

# 目录

前言	1.1
JEB概览	1.2
下载运行JEB	1.3
JEB功能和页面	1.4
Bytecode/Hierarchy	1.4.1
单个文件多个显示模式	1.4.2
Bytecode多个显示模式	1.4.3
底部多个tab页	1.4.4
保存项目文件jdb2	1.4.5
多主题显示效果	1.4.6
反编译器	1.4.7
脚本	1.4.8
插件	1.4.9
JEB静态分析	1.5
反编译安卓apk	1.5.1
反编译出java	1.5.1.1
单个反编译	1.5.1.1.1
全部反编译	1.5.1.1.2
反编译效果对比	1.5.1.1.3
JEB vs jadx	1.5.1.1.3.1
解析so库文件	1.5.1.2
JEB动态调试	1.6
JEB使用心得	1.7
页面显示心得	1.7.1
反编译java代码心得	1.7.2
JEB常见问题	1.8
页面显示问题	1.8.1
反编译java代码问题	1.8.2
附录	1.9
参考资料	1.9.1

# 安卓逆向利器：JEB

- 最新版本： v1.0.0
- 更新时间： 20230916

## 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jad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

此电子书教程的全部内容，如无特别说明，均为本人原创。其中部分内容参考自网络，均已备注了出处。如发现有侵权，请通过邮箱联系我 `admin 艾特 crifan.com`，我会尽快删除。谢谢合作。

各种技术类教程，仅作为学习和研究使用。请勿用于任何非法用途。如有非法用途，均与本人无关。

## 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

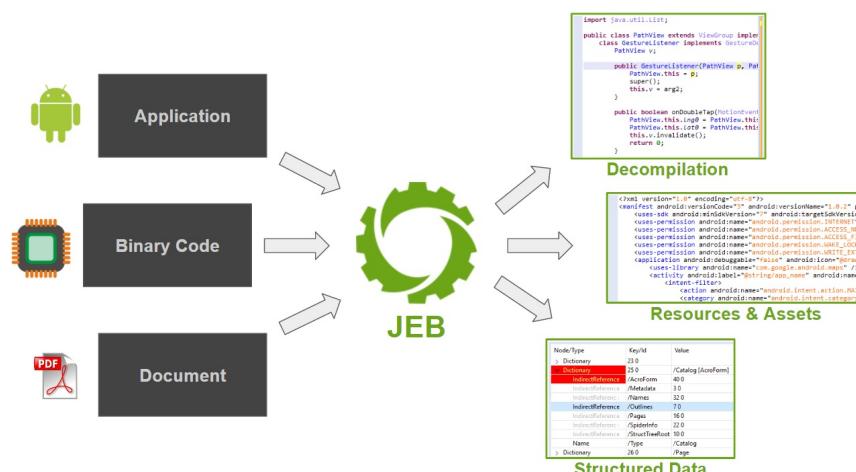
关于作者更多介绍，详见：

[关于CrifanLi李茂 – 在路上](#)

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：  
2023-09-16 21:58:04

# 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

JEB is our reverse-engineering platform to perform disassembly, decompilation, debugging, and analysis of code and document files, manually or as part of an analysis pipeline.

[Purchase a JEB License](#)

### Android Decompiler + Android Debuggers

Use JEB to perform reverse engineering of malicious APKs and security audits.

Cut down on costly reverse engineering time: decompile obfuscated APKs, examine Java code and debug closed-source apps in minutes. Our modular back-end, coupled with a powerful UI front-end for desktop platforms, allow refactoring operation and scripting to automate complex tasks.

Perform static and dynamic analysis of Android applications, goodware or badware, small or large.

- **Decompile** code using our Dalvik decompiler, including multi-dex APK.
- **Refactor** the analysis to defeat obfuscated code generated by application protectors.
- **Reconstruct** resources and obfuscated XML files.
- **Debug** Dalvik code as well as all native code (Intel, ARM), seamlessly.
- **Automate** and script your reverse engineering tasks via our API.

■ Powerful

■ Flexible

■ Extensible



## 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.

<b>Build Type &gt;</b>	<b>JEB CE Community Edition</b>	<b>JEB Android</b>	<b>JEB Pro (or Pro Floating)</b>
<b>Base</b>			
Suitable for enterprise or professional use			✓
<b>Decompilers</b>			
Dalvik Decomplier (a.k.a. Android Decomplier)		✓	✓
Deobfuscation module for "virtualized" code ( <a href="#">details</a> )			✓
Advanced deobfuscators requiring native code emulation (JNI) ( <a href="#">details</a> , <a href="#">details</a> )			✓
Java Decomplier		✓	✓
Intel x86 Decomplier	✓		✓
Intel x86-64 Decomplier	✓		✓
ARM Decomplier		✓	✓
ARM64 Decomplier		✓	✓
MIPS Decomplier			✓
MIPS64 Decomplier			✓
RISC-V Decomplier			✓
S7 PLC Block Decomplier			✓
WebAssembly Decomplier			✓
Ethereum Decomplier for Smart Contracts (EVM Decomplier)			✓
Diem (Libra) Decomplier for Move modules ( <a href="#">open-sourced</a> )			✓
<b>Base Plugins</b>			
Archive files (zip, 7z, tar)	✓	✓	✓
Image files (bmp, png, jpg, gif, ico)	✓	✓	✓
Structured text files (html, xml, json)	✓	✓	✓
Certificate files (x.509)	✓	✓	✓
<b>Application Processors</b>			
Android APK (incl. dex, odex, x-apk)	✓	✓	✓
Windows PE/PE64/COFF, PDB	✓	✓	✓
Linux ELF/ELF64	✓	✓	✓
Mach-O, Mach-O/FAT	✓	✓	✓
Intel HEX (hex)	✓	✓	✓
Chrome Extension (crx)	✓	✓	✓
WebAssembly module (wasm)		✓	✓
Ethereum contracts (evm)		✓	✓
<b>Disassemblers</b>			
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 ( <a href="#">limitations</a> )	✓	✓	✓
<b>Debuggers</b>			
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)		✓	✓
<b>Document Parsers</b>			
Adobe PDF			✓
FAT file systems (file access table)		✓	✓ ( <a href="#">plugin</a> )
<b>Platform</b>			
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

The screenshot shows a web browser window with the URL <https://www.pnfssoftware.com/jeb/buy>. The page title is "Purchase a JEB license - JEB". The content includes a heading "How much does JEB cost?", a brief description about desktop software licenses, and a comparison table for three license types: JEB Android, JEB Pro, and JEB Pro Floating. At the bottom is a green button labeled "Purchase a JEB License".

License Type	JEB Android	JEB Pro	JEB Pro Floating
Price	12 months @ \$1,200 / user Monthly @ \$140 / user	12 months @ \$2,000 / user	12 months @ \$4,000 / seat
List of analysis modules	<a href="#">Android modules</a>	<a href="#">All modules</a>	<a href="#">All modules</a>
JEB with official UI client	✓	✓	✓
Support for extensions (plugins and scripts)	✓	✓	✓
Work without an Internet connection		✓	✓
Execute third-party front-end clients		✓	✓
Floating seats			✓

## JEB收费版的破解版

### 仅供学习研究技术用

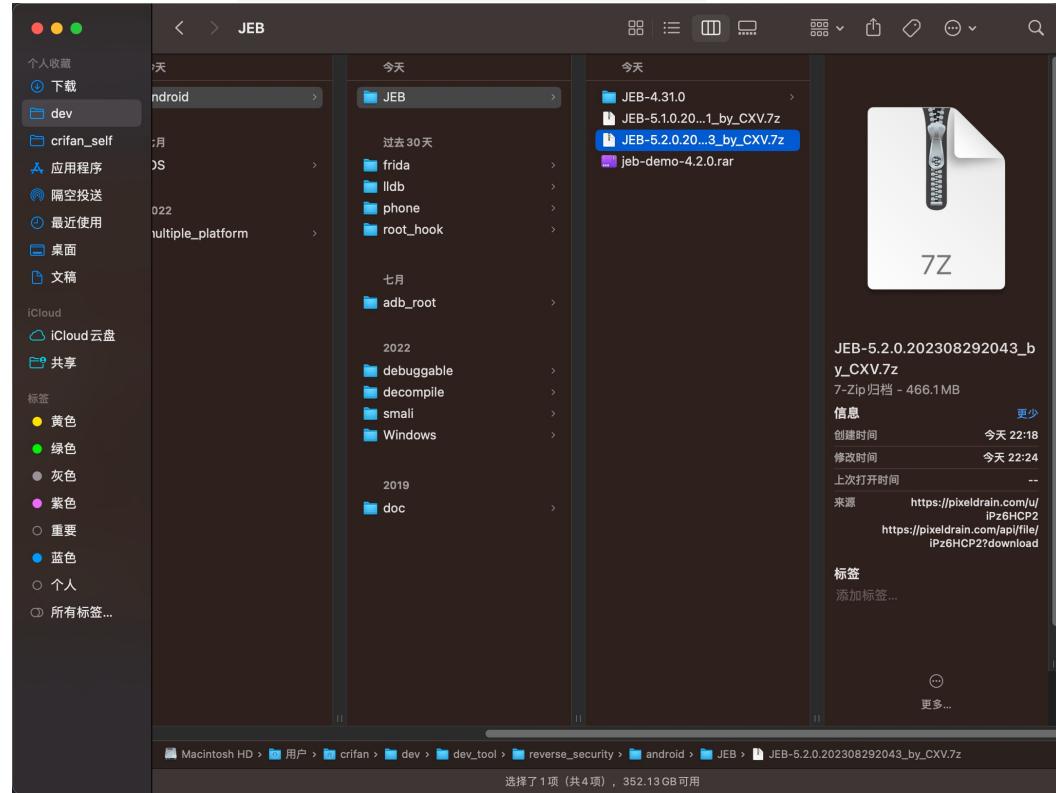
此处提供破解版, 仅供学习研究技术用, 请勿用于其他非法和商业用途。请支持正版。

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](#)

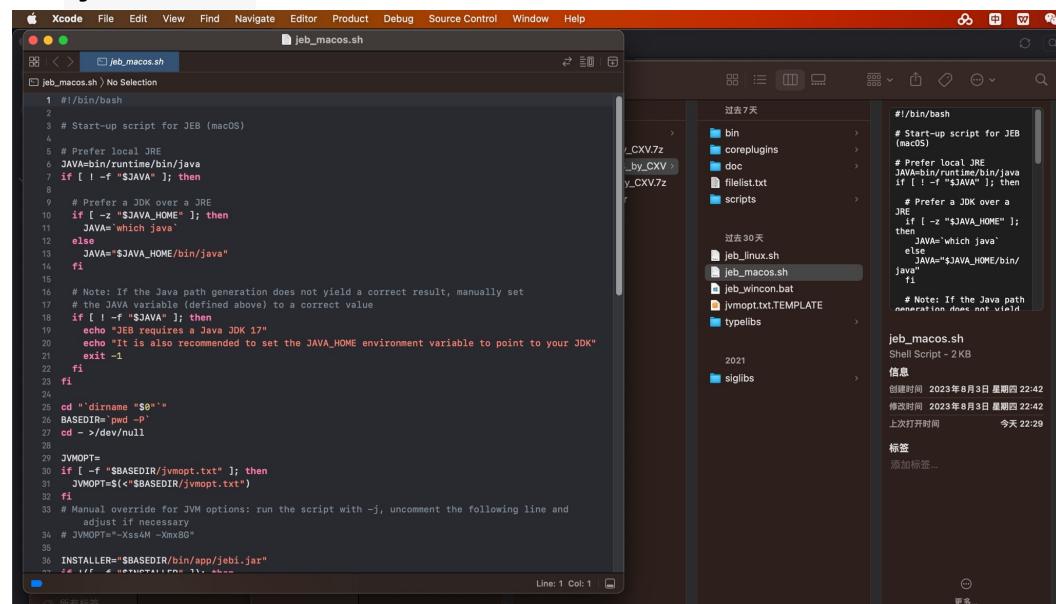
- 下载得到 460MB+ 的: [JEB-5.2.0.202308292043\\_by\\_CXV.7z](#)



- 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](#)

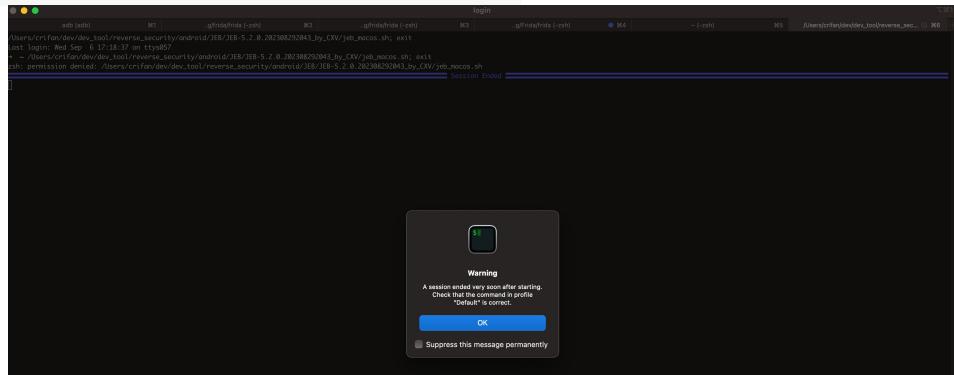
- 运行方式

- 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)
```

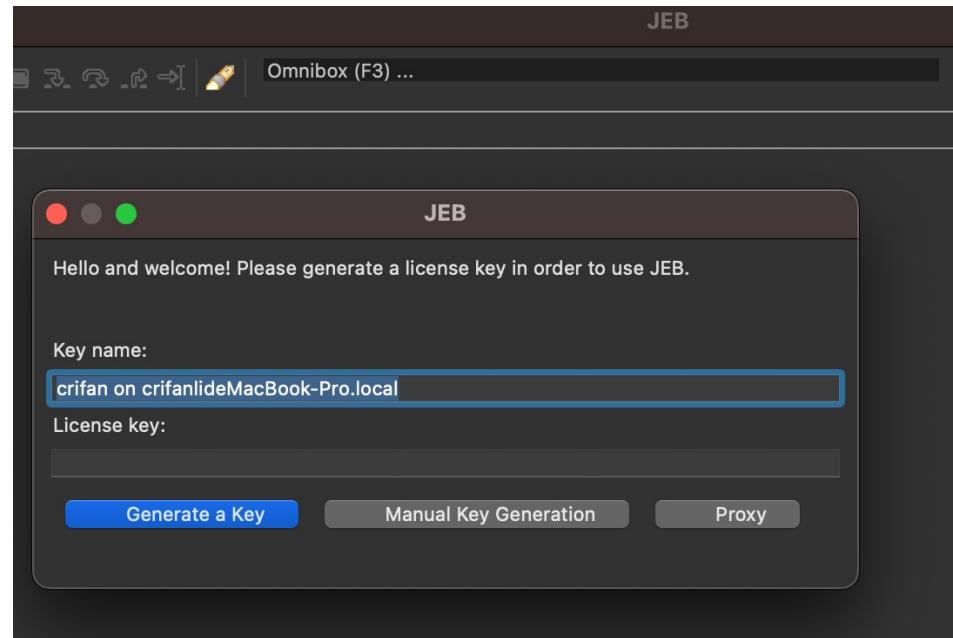
- Linux: 自带shell

- Win: cmd.exe

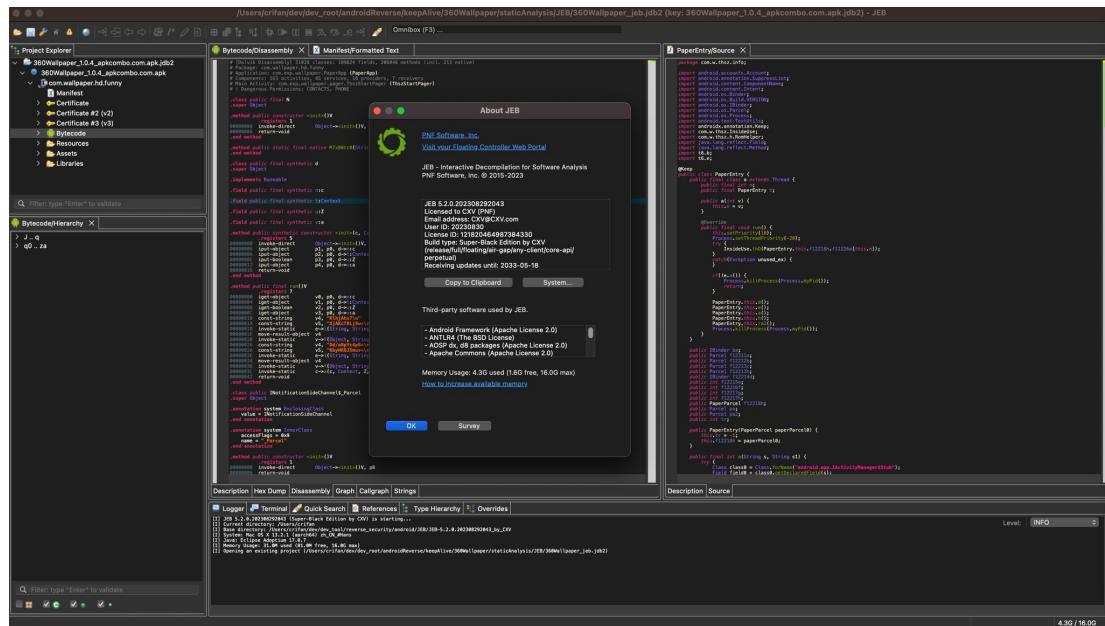
- 首次启动时

- 需要去生成key, 再点击继续, 即可

- 点击 Generate a Key -> Continue



- 主界面

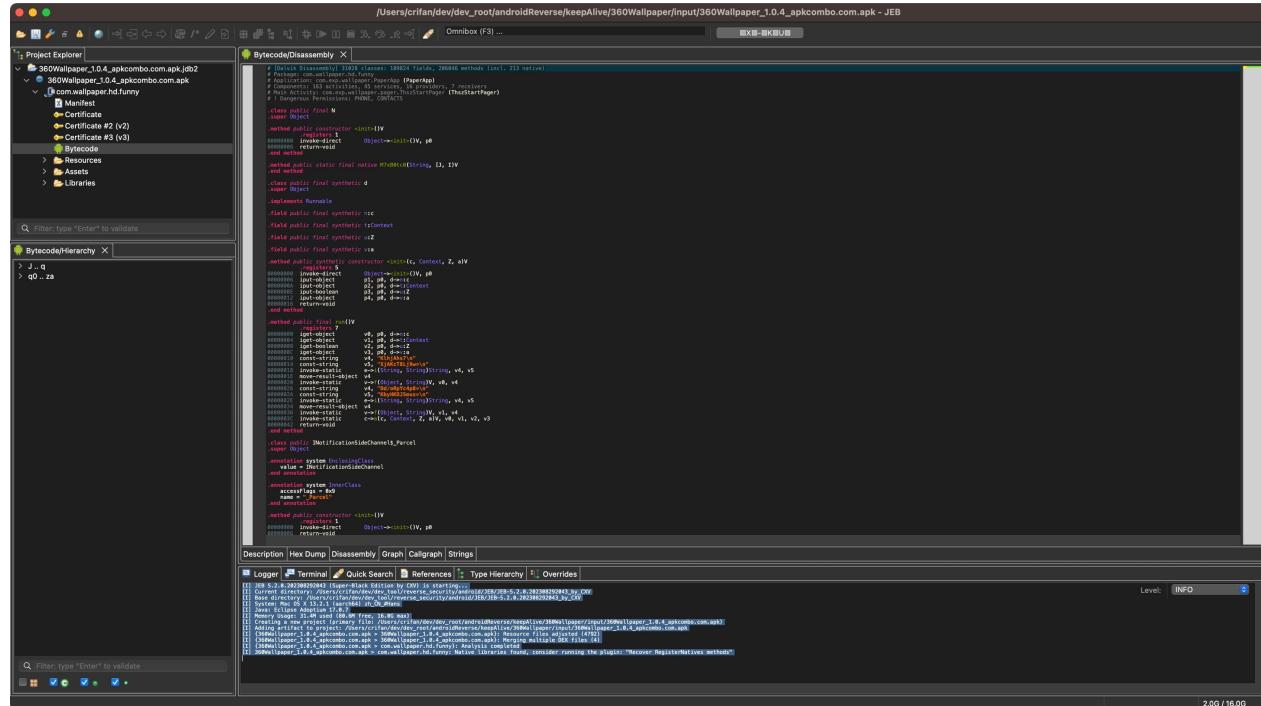


crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新:  
2023-09-16 11:39:36

# 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。

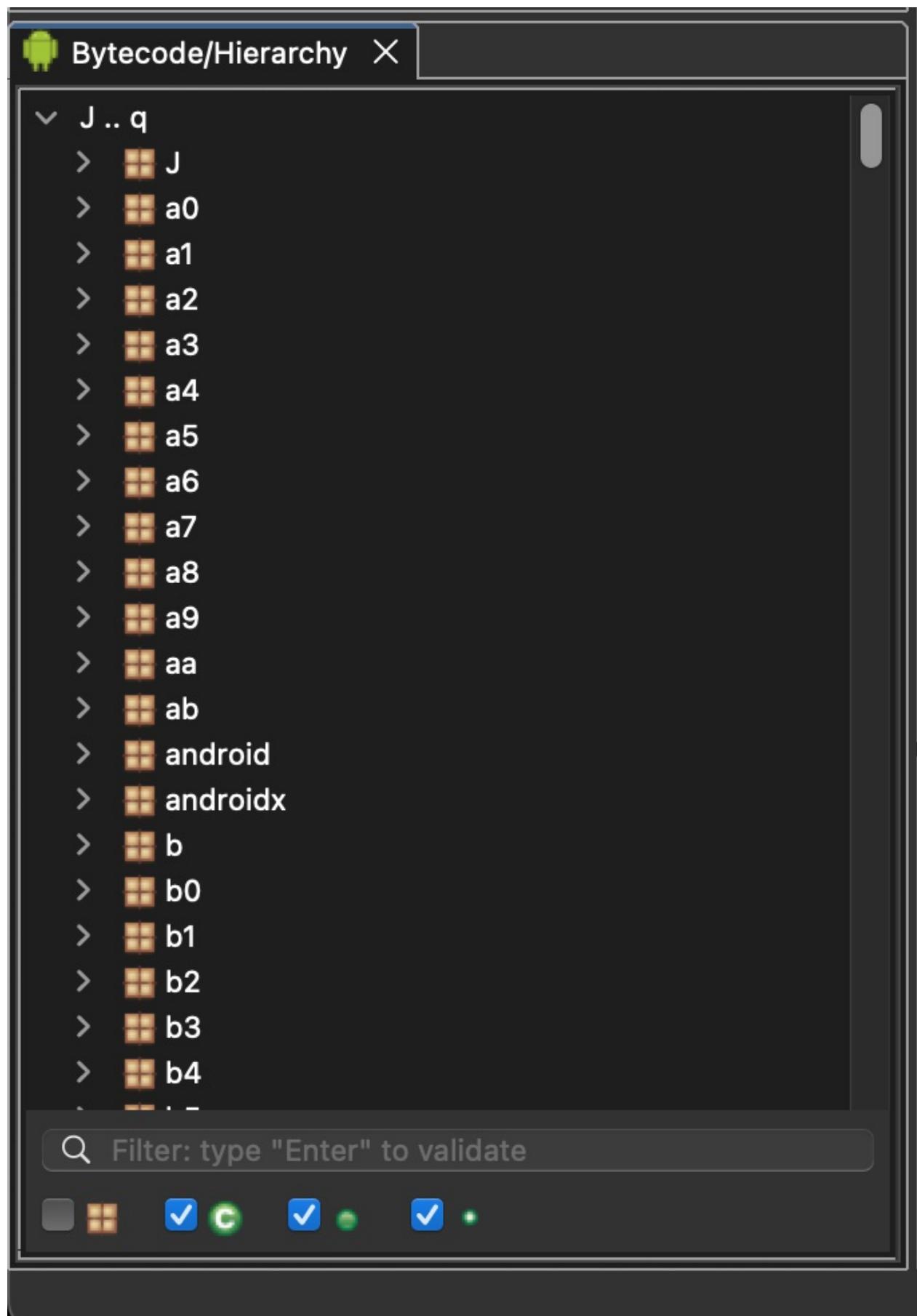
先贴出主要总体界面：

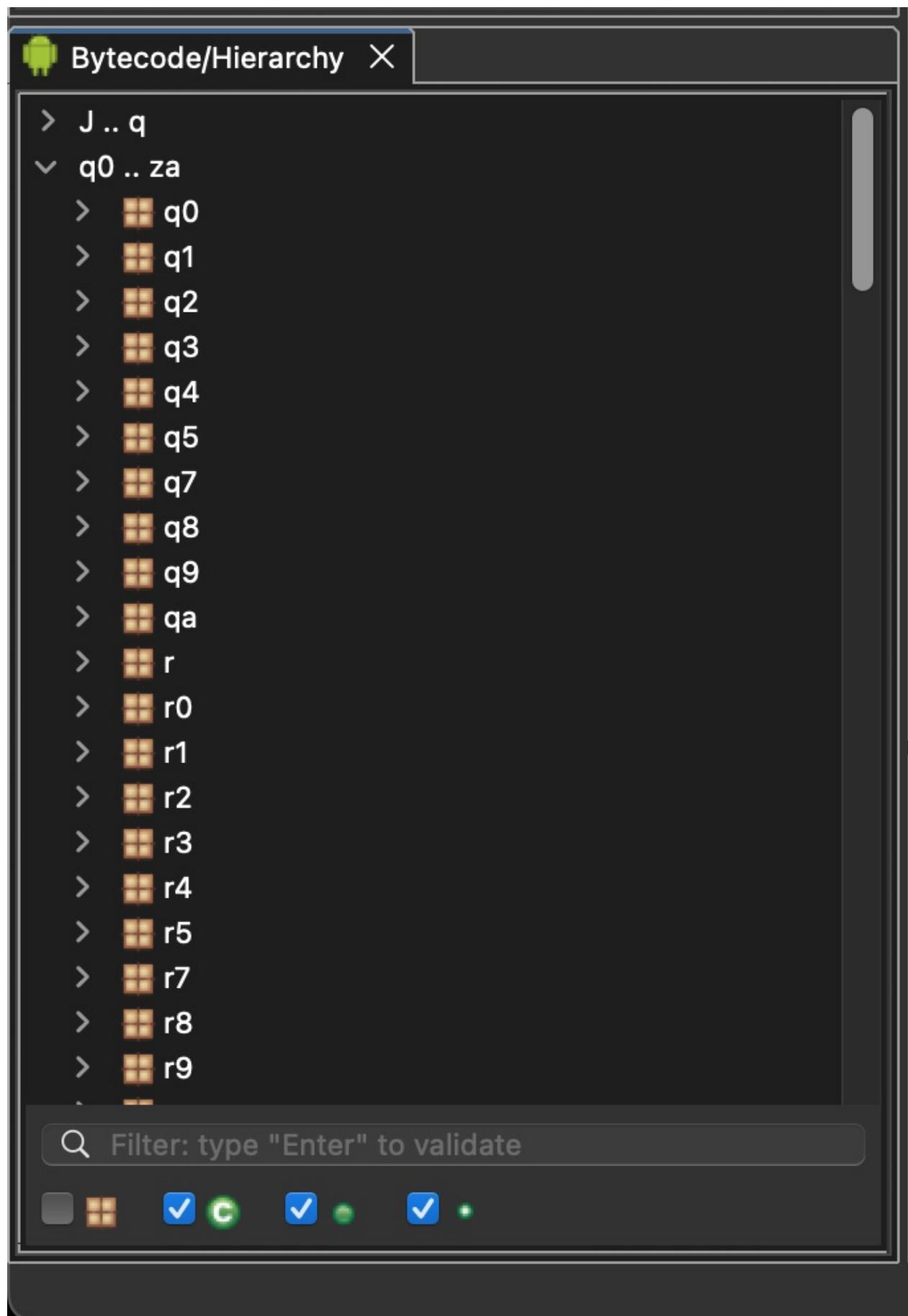


cifran.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：  
2023-09-16 00:04:43

## Bytecode/Hierarchy

点击展开类的节点：







## 单个文件多个显示模式

此处同一个文件 `AndroidManifest.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

◦

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:  
2023-09-15 23:59:46

## Bytecode的多个显示模式

- Description=描述

◦

- Hex Dump

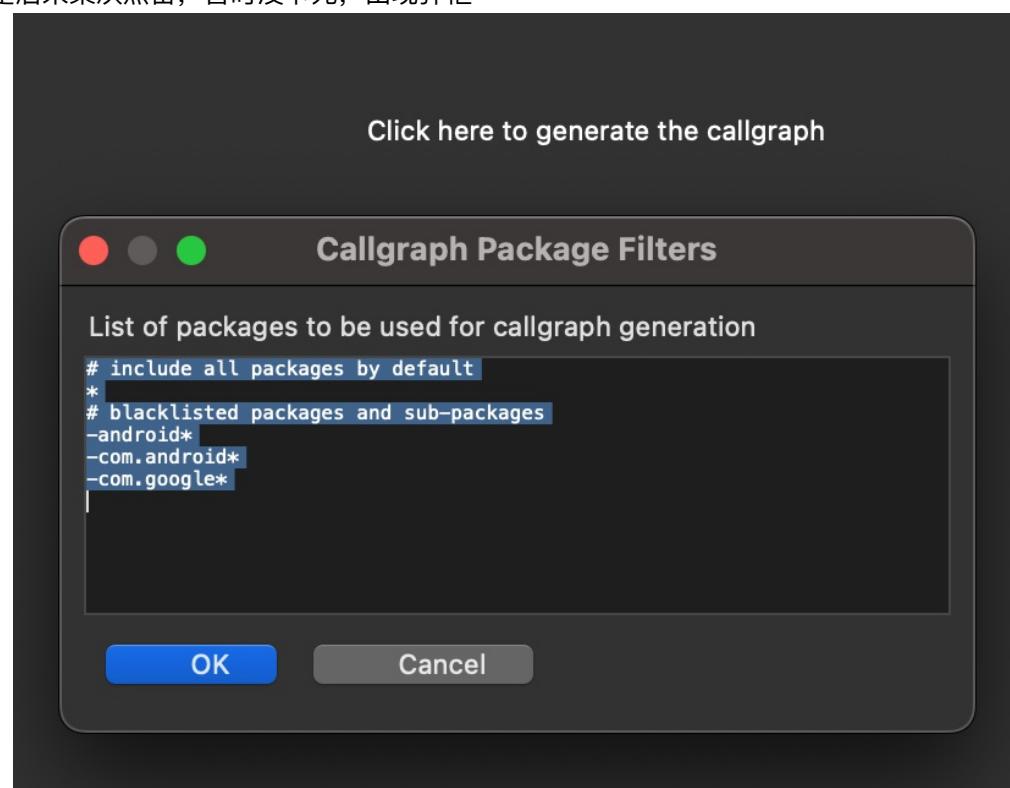
◦

- Disassembly

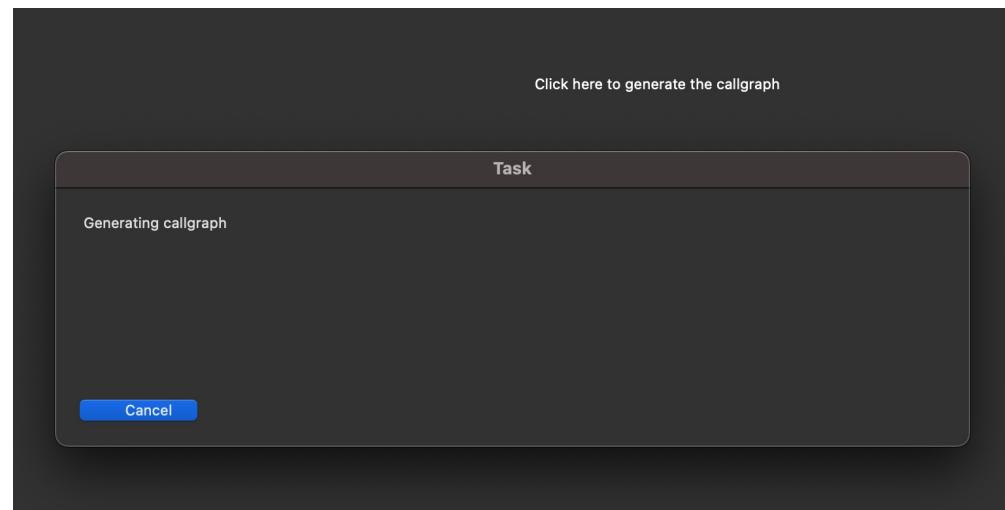
- - Graph

- - 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)
# Dangerous Permissions: PHONE, CONTACTS
.class public final N
.super Object
.method public constructor <init>()
    .registers 1
    00000000 invoke direct Object-><init>(), p0
    00000006 return void
.end method
.method public static final native M7x80tcc(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 input object p1, p0, d>n c
0000000A input object p2, p0, d>t Context
0000000E input boolean p3, p0, d>u Z
00000012 input object p4, p0, d>v a
00000016 return void
.end method
.method public final run()V
registers 7
00000000 igure object v0, p0, d>n c
00000004 igure object v1, p0, d>t Context
00000008 igure boolean v2, p0, d>u Z
0000000C igure 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>()
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$_Parcel >readTypedObject(Parcel, Parcelable$Creator)Object, p0, p1
...
...

