目录

iOS逆向工具：Hopper

- 最新版本：`v1.0.0`
- 更新时间：`20231008`

简介

介绍iOS逆向的常用工具之前：Hopper。先是Hopper概览；然后是下载和安装，接着是如何使用；并且给出了具体例子，比如AwemeCore、Thunder等；且整理了相关心得；以及和IDA的对比。

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/ios_re_tool_hopper: iOS逆向工具：Hopper

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

- iOS逆向工具：Hopper book.crifan.org
- iOS逆向工具：Hopper crifan.github.io

离线下载阅读

- iOS逆向工具：Hopper PDF
- iOS逆向工具：Hopper ePub
- iOS逆向工具：Hopper 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：

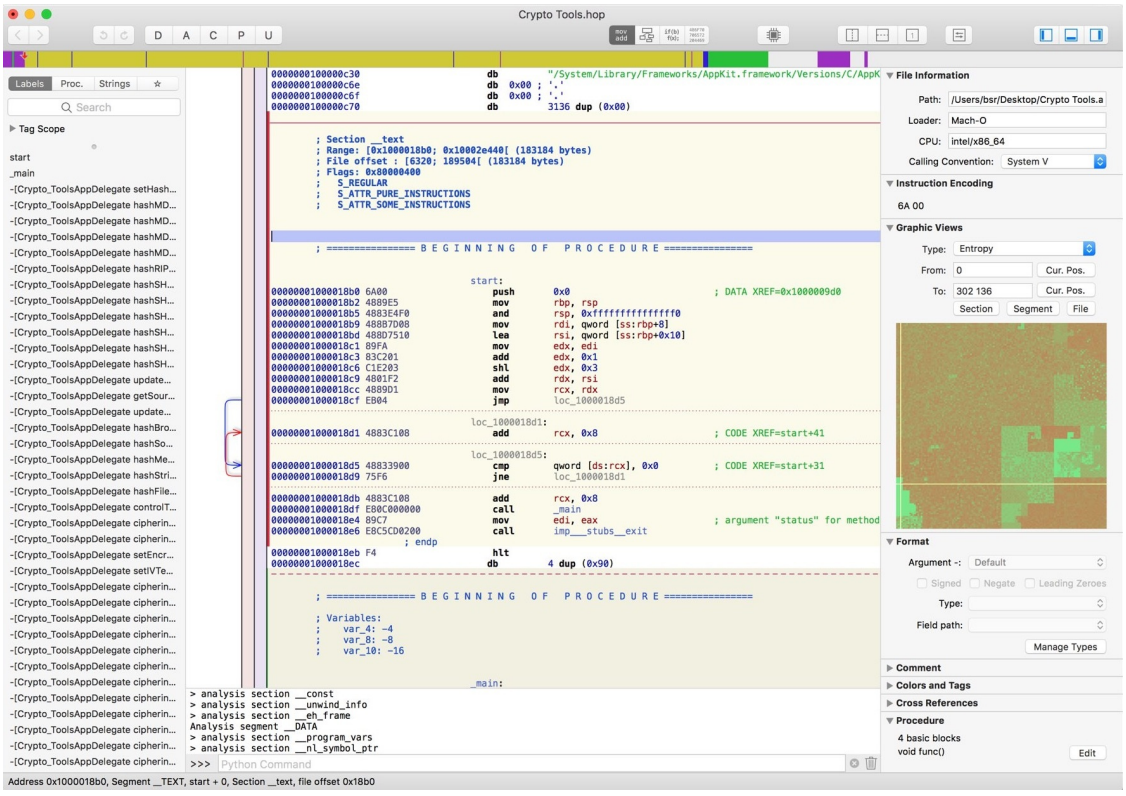
crifan/crifan_ebook_readme: Crifan的电子书的使用说明

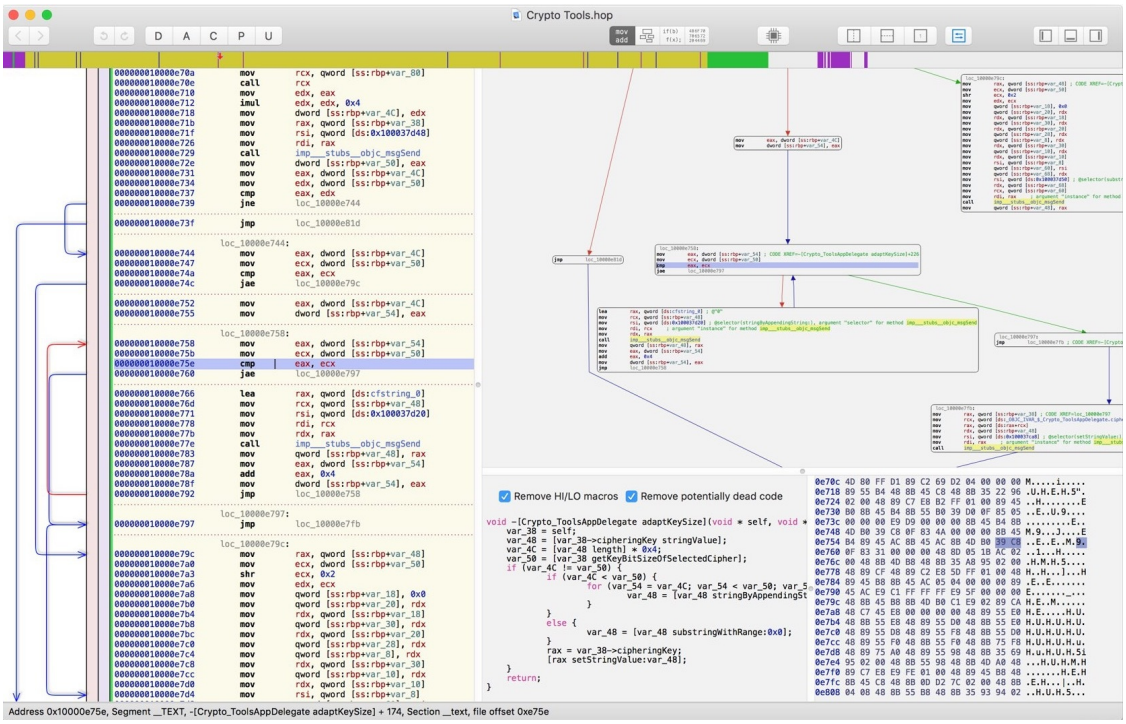
关于作者

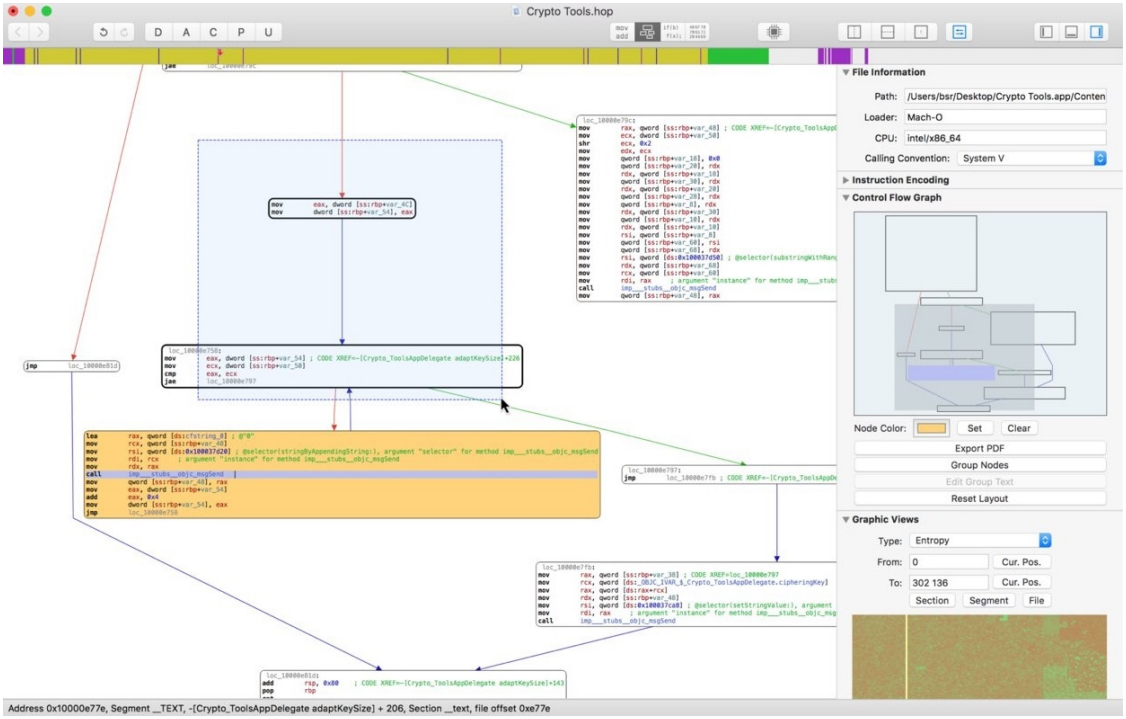
关于作者更多介绍，详见：

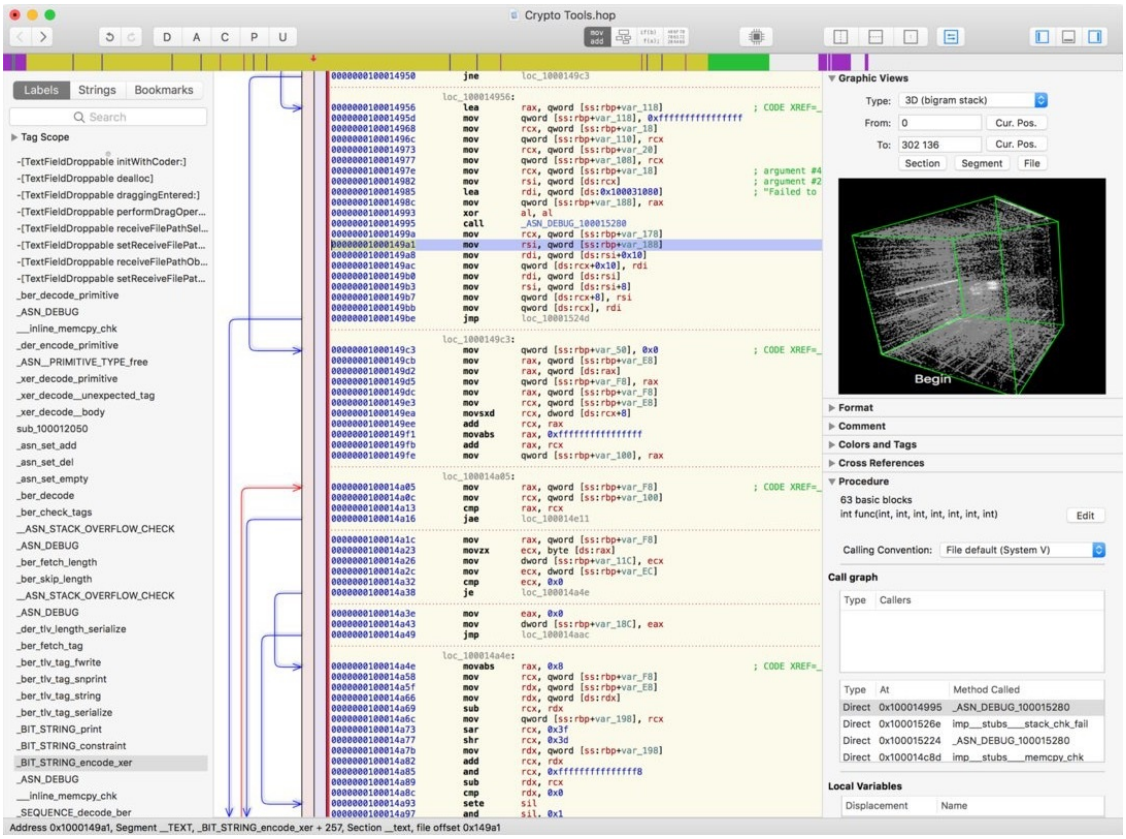
关于CrifanLi李茂 – 在路上

Hopper概览

- `Hopper` = `Hopper Disassembler`
  - 偶尔缩写为：`hd`
  - 是什么：iOS逆向工具
    - Hopper is a reverse engineering tool for OS X and Linux, lets you disassemble, decompile and debug your applications
      - This tool will let you disassemble any binary you want, and provide you all the information about its content, like imported symbols, or the control flow graph! Hopper can retrieve procedural information about the disassembled code like the stack variables, and lets you name all the objects you want.
  - 作用：主要用于二进制的静态逆向分析代码逻辑
    - disassemble
    - decompile
  - 对标：IDA
  - 支持
    - 运行平台：`Mac` 、 `Linux`
    - 目标架构：`32/64bits Intel` / `Apple Silicon` Mac, Linux, Windows and iOS executables
  - 主页
    - Hopper (hopperapp.com)
  - 截图
    - 


Crypto Tools.hop
D A C P U
Remove HI/LO macros
Remove potentially dead code
Address 0x10000e75e, Segment __TEXT, -[Crypto_ToolsAppDelegate adaptKeySize] + 174, Section __text, file offset 0xe75e


Crypto Tools.hop
D A C P U
File Information
Path: /Users/bsr/Desktop/Crypto Tools.app/Conten
Loader: Mach-O
CPU: intel/x86_64
Calling Convention: System V
Instruction Encoding
Control Flow Graph
Node Color: Set Clear
Export PDF
Group Nodes
Edit Group Text
Reset Layout
Graphic Views
Type: Entropy
From: 0 Cur. Pos.
To: 302 136 Cur. Pos.
Section Segment File
Address 0x10000e77e, Segment __TEXT, -[Crypto_ToolsAppDelegate adaptKeySize] + 206, Section __text, file offset 0xe77e

- 

下载和安装Hopper

请支持正版

警告⚠：此处Hopper破解版仅限于技术研究使用，**不准**用于**非法目的**，否则后果自负。

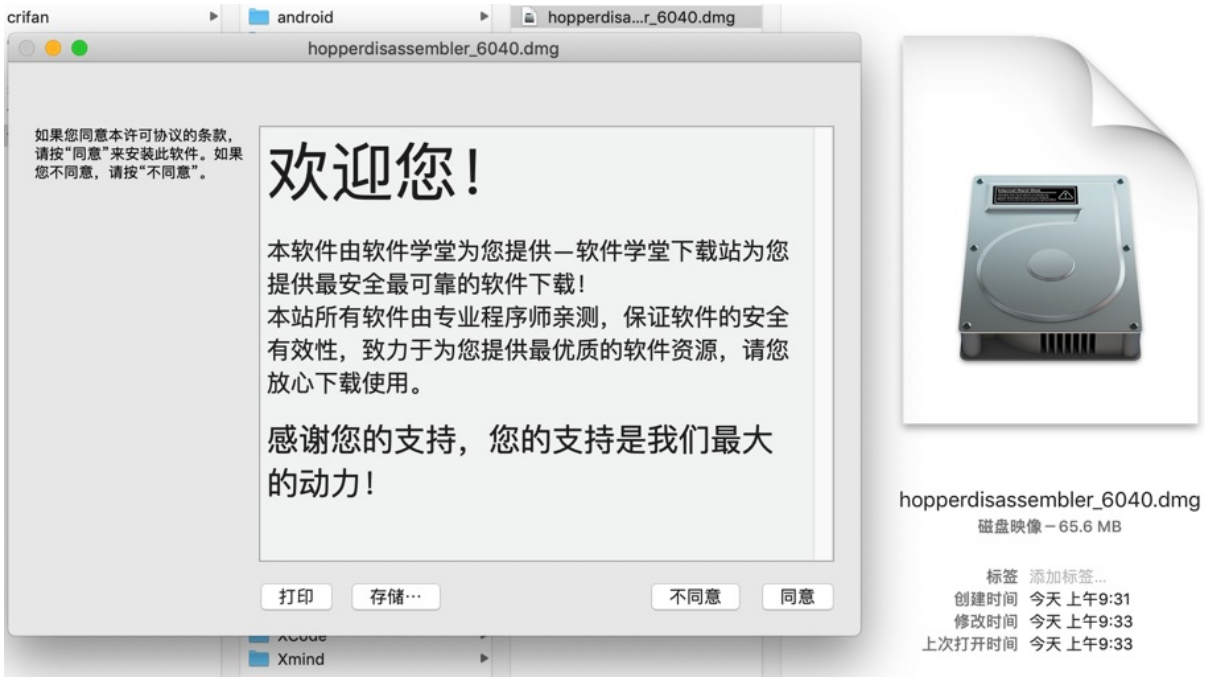
如有侵权，请联系笔者删除。

下载

下载破解版的Hopper Disassembler：

hopper disassembler for mac v4破解版下载(免授权文件/序列号) v4.0.8 - 软件学堂

-> 得到：hopperdisassembler_6040.dmg



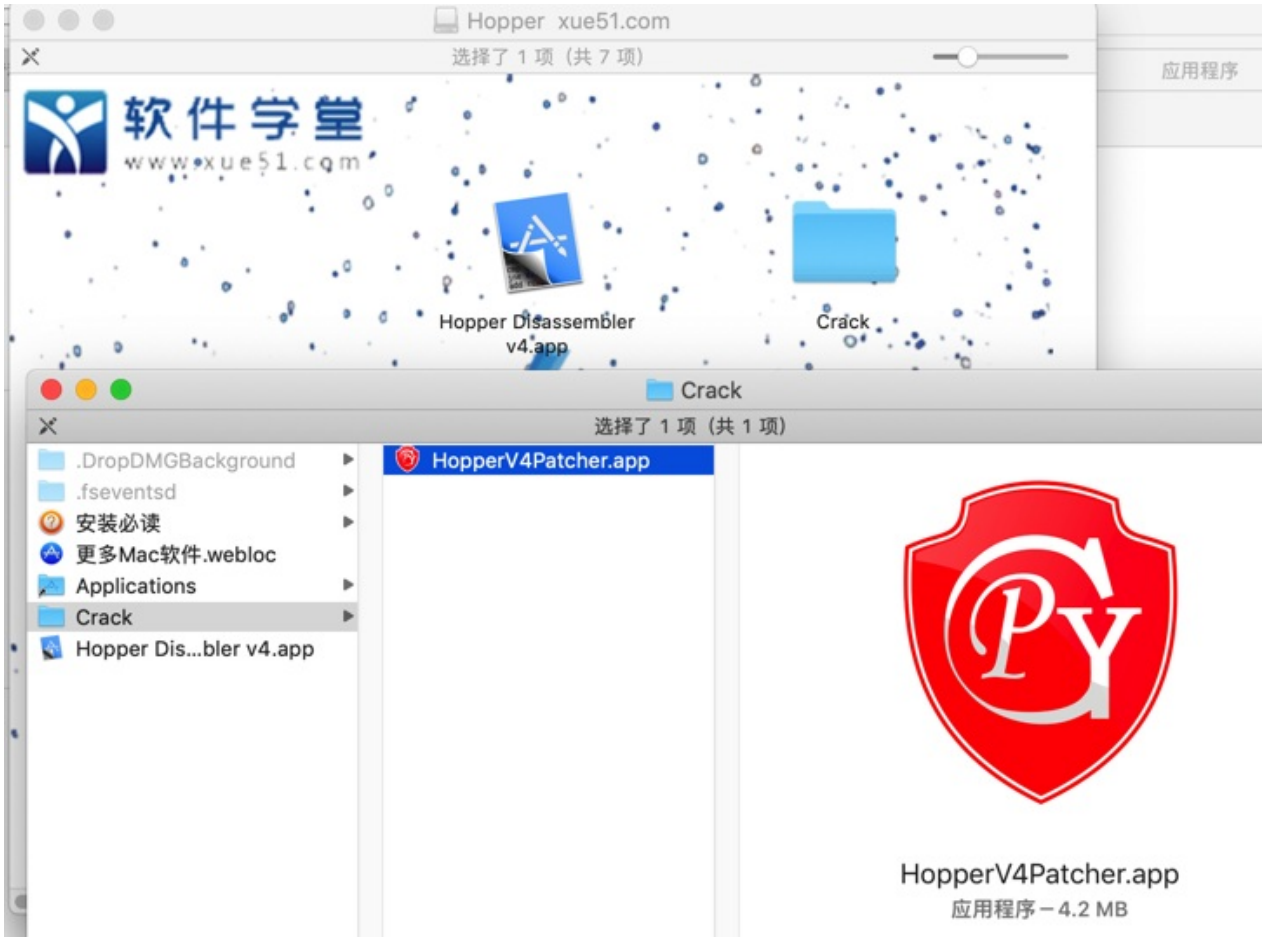
安装

双击 `dmg` ，继续，进入界面：

- 注：不要参考教程中说的，双击，否则会直接运行的。

去把 `Hopper Disassembler v4.app` 拖动到 `应用程序` 中：

另外 `Crack` 中有个： `HopperV4Patcher.app`



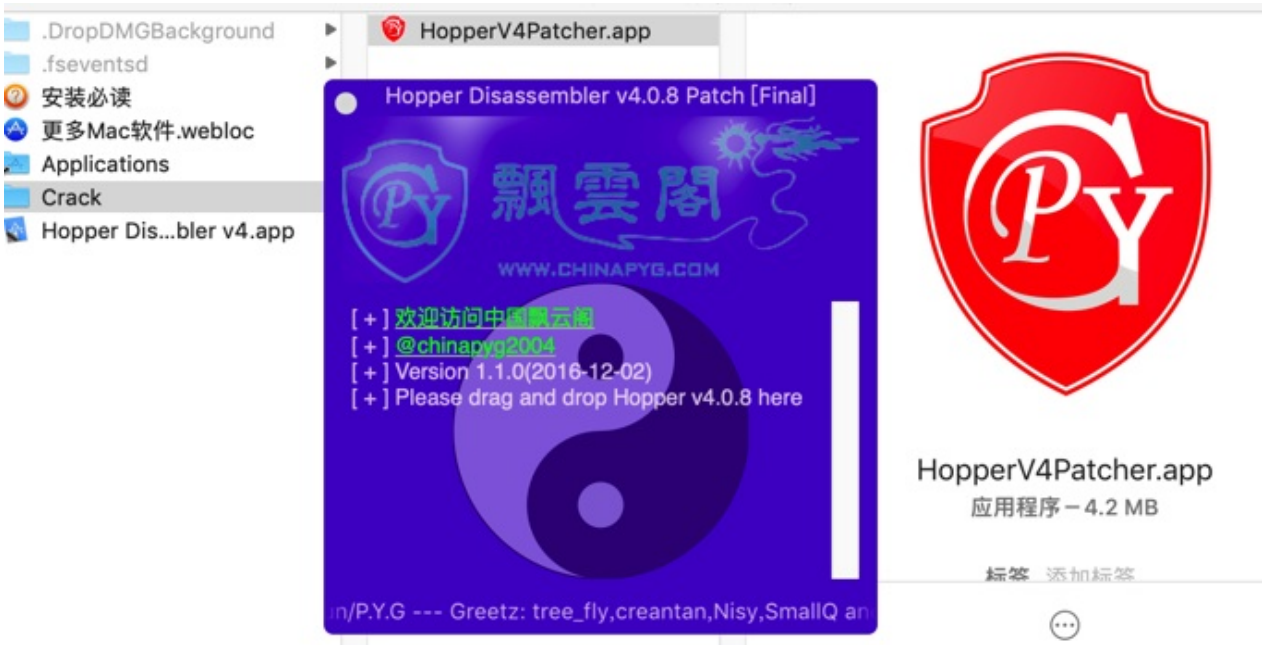
双击运行，出现提示：

- 不明身份开发者

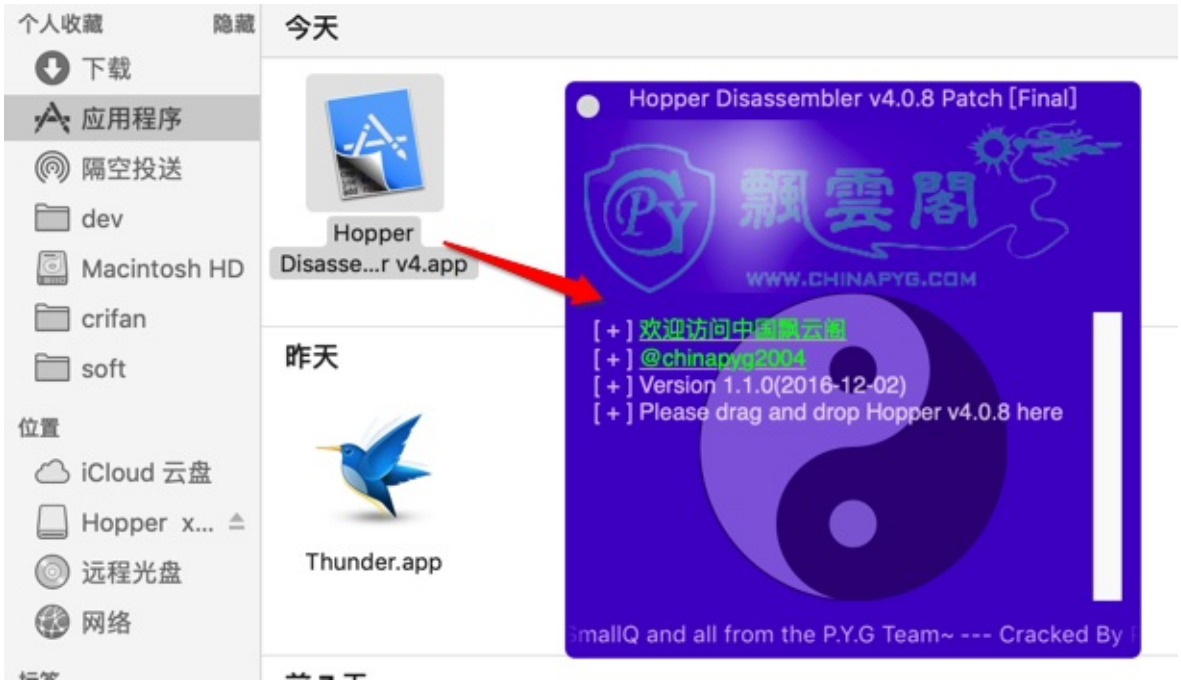
去隐私中允许：

- 仍要打开

即可出现界面：



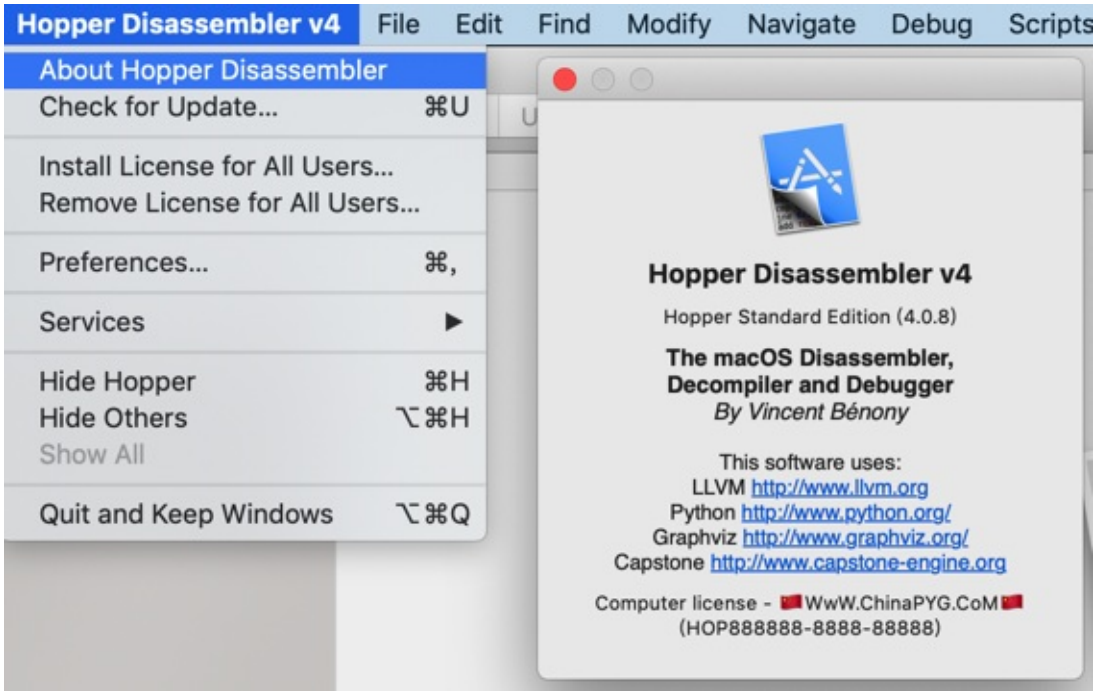
再去把应用程序中的：`Hopper Disassembler v4.app` 拖动到这个patch界面中：



