目录

iOS逆向开发：静态分析

- 最新版本：`v0.9`
- 更新时间：`20221108`

简介

介绍iOS逆向开发期间的静态分析。先是静态分析的概览；接着是从ipa中找到要分析的二进制文件；再是静态分析主要涉及的内容，包括用MachOView、jtool2、rabin2等查看Mach-O二进制信息，用strings、nm、otool等导出字符串资源，用class-dump导出ObjC头文件，用IDA、Hopper等分析代码逻辑等等；然后给出一些相关实例；最后再整理一些相关的经验心得。

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_static_analysis: iOS逆向开发：静态分析

如何使用此HonKit源码去生成发布为电子书

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

在线浏览

- iOS逆向开发：静态分析 book.crifan.org
- iOS逆向开发：静态分析 crifan.github.io

离线下载阅读

- iOS逆向开发：静态分析 PDF
- iOS逆向开发：静态分析 ePub
- iOS逆向开发：静态分析 Mobi

版权和用途说明

鸣谢

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

其他

作者的其他电子书

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

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

关于作者

关于作者更多介绍，详见：

关于CrifanLi李茂 – 在路上

静态分析概览

iOS逆向过程中，在砸壳出ipa后，就是：`静态分析`。

什么是静态分析

iOS逆向的目的，往往要搞清楚，app底层的某些功能和逻辑的具体实现机制和原理。

而搞懂底层逻辑，从过程角度来说，主要分：

- 静态分析：不运行程序的前提下，利用工具和手段，搞懂程序逻辑
- 动态调试：运行程序的前提下，动态运行期间，研究和调试程序的逻辑

从ipa中找到二进制

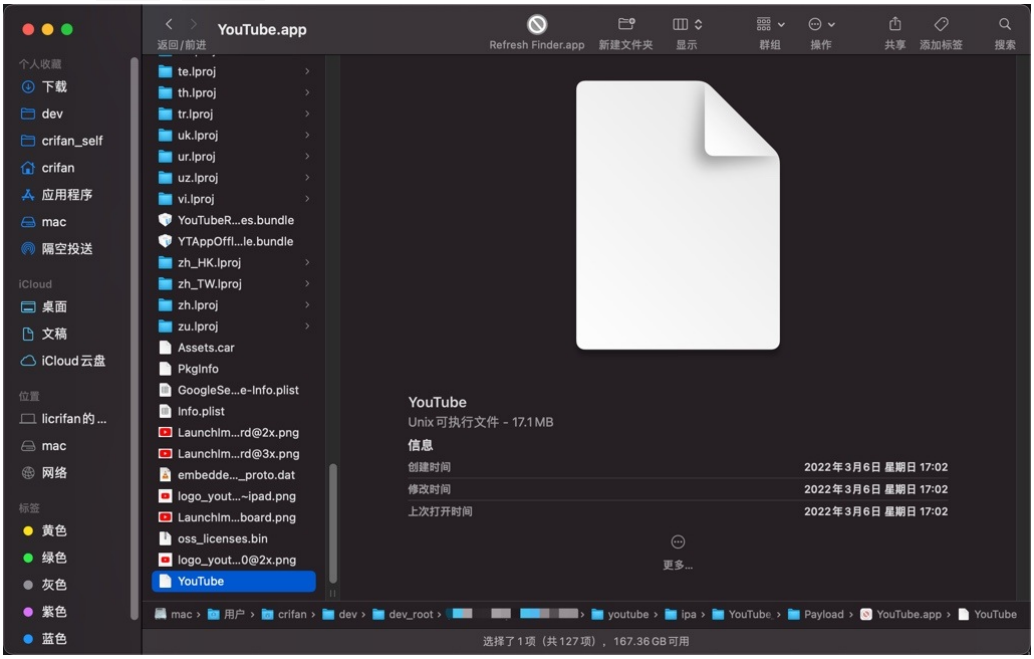
TODO：

- 【未解决】静态分析抖音二进制寻找越狱检测手段
- 【已解决】如何从脱壳后的抖音的IPA文件中得到二进制文件
- 相关
  - 【已解决】越狱iPhone中抖音app的安装目录安装位置
  - 【已解决】研究iOS中app的目录的UUID类的值和app名称如何映射

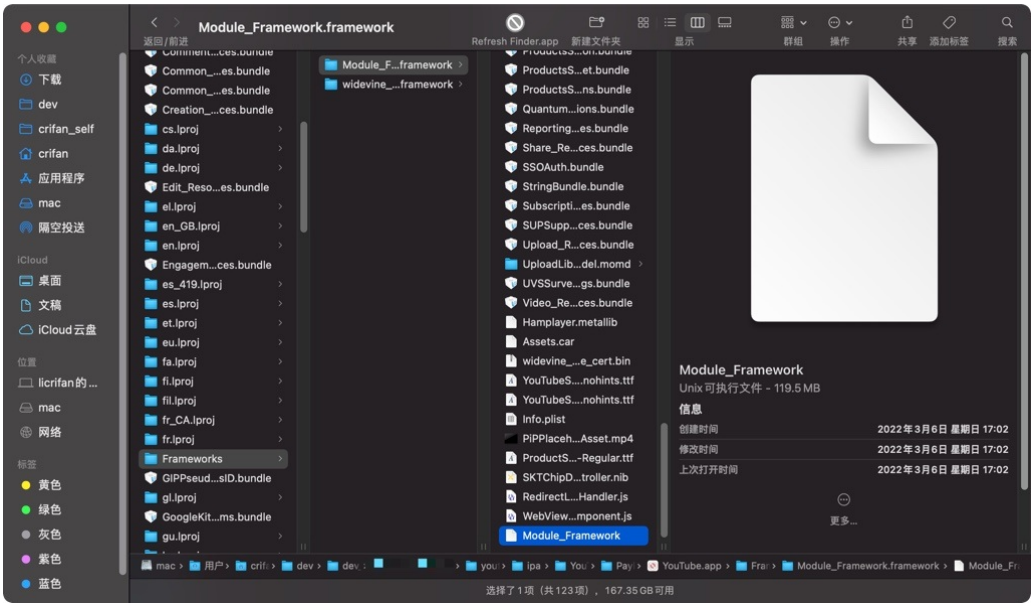
作为iOS逆向的静态分析，其输入文件是iOS的app的二进制文件。

对应的就是，在前一步，从app砸壳得到的ipa文件中，找到对应的二进制文件，用于后续的静态分析。

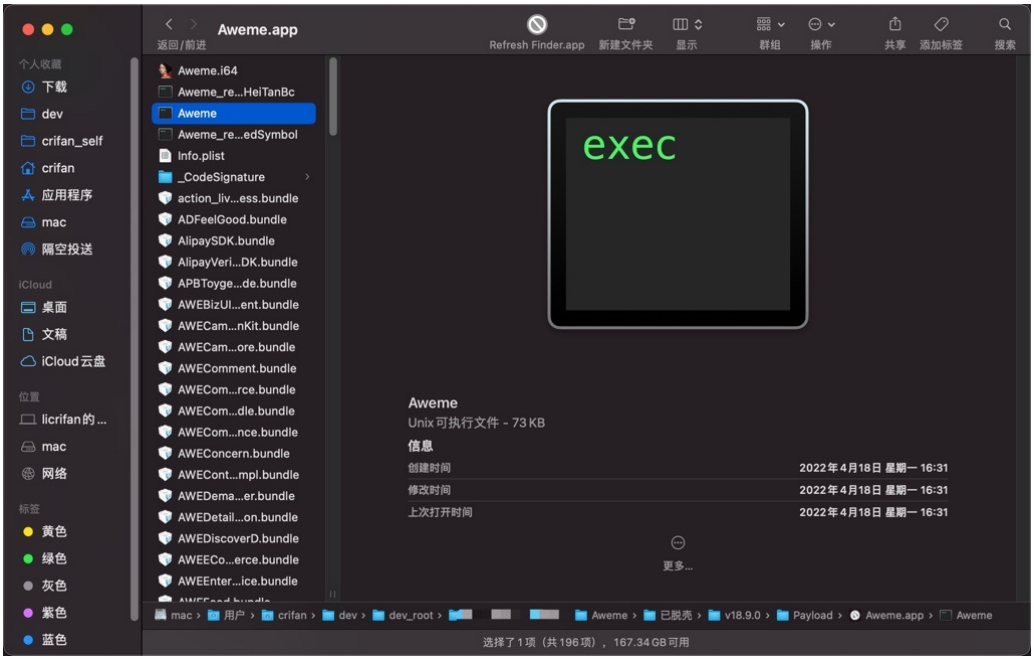
比如：

- YouTube
  - v17.08.2
    - ipa解压后得到：`YouTube.app`
      - 入口二进制：`17MB+` 的 `YouTube`
      - 