```

```

...
.class public abstract interface INotificationSideChannel
.super Object
.implements IInterface
.annotation system MemberClasses
value = {
    INotificationSideChannel$_Parcel,
    INotificationSideChannel$_Stub,
    INotificationSideChannel$_Default
}
.end annotation
.field public static final DESCRIPTOR String
.method static constructor <clinit>()
registers 3
    00000000 const string v0, "android$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
```

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:  
2023-09-16 00:03:31

## JEB底部多个tab页

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

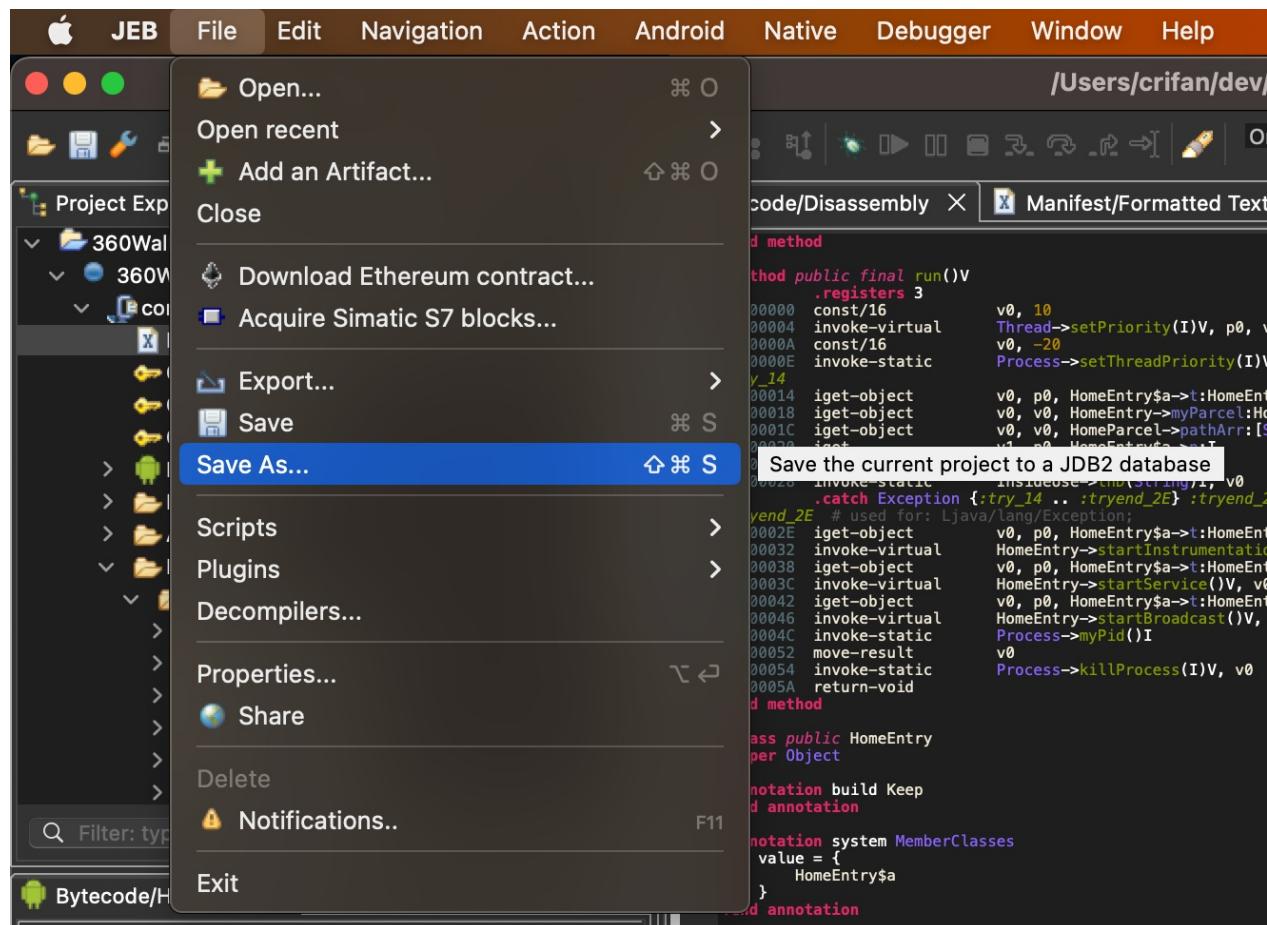
- - Overrides

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：  
2023-09-15 23:59:10

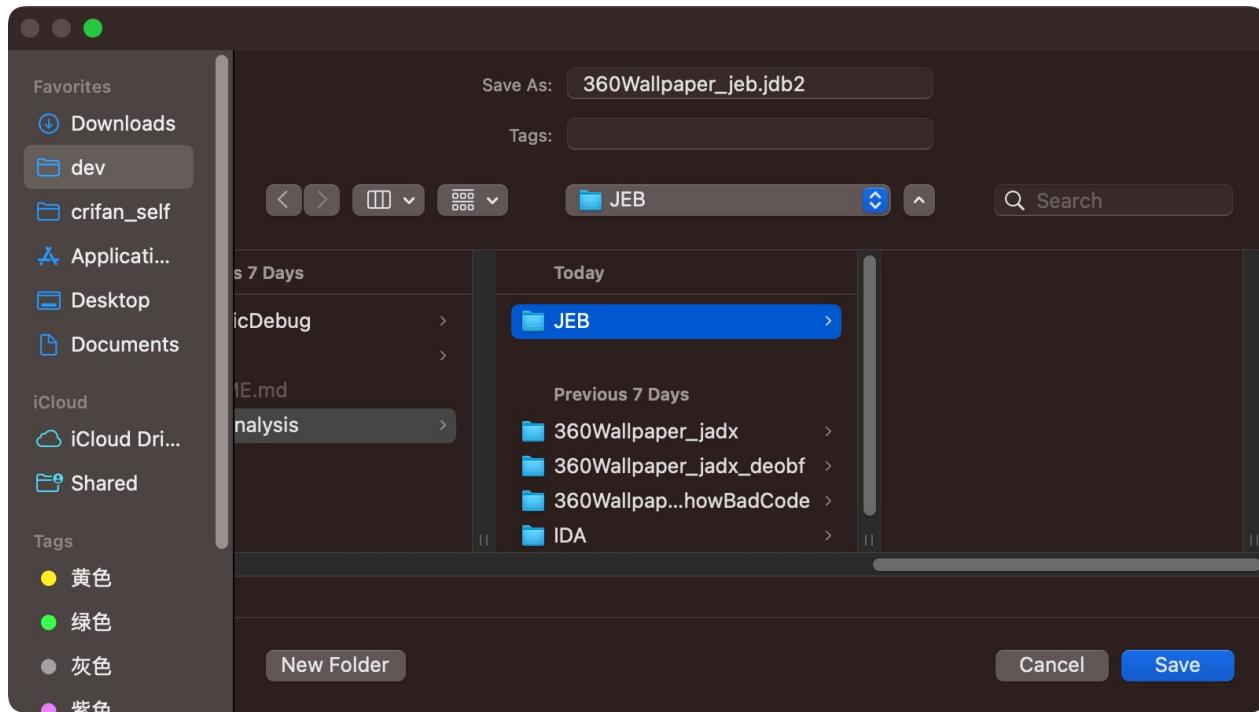
## 保存项目文件jdb2

当需要时，可以去保存当前项目：

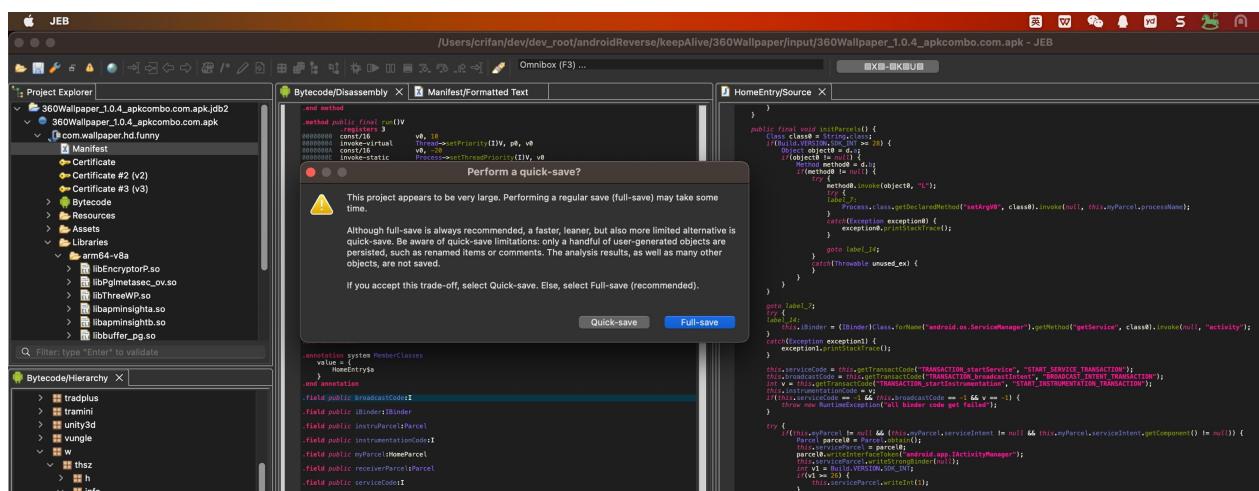
JEB -> File -> Save As , 会提示 Save the current project to JDB2 database



给文件起个名字：



首次保存时，会提示：

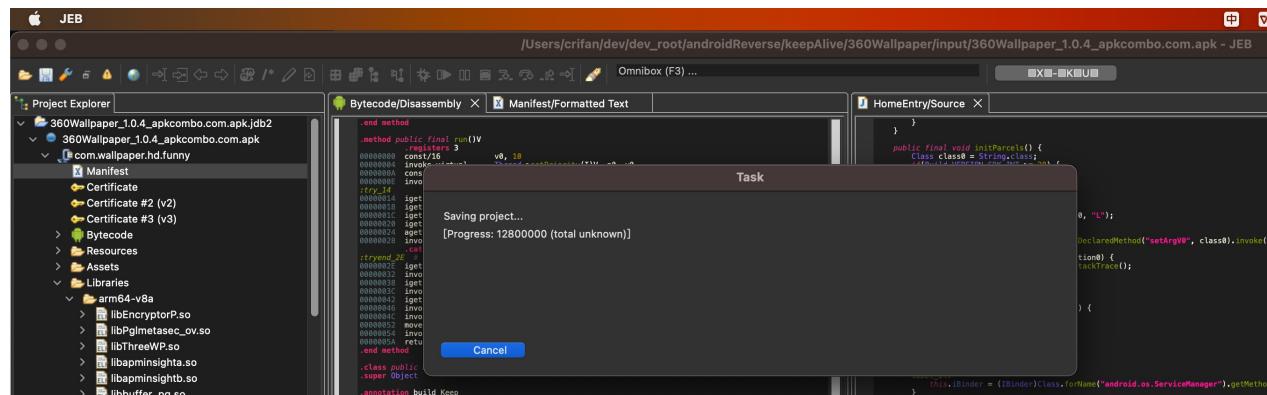


项目看起来很大，保存时选用哪种模式：

- 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

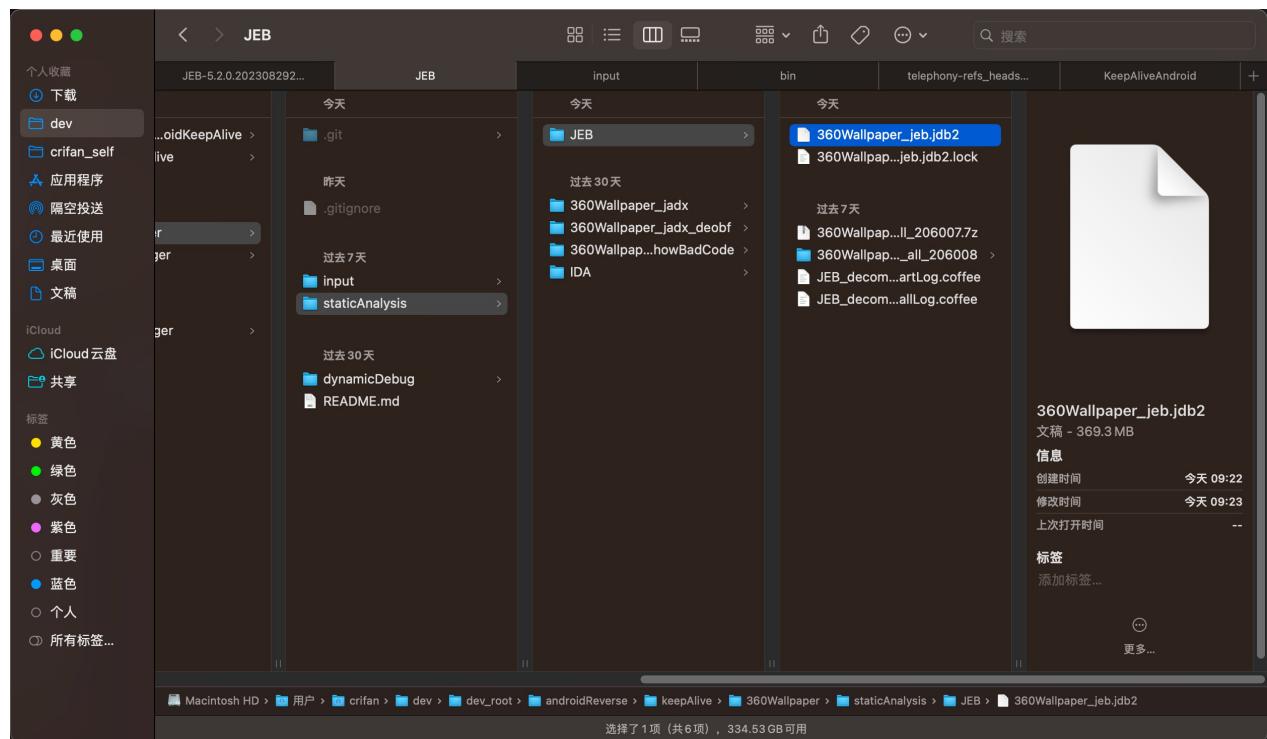
此处为了确保改动都保存，选择了： full-save = 全部保存

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：



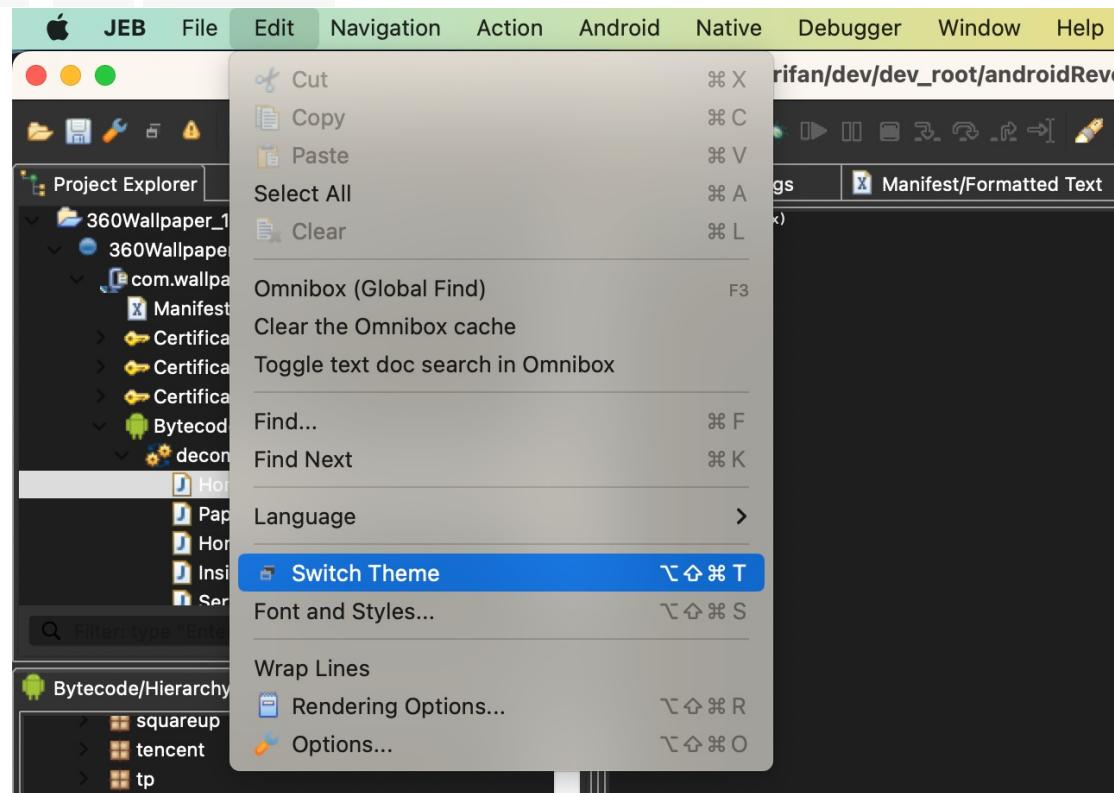
crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：  
2023-09-16 00:01:34

## JEB多主题显示效果

JEB中显示方面，支持多个主题：

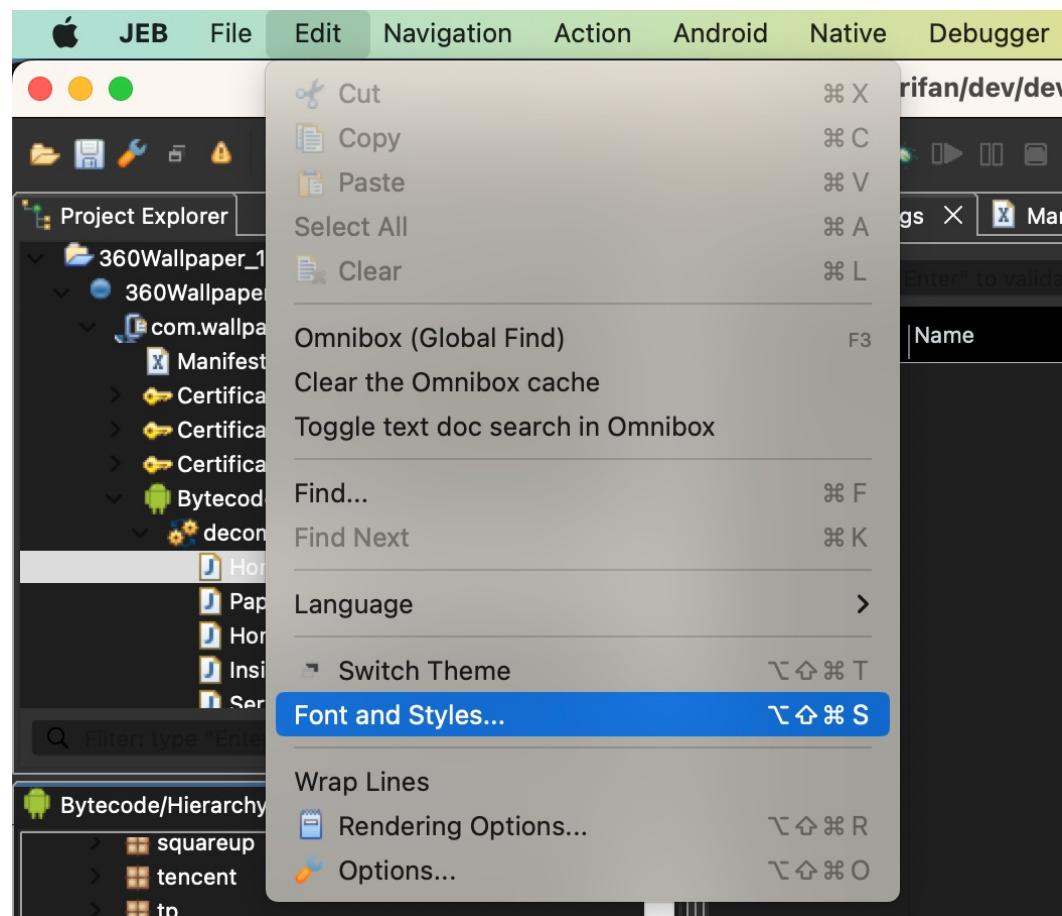
- 如何切换主题？

- JEB -> Edit - Switch Theme

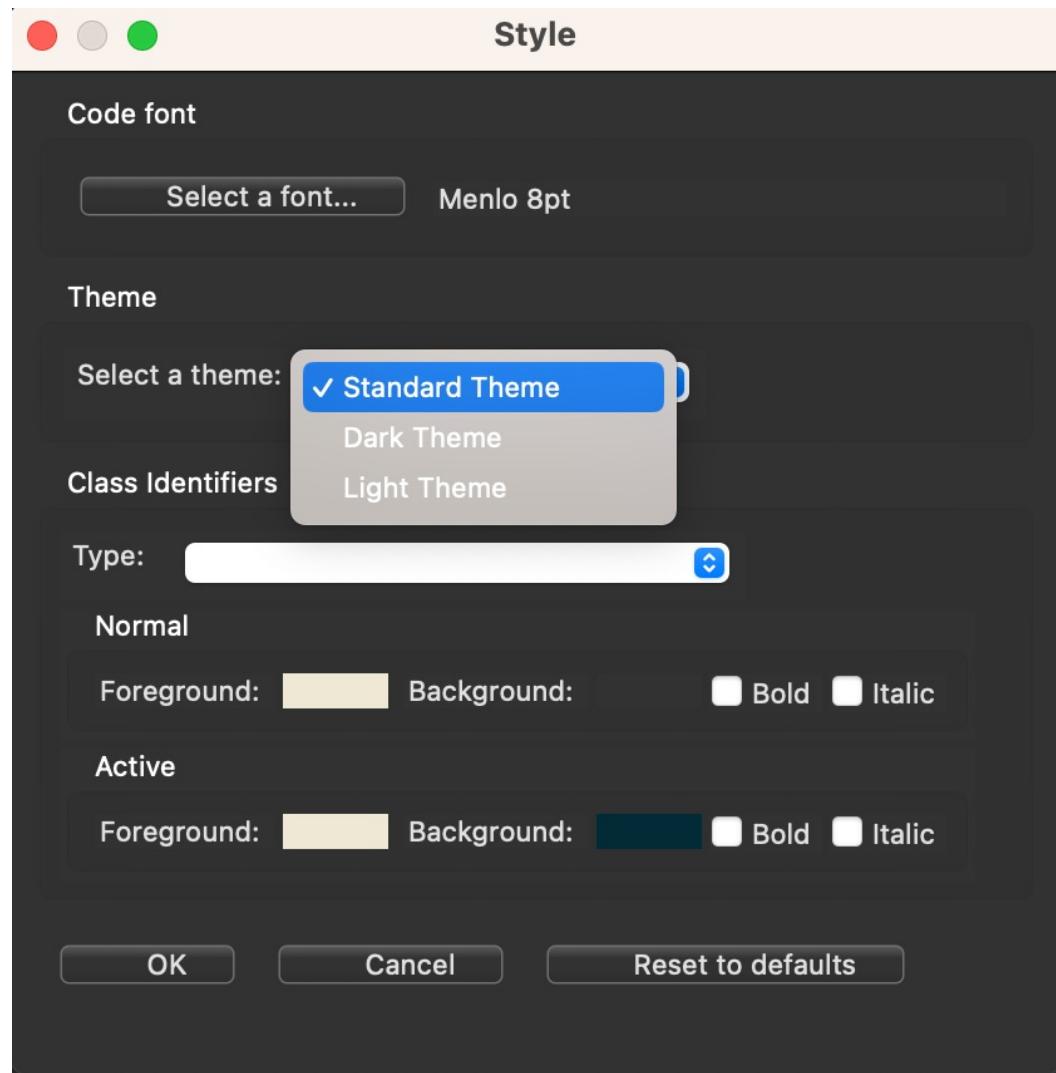


- 不同主题的名称

- 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



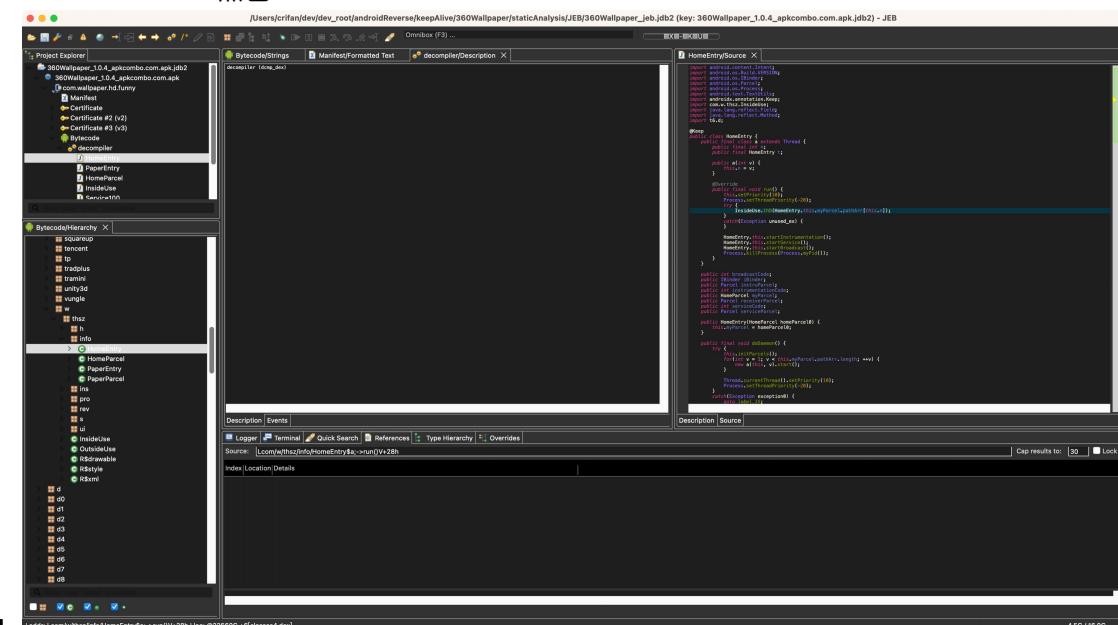
- 然后可以看到主题名称，可以切换主题



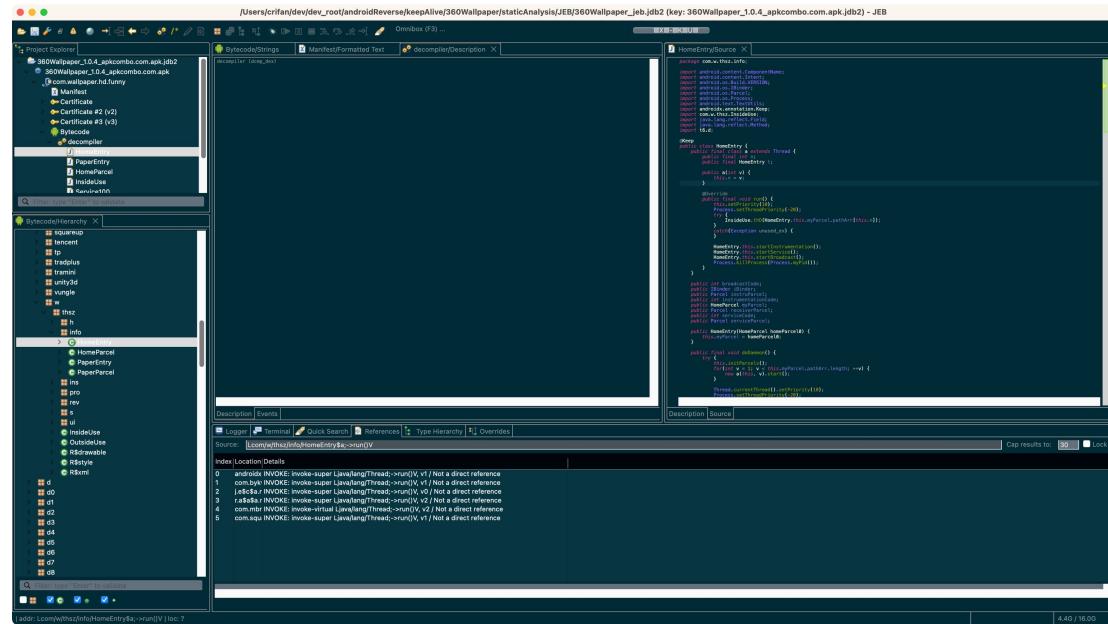
- Standard Theme == 黑色
- Dark Theme == 淡蓝色
- Light Theme == 淡黄色

