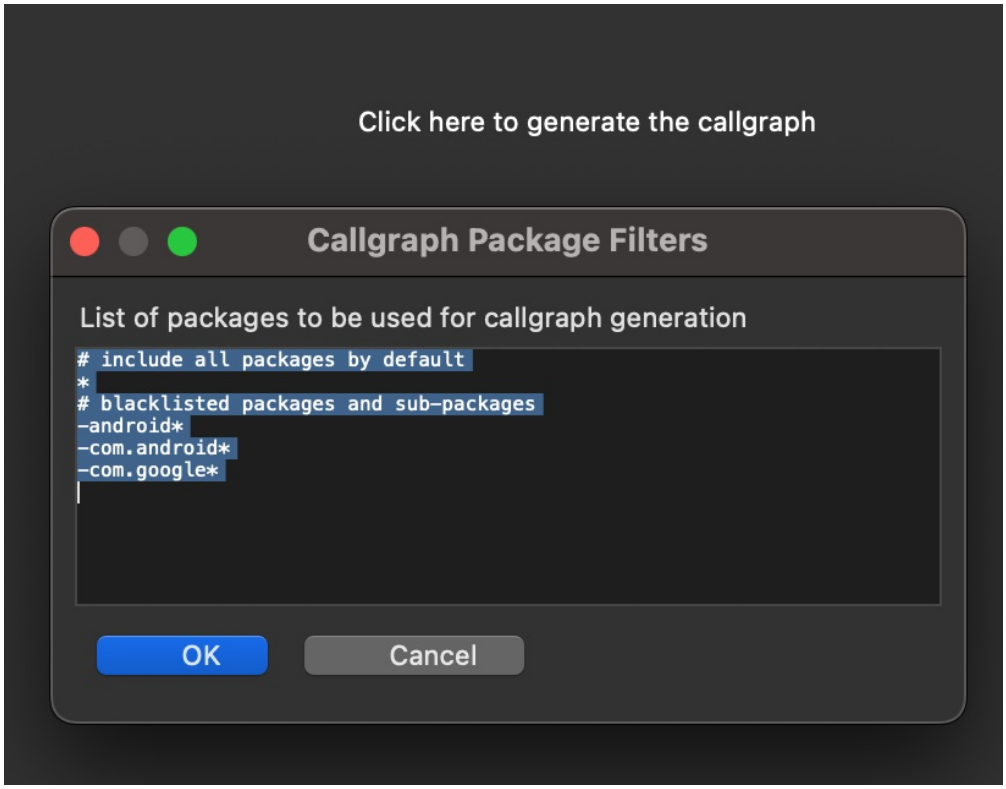
- 。

Bytecode的多个显示模式

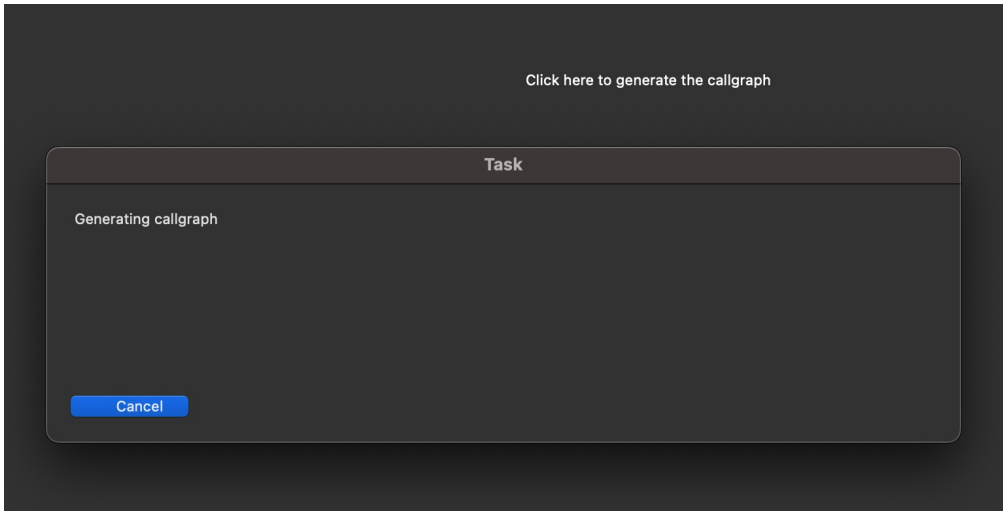
- Description=描述
    -
- Hex Dump
    -
- Disassembly

- Graph

- Callgraph

-
  - 注：点击后，会卡死
    - 但是后来某次点击，暂时没卡死，出现弹框
    - 

    - 弹框提示：开始生成callgraph



    - 但是耗时太久，就放弃了
        - TODO：抽空再试试效果
- Strings
    -

顺带贴出Bytecode的 `Disassembly` 的部分代码：

```
# [Dalvik Disassembly] 31028 classes, 109824 fields, 206046 methods (incl. 213 native)
# Package: com.wallpaper.hd.funny
# Application:                    PaperApp (PaperApp)
# Components: 163 activities, 45 services, 16 providers, 7 receivers
# Main Activity:                          ThszStartPager (ThszStartPager)
# 1 Dangerous Permissions: PHONE, CONTACTS
.class public final N
.super Object
.method public constructor <init>()V
.registers 1
00000000 invoke direct Object-><init>()V, p0
00000006 return void
.end method
.method public static final native M7xB0tc0(String, [J, I)V
.end method
```

```
.class public final synthetic d
.super Object
.implements Runnable
.field public final synthetic n c
.field public final synthetic t Context
.field public final synthetic u Z
.field public final synthetic v a
.method public synthetic constructor <init>(c, Context, Z, a)V
.registers 5
00000000 invoke direct Object-><init>()V, p0
00000006 iput object p1, p0, d->n c
0000000A iput object p2, p0, d->t Context
0000000E iput boolean p3, p0, d->u Z
00000012 iput object p4, p0, d->v a
00000016 return void
.end method
.method public final run()V
.registers 7
00000000 iget object v0, p0, d->n c
00000004 iget object v1, p0, d->t Context
00000008 iget boolean v2, p0, d->u Z
0000000C iget object v3, p0, d->v a
00000010 const string v4, "KlhjAhs7\n"
00000014 const string v5, "XjAKcT8Lj9w=\n"
00000018 invoke static e->i(String, String)String, v4, v5
0000001E move result object v4
00000020 invoke static v->f(Object, String)V, v0, v4
00000026 const string v4, "Dd/oRpYc4p8=\n"
0000002A const string v5, "KbyHKOJ5mus=\n"
0000002E invoke static e->i(String, String)String, v4, v5
00000034 move result object v4
00000036 invoke static v->f(Object, String)V, v1, v4
0000003C invoke static c->a(c, Context, Z, a)V, v0, v1, v2, v3
00000042 return void
.end method
.class public INotificationSideChannel$_Parcel
.super Object
.annotation system EnclosingClass
value = INotificationSideChannel
.end annotation
.annotation system InnerClass
accessFlags = 0x9
name = "_Parcel"
.end annotation
.method public constructor <init>()V
.registers 1
00000000 invoke direct Object-><init>()V, p0
00000006 return void
.end method
.method public static synthetic access$000(Parcel, Parcelable$Creator)Object
.registers 2
00000000 invoke static INotificationSideChannel$_Parcel->readTypedObject(Parcel, Parcel
able$Creator)Object, p0, p1

...
...
```

```
'''

.class public abstract interface INotificationSideChannel
.super Object
.implements IInterface
.annotation system MemberClasses
value = {
INotificationSideChannel$_Parcel,
INotificationSideChannel$Stub,
INotificationSideChannel$Default
}
.end annotation
.field public static final DESCRIPTOR:String
.method static constructor <clinit>()V
.registers 3
00000000 const string v0, "android$support$v4$app$INotificationSideChannel"
00000004 const/16 v1, 36
00000008 const/16 v2, 46
0000000C invoke virtual String->replace(C, C)String, v0, v1, v2
00000012 move result object v0
00000014 sput object v0, INotificationSideChannel->DESCRIPTOR:String
00000018 return void
.end method
.method public abstract cancel(String, I, String)V
.annotation system Throws
value = {
RemoteException
}
.end annotation
.end method
.method public abstract cancelAll(String)V
.annotation system Throws
value = {
RemoteException
}
.end annotation
.end method
.method public abstract notify(String, I, String, Notification)V
.annotation system Throws
value = {
RemoteException
}
.end annotation
.end method
.class MediaBrowserCompat$CallbackHandler
.super Handler
.annotation system EnclosingClass
value = MediaBrowserCompat
.end annotation
.annotation system InnerClass
accessFlags = 0x9
name = "CallbackHandler"
.end annotation
.field private final mCallbackImplRef:WeakReference
.annotation system Signature
value = {
```

```
"Ljava/lang/ref/WeakReference<",
"Landroid/support/v4/media/MediaBrowserCompat$MediaBrowserServiceCallbackImpl;",
">;"
}
.end annotation
.end field
```

JEB底部多个tab页

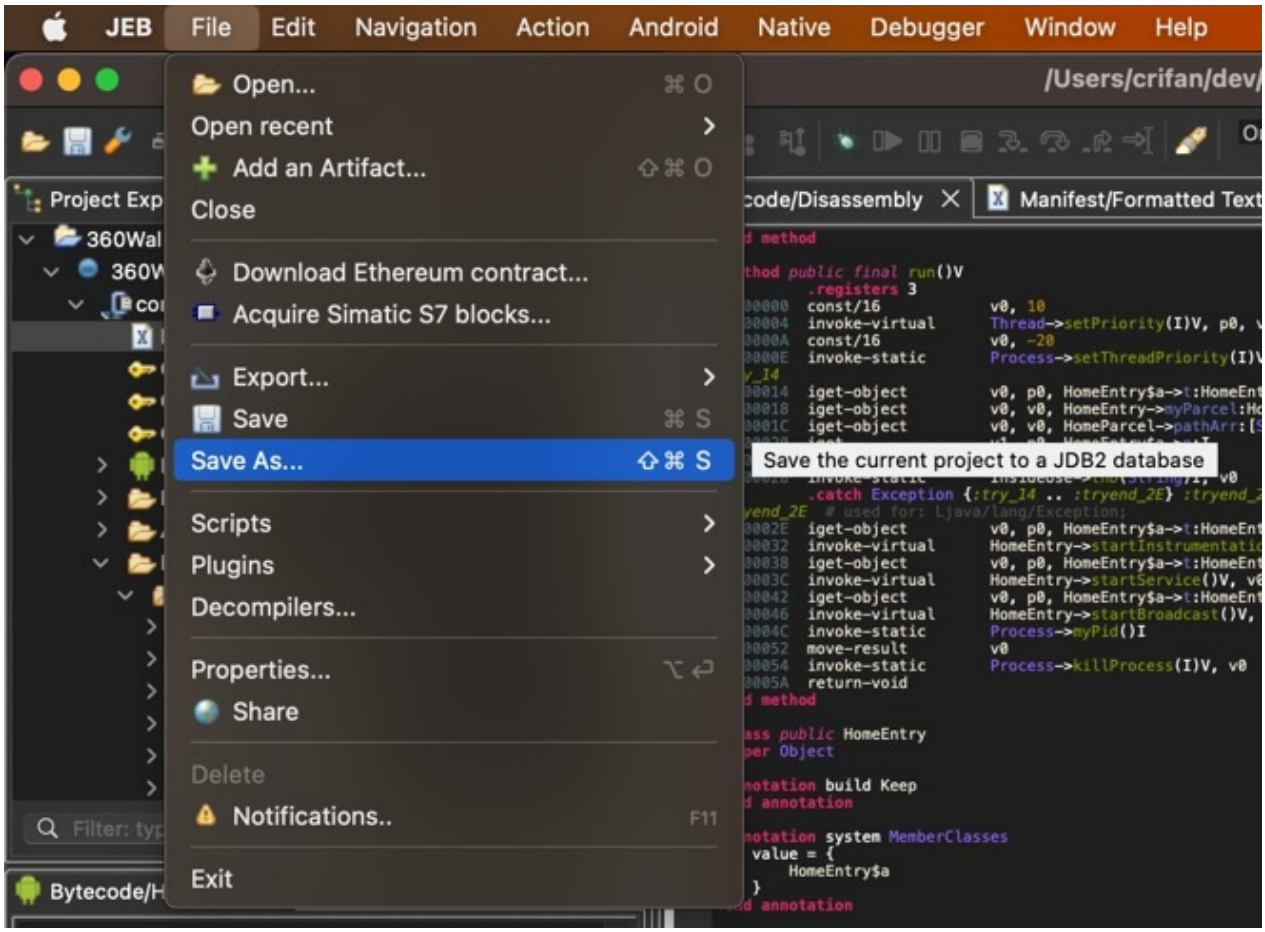
- Logger
    -
- Terminal
    -
- Quick Search
    -
- References
    -
- Type Hierarchy

- Overrides

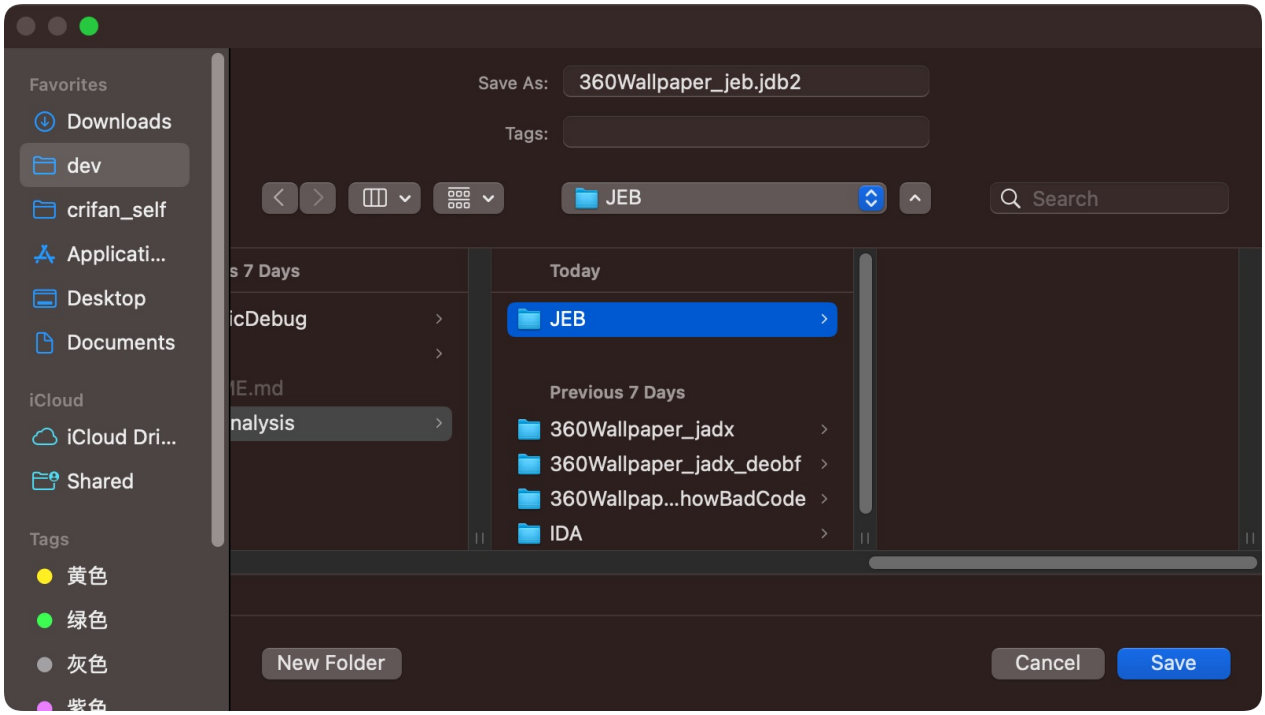
保存项目文件jdb2

当需要时，可以去保存当前项目：

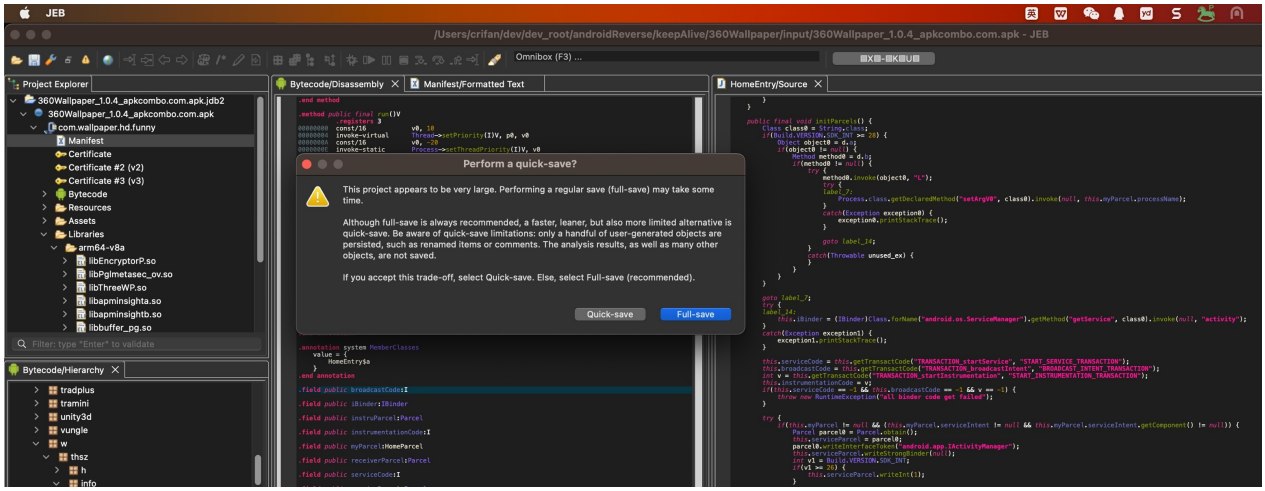
`JEB -> File -> Save As`，会提示 `Save the current projec to JDB2 database`



给文件起个名字：



首次保存时，会提示：

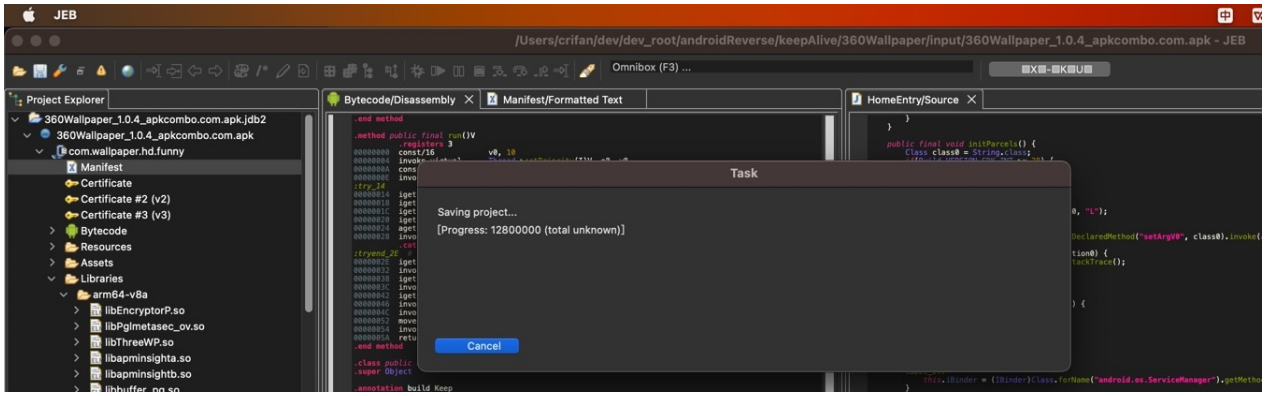


项目看起来很大，保存时选用哪种模式：

- regular save=普通保存=full-save=全部保存
  - 缺点：耗时
  - 优点：所有改动都会保存
- quick save=快速保存=轻量级保存
  - 缺点：只保存一些用户生成的对象，比如重命名、注释等
    - 但是反编译等解析结果不保存
      - 注：这些解析结果，可以每次打开时，重新解析
  - 优点：速度快

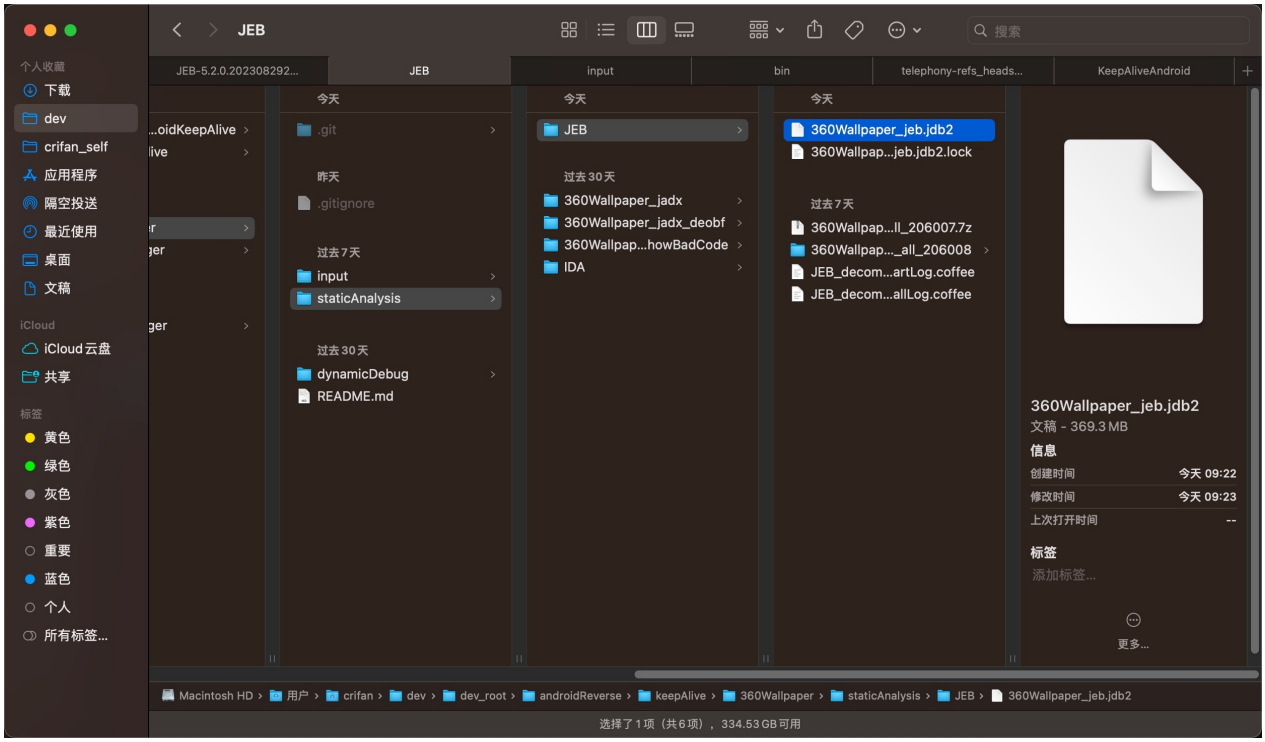
此处为了确保改动都保存，选择了：`full-save` = `全部保存`

然后会有弹框显示保存进度：



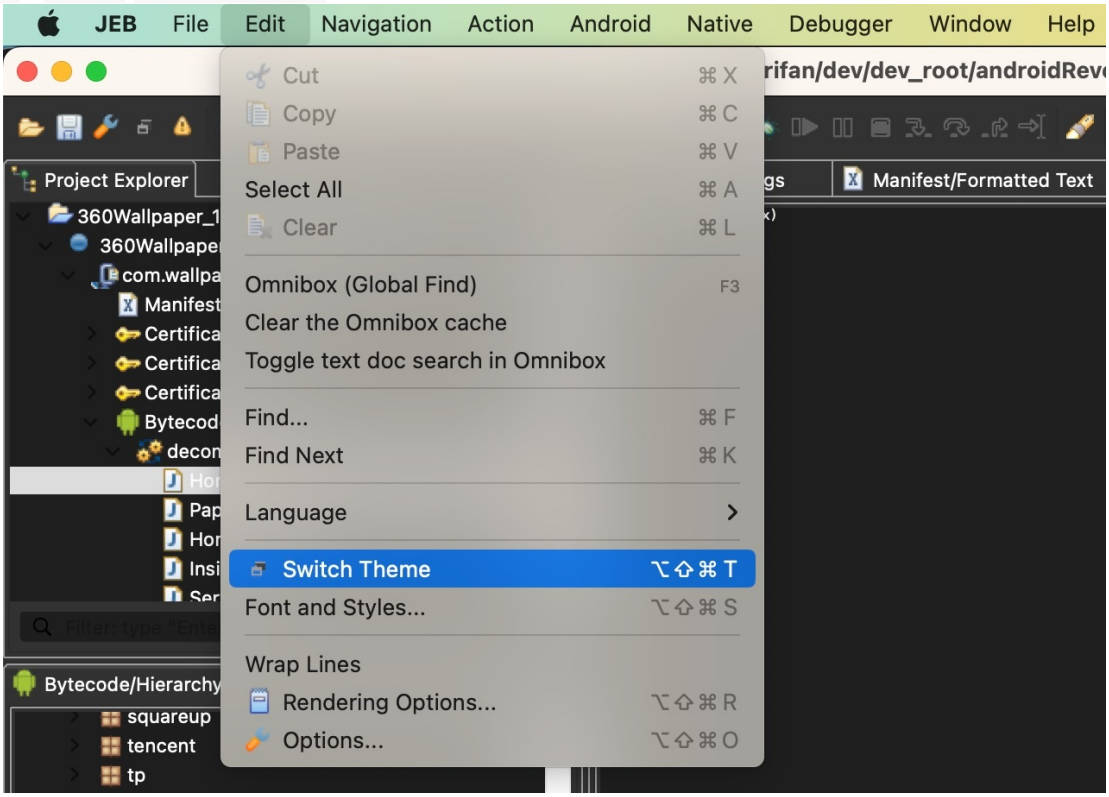
此处保存耗时大概几十秒，总体上还算可以接受。

保存出的`.jdb2`文件：

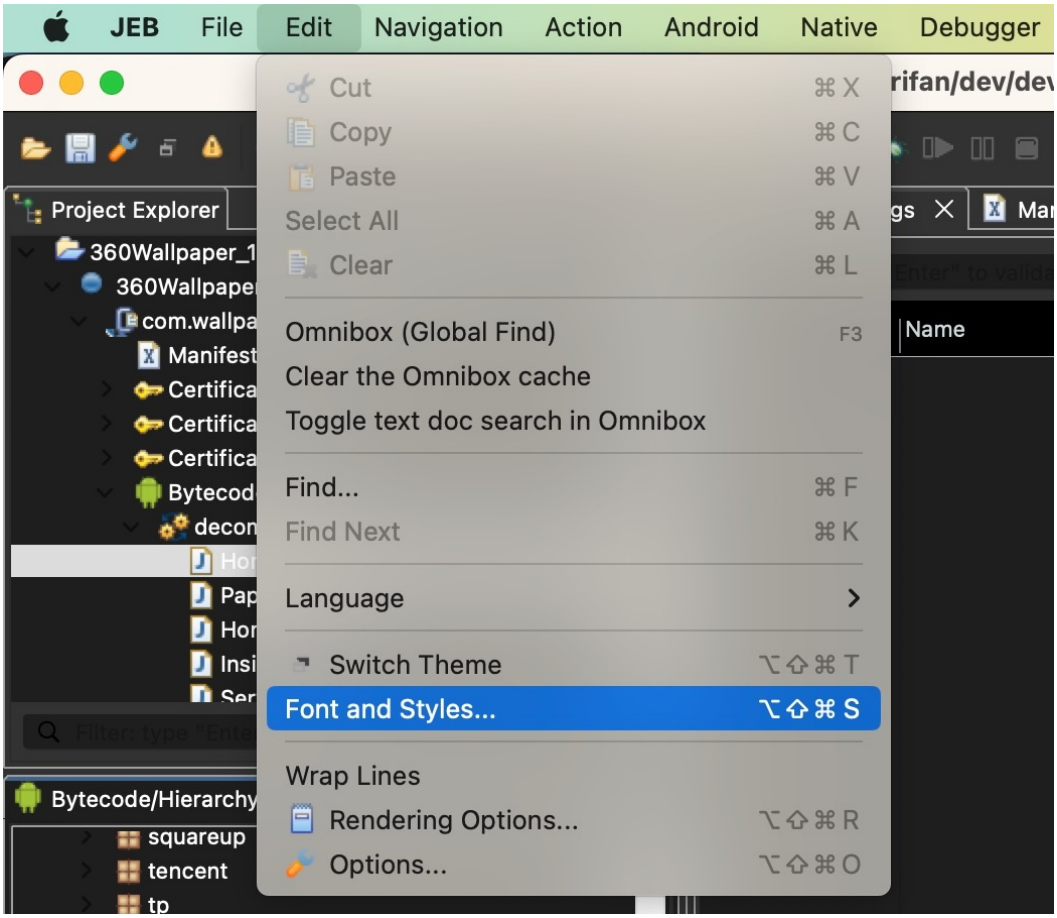


JEB多主题显示效果

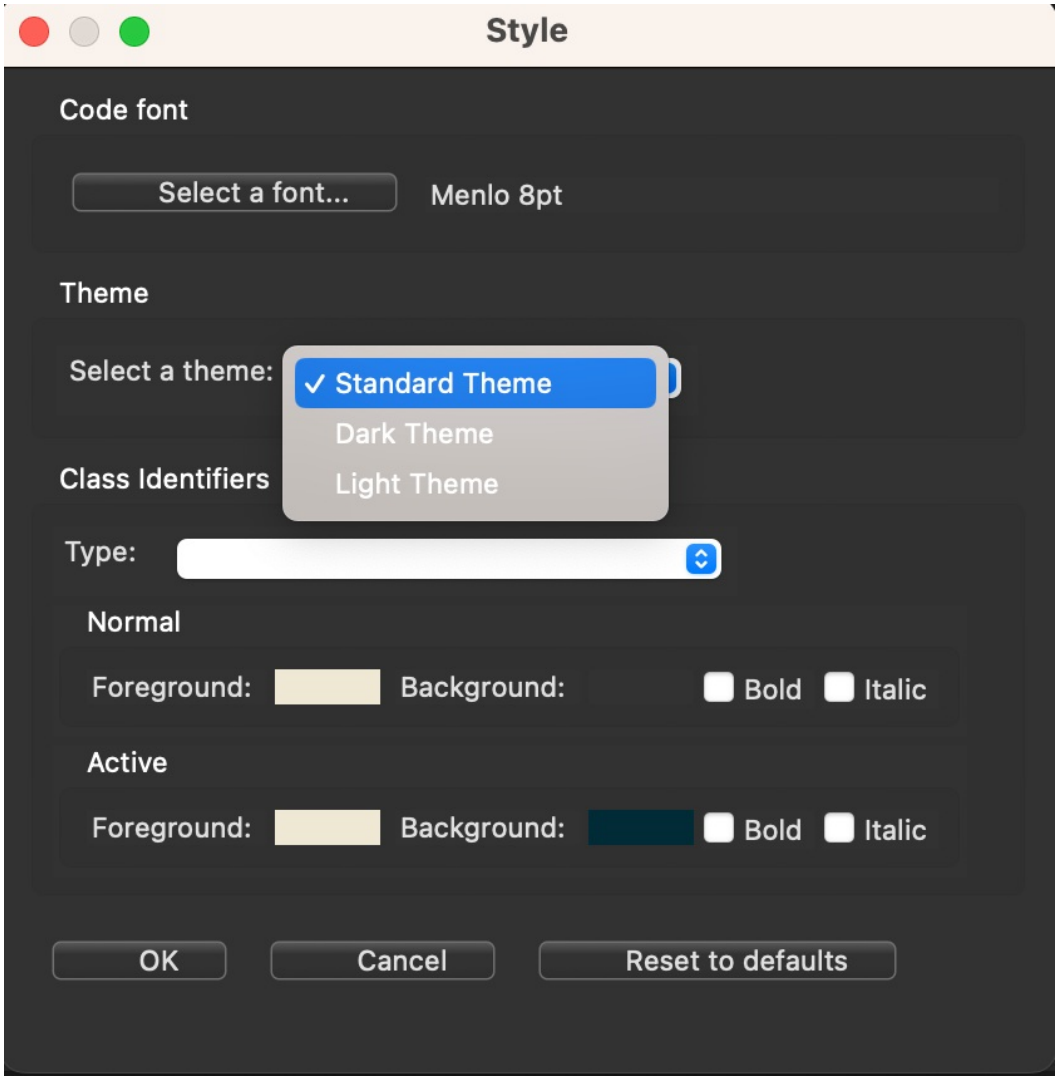
JEB中显示方面，支持多个主题：

- 如何切换主题?
  - `JEB -> Edit - Switch Theme`
  - 

- 不同主题的名称
  - 最开始以为：JEB中无法查看当前主题的名称
    - 只能看到主题效果
  - 后记：后来找到了主题设置，才看到主题名称（吐槽：JEB的设置很分散，很混乱）
    - `JEB -> Edit -> Font and styles`