瞬间就破解好了：

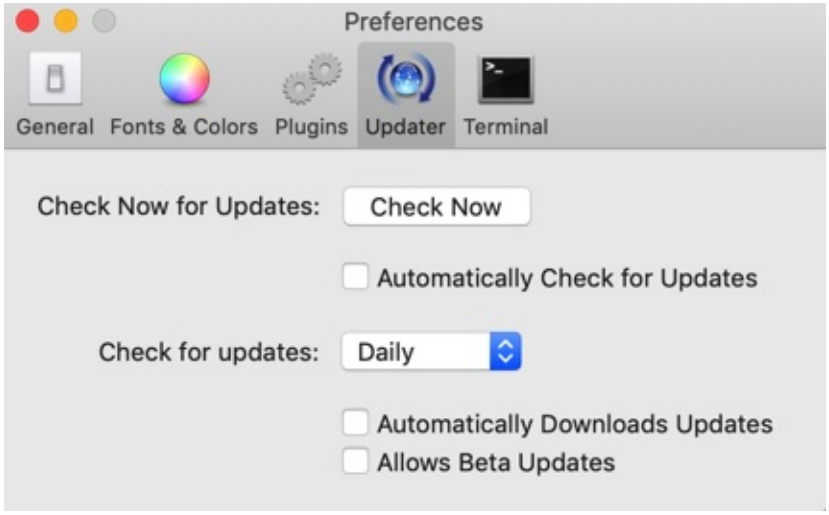
- 显示 `patch success`

如此，即可安装完毕，即可正常打开，已破解的Hopper：



的确没有注册弹框，可以看到 `Computer license` 了，说明破解成功。

另外：为了防止更新导致破解失效，去看看更新设置：

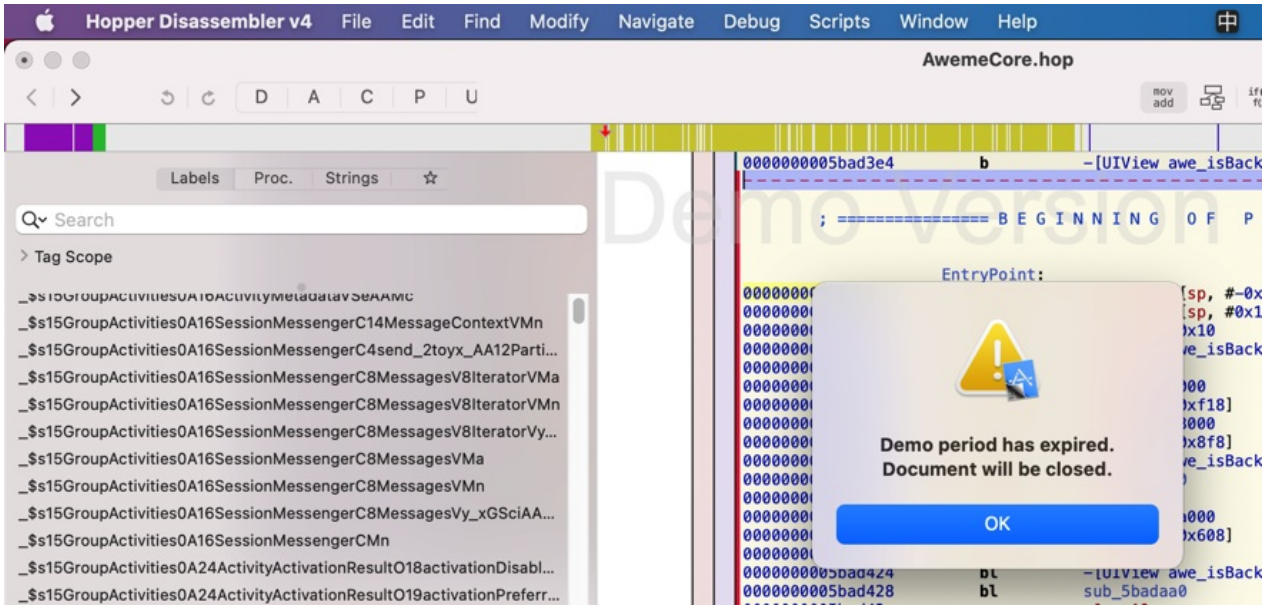


此处已经是关闭自动升级更新了，是我们希望的：不要开启自动更新。

后记：

结果用了几天后，还是会过期，无法继续使用：

```
Demo period has expired
Document will be closed
```



点击OK，就退出了。

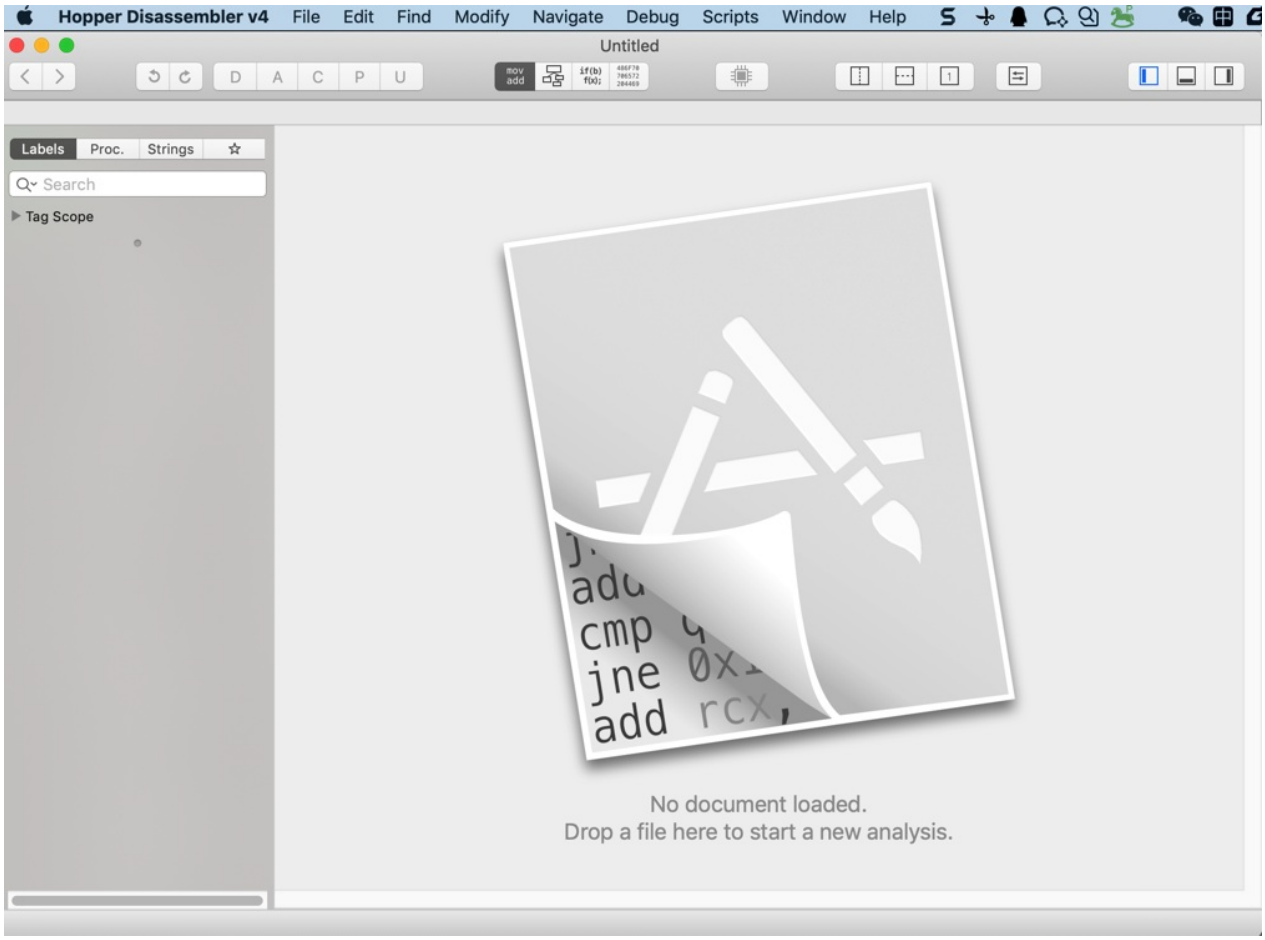
另外，此处感觉Hopper也比较卡顿，且对于大型app也会卡死，所以放弃。

继续转用IDA吧。

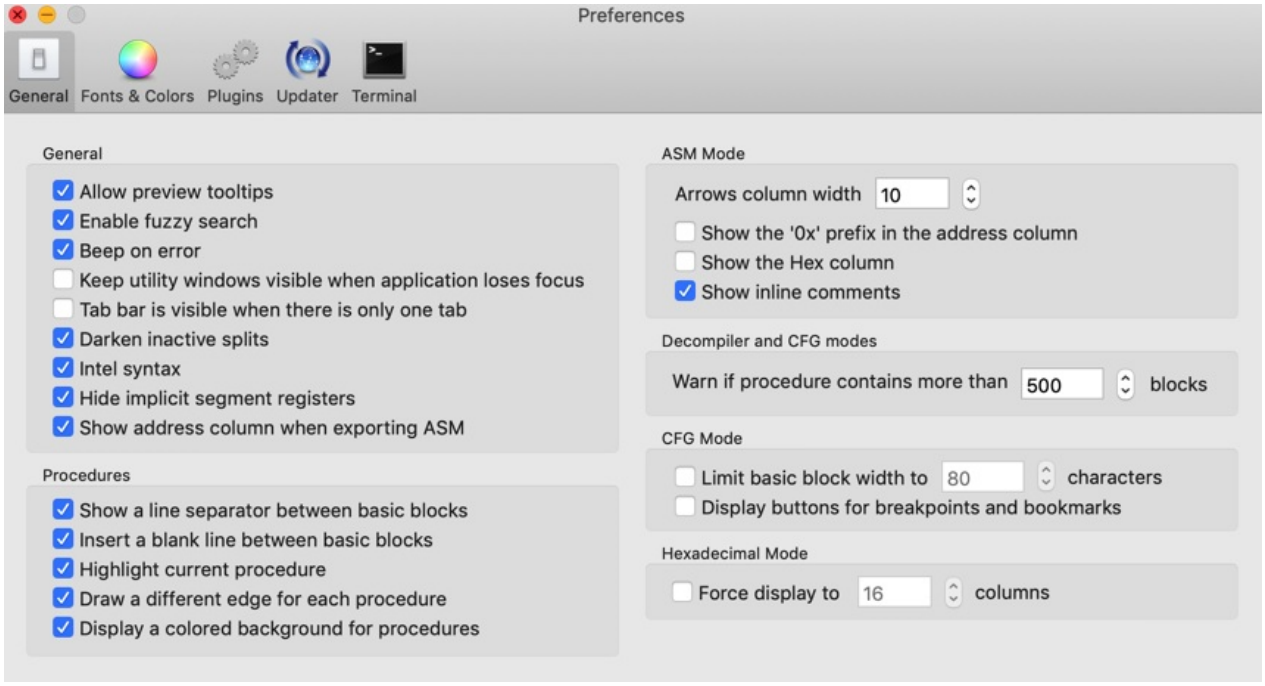
Hopper使用

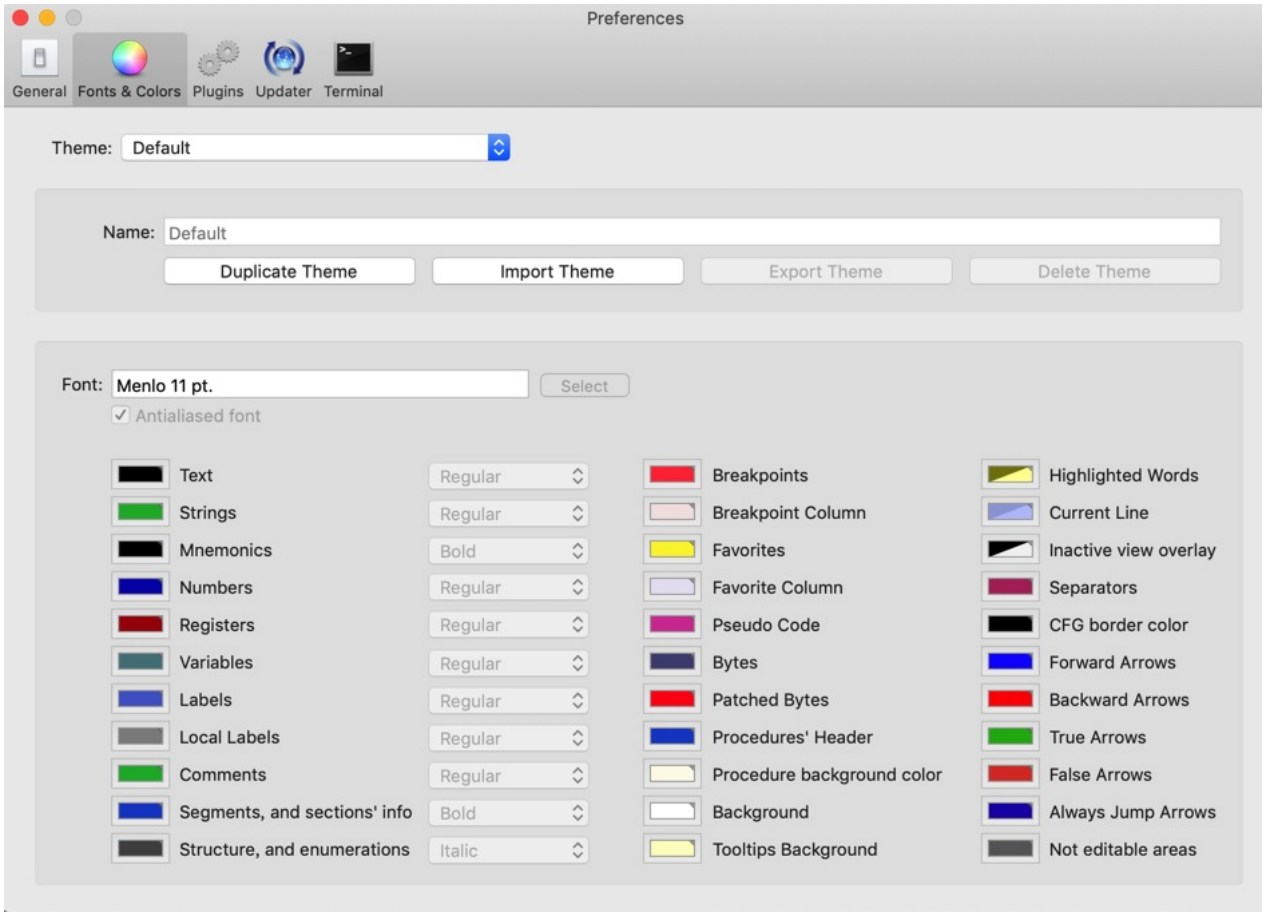
Hopper基本界面

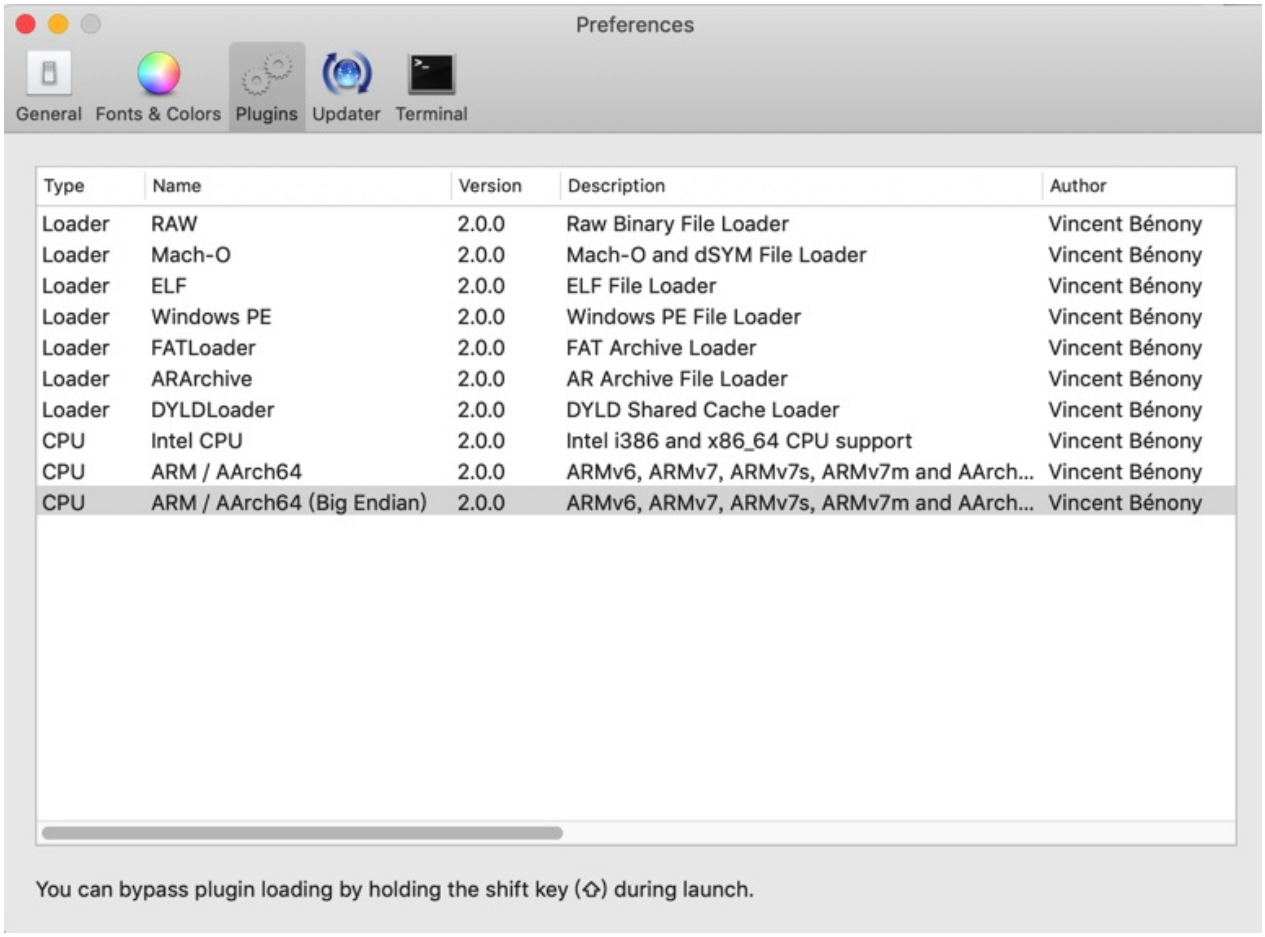
打开后的Hopper主界面是：

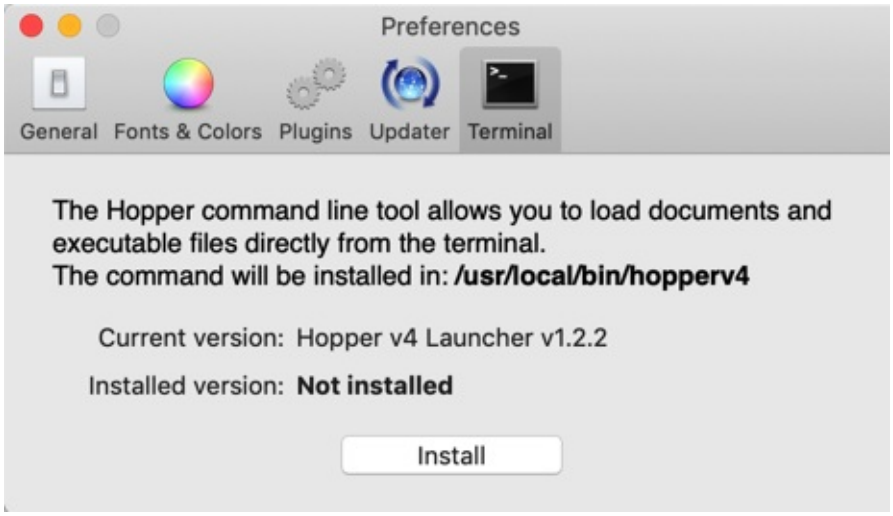


Preferences设置界面


Preferences
General
Fonts & Colors
Plugins
Updater
Terminal
General
Allow preview tooltips
Enable fuzzy search
Beep on error
Keep utility windows visible when application loses focus
Tab bar is visible when there is only one tab
Darken inactive splits
Intel syntax
Hide implicit segment registers
Show address column when exporting ASM
Procedures
Show a line separator between basic blocks
Insert a blank line between basic blocks
Highlight current procedure
Draw a different edge for each procedure
Display a colored background for procedures
ASM Mode
Arrows column width 10
Show the '0x' prefix in the address column
Show the Hex column
Show inline comments
Decompiler and CFG modes
Warn if procedure contains more than 500 blocks
CFG Mode
Limit basic block width to 80 characters
Display buttons for breakpoints and bookmarks
Hexadecimal Mode
Force display to 16 columns


Preferences
General
Fonts & Colors
Plugins
Updater
Terminal
Theme: Default
Name: Default
Duplicate Theme
Import Theme
Export Theme
Delete Theme
Font: Menlo 11 pt.
Select
Antialiased font
Text Regular
Strings Regular
Mnemonics Bold
Numbers Regular
Registers Regular
Variables Regular
Labels Regular
Local Labels Regular
Comments Regular
Segments, and sections' info Bold
Structure, and enumerations Italic
Breakpoints
Breakpoint Column
Favorites
Favorite Column
Pseudo Code
Bytes
Patched Bytes
Procedures' Header
Procedure background color
Background
Tooltips Background
Highlighted Words
Current Line
Inactive view overlay
Separators
CFG border color
Forward Arrows
Backward Arrows
True Arrows
False Arrows
Always Jump Arrows
Not editable areas


Preferences
General
Fonts & Colors
Plugins
Updater
Terminal
Type
Name
Version
Description
Author
Loader RAW 2.0.0 Raw Binary File Loader Vincent Bénony
Loader Mach-O 2.0.0 Mach-O and dSYM File Loader Vincent Bénony
Loader ELF 2.0.0 ELF File Loader Vincent Bénony
Loader Windows PE 2.0.0 Windows PE File Loader Vincent Bénony
Loader FATLoader 2.0.0 FAT Archive Loader Vincent Bénony
Loader ARArchive 2.0.0 AR Archive File Loader Vincent Bénony
Loader DYLDLoader 2.0.0 DYLD Shared Cache Loader Vincent Bénony
CPU Intel CPU 2.0.0 Intel i386 and x86_64 CPU support Vincent Bénony
CPU ARM / AArch64 2.0.0 ARMv6, ARMv7, ARMv7s, ARMv7m and AArch... Vincent Bénony
CPU ARM / AArch64 (Big Endian) 2.0.0 ARMv6, ARMv7, ARMv7s, ARMv7m and AArch... Vincent Bénony
You can bypass plugin loading by holding the shift key (⇧) during launch.