- 多个不同主题的显示效果

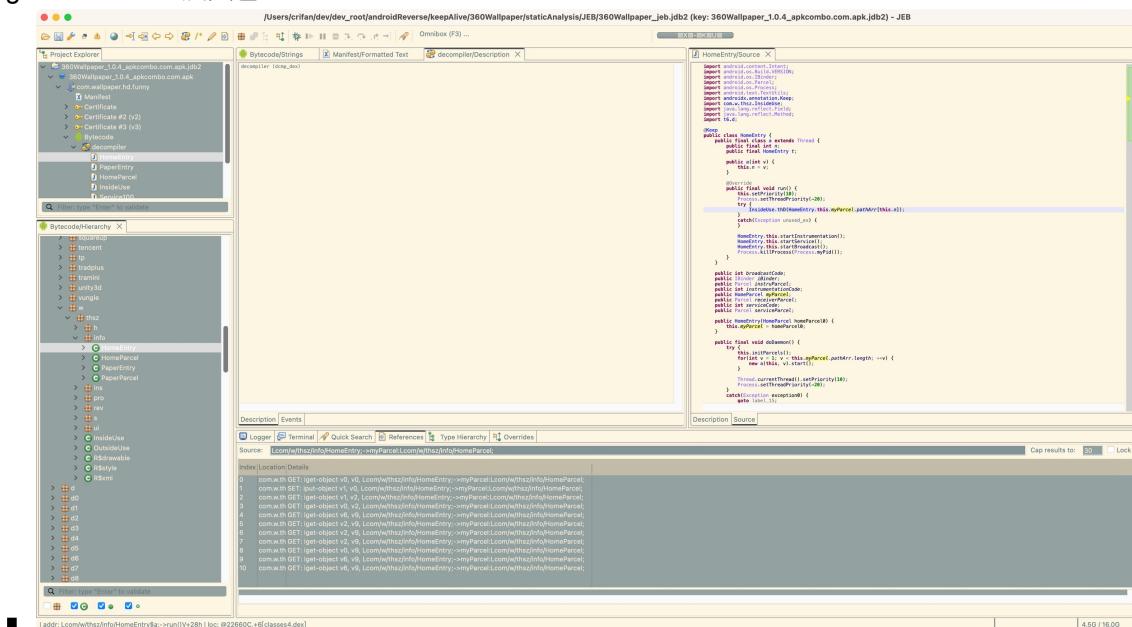
- Standard Theme == 黑色



- Dark Theme == 淡蓝色



- Light Theme == 淡黄色



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:

2023-09-16 21:43:43

## 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)

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：  
2023-09-16 12:31:08

## 脚本

JEB支持脚本。内置就有很多Scripts脚本：

- JEB -> File -> Scripts

◦

◦

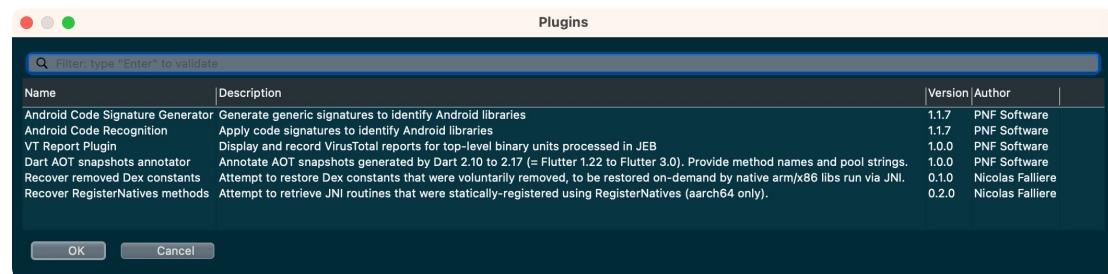
- 鼠标悬停到某个脚本时，还能显示脚本功能简介

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：  
2023-09-16 21:41:51

## 插件

JEB支持插件机制。内置就有一些插件：

- JEB -> File -> Plugins -> Execute and Engines Plugin
- 可以看到内置引擎插件
  - Android Code Signature Generator
  - Android Code Recognition
  - VT Report Plugin
  - Dart AOT snapshots 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



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：  
2023-09-16 12:39:44

# JEB静态反编译

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：  
2023-09-14 22:47:33

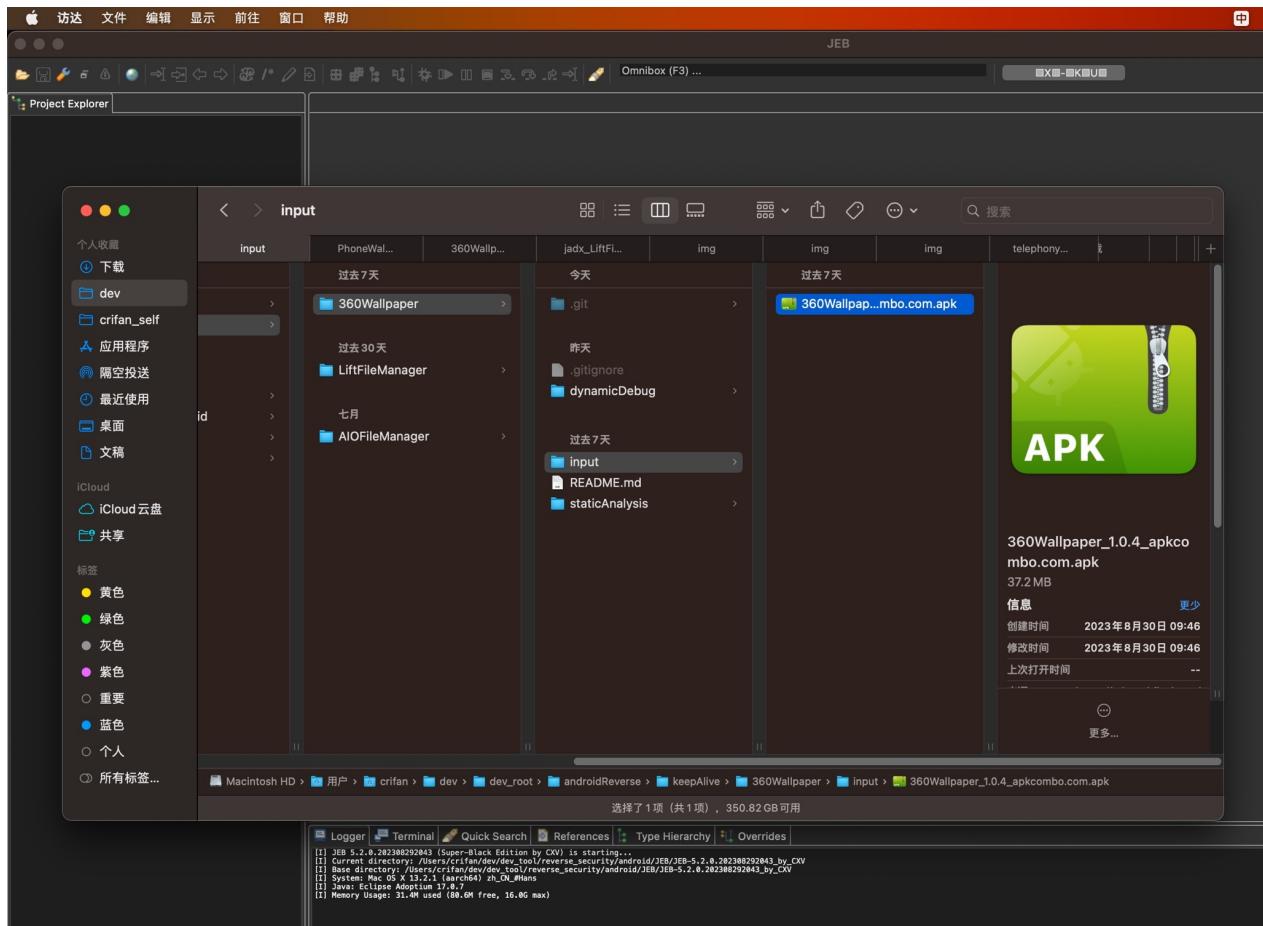
## 用JEB反编译安卓apk

直接把apk拖动到JEB中，即可自动开始反编译。

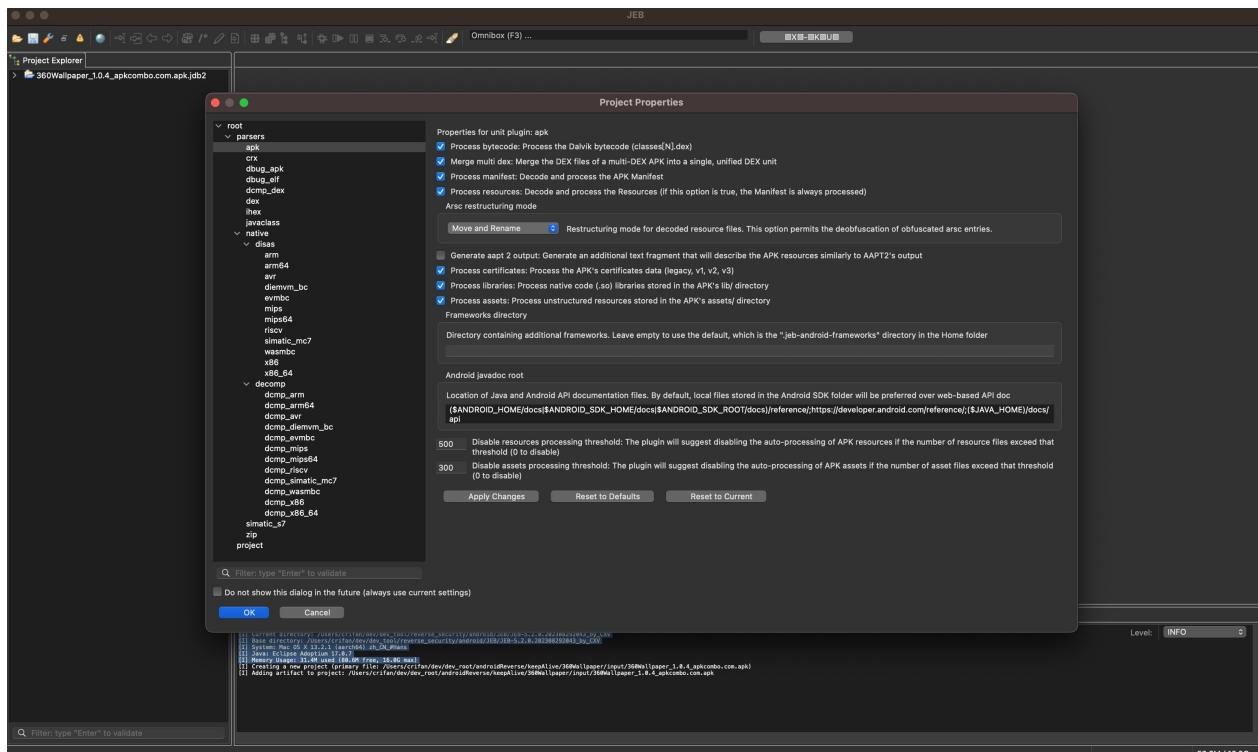
### 举例

#### 用JEB反编译360Wallpaper的apk

直接把 360Wallpaper 的 apk 文件，直接拖动到JEB中：

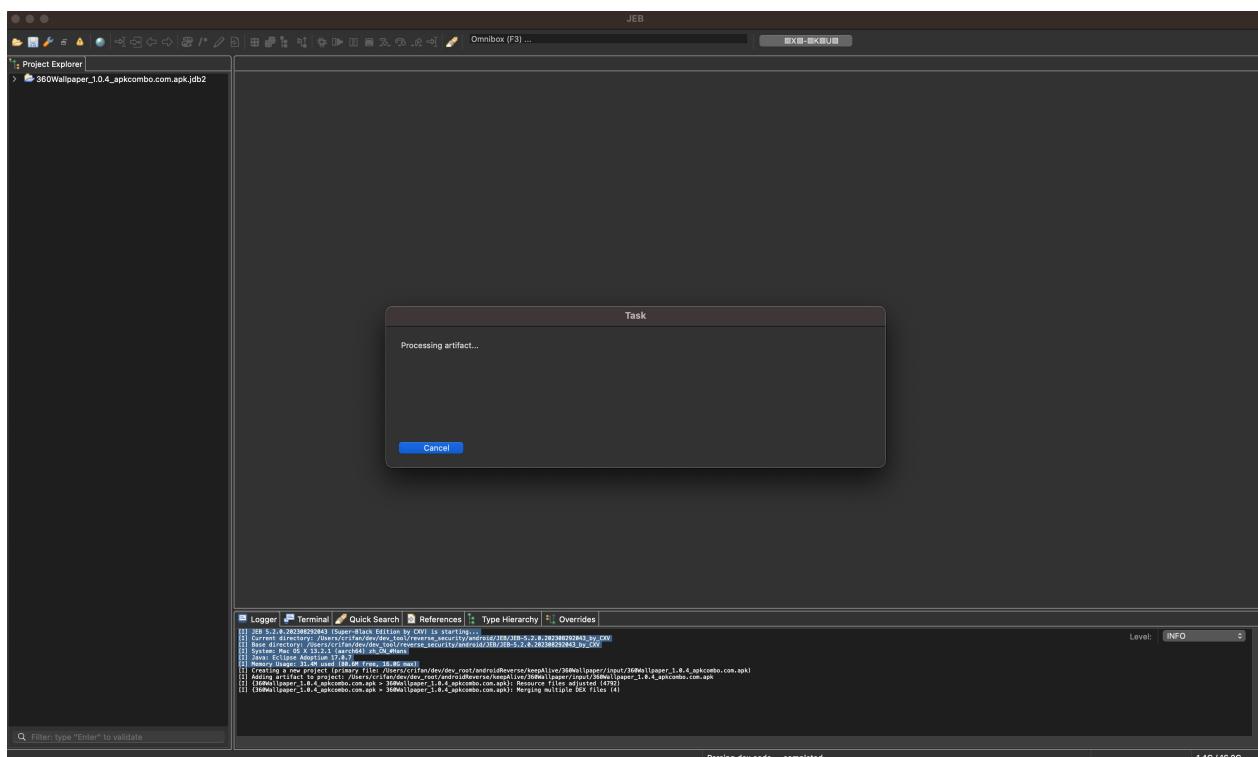


会弹框 Project Property :

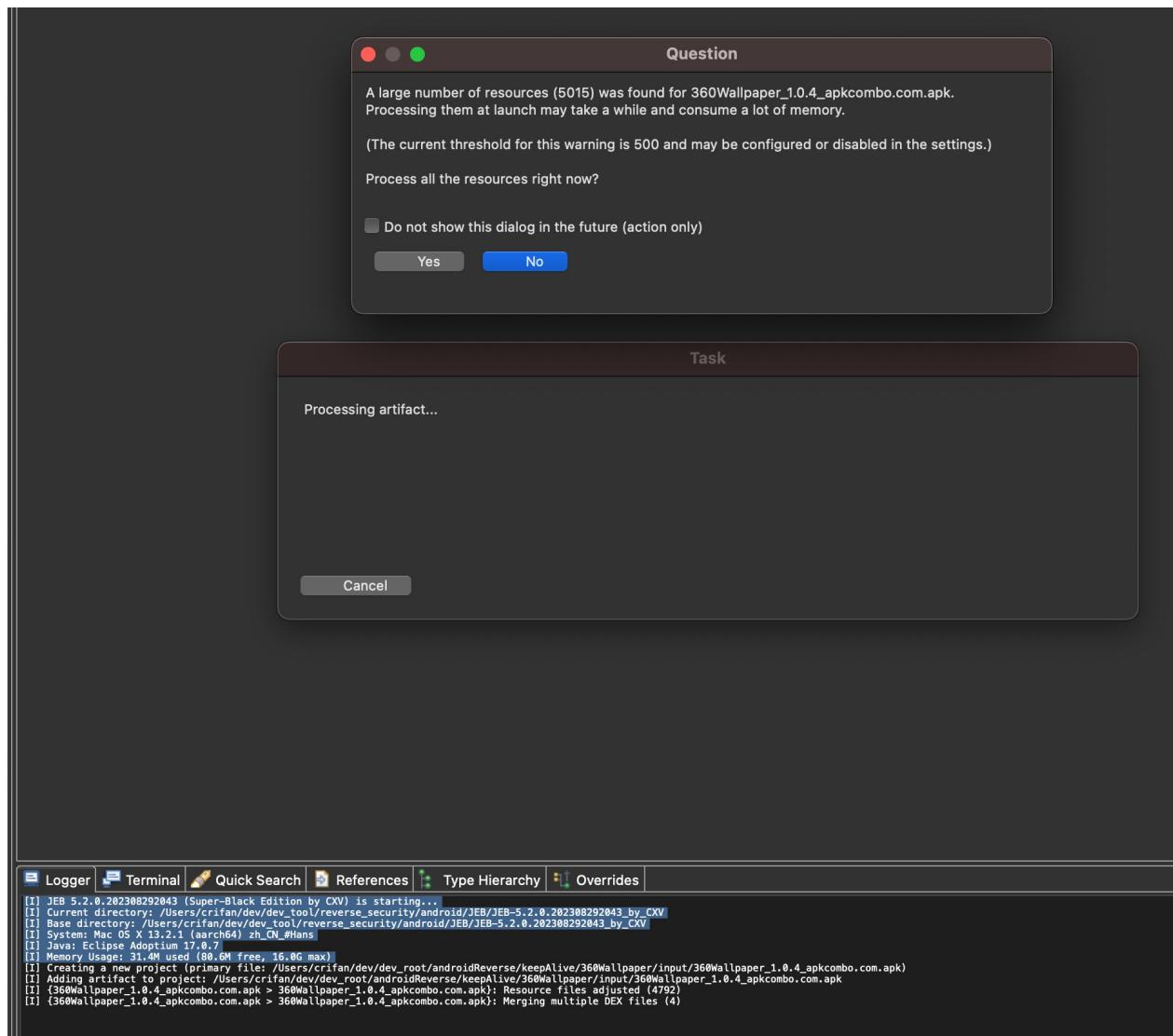


此处可以自动识别输入文件类型是： apk ， 所以自动选择了 apk 的配置，保持默认配置即可，点击： OK ， 继续反编译：

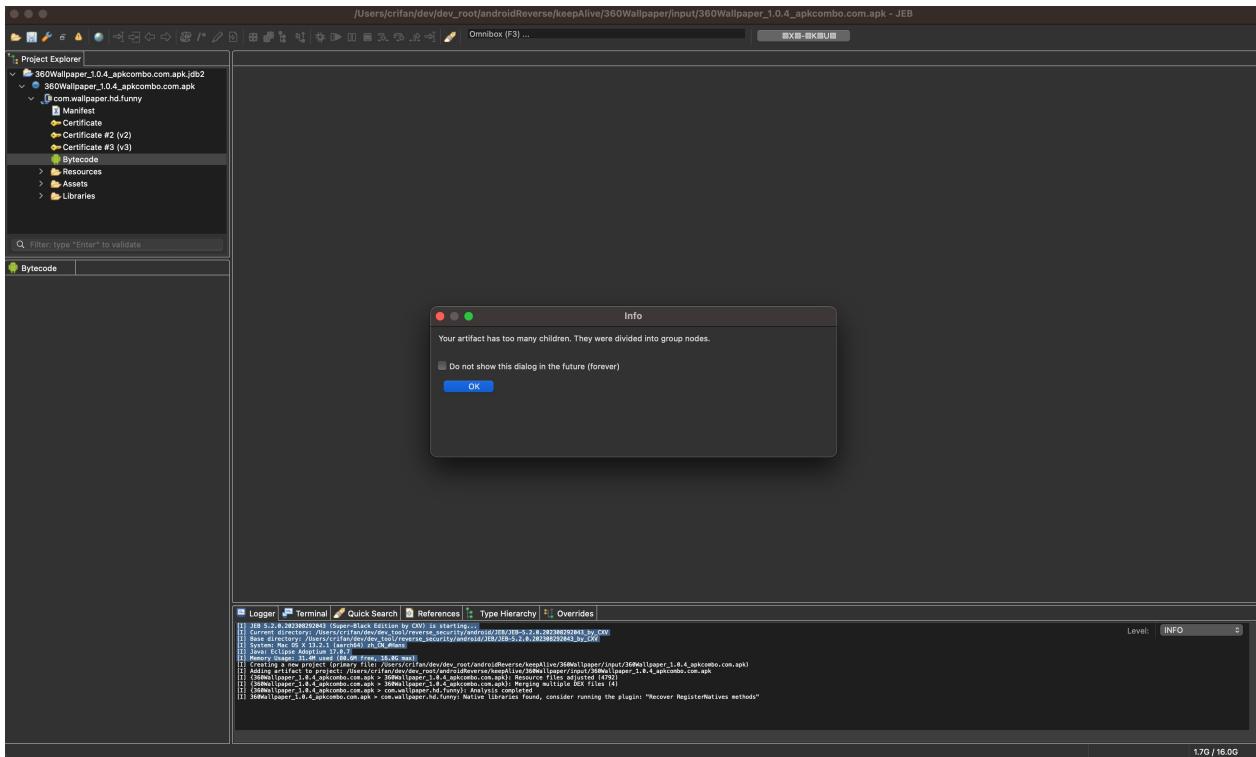
然后会弹框 Task ， 开始处理：



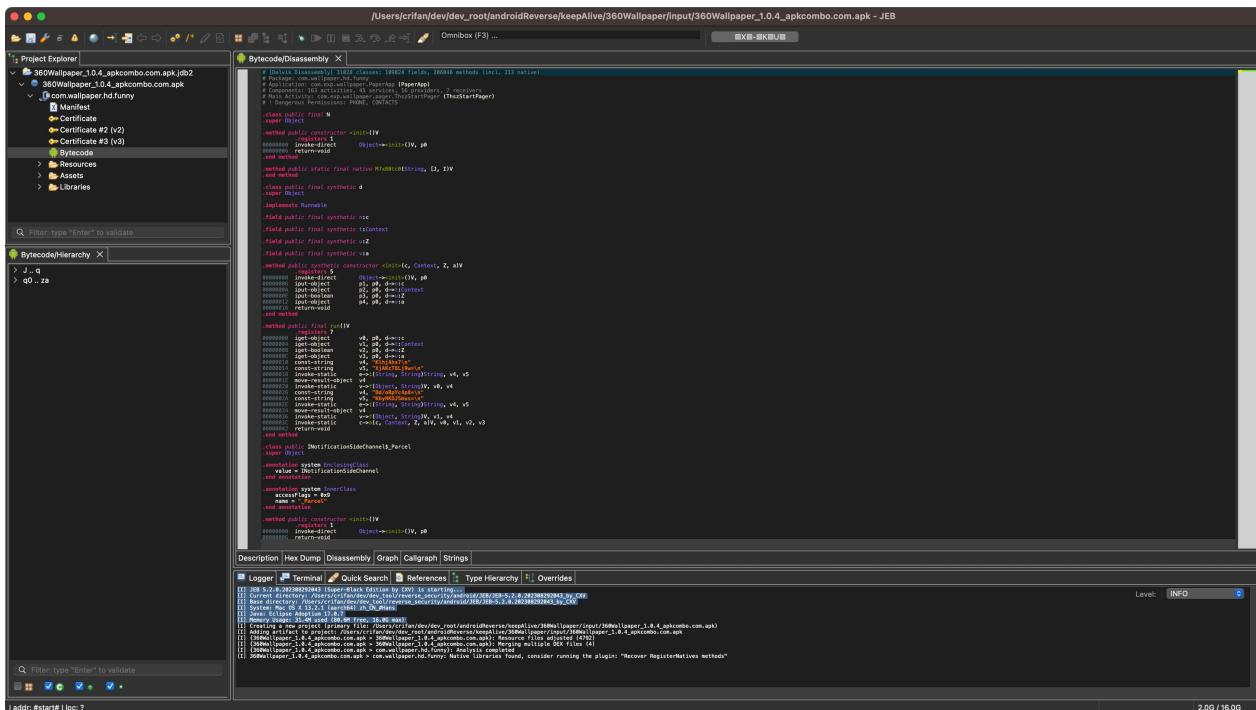
此处由于类太多，所以会提示，问是否继续，点击 Yes :