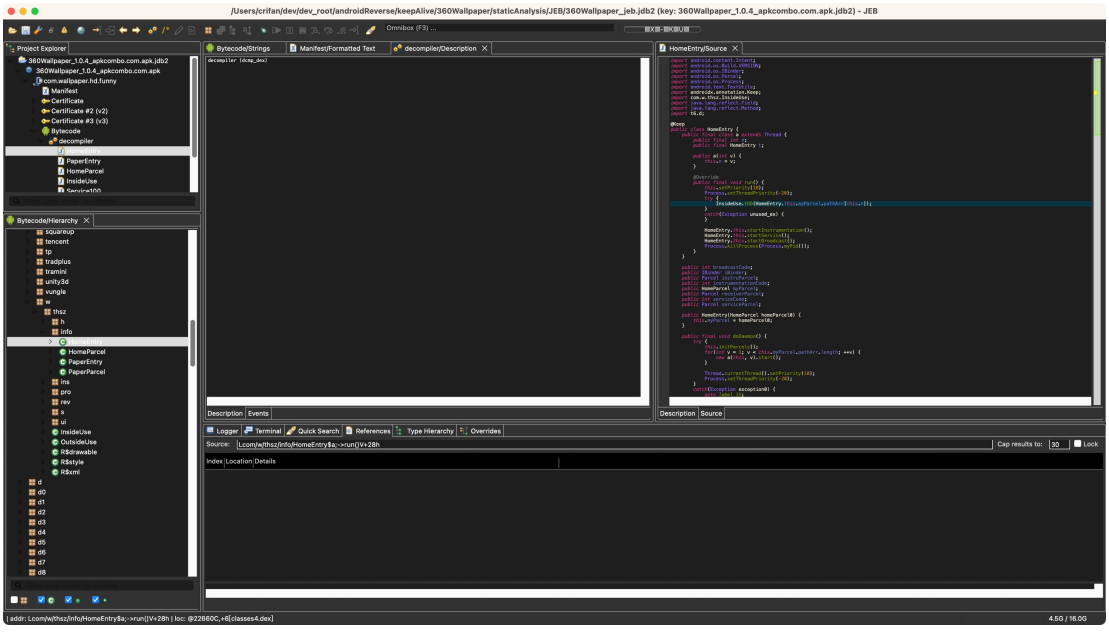
- 

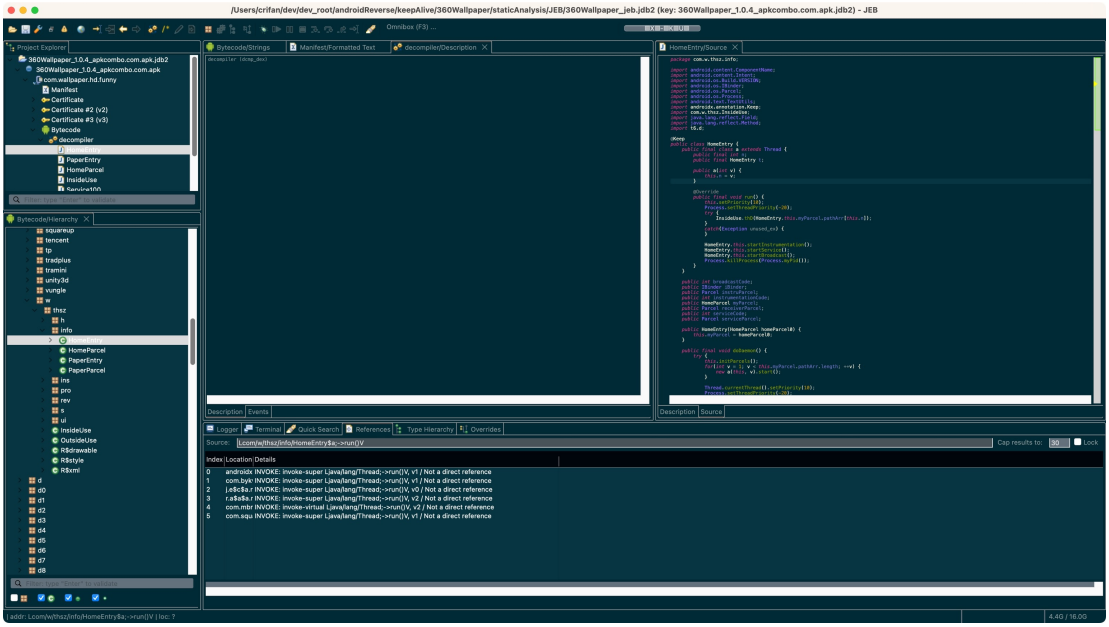
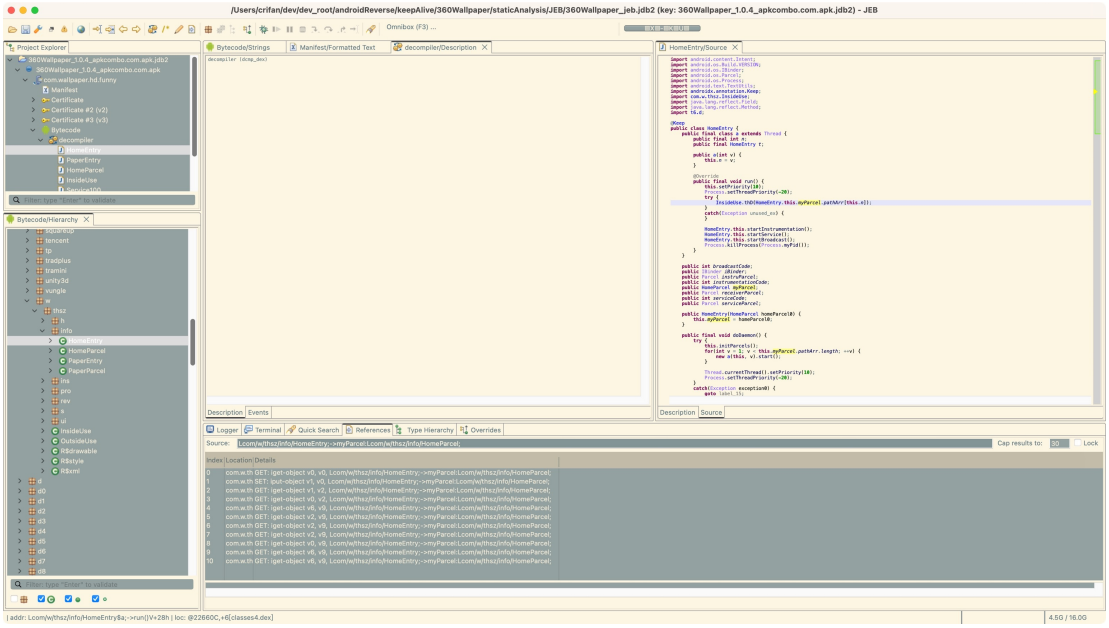
- 然后可以看到主题名称，可以切换主题



- Standard Theme == 黑色
- Dark Theme == 淡蓝色
- Light Theme == 淡黄色

- 多个不同主题的显示效果
  - Standard Theme == 黑色



- Dark Theme == 淡蓝色
  - 
- Light Theme == 淡黄色
  - 

反编译器

- `JEB` -> `File` -> `Decompilers`
  -

- -> 可以看到当前JEB版本，所支持的各种架构的反编译器
  - dex (`DEX` Decompiler)
  - x86 (`X86` Decompiler)
  - x86_64 (`X86-64` Decompiler)
  - arm (`ARM` Decompiler)
  - arm64 (`Aarch64` Decompiler)
  - mips (`MIPS` Decompiler)
  - riscv (`RISC-V` Decompiler)
  - wasmbc (`WebAssembly` Decompiler)
  - evmbc (`EVM` Decompiler)
  - simatic_mc7 (`S7` Decompiler)
  - avr (`AVR` Decompiler)
  - diemvm_bc (`Diem` Decompiler)

脚本

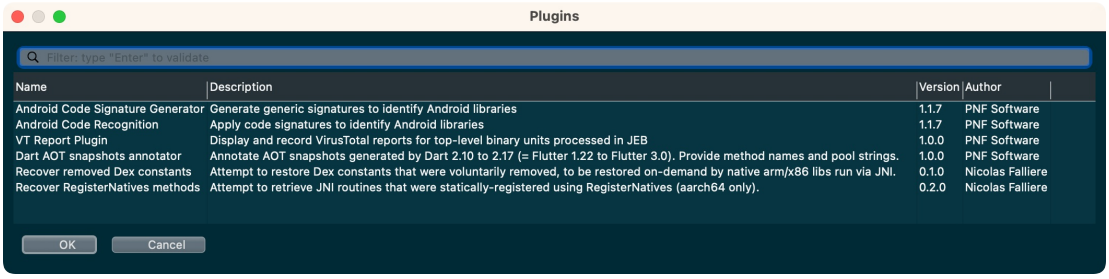
JEB支持脚本。内置就有很多Scripts脚本：

- `JEB` -> `File` -> `Scripts`

-
  - 鼠标悬停到某个脚本时，还能显示脚本功能简介

插件

JEB支持插件机制。内置就有一些插件：

- `JEB -> File -> Plugins -> Execute and Engins Plugin`
  -
    - 可以看到内置引擎插件
      - Android Code Signature Generator
      - Android Code Recogintion
      - VT Report Plugin
      - Dart AOT snapthots annotator
      - Recover removed Dex constants
      - Recover RegisterNatives methods
  - 如果点击：`JEB -> File -> Plugins -> Plugins`，会出现弹框，显示当前插件列表，显示插件详情：名称、功能描述、版本、作者等
    - 

JEB静态反编译

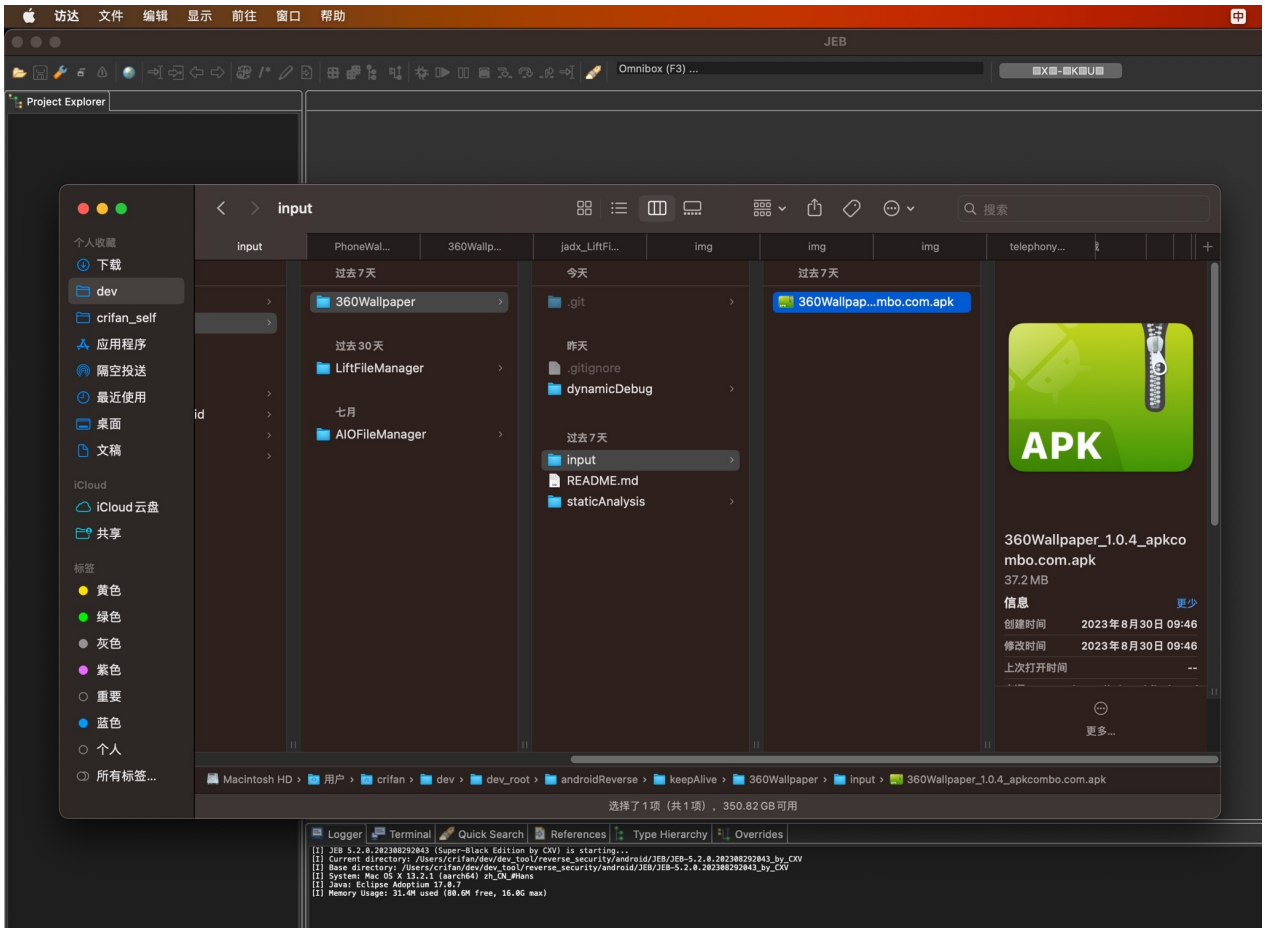
用JEB反编译安卓apk

直接把apk拖动到JEB中，即可自动开始反编译。

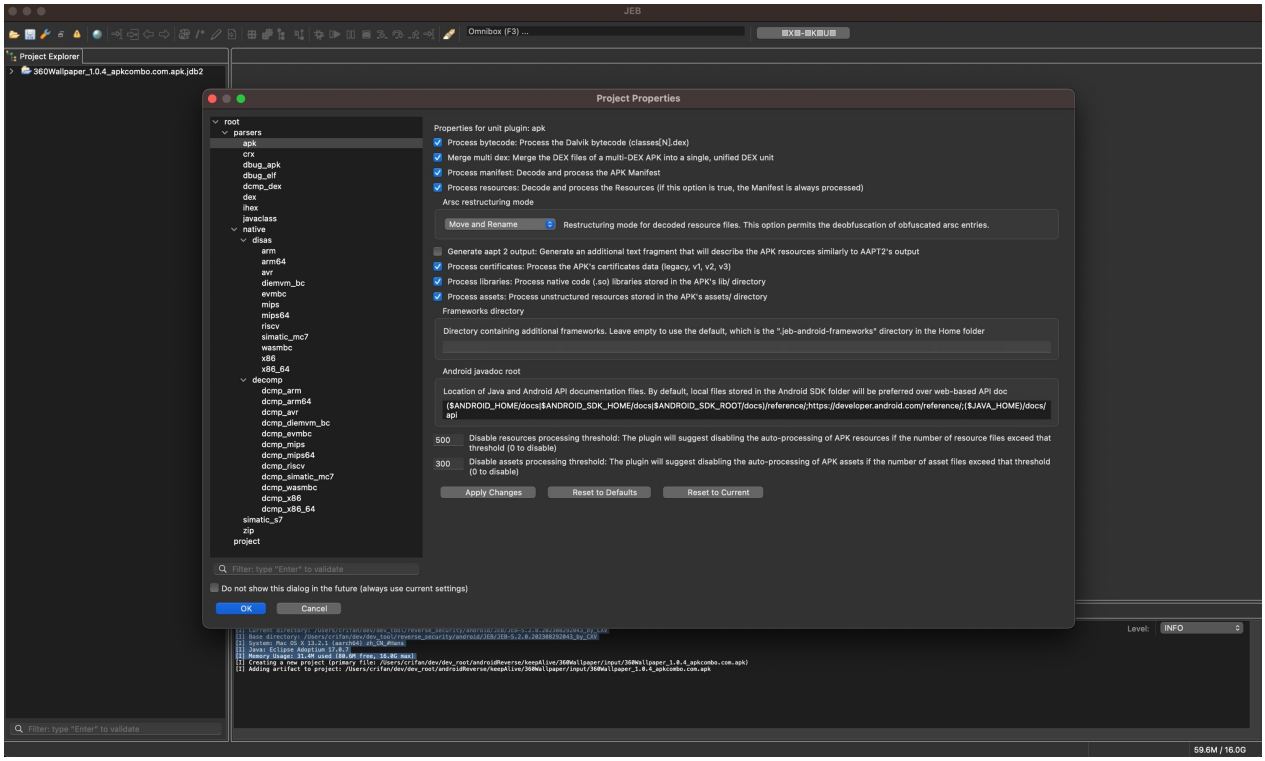
举例

用JEB反编译360Wallpaper的apk

直接把 `360Wallpaper` 的 `apk` 文件，直接拖动到JEB中：

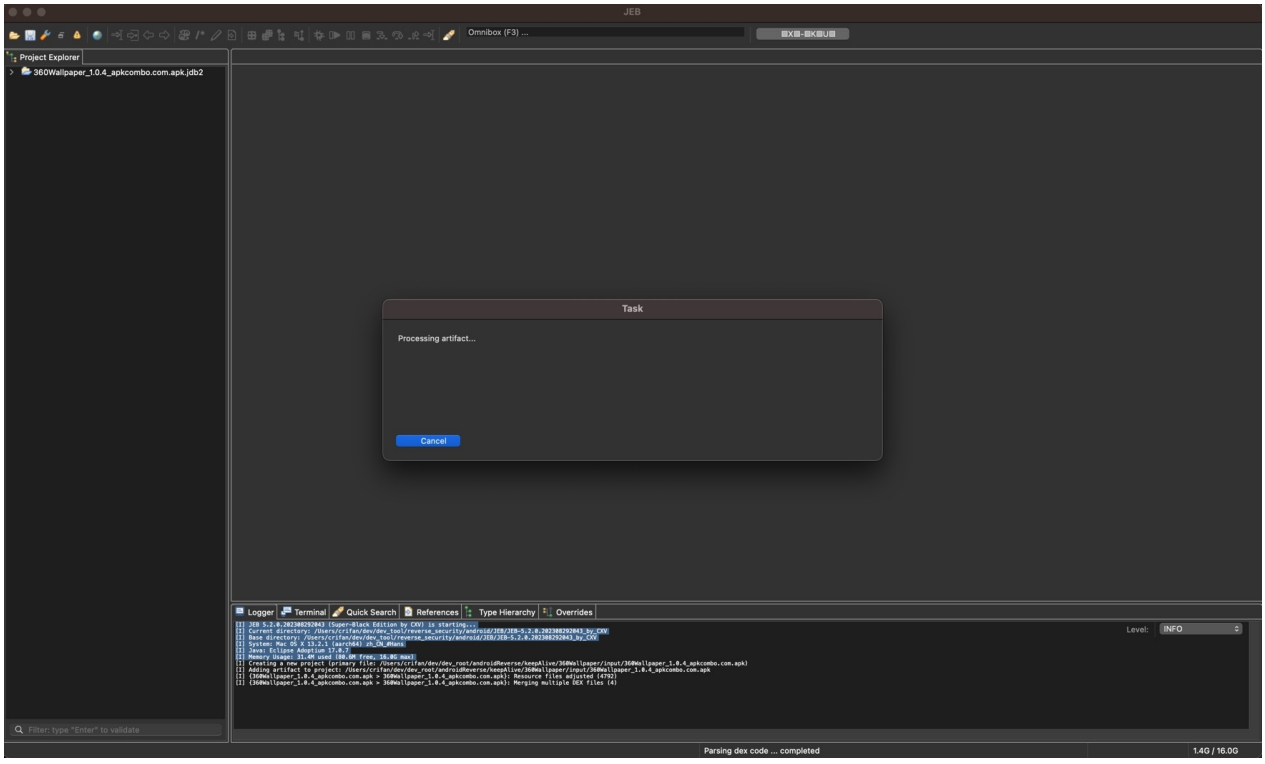


会弹框 `Project Property` ：

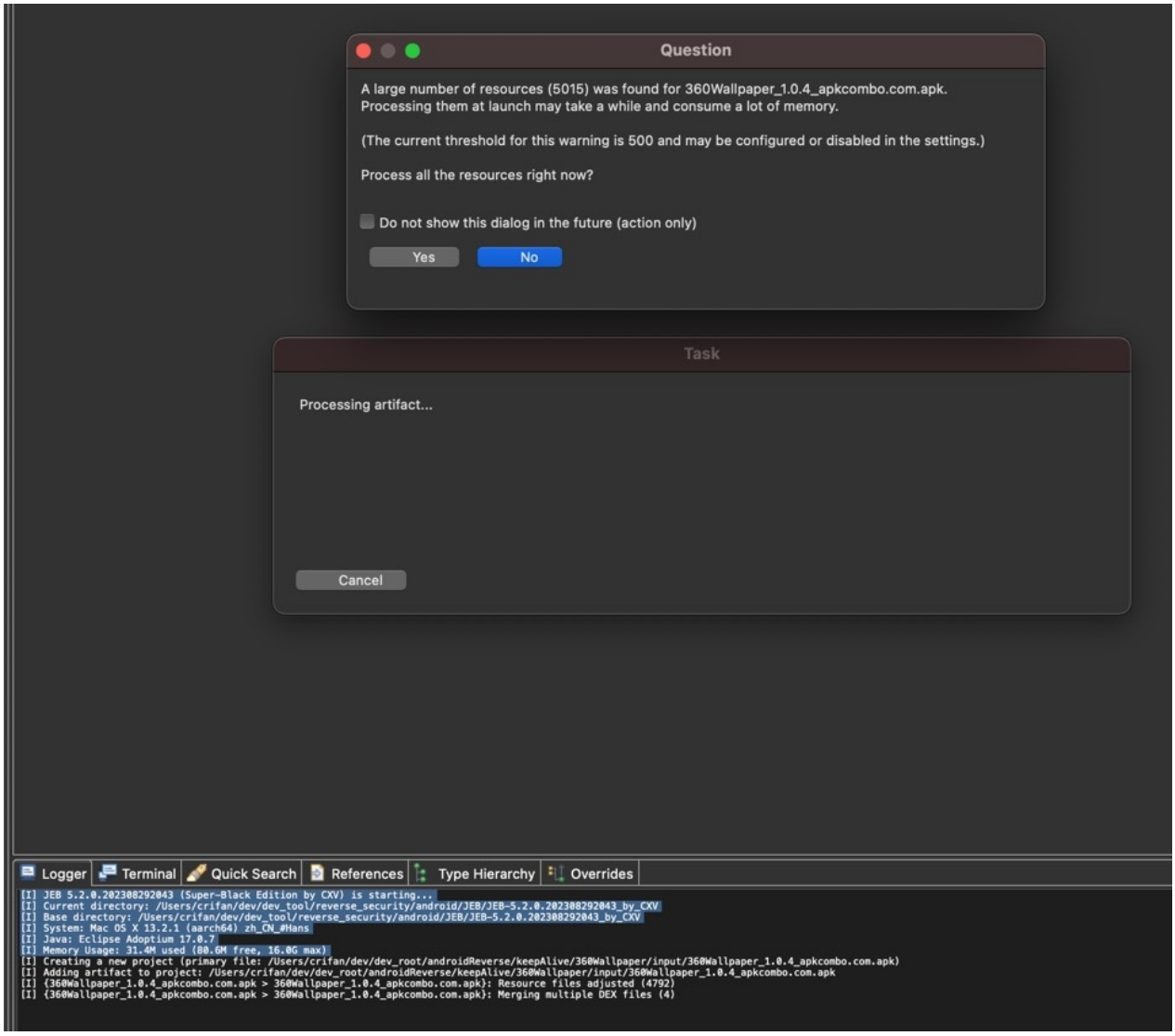


此处可以自动识别输入文件类型是：`apk`，所以自动选择了`apk`的配置，保持默认配置即可，点击：`OK`，继续反编译：

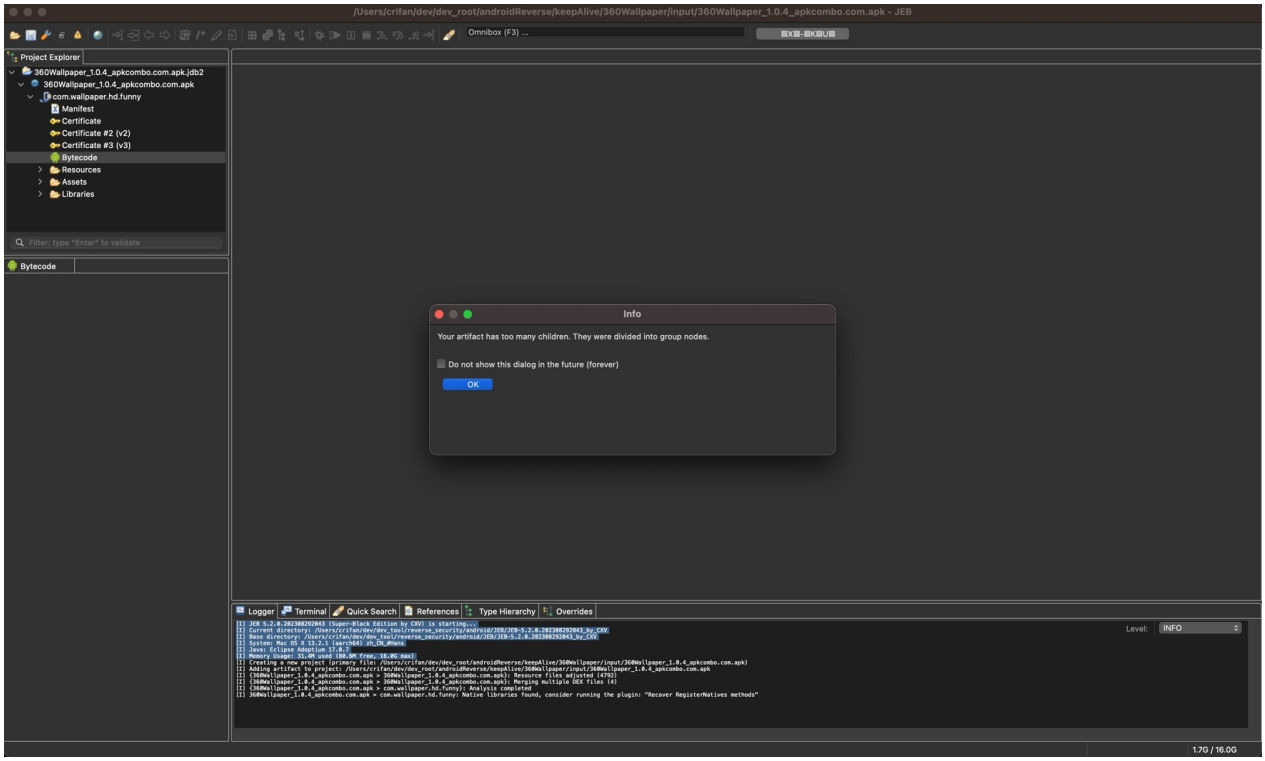
然后会弹框`Task`，开始处理：



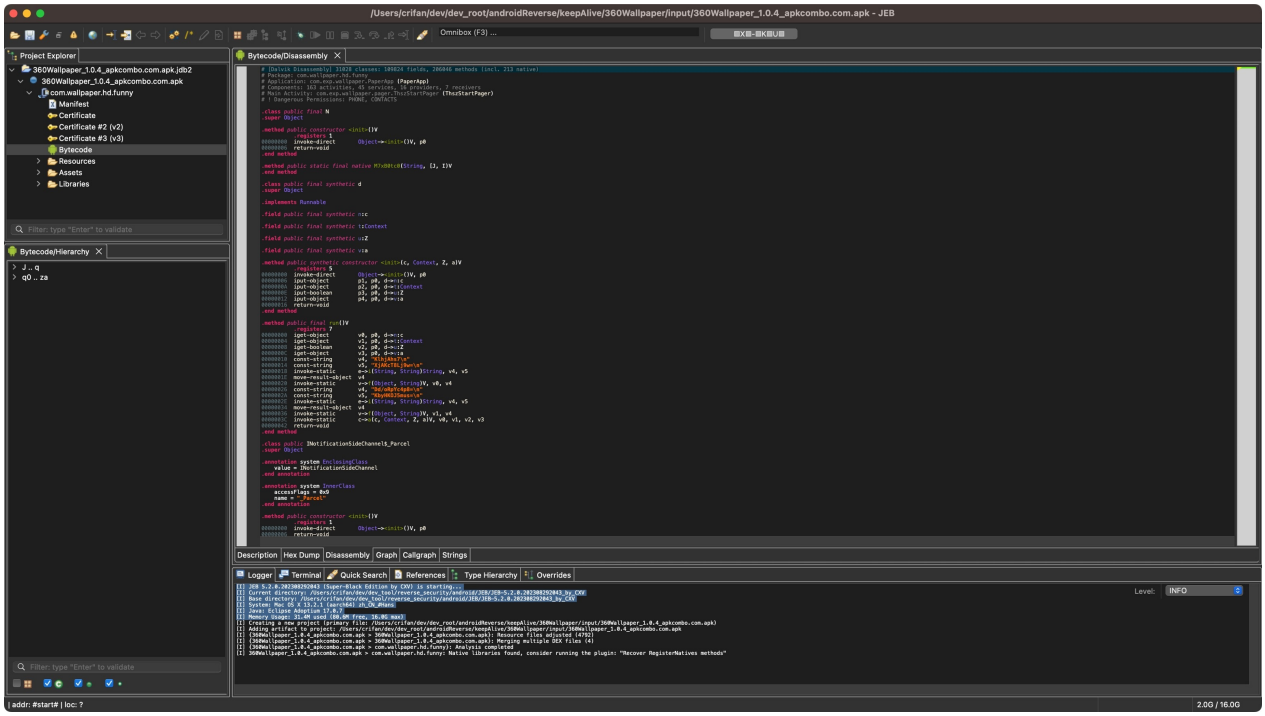
此处由于类太多，所以会提示，问是否继续，点击`Yes`：



后续，由于反编译出来的类太多以及类有很多子类和属性children，所以会提示会按组分类显示，点击 `OK`：



然后就可以看到，反编译后的主界面了：



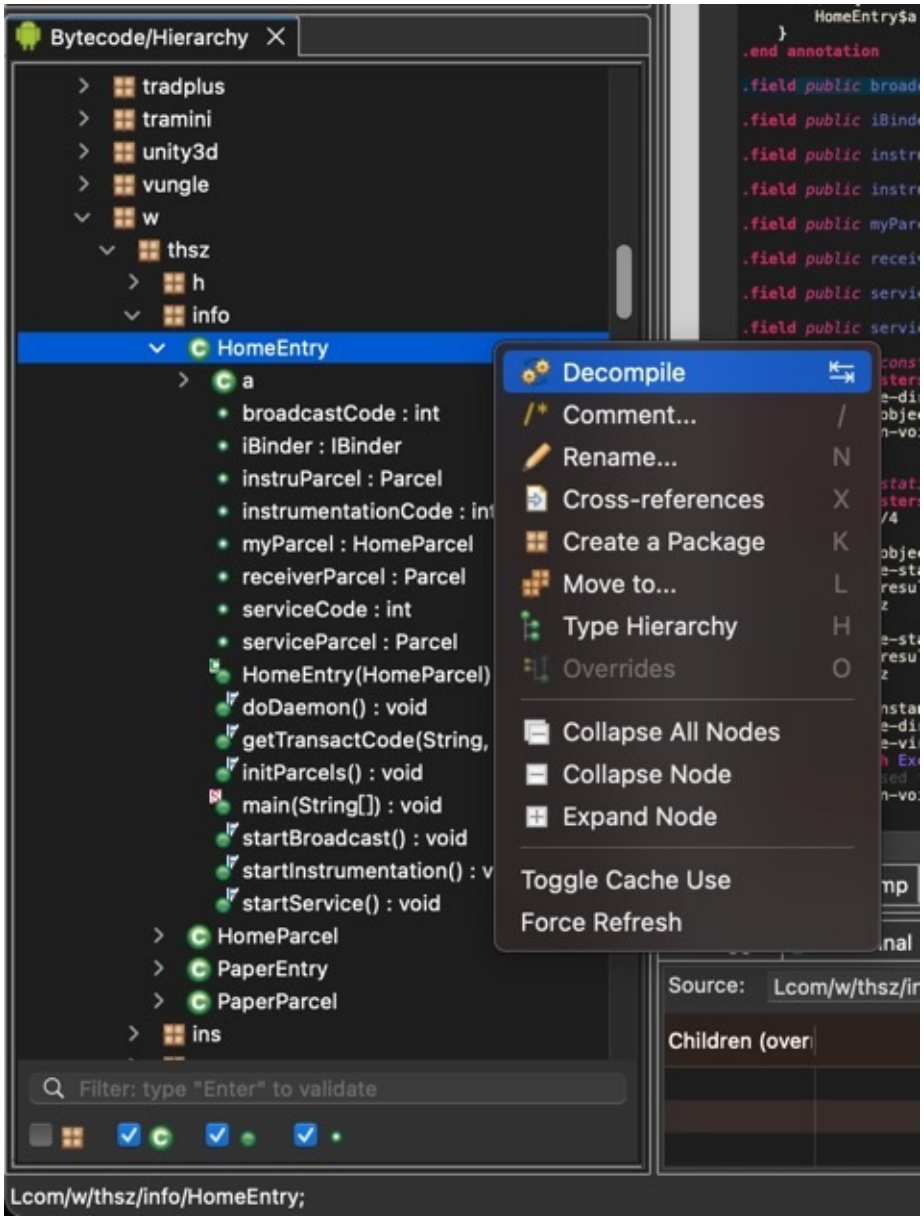
如此，反编译就结束了。

反编译出java

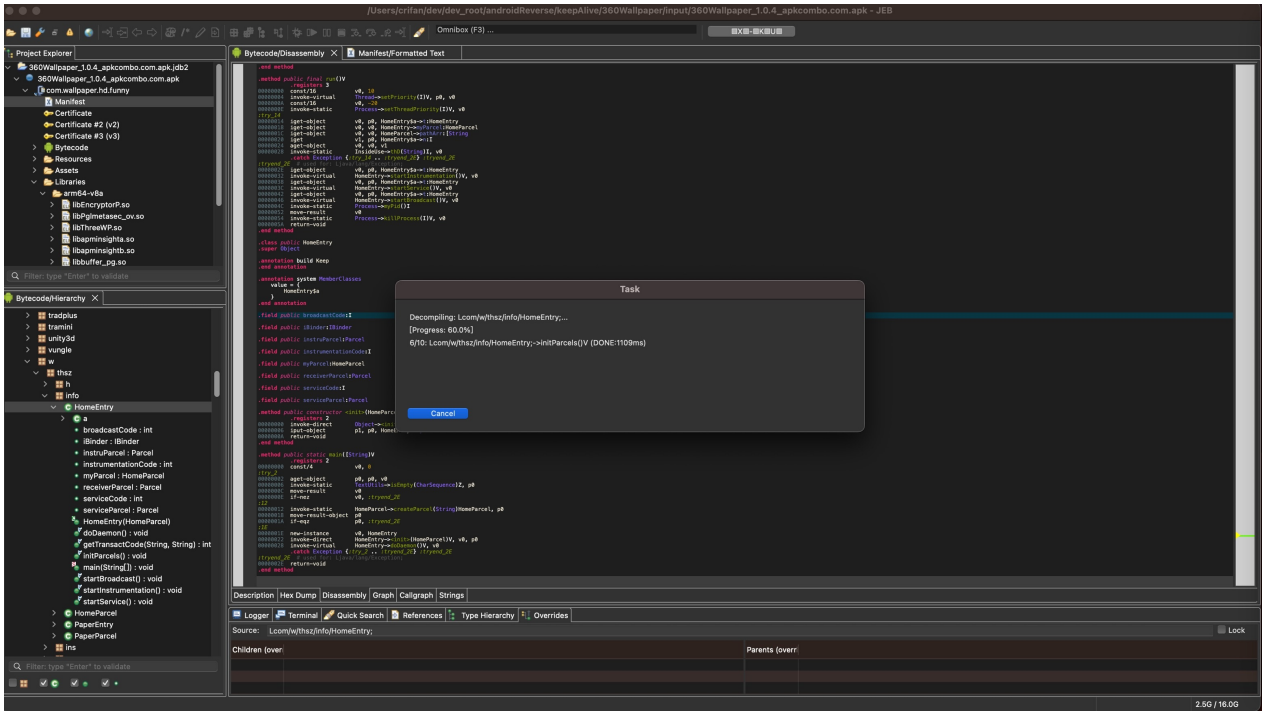
单个反编译

- 背景：JEB默认情况下去反编译apk的话，显示的是Bytecode字节码：
  -
  - 而不是我们要的 `Java` 代码
- 需求：对于单个类去反编译出java代码