Preferences
General
Fonts & Colors
Plugins
Updater
Terminal
The Hopper command line tool allows you to load documents and executable files directly from the terminal.
The command will be installed in: /usr/local/bin/hopperv4
Current version: Hopper v4 Launcher v1.2.2
Installed version: Not installed
Install

Hopper功能介绍

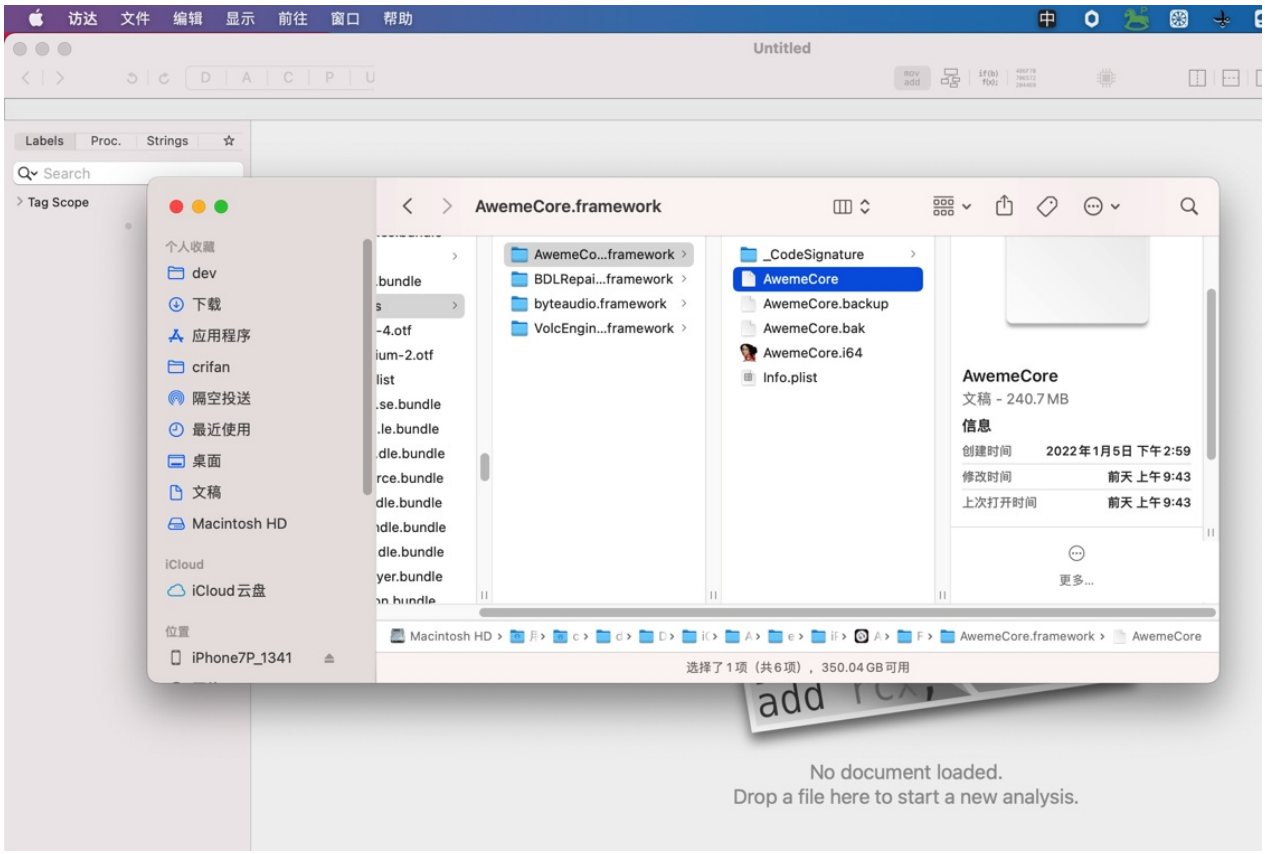
Hopper举例

Hopper使用举例：AwemeCore

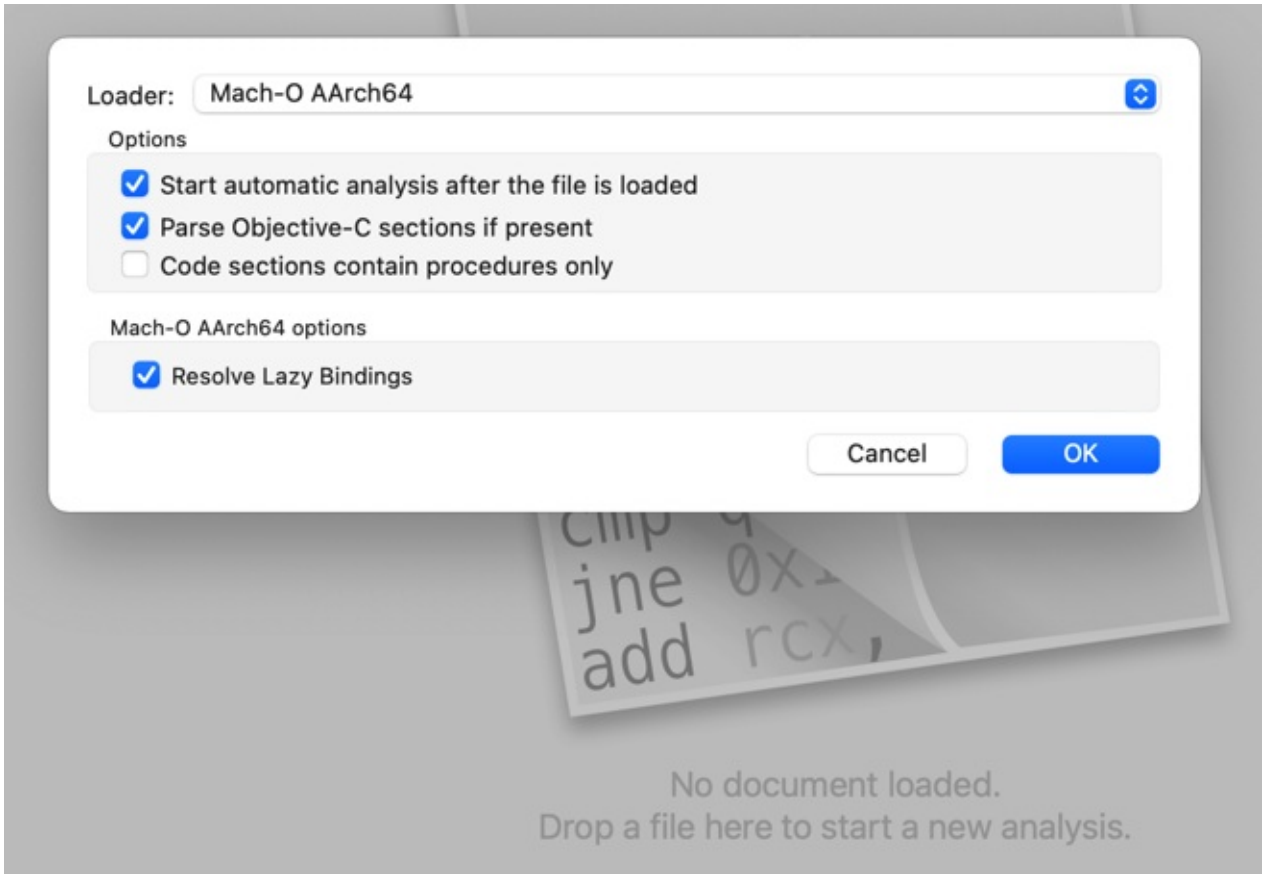
Hopper加载AwemeCore

- 输入文件：`AwemeCore`

然后把`AwemeCore`拖进`Hopper`



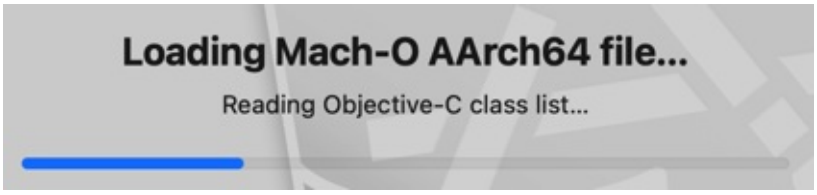
出现Loader弹框：



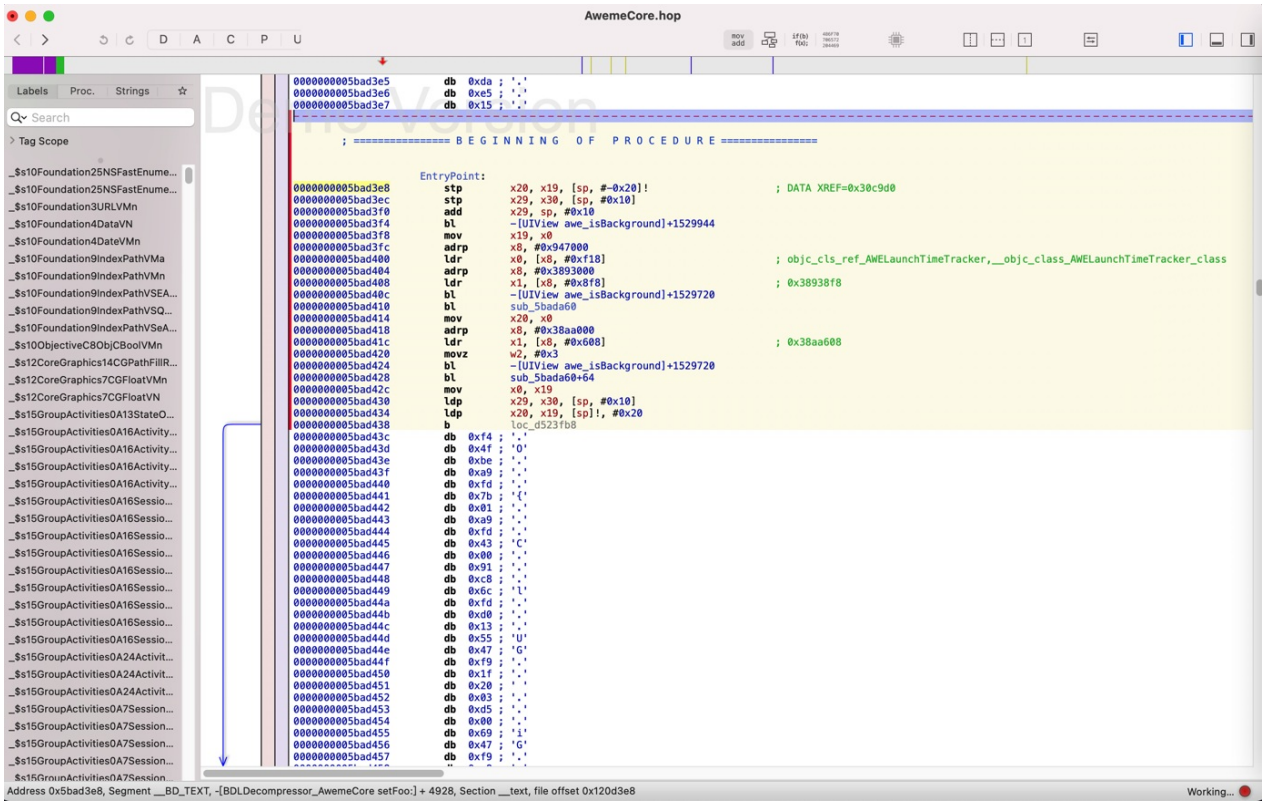
然后开始加载和分析：

- Loading Mach-O AArch64 file
  - Processing bindings
    - 

  - Reading Objective-C
    - Reading Objective-C super refs
      - 

    - Reading Objective-C class list
      - 

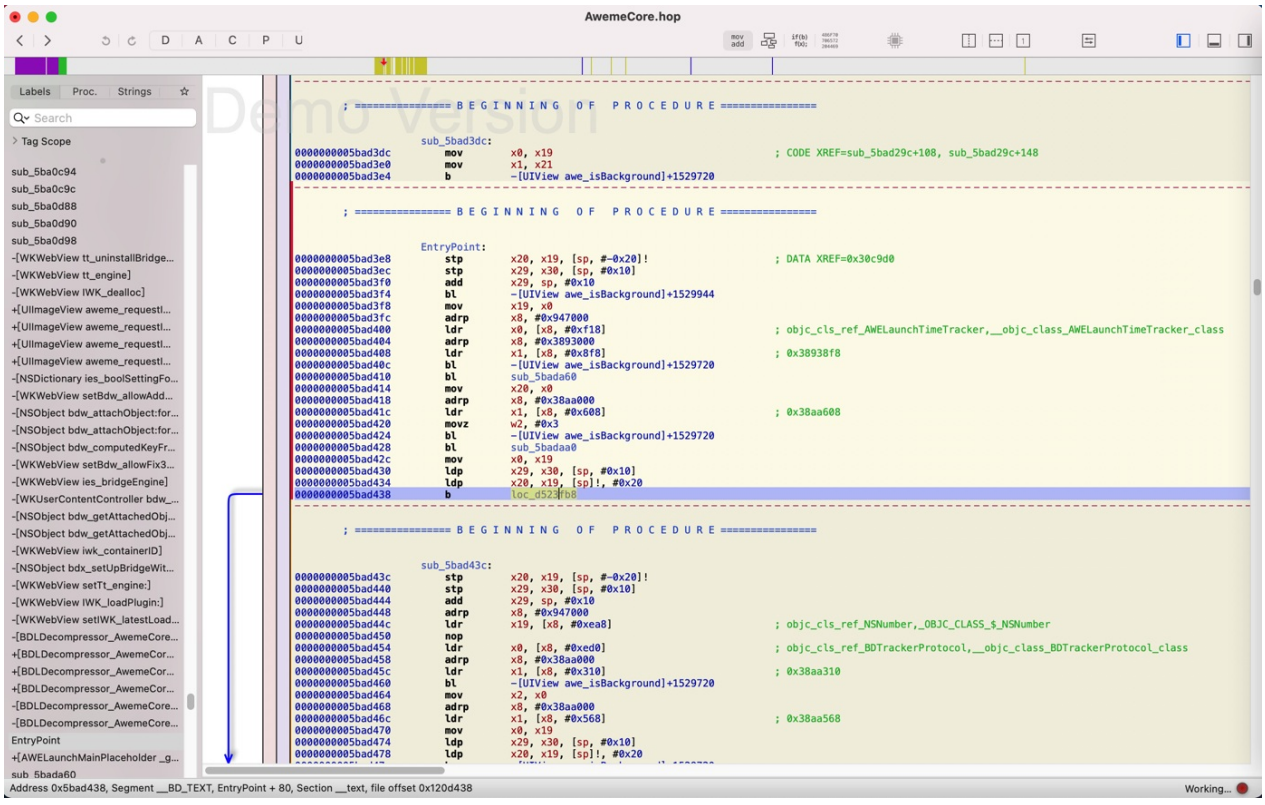
加载完毕，进入主页面：



缺点：目前总体有点卡顿

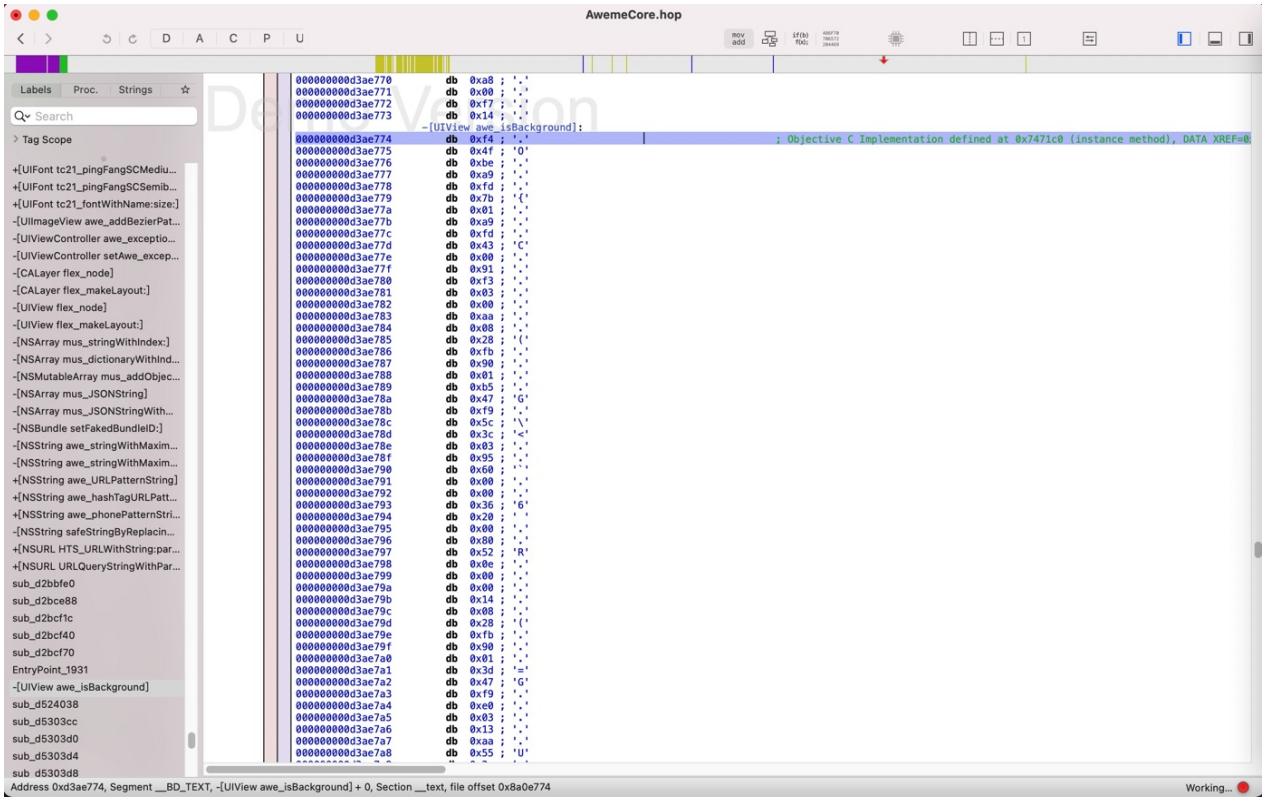
分析代码逻辑

接下来，就是如何具体分析逻辑了

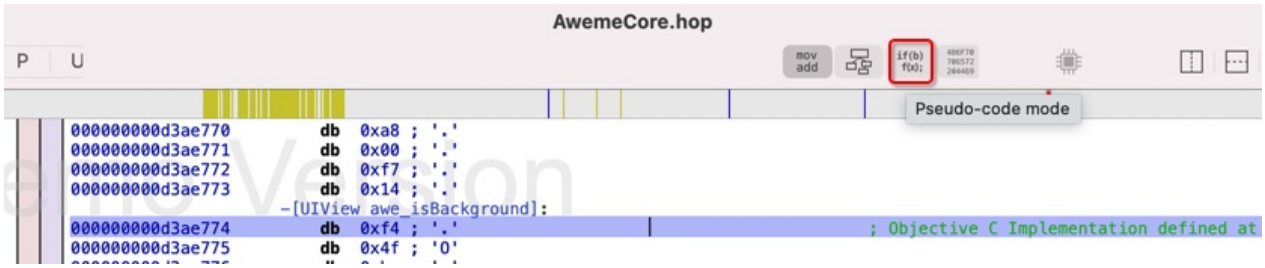


此处可以看到函数名：`awe_isBackground`

双击后，跳转到函数实现：



点击 尝试别人说的，切换到 伪代码 ObjC的



此处出现警告：

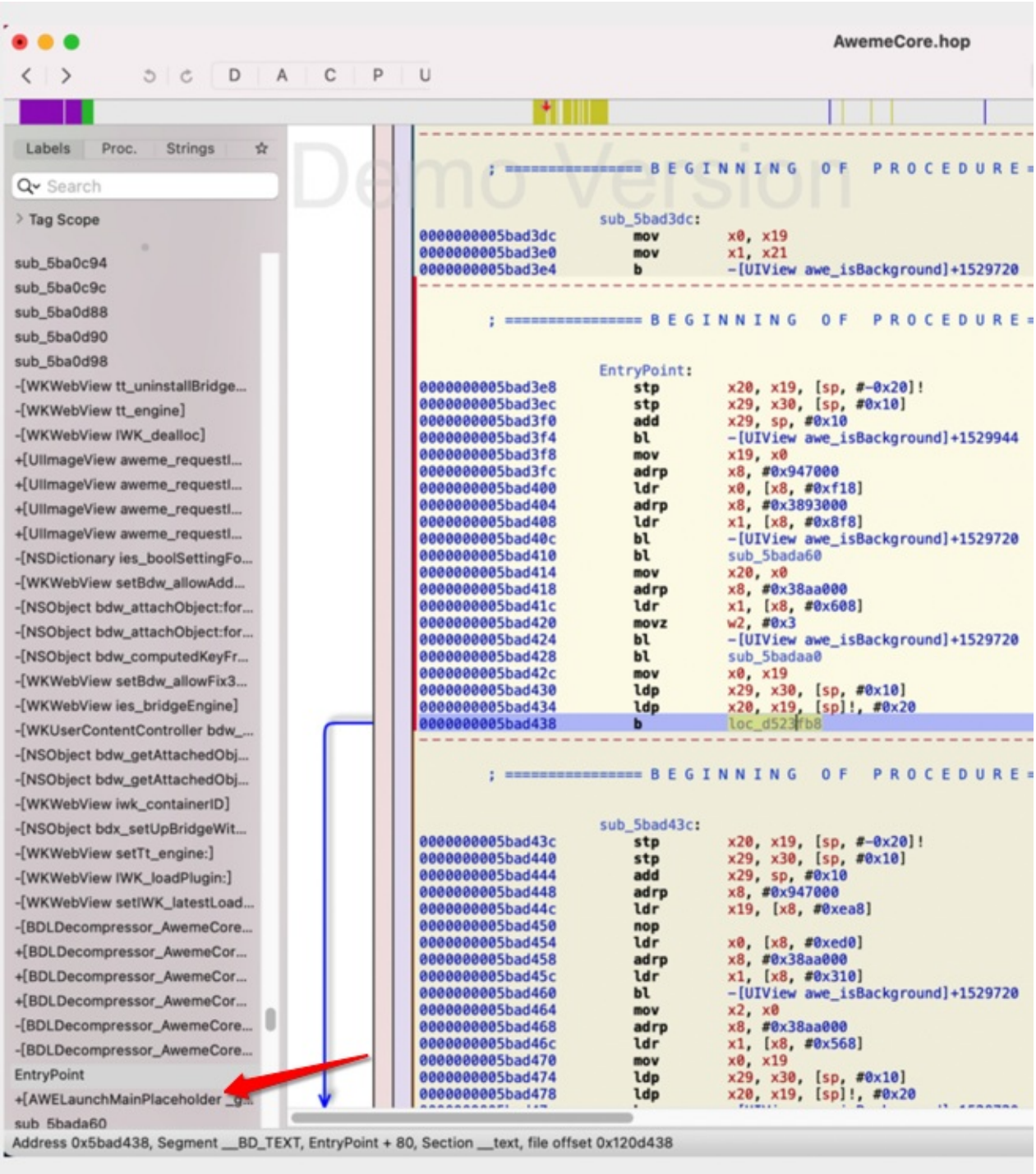
- No procedure at this address
  -

继续去找其他逻辑：比如，是否有越狱相关内容。

从左边的函数列表：

找找哪些和 启动相关的、初始化相关的

比如之前看到的，靠近entry的 `AWELaunchMainPlaceholder` 这种函数：

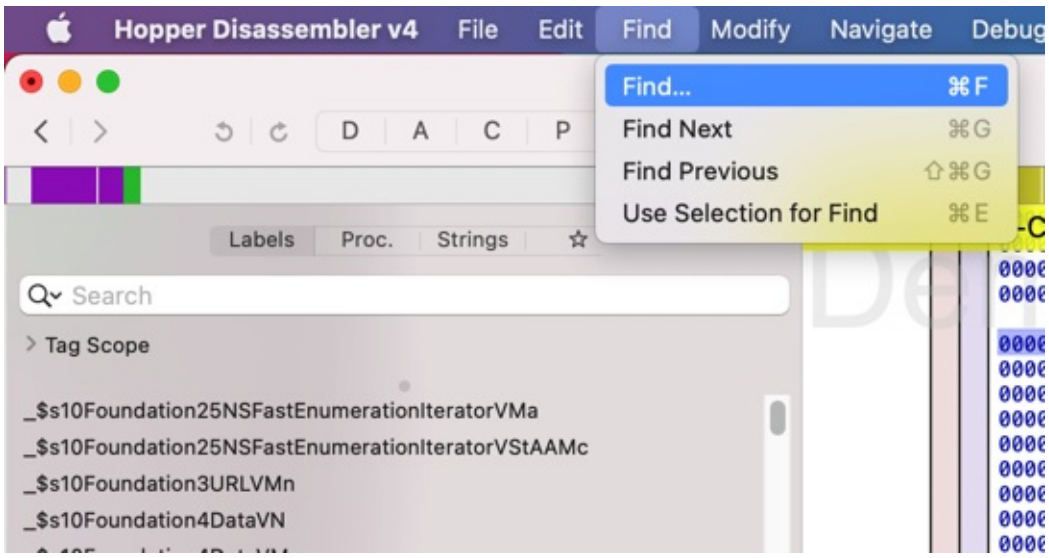


就很值得好好研究看看

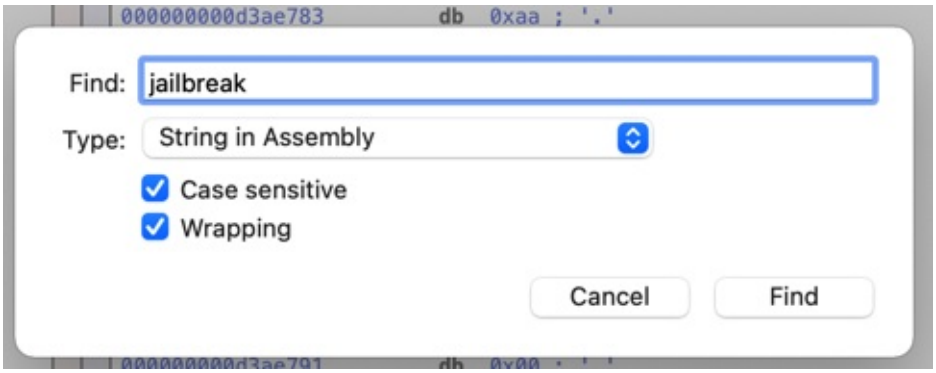
另外继续研究越狱相关：

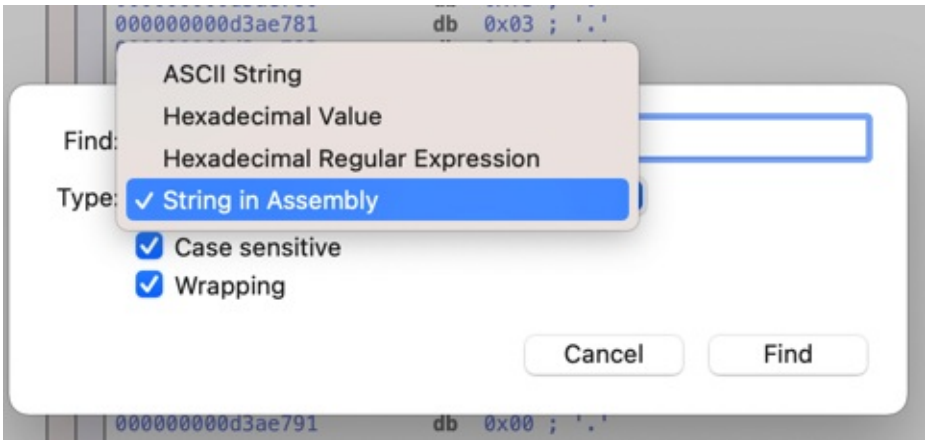
继续研究左边函数列表：

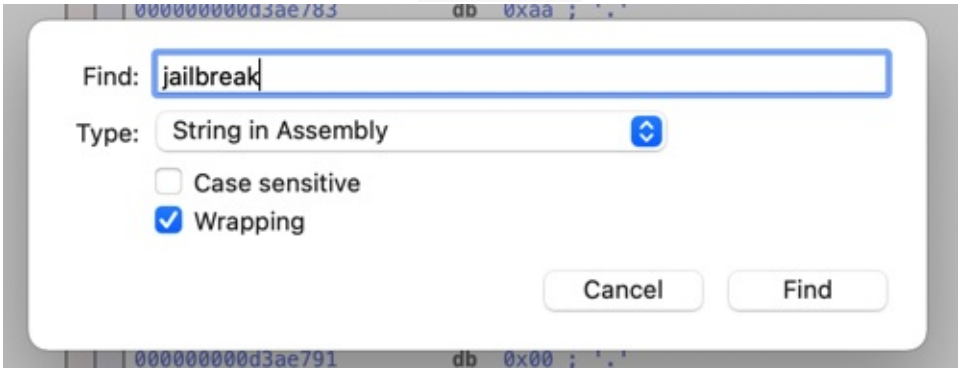
去试试，搜索 `Find -> Find`



越狱：`jailbreak`

- 

  - 看看Type还有哪些类型
    - 

  - 此处为了模糊搜索更多内容，所以取消勾选：`Case sensitive`
    - 

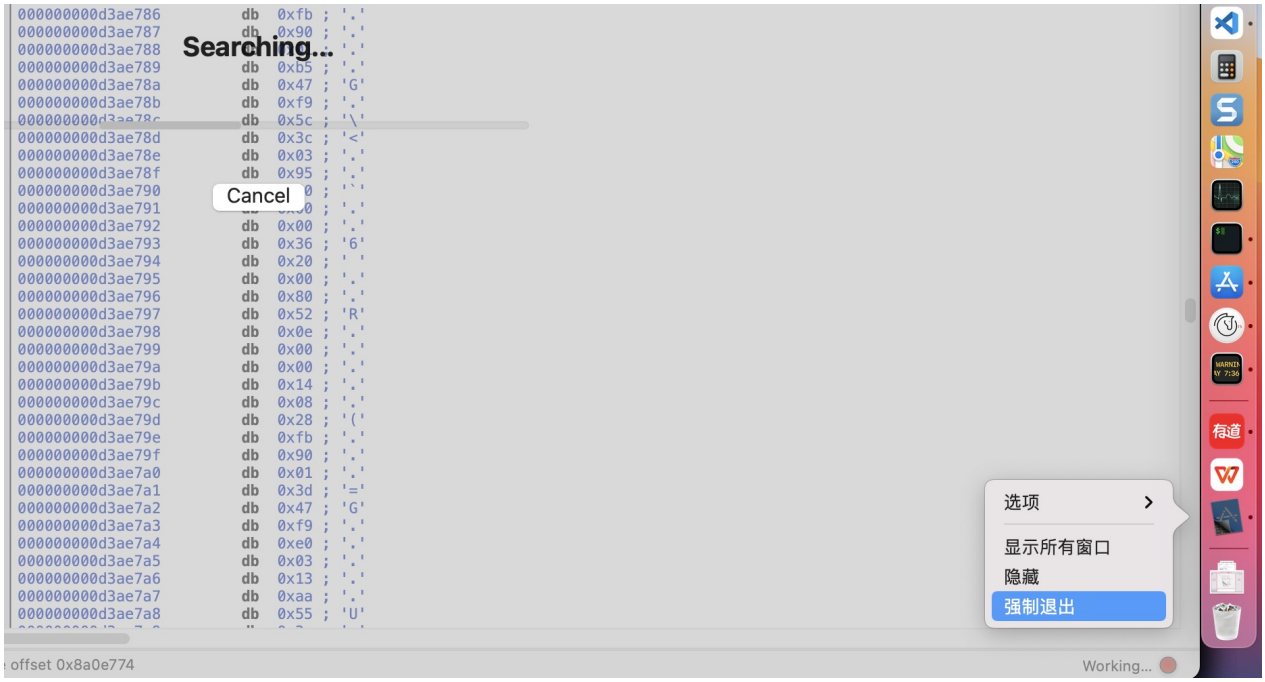
开始搜索：

此处搜索了很多分钟，仍没有结束。

后来是，等待了2天多，依旧没结束，所以放弃。

尝试点击Cancel时，已无法点击。

索性强制退出：



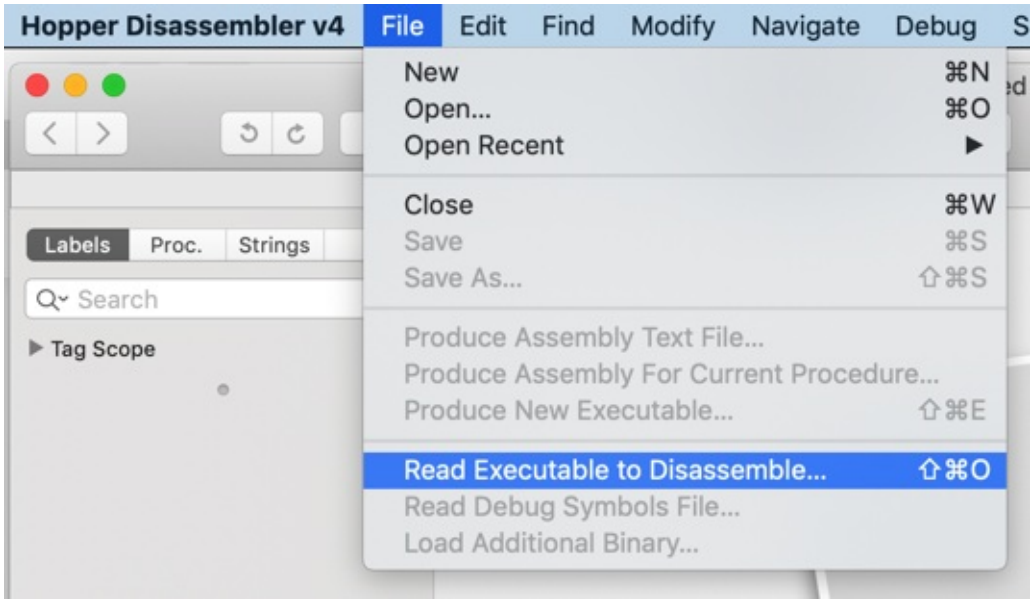
->

- 目前的结论
  - Hopper对于（包含逻辑和内容很多的）大的二进制，基本上无法正常使用。

Thunder迅雷

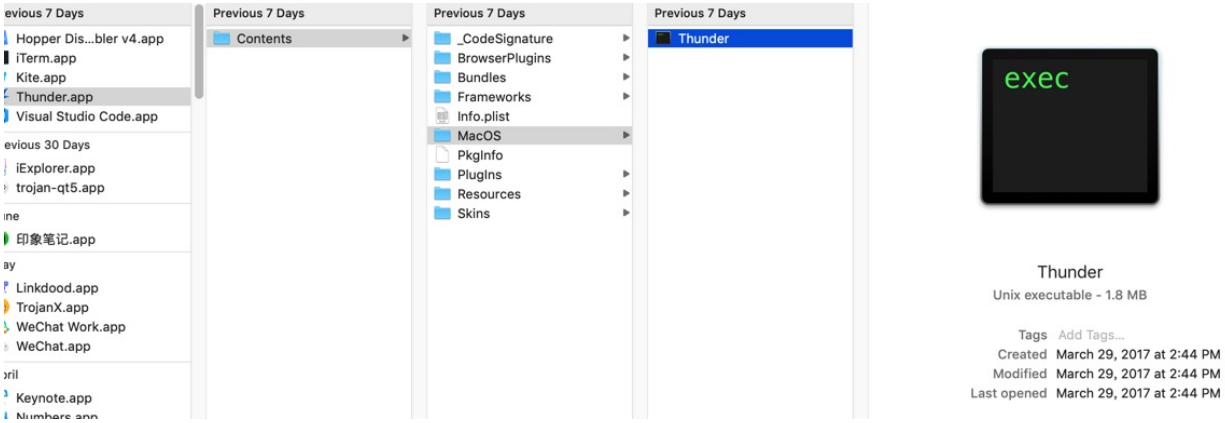
Hopper打开Thunder迅雷

`Hopper Disassembler v4` -> `File` -> `Read Executable to Disassemble`：

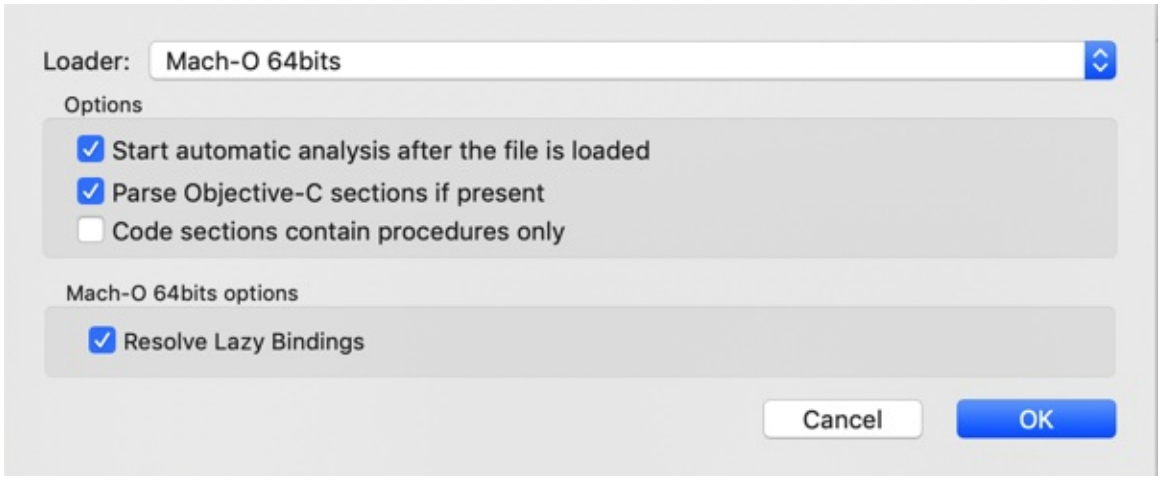


尝试去打开 `Mac` 的app：迅雷

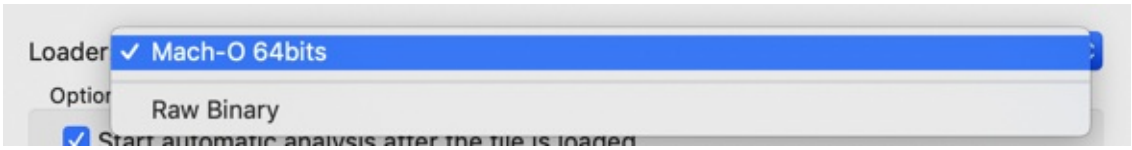
`Thunder.app/Contents/MacOS/Thunder`



弹框显示加载选项：

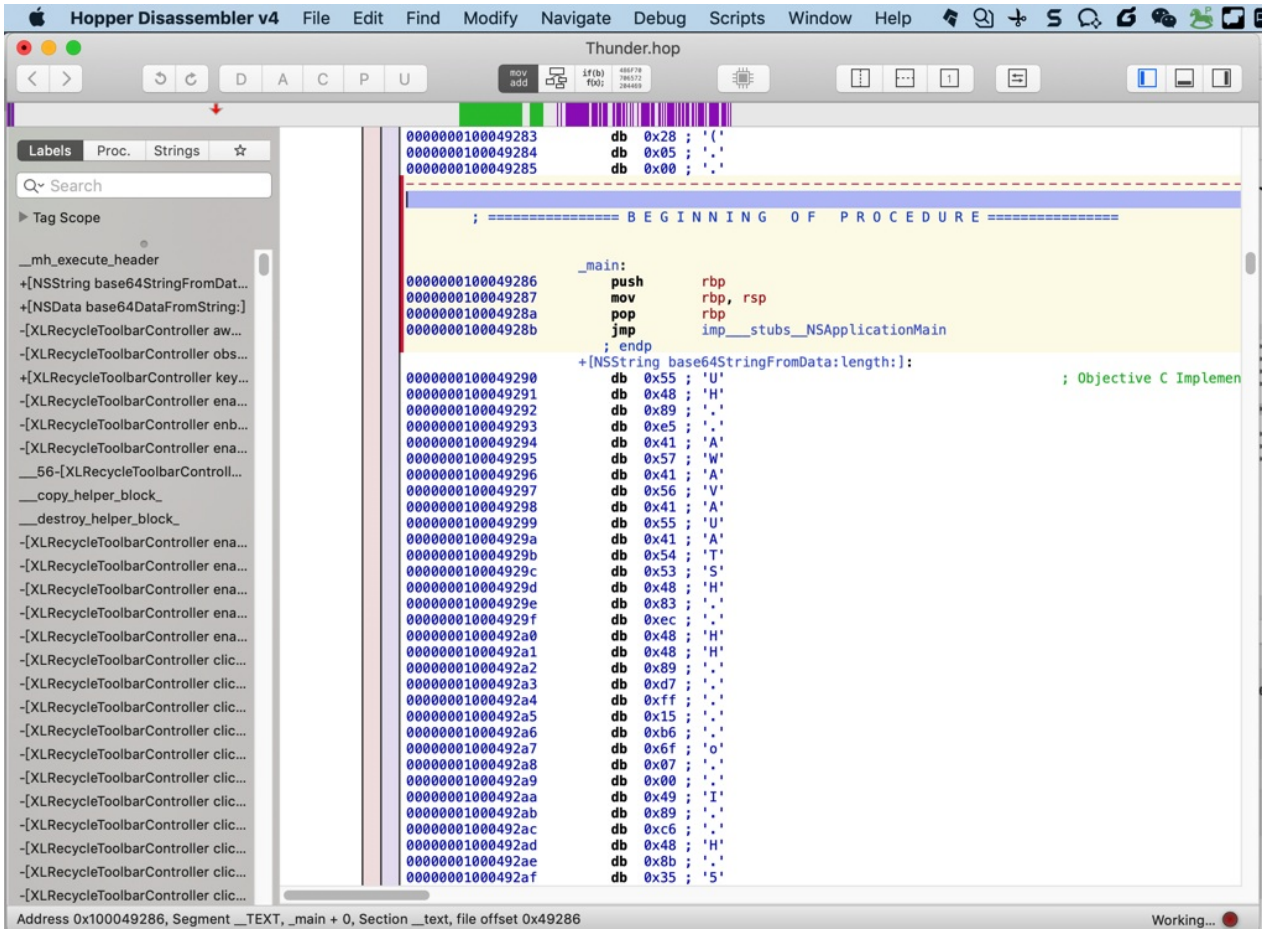


顺带看看Loader参数的其他可选性：

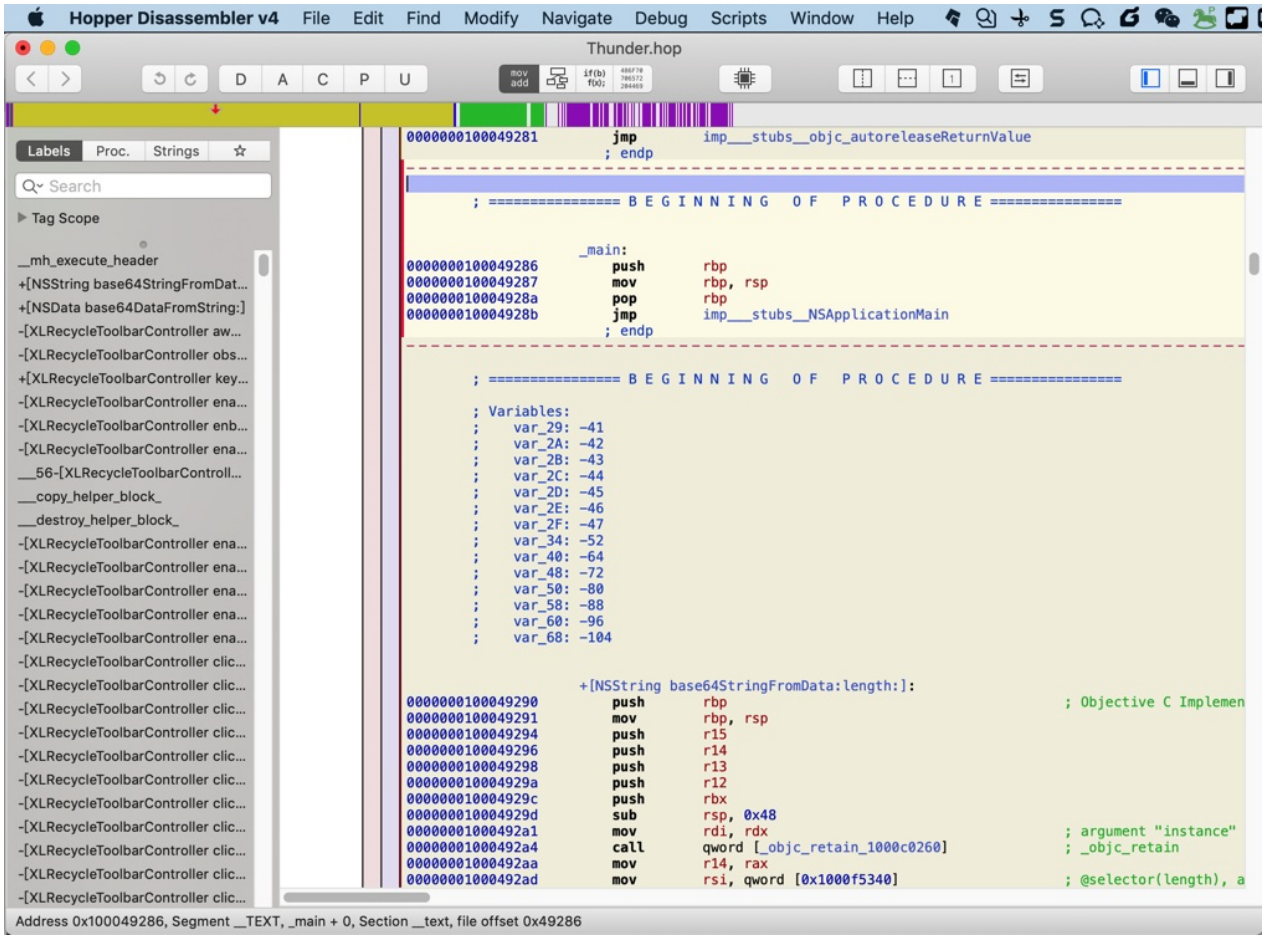


- Loader可选项
  - `Mach-O 64bits`
  - `Raw Binary`

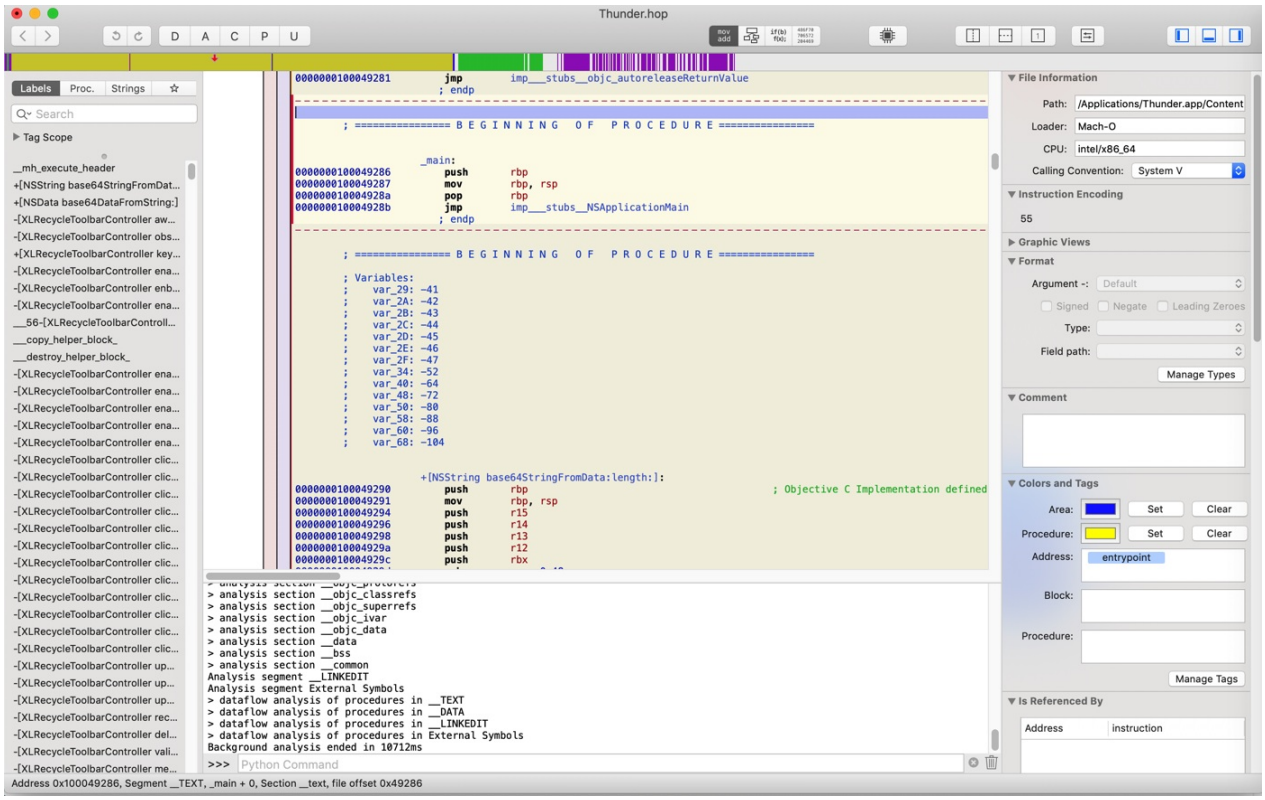
点击OK，即可加载分析：



开始加载和分析。很快分析完毕，效果是：



再去把右边和底部也开启显示：



底部的log是：

```
Hopper is ready
Mach-O 64bits file loaded
```

```
Starting background analysis
Analysis segment __TEXT
  analysis section __text
  analysis section __stubs
  analysis section __stub_helper
  analysis section __gcc_except_tab
  analysis section __objc_methname
  transform section __objc_methname to C strings
  analysis section __cstring
  transform section __cstring to C strings
  analysis section __const
  analysis section __objc_classname
  transform section __objc_classname to C strings
  analysis section __objc_methtype
  transform section __objc_methtype to C strings
  analysis section __ustring
  analysis section __swift3_typeref