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。

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：  
2023-09-16 00:05:14

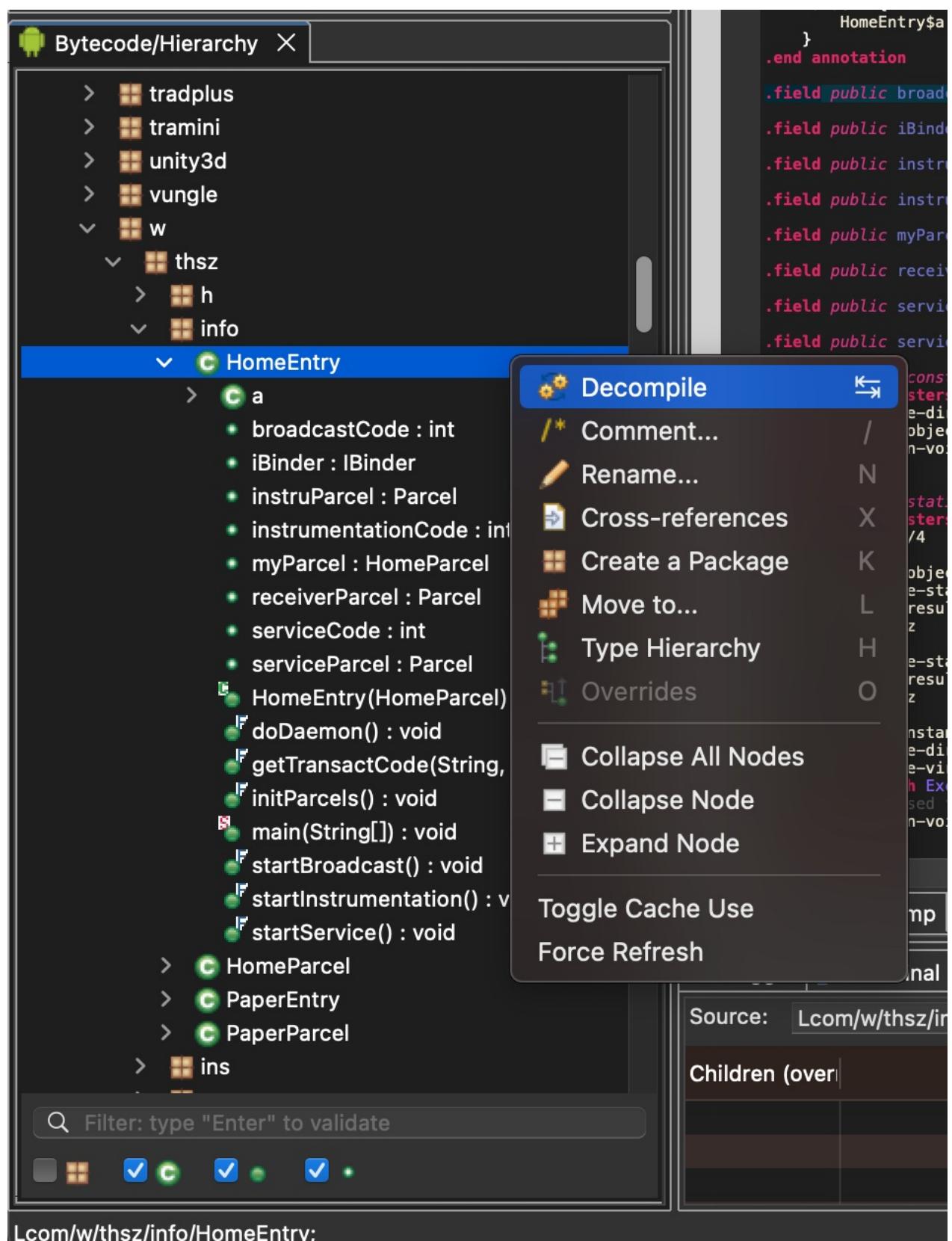
# 反编译出java

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：  
2023-09-16 12:52:20

## 单个反编译

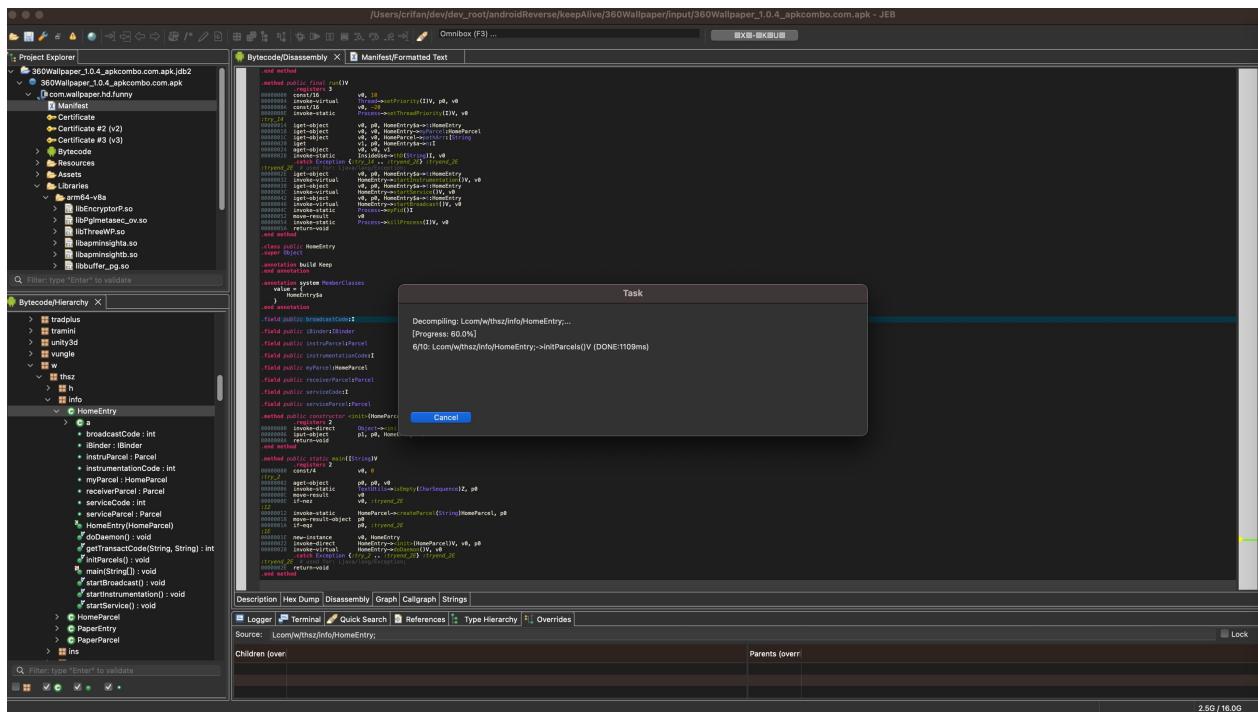
- 背景：JEB默认情况下反编译apk的话，显示的是Bytecode字节码：
  - 而不是我们要的 Java 代码
- 需求：对于单个类去反编译出java代码

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

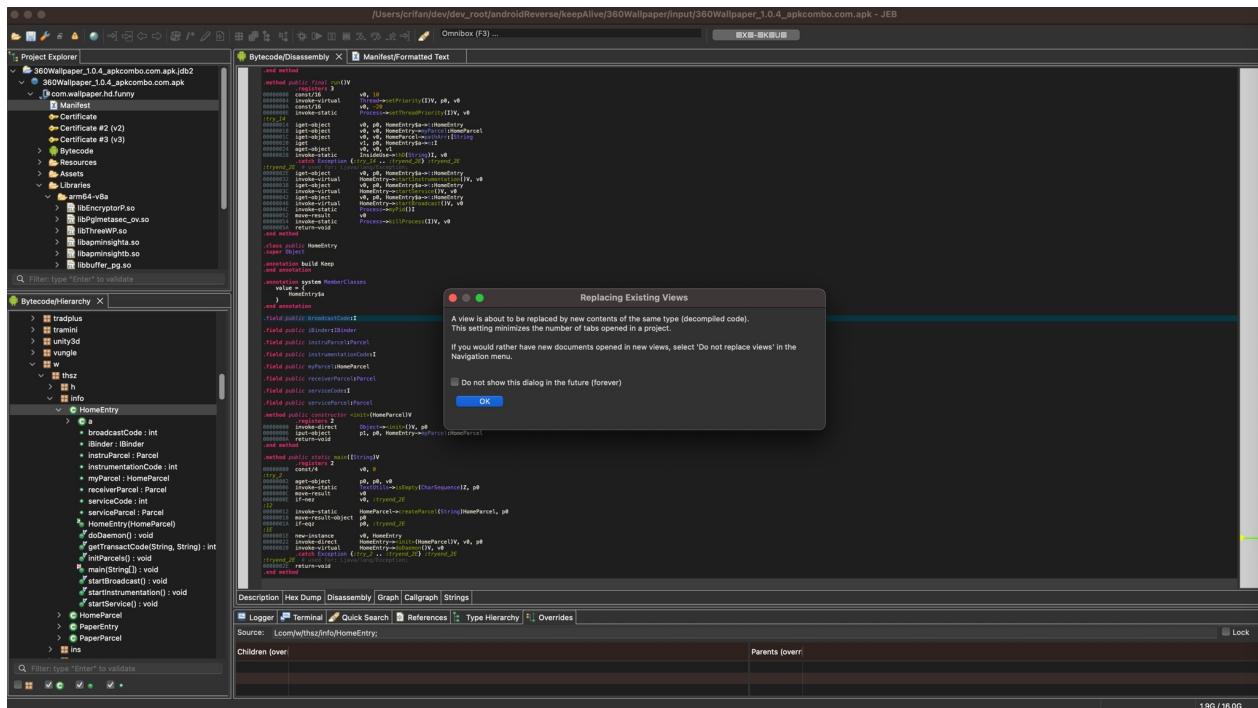


Lcom/w/thsz/info/HomeEntry;

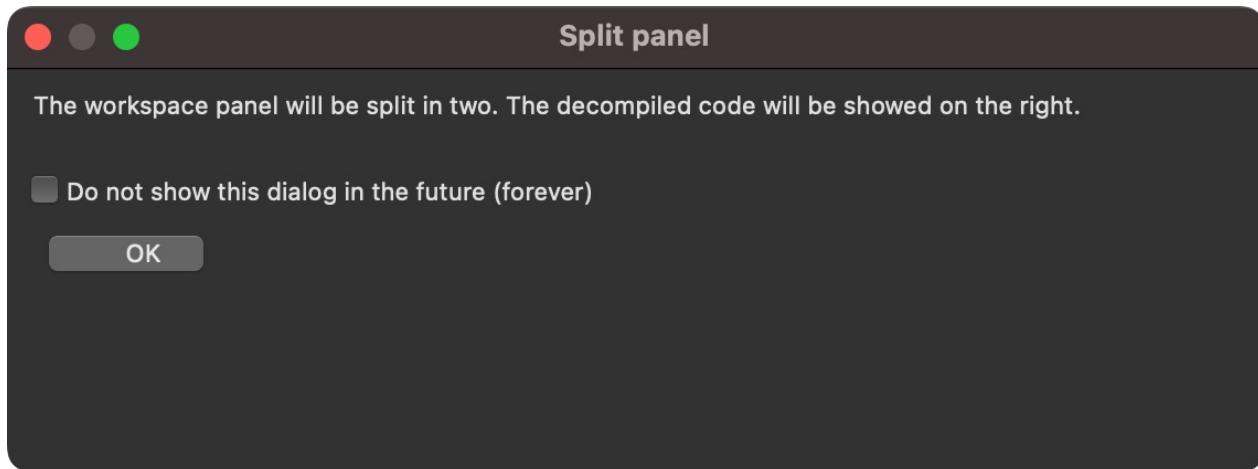
会弹框开始反编译：



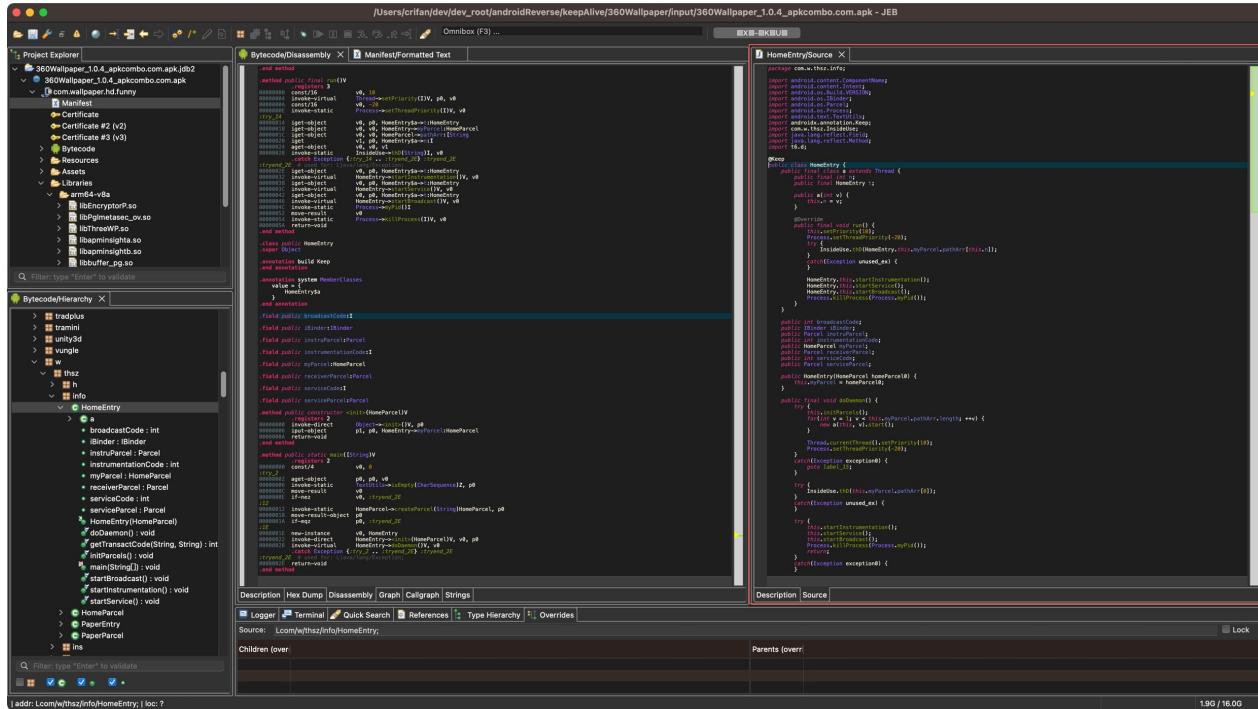
首次会有（勾选不再提示后，之后就不会再显示）额外的弹框提示，新的反编译的内容要显示在新的视图，但要替代当前视图：



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：



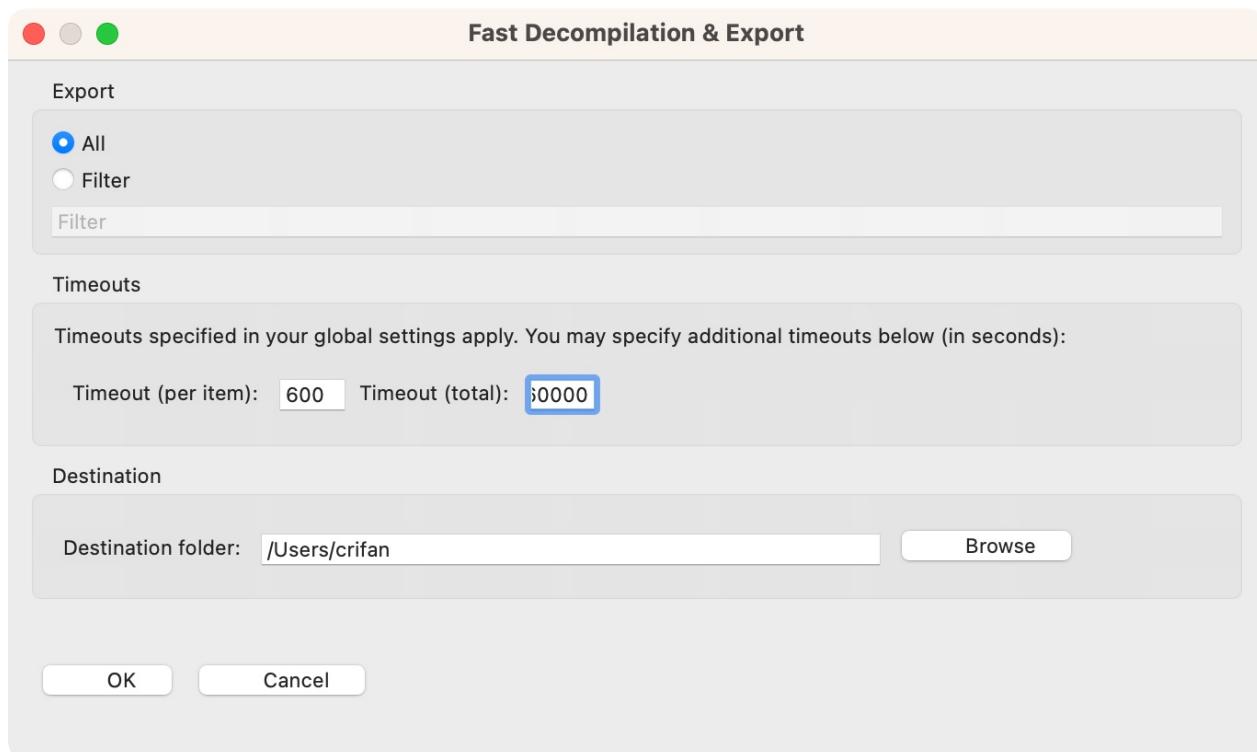
crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新:

2023-09-16 20:21:07

## 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`

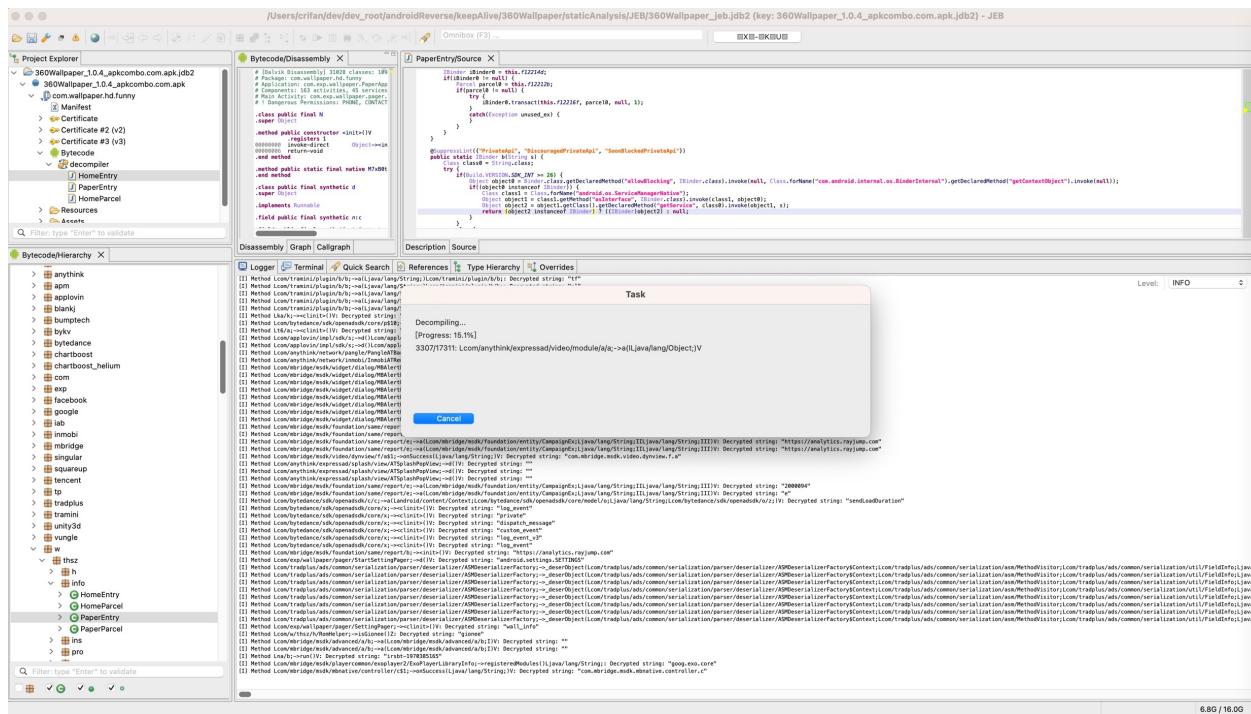
会弹框 `Fast Decomposition & Export` :



此处参数设置：

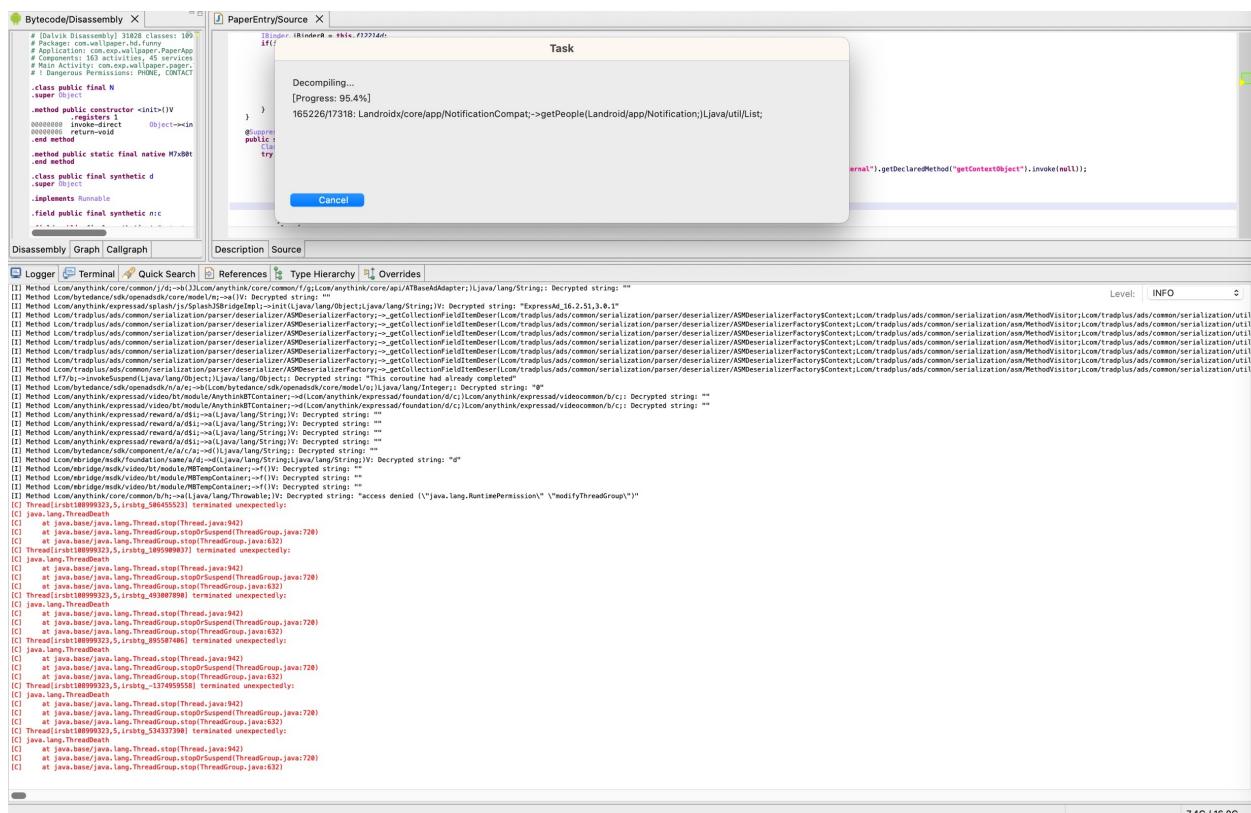
- `Export : All`
  - 说明：（反编译后）导出所有Java代码
- `Timeouts` : 反编译的超时设置
  - `Timeout per item : 600` =10分钟
  - `Timeout total : 60000` =100分钟
- `Destination` : 选择保存目标文件夹