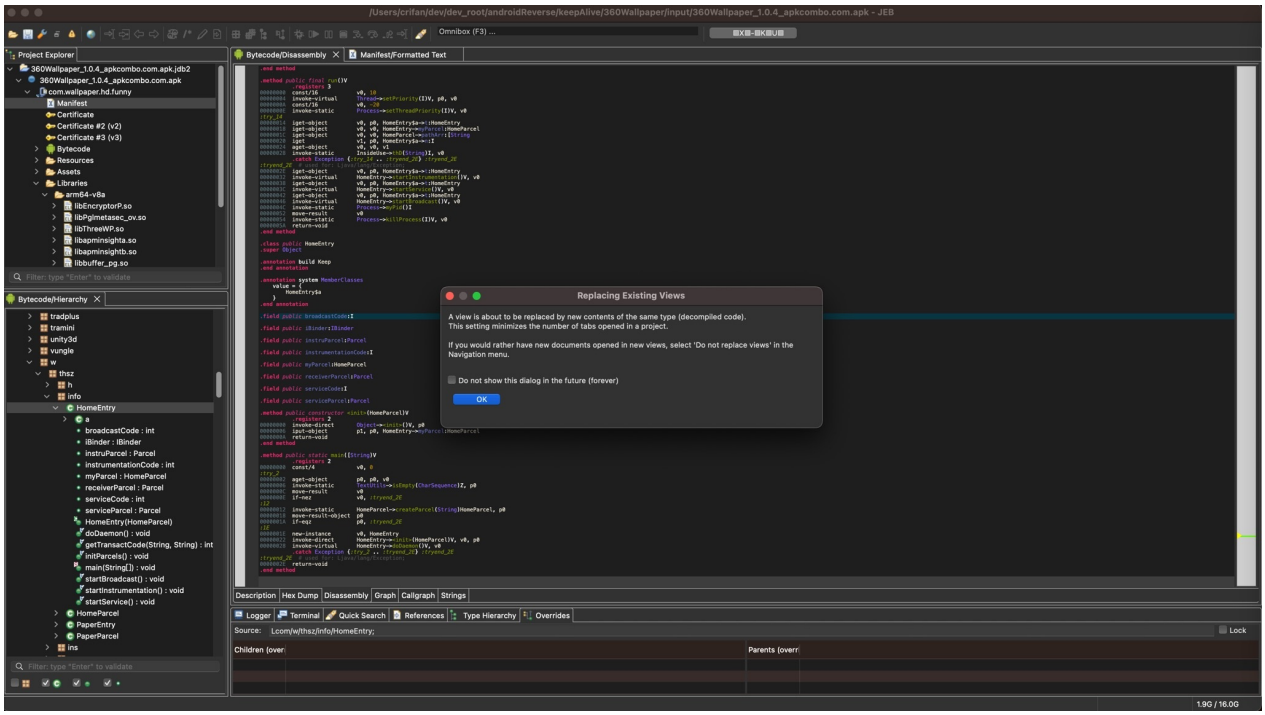
从左下角窗口中，展开找到自己要反编译的java的类->右键-> `Decompile` ：


Bytecode/Hierarchy
tradplus
tramini
unity3d
vungle
w
thsz
h
info
HomeEntry
a
broadcastCode : int
iBinder : IBinder
instruParcel : Parcel
myParcel : HomeParcel
receiverParcel : Parcel
serviceCode : int
serviceParcel : Parcel
HomeEntry(HomeParcel)
doDaemon() : void
initParcels() : void
main(String[]) : void
startBroadcast() : void
startService() : void
HomeParcel
PaperEntry
PaperParcel
ins
Filter: type "Enter" to validate
Decompile
Comment...
Rename...
Cross-references
Create a Package
Move to...
Type Hierarchy
Overrides
Collapse All Nodes
Collapse Node
Expand Node
Toggle Cache Use
Force Refresh
Source:
Children (over
Lcom/w/thsz/info/HomeEntry;

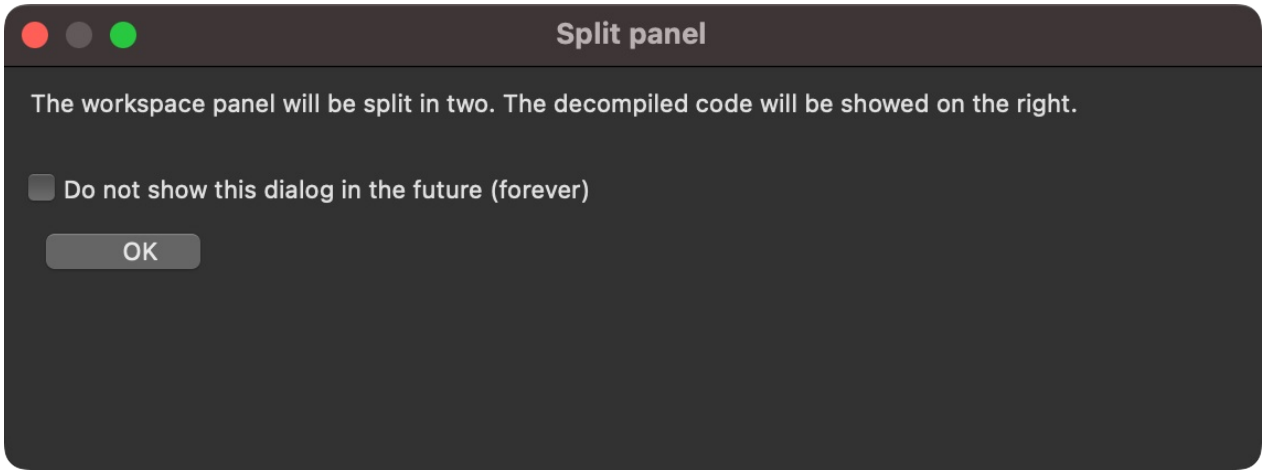
会弹框开始反编译：



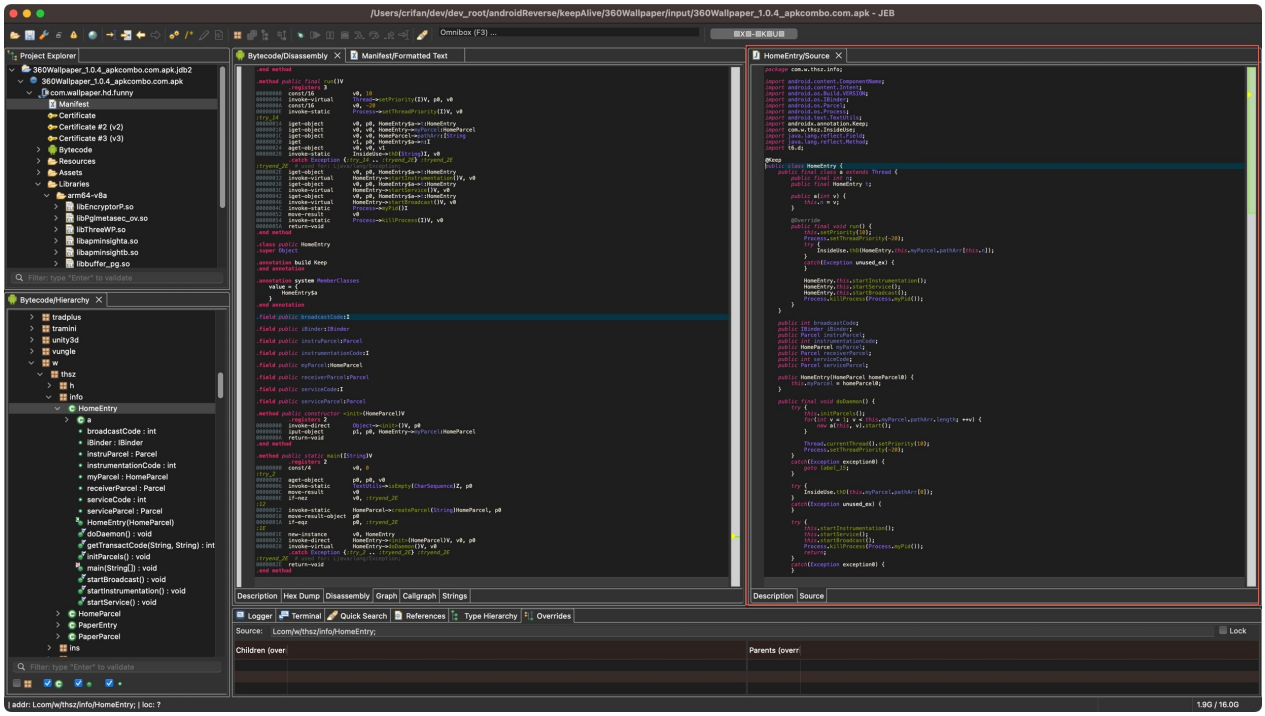
首次会有（勾选不再提示后，之后就不会再显示）额外的弹框提示，新的反编译的内容要显示在新的视图，但要替代当前视图：



以及还会提示，显示窗口要分左右两个子窗口，右边用于显示刚反编译出来的java源代码：



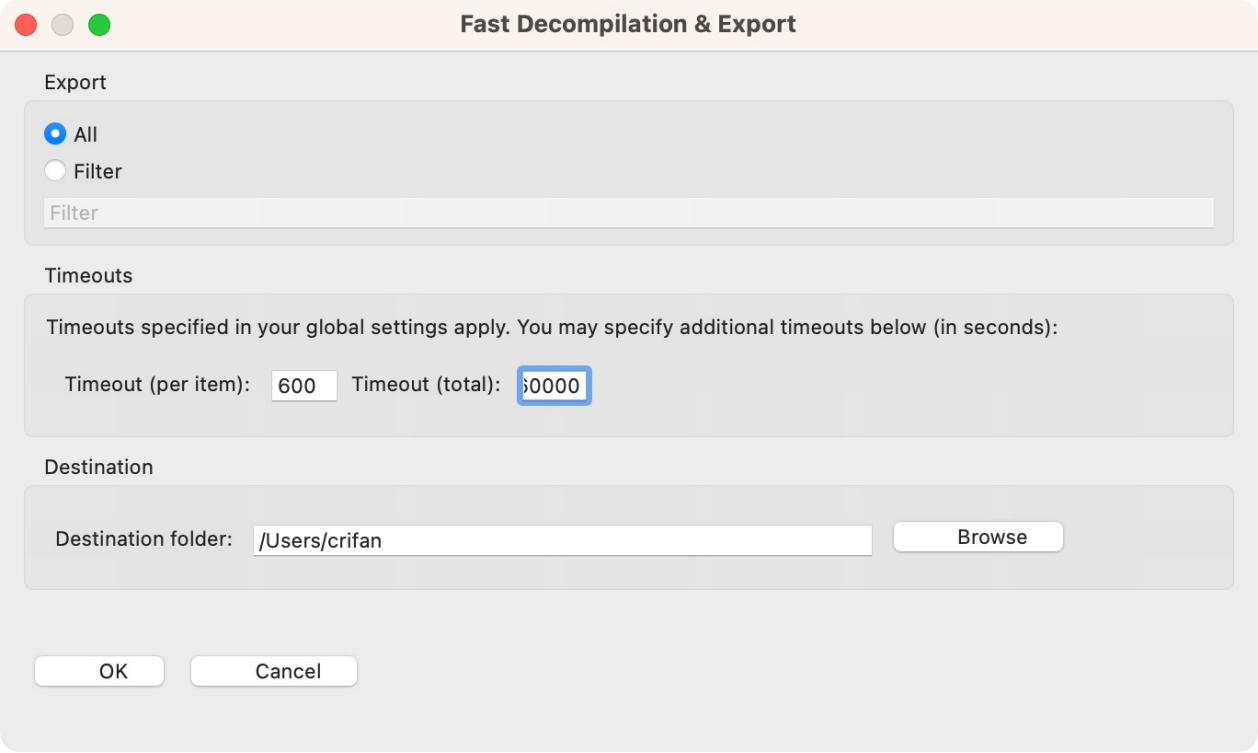
然后就可以正常显示反编译后的java源代码了：



全部反编译

- 需求：JEB中想要（批量）反编译出全部的Java代码
- 步骤：`JEB` -> `File` -> `Export` -> `Decompilation`

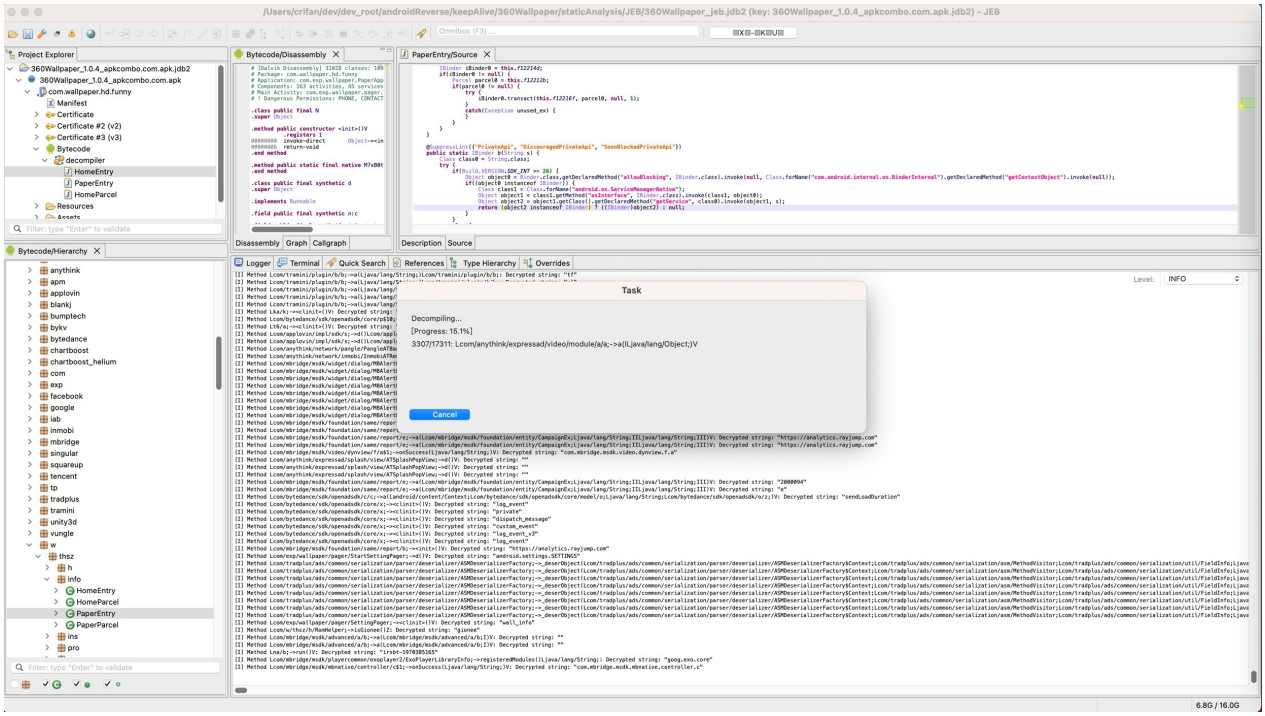
会弹框`Fast Decompilation & Export`：



此处参数设置：

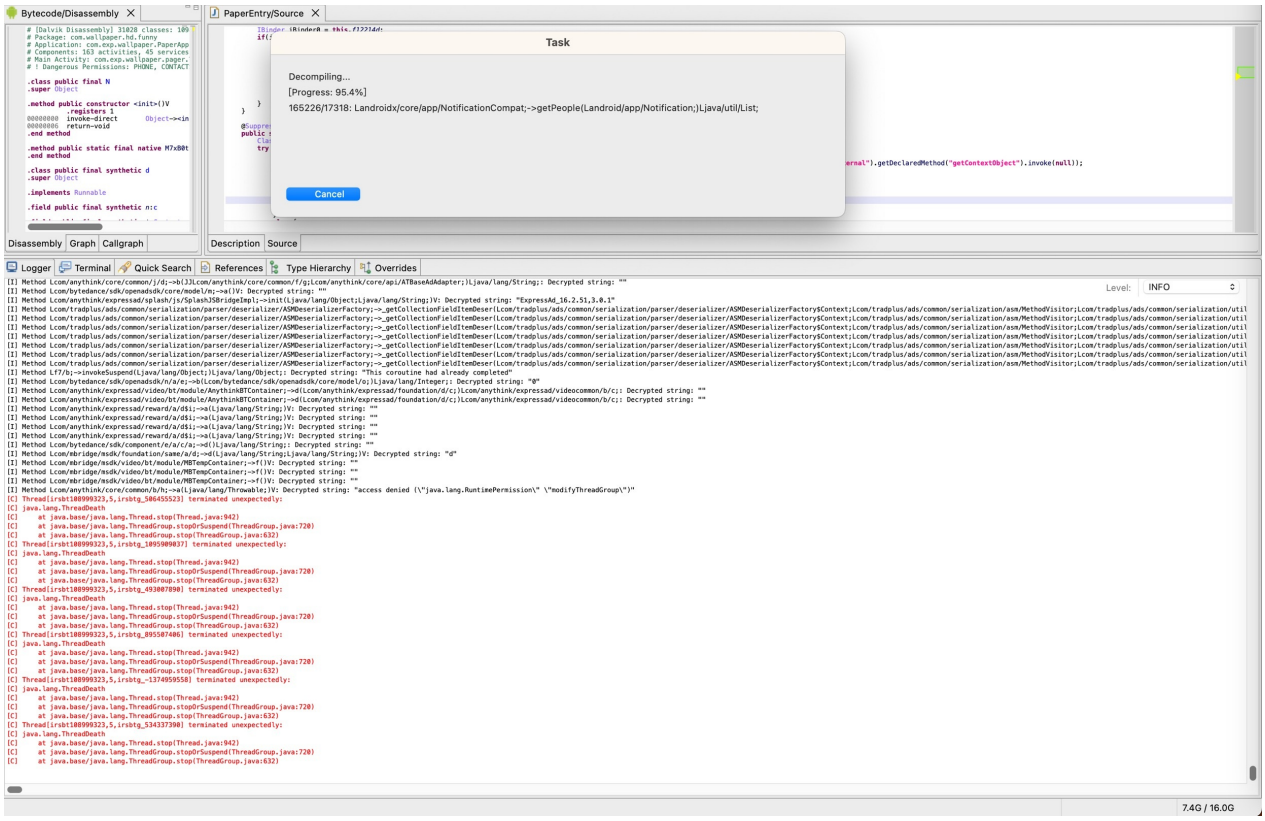
- `Export`：`All`
  - 说明：（反编译后）导出所有Java代码
- `Timeouts`：反编译的超时设置
  - `Timeout per item`：`600` =10分钟
  - `Timeout total`：`60000` =100分钟
- `Destination`：选择保存目标文件夹

然后就是开始反编译，显示进度的弹框了：



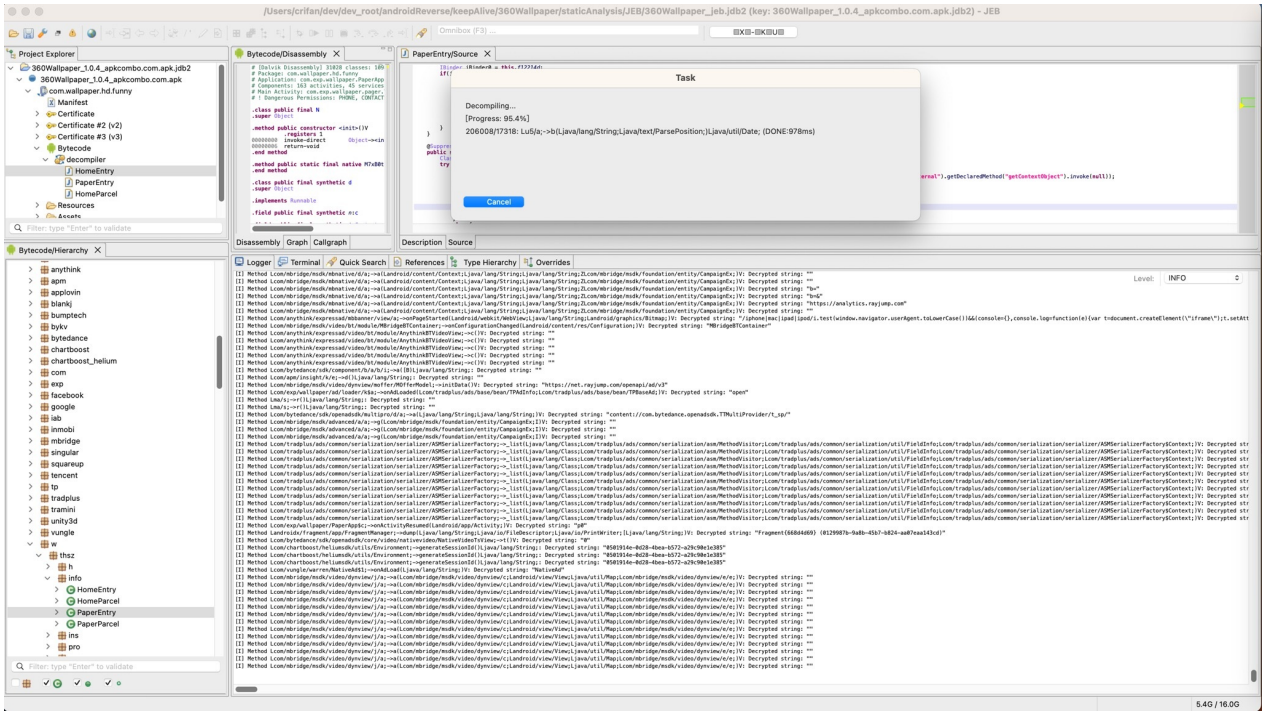
期间底部的Logger会输出对应的日志

偶尔也会看到反编译期间会报错：



正常情况下，应该看到反编译进度 `100%`

但是此处实际上先后2次反编译都出现：反编译进度到 `95%` 左右：

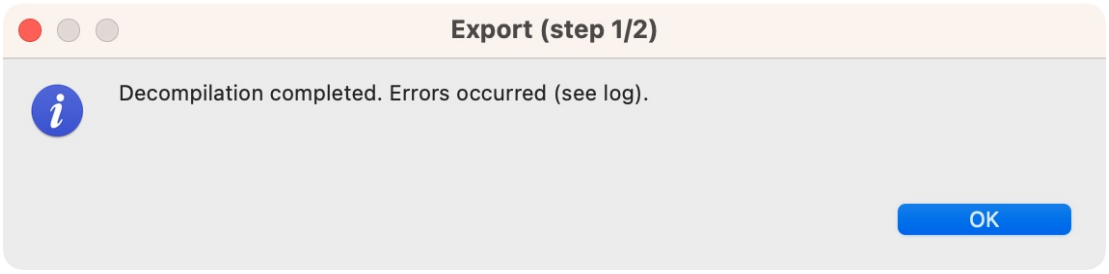


然后就：卡死了，等待很多分钟后，进度也没变化

此时只能：点击 `Cancel` 取消

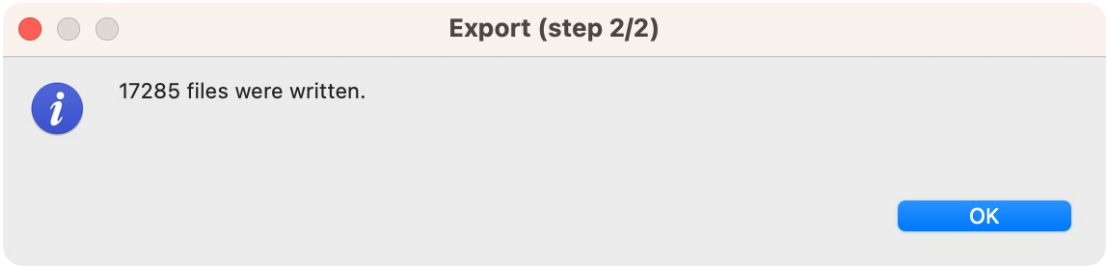
然后可以继续，会弹框提示：

- Export Step (1/2)
  - Decomplilation completed. Errors occurred (see log)