  analysis section __swift3_reflstr
  analysis section __swift3_fieldmd
  analysis section __swift3_assocty
  analysis section __swift2_types
  analysis section __swift2_proto
  analysis section __swift3_capture
  analysis section __unwind_info
  analysis section __eh_frame
Analysis segment __DATA
  analysis section __nl_symbol_ptr
  analysis section __got
  analysis section __la_symbol_ptr
  analysis section __const
  analysis section __cfstring
  analysis section __objc_classlist
  analysis section __objc_nlclslist
  analysis section __objc_catlist
  analysis section __objc_protolist
  analysis section __objc_imageinfo
  analysis section __objc_const
  analysis section __objc_selrefs
  analysis section __objc_protorefs
  analysis section __objc_classrefs
  analysis section __objc_superrefs
  analysis section __objc_ivar
  analysis section __objc_data
  analysis section __data
  analysis section __bss
  analysis section __common
Analysis segment __LINKEDIT
Analysis segment External Symbols
  mark procedures
Analysis segment __TEXT
  analysis section __text
  disassemble section __text
  searching additional procedures in section __text
  analysis section __stubs
  disassemble section __stubs
  searching additional procedures in section __stubs
  analysis section __stub_helper
  disassemble section __stub_helper
  searching additional procedures in section __stub_helper
  analysis section __gcc_except_tab
  analysis section __objc_methname
  analysis section __cstring
  analysis section __const
  analysis section __objc_classname
  analysis section __objc_methtype
  analysis section __ustring
  analysis section __swift3_typeref
  analysis section __swift3_reflstr
  analysis section __swift3_fieldmd
  analysis section __swift3_assocty
  analysis section __swift2_types
  analysis section __swift2_proto
  analysis section __swift3_capture
  analysis section __unwind_info
  analysis section __eh_frame
Analysis segment __DATA
  analysis section __nl_symbol_ptr
  analysis section __got
```

```
  analysis section __la_symbol_ptr
  analysis section __const
  analysis section __cfstring
  analysis section __objc_classlist
  analysis section __objc_nlclslist
  analysis section __objc_catlist
  analysis section __objc_protolist
  analysis section __objc_imageinfo
  analysis section __objc_const
  analysis section __objc_selrefs
  analysis section __objc_protorefs
  analysis section __objc_classrefs
  analysis section __objc_superrefs
  analysis section __objc_ivar
  analysis section __objc_data
  analysis section __data
  analysis section __bss
  analysis section __common
Analysis segment __LINKEDIT
Analysis segment External Symbols
  dataflow analysis of procedures in __TEXT
  dataflow analysis of procedures in __DATA
  dataflow analysis of procedures in __LINKEDIT
  dataflow analysis of procedures in External Symbols
Background analysis ended in 10712ms
```

显示的是加载和分析的过程。

看到有 `__swift3_typeref`，或许表示此处Mac版Thunder是用Swift写的?