然后就是开始反编译，显示进度的弹框了：



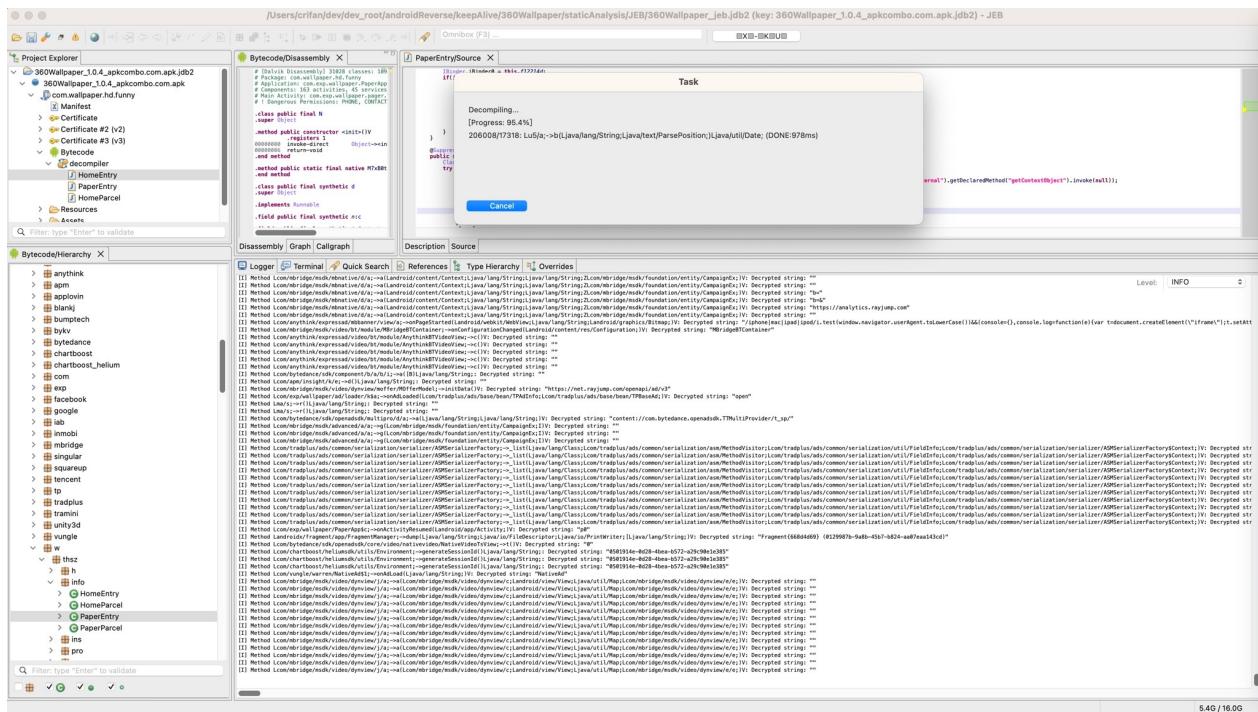
期间底部的Logger会输出对应的日子

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：

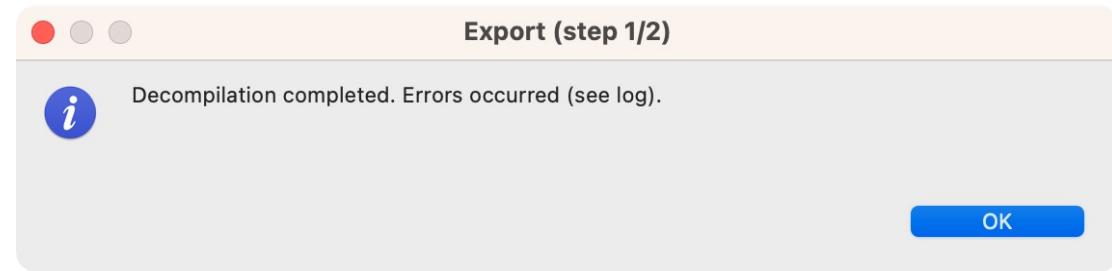


然后就：卡死了，等待很多分钟后，进度也没变化

此时只能：点击 Cancel 取消

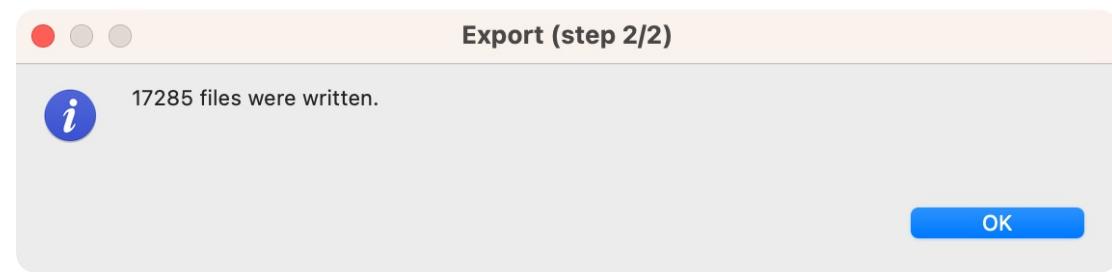
然后可以继续，会弹框提示：

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

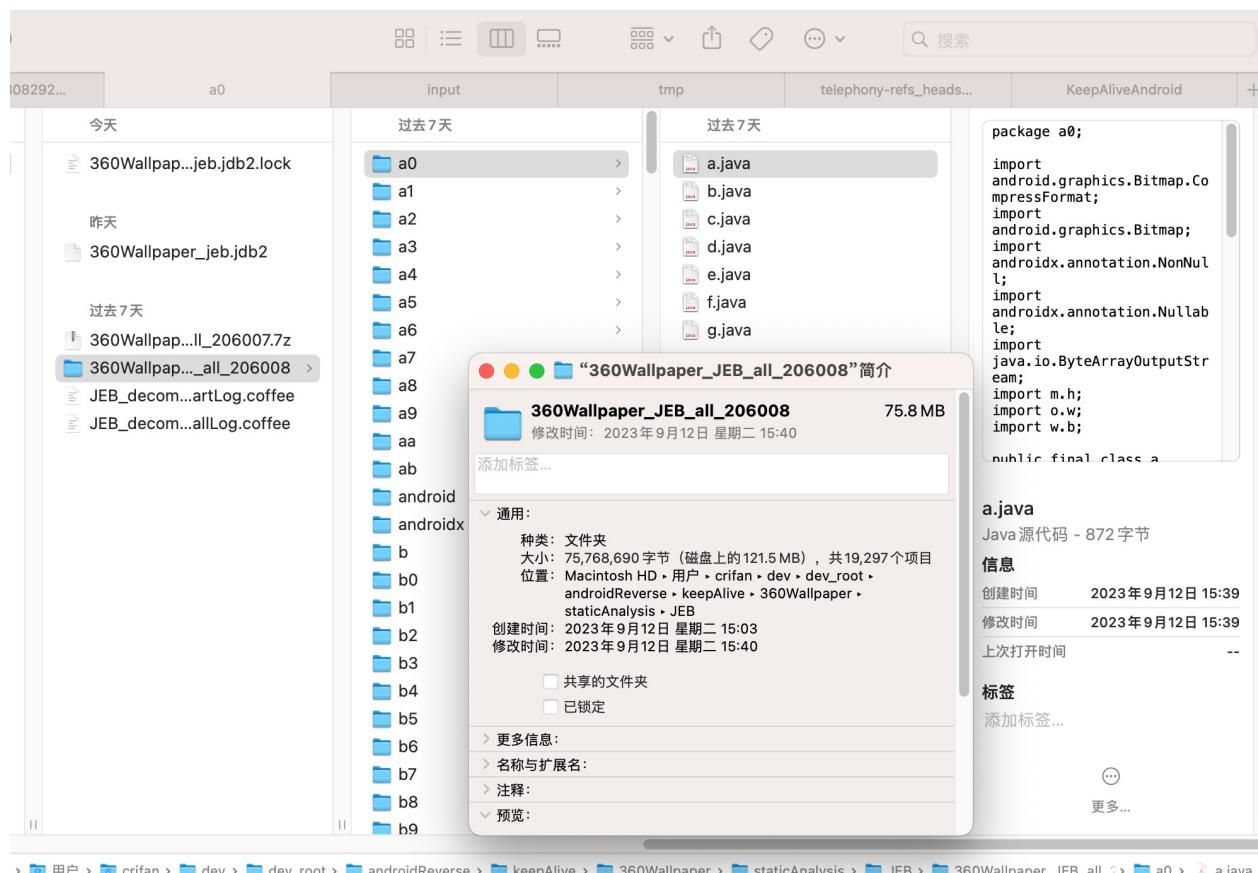


点击 OK 继续，显示：

- 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。

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:  
2023-09-16 21:33:23

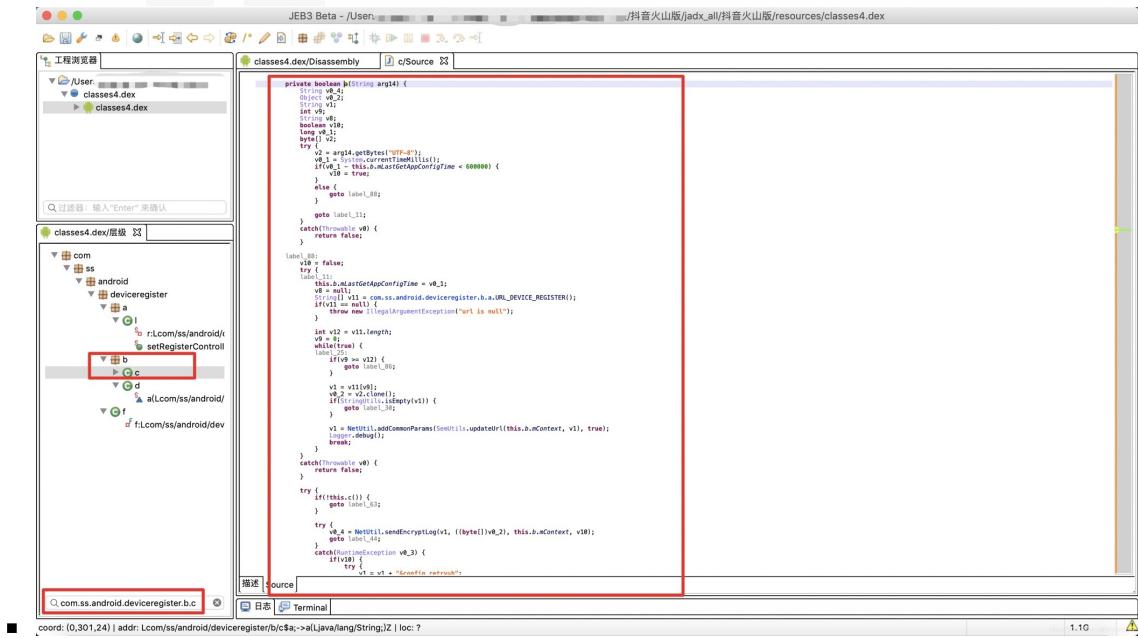
## 反编译效果对比

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



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：

2023-09-16 20:33:24

# 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("97HxJBf07nI=\n", "h
NSFZwWpuEI=\n"), String.class).invoke(null, this.myParcel.processName
);

} catch (Exception e) {
    e.printStackTrace();
}

try {
    this.ibinder = (IBinder) Class.forName(e.i("GzUxplaokQgV
KHuHXL0DTxk+GLVXoJJDC==\n", "eltV1DnB9SY=\n")).getMethod(e.i("jNeOQ
4U47DWI1w==\n", "67LAE0BKmlw=\n"), String.class).invoke(null, e.i("h
QG5PXN9gEE=\n", "5GLNVAAU9Dg=\n"));
} catch (Exception e10) {
    e10.printStackTrace();
}

this.serviceCode = getTransactCode(e.i("GOED4uC5HiCf/AzzwIw+
oTjgJ97FkTy2\n", "TLNCrLP4X9M=\n"), e.i("61bSaFhtSRDrJNp5SW10B/g8wHt
PZ1Ma9w==\n", "uXKTogwyGlU=\n"));

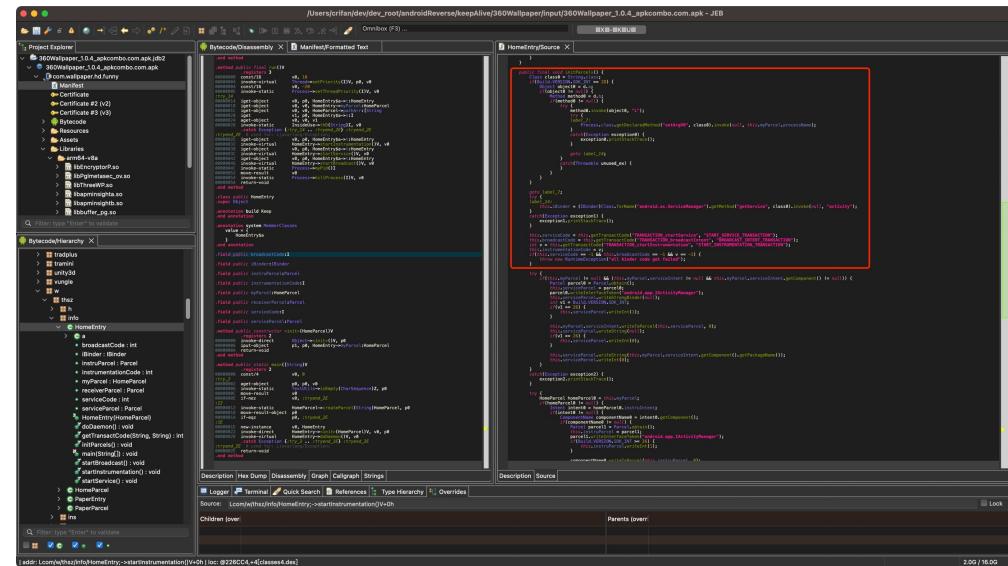
this.broadcastCode = getTransactCode(e.i("hDg9W0xEc3iZJTJKfx
cnTbQJHWzrTCZYtQQI\n", "0Gp8FR8FScw=\n"), e.i("qbf04YGsTom/usjukapBj
rSx0+GLvE6Zv6z07g==\n", "6+wBoMxVd9o=\n"));

int transactCode = getTransactCode(e.i("wnwPh06qs6bfaICVzp+R
gOJuolnJmYwf80m6q8mCn5w=\n", "1if0yr3r8PI=\n"), e.i("RGOVY5WGeT9EY4Z
kjJx+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.ServiceManager")
.getMethod("getService", class0).invoke(null, "activity");

this.serviceCode = this.getTransactCode("TRANSACTION_startService",
"START_SERVICE_TRANSACTION");
this.broadcastCode = this.getTransactCode("TRANSACTION_broadcastIntent",
"BROADCAST_INTENT_TRANSACTION");
int v = this.getTransactCode("TRANSACTION_startInstrumentation",
"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 ;
```

```

@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/hF0vxgTGbL5S3y5AUkpvtRdKgNT
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("97HxJBf07nI=\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", "eltViDnB9SY=\n")).getMethod(e.i("jNe0Q4U47DWI1w==\n", "67LAE0BKmlw=\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("6ibSaFhtSRDrJNp5SW10B/g8wHtPZ1Ma9w==\n", "uXKT0gwyGlU=\n"));
    this.broadcastCode = getTransactCode(e.i("hdg9W0xEc3iZJTJKfXcnTbQJHWZrTCZYtQQI\n", "0Gp8FR8FSCw=\n"), e.i("qbf04YGsTom/usjukapBjrSx0+GLvE6Zv6z07g==\n", "6+wBoMXvD9o=\n"));
    int transactCode = getTransactCode(e.i("wnWPh06qs6bfaICVzp+RgOJuoLnJmYwf80m6q8mCn5w=\n", "lif0yr3r8PI=\n"), e.i("RG0VY5WGeT9EY4ZkjJx+JVZjnX6PhmQjVnmHcIKNeT5Z\n", "FzfUmchZMHE=\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", "CpogdR2AyRI=\n"));
    }
    try {
        HomeParcel homeParcel = this.myParcel;
        if (homeParcel != null && (intent3 = homeParcel.serviceIntent) != null && intent3.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/d60/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.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

o

◦

◦

◦

- 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", "BROADCAST_INTENT_TRANSACTION");
        int v = this.getTransactCode("TRANSACTION_startInstrumentation", "START_INSTRUMENTATION_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.myParcel.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);
        this.receiverParcel.writeInt(0);
    }
}
catch(Exception exception4) {
    exception4.printStackTrace();
}

Insidelise.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) {
            }
        }
    }
}

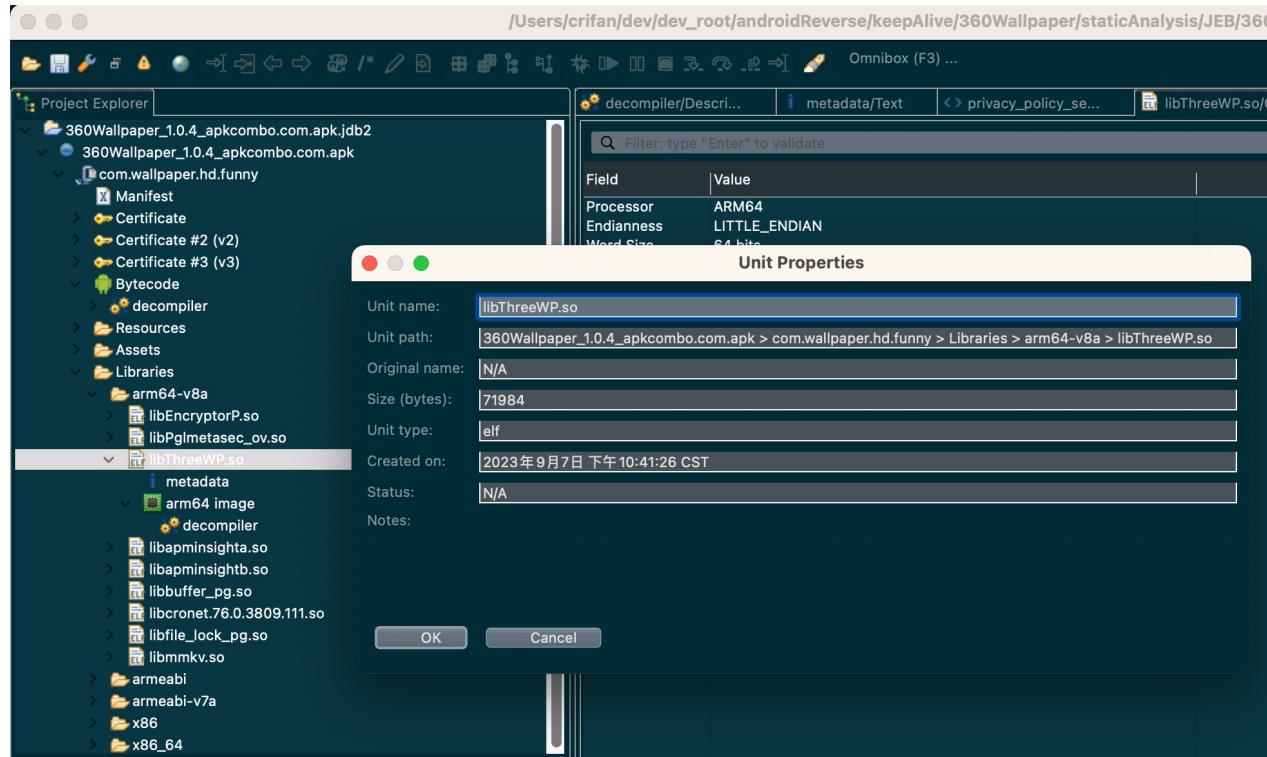
public final void startService() {
    Parcel parcel0 = this.serviceParcel;
    if(parcel0 != null) {
        IBinder iBinder0 = this.iBinder;
        if(iBinder0 != null) {
            try {
                iBinder0.transact(this.serviceCode, parcel0, null, 1);
            }
            catch(Exception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



# 解析so库文件

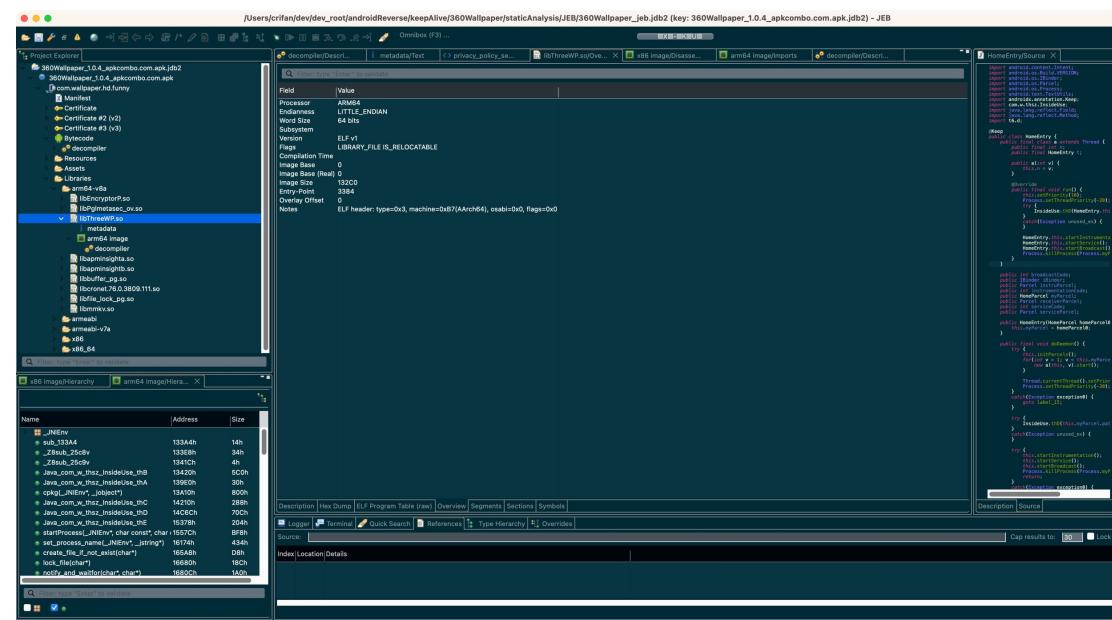
继续查看，JEB反编译apk后，对于其中的so库文件的解析。

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

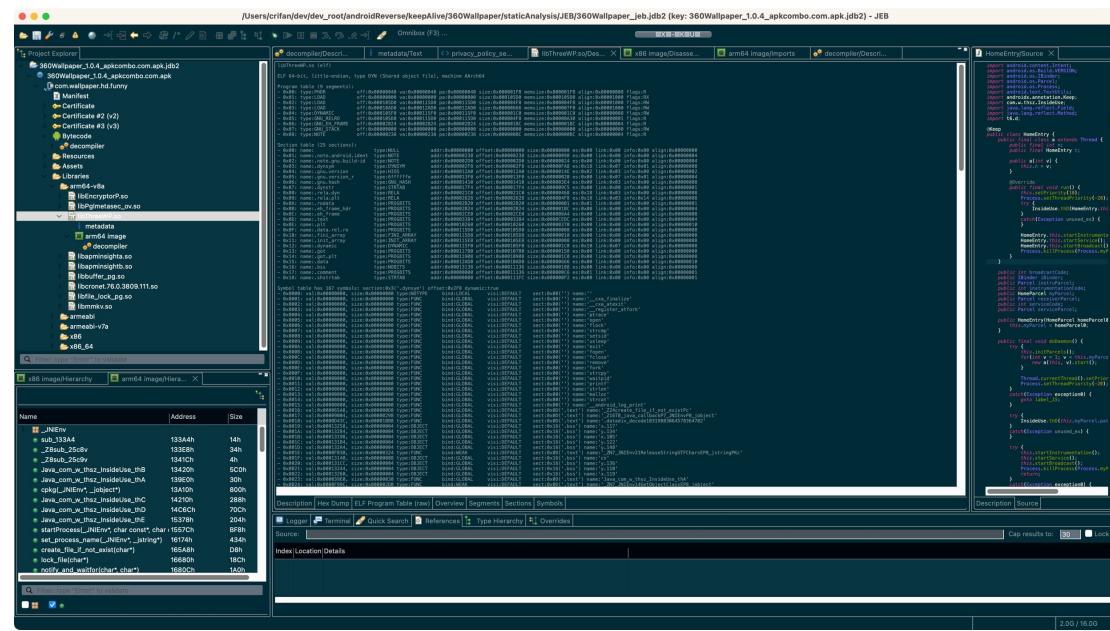


然后可以分别查看解析效果：

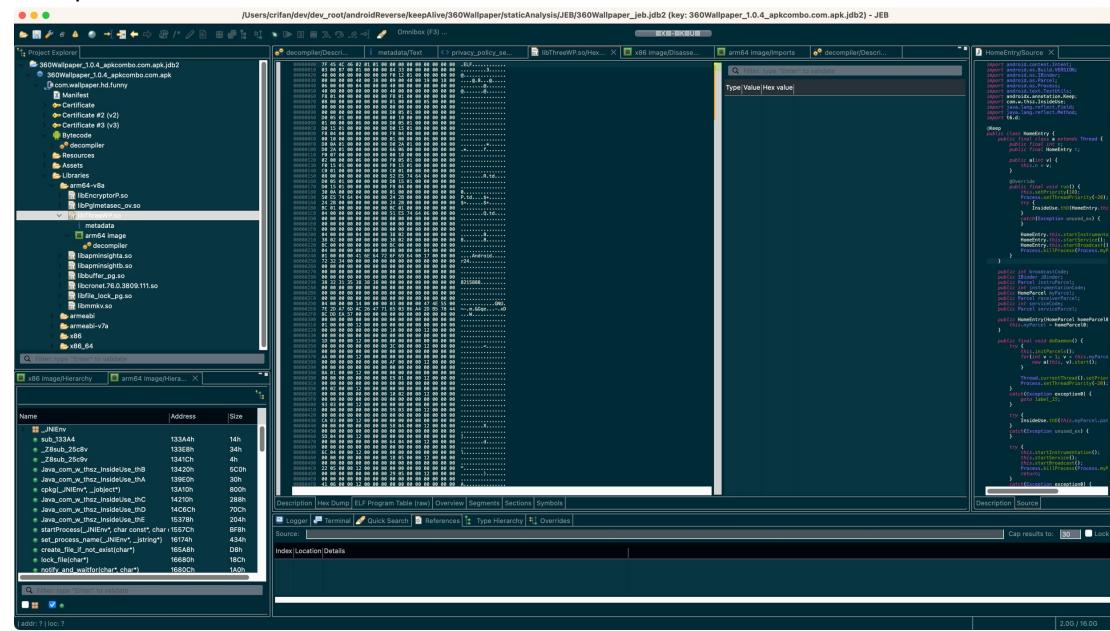
- (ELF格式的) so库文件
  - Overview



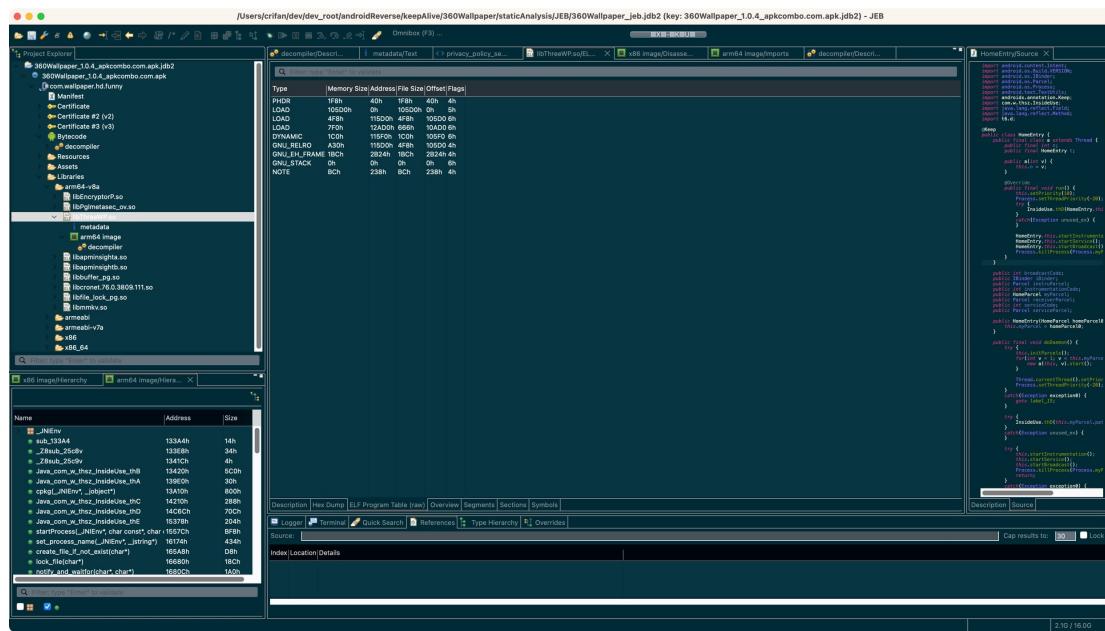
- Description



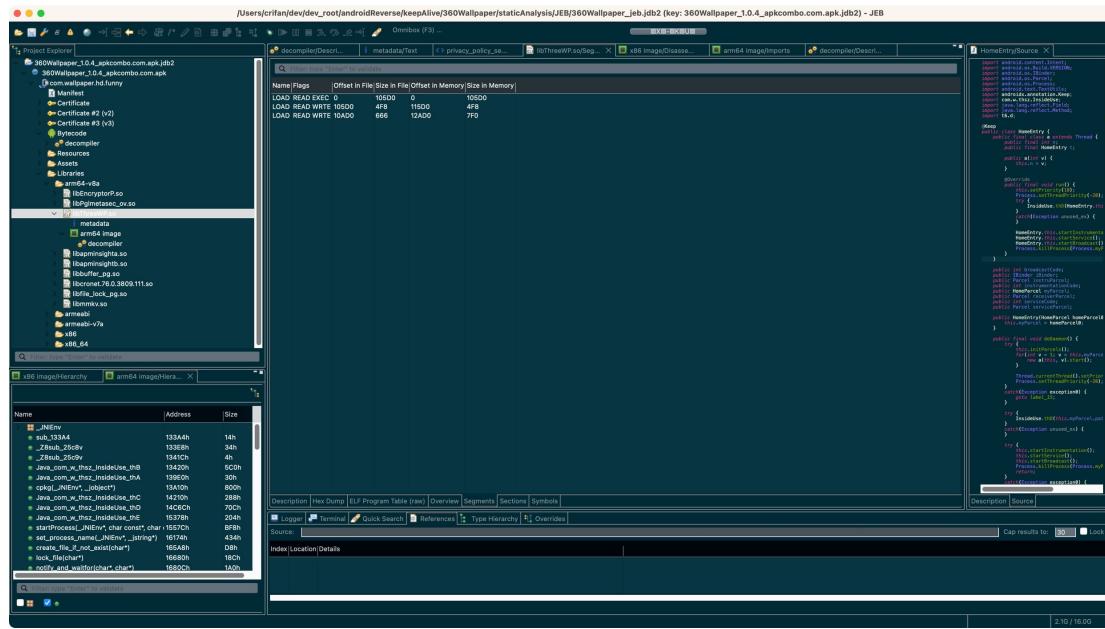
- Hex Dump



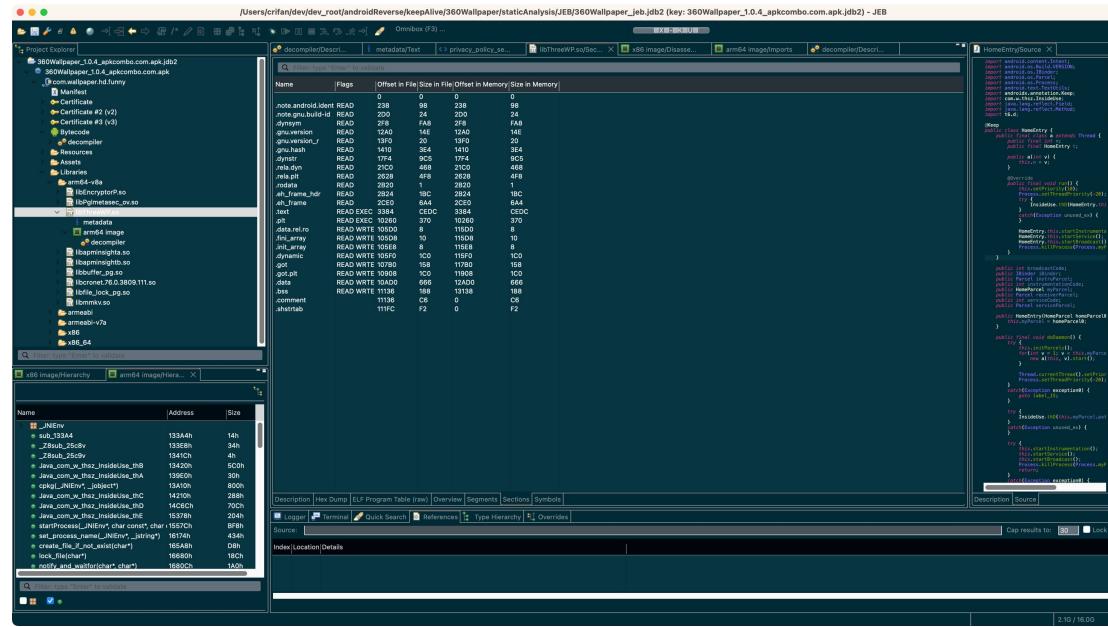
- ELF Program Table (raw)