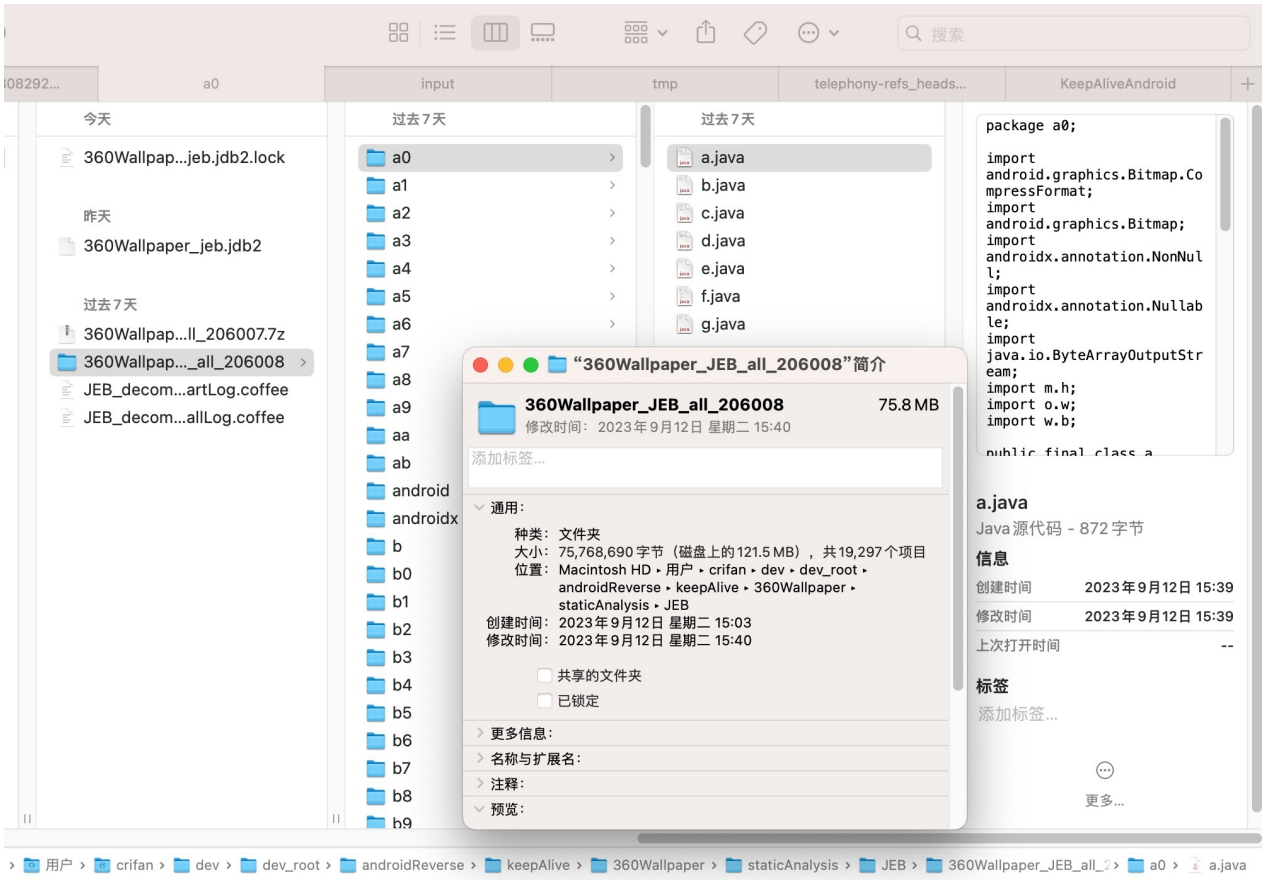


点击 `OK` 继续，显示：

- Export Step (2/2)
  - xxx file were written



最后确认，的确导出了17285个文件：



如此，反编译就算完成了。

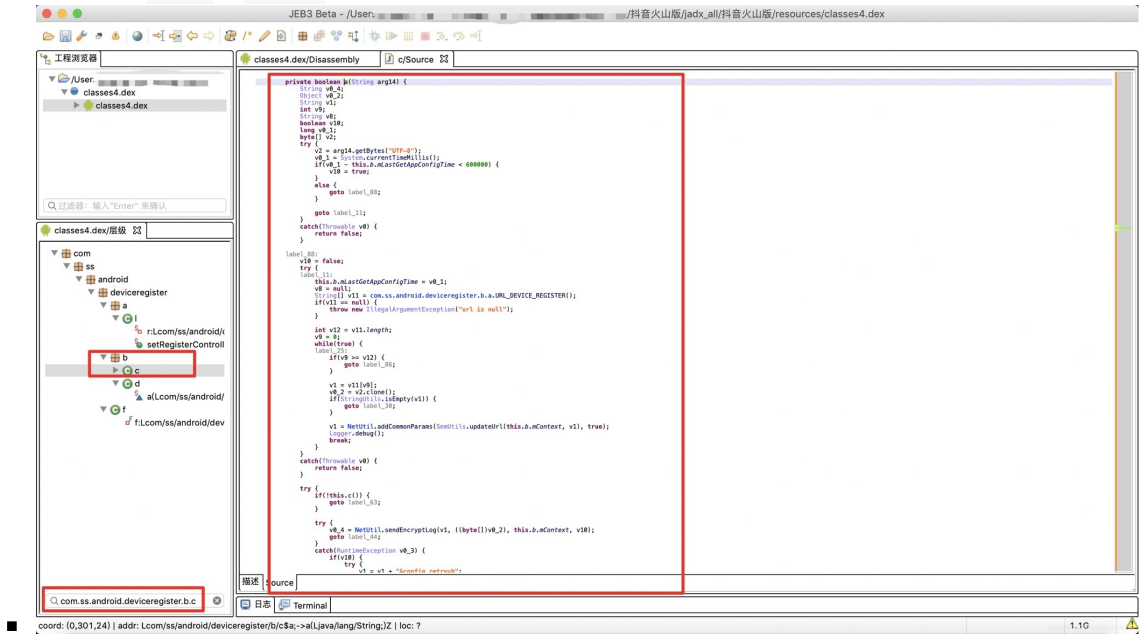
反编译效果对比

JEB的反编译java代码的效果，总体上说，是很好的。

下面通过具体例子对比来说明：

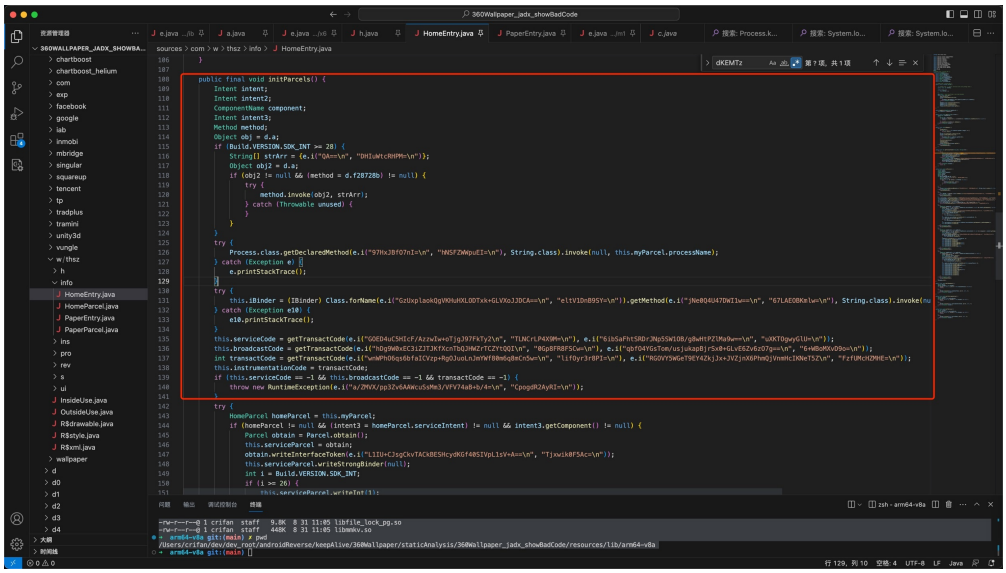
举例：JEB反编译匿名内部类效果比较好

- JEB反编译效果
  - 匿名内部类：`JEB` > `jadx`

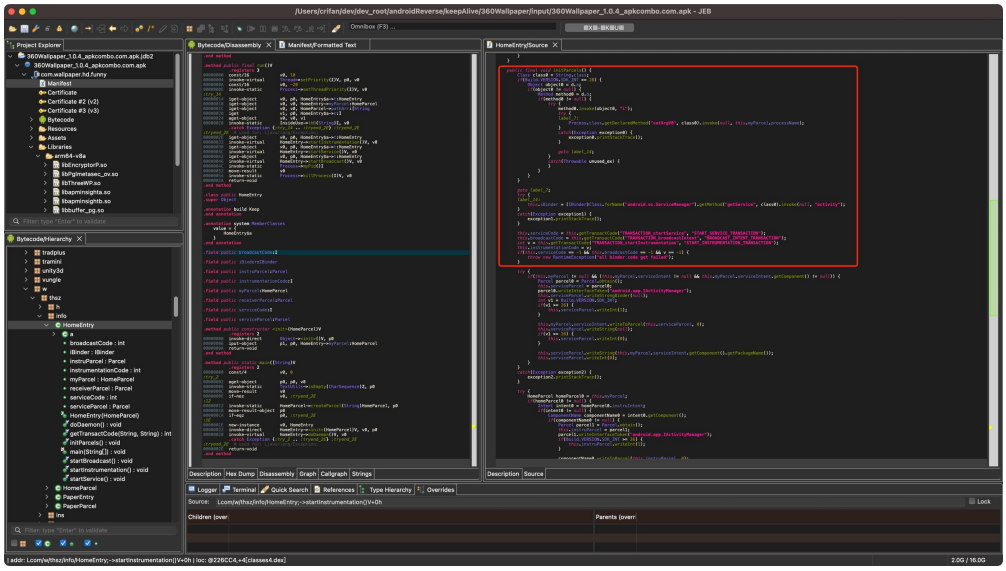


反编译效果对比：JEB vs jadx

概述

- `JEB` vs `jadx` 反编译效果对比概述
  - `JEB` 比 `jadx` 反编译效果还好
    - 可以自动把加密混淆的乱码类字符串，自动反混淆解密出原始正常字符串
      - jadx：无法自动反混淆字符串
      - 

```
    public final void initParcels() {
...
            Process.class.getDeclaredMethod(e.i("97HxJBfO7nI=\n", "hNSFZWWpuEI=\n"), String.class).invoke(null, this.myParcel.processName);
        } catch (Exception e) {
            e.printStackTrace();
        }
        try {
            this.iBinder = (IBinder) Class.forName(e.i("GzUxplaokQgVKHuHXLODTxk+GLVXoJJDCA==\n", "eltV1DnB9SY=\n")).getMethod(e.i("jNe0Q4U47DWI1w==\n", "67LAEOBKmlw=\n"), String.class).invoke(null, e.i("hQG5PXN9gEE=\n", "5GLNVAUU9Dg=\n"));
        } catch (Exception e10) {
            e10.printStackTrace();
        }
        this.serviceCode = getTransactCode(e.i("GOED4uC5HIcF/AzzwIw+oTjgJ97FkTy2\n", "TLNCrLP4X9M=\n"), e.i("6ibSaFhtSRDrJNp5SW1OB/g8wHtPZlMa9w==\n", "uXKTOgwyGlU=\n"));
        this.broadcastCode = getTransactCode(e.i("hDg9W0xEC3iZJTJKfXcnTbQJHWZrTCZYtQQI\n", "0Gp8FR8FSCw=\n"), e.i("qbfO4YGsTom/usjukapBjrSx0+GLvE6Zv6zO7g==\n", "6+WBoMXvD9o=\n"));
        int transactCode = getTransactCode(e.i("wnWPhO6qs6bfaICVzp+RgOJuoLnJmYWf80m6q8mCn5w=\n", "lifOyr3r8PI=\n"), e.i("RGOVY5WGeT9EY4ZkjJx+JVZjnX6PhmQjVnmHcIKNeT5Z\n", "FzfUMcHZMHE=\n"));
```

- JEB：可以自动反混淆字符串 = 还原出原始字符串
  - 

```
    public final void initParcels() {
...
                                Process.class.getDeclaredMethod("setArgV
0", class0).invoke(null, this.myParcel.processName);
...
        }
...
            this.iBinder = (IBinder)Class.forName("android.os.Servic
eManager").getMethod("getService", class0).invoke(null, "activity");
...
        this.serviceCode = this.getTransactCode("TRANSACTION_startSe
rvice", "START_SERVICE_TRANSACTION");
        this.broadcastCode = this.getTransactCode("TRANSACTION_broad
castIntent", "BROADCAST_INTENT_TRANSACTION");
        int v = this.getTransactCode("TRANSACTION_startInstrumentati
on", "START_INSTRUMENTATION_TRANSACTION");
```

举例详解：`360Wallpaper` 反编译效果对比

类：`com.w.thsz.info.HomeEntry` 反编译效果对比

JEB

- 截图

-
-

-
-

-
-
- 代码

```
package                ;

import                 ComponentName;
import                 Intent;
import            Build;
import            IBinder;
import            Parcel;
import            Process;
import              TextUtils;
import                     Keep;
import            InsideUse;
import                   Field;
import                   Method;
import     ;
import     ;
```

```java
@Keep
/* loaded from: classes4.dex */
public class HomeEntry {
    public int broadcastCode;
    public IBinder iBinder;
    public Parcel instruParcel;
    public int instrumentationCode;
    public HomeParcel myParcel;
    public Parcel receiverParcel;
    public int serviceCode;
    public Parcel serviceParcel;

    /* loaded from: classes4.dex */
    public class a extends Thread {

        /* renamed from: n  reason: collision with root package name */
        public final int f23751n;

        public a(int i) {
            this.f23751n = i;
        }

        @Override // java.lang.Thread, java.lang.Runnable
        public final void run() {
            setPriority(10);
            Process.setThreadPriority(-20);
            try {
                InsideUse.thD(HomeEntry.this.myParcel.pathArr[this.f23751n]);
            } catch (Exception unused) {
            }
            HomeEntry.this.startInstrumentation();
            HomeEntry.this.startService();
            HomeEntry.this.startBroadcast();
            Process.killProcess(Process.myPid());
        }
    }

    public HomeEntry(HomeParcel homeParcel) {
        this.myParcel = homeParcel;
    }

    public static void main(String[] strArr) {
        HomeParcel createParcel;
        try {
            String str = strArr[0];
            if (TextUtils.isEmpty(str) || (createParcel = HomeParcel.createParcel(str))
== null) {
                return;
            }
            new HomeEntry(createParcel).doDaemon();
        } catch (Exception unused) {
        }
    }

    public final void doDaemon() {
        try {
```

```
            initParcels();
            for (int i = 1; i < this.myParcel.pathArr.length; i++) {
                new a(i).start();
            }
            Thread.currentThread().setPriority(10);
            Process.setThreadPriority(-20);
            try {
                InsideUse.thD(this.myParcel.pathArr[0]);
            } catch (Exception unused) {
            }
            startInstrumentation();
            startService();
            startBroadcast();
            Process.killProcess(Process.myPid());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public final int getTransactCode(String str, String str2) {
        try {
            try {
                Class<?> cls = Class.forName(e.i("dKEMTz+a/hF0vxgTGbL5S3y5AUkpvvtRdKgNT
3Sg7kp3\n", "Fc9oPVDzmj8=\n"));
                Field declaredField = cls.getDeclaredField(str);
                declaredField.setAccessible(true);
                return declaredField.getInt(cls);
            } catch (Exception unused) {
                Class<?> cls2 = Class.forName(e.i("G96QImCjHHQbwIR+RosbLhPGnSR2hxk0G9eR
Ig==\n", "erD0UA/KeFo=\n"));
                Field declaredField2 = cls2.getDeclaredField(str2);
                declaredField2.setAccessible(true);
                return declaredField2.getInt(cls2);
            }
        } catch (Exception unused2) {
            return -1;
        }
    }

    public final void initParcels() {
        Intent intent;
        Intent intent2;
        ComponentName component;
        Intent intent3;
        Method method;
        Object obj = d.a;
        if (Build.VERSION.SDK_INT >= 28) {
            String[] strArr = {e.i("QA==\n", "DHIuWtcRHPM=\n")};
            Object obj2 = d.a;
            if (obj2 != null && (method = d.f28728b) != null) {
                try {
                    method.invoke(obj2, strArr);
                } catch (Throwable unused) {
                }
            }
        }
```

```
        try {
            Process.class.getDeclaredMethod(e.i("97HxJBfO7nI=\n", "hNSFZWWpuEI=\n"), St
ring.class).invoke(null, this.myParcel.processName);
        } catch (Exception e) {
            e.printStackTrace();
        }
        try {
            this.iBinder = (IBinder) Class.forName(e.i("GzUxplaokQgVKHuHXLODTxk+GLVXoJJ
DCA==\n", "eltV1DnB9SY=\n")).getMethod(e.i("jNe0Q4U47DWI1w==\n", "67LAEOBKmlw=\n"), Str
ing.class).invoke(null, e.i("hQG5PXN9gEE=\n", "5GLNVAUU9Dg=\n"));
        } catch (Exception e10) {
            e10.printStackTrace();
        }
        this.serviceCode = getTransactCode(e.i("GOED4uC5HIcF/AzzwIw+oTjgJ97FkTy2\n", "T
LNCrLP4X9M=\n"), e.i("6ibSaFhtSRDrJNp5SW1OB/g8wHtPZlMa9w==\n", "uXKTOgwyGlU=\n"));
        this.broadcastCode = getTransactCode(e.i("hDg9W0xEC3iZJTJKfXcnTbQJHWZrTCZYtQQI\
n", "0Gp8FR8FSCw=\n"), e.i("qbfO4YGsTom/usjukapBjrSx0+GLvE6Zv6zO7g==\n", "6+WBoMXvD9o=\
n"));
        int transactCode = getTransactCode(e.i("wnWPhO6qs6bfaICVzp+RgOJuoLnJmYWf80m6q8m
Cn5w=\n", "lifOyr3r8PI=\n"), e.i("RGOVY5WGeT9EY4ZkjJx+JVZjnX6PhmQjVnmHcIKNeT5Z\n", "FzfF
UMcHZMHE=\n"));
        this.instrumentationCode = transactCode;
        if (this.serviceCode == -1 && this.broadcastCode == -1 && transactCode == -1) {
            throw new RuntimeException(e.i("a/ZMVX/pp3Zv6AAWcuSsMm3/VFV74aB+b/4=\n", "C
pogdR2AyRI=\n"));
        }
        try {
            HomeParcel homeParcel = this.myParcel;
            if (homeParcel != null && (intent3 = homeParcel.serviceIntent) != null && i
ntent3.getComponent() != null) {
                Parcel obtain = Parcel.obtain();
                this.serviceParcel = obtain;
                obtain.writeInterfaceToken(e.i("L1IU+CJsgCkvTACkBESHcydKGf40SIVpL1sV+A=
=\n", "Tjxwik0F5Ac=\n"));
                this.serviceParcel.writeStrongBinder(null);
                int i = Build.VERSION.SDK_INT;
                if (i >= 26) {
                    this.serviceParcel.writeInt(1);
                }
                this.myParcel.serviceIntent.writeToParcel(this.serviceParcel, 0);
                this.serviceParcel.writeString(null);
                if (i >= 26) {
                    this.serviceParcel.writeInt(0);
                }
                this.serviceParcel.writeString(this.myParcel.serviceIntent.getComponent(
).getPackageName());
                this.serviceParcel.writeInt(0);
            }
        } catch (Exception e11) {
            e11.printStackTrace();
        }
        try {
            HomeParcel homeParcel2 = this.myParcel;
            if (homeParcel2 != null && (intent2 = homeParcel2.instruIntent) != null && (
component = intent2.getComponent()) != null) {
                Parcel obtain2 = Parcel.obtain();
```

```
                this.instruParcel = obtain2;
                obtain2.writeInterfaceToken(e.i("23Kbgc7H69HbbI/d6O/si9NqlofY4+6R23uagQ
==\n", "uhz/86Guj/8=\n"));
                if (Build.VERSION.SDK_INT >= 26) {
                    this.instruParcel.writeInt(1);
                }
                component.writeToParcel(this.instruParcel, 0);
                this.instruParcel.writeString(null);
                this.instruParcel.writeInt(0);
                this.instruParcel.writeInt(0);
                this.instruParcel.writeStrongBinder(null);
                this.instruParcel.writeStrongBinder(null);
                this.instruParcel.writeInt(0);
                this.instruParcel.writeString(null);
            }
        } catch (Exception e12) {
            e12.printStackTrace();
        }
        try {
            HomeParcel homeParcel3 = this.myParcel;
            if (homeParcel3 != null && (intent = homeParcel3.dReceiverIntent) != null) {

                intent.setFlags(16);
                Parcel obtain3 = Parcel.obtain();
                this.receiverParcel = obtain3;
                obtain3.writeInterfaceToken(e.i("pMYKuJux1TCk2B7kvZnSaqzeB76NldBwpM8LuA
==\n", "xahuyvTYsR4=\n"));
                this.receiverParcel.writeStrongBinder(null);
                if (Build.VERSION.SDK_INT >= 26) {
                    this.receiverParcel.writeInt(1);
                }
                this.myParcel.dReceiverIntent.writeToParcel(this.receiverParcel, 0);
                this.receiverParcel.writeString(null);
                this.receiverParcel.writeStrongBinder(null);
                this.receiverParcel.writeInt(-1);
                this.receiverParcel.writeString(null);
                this.receiverParcel.writeInt(0);
                this.receiverParcel.writeStringArray(null);
                this.receiverParcel.writeInt(-1);
                this.receiverParcel.writeInt(0);
                this.receiverParcel.writeInt(0);
                this.receiverParcel.writeInt(0);
                this.receiverParcel.writeInt(0);
            }
        } catch (Exception e13) {
            e13.printStackTrace();
        }
        InsideUse.thC();
    }

    public final void startBroadcast() {
        Parcel parcel;
        IBinder iBinder = this.iBinder;
        if (iBinder == null || (parcel = this.receiverParcel) == null) {
            return;
        }
```

```
        try {
            iBinder.transact(this.broadcastCode, parcel, null, 1);
        } catch (Exception unused) {
        }
    }

    public final void startInstrumentation() {
        IBinder iBinder;
        Parcel parcel = this.instruParcel;
        if (parcel == null || (iBinder = this.iBinder) == null) {
            return;
        }
        try {
            iBinder.transact(this.instrumentationCode, parcel, null, 1);
        } catch (Exception unused) {
        }
    }

    public final void startService() {
        IBinder iBinder;
        Parcel parcel = this.serviceParcel;
        if (parcel == null || (iBinder = this.iBinder) == null) {
            return;
        }
        try {
            iBinder.transact(this.serviceCode, parcel, null, 1);
        } catch (Exception unused) {
        }
    }
}
```

jadx

- 截图

-
-

-
- 代码

```
package                ;

import                 ComponentName;
import                 Intent;
import            Build.VERSION;
import            IBinder;
import            Parcel;
import            Process;
import              TextUtils;
import                    Keep;
import            InsideUse;
import                  Field;
import                  Method;
import     ;

@Keep
public class HomeEntry {
    public final class a extends Thread {
        public final int n;
        public final HomeEntry t;

        public a(int v) {
            this.n = v;
        }

        @Override
        public final void run() {
            this.setPriority(10);
            Process.setThreadPriority(-20);
            try {
                InsideUse.thD(HomeEntry.this.myParcel.pathArr[this.n]);
            }
            catch(Exception unused_ex) {
            }
```

```
            HomeEntry.this.startInstrumentation();
            HomeEntry.this.startService();
            HomeEntry.this.startBroadcast();
            Process.killProcess(Process.myPid());
        }
    }

    public int broadcastCode;
    public IBinder iBinder;
    public Parcel instruParcel;
    public int instrumentationCode;
    public HomeParcel myParcel;
    public Parcel receiverParcel;
    public int serviceCode;
    public Parcel serviceParcel;

    public HomeEntry(HomeParcel homeParcel0) {
        this.myParcel = homeParcel0;
    }

    public final void doDaemon() {
        try {
            this.initParcels();
            for(int v = 1; v < this.myParcel.pathArr.length; ++v) {
                new a(this, v).start();
            }

            Thread.currentThread().setPriority(10);
            Process.setThreadPriority(-20);
        }
        catch(Exception exception0) {
            goto label_15;
        }

        try {
            InsideUse.thD(this.myParcel.pathArr[0]);
        }
        catch(Exception unused_ex) {
        }

        try {
            this.startInstrumentation();
            this.startService();
            this.startBroadcast();
            Process.killProcess(Process.myPid());
            return;
        }
        catch(Exception exception0) {
        }

    label_15:
        exception0.printStackTrace();
    }

    public final int getTransactCode(String s, String s1) {
```

```
        try {
            Class class0 = Class.forName("android.app.IActivityManager$Stub");
            Field field0 = class0.getDeclaredField(s);
            field0.setAccessible(true);
            return field0.getInt(class0);
        }
        catch(Exception unused_ex) {
            try {
                Class class1 = Class.forName("android.app.IActivityManager");
                Field field1 = class1.getDeclaredField(s1);
                field1.setAccessible(true);
                return field1.getInt(class1);
            }
            catch(Exception unused_ex) {
                return -1;
            }
        }
    }

    public final void initParcels() {
        Class class0 = String.class;
        if(Build.VERSION.SDK_INT >= 28) {
            Object object0 = d.a;
            if(object0 != null) {
                Method method0 = d.b;
                if(method0 != null) {
                    try {
                        method0.invoke(object0, "L");
                        try {
                        label_7
                            Process.class.getDeclaredMethod("setArgV0", class0).invoke(
null, this.myParcel.processName);
                        }
                        catch(Exception exception0) {
                            exception0.printStackTrace();
                        }

                        goto label_14;
                    }
                    catch(Throwable unused_ex) {
                    }
                }
            }
        }

        goto label_7;
        try {
        label_14:
            this.iBinder = (IBinder)Class.forName("android.os.ServiceManager").getMethod
("getService", class0).invoke(null, "activity");
        }
        catch(Exception exception1) {
            exception1.printStackTrace();
        }

        this.serviceCode = this.getTransactCode("TRANSACTION_startService", "START_SERV
```

```
ICE_TRANSACTION");
        this.broadcastCode = this.getTransactCode("TRANSACTION_broadcastIntent", "BROAD
CAST_INTENT_TRANSACTION");
        int v = this.getTransactCode("TRANSACTION_startInstrumentation", "START_INSTRUM
ENTATION_TRANSACTION");
        this.instrumentationCode = v;
        if(this.serviceCode == -1 && this.broadcastCode == -1 && v == -1) {
            throw new RuntimeException("all binder code get failed");
        }

        try {
            if(this.myParcel != null && (this.myParcel.serviceIntent != null && this.my
Parcel.serviceIntent.getComponent() != null)) {
                Parcel parcel0 = Parcel.obtain();
                this.serviceParcel = parcel0;
                parcel0.writeInterfaceToken("android.app.IActivityManager");
                this.serviceParcel.writeStrongBinder(null);
                int v1 = Build.VERSION.SDK_INT;
                if(v1 >= 26) {
                    this.serviceParcel.writeInt(1);
                }

                this.myParcel.serviceIntent.writeToParcel(this.serviceParcel, 0);
                this.serviceParcel.writeString(null);
                if(v1 >= 26) {
                    this.serviceParcel.writeInt(0);
                }

                this.serviceParcel.writeString(this.myParcel.serviceIntent.getComponent(
).getPackageName());
                this.serviceParcel.writeInt(0);
            }
        }
        catch(Exception exception2) {
            exception2.printStackTrace();
        }

        try {
            HomeParcel homeParcel0 = this.myParcel;
            if(homeParcel0 != null) {
                Intent intent0 = homeParcel0.instruIntent;
                if(intent0 != null) {
                    ComponentName componentName0 = intent0.getComponent();
                    if(componentName0 != null) {
                        Parcel parcel1 = Parcel.obtain();
                        this.instruParcel = parcel1;
                        parcel1.writeInterfaceToken("android.app.IActivityManager");
                        if(Build.VERSION.SDK_INT >= 26) {
                            this.instruParcel.writeInt(1);
                        }

                        componentName0.writeToParcel(this.instruParcel, 0);
                        this.instruParcel.writeString(null);
                        this.instruParcel.writeInt(0);
                        this.instruParcel.writeInt(0);
                        this.instruParcel.writeStrongBinder(null);
```

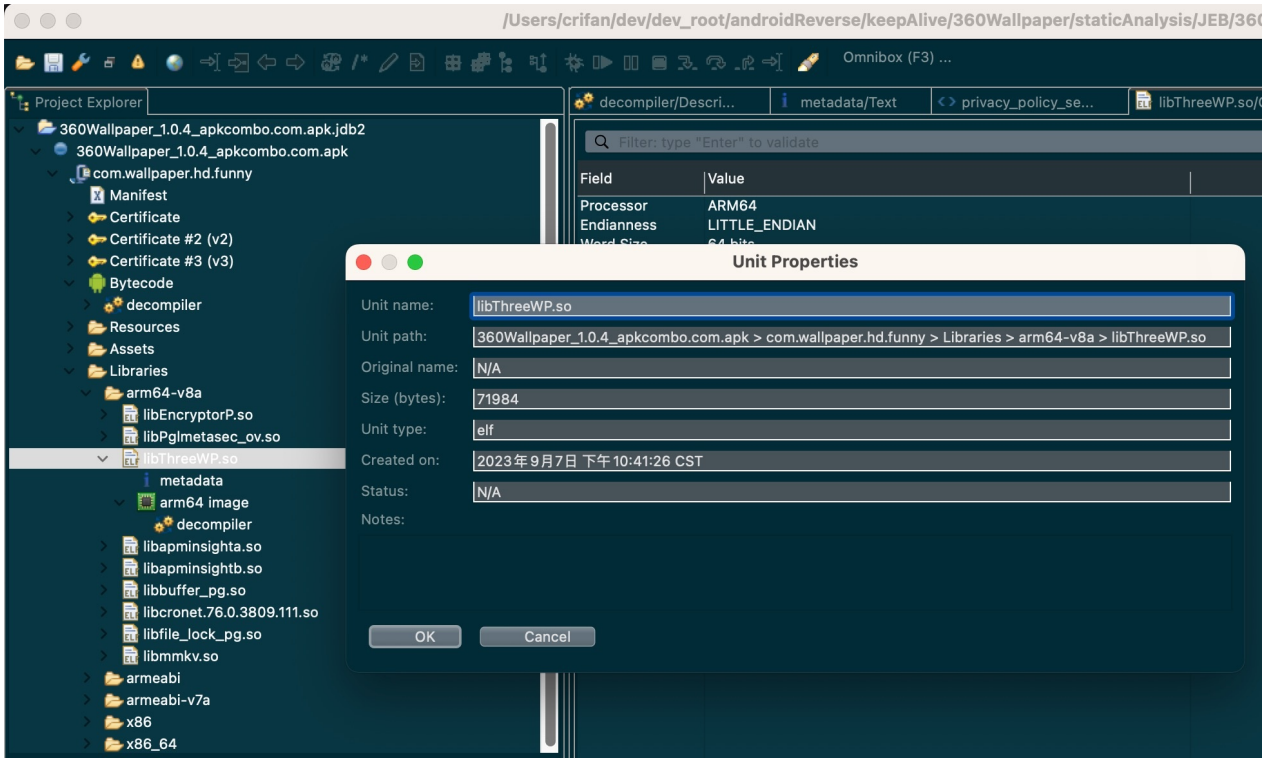
```
                    this.instruParcel.writeStrongBinder(null);
                    this.instruParcel.writeInt(0);
                    this.instruParcel.writeString(null);
                }
            }
        }
    }
    catch(Exception exception3) {
        exception3.printStackTrace();
    }

    try {
        HomeParcel homeParcel1 = this.myParcel;
        if(homeParcel1 != null) {
            Intent intent1 = homeParcel1.dReceiverIntent;
            if(intent1 != null) {
                intent1.setFlags(16);
                Parcel parcel2 = Parcel.obtain();
                this.receiverParcel = parcel2;
                parcel2.writeInterfaceToken("android.app.IActivityManager");
                this.receiverParcel.writeStrongBinder(null);
                if(Build.VERSION.SDK_INT >= 26) {
                    this.receiverParcel.writeInt(1);
                }

                this.myParcel.dReceiverIntent.writeToParcel(this.receiverParcel, 0);

                this.receiverParcel.writeString(null);
                this.receiverParcel.writeStrongBinder(null);
                this.receiverParcel.writeInt(-1);
                this.receiverParcel.writeString(null);
                this.receiverParcel.writeInt(0);
                this.receiverParcel.writeStringArray(null);
                this.receiverParcel.writeInt(-1);
                this.receiverParcel.writeInt(0);
                this.receiverParcel.writeInt(0);
                this.receiverParcel.writeInt(0);
                this.receiverParcel.writeInt(0);
            }
        }
    }
    catch(Exception exception4) {
        exception4.printStackTrace();
    }

    InsideUse.thC();
}

public static void main(String[] arr_s) {
    try {
        String s = arr_s[0];
        if(!TextUtils.isEmpty(s)) {
            HomeParcel homeParcel0 = HomeParcel.createParcel(s);
            if(homeParcel0 != null) {
                new HomeEntry(homeParcel0).doDaemon();
            }
```

```
            }
        }
        catch(Exception unused_ex) {
        }
    }

    public final void startBroadcast() {
        IBinder iBinder0 = this.iBinder;
        if(iBinder0 != null) {
            Parcel parcel0 = this.receiverParcel;
            if(parcel0 != null) {
                try {
                    iBinder0.transact(this.broadcastCode, parcel0, null, 1);
                }
                catch(Exception unused_ex) {
                }
            }
        }
    }

    public final void startInstrumentation() {
        Parcel parcel0 = this.instruParcel;
        if(parcel0 != null) {
            IBinder iBinder0 = this.iBinder;
            if(iBinder0 != null) {
                try {
                    iBinder0.transact(this.instrumentationCode, parcel0, null, 1);
                }
                catch(Exception unused_ex) {
                }
            }
        }
    }

    public final void startService() {
        Parcel parcel0 = this.serviceParcel;
        if(parcel0 != null) {
            IBinder iBinder0 = this.iBinder;
            if(iBinder0 != null) {
                try {
                    iBinder0.transact(this.serviceCode, parcel0, null, 1);
                }
                catch(Exception unused_ex) {
                }
            }
        }
    }
}
```