继续看看其他的:

- File Information

  -
    - `Calling Convention` 除了 `System V` 外还有 `Microsoft x64`

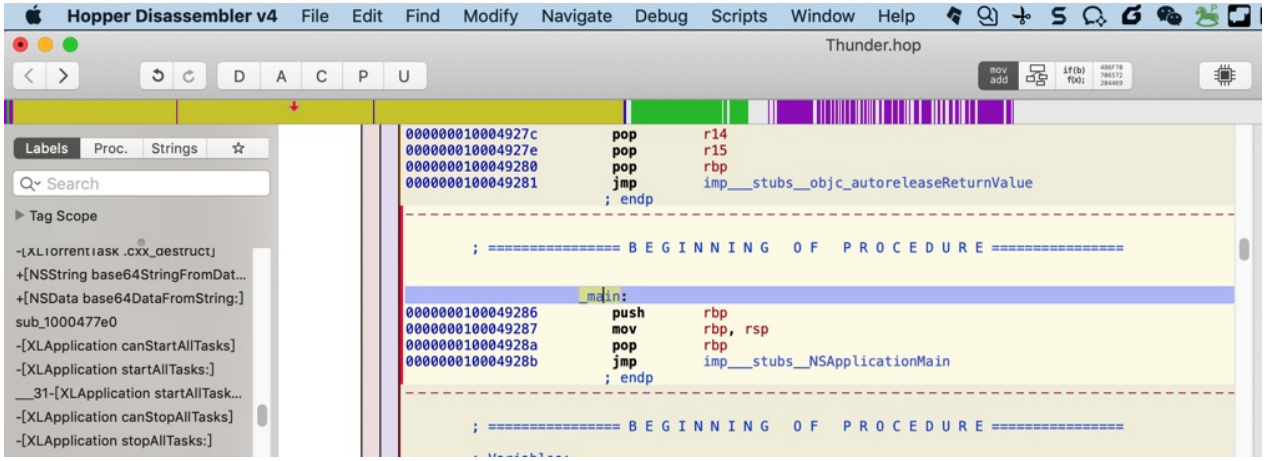
中间部分解析出的内容:

```
        ; ============== B E G I N N I N G   O F   P R O C E D U R E ==============

        ; Variables
        ;    var_29   41
        ;    var_2A   42
        ;    var_2B   43
        ;    var_2C   44
        ;    var_2D   45
        ;    var_2E   46
        ;    var_2F   47
        ;    var_34   52
        ;    var_40   64
        ;    var_48   72
        ;    var_50   80
        ;    var_58   88
        ;    var_60   96
        ;    var_68   104
```

从 `Beginning of procedure` 感觉是：main函数 入口处

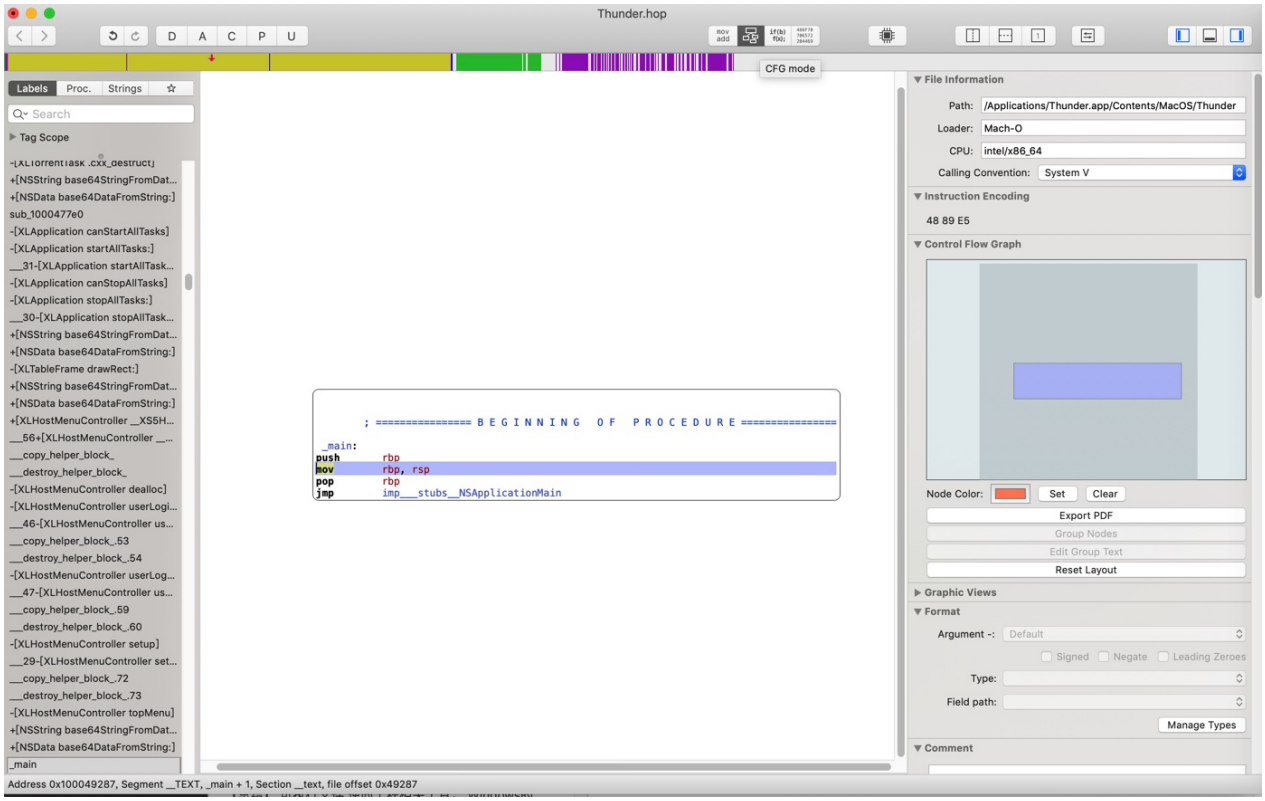
看到了，上面就是 `_main` ：

```
        ; ================ B E G I N N I N G   O F   P R O C E D U R E ================


                        _main
0000000100049286        push    rbp
0000000100049287        mov     rbp, rsp
000000010004928a        pop     rbp
000000010004928b        jmp     imp___stubs__NSApplicationMain
```



感觉是：Hopper自动帮我们定位到了main入口所在的位置

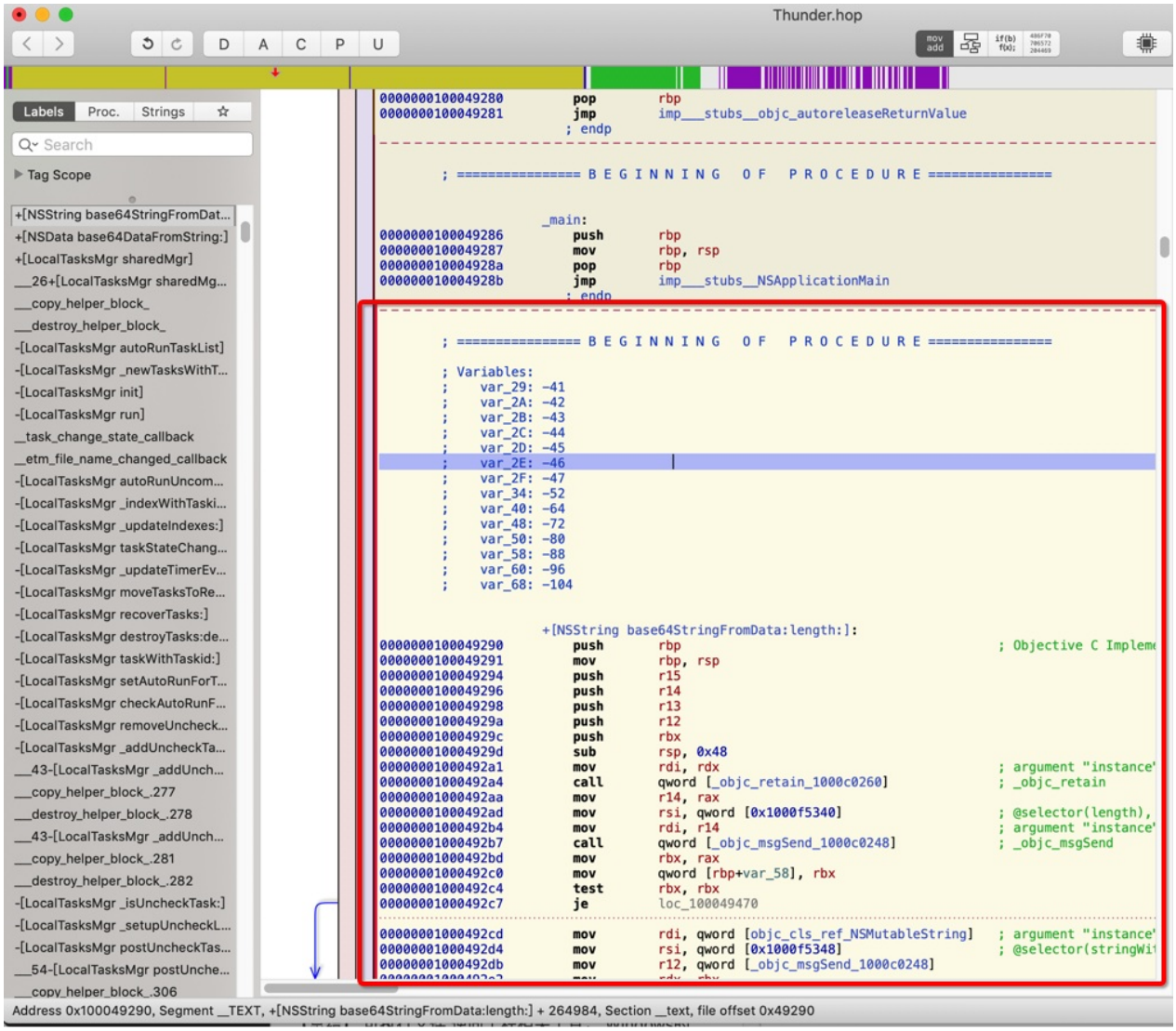
从 `main` 切换到 `CTF mode` ，结果显示的内容，很简单：



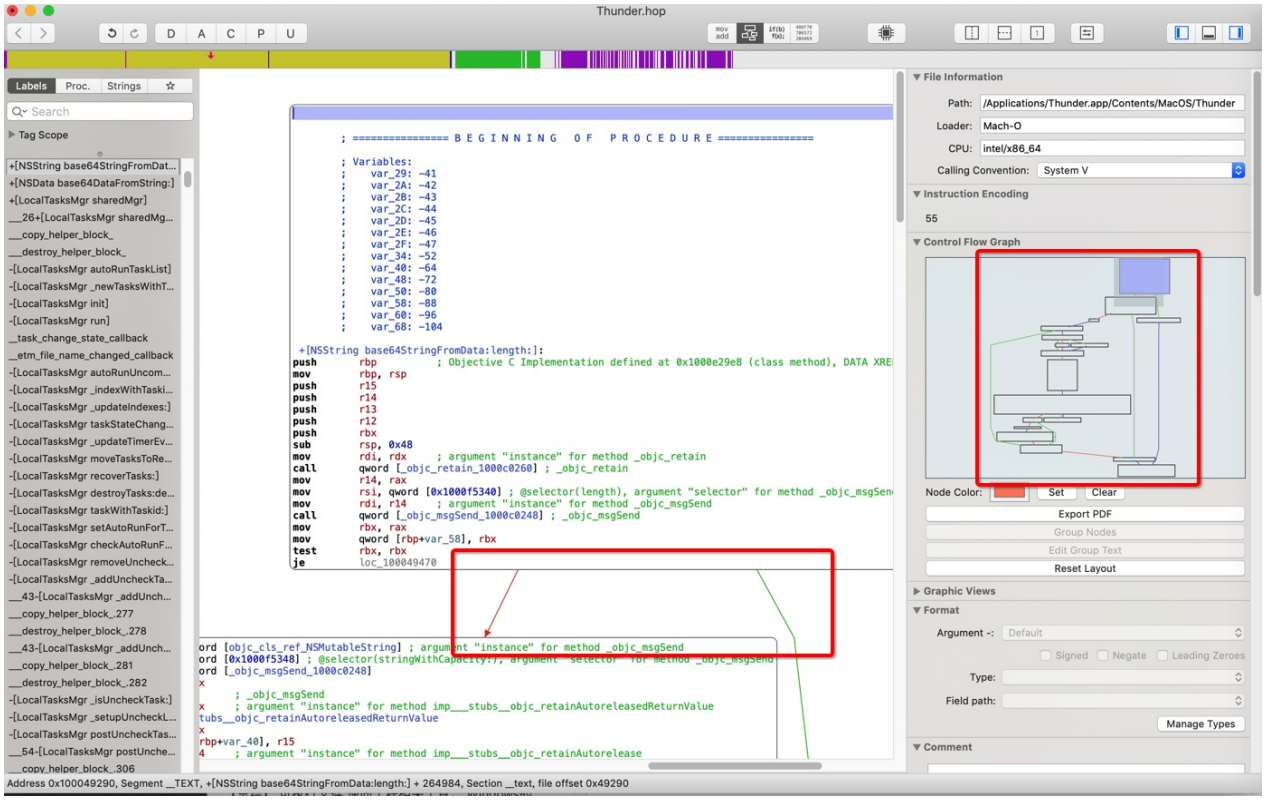
没有我希望的：函数调用顺序

不过后来搞懂了：是此处函数本身简单。

换个下面的函数代码段：

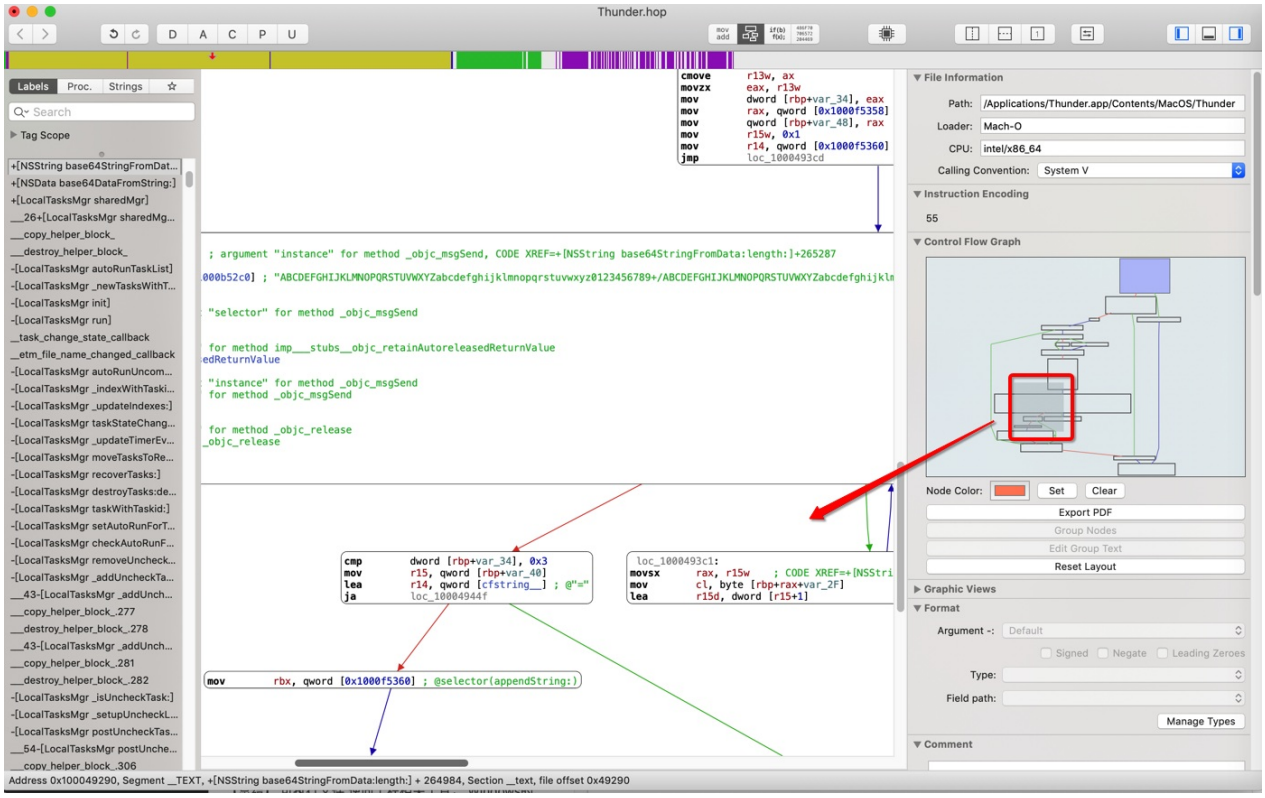


切换到 `CTF mode` 后，就可以看到希望看到的：调用关系了：



以及右边有 `Control Flow Graph` = 控制流程图 了

移动后，随时可以看到占整体的比例和位置：



重点再去看看，找找，左边的：

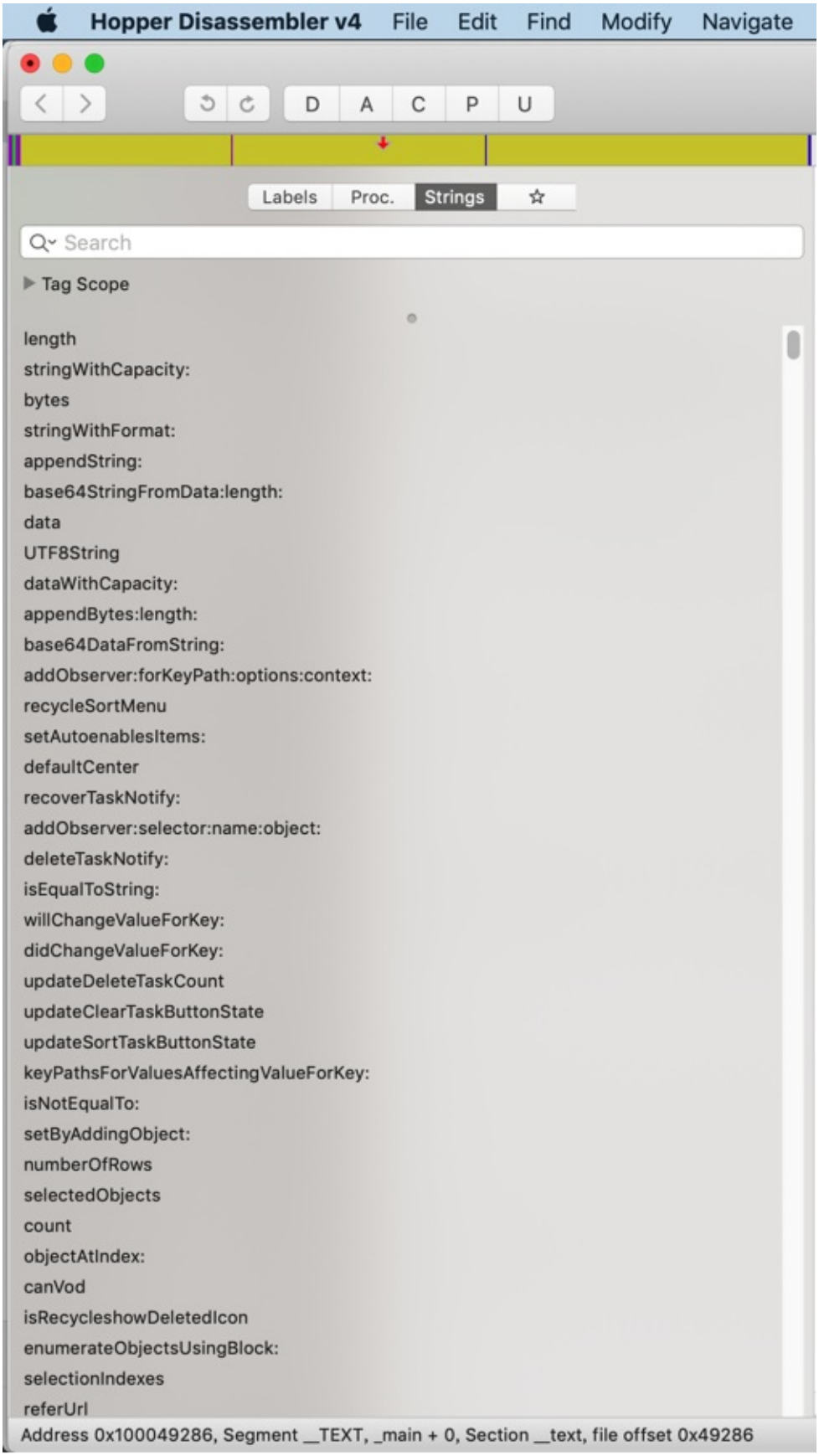
- 函数列表
  - Labels


Hopper Disassembler v4
File
Edit
Find
Modify
D
A
C
P
U
Labels
Proc.
Strings
Search
Tag Scope
__mh_execute_header
+[NSString base64StringFromData:length:]
+[NSData base64DataFromString:]
-[XLRecycleToolbarController awakeFromNib]
-[XLRecycleToolbarController observeValueForKeyPath:ofObject:change:c...
+[XLRecycleToolbarController keyPathsForValuesAffectingValueForKey:]
-[XLRecycleToolbarController enableClearToolbarItem]
-[XLRecycleToolbarController enbaleSubtitleToolbarItem]
-[XLRecycleToolbarController enableDeleteTaskRecordItem]
__56-[XLRecycleToolbarController enableDeleteTaskRecordItem]_block_in...
___copy_helper_block_
___destroy_helper_block_
-[XLRecycleToolbarController enableSelectAllMenuItem]
-[XLRecycleToolbarController enableMoveTasksToHeadMenuItem]
-[XLRecycleToolbarController enableMoveTasksToTailMenuItem]
-[XLRecycleToolbarController enableOpenReferUrl]
-[XLRecycleToolbarController enablePropertyMenuItem]
-[XLRecycleToolbarController clickPropertyMenu:]
-[XLRecycleToolbarController clickRecoverTaskMenu:]
-[XLRecycleToolbarController clickRemoveTaskMenu:]
-[XLRecycleToolbarController clickClearTaskMenu:]
-[XLRecycleToolbarController clickDeleteTaskRecords:]
-[XLRecycleToolbarController clickSearchSubtitleMenu:]
-[XLRecycleToolbarController clickSelectAll:]
-[XLRecycleToolbarController clickRedownloadTask:]
-[XLRecycleToolbarController clickOpenReferUrlPage:]
-[XLRecycleToolbarController clickCopyUrlToClippedBoard:]
-[XLRecycleToolbarController clickSortMenuAction:]
-[XLRecycleToolbarController clickSortMenuDidChange:]
-[XLRecycleToolbarController updateDeleteTaskCount]
-[XLRecycleToolbarController updateClearTaskButtonState]
-[XLRecycleToolbarController updateSortTaskButtonState]
-[XLRecycleToolbarController recoverTaskNotify:]
-[XLRecycleToolbarController deleteTaskNotify:]
-[XLRecycleToolbarController validateMenuItem:]
-[XLRecycleToolbarController menuNeedsUpdate:]
Address 0x100049290, Segment __TEXT, +[NSString base64StringFromData:leng