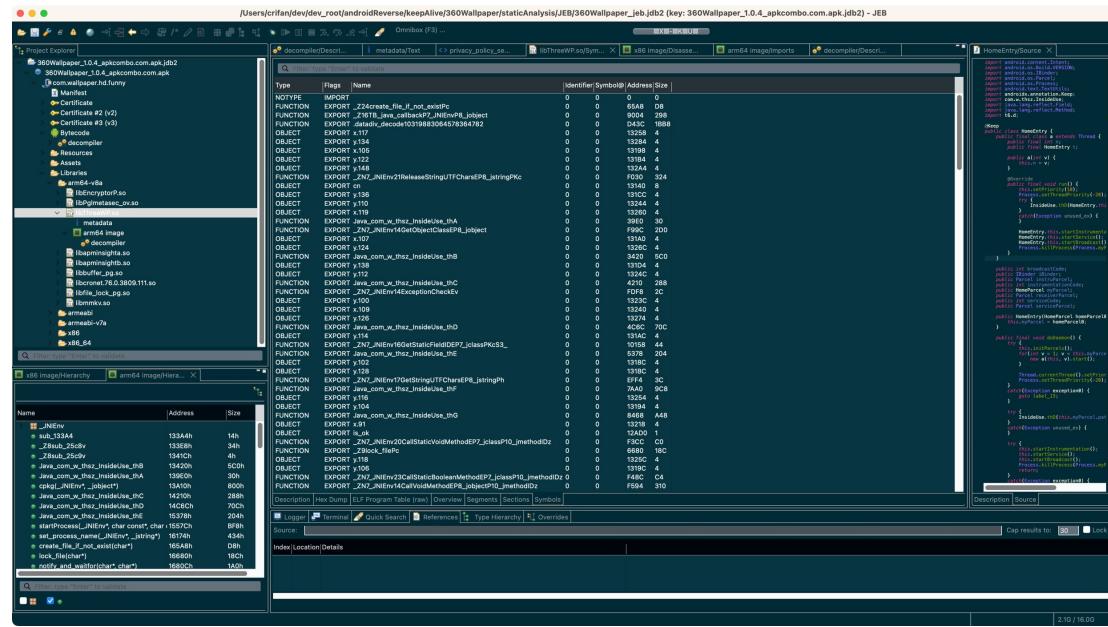
- Segments



- Sections

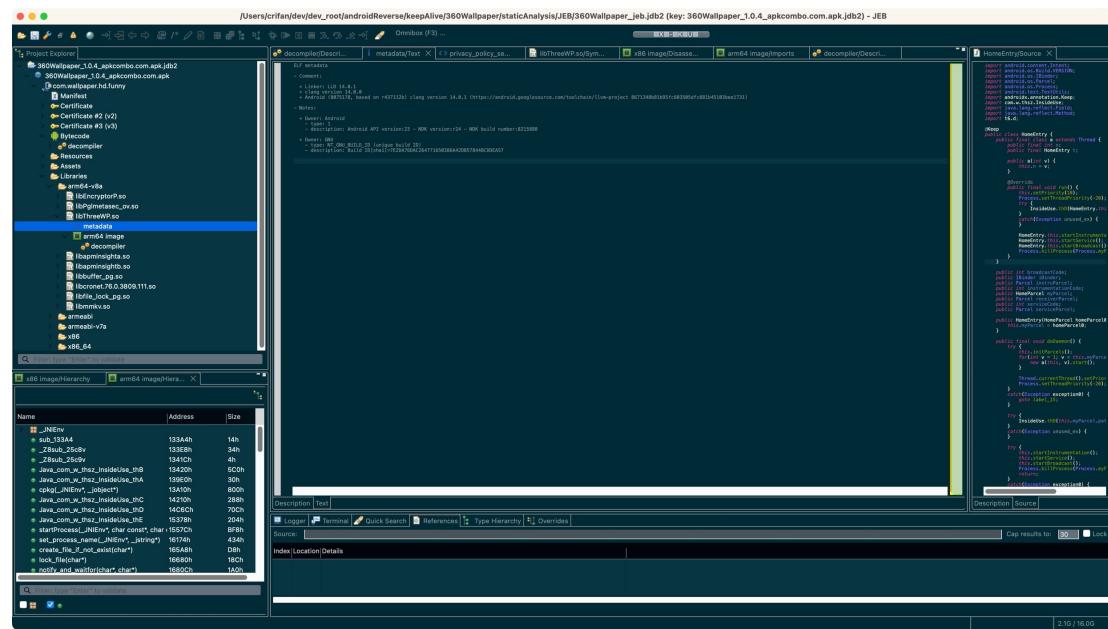


- Symbols

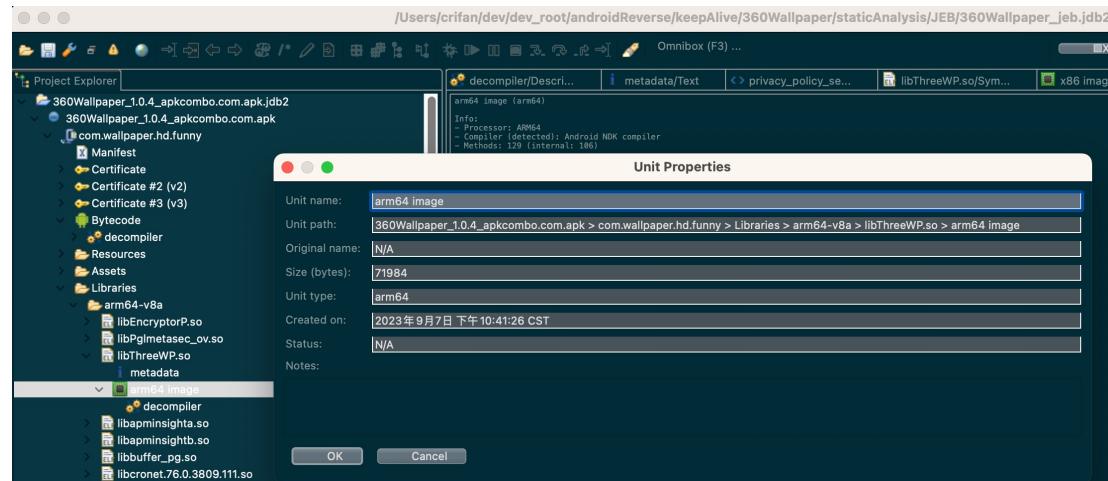


- so下面还有2个子项

- metadata

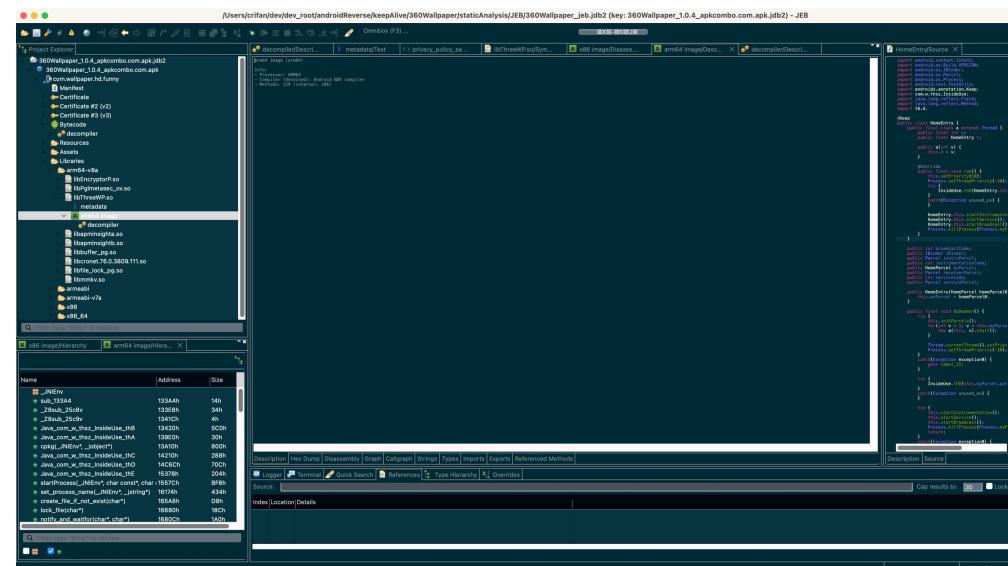


- arm64 image



▪ 其也有各个子页面

- Description



### arm64 image (arm64)

**Info:**

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

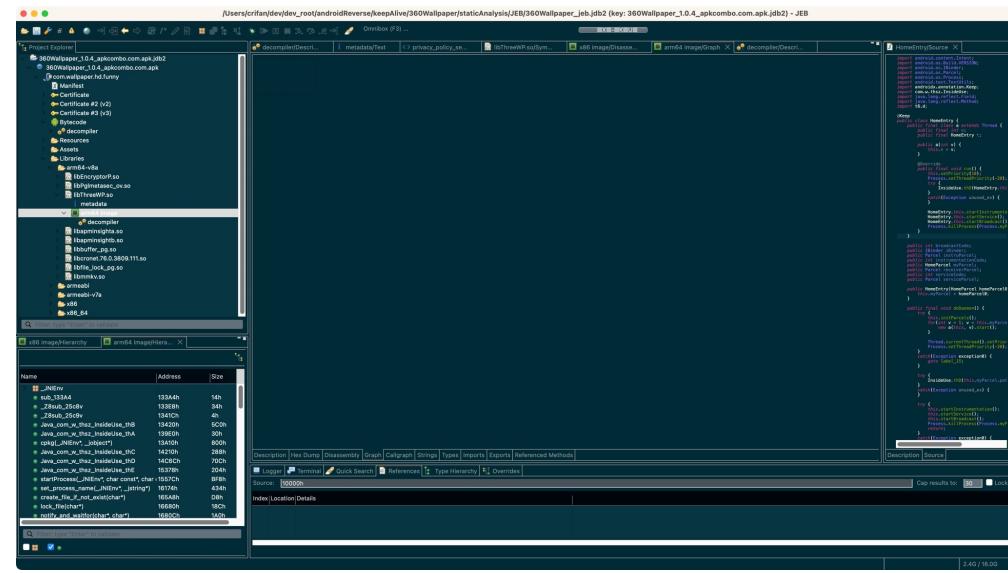
#### ■ Hex Dump

The screenshot shows the JEB static analysis interface. The top navigation bar includes 'Project Explorer', 'decompiler(Dex)', 'metadata/test', 'privacy\_policy.ee...', 'sdBtHewWP.sxSym', 'arm64 imageDiss...', 'arm64 imageHex Dump', and 'decompiler(Dex...)'. The left sidebar shows the project structure for '360Wallpaper\_1.0.4\_mpkcombo.com.apk.jdb2' with sections like Manifest, Certificate # (v2), Certificate # (v3), Resources, Assets, Libraries, and Decompile. The main area displays a large hex dump of the APK file, with the address column on the left and the hex value column on the right. Below the dump are tabs for 'Description', 'Hex Dump', 'ClassAssembly', 'Graph', 'Callgraph', 'Storage', 'Types', 'Imports', 'Exports', and 'Referenced Methods'. The bottom status bar indicates 'Index:Location|Details' and '2.00 / 16.00'.

#### ■ Disassembly

The screenshot shows the JEB static analysis interface with the 'disassembly' tab selected. The left sidebar and main area are identical to the hex dump view, showing the project structure and the large assembly dump. The bottom status bar indicates 'Index:Location|Details' and '2.00 / 16.00'.

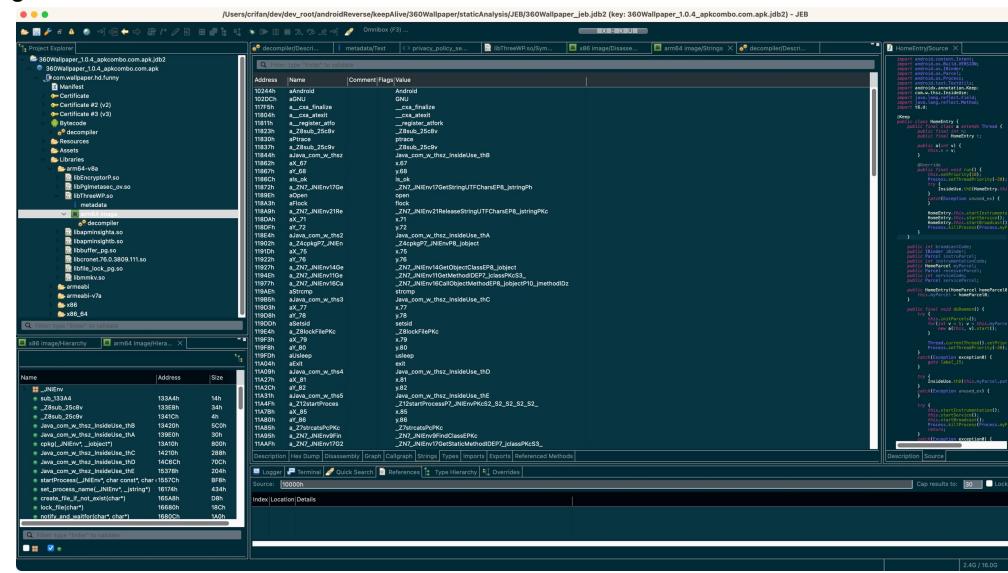
#### ■ Graph



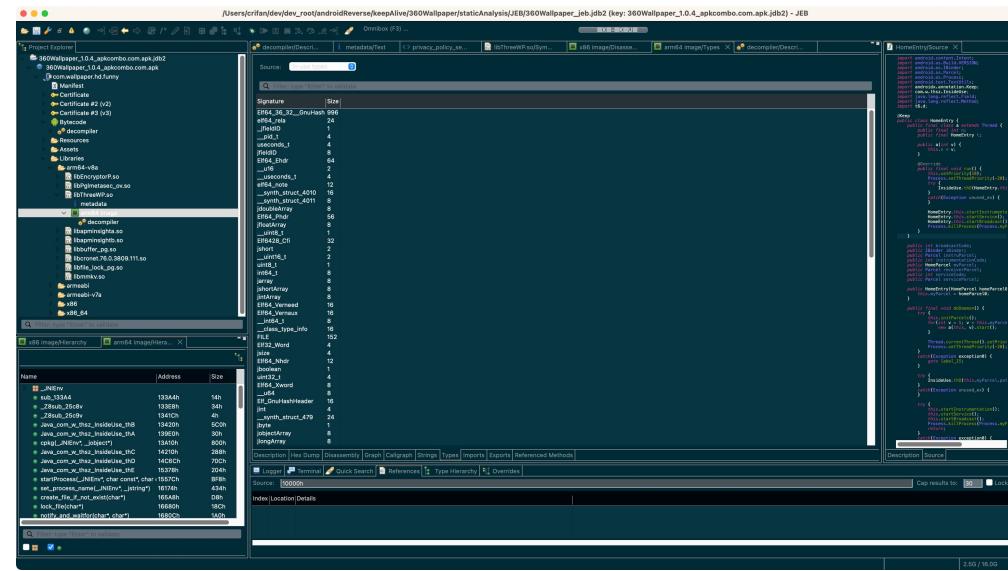
## ■ Callgraph

- 注：容易导致卡死
- TODO：有空再去试试

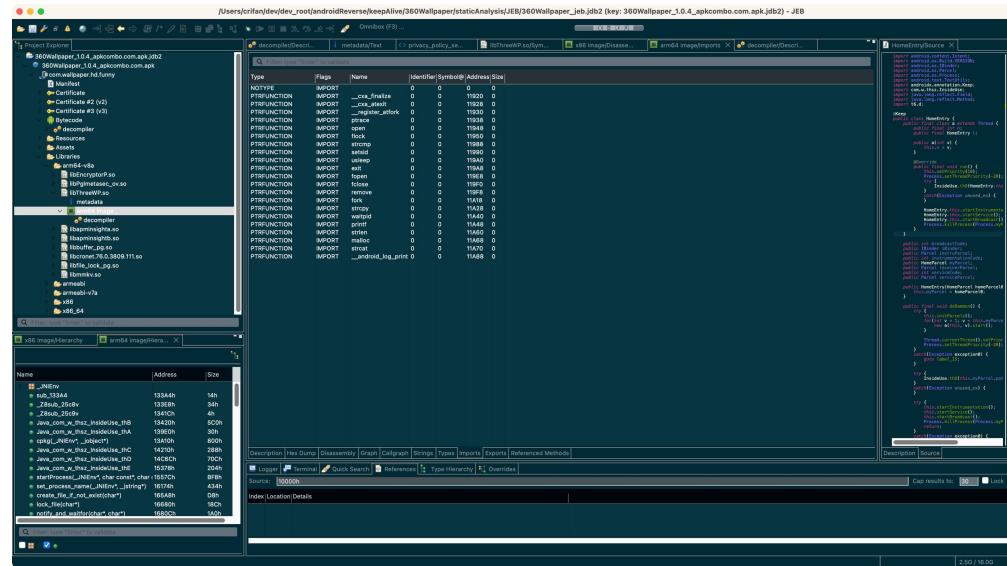
## ■ Strings



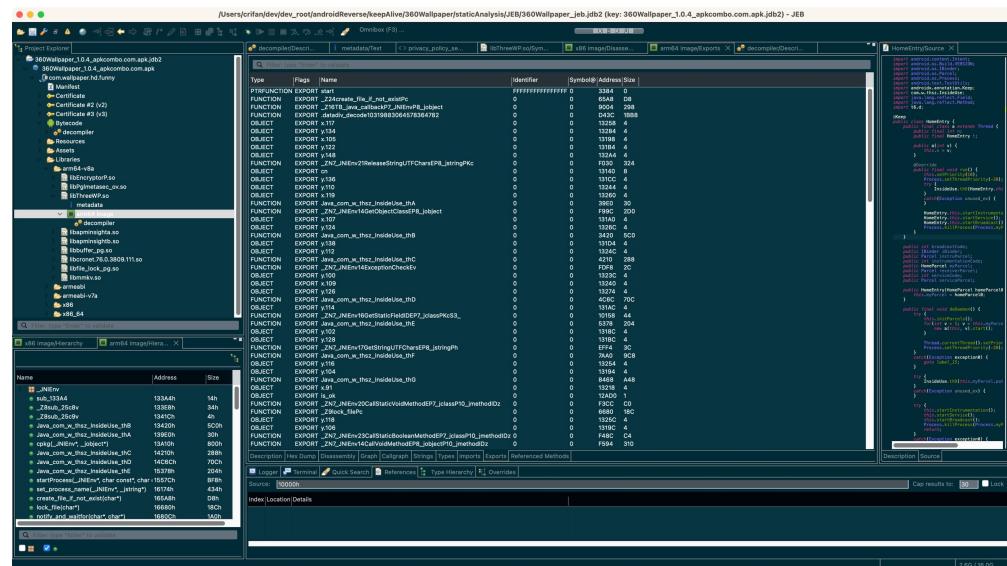
## ■ Types



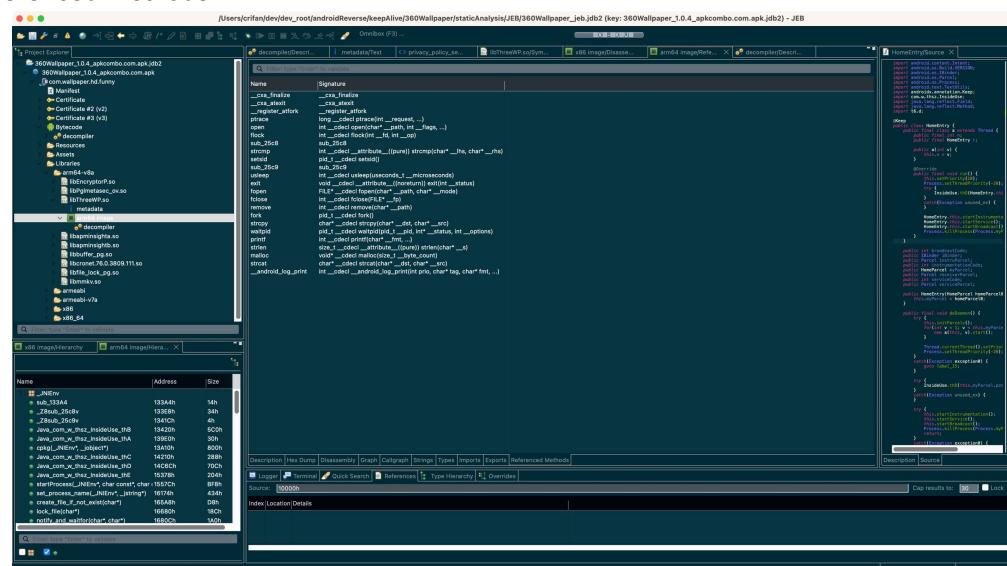
## ■ Imports



## ■ Exports

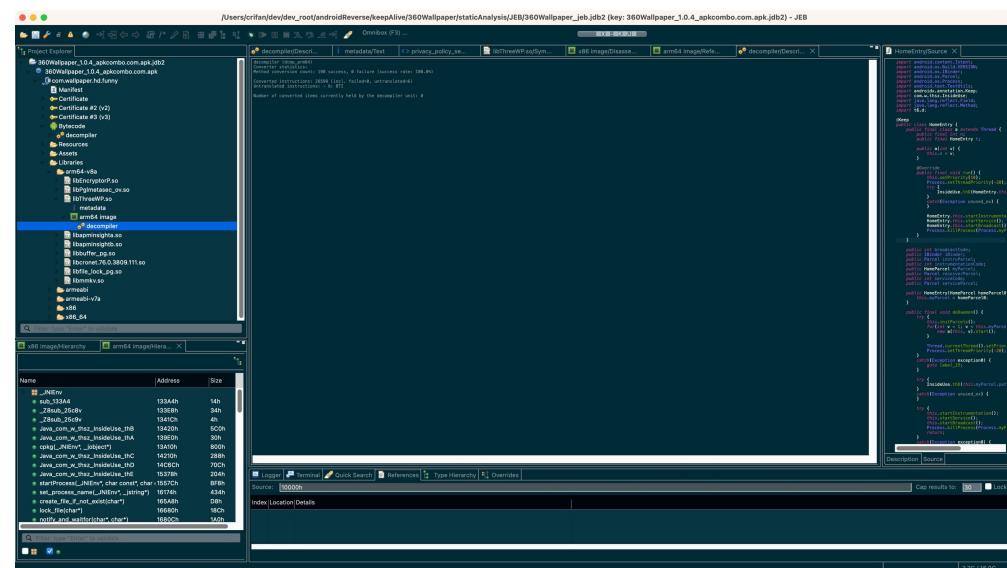


## ■ Referenced Methods



## ■ 其下也有子项

### ■ decompiler



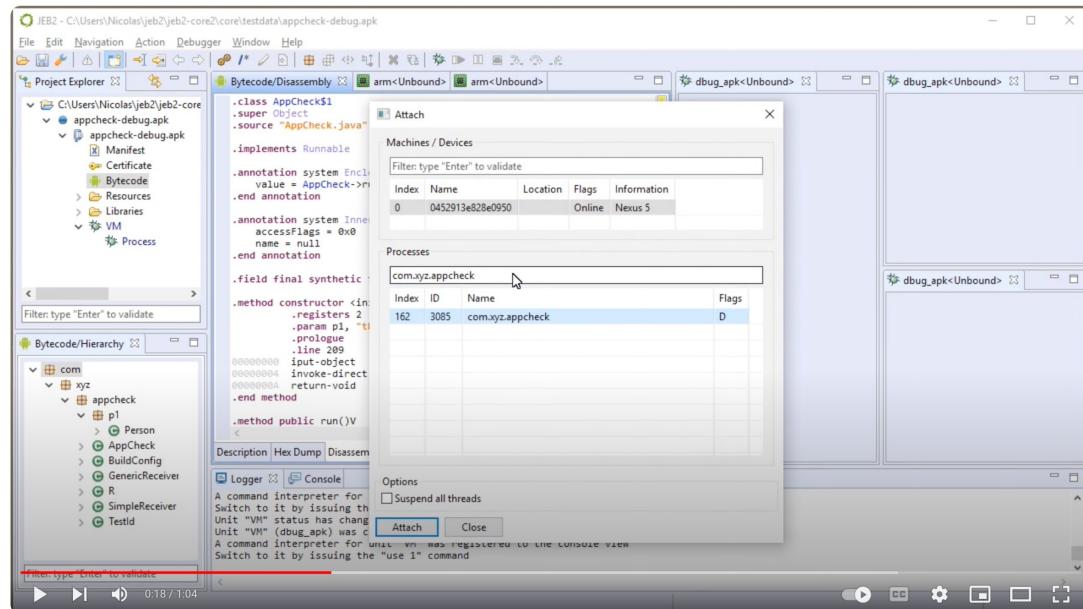
crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新:  
2023-09-16 21:31:47

# JEB动态调试

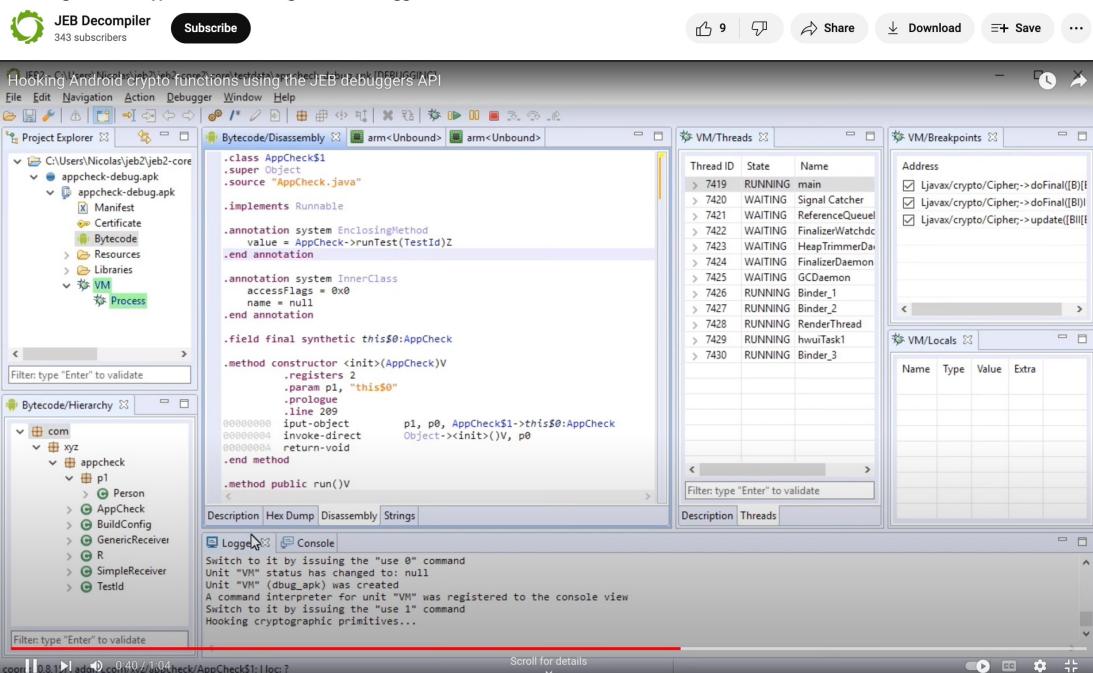
先贴出一些：

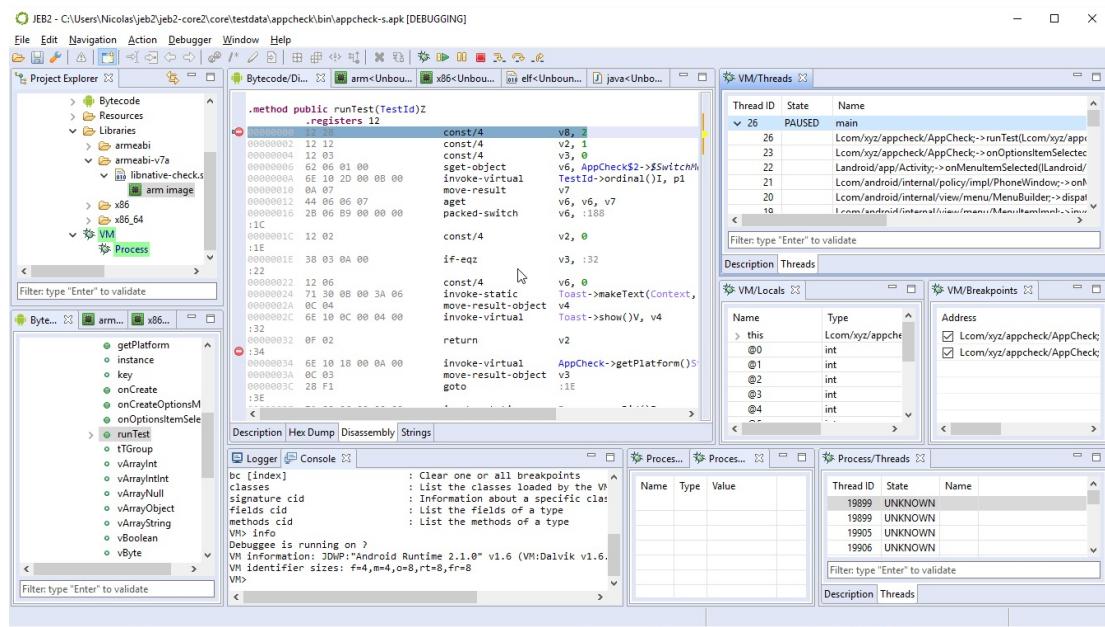
- JEB动态调试

- 相关截图

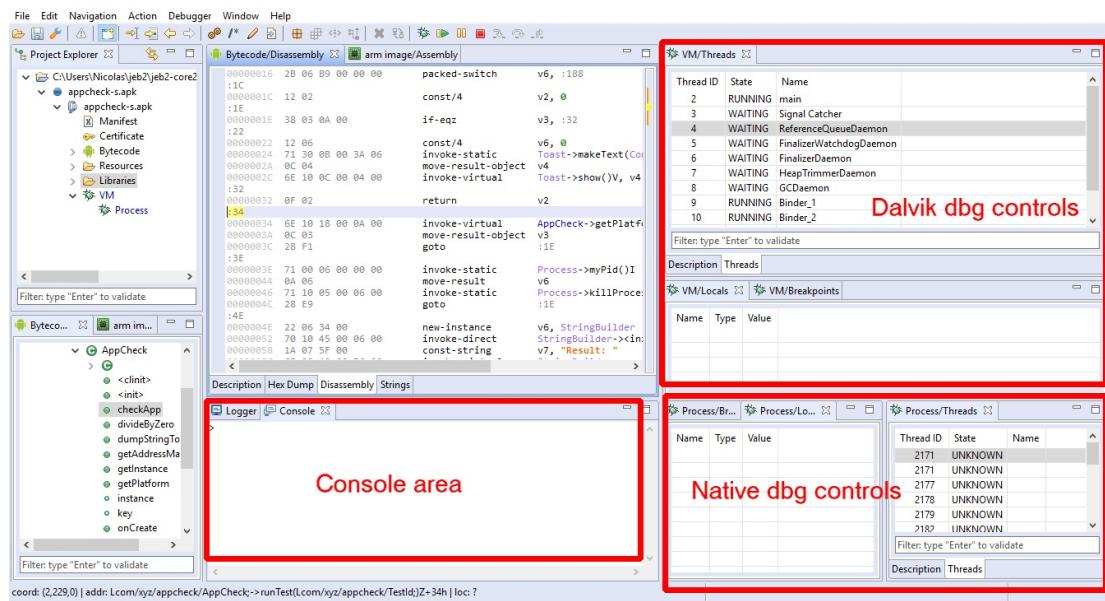


Hooking Android crypto functions using the JEB debuggers API





## ○ 调试界面功能模块介绍



**Threads**

Thread ID	State	Name
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

**Local variables and registers**

Name	Type	Value
@3	int	0 / 0h
@4	int	1 / 1h
@5	int	0 / 0h
@6	int	0 / 0h

**Breakpoints**

Address
Lcom/xyz/appcheck/AppCheck; -> runTest

**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/Menut
21		Lcom/android/internal/policy/impl/PhoneWindow; -> onMenuItemSel
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; -> performItemAction