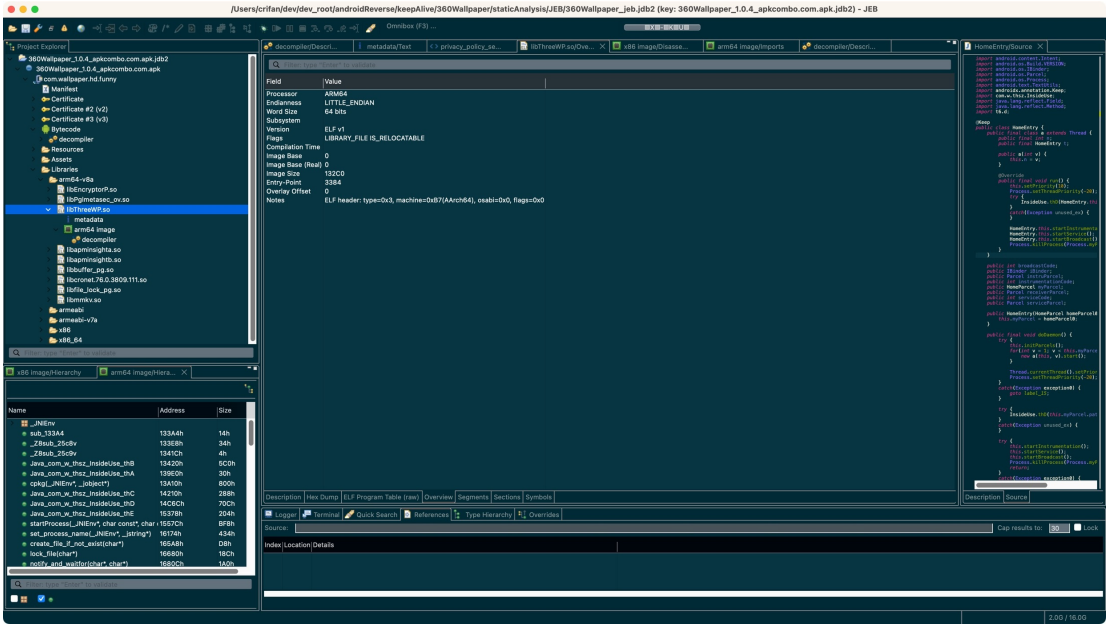
解析so库文件

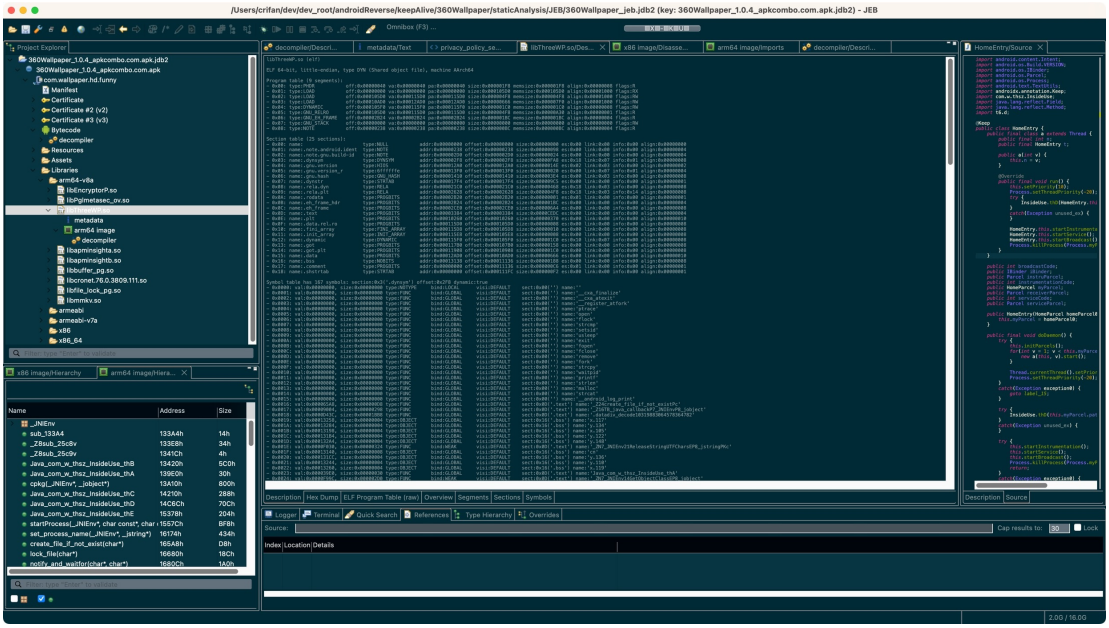
继续查看，JEB反编译apk后，对于其中的so库文件的解析。

展开到：`apk` -> `app包名` -> `Libraries` ->此处的某个so库文件：`libThreeWP.so`

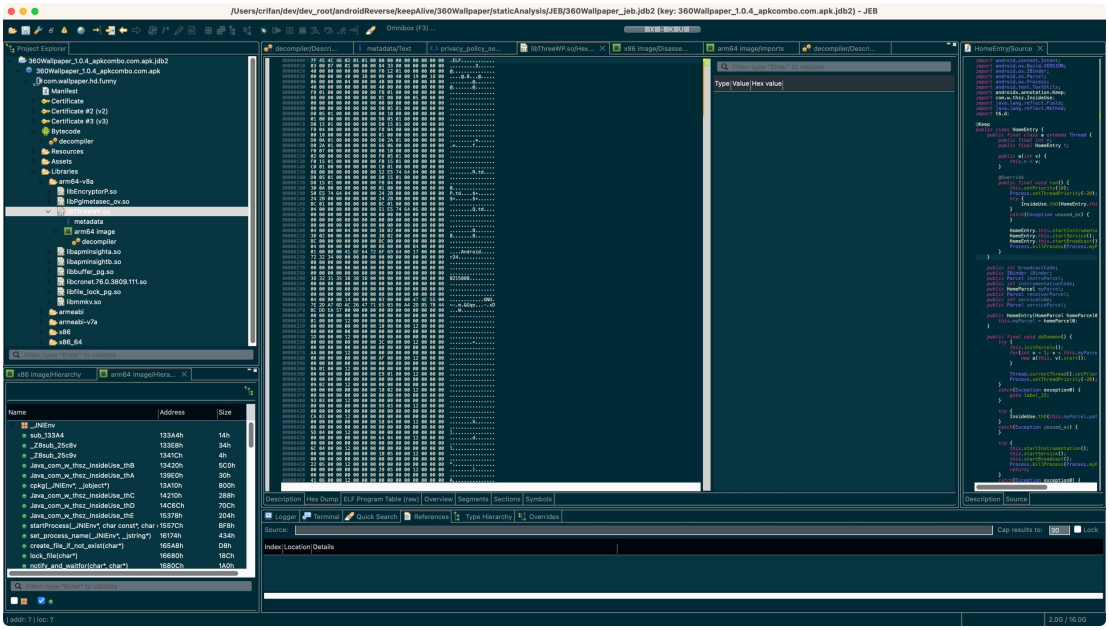


然后可以分别查看解析效果：

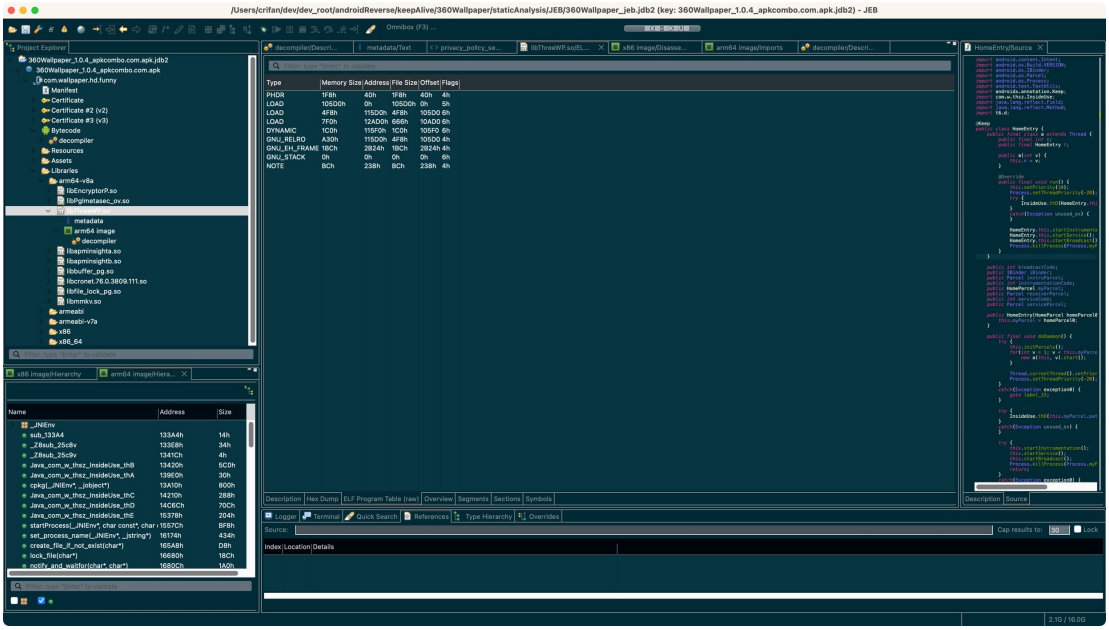
- （ELF格式的）so库文件
  - Overview
    - 
  - Description



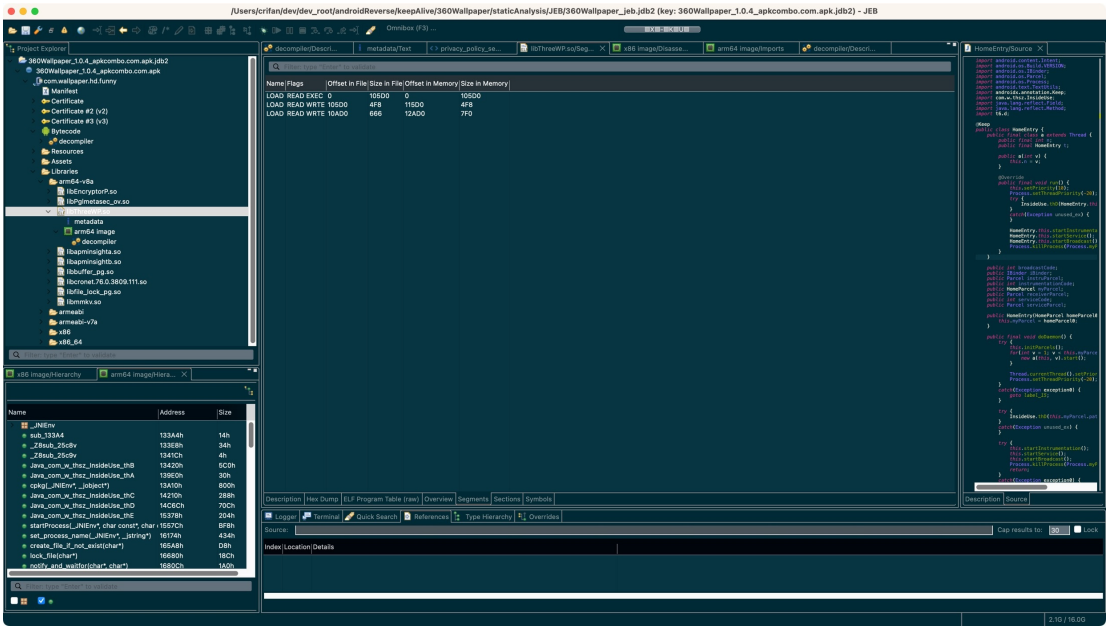
- Hex Dump



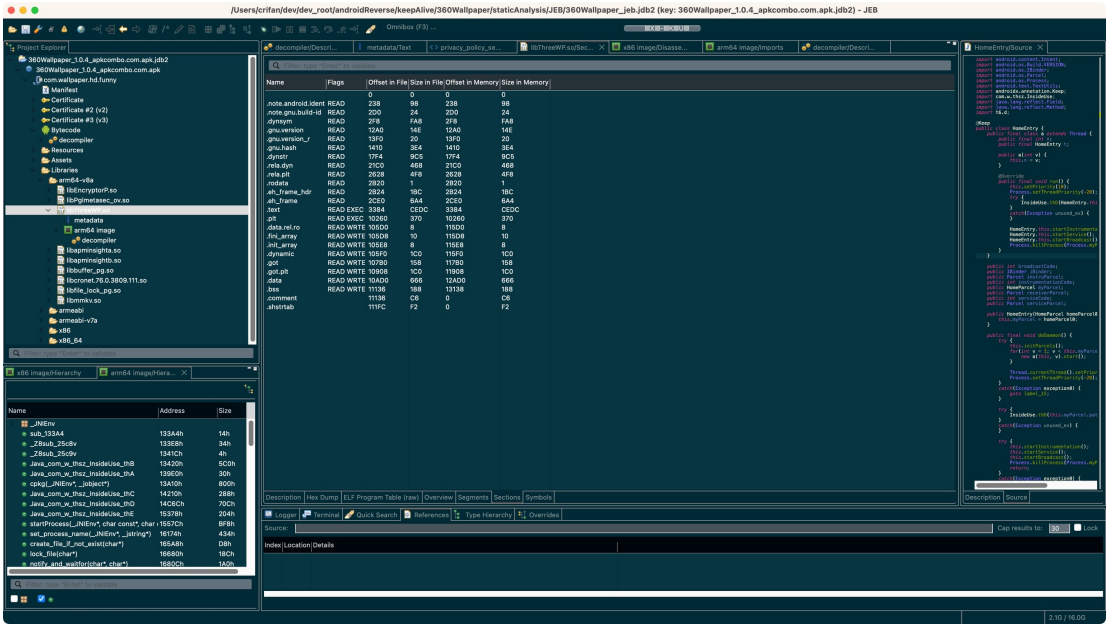
- ELF Program Table (raw)



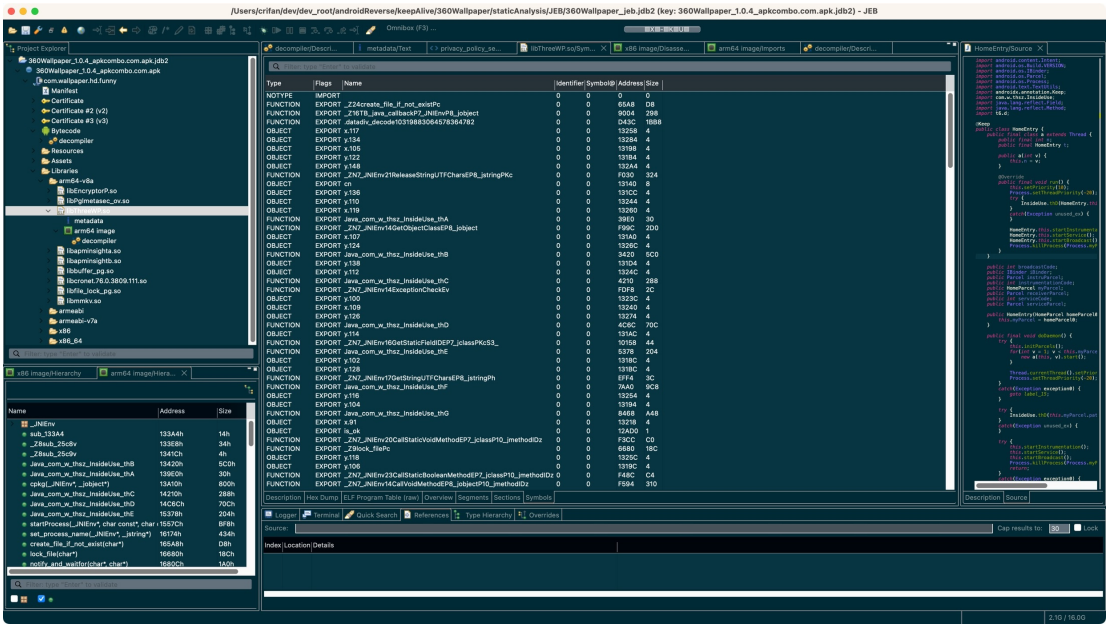
- Segements



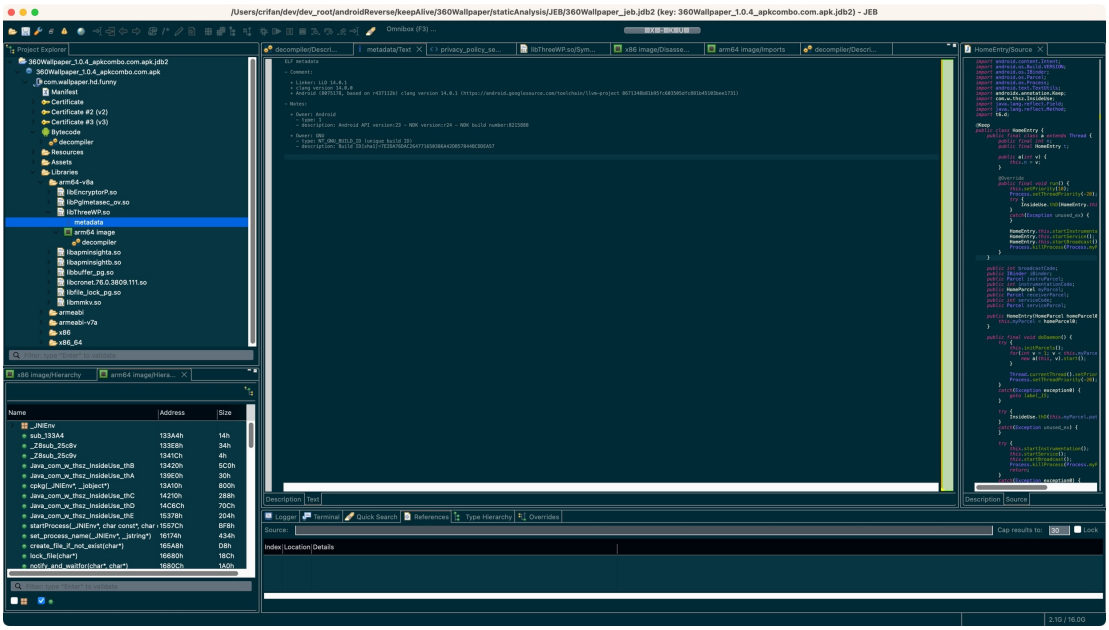
- Sections



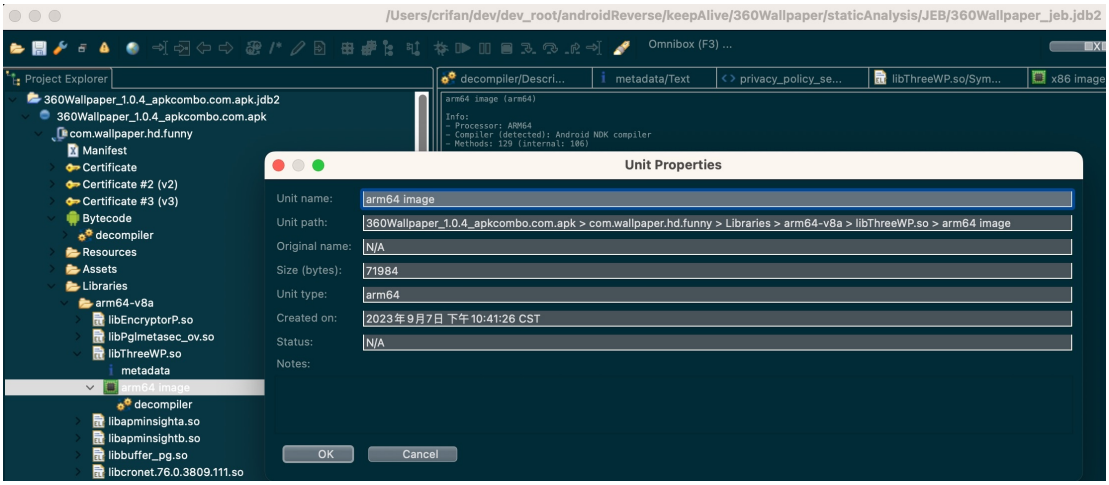
- Symbols



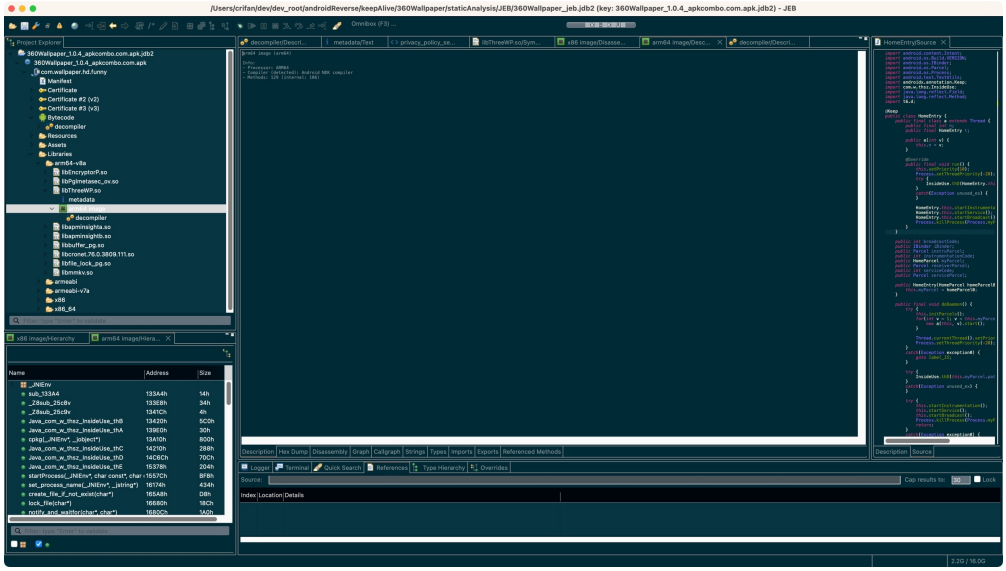
- so下面还有2个子项
  - metadata



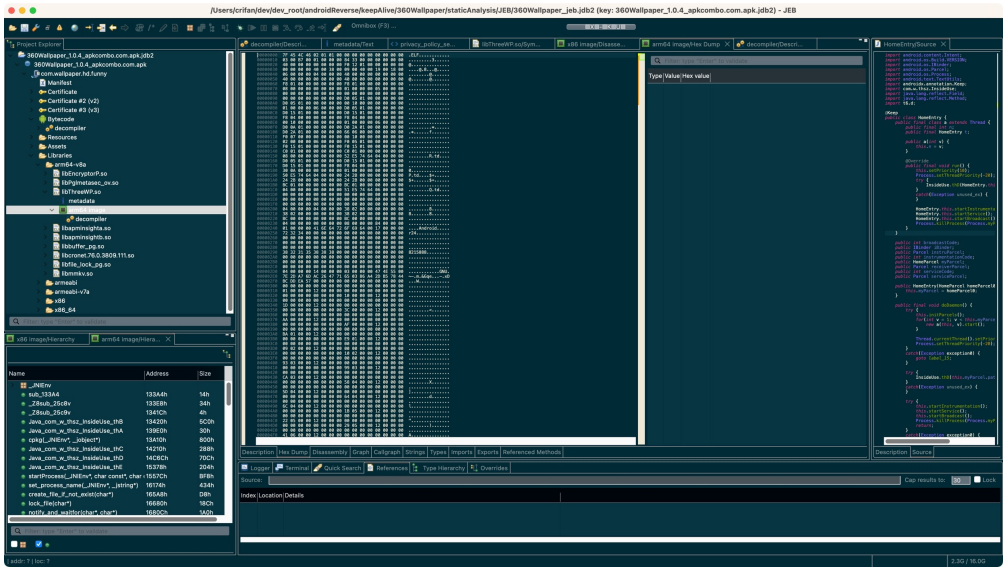
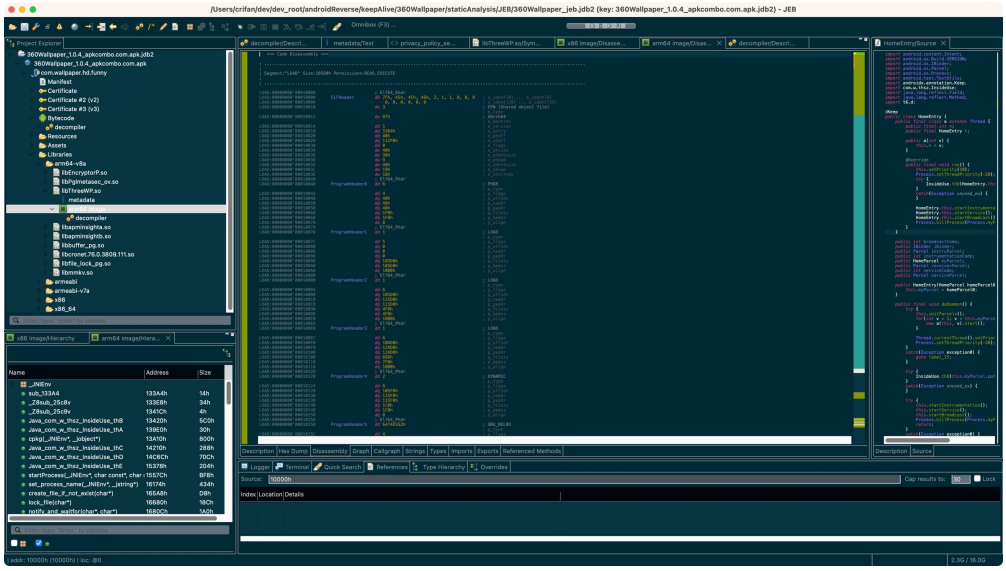
- arm64 image

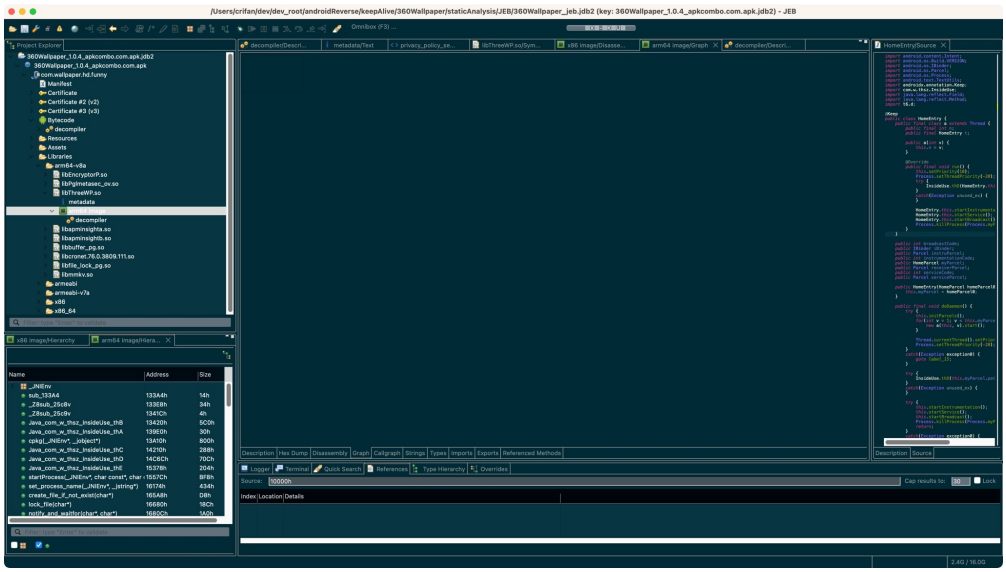
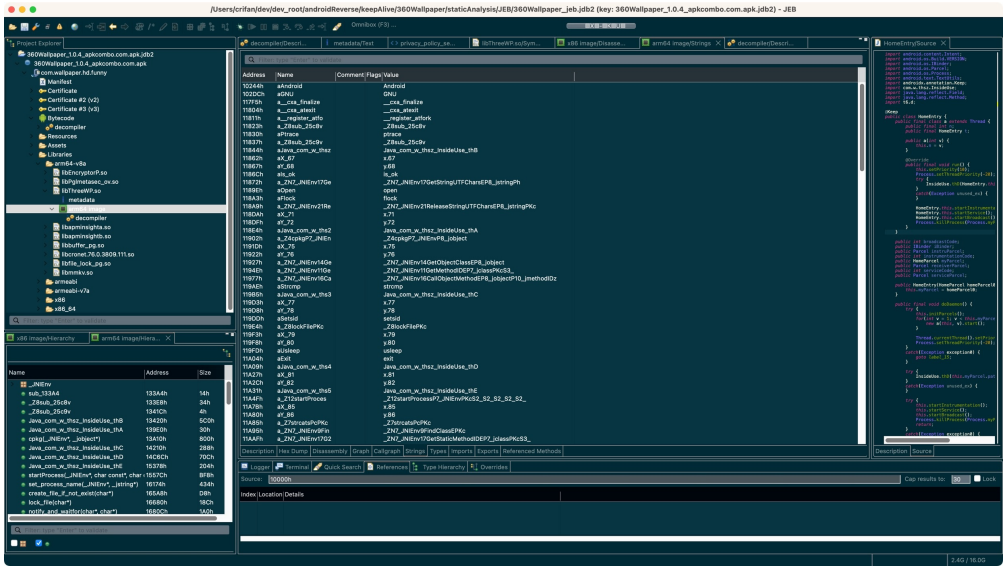
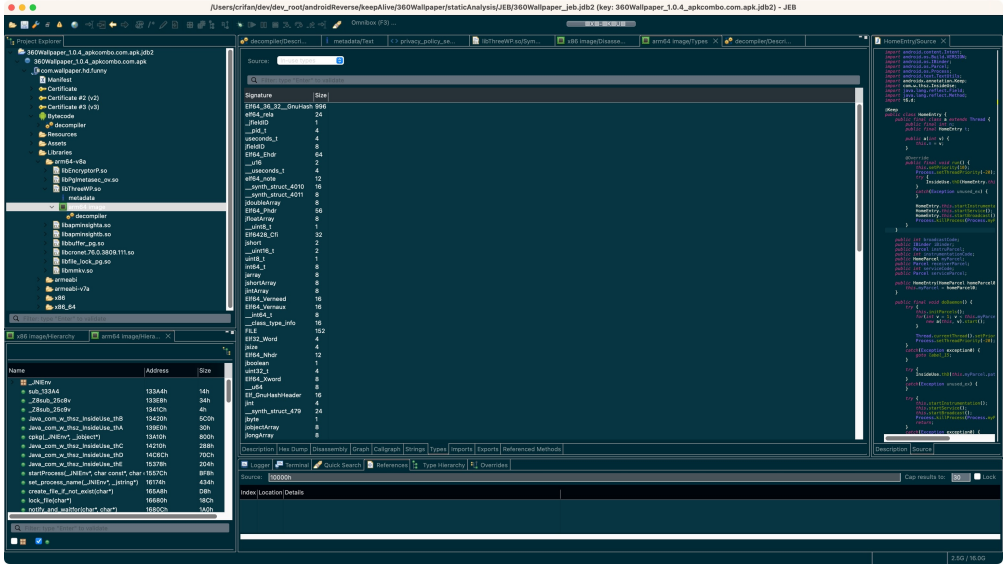


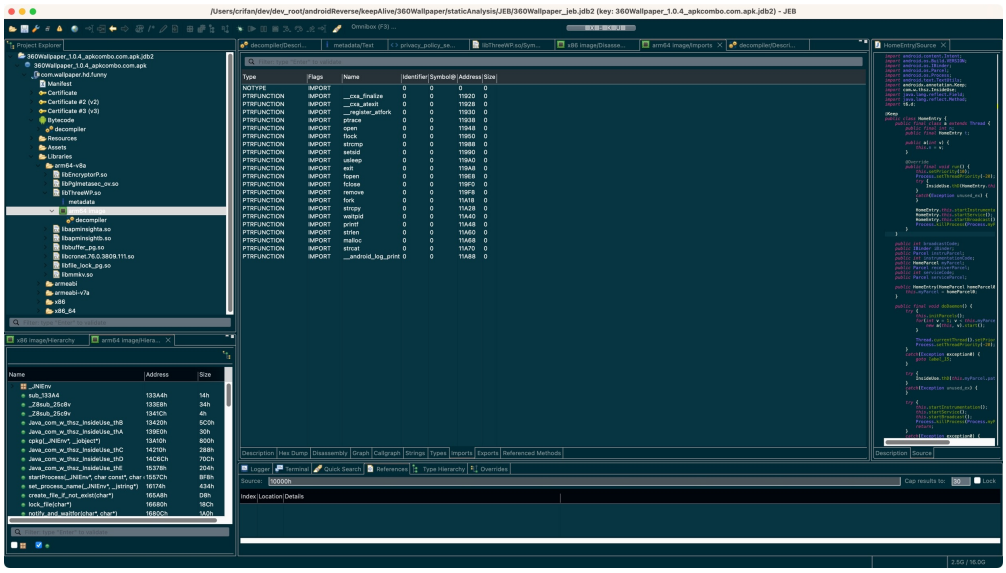
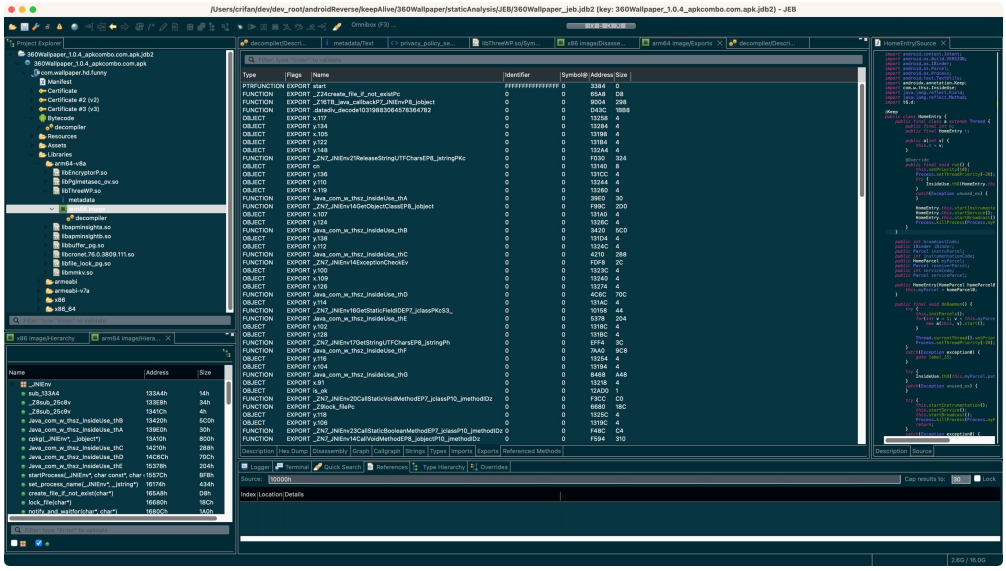
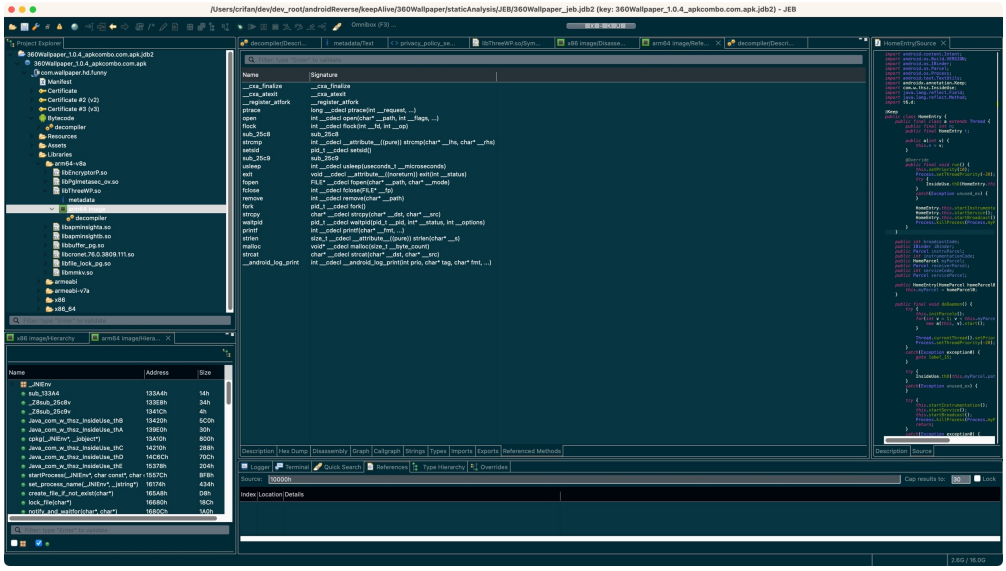
- 其也有各个子页面
  - Description

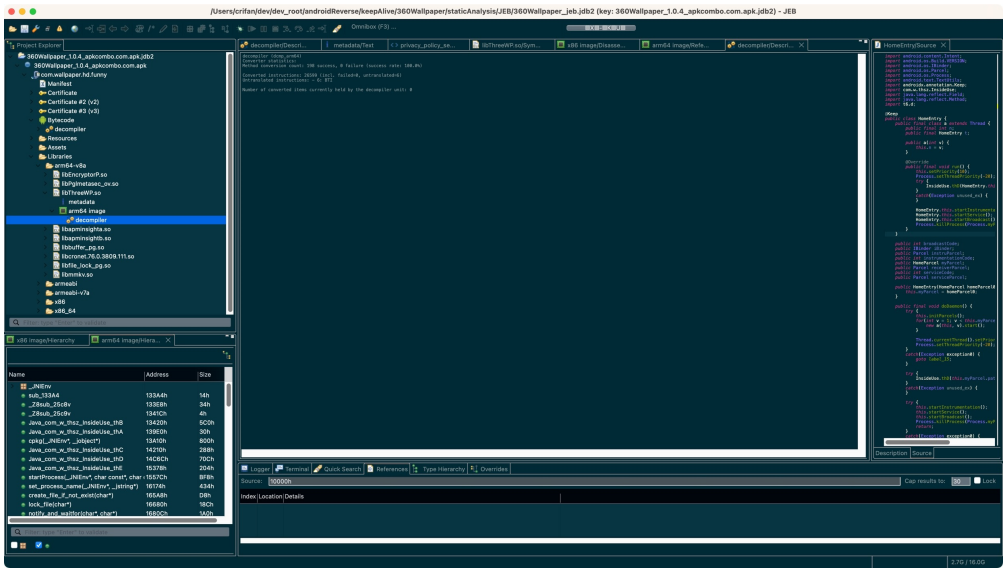


```
arm64 image (arm64)

Info:
- Processor: ARM64
- Compiler (detected): Android NDK compiler
- Methods: 129 (internal: 106)
```