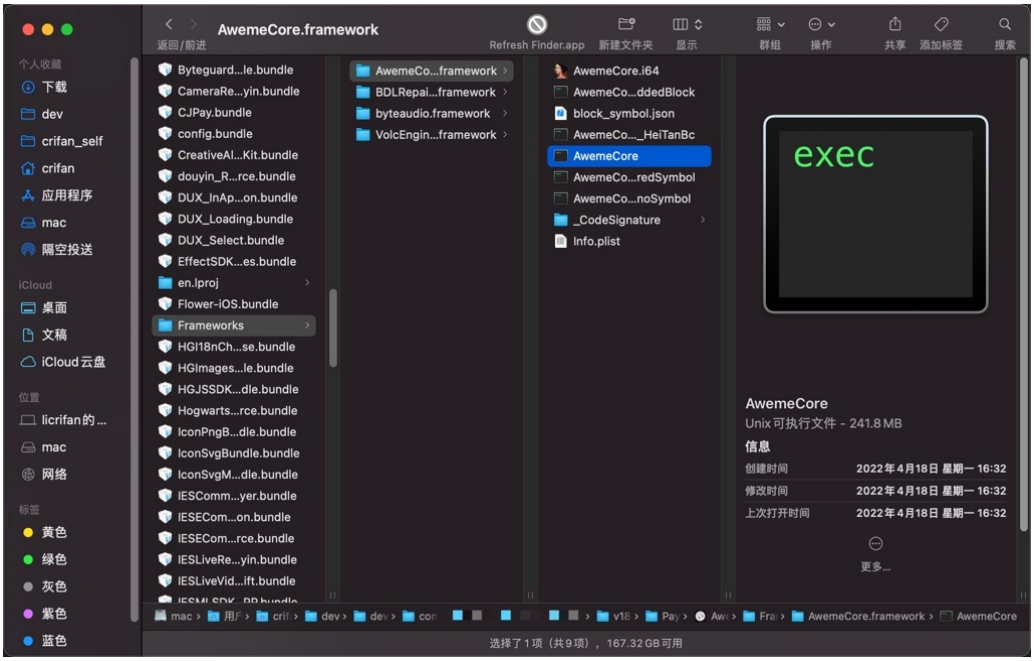
      - 核心二进制：`100MB+` 的 `Frameworks/Module_Framework.framework/Module_Framework`



- 抖音
  - v18.9.0
    - ipa解压后得到：`Aweme.app`
      - 入口二进制：`70KB+` 的 `Aweme`



      - 核心二进制：`240MB+` 的 `Frameworks/AwemeCore.framework/AwemeCore`

- 

静态分析内容

静态分析 的 `相关内容` = `涉及内容` = `常用手段` ：

- 查看二进制信息
  - `Mach-O`
    - `MachOView`
    - `jtool2`
    - `rabin2`
- 导出字符串等资源：用于后续的搜索函数、类等用途
  - `strings`
  - `nm`
  - `otool`
- 导出头文件：研究类的具体函数和属性
  - `class-dump`
- 分析代码逻辑：分析研究代码实现逻辑和其他各种细节
  - `IDA`
  - `Hopper`

查看二进制信息

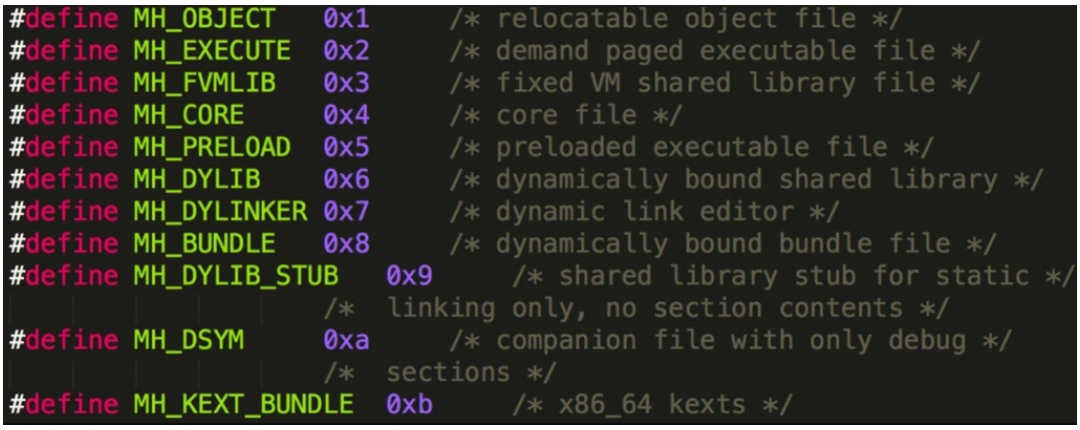
iOS逆向中的静态分析，涉及到，查看iOS的二进制文件的信息。

而iOS的二进制格式是 `Mach-O` ，所以也就是用工具查看 `Mach-O` 的相关信息。

Mach-O

TODO：

【记录】静态分析Mask的动态库：MaskPro.dylib

- `Mach-O`
  - 名称来源：`Mach Object`的缩写
  - 是什么：`Apple`的`Mac`、`iOS`等平台的二进制程序、库等`可执行文件`的底层文件数据格式
  - 常见文件类型
    - 图
      - 

    - 文字
      - `MH_EXECUTE`=可执行文件=`executable`=应用
        - 文件：`.app/xxx`
      - `MH_OBJECT`
        - 目标文件
          - 文件：`.o`
        - 静态库文件=静态链接库=`static library`：N个`.o`合并在一起
          - 文件：`.a`
      - `MH_DYLIB`=动态链接库=`dylib library`：类似于Win中的`DLL`
        - 文件：`.dylib`、`.framework/xxx`
      - `MH_DYLINKER`：动态链接编辑器
        - 文件：`/usr/lib/dyld`
      - `MH_DSYM`：存储着二进制文件符号信息的文件
        - 文件：`.dSYM/Contents/Resources/DWARF/xxx`
          - 常用于分析APP的崩溃信息
  - 基本结构
    - `Header`：文件类型、目标架构类型等
    - `Load commands`：描述文件在虚拟内存中的逻辑结构、布局
    - `Raw segment data`：在Load commands中定义的Segment的原始数据
  - 详细定义
    - 详见：xnu源码
      - https://opensource.apple.com/tarballs/xnu/
        - `EXTERNAL_HEADERS/mach-o/fat.h`
        - `EXTERNAL_HEADERS/mach-o/loader.h`
  - 相关工具：查看和处理Mach-O格式的文件
    - 查看信息
      - `file`：查看Mach-O的文件类型

        - `file inputMacOFile`
    - `MachOView`
    - `jtool2` / `jtool`
    - `otool`：查看Mach-O特定部分和段的内容
    - `lipo`：常用于多架构Mach-O文件的处理
        - 查看架构信息：`lipo -info inputMacOFile`
        - 导出某种特定架构：`lipo inputMachOFile -thin ArchType -output OutputFile`
        - 合并多种架构：`lipo inputMachOFile1 inputMachOFile2 -output OutputFile`
    - `rabin2`
- 逆向处理
    - 导出头文件
        - `class-dump`

`FAT Binary` = 胖二进制

- `Fat Binary` = 胖二进制 = `Fat File` = 胖二进制文件 = `Universal Binary` = 通用二进制文件
    - 把多个架构的二进制（比如 `armv7` 、 `arm64` 等）合并在一起，成了个胖子，所以叫 `Fat Binary`
        - 一个由不同的编译架构后的 `Mach-O` 产物所合成的集合体
        - 一个架构的 `Mach-O` 只能在相同架构的机器或者模拟器上用
        - 为了支持不同架构需要一个集合体
    - 文件大小
        - 一般比单一架构的文件要大
        - 但是由于多架构会共用一部分资源，所以往往比多个（常常是2个）的总大小要小