Filter: type "Enter" to validate

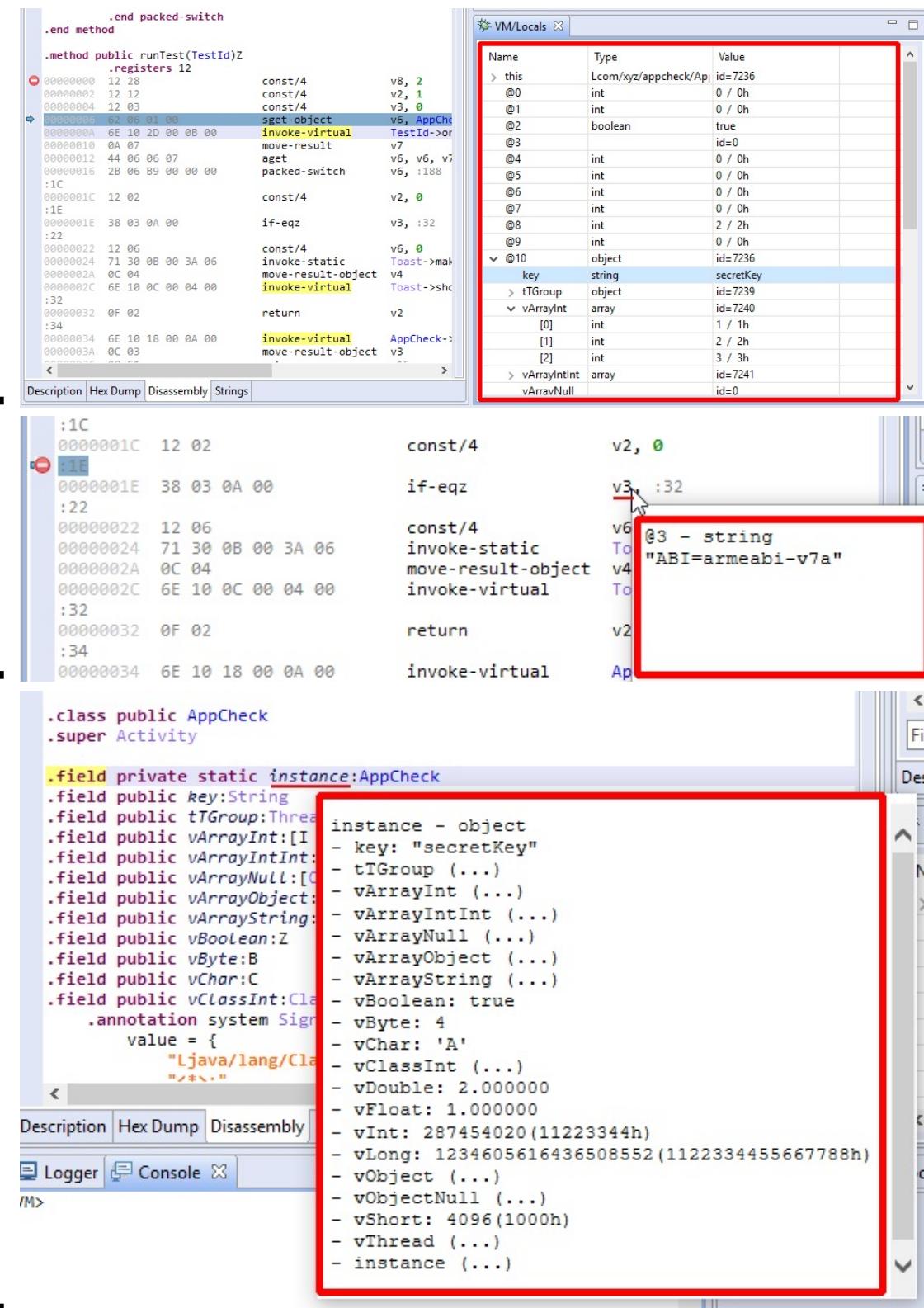
Description Threads

**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。

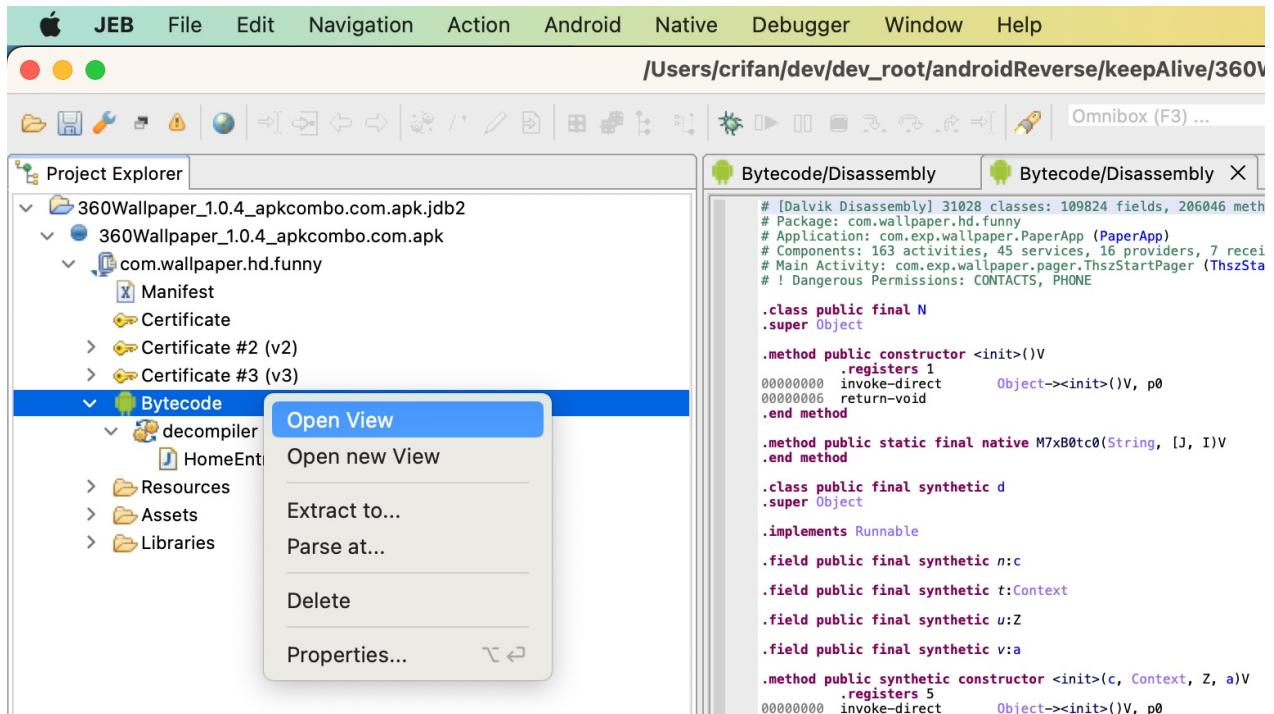
crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新:  
2023-09-14 23:02:13

## 页面显示心得

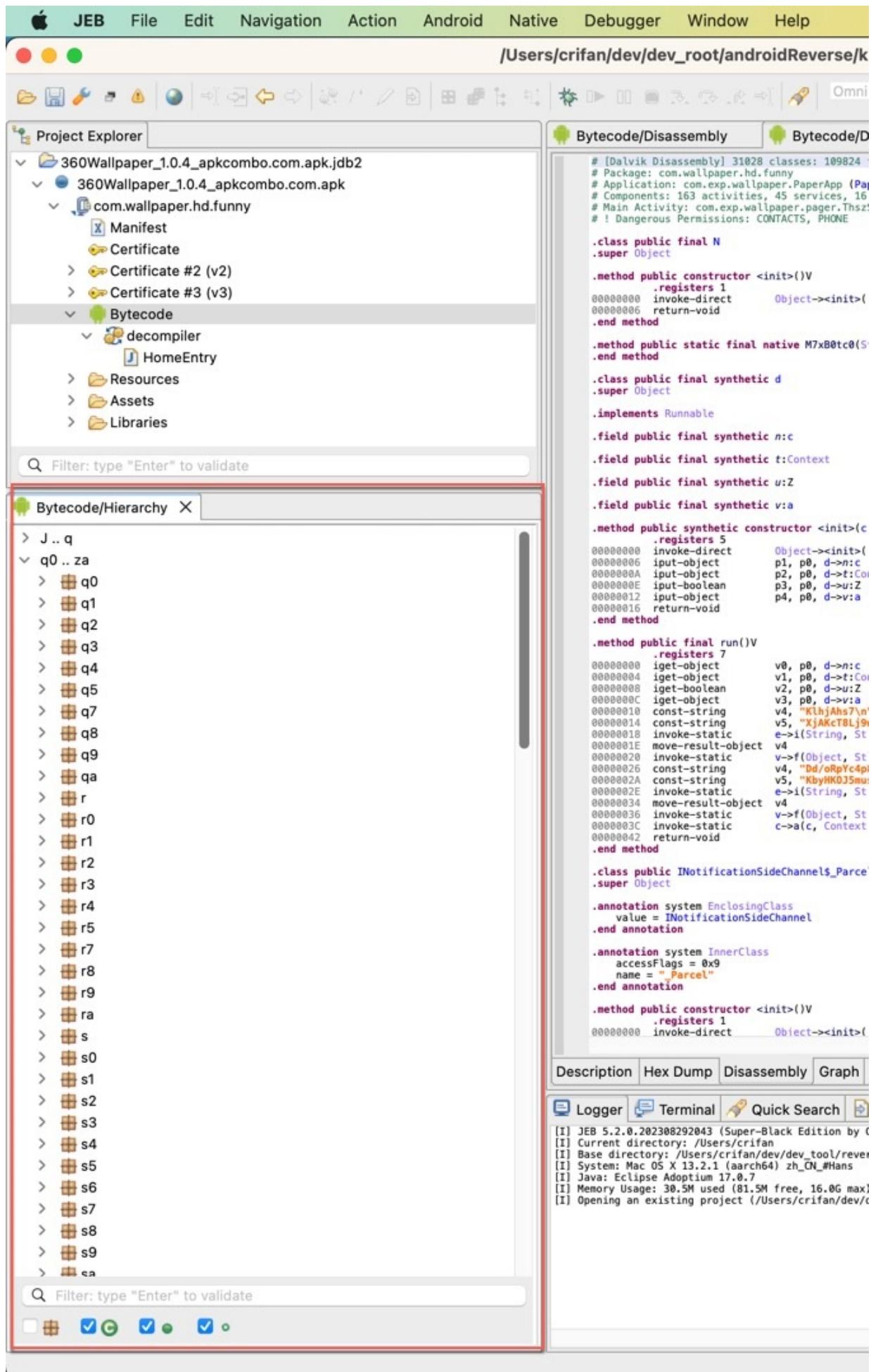
### 关闭Bytecode/Hierarchy窗口后，如何恢复显示

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：



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:  
2023-09-15 22:58:56

# 反编译java代码心得

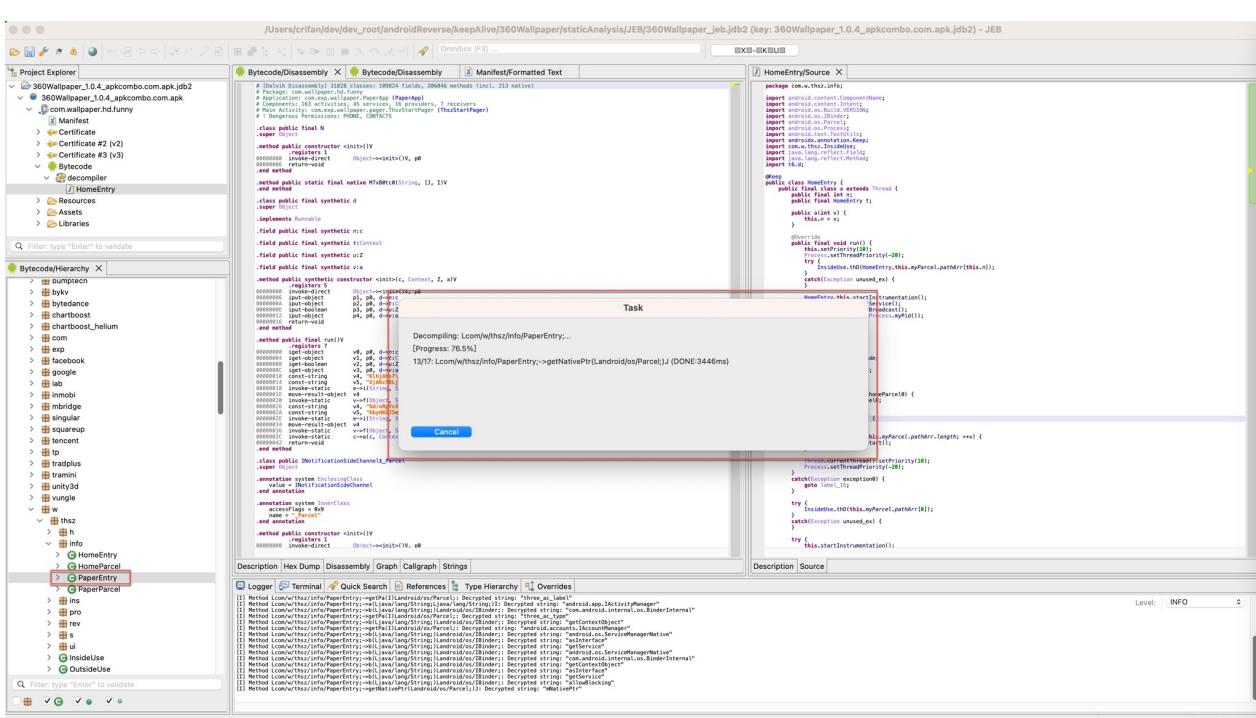
## 单个类（文件）的反编译java代码

### 反编译单个类的java代码

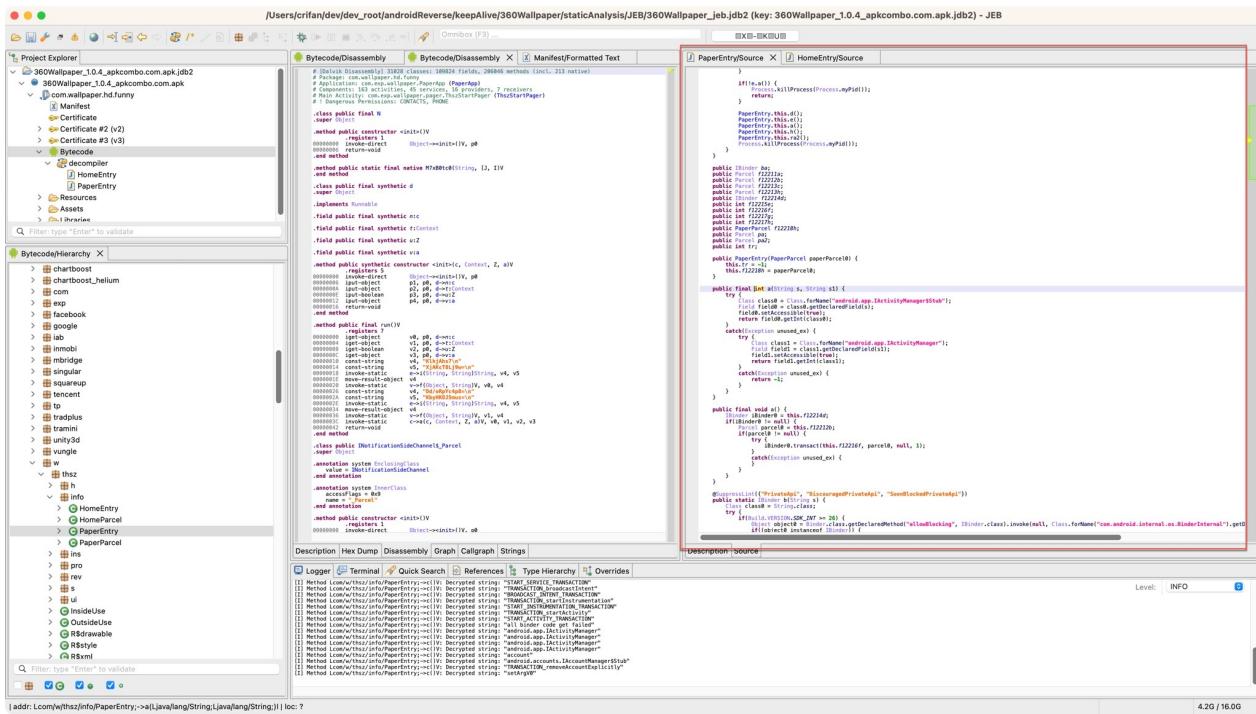
对于Bytecode窗口中，对于某个类（`com.w.thsz.info.PaperEntry`，双击后，等价于右键 -

> 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：

```

PaperEntry/Source X

    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 a() {
IBinder IBinder0 = this.f12214d;
if(IBinder0 != null) {
    Parcel parcel0 = this.f12212b;
    if(parcel0 != null) {
        try {
            IBinder0.transact(this.f12216f, parcel0, null, 1);
        }
        catch(Exception unused_ex) {
        }
    }
}
}

@SuppressLint({"PrivateApi", "DiscouragedPrivateApi", "SoonBlockedPrivateApi"})
public static IBinder b(String s) {
Class class0 = String.class;
try {
    if(Build.VERSION.SDK_INT >= 26) {
        Object obj = Lcom/w/thsz/info/PaperEntry.b(java.lang.String) : android.os.IBinder
        Descriptor: Lcom/w/thsz/info/PaperEntry;-->b(Ljava/lang/String;)Landroid/os/IBinder;
        [AST Objects Stack]
        Class: class:Lcom/w/thsz/info/PaperEntry;
        Object: obj
        Method: method:com.w.thsz.info.PaperEntry.b
        return obj;
    }
    else {
        Class class1;
        Object obj1;
        if((object1 = Object.class) != null) {
            Object object2;
            if((object2 = object1.newInstance()) != null) {
                return (IBinder)object2;
            }
        }
    }
}
catch(Throwable unused_ex) {
}
return null;
}

public final void b() {
try {
    this.c();
    for(int v = 1; v < this.f12218h.f12226a.length; ++v) {
        new a(this, v).start();
    }
    Thread.currentThread().setPriority(10);
    Process.setThreadPriority(-20);
}
catch(Exception exception0) {
    goto label_20;
}
try {
    InsideUse.thD(this.f12218h.f12226a[0]);
}
}
}

Description Source

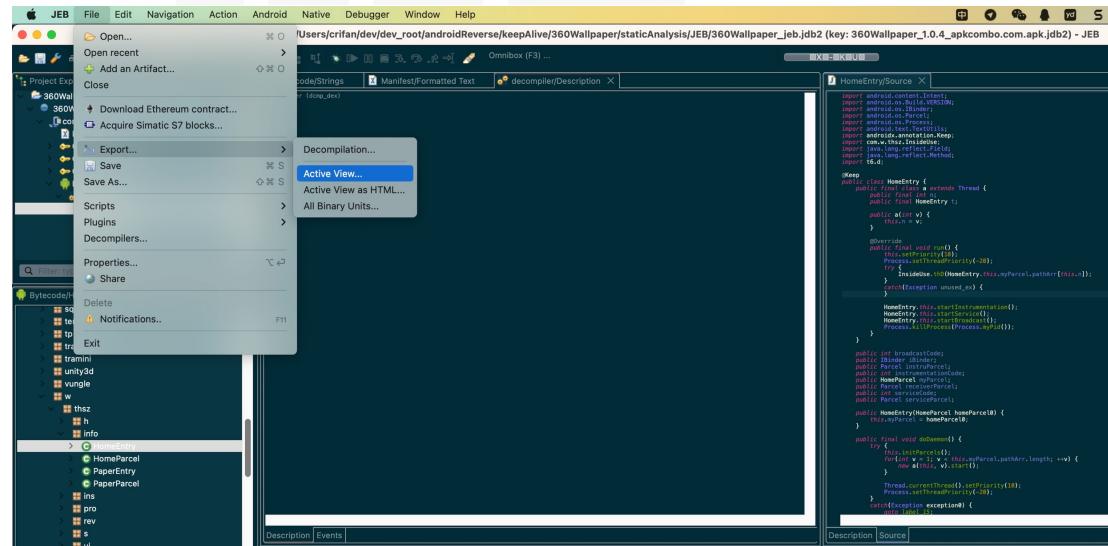
```

## 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

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());
        }
    }

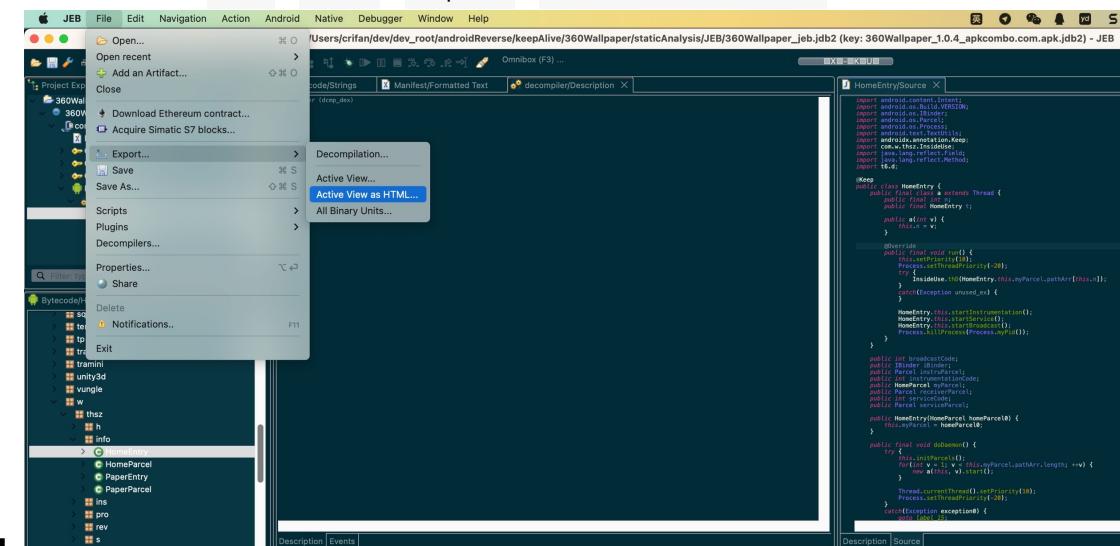
    public int broadcastCode;
    public IBinder iBinder;
    public Parcel instruParcel;
    public int instrumentationCode;
    public HomeParcel myParcel;
    public Parcel receiverParcel;
    public int serviceCode;
    public Parcel serviceParcel;

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

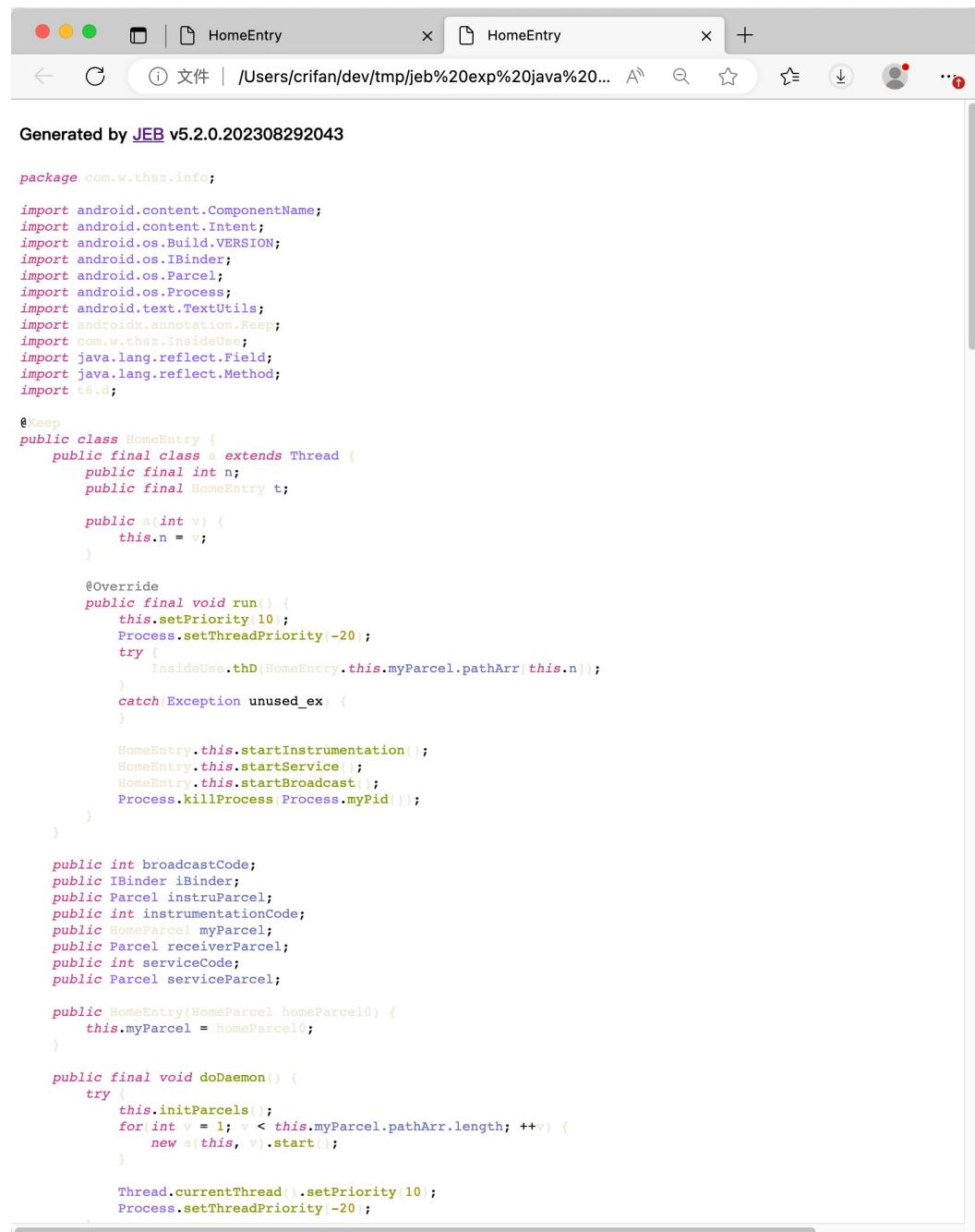
    public final void doDaemon() {
        try {
            this.initParcels();
            for (int v = 1; v < this.myParcel.pathArr.length; ++v) {
                new a(this, v).start();
            }
        }
    }
}

```

- 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



The screenshot shows the JEB debugger interface with the title bar "HomeEntry". The code editor displays Java code for the "HomeEntry" class. The code includes imports for various Android packages like android.content, android.os, and android.text. It features a constructor "a(int v)" and a run() method that sets thread priority and calls startInstrumentation(), startService(), and startBroadcast(). The code also includes several fields and methods related to Parcel objects.