- Hex Dump
  - 
- Disassembly
  - 
- Graph

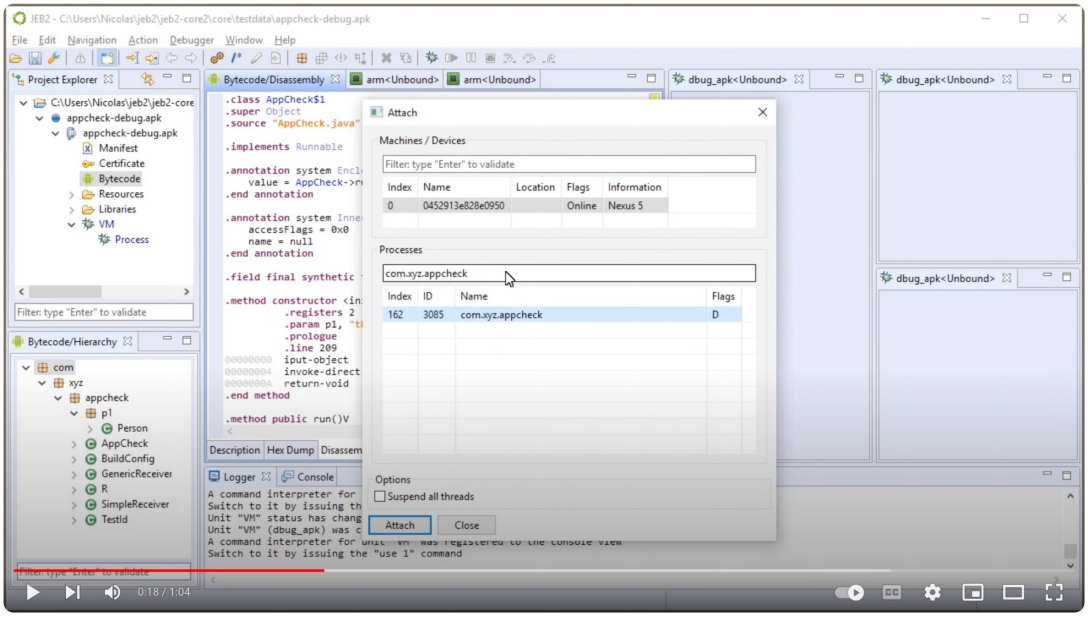
- 
- Callgraph
  - 注：容易导致卡死
    - TODO：有空再去试试
- Stings
  - 
- Types
  - 

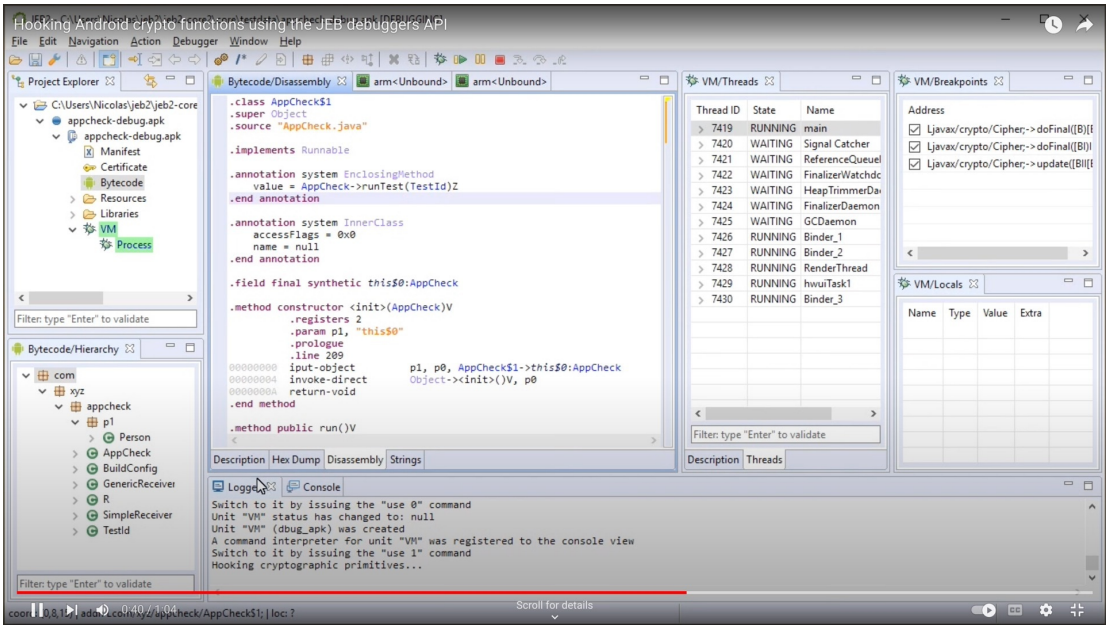
- Imports
  - 
- Exports
  - 
- Referenced Methods
  - 
- 其下也有子项
  - decompiler

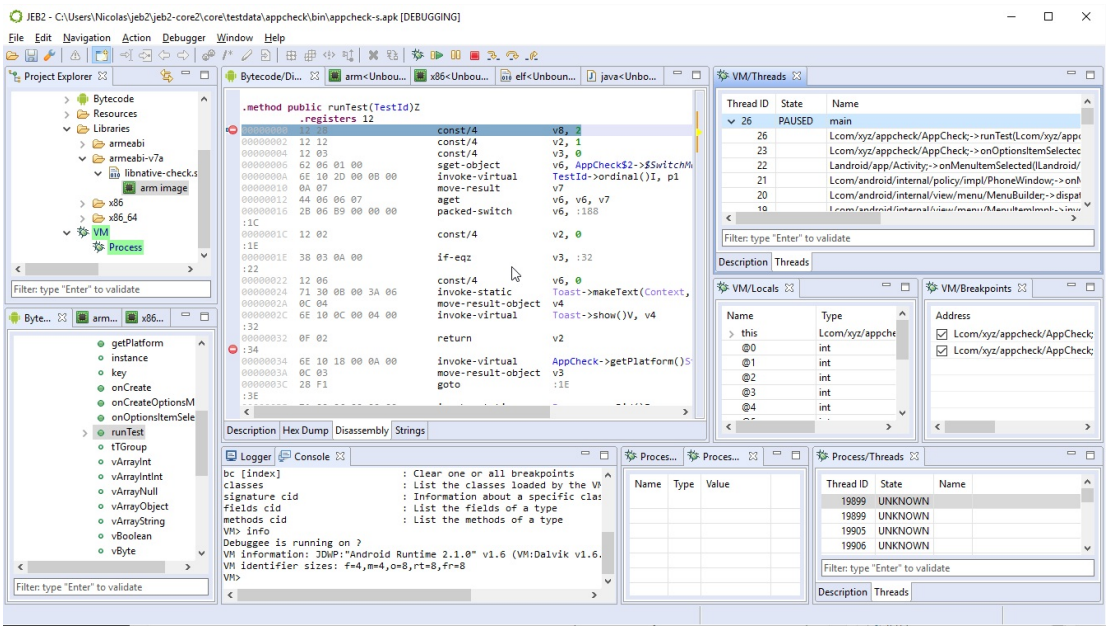
- 

JEB动态调试

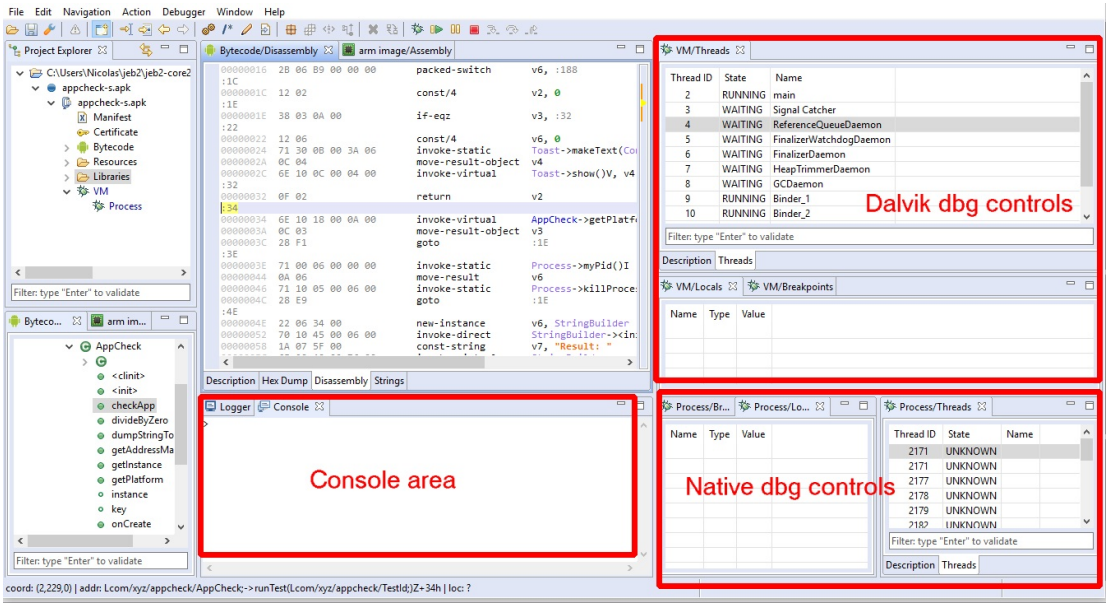
先贴出一些：

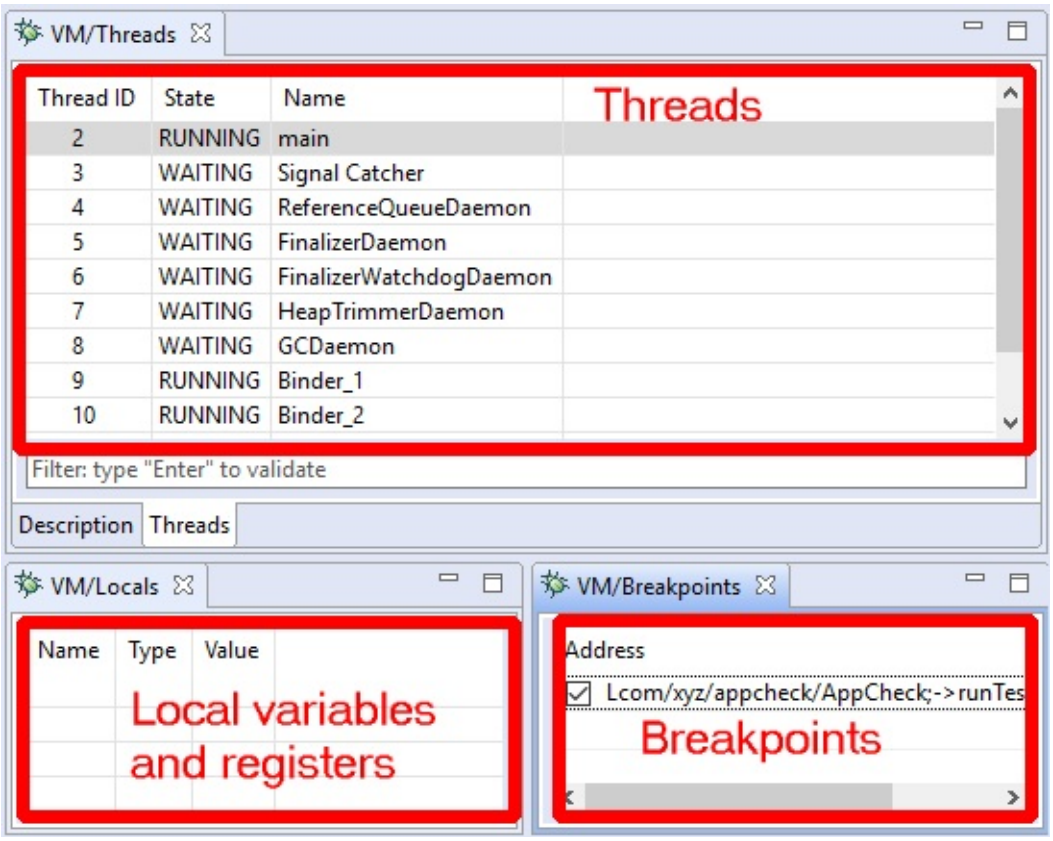
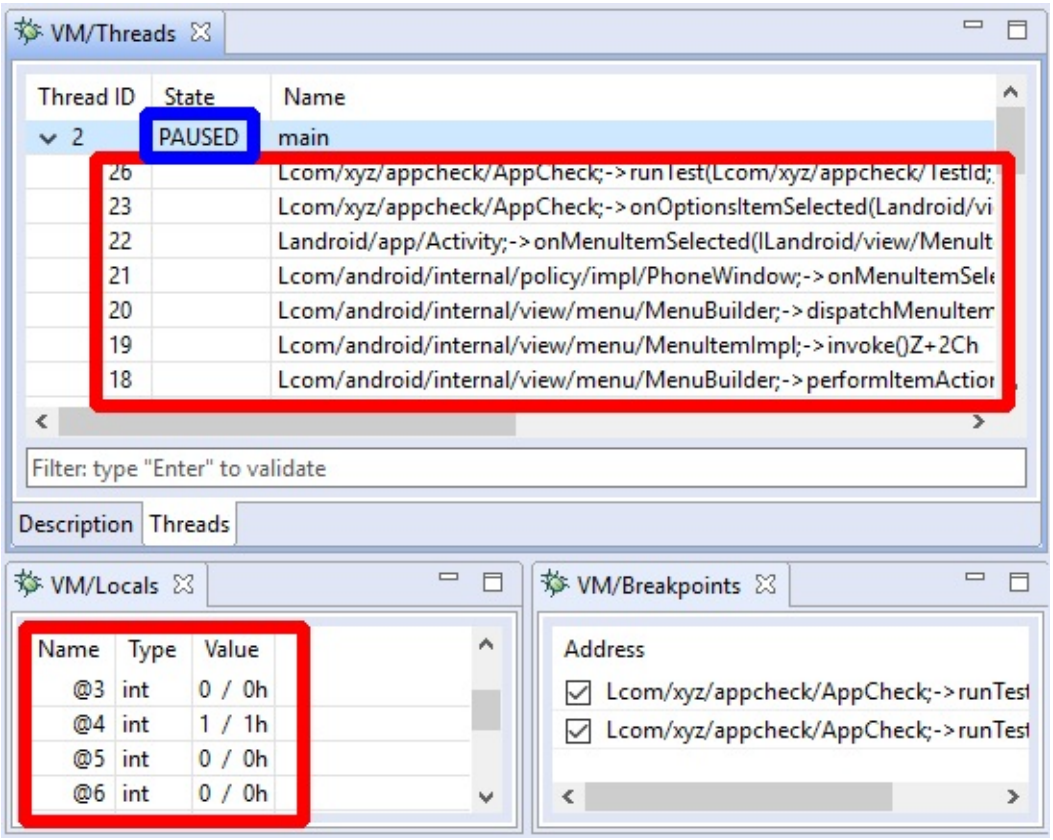
- JEB动态调试
  - 相关截图
    - 

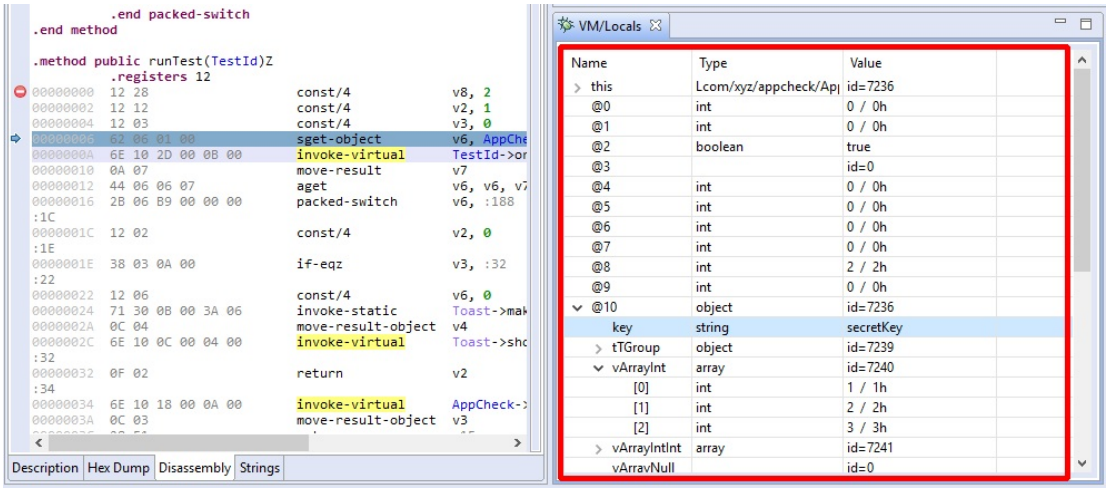
    - 

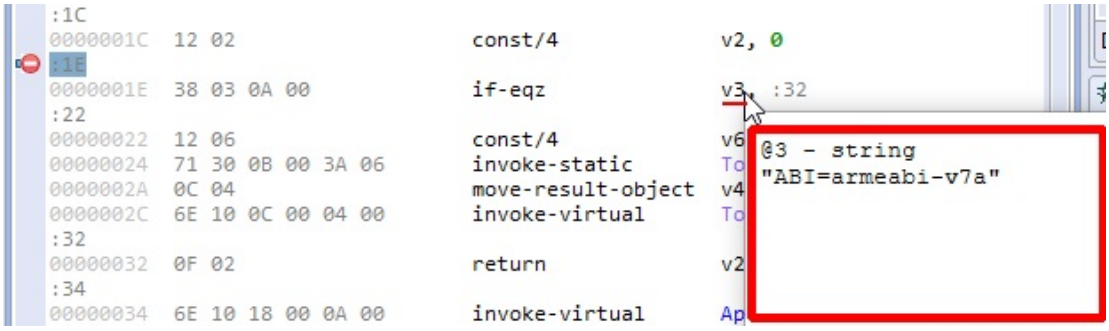


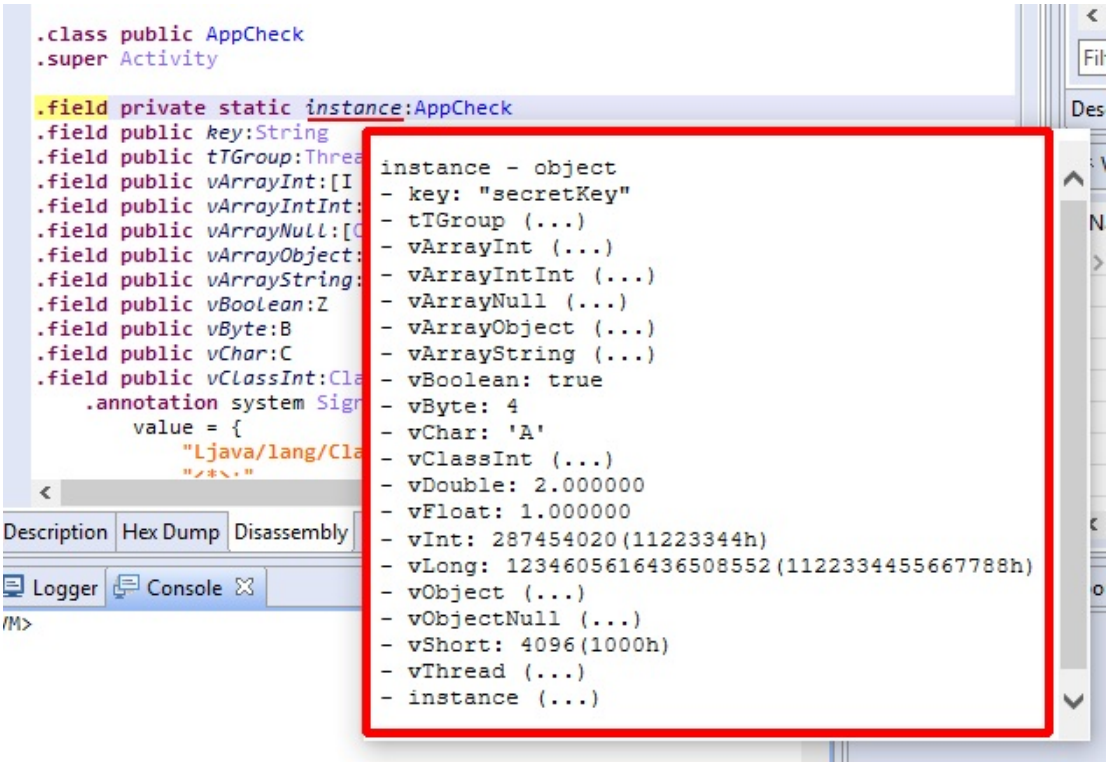
- 调试界面功能模块介绍



- 
VM/Threads
Thread ID
State
Name
Threads
2 RUNNING main
3 WAITING Signal Catcher
4 WAITING ReferenceQueueDaemon
5 WAITING FinalizerDaemon
6 WAITING FinalizerWatchdogDaemon
7 WAITING HeapTrimmerDaemon
8 WAITING GCDaemon
9 RUNNING Binder_1
10 RUNNING Binder_2
Filter: type "Enter" to validate
Description
Threads
VM/Locals
Name
Type
Value
Local variables and registers
VM/Breakpoints
Address
Lcom/xyz/appcheck/AppCheck;->runTes
Breakpoints
- 
VM/Threads
Thread ID
State
Name
2 PAUSED main
26 Lcom/xyz/appcheck/AppCheck;->runTest(Lcom/xyz/appcheck/TestId;
23 Lcom/xyz/appcheck/AppCheck;->onOptionsItemSelected(Landroid/vi
22 Landroid/app/Activity;->onMenuItemSelected(ILandroid/view/Menult
21 Lcom/android/internal/policy/impl/PhoneWindow;->onMenuItemSele
20 Lcom/android/internal/view/menu/MenuBuilder;->dispatchMenuItem
19 Lcom/android/internal/view/menu/MenuItemImpl;->invoke()Z+2Ch
18 Lcom/android/internal/view/menu/MenuBuilder;->performItemActio
Filter: type "Enter" to validate
Description
Threads
VM/Locals
Name
Type
Value
@3 int 0 / 0h
@4 int 1 / 1h
@5 int 0 / 0h
@6 int 0 / 0h
VM/Breakpoints
Address
Lcom/xyz/appcheck/AppCheck;->runTest
Lcom/xyz/appcheck/AppCheck;->runTest

- 

- 

- 

JEB使用心得

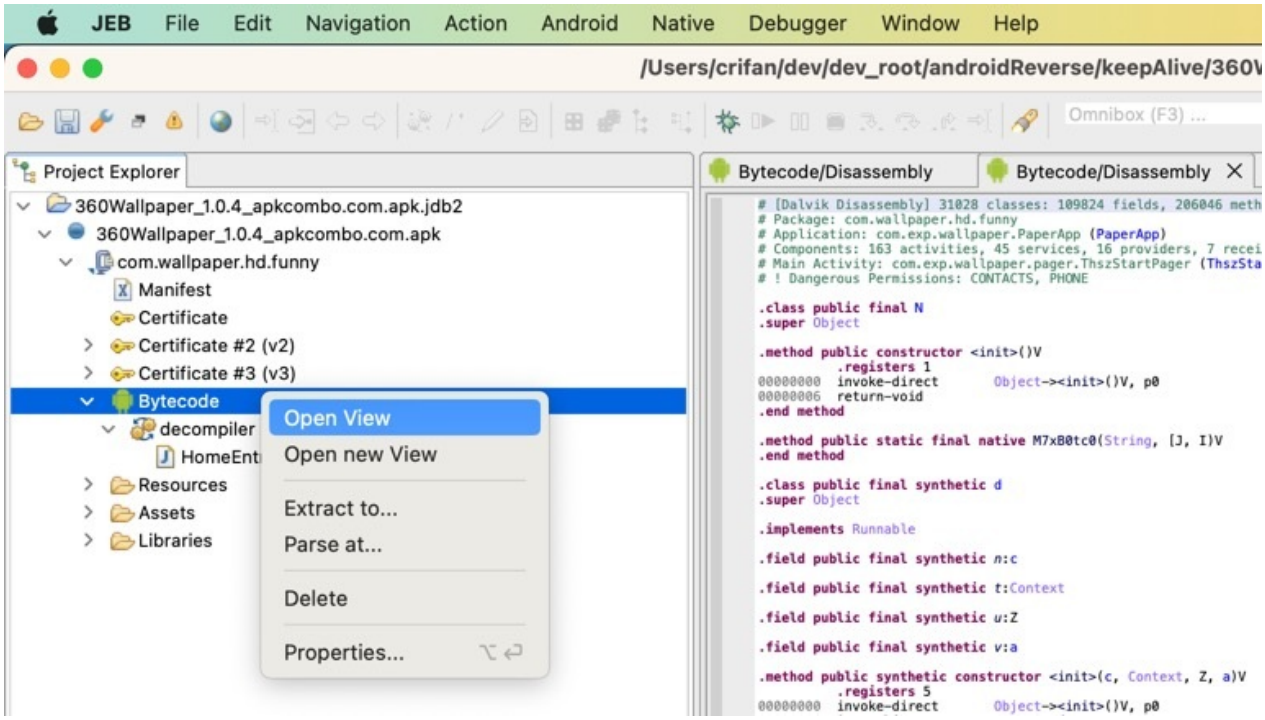
此处整理JEB使用心得。

页面显示心得

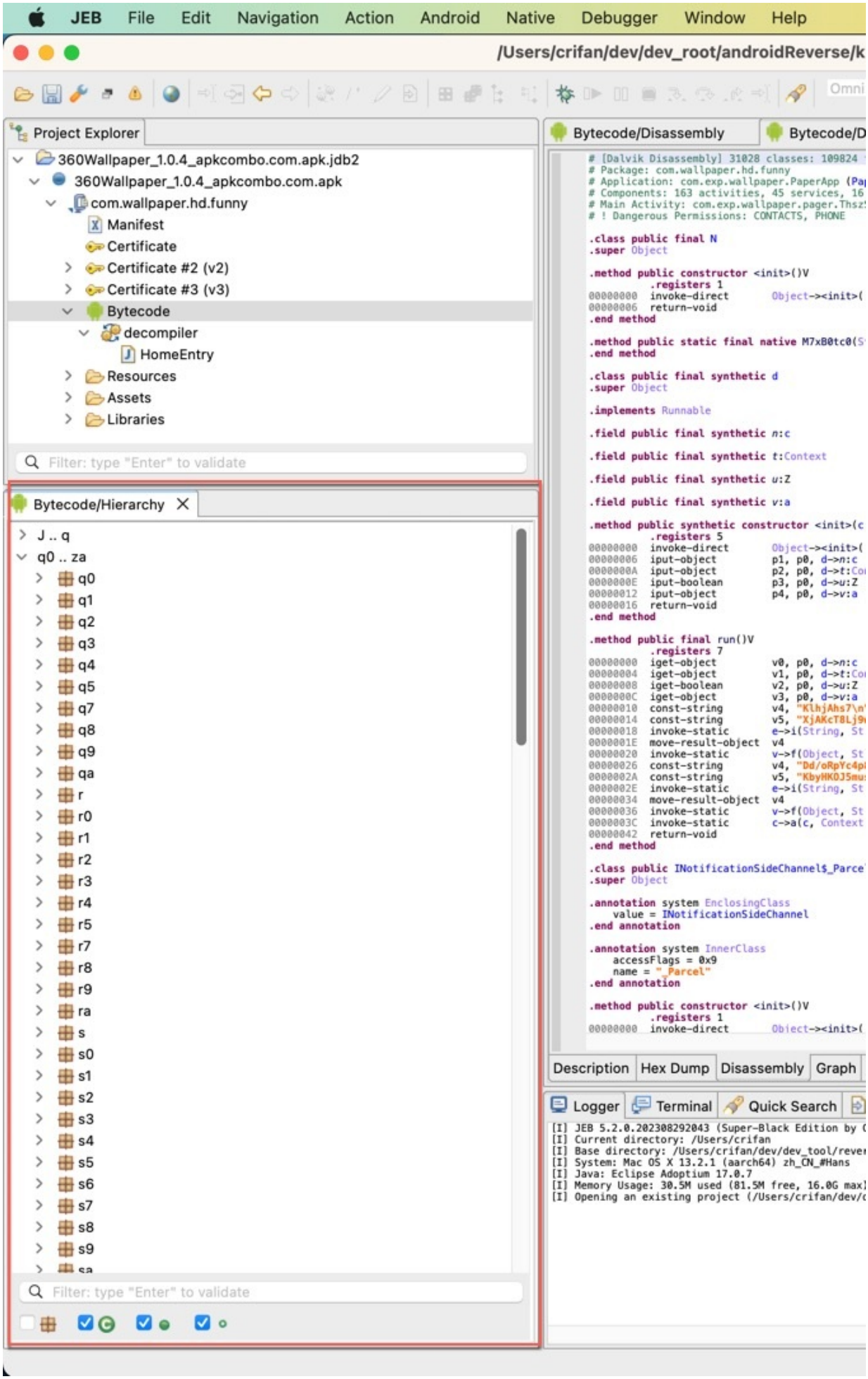
关闭Bytecode/Hierarchy窗口后，如何恢复显示

当关闭 `Bytecode/Hierarchy` 窗口后，想要重新打开

具体步骤： `JEB` -> `Project Explorer` ->展开apk列表-> `Bytecode` ->右键-> `Open View`



即可重新看到，左下方显示我们要的： `Bytecode/Hierarchy` 窗口：


JEB File Edit Navigation Action Android Native Debugger Window Help
/Users/crifan/dev/dev_root/androidReverse/k
Project Explorer
360Wallpaper_1.0.4_apkcombo.com.apk.jdb2
360Wallpaper_1.0.4_apkcombo.com.apk
com.wallpaper.hd.funny
Manifest
Certificate
Certificate #2 (v2)
Certificate #3 (v3)
Bytecode
decompiler
HomeEntry
Resources
Assets
Libraries
Filter: type "Enter" to validate
Bytecode/Hierarchy
J .. q
q0 .. za
q0
q1
q2
q3
q4
q5
q7
q8
q9
qa
r
r0
r1
r2
r3
r4
r5
r7
r8
r9
ra
s
s0
s1
s2
s3
s4
s5
s6
s7
s8
s9
Filter: type "Enter" to validate
Bytecode/Disassembly
Description
Hex Dump
Disassembly
Graph
Logger
Terminal
Quick Search