胖二进制的实例

折腾Mask的dylib期间就遇到了 `FAT Binary`：

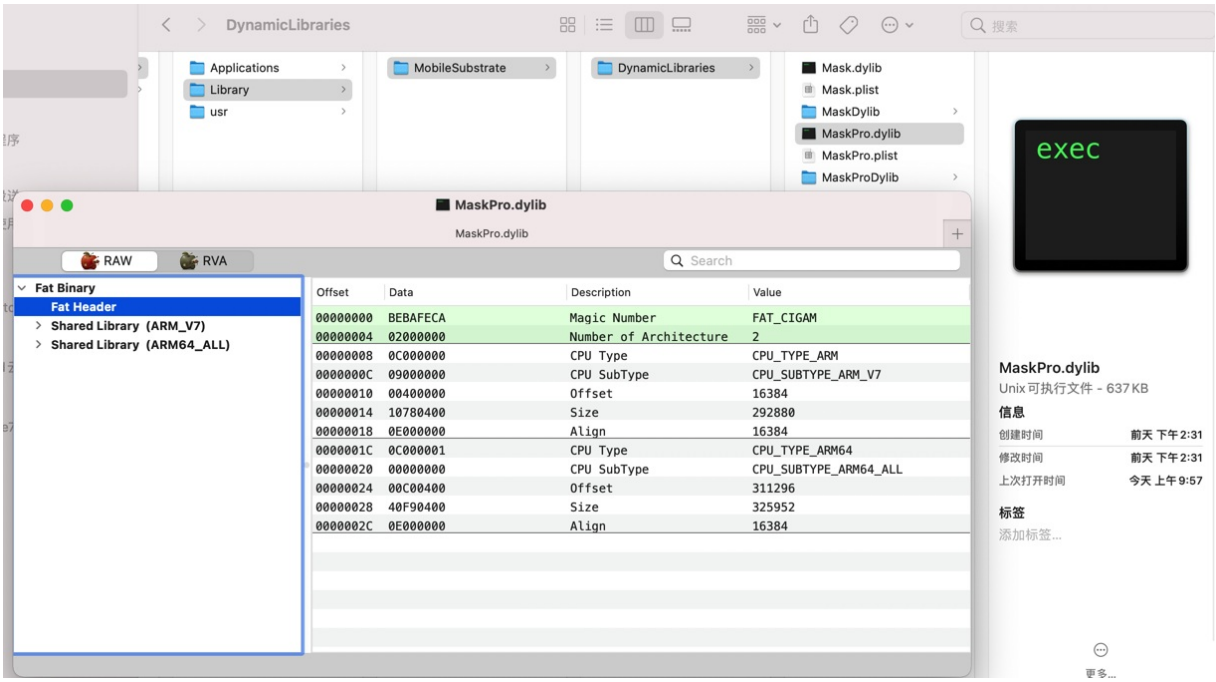
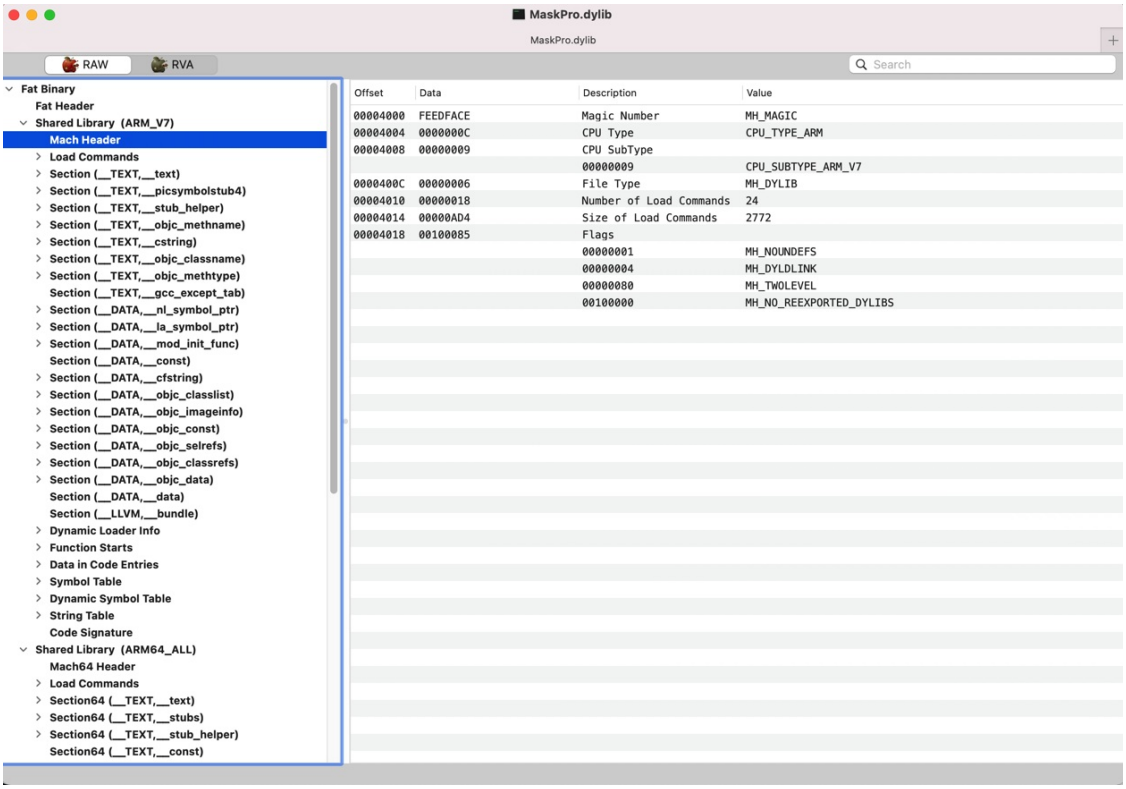
```
➜  DynamicLibraries jtool2 -h MaskPro.dylib > MaskProDylib/MaskProDylib_jtool2_h_header.txt
Fat binary, little-endian, 2 architectures: armv7, arm64
Select an architecture setting the ARCH environment variable
```

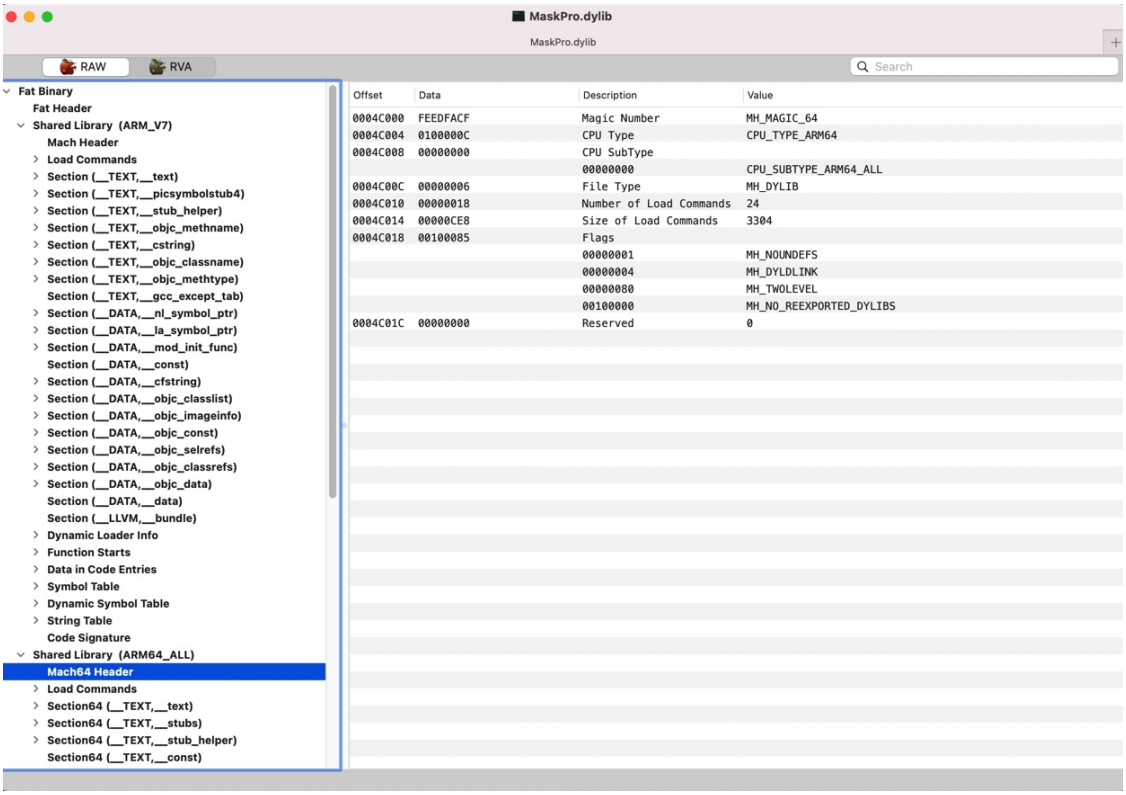
即，一个Dylib中，包含了多种架构，此处是 `armv7` 和 `arm64`