- Proc

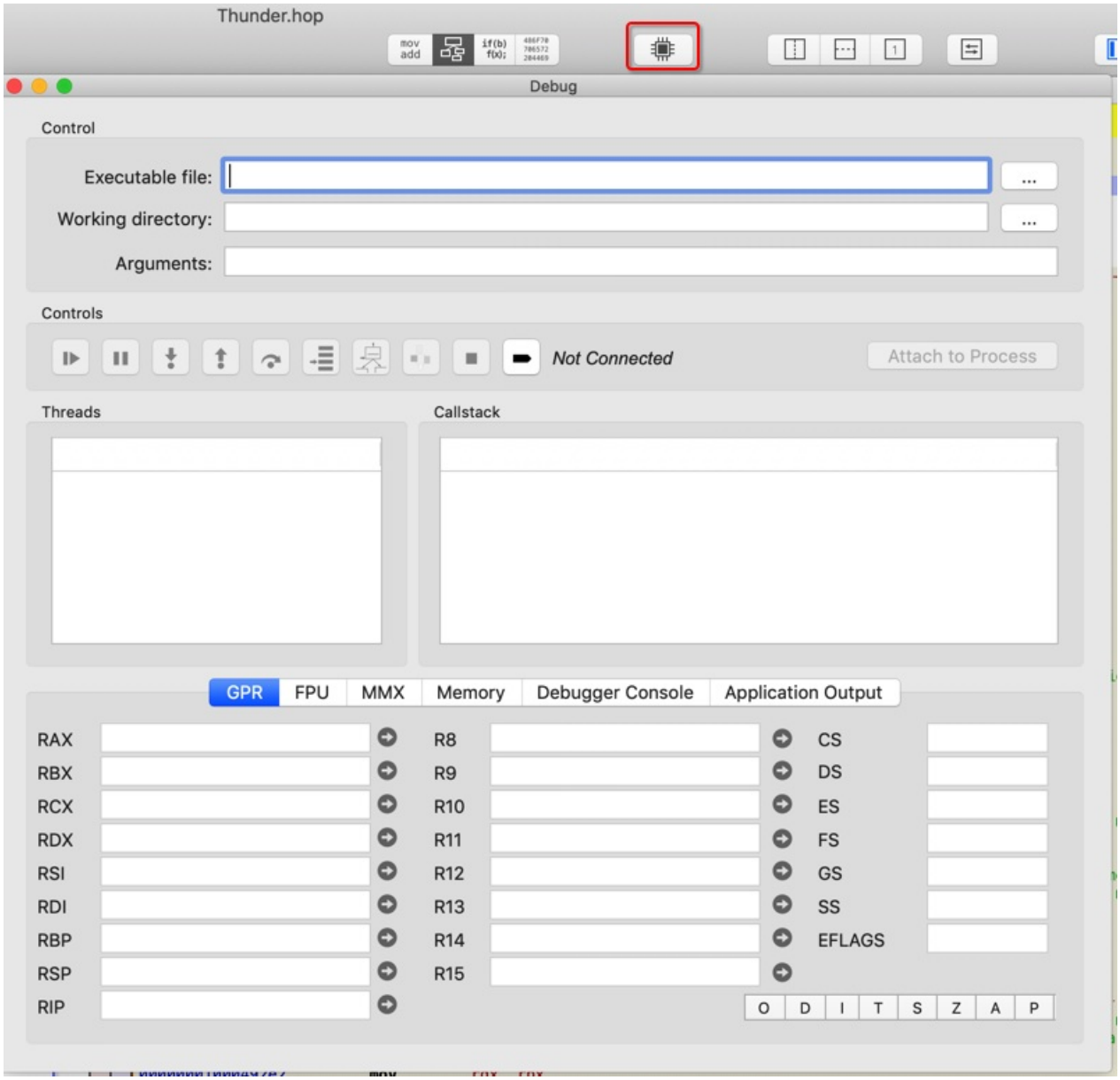

Hopper Disassembler v4
File
Edit
Find
Modify
Navigate
D
A
C
P
U
Labels
Proc.
Strings
Search
Tag Scope
+[NSString base64StringFromData:length:]
+[NSData base64DataFromString:]
-[XLRecycleToolbarController awakeFromNib]
-[XLRecycleToolbarController observeValueForKeyPath:ofObject:change:context:]
+[XLRecycleToolbarController keyPathsForValuesAffectingValueForKey:]
-[XLRecycleToolbarController enableClearToolbarItem]
-[XLRecycleToolbarController enbaleSubtitleToolbarItem]
-[XLRecycleToolbarController enableDeleteTaskRecordItem]
___56-[XLRecycleToolbarController enableDeleteTaskRecordItem]_block_invoke
___copy_helper_block_
___destroy_helper_block_
-[XLRecycleToolbarController enableSelectAllMenuItem]
-[XLRecycleToolbarController enableMoveTasksToHeadMenuItem]
-[XLRecycleToolbarController enableMoveTasksToTailMenuItem]
-[XLRecycleToolbarController enableOpenReferUrl]
-[XLRecycleToolbarController enablePropertyMenuItem]
-[XLRecycleToolbarController clickPropertyMenu:]
-[XLRecycleToolbarController clickRecoverTaskMenu:]
-[XLRecycleToolbarController clickRemoveTaskMenu:]
-[XLRecycleToolbarController clickClearTaskMenu:]
-[XLRecycleToolbarController clickDeleteTaskRecords:]
-[XLRecycleToolbarController clickSearchSubtitleMenu:]
-[XLRecycleToolbarController clickSelectAll:]
-[XLRecycleToolbarController clickRedownloadTask:]
-[XLRecycleToolbarController clickOpenReferUrlPage:]
-[XLRecycleToolbarController clickCopyUrlToClippedBoard:]
-[XLRecycleToolbarController clickSortMenuAction:]
-[XLRecycleToolbarController clickSortMenuDidChange:]
-[XLRecycleToolbarController updateDeleteTaskCount]
-[XLRecycleToolbarController updateClearTaskButtonState]
-[XLRecycleToolbarController updateSortTaskButtonState]
-[XLRecycleToolbarController recoverTaskNotify:]
-[XLRecycleToolbarController deleteTaskNotify:]
-[XLRecycleToolbarController validateMenuItem:]
-[XLRecycleToolbarController menuNeedsUpdate:]
-[XLRecycleToolbarController dragTableRow:to:]
Address 0x100049286, Segment __TEXT, _main + 0, Section __text, file offset 0x49286

- Strings



继续看看其他的：

点击 芯片图标 按钮，弹出 调试窗口 ：



Hopper分析Thunder逻辑

对于 `Thunder迅雷` 的基本界面：

想要去研究其中对应的 `精选` 、 `搜索` 、 `应用` 之类的内容。

去搜：

`search`

第一个，看起来就是我们希望的要找的

```
-[XLRecycleToolbarController clickSearchSubtitleMenu:]:
```

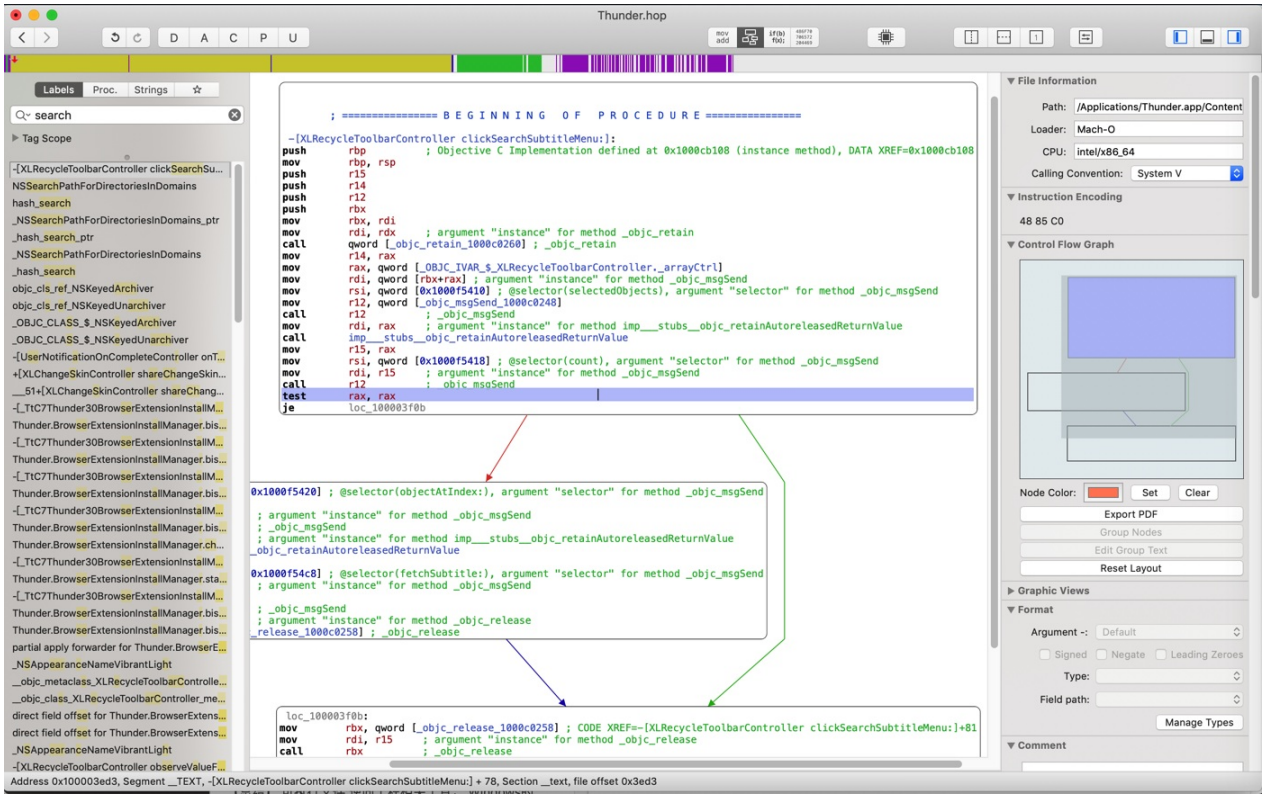
点击 `搜索` 子菜单

-》那顶部的5个：

精选 正在下载 已完成 搜索 应用

就应该叫做 主菜单了

看了看代码：



也看不出什么头绪

十秒逆向九秒猜

再去找 精选

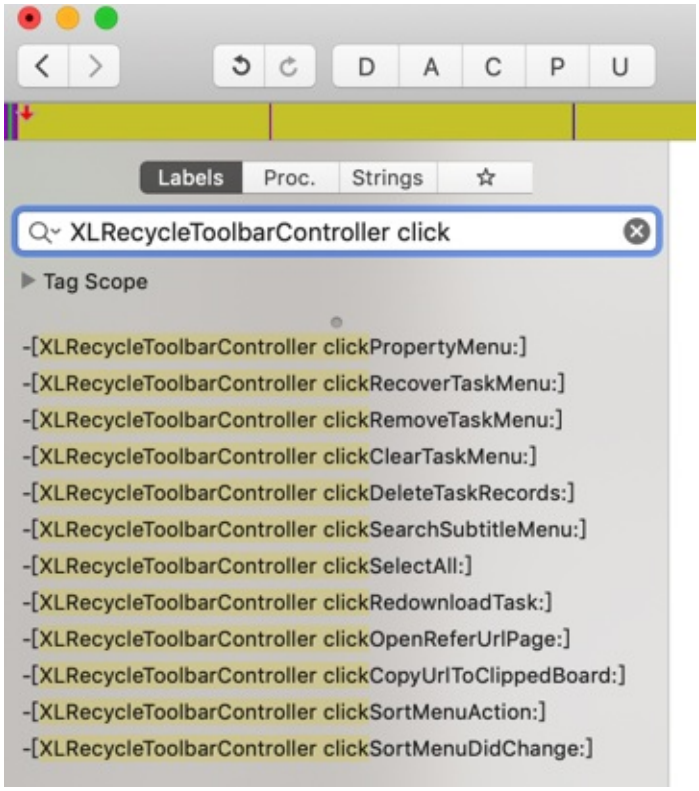
但是不知道英文如何翻译 精选 才能找到

不过想到了，去找类似的

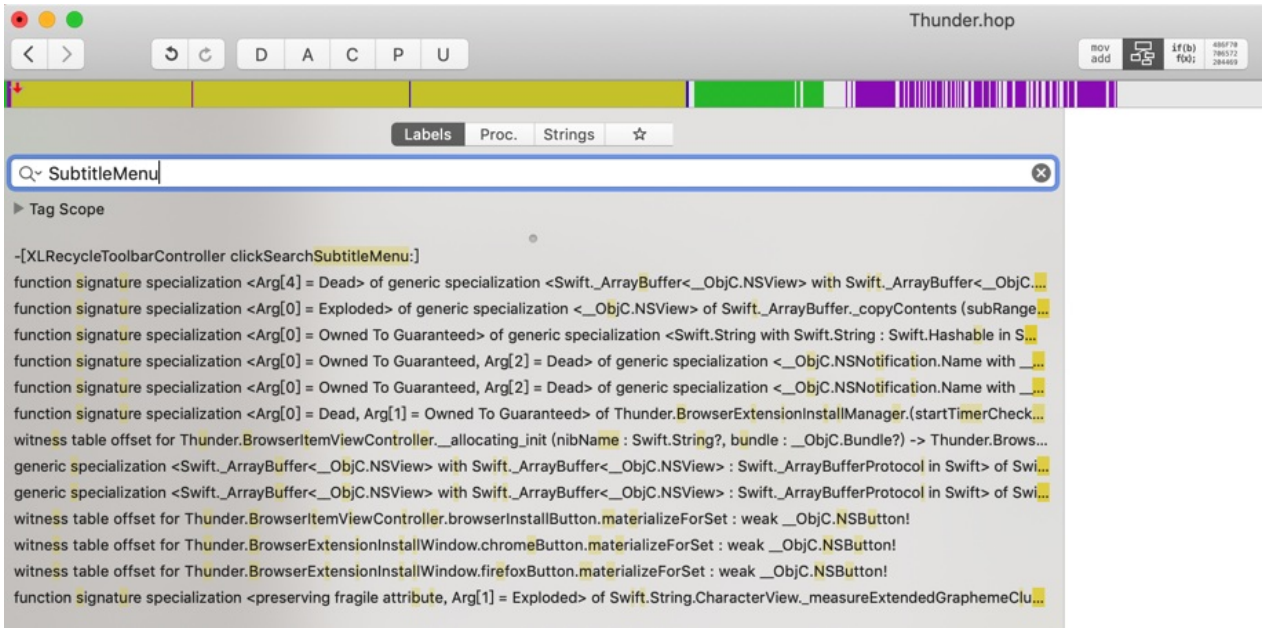
`SubtitleMenu`

或

`XLRecycleToolbarController click`



只有其他的一些 但不是我们要的

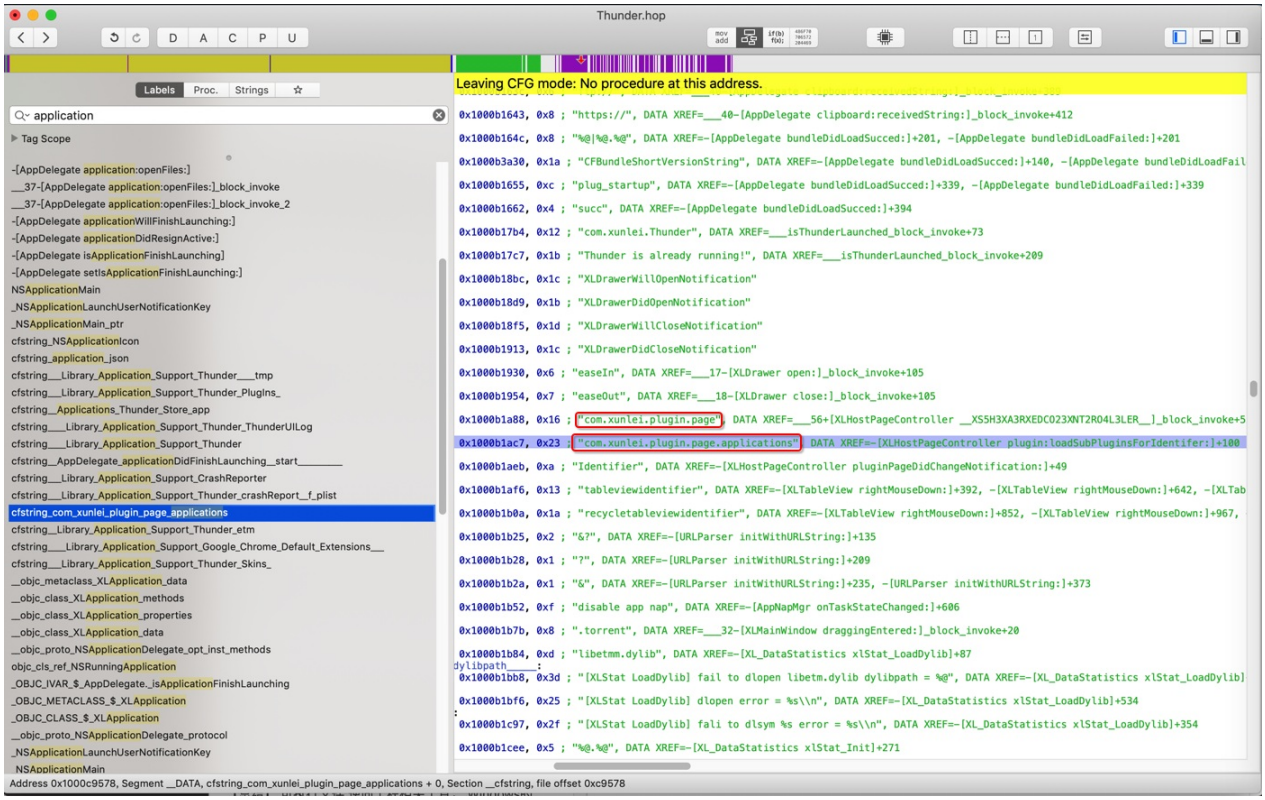


也没有我们要的

搜：`jingxuan`，也没有。

再去找找其他的：`application`

找到：




```
00000001000c9558        dq      ___CFConstantStringClassReference, 0x7c8, 0x1000b1a88, 0x16 ; "com.xunlei.plugin.page", DATA XREF ___56 [XLHostPageController __XS5H3XA3RXEDCO23XNT2RO4L3LER__]_block_invoke 55
                                cfstring_com_xunlei_plugin_page_applications
00000001000c9578        dq      ___CFConstantStringClassReference, 0x7c8, 0x1000b1ac7, 0x23 ; "com.xunlei.plugin.page.applications", DATA XREF [XLHostPageController plugin loadSubPluginsForIdentifer ] 100
                                cfstring_Identifier
```

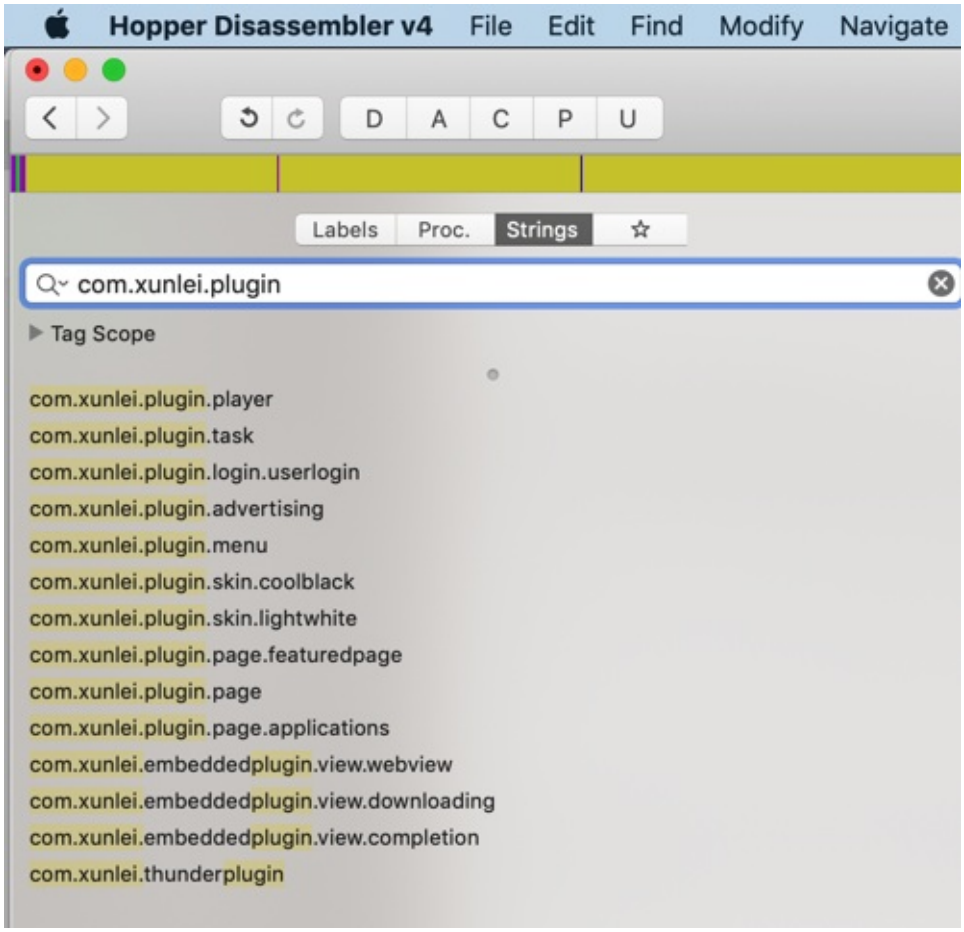
-》或许还有其他的

`com.xunlei.plugin.page.xxx`

？

`labels` 中没找到

不过发现strings中有一些：



不过其中的：

```
00000001000ab6a4        db        "com.xunlei.embeddedplugin.view.webview", 0 ; DATA XREF cfstring_com_xunlei_embedde
dplugin_view_webview
```