反编译java代码心得

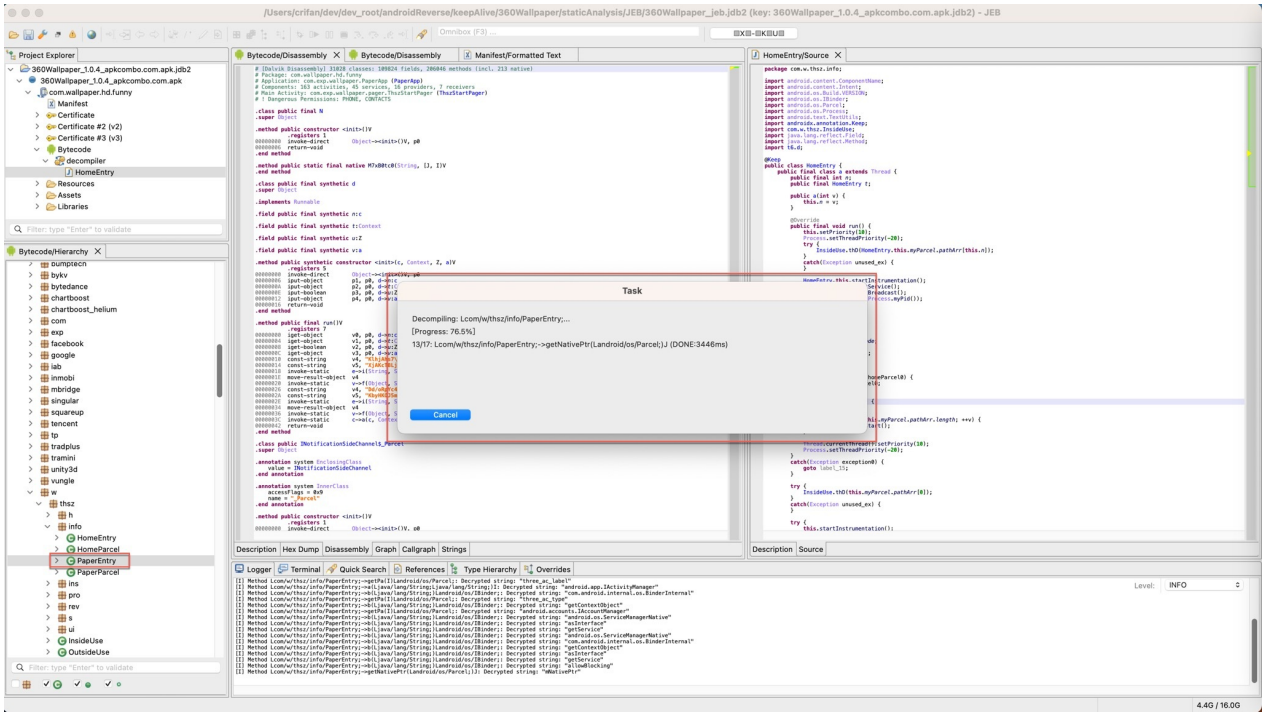
单个类（文件）的反编译java代码

反编译单个类的java代码

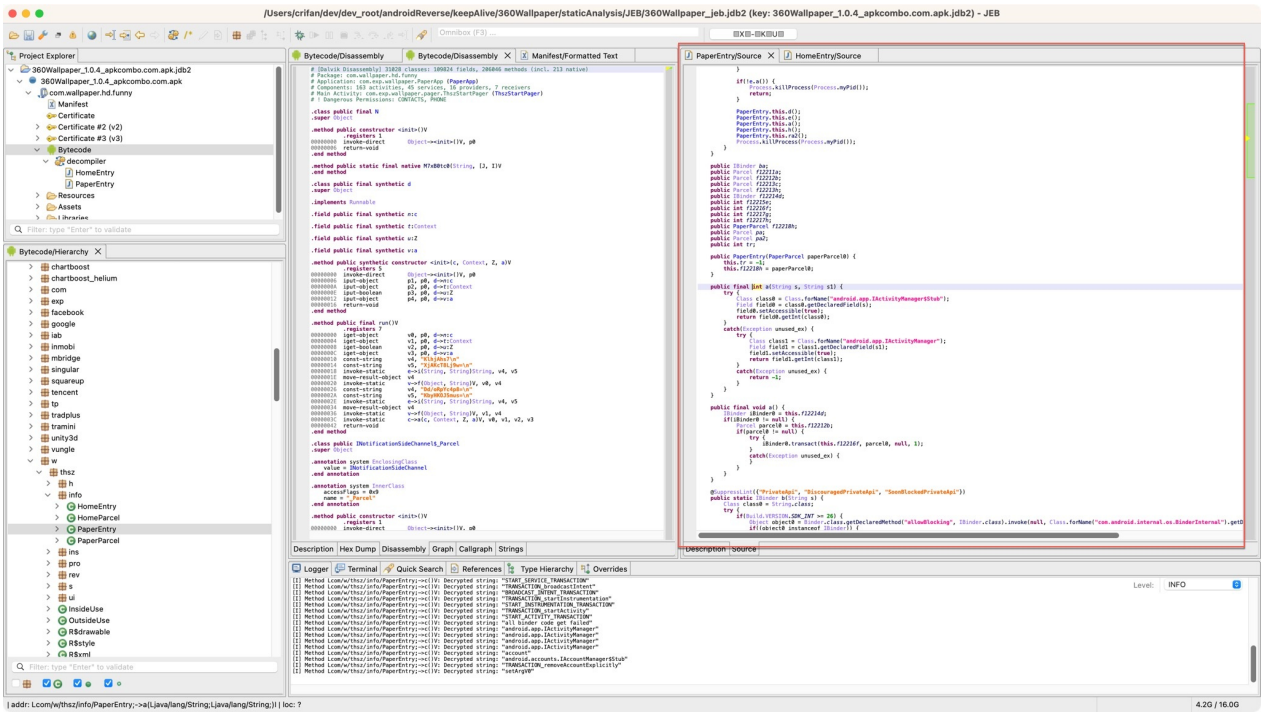
双击Bytecode中对应的类可以自动反编译

对于Bytecode窗口中，对于某个类（`com.w.thsz.info.PaperEntry`，双击后，等价于`右键 -> Decompile`

会弹框提示开始反编译：



反编译完成后，此处右上角显示出反编译后的java源码：

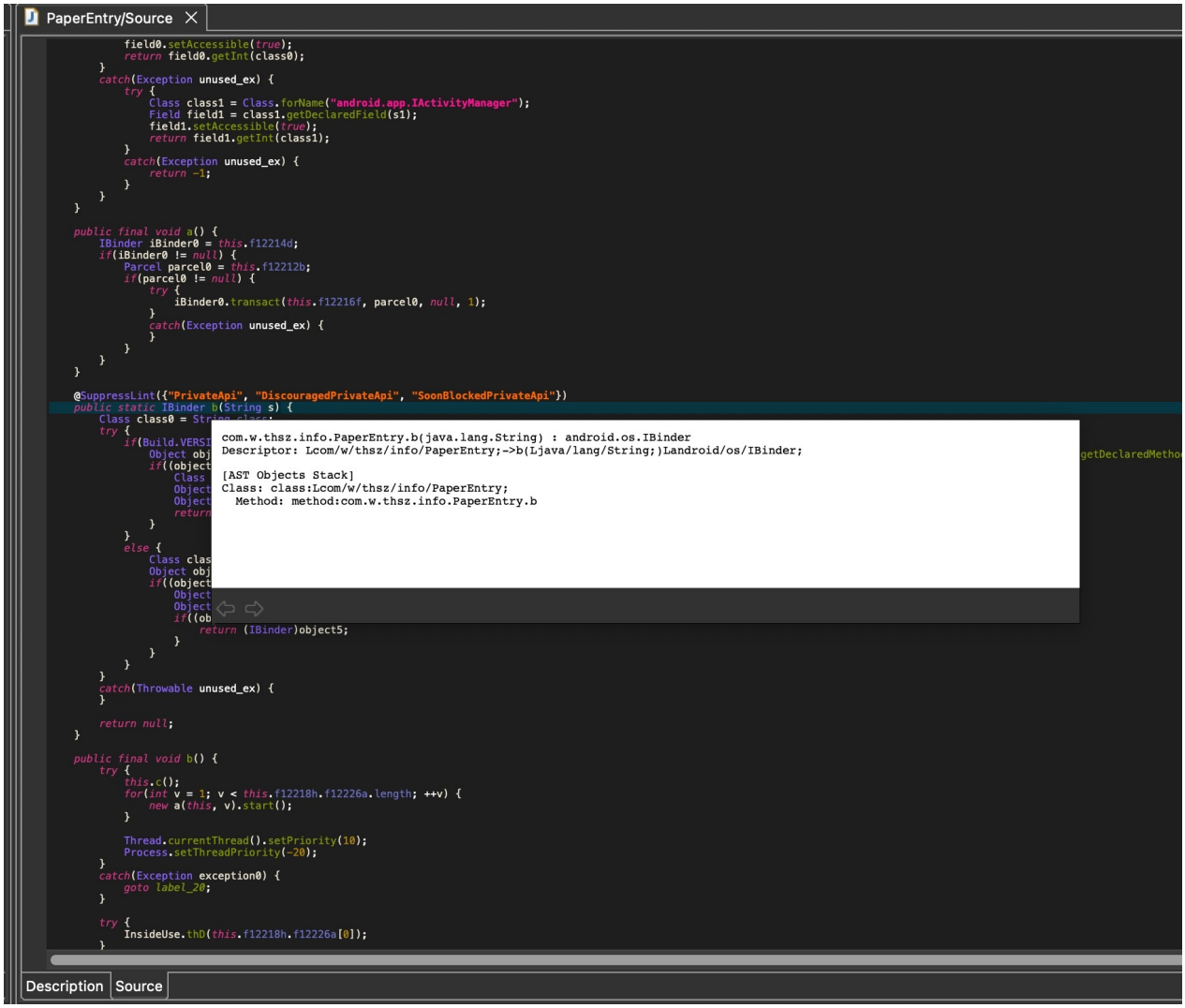


反编译后的单个类的java代码

java代码支持动态提示详情

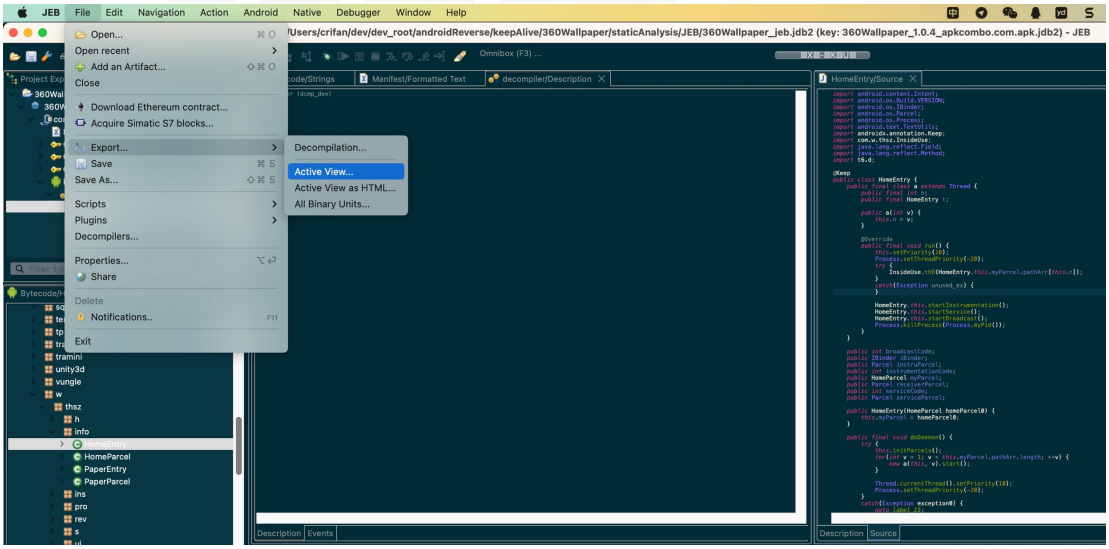
反编译后的java代码窗口中，鼠标移动到某个内容上，比如类，函数等，此时可以：显示出对应内容的详情

举例：鼠标移动到某函数上，可以显示出函数的详情：



导出当前反编译代码

JEB支持，导出当前视图View的内容：

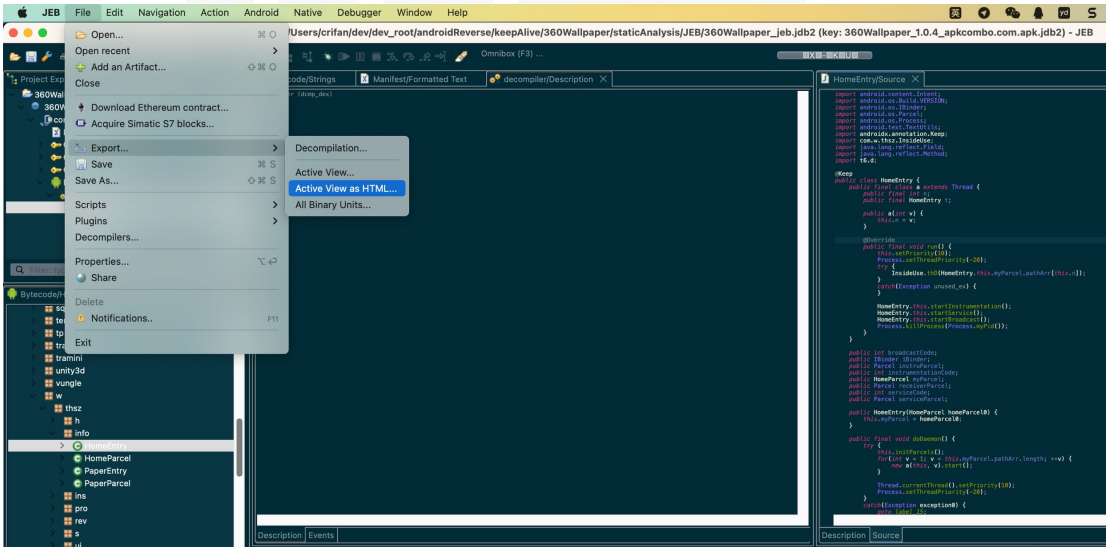
- 确保当前视图（鼠标所在位置是）反编译代码窗口，然后
  - 导出为普通文本： `JEB -> File -> Export -> Active View`
    

    - ->导出效果：普通文本文件

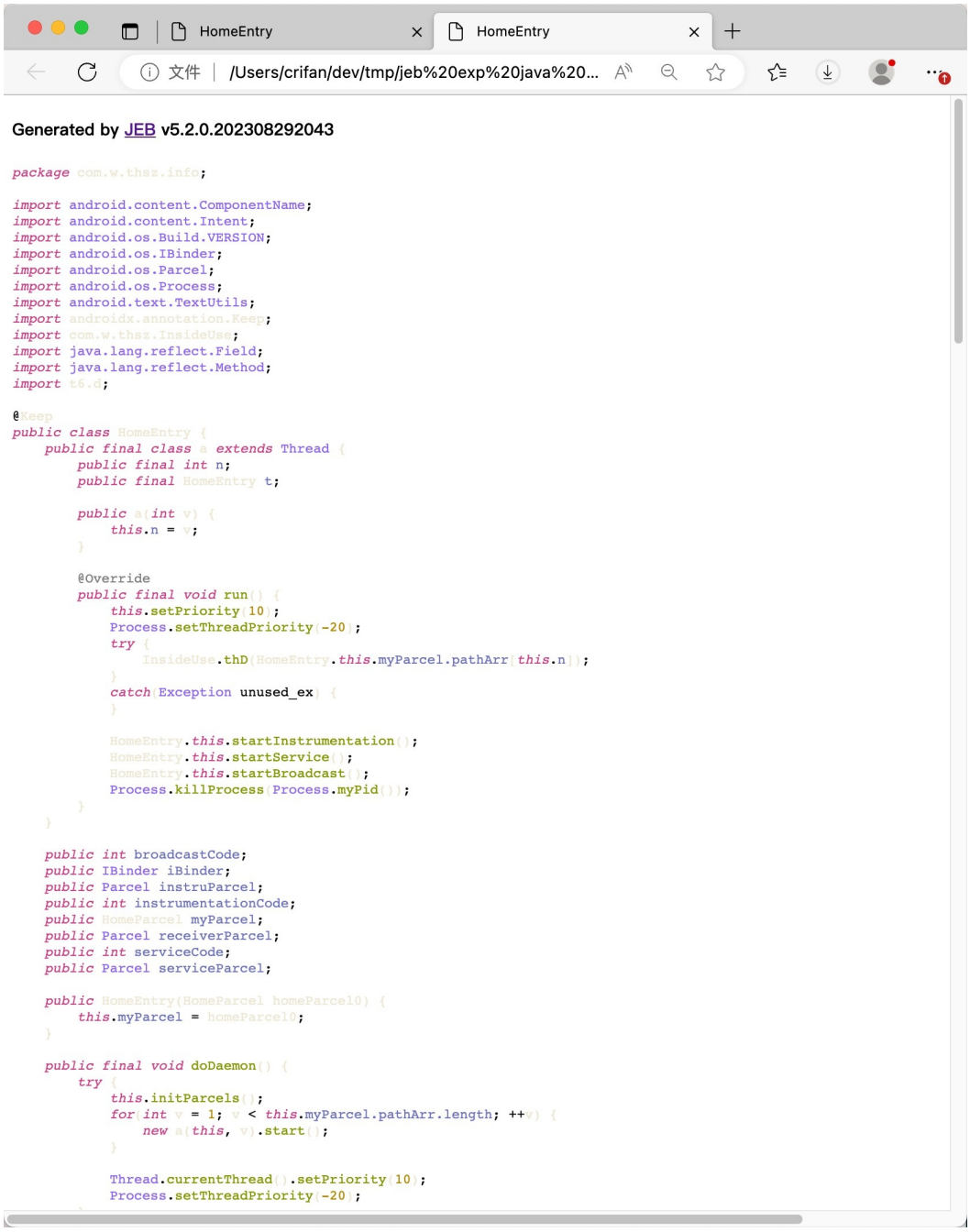
jeb exp java.txt　　使用“文本编辑”打开

```
package com.w.thsz.info;

import android.content.ComponentName;
import android.content.Intent;
import android.os.Build.VERSION;
import android.os.IBinder;
import android.os.Parcel;
import android.os.Process;
import android.text.TextUtils;
import androidx.annotation.Keep;
import com.w.thsz.InsideUse;
import java.lang.reflect.Field;
import java.lang.reflect.Method;
import t6.d;

@Keep
public class HomeEntry {
    public final class a extends Thread {
        public final int n;
        public final HomeEntry t;

        public a(int v) {
            this.n = v;
        }

        @Override
        public final void run() {
            this.setPriority(10);
            Process.setThreadPriority(-20);
            try {
                InsideUse.thD(HomeEntry.this.myParcel.pathArr[this.n]);
            }
            catch(Exception unused_ex) {
            }

            HomeEntry.this.startInstrumentation();
            HomeEntry.this.startService();
            HomeEntry.this.startBroadcast();
            Process.killProcess(Process.myPid());
        }
    }

    public int broadcastCode;
    public IBinder iBinder;
    public Parcel instruParcel;
    public int instrumentationCode;
    public HomeParcel myParcel;
    public Parcel receiverParcel;
    public int serviceCode;
    public Parcel serviceParcel;

    public HomeEntry(HomeParcel homeParcel0) {
        this.myParcel = homeParcel0;
    }

    public final void doDaemon() {
        try {
            this.initParcels();
            for(int v = 1; v < this.myParcel.pathArr.length; ++v) {
                new a(this, v).start();
            }
```

- 导出为HTML网页： `JEB -> File -> Export -> Active View as HTML`



- -> 导出效果：html网页，用浏览器打开后的效果

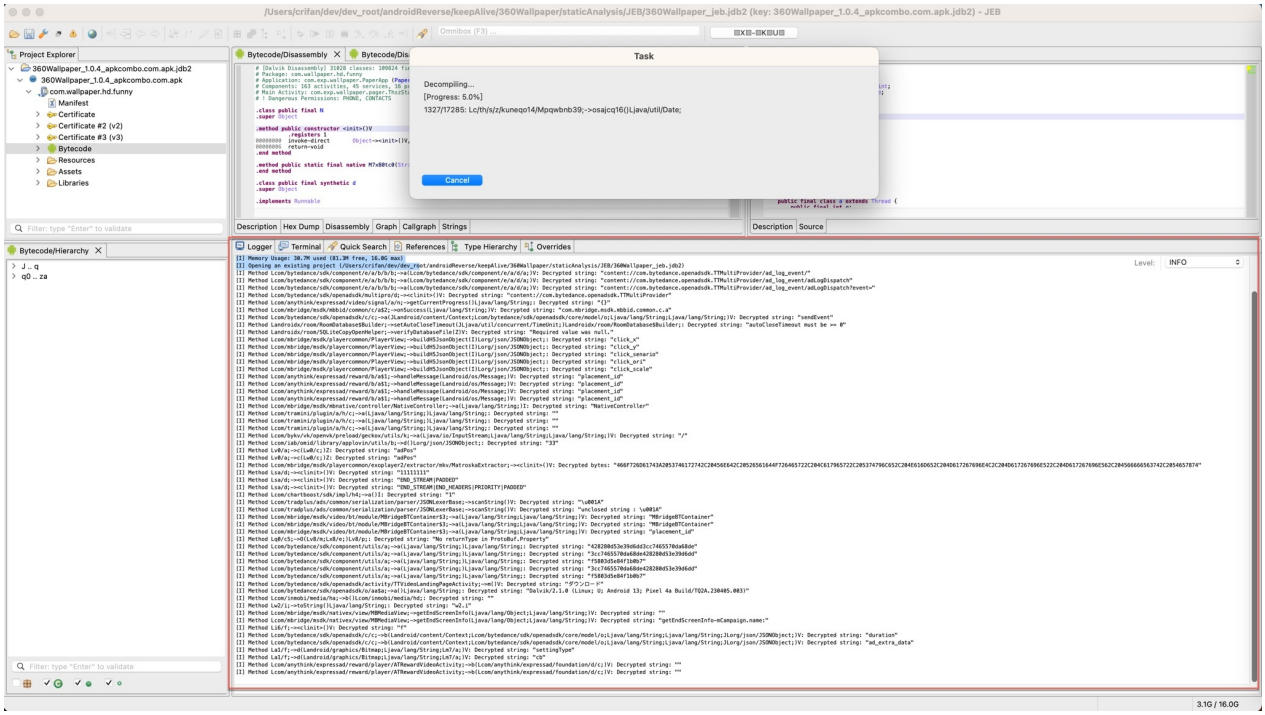


全部所有（的类）的反编译java代码

查看（导出）所有反编译的log日志

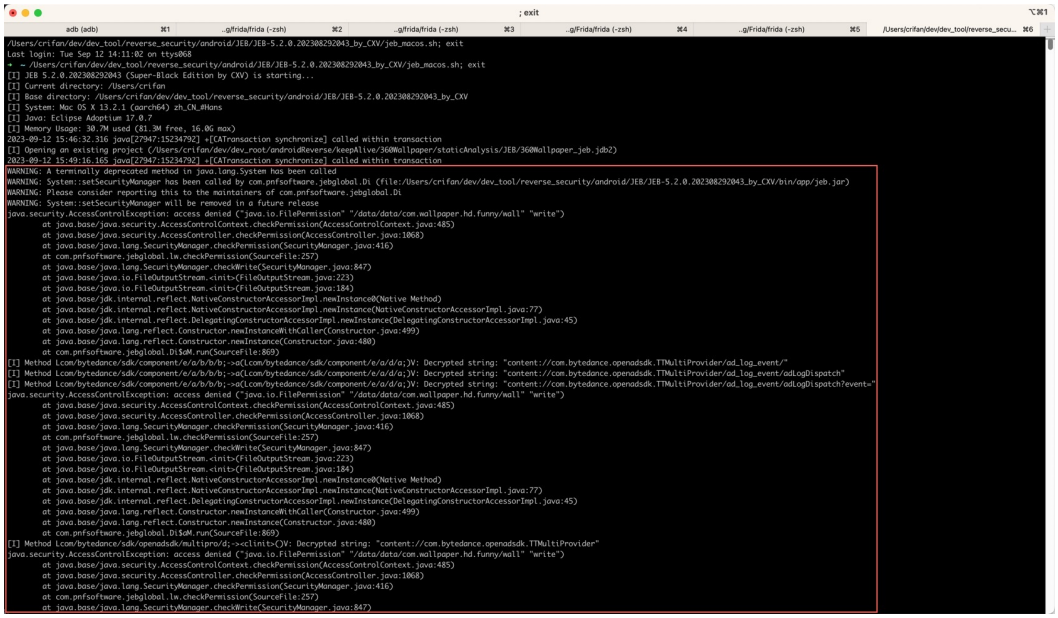
在批量反编译全部代码期间，对于想要查看反编译期间的日志：

是可以通过JEB底部的Logger的tab页，查看到对应日志的：

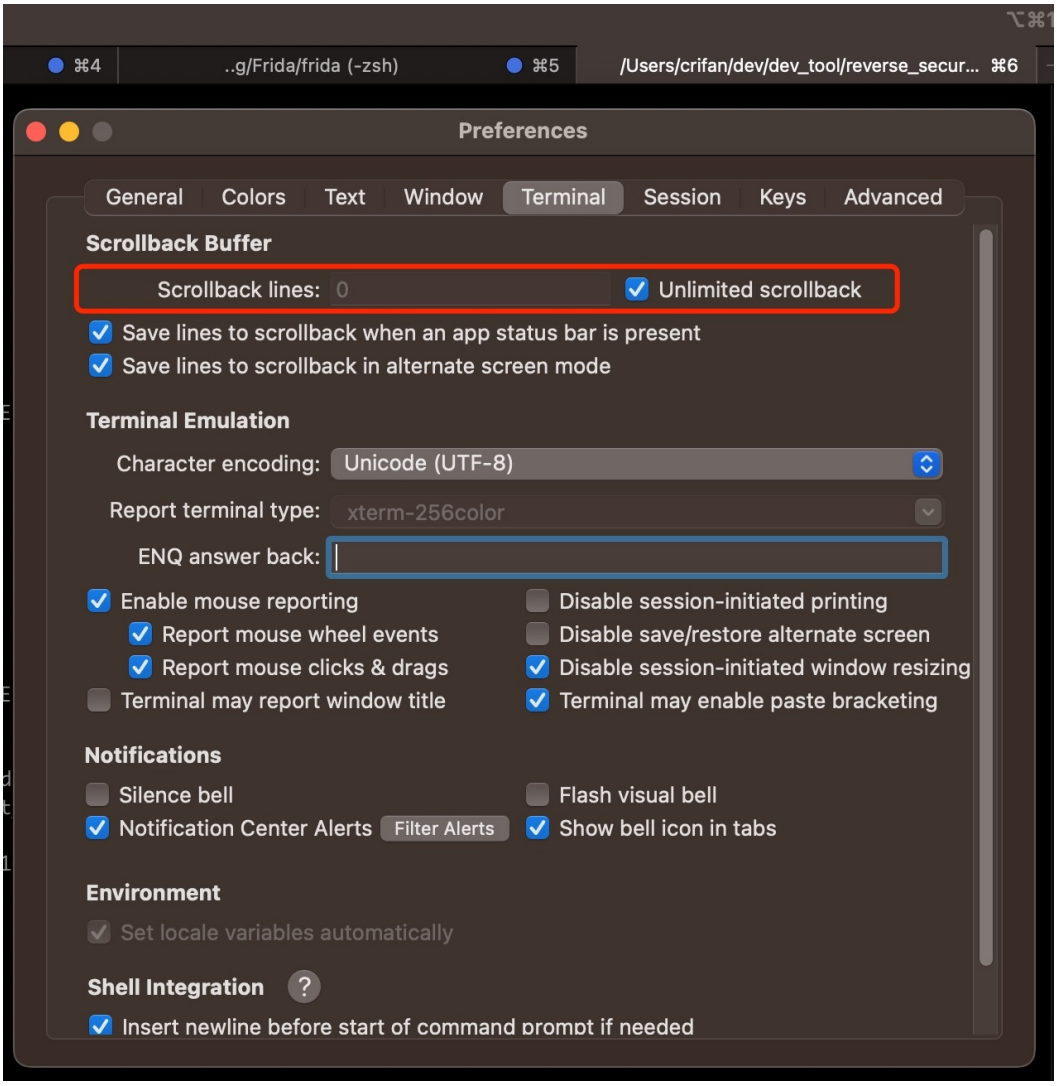


但是实测发现，最多只能保留2000行左右的日志。更多的日志，会被冲掉，无法保留。

- 想要查看=保留=导出全部反编译的日志
  - 背景=前提：此处Mac中通过iTerm2中运行 `jeb_macos.sh` 而启动JEB
  - 办法：
    - 发现终端中（和Logger中同步）也会显示反编译期间的log日志



    - 所以可以去给iTerm2的buffer设置足够大或无限大小

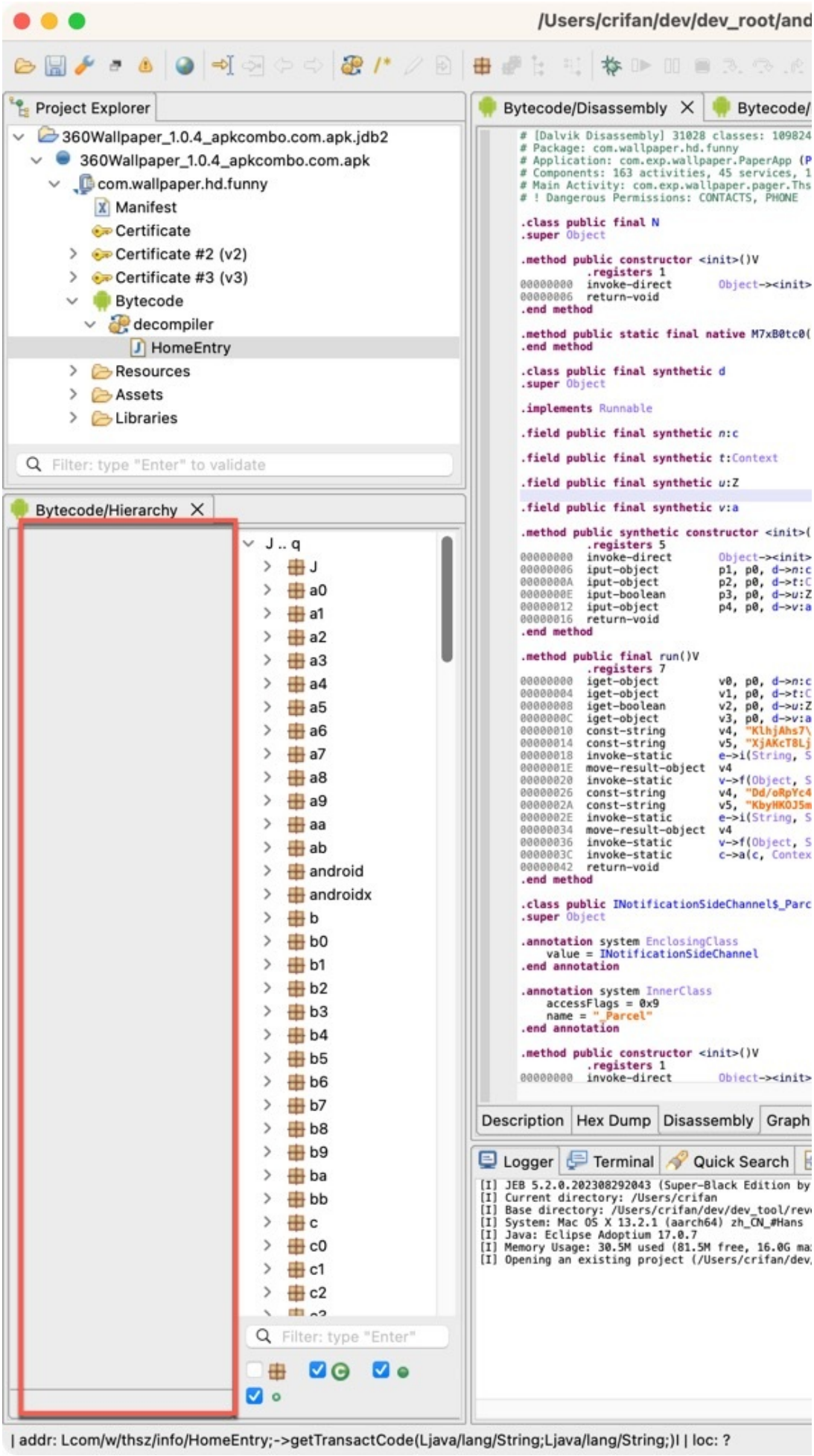


- 即可显示=保存出全部反编译期间的所有的log日志
  - 后续去选择并拷贝，即可导出全部日志，供后续分析研究用

JEB常见问题

Bytecode/Hierarchy窗口出现多余的一列

- 问题：重新打开加载之前的jdb2项目后，往往会看到：`Bytecode/Hierarchy`窗口有多余的一列
  。


/Users/crifan/dev/dev_root/and
Project Explorer
360Wallpaper_1.0.4_apkcombo.com.apk.jdb2
360Wallpaper_1.0.4_apkcombo.com.apk
com.wallpaper.hd.funny
Manifest
Certificate
Certificate #2 (v2)
Certificate #3 (v3)
Bytecode
decompiler
HomeEntry
Resources
Assets
Libraries
Filter: type "Enter" to validate
Bytecode/Hierarchy
J .. q
J
a0
a1
a2
a3
a4
a5
a6
a7
a8
a9
aa
ab
android
androidx
b
b0
b1
b2
b3
b4
b5
b6
b7
b8
b9
ba
bb
c
c0
c1
c2
Filter: type "Enter"
Bytecode/Disassembly
Bytecode/
[Dalvik Disassembly] 31028 classes: 109824
Package: com.wallpaper.hd.funny
Application: com.exp.wallpaper.PaperApp (P
Components: 163 activities, 45 services, 1
Main Activity: com.exp.wallpaper.pager.Ths
! Dangerous Permissions: CONTACTS, PHONE
.class public final N
.super Object
.method public constructor <init>()V
.registers 1
00000000 invoke-direct Object-><init>
00000006 return-void
.end method
.method public static final native M7xB0tc0(
.end method
.class public final synthetic d
.super Object
.implements Runnable
.field public final synthetic n:c
.field public final synthetic t:Context
.field public final synthetic u:Z
.field public final synthetic v:a
.method public synthetic constructor <init>(
.registers 5
00000000 invoke-direct Object-><init>
00000006 iput-object p1, p0, d->n:c
0000000A iput-object p2, p0, d->t:C
0000000E iput-boolean p3, p0, d->u:Z
00000012 iput-object p4, p0, d->v:a
00000016 return-void
.end method
.method public final run()V
.registers 7
00000000 iget-object v0, p0, d->n:c
00000004 iget-object v1, p0, d->t:C
00000008 iget-boolean v2, p0, d->u:Z
0000000C iget-object v3, p0, d->v:a
00000010 const-string v4, "KlhjAhs7\
00000014 const-string v5, "XjAKcT8Lj
00000018 invoke-static e->i(String, S
0000001E move-result-object v4
00000020 invoke-static v->f(Object, S
00000026 const-string v4, "Dd/oRpYc4
0000002A const-string v5, "KbyHKOJ5m
0000002E invoke-static e->i(String, S
00000034 move-result-object v4
00000036 invoke-static v->f(Object, S
0000003C invoke-static c->a(c, Contex
00000042 return-void
.end method
.class public INotificationSideChannel$Parc
.super Object
.annotation system EnclosingClass
value = INotificationSideChannel
.end annotation
.annotation system InnerClass
accessFlags = 0x9
name = "_Parcel"
.end annotation
.method public constructor <init>()V
.registers 1
00000000 invoke-direct Object-><init>
Description
Hex Dump
Disassembly
Graph
Logger
Terminal
Quick Search
[I] JEB 5.2.0.202308292043 (Super-Black Edition by
[I] Current directory: /Users/crifan
[I] Base directory: /Users/crifan/dev/dev_tool/rev
[I] System: Mac OS X 13.2.1 (aarch64) zh_CN_#Hans
[I] Java: Eclipse Adoptium 17.0.7
[I] Memory Usage: 30.5M used (81.5M free, 16.0G ma
[I] Opening an existing project (/Users/crifan/dev,
| addr: Lcom/w/thsz/info/HomeEntry;->getTransactCode(Ljava/lang/String;Ljava/lang/String;)I | loc: ?