此处要指定具体架构，才能继续用 `jtool2` 查看信息：

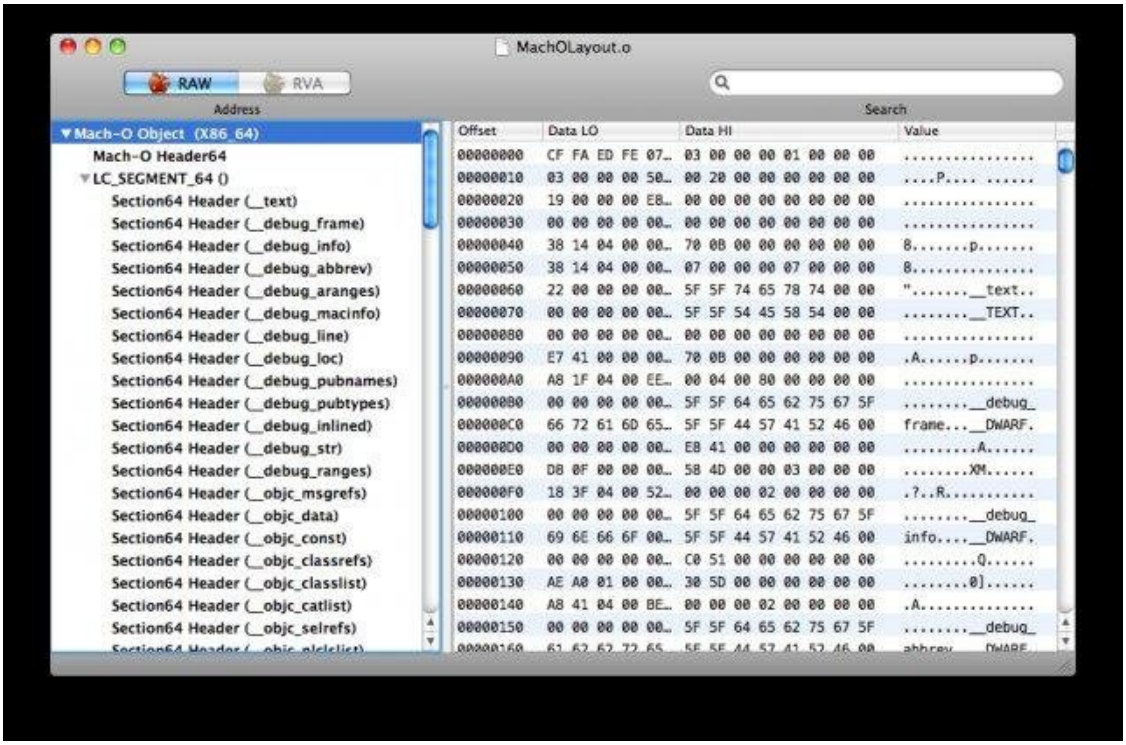
```
➜  DynamicLibraries export ARCH=arm64
➜  DynamicLibraries jtool2 -h MaskPro.dylib > MaskProDylib/MaskProDylib_jtool2_h_header.txt
```

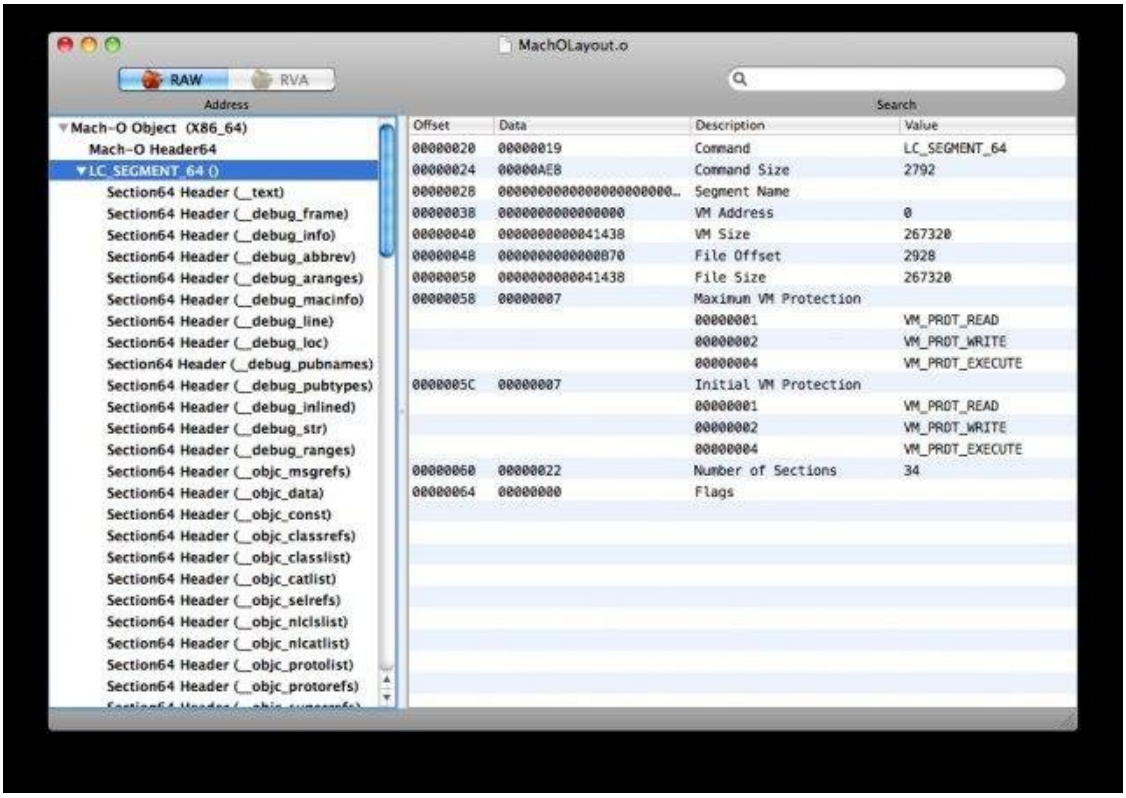
类似的，后续去用 `MachOView` 查看信息，也能看到是：`FAT Binary`