反推，倒是很可能是此处的：`精选`

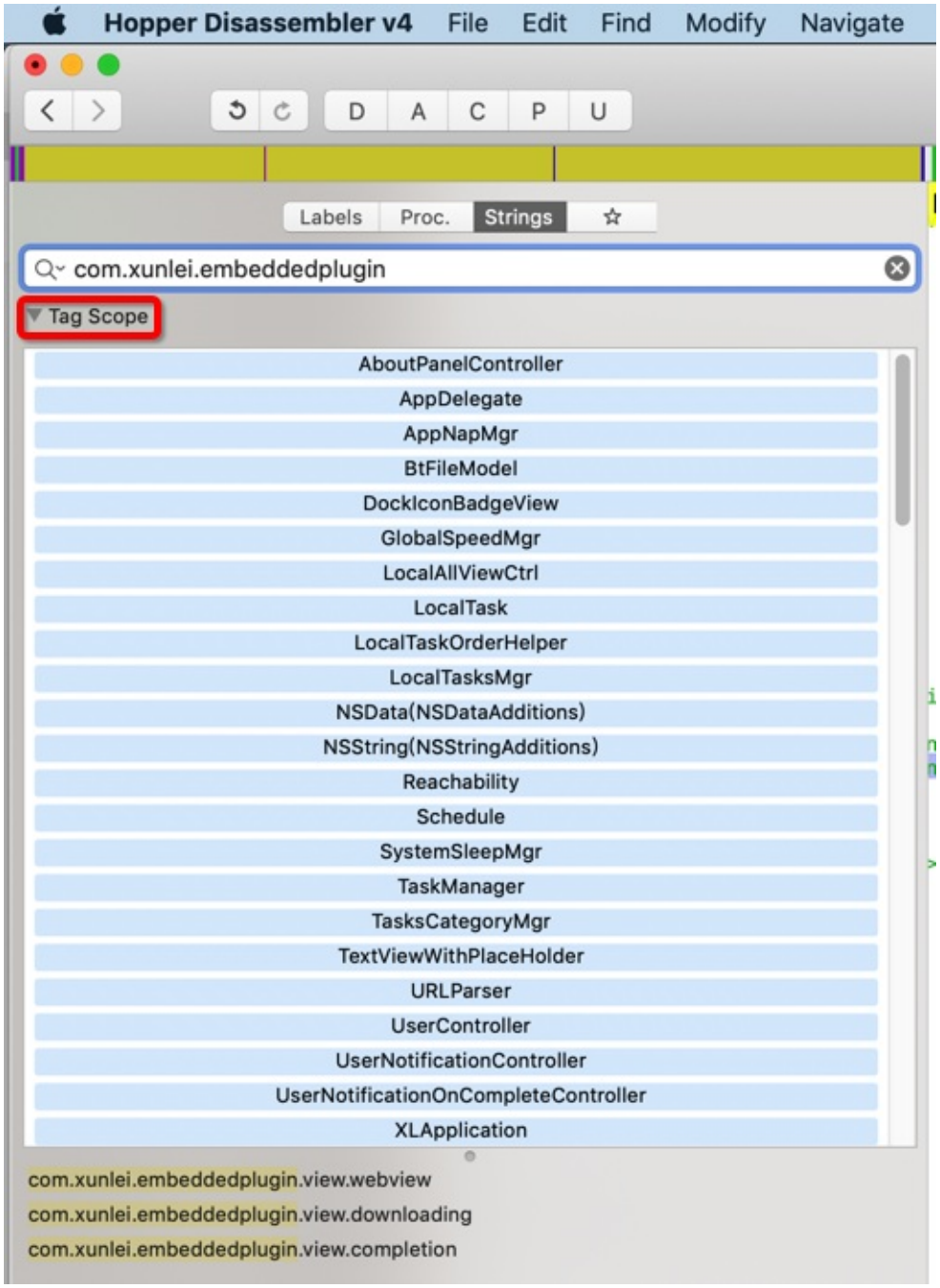
表示用webview 显示 精选内容

那再去搜搜：

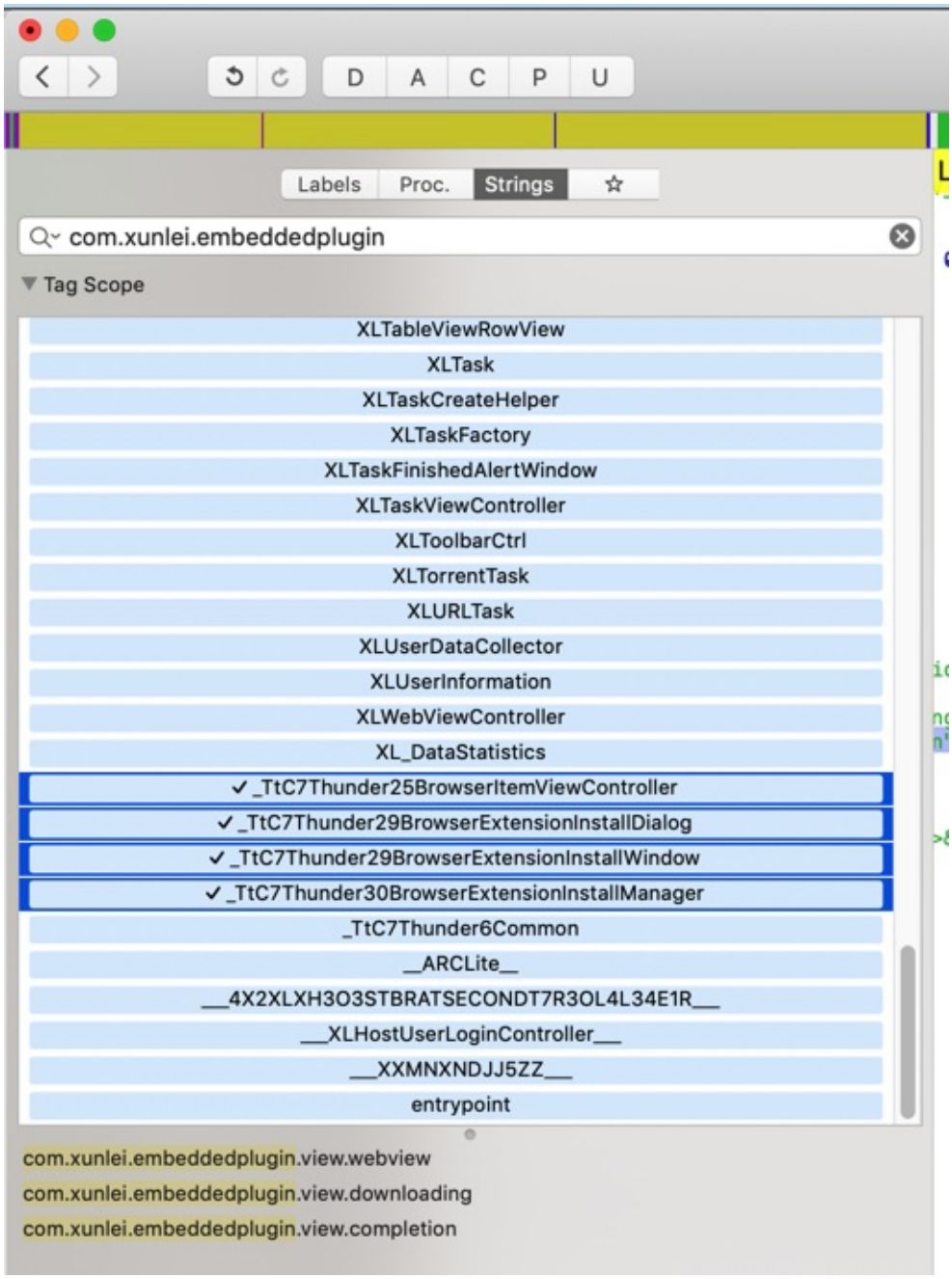
- `com.xunlei.embeddedplugin.view`
- `com.xunlei.embeddedplugin`

没有找到其他的

顺带看到一个 `Tag Scope`



是一个：好像是 内部的类 方法 函数 的列表



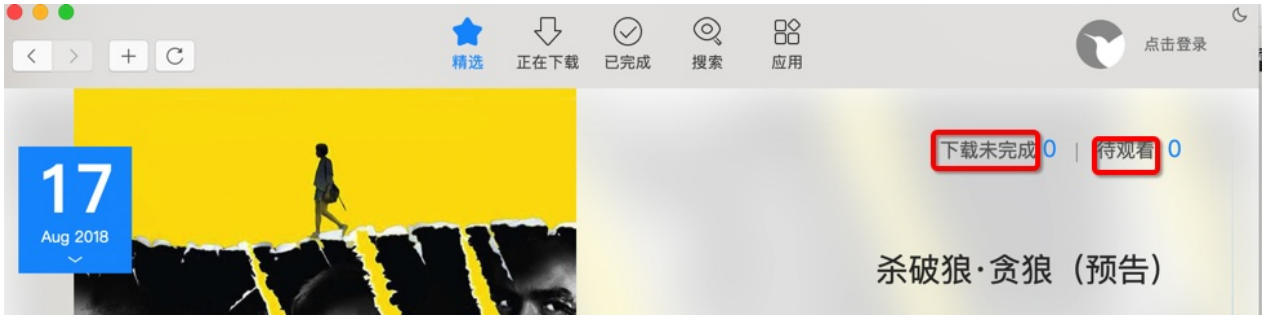
另外顺带看看：

- `Proc` = `Procedure` = `进程` = `函数`

- Strings

-

去看看app截图：



有下载未完成 待观看

或许能找到这些字符串?

当然如果加密了，是找不到的。

- 搜 `下载` ： 搜不到。
- 搜 `未完成` = `uncomplete` ： 搜到一些

继续:

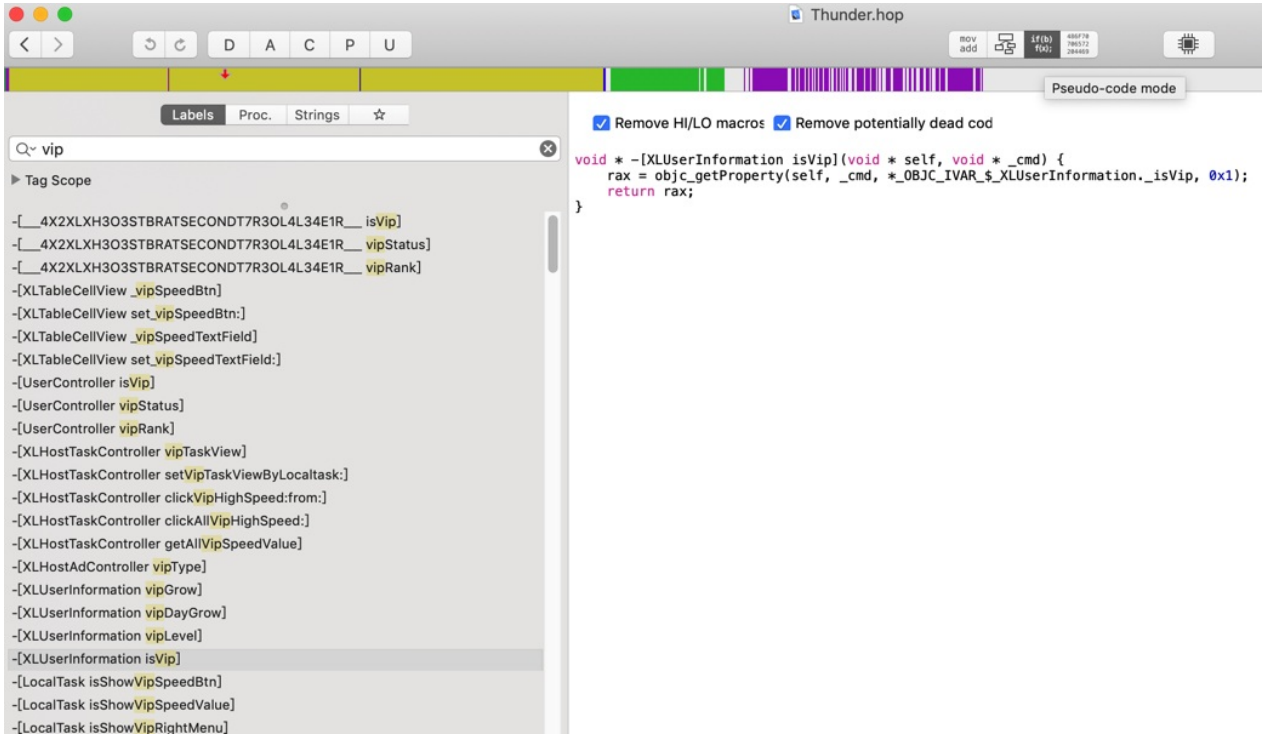
- 搜 `vip` ： 还真能搜到些内容

-

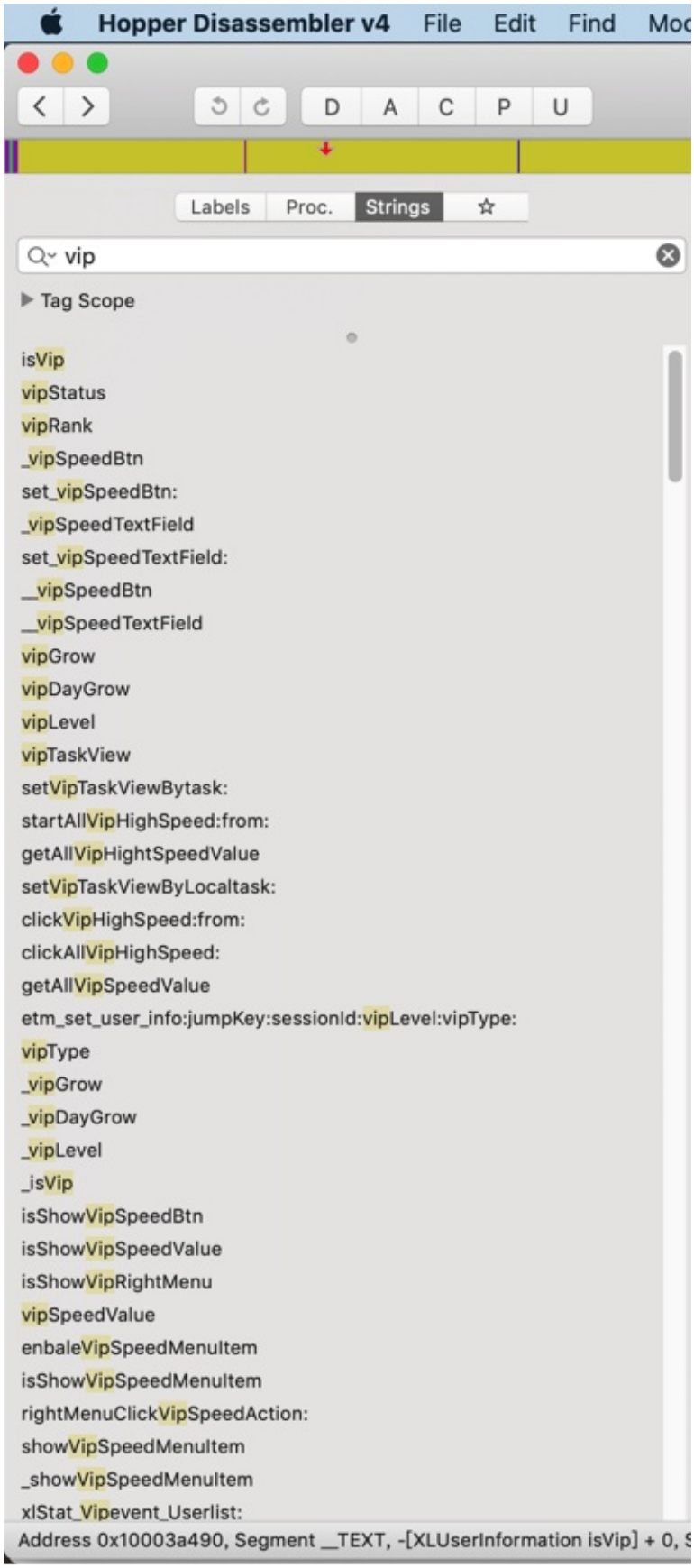
很明显有些是相关内容：

```
isVip
vipStatus
vipRank

UserController isVip
UserController vipStatus
UserController vipRank
```

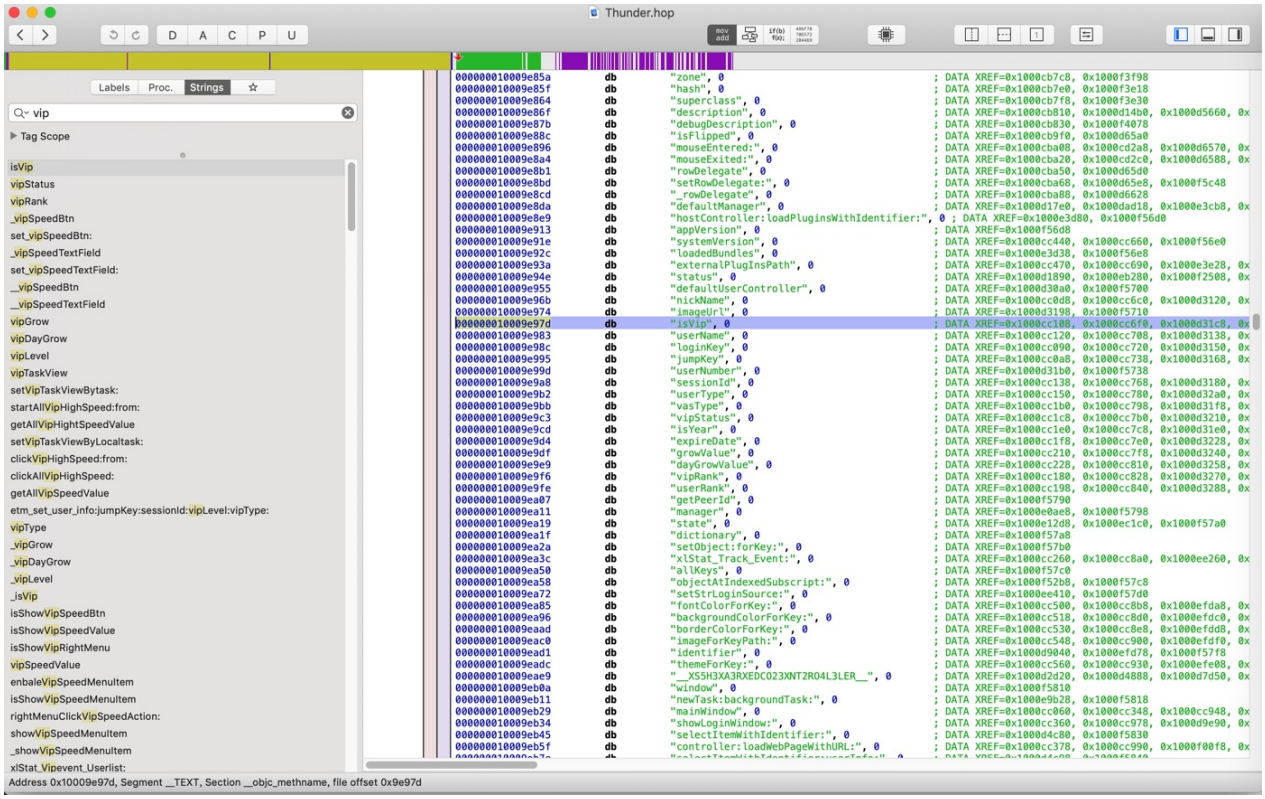
去看看：`伪代码` = `pseudo code`



再去找找String中是否有我们要的重要的内容



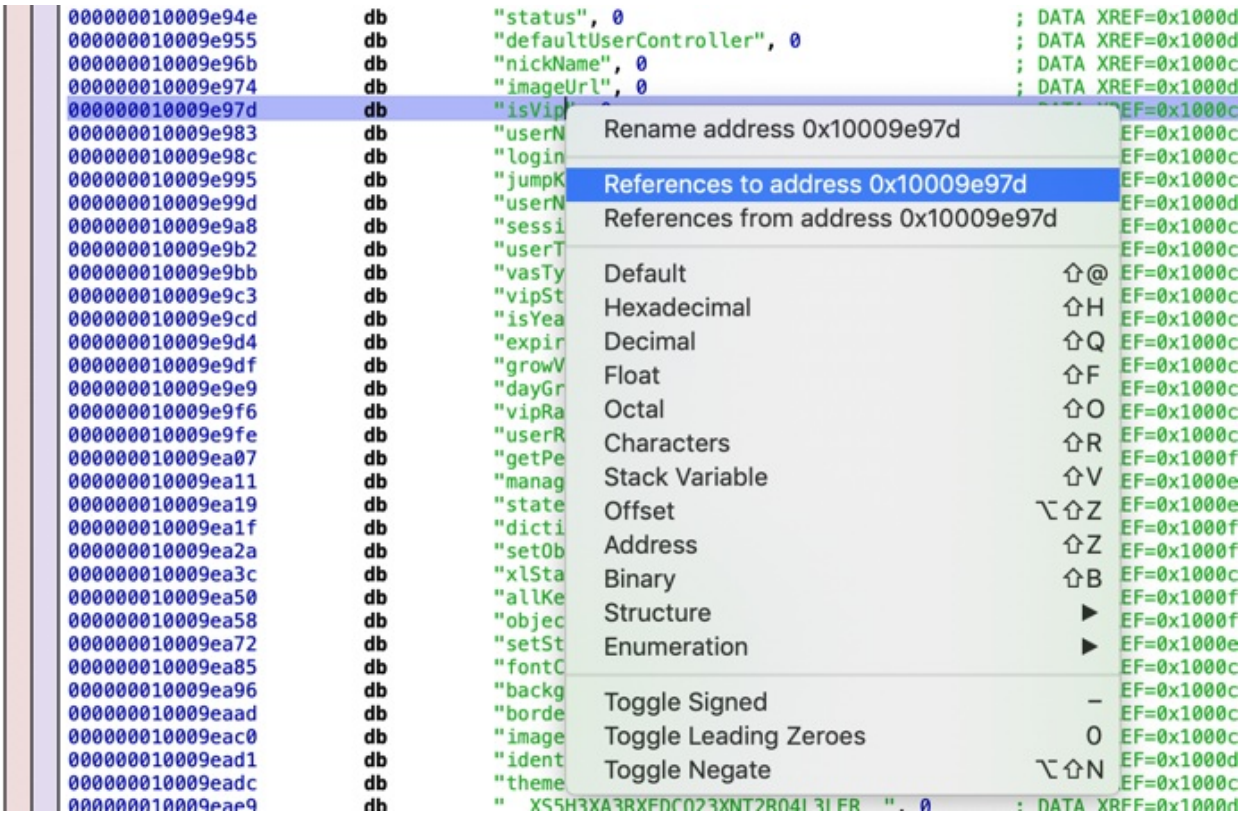
可以找到很多 `isVip` 相关的内容



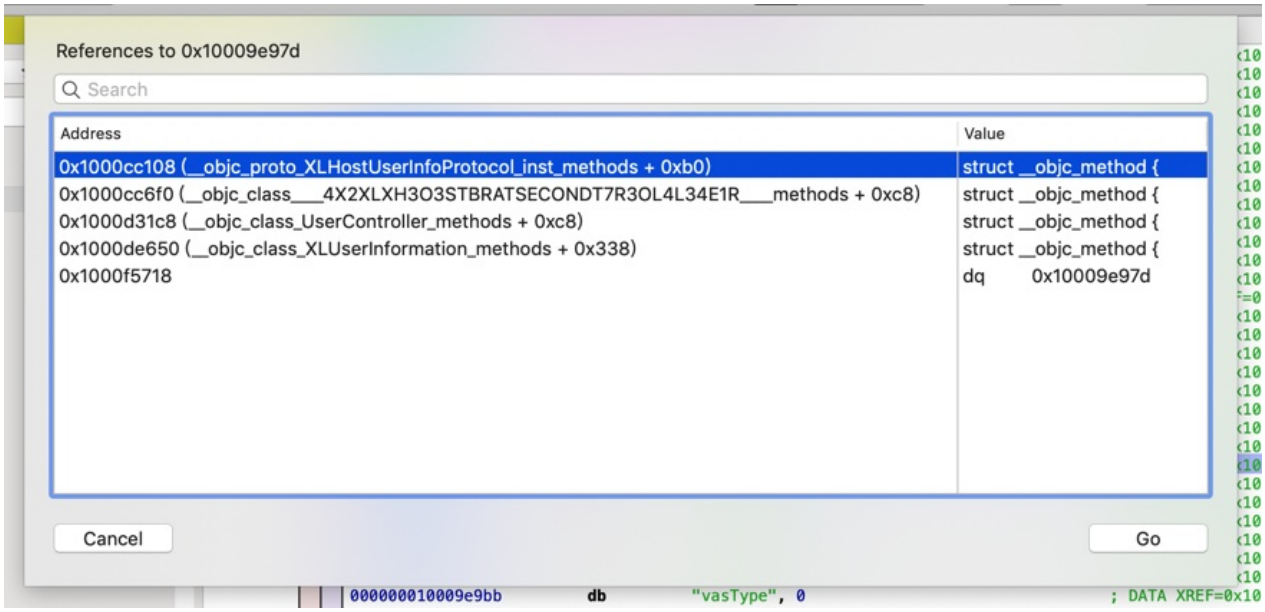
有点看起来是：和user用户相关的各种属性

```
000000010009e96b          db          "nickName", 0                                   ; DATA XREF 0x1000cc0d8, 0x1000cc6c0,
0x1000d3120, 0x1000de3b0, 0x1000f5708
000000010009e974          db          "imageUrl", 0                                   ; DATA XREF 0x1000d3198, 0x1000f5710
000000010009e983          db          "userName", 0                                   ; DATA XREF 0x1000cc120, 0x1000cc708,
0x1000d3138, 0x1000de380, 0x1000f5720
```

继续找被调用的地方：



找到了几处：



如此，根据需求，继续深入研究，即可慢慢分析出自己要找的逻辑。

Hopper心得

Hopper导出伪代码

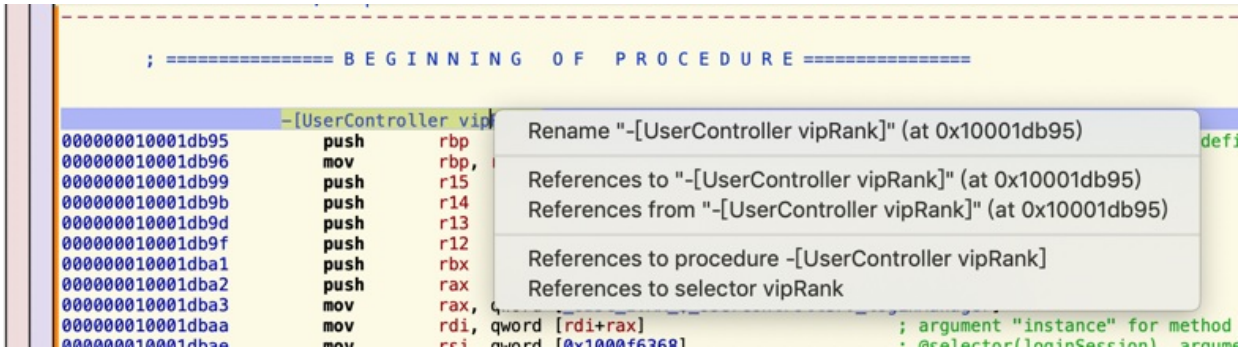
poboke/Class-Decompile: Class Decompile is a python script for Hopper Disassembler. This script can export pseudo code of the classes. (github.com)