- 解决办法：关闭后重新打开 `Bytecode/Hierarchy` 窗口
- 具体步骤：
  - 概述： `Project Explorer` 中apk展开列表中的 `Bytecode` 右键选 `Open View`
  - 详解：页面显示心得 中的 `关闭Bytecode/Hierarchy窗口后，如何恢复显示`

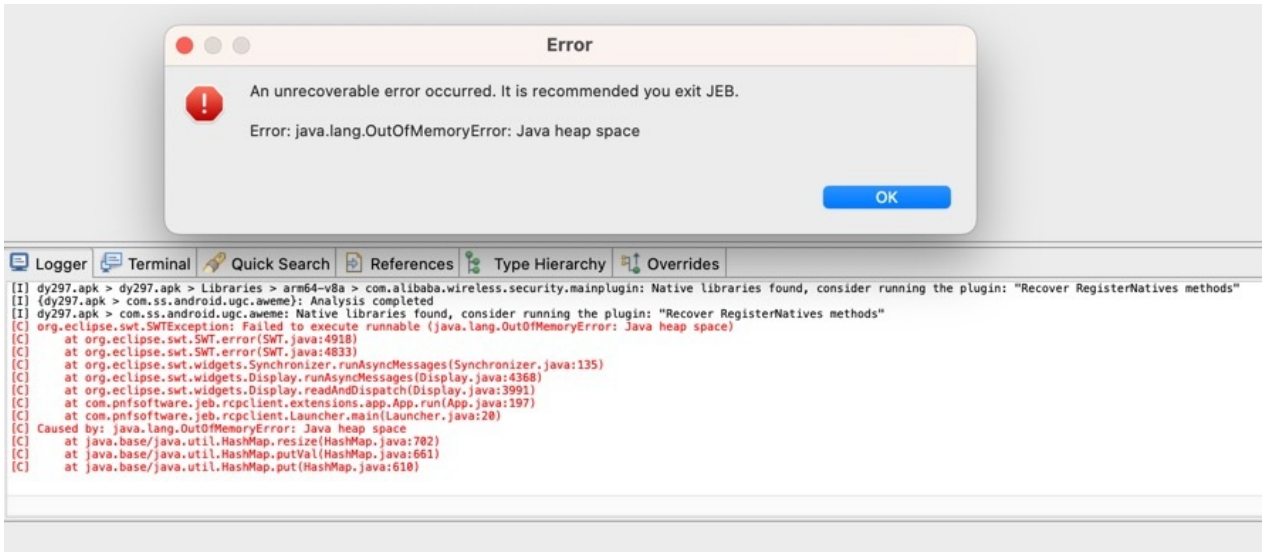
apk解析问题

JEB导入apk解析过程期间，也会遇到一些常见问题。

Error: java.lang.OutOfMemoryError: Java heap space

- 现象：JEB导入apk后，开始解析，最后报错

```
[I] {dy297.apk > com.ss.android.ugc.aweme}: Analysis completed
[I] dy297.apk > com.ss.android.ugc.aweme: Native libraries found, consider running the
plugin: "Recover RegisterNatives methods"
[C] org.eclipse.swt.SWTException: Failed to execute runnable (java.lang.OutOfMemoryErro
r: Java heap space)
[C]     at org.eclipse.swt.SWT.error(SWT.java:4918)
[C]     at org.eclipse.swt.SWT.error(SWT.java:4833)
[C]     at org.eclipse.swt.widgets.Synchronizer.runAsyncMessages(Synchronizer.java:135)
[C]     at org.eclipse.swt.widgets.Display.runAsyncMessages(Display.java:4368)
[C]     at org.eclipse.swt.widgets.Display.readAndDispatch(Display.java:3991)
[C]     at com.pnfsoftware.jeb.rcpclient.extensions.app.App.run(App.java:197)
[C]     at com.pnfsoftware.jeb.rcpclient.Launcher.main(Launcher.java:20)
[C] Caused by: java.lang.OutOfMemoryError: Java heap space
[C]     at java.base/java.util.HashMap.resize(HashMap.java:702)
[C]     at java.base/java.util.HashMap.putVal(HashMap.java:661)
[C]     at java.base/java.util.HashMap.put(HashMap.java:610)
```



- 原因：Java的JVM的内存不够用了，超过最大允许的内存，而崩溃
- 解决办法：增加最大内存
- 具体步骤

给此处的JEB的脚本

JEB/JEB-5.2.0.202308292043_by_CXV/jeb_macos.sh

改为：

```
# Manual override for JVM options: run the script with -j, uncomment the following line
 and adjust if necessary
# JVMOPT="-Xss4M -Xmx8G"
JVMOPT="-Xss4M -Xmx48G"
```

意思是：

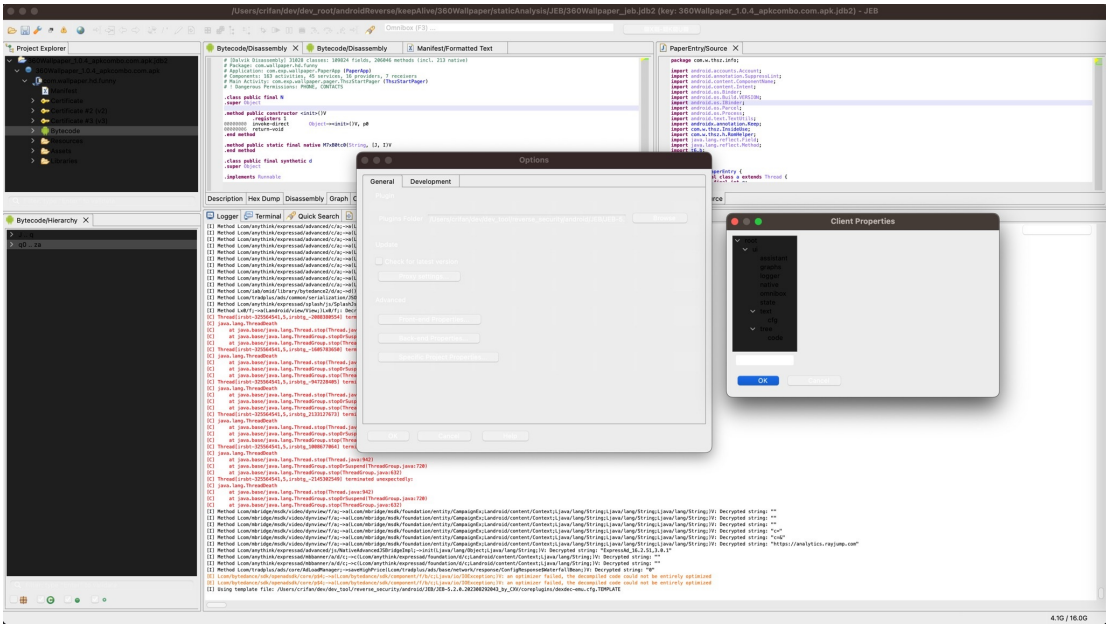
- `-Xss4M`
  - 设置 Java 线程堆栈大小：`4MB`
- `-Xmx48G`
  - 设置最大 Java 堆大小：`48GB`

即：给Java的JVM虚拟机的最大内存是48G（此处是Mac M2 Max，总内存是64GB）

如此确保有足够的内存可用，避免了内存不够用的报错

页面显示问题

白天和夜晚模式切换后，界面显示不正常

- 问题：Mac中，从白天进入夜晚（或者反过来，从夜晚过了一夜到了第二天白天），白天和夜晚模式切换后，所有页面全都显示异常：
  - Project Explorer、Bytecode/Hierarchy，Preferences设置页面等窗口，都无法看清内容
    - 
- 原因：JEB的bug
- 解决办法：
  - 办法1：重启JEB
  - 办法2：多主题显示效果中的 `切换主题`
    - 通过切换主题，即可消除显示异常的部分，起到刷新显示的效果
- 结果：即可正常显示内容
  -

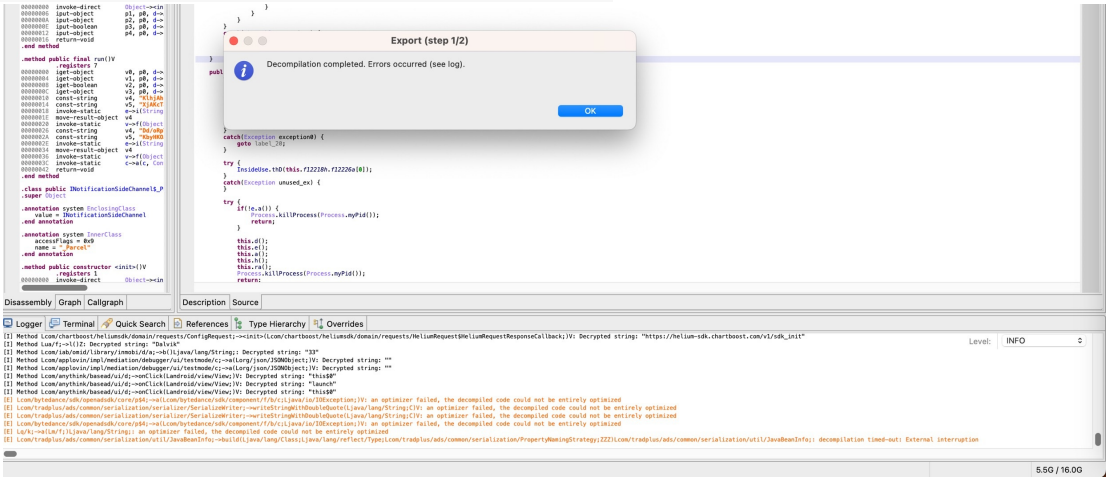
反编译java代码问题

此处整理，JEB反编译java代码方面的问题：

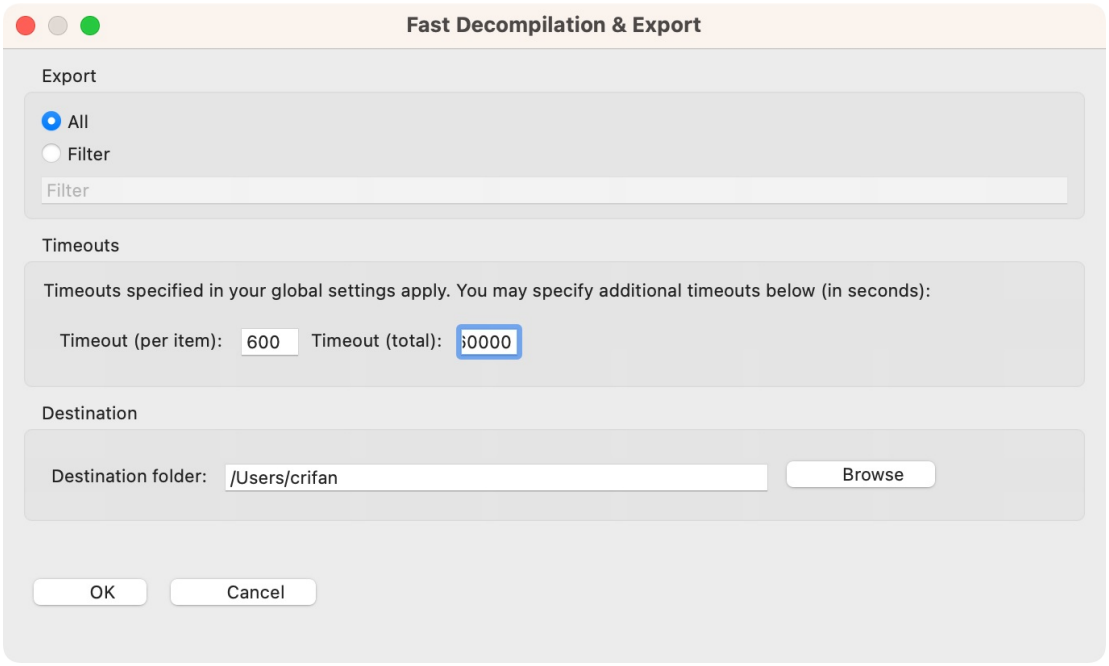
批量反编译全部java代码

用默认超时设置会导致最后由于超时而被中断停止

- 问题：在批量反编译全部代码最后报错
  - 弹框：`Decompilation completed. Errors occurred` + Logger日志中显示错误原因：`decompilation timed-out: External interruption`



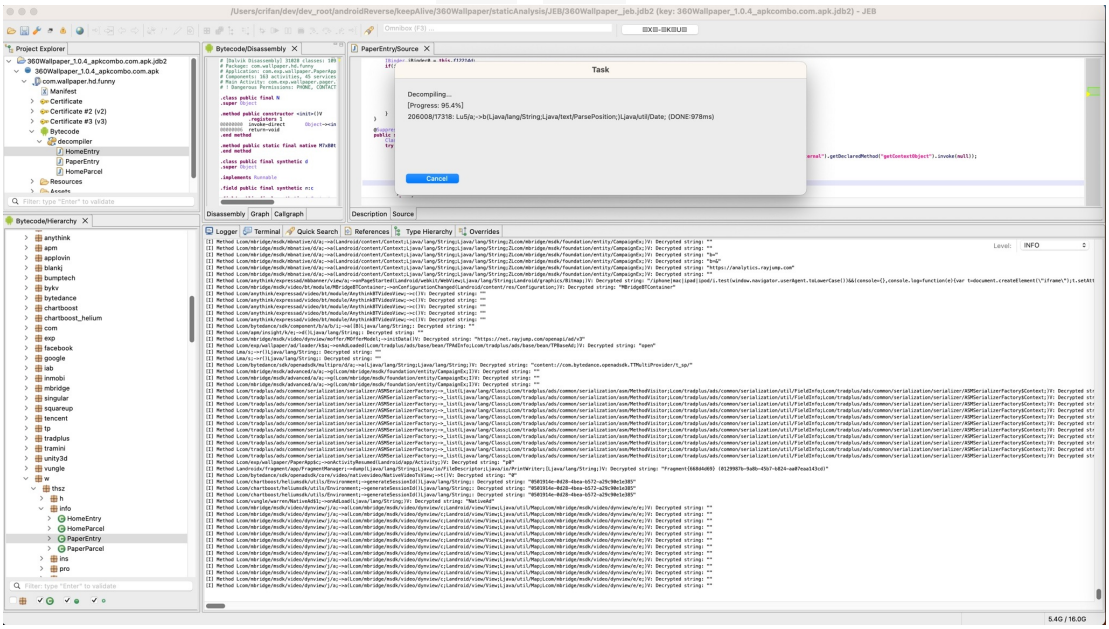
- 原因：JEB默认的超时设置太短了：
  - Timeout per item: `60` =1分钟
  - Timeout total: `600` =10分钟

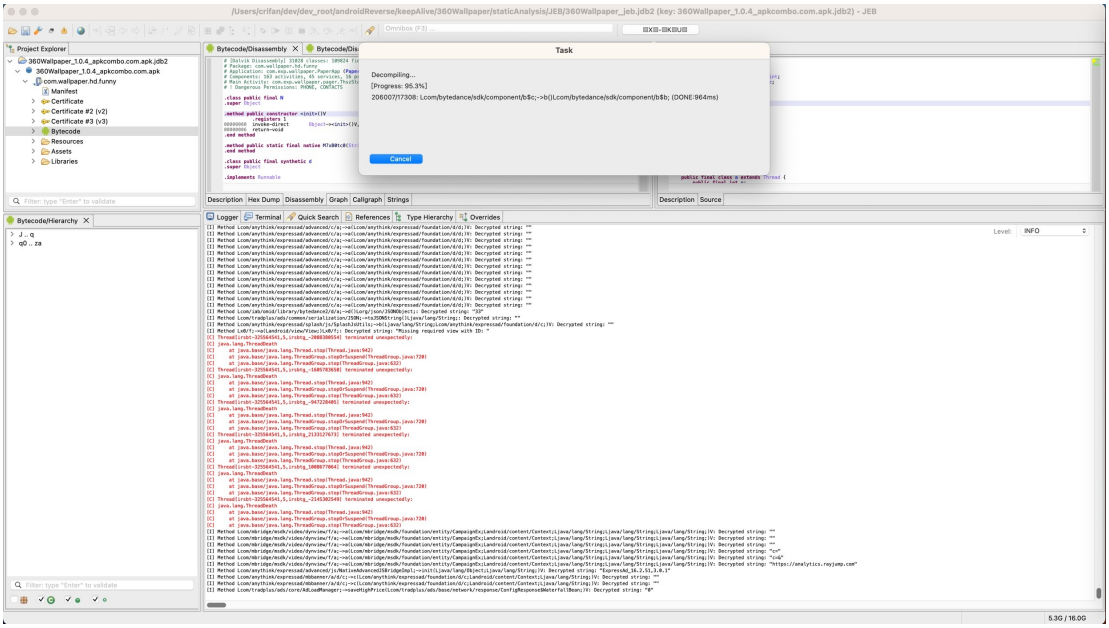
    - 而此处要反编译的内容较多，耗时比较久，超过了10分钟的总时长超时限制，因此中断退出
- 解决办法：增大超时方面的设置
    - 比如
        - 

            - Timeouts:
                - Timeout per item: `600` =10分钟
                - Timeout total: `60000` =100分钟

最后会卡死和多次的反编译输出结果不稳定

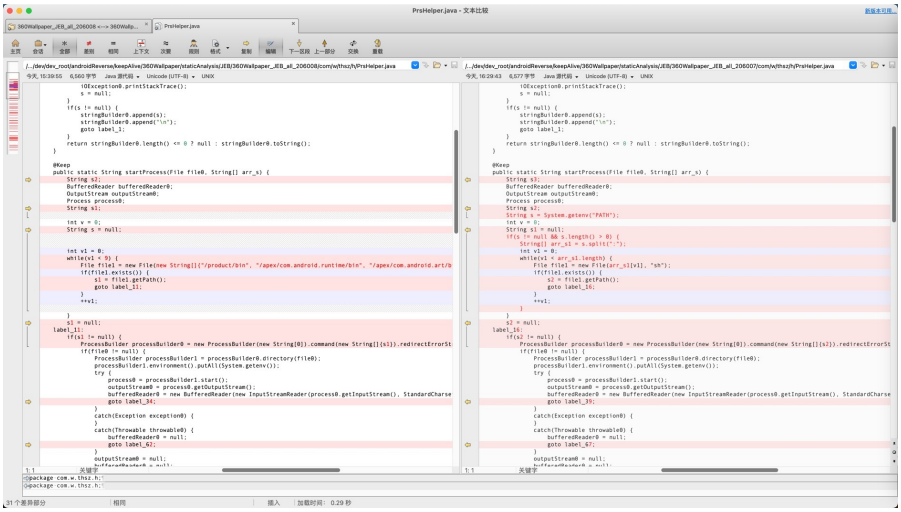
在批量反编译全部代码期间，遇到一些问题：

- 反编译到最后进度95%左右时，会卡死
    - 先后遇到多次，都是一样的问题：最后到 `95.3` 、 `95.4` 左右的进度后，就卡死了
        - 



- 如果要继续等待：估计永远没有结果
  - 之前等待了足够长时间：几十分钟，仍然没有任何进度更新
- 最后只能去：点击 `Cancel` 去取消
- 之后即可正常继续去保存和导出已反编译出的所有java代码

- 多次（2次）反编译的结果，内容不一致=输出结果不稳定
  - 专门测试了2次
    - 一次最终反编译总数是：`206008`
    - 另一次最终反编译总数是：`206007`
  - 发现前后反编译出的java代码，不一致
    - 总体上反编译的内容的效果，算是各有优劣
      - 有时候是 `206007` 的版本更好点
        - 举例
          - 此处还原了 `System.getenv("PATH")` 的逻辑，而不是输出所有返回的结果的列表



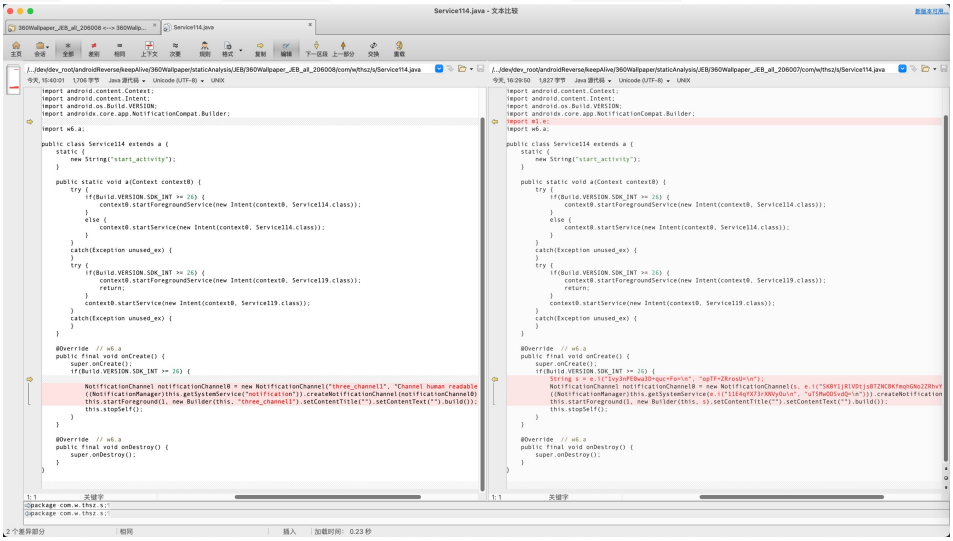
- `206007` 版本相关代码

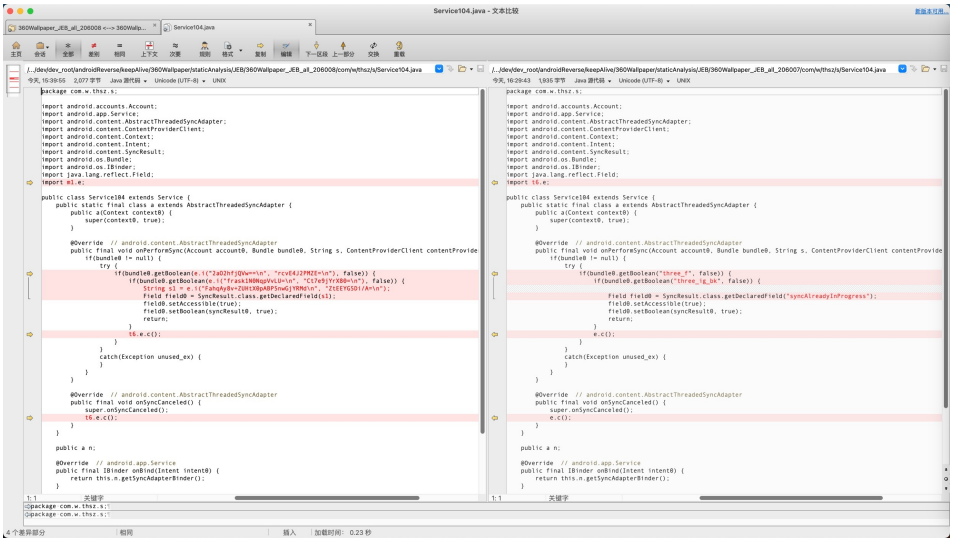
```
String s = System.getenv("PATH");
int v = 0;
String s1 = null;
if(s != null && s.length() > 0) {
```

```
        String[] arr_s1 = s.split(":");
        int v1 = 0;
        while(v1 < arr_s1.length) {
            File file1 = new File(arr_s1[v1], "sh");
            if(file1.exists()) {
                s2 = file1.getPath();
                goto label_16;
            }
            ++v1;
        }
    }
```

- `206008` 版本相关代码

```
    String s1;
    int v = 0;
    String s = null;
    int v1 = 0;
    while(v1 < 9) {
        File file1 = new File(new String[]{"/product/bin",
"/apex/com.android.runtime/bin", "/apex/com.android.art/b
in", "/system_ext/bin", "/system/bin", "/system/xbin", "/
odm/bin", "/vendor/bin", "/vendor/xbin"}[v1], "sh");
        if(file1.exists()) {
            s1 = file1.getPath();
            goto label_11;
        }
        ++v1;
    }
```

- （个别）加密字符串被自动解密还原成原始字符串了
  - 注：不论哪个版本，其实绝大多数加密字符串，都是已经被解密还原的。此处只是个别字符串是否被解密还原，有区别
  - `206008` 版本更好点：`206008` 自动解密还原了，而 `206007` 没有
    - 
  - `206007` 版本更好点：`206007` 自动解密还原了，而 `206008` 没有



然后只能根据自己的实际情况，决定保留哪份结果。

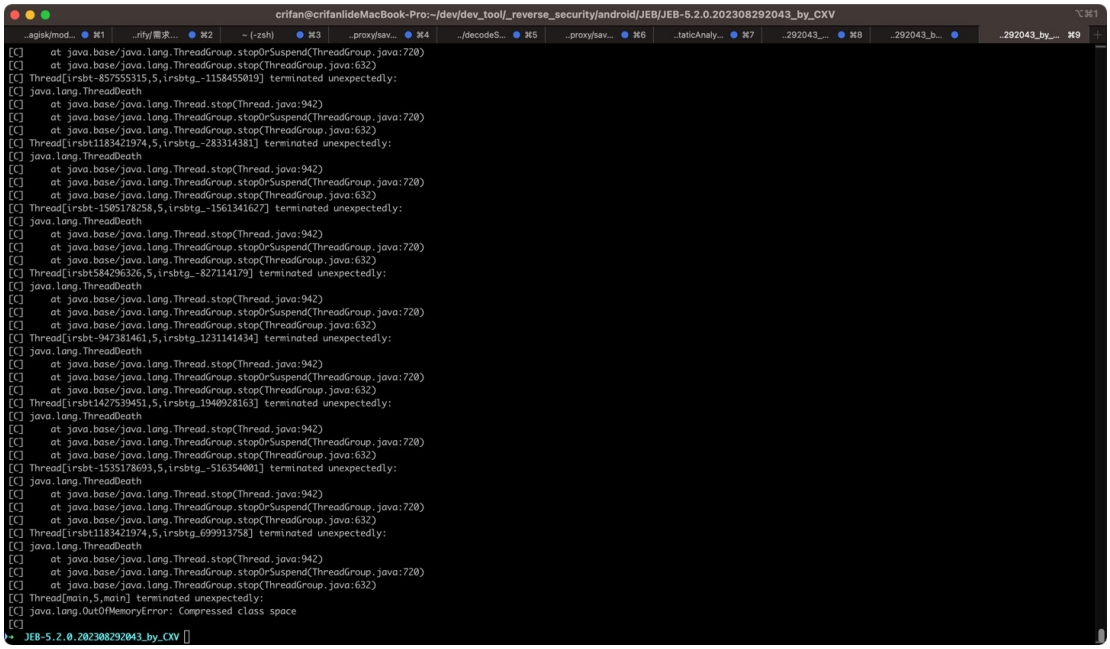
或者实在不行，再多弄几次代码的全部反编译，对比找到相对最优的结果。

总之对于反编译结果最后卡死，以及每次输出结果不一致不稳定，还是有点麻烦的。

OOM内存崩溃问题：java.lang.OutOfMemoryError Compressed class space

- 现象
  - 用JEB反编译抖音，导出全部源码时，最后报错：

```
[C] Thread[main,5,main] terminated unexpectedly:
[C] java.lang.OutOfMemoryError: Compressed class space
```



- 原因：抖音内部代码量太大，反编译全部源码所要消耗资源太多，导致默认的 `CompressedClassSpaceSize=1G` ，不够用，从而造成此处报错，Compressed class space出现OOM内存崩溃不够用的问题

- 解决办法：增大 `CompressedClassSpaceSize` 的大小
- 具体步骤：
  - 先要确认： `CompressedClassSpaceSize` 应该设置为多大的值
    - 此处 `CompressedClassSpaceSize` 的范围是：最小是 `1048576=1M` ，最大是 `3221225472=3G` ，所以此处设置为最大的**3G**
  - 再去设置参数
    - 此处JEB的Mac启动脚本： `/Users/crifan/dev/dev_tool/_reverse_security/android/JEB/JEB-5.2.0.202308292043_by_CXV/jeb_macos.sh` 中

```
JVMOPT="-XX:+UseCompressedClassPointers -XX:+UseCompressedOops -XX:CompressedClassSpaceSize=3G -Xss4M -Xmx42G"
```

      - 参数含义解释
        - `-XX:+UseCompressedClassPointers`
          - （默认其实已开启，但是此处确保的确）开启 `UseCompressedClassPointers`
        - `-XX:+UseCompressedOops`
          - （默认其实已开启，但是此处确保的确）开启 `UseCompressedOops`
        - `-XX:CompressedClassSpaceSize=3G`
          - 设置 `CompressedClassSpaceSize` 为 `3G`
            - 注：默认大小是**1G**
        - `-Xss4M`
          - == `-XX:ThreadStackSize=4M`
          - 含义：每个线程的堆栈大小：4MB
        - `-Xmx42G`
          - 最大堆栈内存： `42G`

从而使得 `CompressedClassSpaceSize` 的空间足够大，可以避免最后反编译导出全部源码时， `CompressedClassSpaceSize` 空间超过最大大小，就不会报 `java.lang.OutOfMemoryError Compressed class space` 的错了。

附录

下面列出相关参考资料。

参考资料

- 【已解决】Mac中下载安装JEB
- 【已解决】Mac中打开运行JEB
- 【已解决】安卓保活逆向360Wallpaper：用JEB反编译
- 【已解决】安卓保活逆向360Wallpaper：用JEB反编译得到java代码
- 【记录】安卓逆向：JEB使用心得
- 【已解决】安卓保活逆向360Wallpaper：JEB的基本使用
- 【整理】安卓逆向：JEB功能介绍
- 【整理】JEB相关内容
- 【已解决】安卓逆向：JEB批量反编译所有java源码
- 【记录】用JEB反编译360Wallpaper最后由于超时而结束
- 【已解决】安卓逆向：JEB批量反编译360Wallpaper的全部log日志
- 【记录】用JEB反编译360Wallpaper：反编译总数206008
- 【记录】用JEB反编译360Wallpaper最后由于超时而结束
- 【记录】安卓apk反编译java代码效果对比：JEB对比jadx
- 【已解决】安卓保活逆向360Wallpaper：jadx反编译代码中e.i的通用字符串加密解密函数
- 【已解决】Mac中JEB分析抖音apk报错：Error java.lang.OutOfMemoryError Java heap space
- 【已解决】Mac中JEB反编译抖音apk全部源码最后报错：java.lang.OutOfMemoryError Compressed class space
-
- Android逆向开发
- Android逆向：静态分析
- Android逆向：动态调试
- 安卓反编译利器：jadx
-
- JEB Decompiler by PNF Software
- JEB Community Edition - JEB Decompiler by PNF Software
- JEB - JEB Decompiler by PNF Software
- JEB Android Decompiler - JEB Decompiler by PNF Software
- 抖音设备注册生成device*id*与*install_id hook*分析记录_*Android*-逆向的博客-*CSDN*博客抖音设备id
- Tips - JEB Decompiler (pnfsoftware.com)
- Debugging Android Dalvik and native code seamlessly with JEB - YouTube
- Crypto Monitoring with the Android Debuggers API – JEB in Action (pnfsoftware.com)
- Comparison of Java output decompiled by JEB and other decompilers - JEB Decompiler by PNF Software
- Bad Java decompilation means erroneous statement in research paper – JEB in Action (pnfsoftware.com)
- An introduction to JEB Android Debuggers – JEB in Action (pnfsoftware.com)
- Android 反混淆神器JEB2的使用简介 - 飞少的博客 | Jack's Blog
- [原创]JEB2反混淆神器-『Android安全』-看雪安全论坛
- Android 逆向工具篇—反编译工具的选择与使用 - 腾讯云开发者社区-腾讯云 (tencent.com)
- 【反编译系列】二、反编译代码（jeb） - HaiyuKing - 博客园 (cnblogs.com)
- JEB动态调试Smali-真机/模拟器（详细，新手必看） - 『移动安全区』 - 吾爱破解 - LCG - LSG |安卓

破解|病毒分析|www.52pojie.cn

-