- 
DynamicLibraries
Applications
Library
usr
MobileSubstrate
DynamicLibraries
Mask.dylib
Mask.plist
MaskDylib
MaskPro.dylib
MaskPro.plist
MaskProDylib
exec
MaskPro.dylib
MaskPro.dylib
RAW
RVA
Search
Fat Binary
Fat Header
Shared Library (ARM_V7)
Shared Library (ARM64_ALL)
Offset Data Description Value
00000000 BEBAFECA Magic Number FAT_CIGAM
00000004 02000000 Number of Architecture 2
00000008 0C000000 CPU Type CPU_TYPE_ARM
0000000C 09000000 CPU SubType CPU_SUBTYPE_ARM_V7
00000010 00400000 Offset 16384
00000014 10780400 Size 292880
00000018 0E000000 Align 16384
0000001C 0C000001 CPU Type CPU_TYPE_ARM64
00000020 00000000 CPU SubType CPU_SUBTYPE_ARM64_ALL
00000024 00C00400 Offset 311296
00000028 40F90400 Size 325952
0000002C 0E000000 Align 16384
MaskPro.dylib
Unix 可执行文件 - 637 KB
信息
创建时间 前天 下午2:31
修改时间 前天 下午2:31
上次打开时间 今天 上午9:57
标签
添加标签...
更多...
  - ARMV7
  - 
MaskPro.dylib
MaskPro.dylib
RAW
RVA
Search
Fat Binary
Fat Header
Shared Library (ARM_V7)
Mach Header
Load Commands
Section (__TEXT,__text)
Section (__TEXT,__picsymbolstub4)
Section (__TEXT,__stub_helper)
Section (__TEXT,__objc_methname)
Section (__TEXT,__cstring)
Section (__TEXT,__objc_classname)
Section (__TEXT,__objc_methtype)
Section (__TEXT,__gcc_except_tab)
Section (__DATA,__nl_symbol_ptr)
Section (__DATA,__la_symbol_ptr)
Section (__DATA,__mod_init_func)
Section (__DATA,__const)
Section (__DATA,__cfstring)
Section (__DATA,__objc_classlist)
Section (__DATA,__objc_imageinfo)
Section (__DATA,__objc_const)
Section (__DATA,__objc_selrefs)
Section (__DATA,__objc_classrefs)
Section (__DATA,__objc_data)
Section (__DATA,__data)
Section (__LLVM,__bundle)
Dynamic Loader Info
Function Starts
Data in Code Entries
Symbol Table
Dynamic Symbol Table
String Table
Code Signature
Shared Library (ARM64_ALL)
Mach64 Header
Load Commands
Section64 (__TEXT,__text)
Section64 (__TEXT,__stubs)
Section64 (__TEXT,__stub_helper)
Section64 (__TEXT,__const)
Offset Data Description Value
00004000 FEEDFACE Magic Number MH_MAGIC
00004004 0000000C CPU Type CPU_TYPE_ARM
00004008 00000009 CPU SubType
00000009 CPU_SUBTYPE_ARM_V7
0000400C 00000006 File Type MH_DYLIB
00004010 00000018 Number of Load Commands 24
00004014 00000AD4 Size of Load Commands 2772
00004018 00100085 Flags
00000001 MH_NOUNDEFS
00000004 MH_DYLDLINK
00000080 MH_TWOLEVEL
00100000 MH_NO_REEXPORTED_DYLIBS
  - ARM64



MachOView

- `MachOView`
  - 是什么：查看和编辑Intel的 `x86` 和 `ARM` 的 `Mach-O` 二进制文件的工具
  - 用途：常用来查看iOS的app的二进制文件的信息
  - 截图
    - 

    - 

  - 资料
    - 最早好像是在sourceforge
      - MachOView download | SourceForge.net

        - https://sourceforge.net/projects/machoview/
    - 后来有人fork到GitHub
        - gdbinit/MachOView: MachOView fork
            - https://github.com/gdbinit/MachOView
    - 现在有国人fork后继续维护
        - fangshufeng/MachOView: 分析Macho必备工具
            - https://github.com/fangshufeng/MachOView

一些心得

- 如果二进制太大，或者本身防护做的比较好，则：MachOView完全加载出来二进制信息
    - 往往耗时很久
    - 也往往会直接崩溃，无法继续使用
        - 比如抖音的二进制加载到最后，就被崩溃。
            - 只能在崩溃之前，及时查看（大）部分已解析出的信息

jtool2

- `jtool`
  - 新版叫：`jtool2`
  - 类似于 `otool` 的，解析查看 `Mach-O` 文件格式信息
    - 区别：添加了许多Mach-O相关的命令
      - jtool比otool功能更完善
  - 支持多种运行平台
    - `OS X` = `Mac`
    - `iOS`
    - `Linux`
  - 功能
    - in-binary search functionality
    - symbol injection
    - built-in disassembler functionality with (limited but constantly improving) emulation capabilities, which already outdo fancy commercial GUI disassemblers.
    - Color terminal output, enabled by JCOLOR=1
  - 资料
    - 官网
      - JTool2 - Taking the O out of otool - squared
        - http://www.newosxbook.com/tools/jtool.html

rabin2

TODO：

- 【记录】静态分析Mask的动态库：MaskPro.dylib
- 【记录】静态分析Mask的动态库：Mask.dylib
- 【记录】用radare2查看抖音二进制信息
- 【记录】用rabin2查看抖音AwemeCore二进制的信息

导出字符串资源

TODO：

- 【未解决】静态分析符号函数字符串：Aweme
- 【未解决】静态分析符号函数字符串：AwemeCore

---

iOS逆向期间，静态分析，往往涉及到：从二进制中导出字符串等信息和其他资源，供后续研究很分析。

常用的导出字符串等资源的工具有：

- `strings`
- `nm`
- `otool`
- `jtool2`
- `rabin2`

strings

TODO：

- 【记录】用strings查看dylib库中包含的字符串

---

nm

otool

TODO：

- 【记录】用otool查看分析二进制和库文件信息
- 【记录】用otool去分析抖音二进制AwemeCore
- 【未解决】iOS中从otool输出的B16@0:8搞懂函数类型和参数定义

- `otool`
  - = `object file displaying tool`
  - 是什么：针对目标文件的展示工具
  - 用途：用来发现应用中使用到了哪些系统库，调用了其中哪些方法，使用了库中哪些对象及属性
    - 比如
      - 查看iOS的 `Mach-O` 格式的二进制文件的信息
  - 来源：Xcode自带的常用工具
- 相关
  - 比otool更好的： `jtool`
  - `otool` 的 `GUI` 版： `otx`
    - x43x61x69/otx: The Mach-O disassembler. Now 64bit and Xcode 6 compatible.
      - https://github.com/x43x61x69/otx

查看当前otool位置：

```
✘ crifan@licrifandeMacBook-Pro  ~    which otool
/usr/bin/otool
```

当前版本：

```
✘ crifan@licrifandeMacBook-Pro  ~  otool --version
llvm-otool(1): Apple Inc. version cctools-927.0.2
Apple LLVM version 10.0.1 (clang-1001.0.46.4)
  Optimized build.
  Default target: x86_64-apple-darwin19.2.0
  Host CPU: broadwell

  Registered Targets:
    aarch64    - AArch64 (little endian)
    aarch64_be - AArch64 (big endian)
    arm        - ARM
    arm64      - ARM64 (little endian)
    armeb      - ARM (big endian)
    thumb      - Thumb
    thumbeb    - Thumb (big endian)
    x86        - 32-bit X86: Pentium-Pro and above
    x86-64     - 64-bit X86: EM64T and AMD64
```

help帮助语法

```
✘ crifan@licrifandeMacBook-Pro  ~  otool -help
error: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/otool: unknown char `p' in flag -
help

Usage: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/otool [-arch arch_type] [-fahlLDt
dorSTMRIHGvVcXmqQjCP] [-mcpu=arg] [--version] <object file> ...
    -f print the fat headers
    -a print the archive header
    -h print the mach header
    -l print the load commands
```

```
-L print shared libraries used
-D print shared library id name
-t print the text section (disassemble with -v)
-p <routine name>  start dissassemble from routine name
-s <segname> <sectname> print contents of section
-d print the data section
-o print the Objective-C segment
-r print the relocation entries
-S print the table of contents of a library (obsolete)
-T print the table of contents of a dynamic shared library (obsolete)
-M print the module table of a dynamic shared library (obsolete)
-R print the reference table of a dynamic shared library (obsolete)
-I print the indirect symbol table
-H print the two-level hints table (obsolete)
-G print the data in code table
-v print verbosely (symbolically) when possible
-V print disassembled operands symbolically
-c print argument strings of a core file
-X print no leading addresses or headers
-m don't use archive(member) syntax
-B force Thumb disassembly (ARM objects only)
-q use llvm's disassembler (the default)
-Q use otool(1)'s disassembler
-mcpu=arg use `arg' as the cpu for disassembly
-j print opcode bytes
-P print the info plist section as strings
-C print linker optimization hints
--version print the version of /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/otool
```