据说可以导出Hopper的全部伪代码。有空去试试。

没有右键复制，直接快捷键复制

点击了函数，竟然没有直接双击选中并复制的功能

且右键也没有复制：



-》后来发现，其实没有双击和右键复制选择

-》当光标处于某行，直接Control+C就是复制整行内容

常见问题

卡死

Hopper打开太大的二进制，比如：

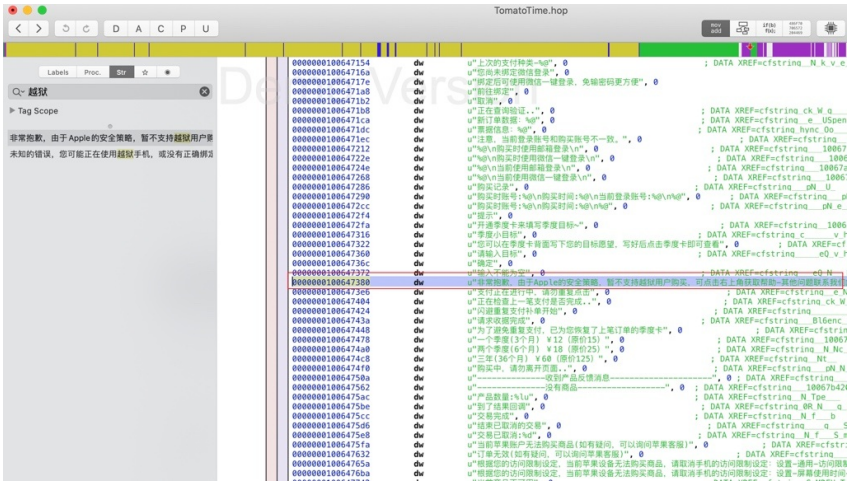
- YouTube的 `Module_Framework`
- Aweme的 `AwemeCore`

经常直接卡死，无法正常打开

-》导致无法正常使用。

Hopper vs IDA

- Hopper vs IDA
  - 概述
    - `IDA` 算**95分**，`Hopper` 算**70分**
      - IDA比Hopper强大很多
  - 详解
    - 总体对比
      - 功能对比
        - IDA：更强大
          - 伪代码：逻辑更清晰
        - Hopper：功能简洁，基本够用
          - 伪代码：代码逻辑不够清晰
      - IDA和Hopper类比
        - Hopper：小工具箱
        - 
        - IDA：各种专业工具的工作室
        - 
    - 分项对比
      - 平台支持
        - Hopper：更倾向于 `Mac`
        - IDA：支持多平台：`Windows` 、 `Linux` 、 `Mac`

- 功能支持
  - 总体上还是IDA更强大，Hopper相对较弱
  - 不过据说部分细节方面，有些Hopper支持更好?
    - 比如
      - 中文字符搜索
        - IDA 7.0+：不支持
        - Hopper：支持
        - 

- 价格
  - 正版IDA比Hopper贵**很多很多**
    - IDA Pro：**1000+**美元
    - Hopper：**~100**美元

Hopper vs IDA：反编译代码对比

某同一段代码的反汇编效果对比：

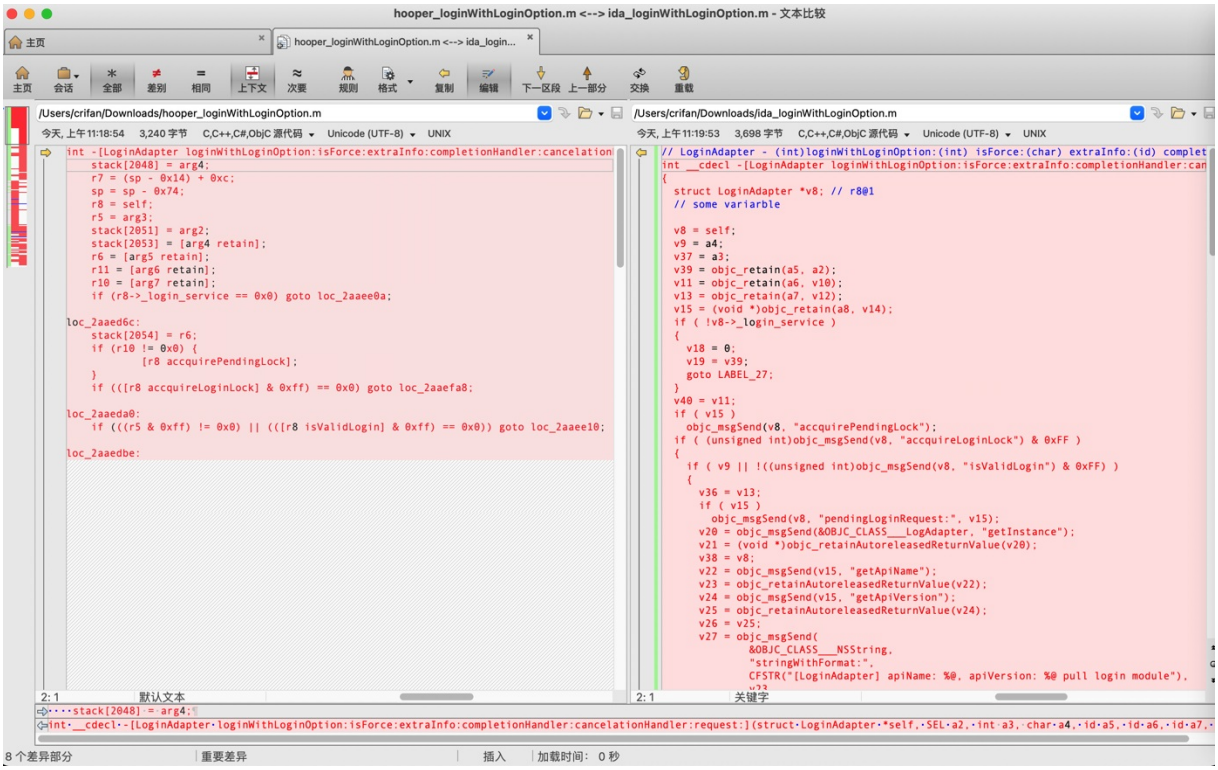
- IDA

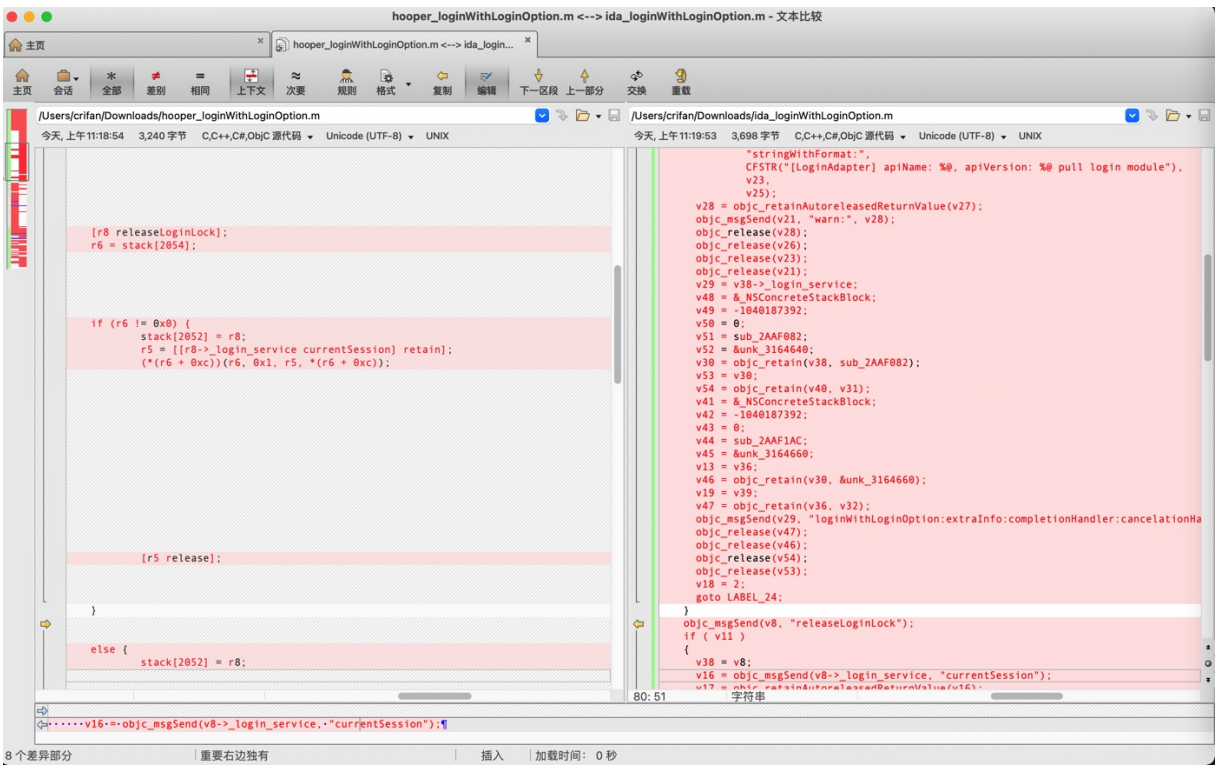
- Hopper

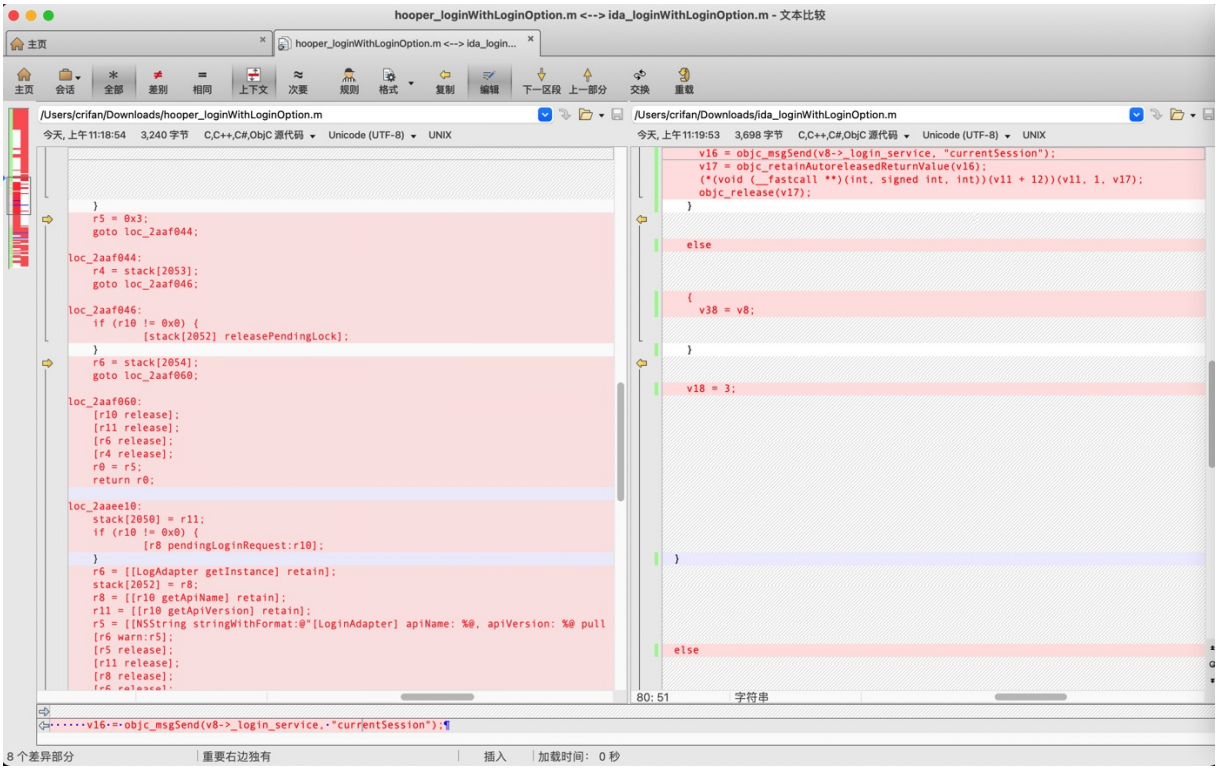
-

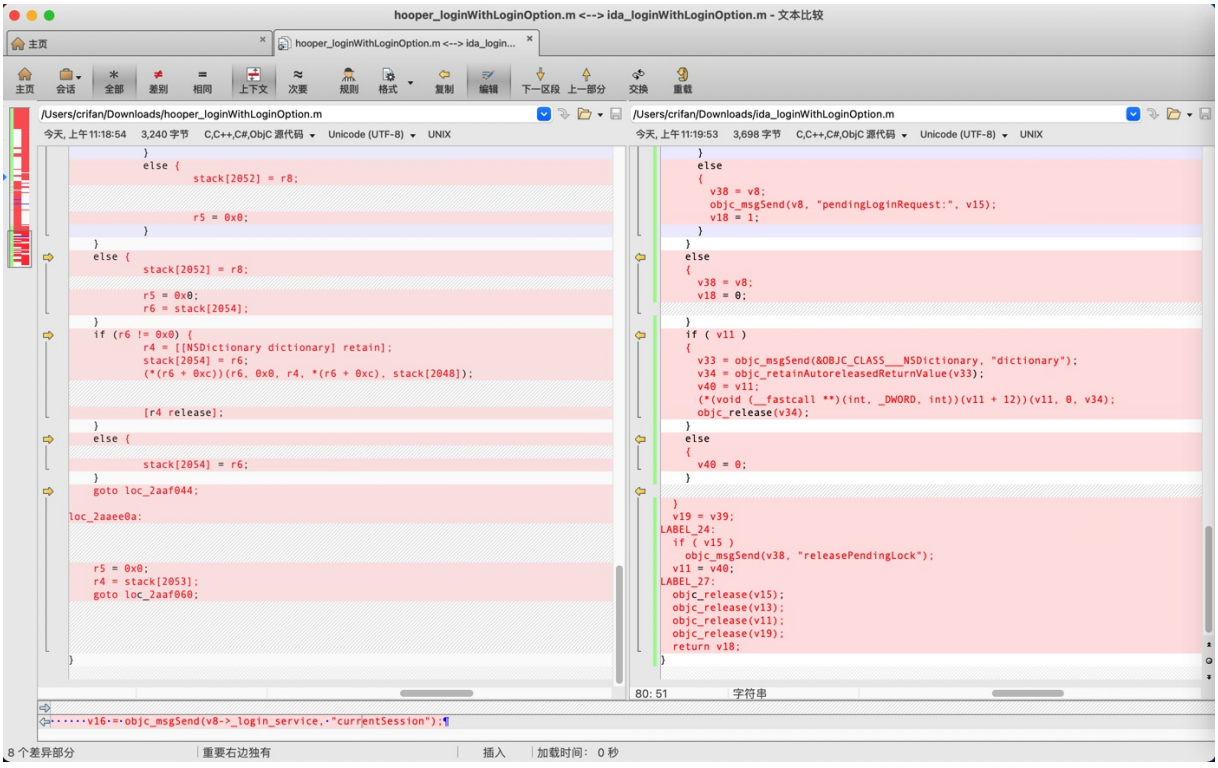
-

再用 `BeyondCompare` 详细对比：


hooper_loginWithLoginOption.m <--> ida_loginWithLoginOption.m - 文本比较
主页
hooper_loginWithLoginOption.m <--> ida_login...
/Users/crifan/Downloads/hooper_loginWithLoginOption.m
/Users/crifan/Downloads/ida_loginWithLoginOption.m
8 个差异部分
重要差异
插入
加载时间： 0 秒


hooper_loginWithLoginOption.m <--> ida_loginWithLoginOption.m - 文本比较
主页
hooper_loginWithLoginOption.m <--> ida_login...
/Users/crifan/Downloads/hooper_loginWithLoginOption.m
/Users/crifan/Downloads/ida_loginWithLoginOption.m
8 个差异部分
重要右边独有
插入
加载时间： 0 秒

- 

- 

结论：

明显能发现，反编译后的伪代码的效果：

IDA更好，Hooper不够好

再具体的说说细节：

- Hopper的ObjC函数调用的写法做了优化，比IDA更易读
  - IDA

```
v22 = objc_msgSend(v15, "getApiName");
v23 = objc_retainAutoreleasedReturnValue(v22);
```

- Hopper

```
r8 = [[r10 getApiName] retain];
```

- 但是Hopper的核心代码调用逻辑，没有IDA清楚
  - IDA

```
int __cdecl -[LoginAdapter loginWithLoginOption isForce extraInfo completionHandler cancelationHandler reque
st ](struct LoginAdapter *self, SEL a2, int a3, char a4, id a5, id a6, id a7, id a8)
...
  if ( !v8 || _login_service )
  {
    v18 = 0;
    v19 = v39;
    goto LABEL_27;
  }
...
LABEL_27:
  objc_release(v15);
  objc_release(v13);
  objc_release(v11);
  objc_release(v19);
  return v18;
}
```

  - Hopper

```
int -[LoginAdapter loginWithLoginOption isForce extraInfo completionHandler cancelationHandler request ](void
 * self, void * _cmd, int arg2, char arg3, void * arg4, void * arg5, void * arg6, void * arg7) {
...
loc_2aaf046:
    if (r10 != 0x0) {
            [stack[2052] releasePendingLock];
    }
    r6 = stack[2054];
    goto loc_2aaf060;

loc_2aaf060:
    [r10 release];
    [r11 release];
    [r6 release];
    [r4 release];
    r0 = r5;
    return r0;

...

loc_2aaee0a:
    r5 = 0x0;
    r4 = stack[2053];
    goto loc_2aaf060;
}
```

后记：

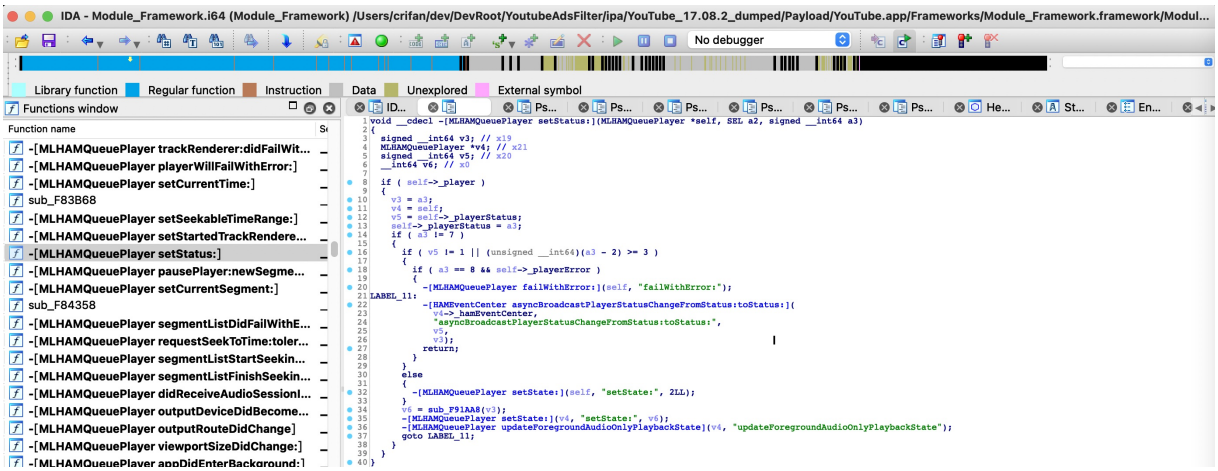
自己在给YouTube恢复符号表之后，发现：

IDA的函数调用，也已经自动为iOS的ObjC做了优化，比如：

- IDA

```
void __cdecl -[MLHAMQueuePlayer setStatus ](MLHAMQueuePlayer *self, SEL a2, signed __int64 a3)
{
  signed __int64 v3; // x19
  MLHAMQueuePlayer *v4; // x21
  signed __int64 v5; // x20
  __int64 v6; // x0
```

```
  if ( self->_player )
  {
    v3 = a3;
    v4 = self;
    v5 = self->_playerStatus;
    self->_playerStatus = a3;
    if ( a3 != 7 )
    {
      if ( v5 != 1 || (unsigned __int64)(a3 - 2) >= 3 )
      {
        if ( a3 == 8 && self->_playerError )
        {
          [MLHAMQueuePlayer failWithError:](self, "failWithError:");
LABEL_11
          [HAMEventCenter asyncBroadcastPlayerStatusChangeFromStatus:toStatus:](
            v4->_hamEventCenter,
            "asyncBroadcastPlayerStatusChangeFromStatus:toStatus:",
            v5,
            v3);
          return;
        }
      }
      else
      {
        [MLHAMQueuePlayer setState:](self, "setState:", 2LL);
      }
      v6 = sub_F91AA8(v3);
      [MLHAMQueuePlayer setState:](v4, "setState:", v6);
      [MLHAMQueuePlayer updateForegroundAudioOnlyPlaybackState](v4, "updateForegroundAudioOnlyPlaybackState");
      goto LABEL_11;
    }
  }
}
```

- 

其中的：

```
[MLHAMQueuePlayer failWithError:](self, "failWithError:");

[HAMEventCenter asyncBroadcastPlayerStatusChangeFromStatus:toStatus:](
          v4->_hamEventCenter,
          "asyncBroadcastPlayerStatusChangeFromStatus:toStatus:",
          v5,
          v3);
```

等代码中ObjC函数调用的写法，已经优化为我们希望的效果了：

`[ObjcClass function](para1, para2)`

即：

IDA中其实也已支持ObjC函数调用的优化的写法了。

附录

下面列出相关参考资料。

参考资料

- 【整理】iOS逆向工具：IDA和 Hopper 对比
- 【已解决】用Hopper去静态分析iOS版抖音
- 【未解决】Mac中用Hopper分析iOS抖音为二进制：找越狱相关内容
- 【已解决】Mac中下载和安装Hopper Disassembler
- 【未解决】Mac中安装和破解Hopper Disassembler v4.app
- 【未解决】Mac中安装Hopper Disassembler破解版
- 【已解决】Mac中用Hopper Disassembler打开迅雷Thunder
-
- iOS程序逆向Mac下常用工具——Reveal、HopperDisassemble、IDA - 时间已静止 - 博客园
- Hopper Alternatives and Similar Software - AlternativeTo.net
- IDA Pro反汇编工具初识及逆向工程解密实战 - 知乎 (zhihu.com)
- Hopper vs. IDA (wine) on OS X? : ReverseEngineering (reddit.com)
- Ghidra vs Cutter vs Radare2 vs IDA : LiveOverflow (reddit.com)
- A (completely unfair) comparison between radare2, IDA Pro and Hopper : ReverseEngineering (reddit.com)
- RE without IDA? : securityCTF (reddit.com)
- Hopper, an IDA-ish disassembler for OSX (it does Windows binaries!) : ReverseEngineering (reddit.com)
- Hopper VS IDA - compare differences & reviews? (saashub.com)
- tools - Is there any disassembler to rival IDA Pro? - Reverse Engineering Stack Exchange
- ARM binary dissasembly, Hopper works, Ghidra and Radare2 don't for some functions - Reverse Engineering Stack Exchange
- Reverse engineering and malware analysis tools - Infosec Resources (infosecinstitute.com)
- 今天开始学逆向：反汇编的利器 IDA 和 Hopper 的基本使用 - iOS开发 - 开发语言与工具 - 深度开源 (open-open.com)
- 今天开始学逆向：反汇编的利器 IDA 和 Hopper 的基本使用 (daimajiaoliu.com)
- 逆向工程 - Reveal、IDA、Hopper、HTTPS抓包 等 - 大河_大河 - 博客园 (cnblogs.com)
- Objective-C语言的逆向(Mac OS) - 知乎 (zhihu.com)
- 利用Hopper Disassembler和IDA Pro修改函数返回值_普通网友的博客-CSDN博客_ida修改函数返回值
- IDA Pro - 如何得到比较清楚的逆向伪代码 - 码上快乐 (codeprj.com)
- hopper逆向的伪代码令人大跌眼镜 - 码上快乐 (codeprj.com)
- iOS逆向之砸壳原理 - 掘金 (juejin.cn)
- iOS逆向学习（一）基础 | BenArvin's blog (benarvintec.com)
- TikTok(抖音国际版)逆向，全球的小姐姐们，我来啦！ - 尚码园 (shangmayuan.com)
- iOS SSL Certificate Pinning: Prevent Bypassing | Guardsquare
- iOS底层原理班（上）/APP逆向实战/加壳脱壳/数据安全/编译原理-学习视频教程-腾讯课堂 (qq.com)
- iOS 崩溃分析 - 掘金 (juejin.cn)
- ios(越狱) 应用脱壳反编译hook教程 (系统ios11.3.1) - 掘金 (juejin.cn)
- 十 iOS逆向- hopper disassembler - 简书 (jianshu.com)
- 飘云阁-PYG|软件安全|破解软件|内购破解|移动安全|chinapyg.com - Powered by Discuz!
- 飘云阁安全论坛的微博_微博
- [原创]破解Hopper Disassembler v3.7.8 for mac的艰难历程-『iOS安全』-看雪安全论坛
- iOS逆向指南：静态分析 | 黑超熊猫zuik's blog
- 最简单的Hopper Disassembler玩转Mac逆向 - 简书
- 逆向破解MacOS App - 简书
- Mac OSX 之自己动手初步学习破解软件入门 - 简书
- Hopper 系列教程之入门 CrackMe 分析 | ★tree_fly 's Blog
- 写给 iOS 开发者的 Hopper + lldb 简介 - OneAPM 博客
- 发现iOS SDK的Bug - Hopper使用教程向 | 小猪的博客
- iOS 13.1.3 Runtime Headers
- 动态调试及LLDB技巧集合_TuGeLe的博客-CSDN博客_lldb
- 【求问】Hopper每个方法的blocks和size怎么算的？ - 技能讨论 - 睿论坛 (iosre.com)

- ios逆向工具Hopper Disassembler的基本使用功能整理(持续更新)_小手琴师的博客-CSDN博客
- iOS逆向之Reveal、Hopper、MachOView等逆向工具的安装使用 - 简书 (jianshu.com)
- 3.8 Hopper: 另一款反汇编工具 | iOS 安全 Wiki (gitbooks.io)
- 十 iOS逆向- hopper disassembler - 简书 (jianshu.com)
- iOS小技能：逆向工具hopper的使用 - 掘金 (juejin.cn)
-