```

Generated by JEB v5.2.0.202308292043

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.Insideus;
import java.lang.reflect.Field;
import java.lang.reflect.Method;
import kb.v;

@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 {
                Insideus.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);
    }
}

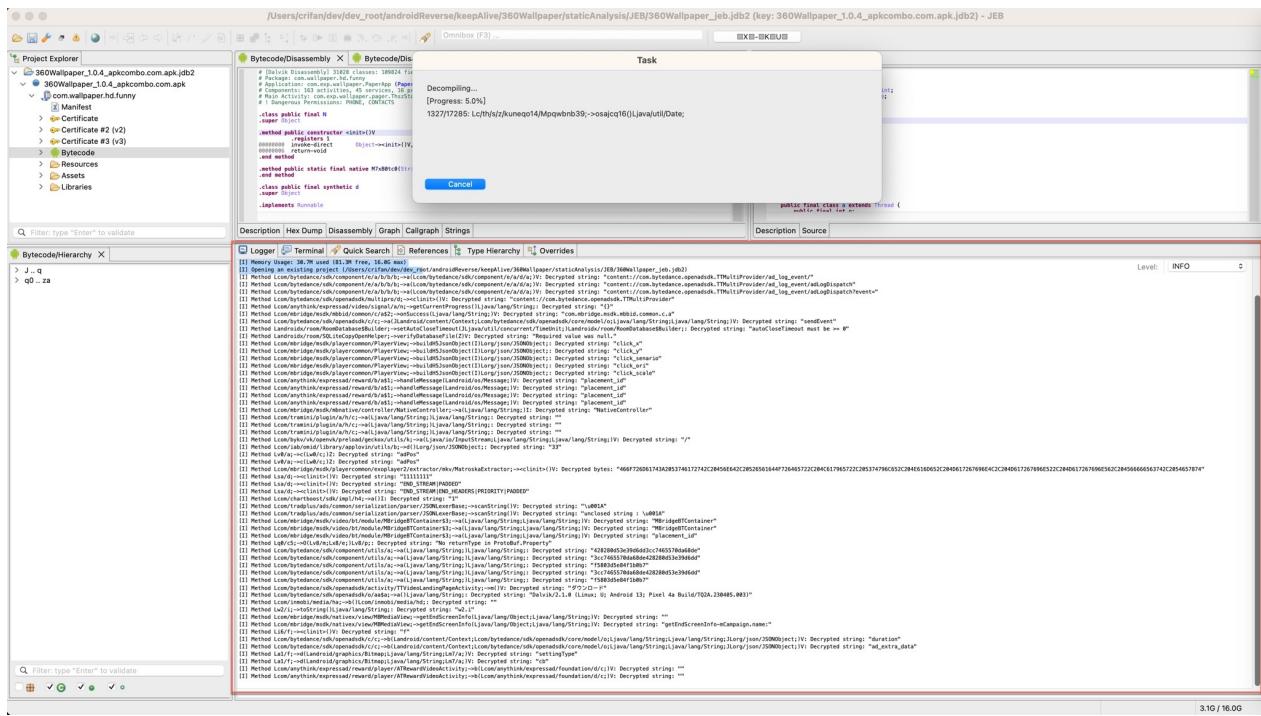
```

## 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

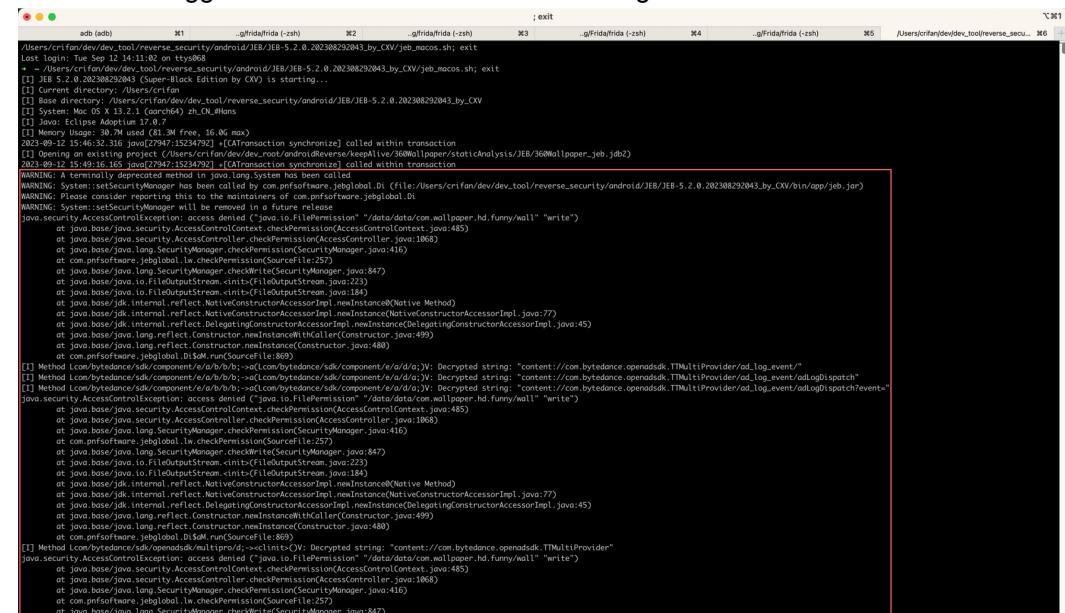
在[批量反编译全部代码](#)期间，对于想要查看反编译期间的日志：

是可以通过JEB底部的Logger的tab页，查看到对应日志的：

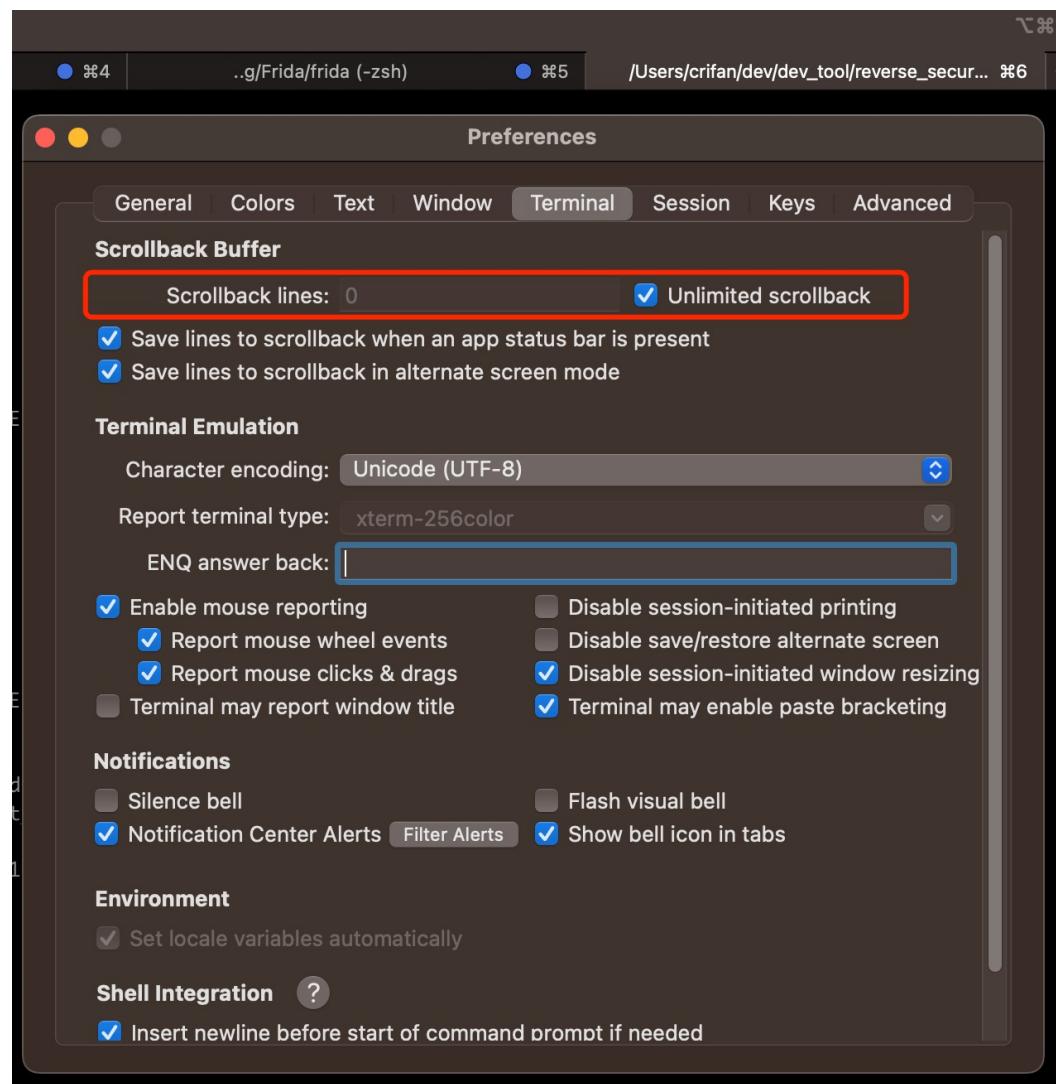


但是实测发现，最多只能保留2000行左右的日志。更多的日志，会被冲掉，无法保留。

- 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



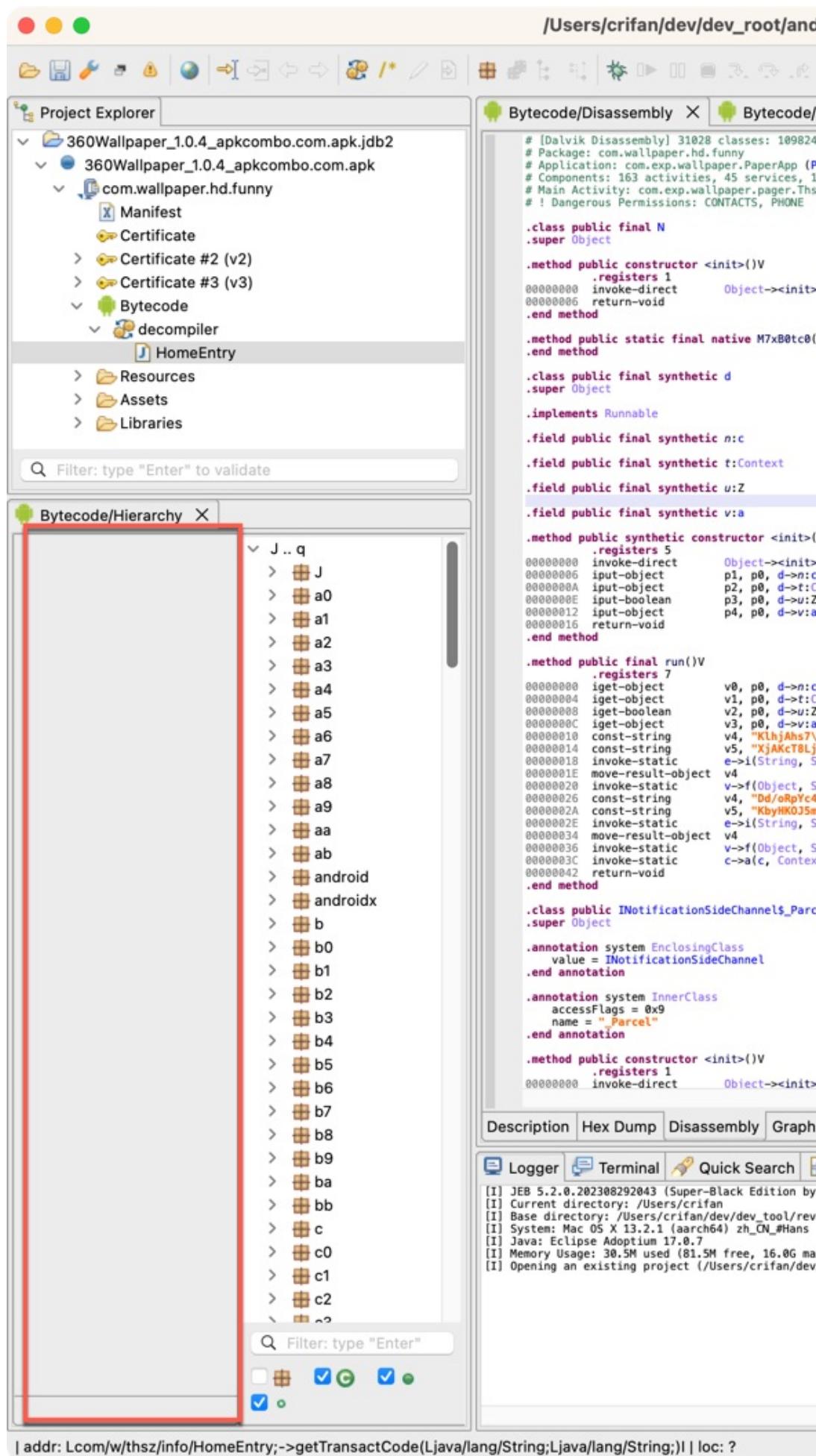
- 即可显示=保存出全部反编译期间的所有log日志
  - 后续去选择并拷贝，即可导出全部日志，供后续分析研究用

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：  
2023-09-16 21:32:51

## 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，往往能看到： Bytecode/Hierarchy 窗口有多余的一列
  -



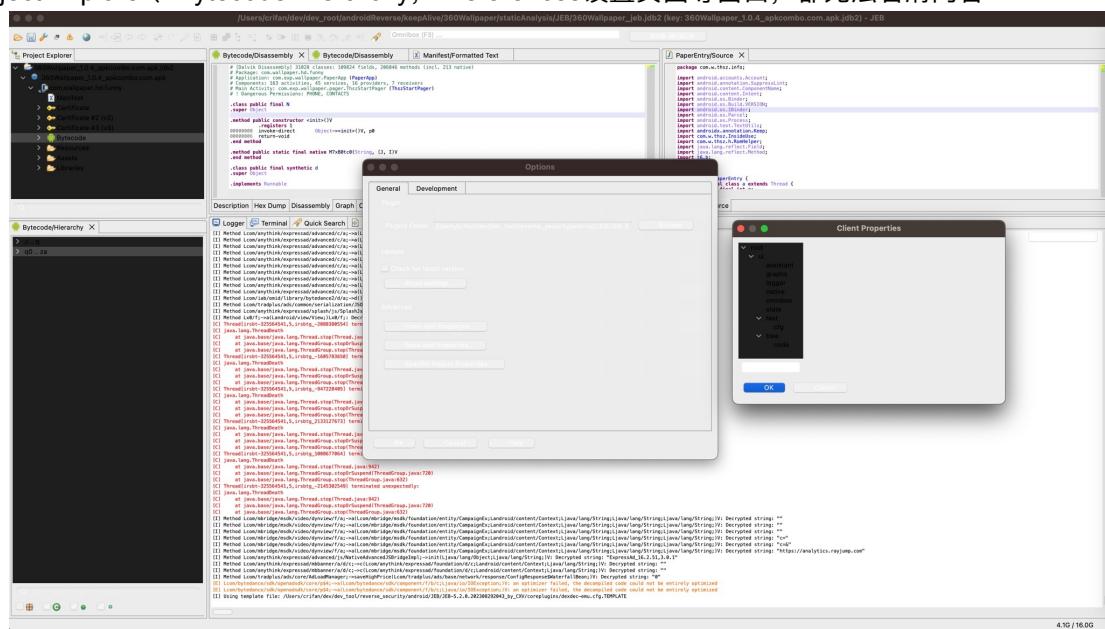
- 解决办法：关闭后重新打开 Bytecode/Hierarchy 窗口
- 具体步骤：
  - 概述：Project Explorer 中 apk 展开列表中的 Bytecode 右键选 Open View
  - 详解：页面显示心得 中的 关闭Bytecode/Hierarchy窗口后，如何恢复显示

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：  
2023-09-16 21:56:20

## 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

o

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：  
2023-09-16 21:57:24

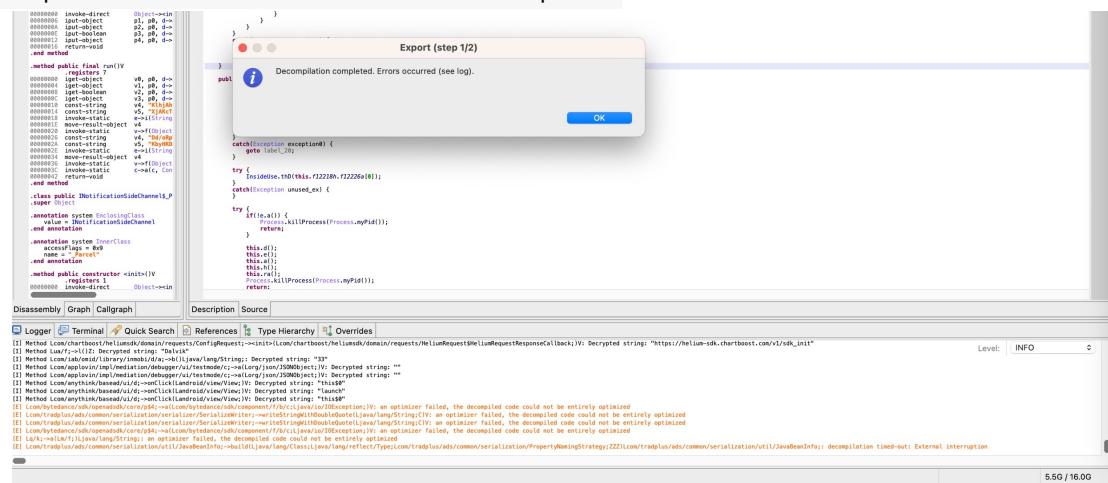
# 反编译java代码问题

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

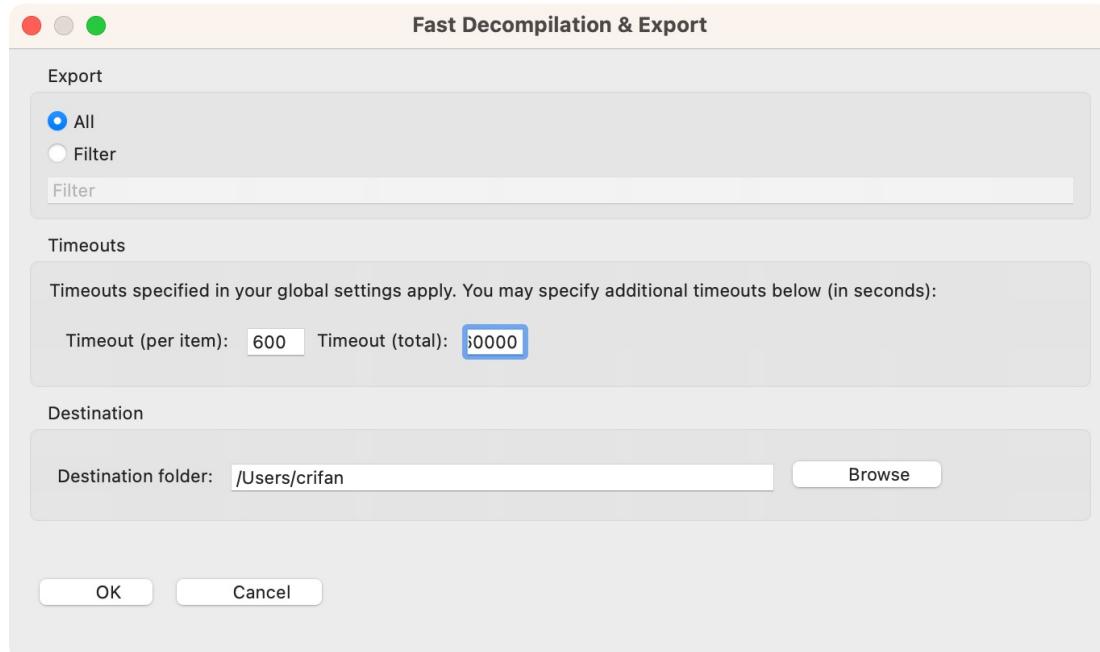
用默认超时设置会导致最后由于超时而被中断停止

- 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

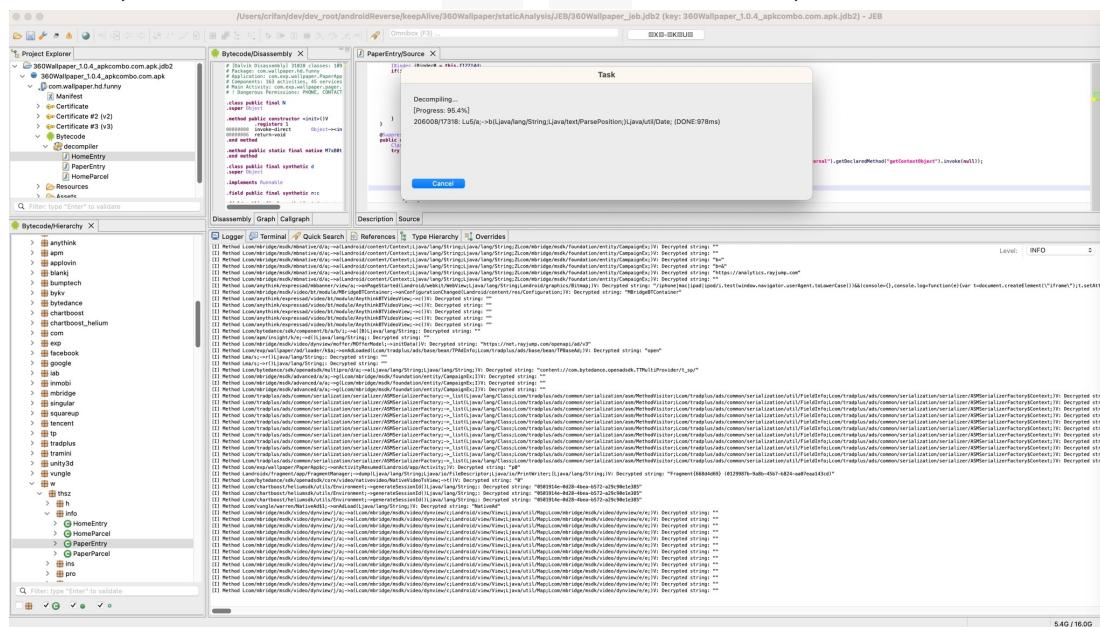


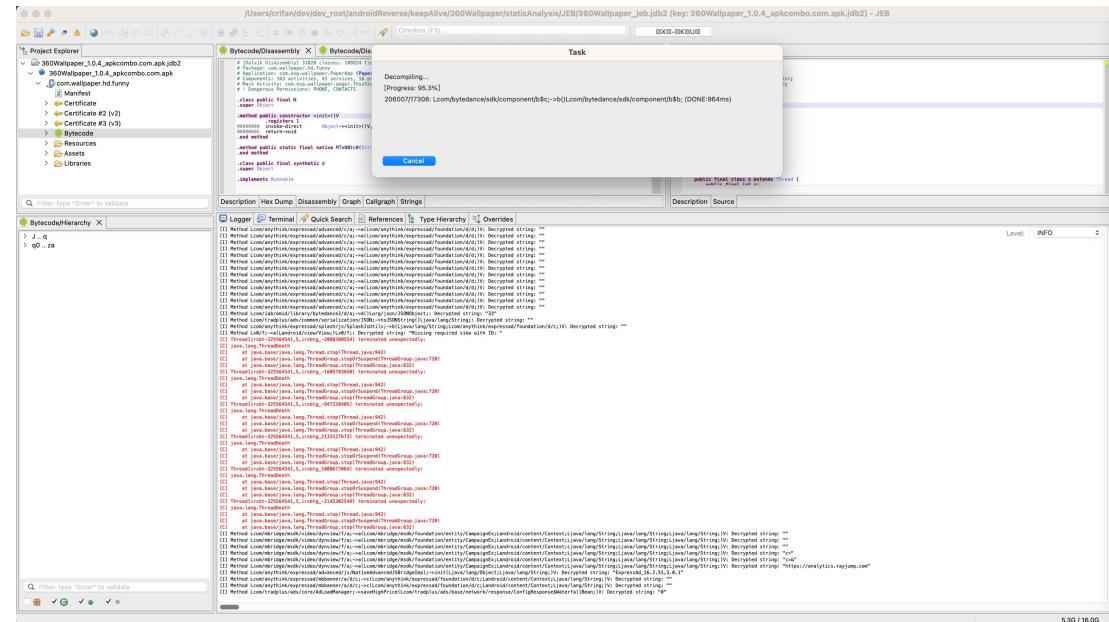
- Timeouts:
  - Timeout per item: 600 =10分钟
  - Timeout total: 60000 =100分钟

## 最后会卡死和多次的反编译输出结果不稳定

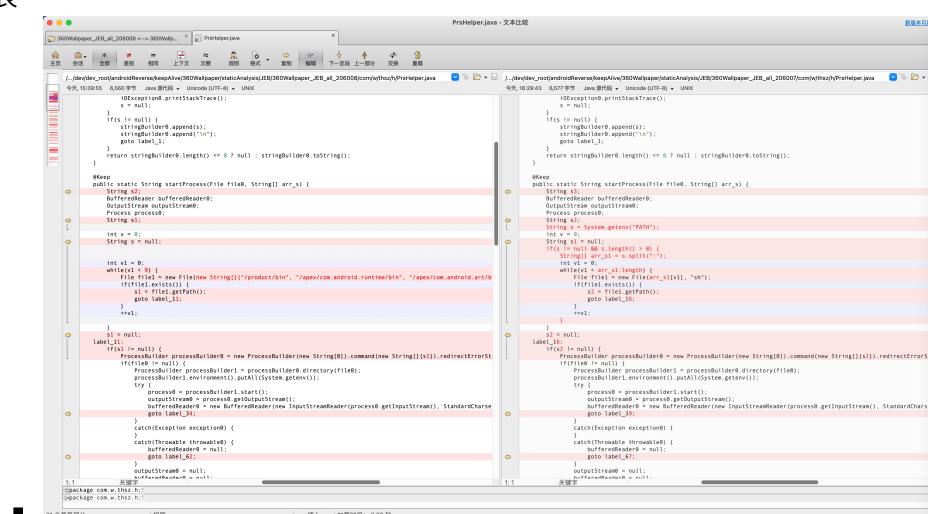
在**批量反编译全部代码**期间，遇到一些问题：

- 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++;
        }
    }
}

```

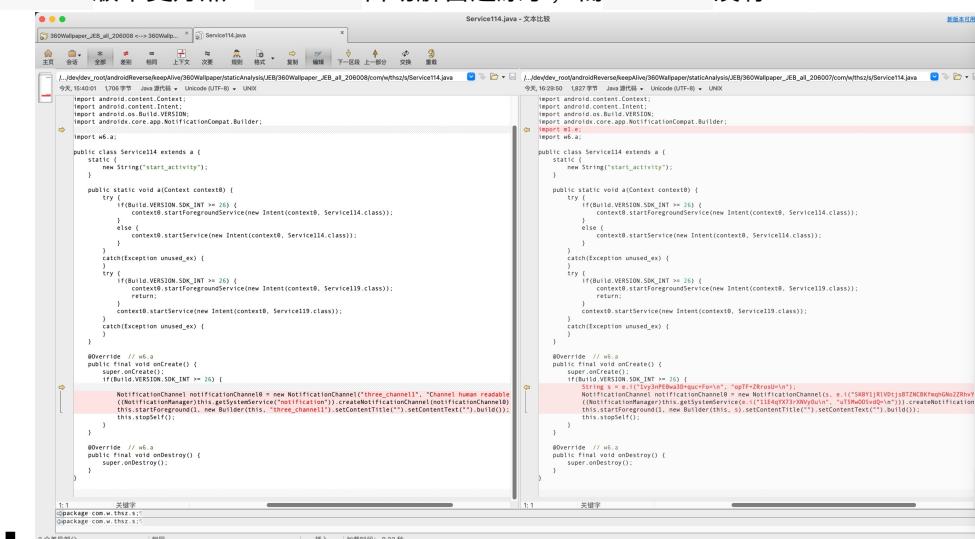
■ 206008 版本相关代码

```

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/bin",
        "/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

```

Service104.java - 文本比较

package com.w.thz.s;
import android.accounts.Account;
import android.accounts.AbstractAccountSyncAdapter;
import android.accounts.AccountSyncAdapter;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.os.IBinder;
import java.lang.reflect.Field;

public class Service104 extends Service {
    public Service104() {
        super();
    }

    @Override // android.accounts.AbstractAccountSyncAdapter
    public void onPerformSync(Account account, Bundle bundle, String s, ContentProviderClient contentProvide
    if(bundle != null) {
        try {
            if(bundle.getBoolean("cvv1234567890", false)) {
                if(bundle.getBoolean("cvv1234567890", false)) {
                    String as = "123456789012345678901234567890";
                    Field field = SyncResult.class.getDeclaredField("syncResult");
                    field.setAccessible(true);
                    field.setBoolean(syncResult, true);
                }
            }
        } catch(Exception unused_ex) {
        }
    }

    @Override // android.accounts.AbstractAccountSyncAdapter
    public final void onSyncCanceled() {
        super.onSyncCanceled();
    }
}

public class Service104 extends Service {
    public static class Sync extends AbstractAccountSyncAdapter {
        public Sync(Context context) {
            super(context);
        }

        @Override // android.accounts.AbstractAccountSyncAdapter
        public void onPerformSync(Account account, Bundle bundle, String s, ContentProviderClient contentProvide
        if(bundle != null) {
            try {
                if(bundle.getBoolean("cvv1234567890", false)) {
                    if(bundle.getBoolean("cvv1234567890", false)) {
                        String as = "123456789012345678901234567890";
                        Field field = SyncResult.class.getDeclaredField("syncResult");
                        field.setAccessible(true);
                        field.setBoolean(syncResult, true);
                    }
                }
            } catch(Exception unused_ex) {
            }
        }

        @Override // android.accounts.AbstractAccountSyncAdapter
        public final void onSyncCanceled() {
            super.onSyncCanceled();
        }
    }

    public class Sync {
        public void onSyncCanceled() {
        }
    }
}

```

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。

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：  
2023-09-16 21:44:18

## 附录

下面列出相关参考资料。

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：  
2023-09-14 21:55:35

## 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jad
- 【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
- 
- [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deviceid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](#)
-

2023-09-16 20:44:45