导出头文件

iOS逆向的静态分析中，往往涉及到：从二进制文件中，导出ObjC的类的头文件，简称：导出头文件。

- 导出头文件的前提
  - iOS的app的二进制是已经砸壳后的 = 解密后的
  - 待确认：iOS的app的代码是ObjC的

一些经验和心得

- 关于其中：待确认：iOS的app的代码是ObjC的
  - ObjC由于底层机制（Runtime运行时的原因？），使得导出ObjC的类的头文件，成为可能
    - 而iOS的新的编程语言Swift，从底层机制上说，就无法导出Swift的类的头文件
  - 从而使得=导致：
    - 对于iOS的app的实现是ObjC和Swift混合都有的话，之前旧版本class-dump在导出时会报错，导致无法导出头文件
      - 需要找到新版，支持ObjC和Swift混合代码，才能继续导出（ObjC的类的）头文件
    - 从iOS正向防护角度，增加逆向导出头文件的难度：尽量把iOS的代码从ObjC改为Swift
      - 待确认：即可避免头文件被导出

class-dump

TODO：

- 【基本解决】砸壳抖音ipa后导出iOS抖音头文件

- class-dump
  - 一句话描述：iOS逆向中导出ObjC的头文件的常用工具
    - 用于处理 `Objective-C` 的 `Mach-O` 文件信息的命令行工具，可以导出类的定义、分组和协议。
      - command-line utility for examining the Objective-C segment of Mach-O files
  - 说明
    - 和 `otool -ov` 导出的信息是一样的
      - 但是显示为 `Objective-C` 定义，更易读
  - 原理
    - 利用了 `Objective-C` 语言的运行时的特性
      - 将存储在 `Mach-O` 文件中的头文件信息提取出来，并生成对应的 `.h` 文件
  - 用途
    - 查看闭源的 `应用` 、 `frameworks` 、 `bundles`
      - 查看其中的头文件信息
    - 对比一个 APP 不同版本之间的接口变化
      - 通过导出不同版本的库的头文件的对比看出来
    - 对一些私有 `frameworks` 做些有趣的试验
  - 资料
    - GitHub
      - nygard/class-dump: Generate Objective-C headers from Mach-O files.
        - https://github.com/nygard/class-dump
    - 官网
      - class-dump - Steve Nygard
        - http://stevenygard.com/projects/class-dump/

下载

- class-dump-3.5.dmg
- class-dump-3.5.tar.gz
- class-dump-3.5.tar.bz2

用法举例

- class-dump AppKit
  - `class-dump /System/Library/Frameworks/AppKit.framework`
- class-dump UIKit
  - `class-dump /Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS4.3.sdk/System/Library/Frameworks/UIKit.framework`
- class-dump UIKit and all the frameworks it uses
  - `class-dump /Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS4.3.sdk/System/Library/Frameworks/UIKit.framework -r --sdk-ios 4.3`
- class-dump UIKit (and all the frameworks it uses) from developer tools that have been installed in /Dev42 instead of /Developer

- `class-dump /Dev42/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS5.0.sdk/System/Library/Frameworks/UIKit.framework -r --sdk-root /Dev42/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS5.0.sdk`

实际使用举例

之前从WebDriverAgent的源码中看到很多头文件的头部都有：`Generated by class-dump`

举例：

`refer/WebDriverAgent/PrivateHeaders/XCTest/XCTestDriver.h`

```
//
//     Generated by class-dump 3.5 (64 bit).
//
//     class-dump is Copyright (C) 1997-1998, 2000-2001, 2004-2013 by Steve Nygard.
//
```

-》说明这些文件都是通过 `class-dump` 从库文件中导出生成的。

常见问题

TODO：

- 【已解决】class-dump导出砸壳后抖音ipa的头文件为空
- 【已解决】class-dump导出Framework二进制AwemeCore报错：Cannot find offset for address in dataOffsetForAddress
- 【未解决】Mac中无法删除临时目录出现没有权限Operation not permitted
- 【已解决】砸壳后抖音ipa安装失败：DeviceNotSupportedByThinning

---

help帮助语法

```
class-dump 3.5 (64 bit)
Usage: class-dump [options] mach-o-file

  where options are:
    -a             show instance variable offsets
    -A             show implementation addresses
    --arch  arch   choose a specific architecture from a universal binary (ppc, ppc64, i386, x86_64)
    -C  regex      only display classes matching regular expression
    -f  str        find string in method name
    -H             generate header files in current directory, or directory specified with -o
    -I             sort classes, categories, and protocols by inheritance (overrides -s)
    -o  dir        output directory used for -H
    -r             recursively expand frameworks and fixed VM shared libraries
    -s             sort classes and categories by name
    -S             sort methods by name
    -t             suppress header in output, for testing
    --list-arches  list the arches in the file, then exit
    --sdk-ios      specify iOS SDK version (will look in /Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS version
.sdk
    --sdk-mac      specify Mac OS X version (will look in /Developer/SDKs/MacOSX version .sdk
    --sdk-root     specify the full SDK root path (or use --sdk-ios/--sdk-mac for a shortcut)
```

分析代码逻辑

iOS逆向的静态分析期间，最大的最有用的，还是：分析代码逻辑

iOS逆向方面，常用的分析代码逻辑的工具有：

- `IDA`
- `Hopper`

IDA

TODO：

- 【记录】IDA分析脱壳后抖音ipa的二进制AwemeCore
- 【已解决】iOS底层函数：objc_enumerationMutation
- 【未解决】研究抖音越狱检测逻辑：___lldb_unnamed_symbol1841884即sub_11326A84

后已整理出独立教程，详见：

逆向利器：IDA (crifan.org)

iOS逆向之IDA使用心得

objc_enumerationMutation表示循环的逻辑

iOS逆向期间，IDA的伪代码中，有时候会看到：`objc_enumerationMutation`

其含义是：`只是表示这段代码是循环遍历而已`

代码举例：

```
v4 = (void *)objc_retain(v2->_allTrackRenderers);
v5 = v4;
v6 = objc_msgSend(v4, "countByEnumeratingWithState:objects:count:", &v24, v28, 16LL);
if ( v6 )
{
  v7 = v6;
  v8 = **(_QWORD **)&v25;
  do
  {
    v9 = 0LL;
    do
    {
      if ( **(_QWORD **)&v25 != v8 )
        objc_enumerationMutation(v5);
      objc_msgSend(*(void **)(*((_QWORD *)&v24 + 1) + 8 * v9), "terminate");
    }
    while ( v9 < (unsigned __int64)v7 );
    v7 = objc_msgSend(v5, "countByEnumeratingWithState:objects:count:", &v24, v28, 16LL);
  }
  while ( v7 );
}
```

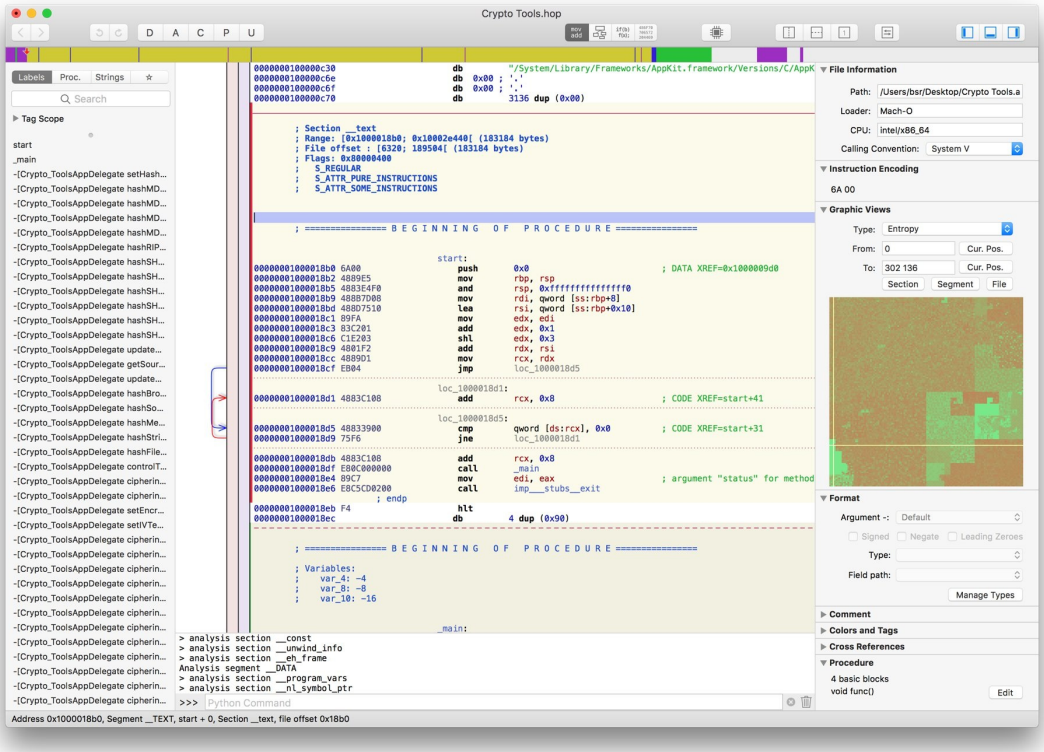
和：

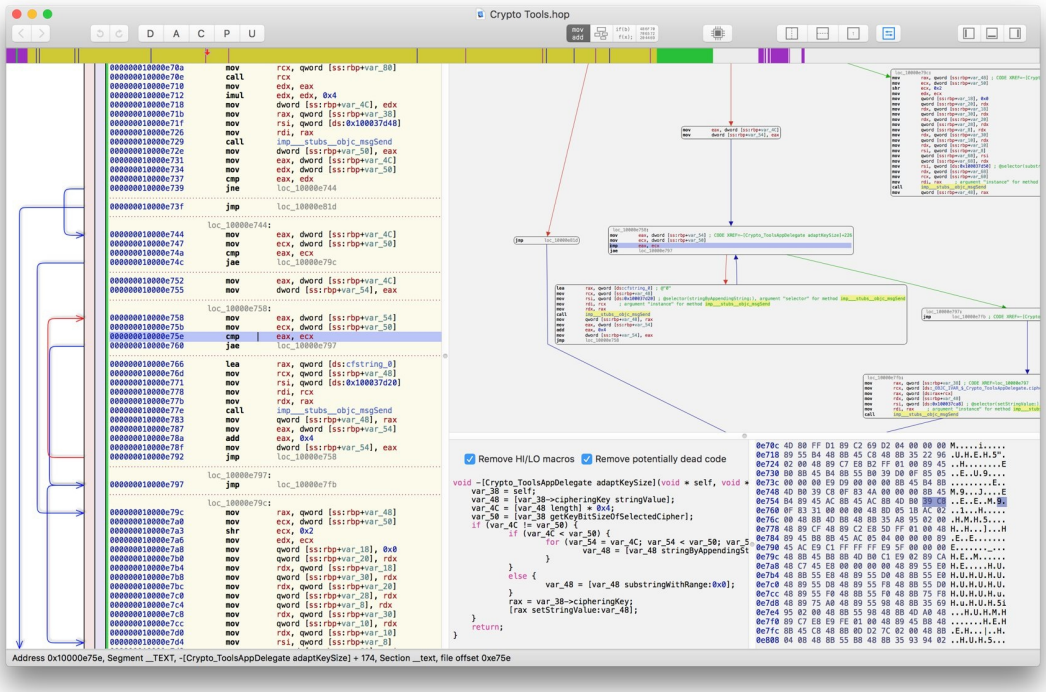
```
    do
    {
      v34 = 0LL;
* * *
      v81 = v31;
      do
      {
        if ( **(_QWORD **)&v98 != v32 )
          objc_enumerationMutation(v28);
        v86 = v34;
        v35 = *(_QWORD *)(*((_QWORD *)&v97 + 1) + 8 * v34);
        v85 = jmp_objc_msgSend_x19tox2(v26[83], objectForKeyedSubscript__);
* * *
      }
      while ( v86 + 1 < (unsigned __int64)v31 );
```

```
    v31 = objc_msgSend(v28, v73, v97, v109, 16LL);
  }
  while ( v31 );
```

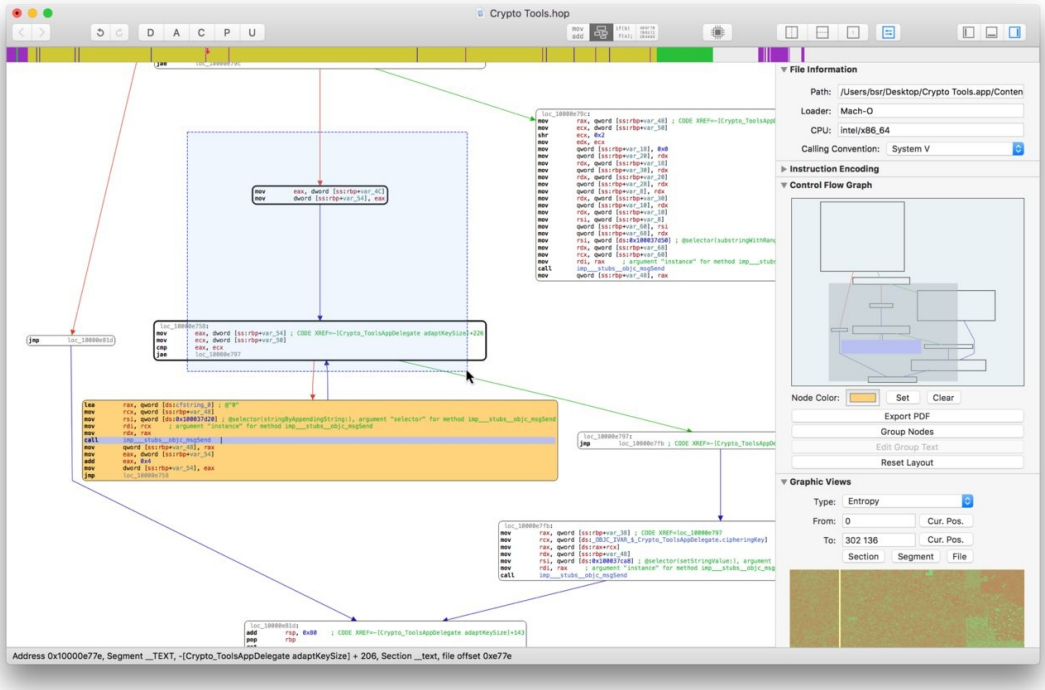
Hopper

- `Hopper Disassembler`
  - 常简称：`Hopper`
    - 偶尔缩写为：`hd`
  - 是什么：Hopper is a reverse engineering tool for OS X and Linux
  - 一句话描述：
    - the reverse engineering tool that lets you disassemble, decompile and debug your applications
  - 功能：disassemble and decompile
  - 支持平台、架构：32/64bits Intel Mac, Linux, Windows and iOS executables
  - 详解
    - 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.
  - 主要用于：二进制的静态分析
  - 官网
    - https://www.hopperapp.com
  - 截图

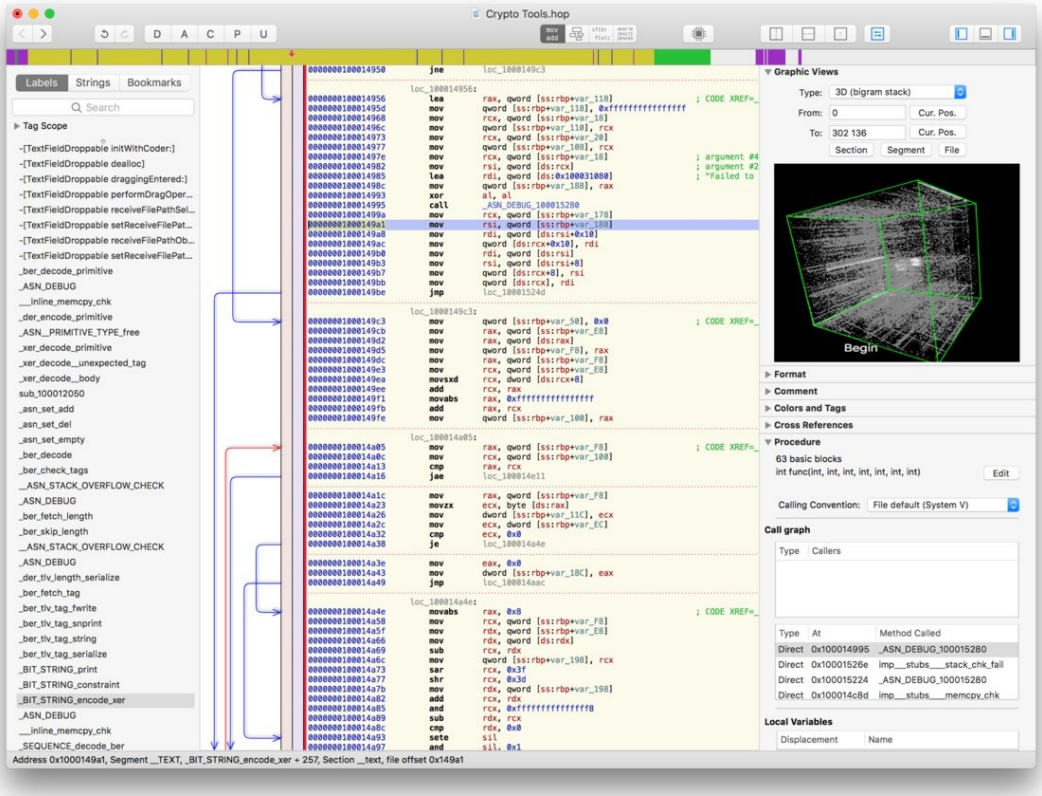



Crypto Tools.hop
Remove HI/LO macros
Remove potentially dead code
Address 0x10000e75e, Segment __TEXT, -[Crypto_ToolsAppDelegate adaptKeySize] + 174, Section __text, file offset 0xe75e

■


Crypto Tools.hop
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

■



Hooper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。

注：Hooper本身，想要打开二进制的话，如果二进制太大，比如：

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

经常直接卡死，无法正常打开

-》导致估计导出全部伪代码，也很困难。

-》想要尝试的话，估计也要找个小一点的二进制去测试。

静态分析实例

此处给出具体的实际的例子，来说，具体如何去静态分析，以及输出结果如何。

查看信息和导出字符串

TODO：

- 【记录】静态分析iOS的17.8.0旧版抖音
- 【记录】静态分析黑豹二进制HeiBao
- 【记录】静态分析黑豹动态库zzzzHeiBaoLib.dylib
- 【记录】用jtool查看抖音二进制信息
- 【记录】用rabin2查看抖音AwemeCore二进制的信息
- 【记录】静态分析Mask的动态库：Mask.dylib

此处对于静态分析中的，查看二进制信息和导出字符串等资源，给出实例，供参考。

HeiBao的dylib

对于二级制文件，此处是dylib的动态库：

```
➜  DynamicLibraries pwd
/Users/crifan/dev/DevRoot/Aweme/exportFromiPhone/iPhoneX-137/Library/MobileSubstrate/DynamicLibraries
➜  DynamicLibraries ll
total 152568
-rwxr-xr-x@ 1 crifan  staff   6.2M  3 14 10:34 zzzzHeiBaoLib.dylib
-rw-r--r--  1 crifan  staff    68M  3 17 21:39 zzzzHeiBaoLib.i64
```

去导出字符串等资源：

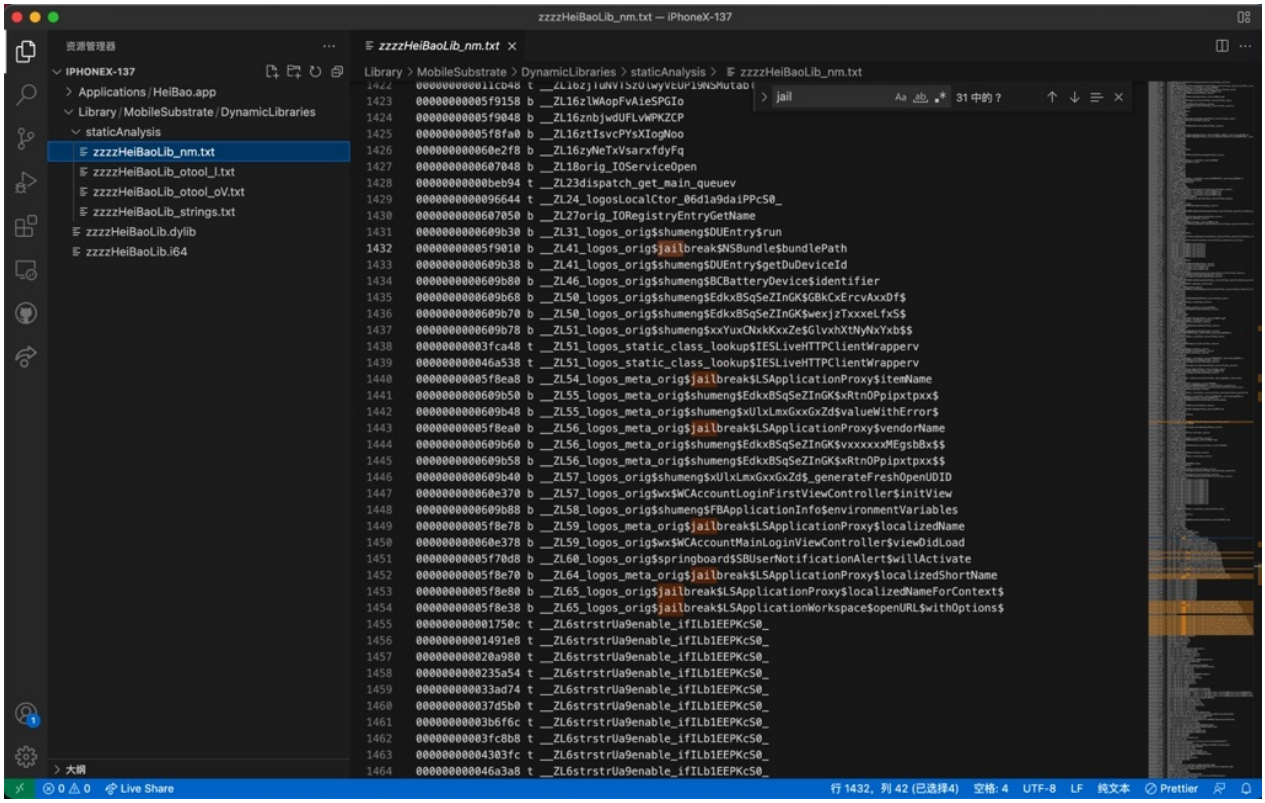
```
➜  DynamicLibraries otool -l zzzzHeiBaoLib.dylib > HeiBaoLib_otool_l.txt
➜  DynamicLibraries otool -oV zzzzHeiBaoLib.dylib > HeiBaoLib_otool_oV.txt
➜  DynamicLibraries nm zzzzHeiBaoLib.dylib > HeiBaoLib_nm.txt
➜  DynamicLibraries strings zzzzHeiBaoLib.dylib > HeiBaoLib_strings.txt

➜  DynamicLibraries ll
total 153112
-rw-r--r--  1 crifan  staff   108K  3 21 10:00 HeiBaoLib_nm.txt
-rw-r--r--  1 crifan  staff    12K  3 21 10:00 HeiBaoLib_otool_l.txt
-rw-r--r--  1 crifan  staff    51K  3 21 10:00 HeiBaoLib_otool_oV.txt
-rw-r--r--  1 crifan  staff    89K  3 21 10:00 HeiBaoLib_strings.txt
-rwxr-xr-x@ 1 crifan  staff   6.2M  3 14 10:34 zzzzHeiBaoLib.dylib
-rw-r--r--  1 crifan  staff    68M  3 17 21:39 zzzzHeiBaoLib.i64
```

后续即可去分析和搜索想要研究的值了。

比如：

搜索越狱 jailbreak 相关内容：



Mask的dylib

对于一个二进制，此处是一个动态库文件 `Mask.dylib`，想要导出字符串等资源，供后续分析。

典型的成套的做法是：

```
otool -l Mask.dylib > MaskDylib_otool_l.txt
otool -oV Mask.dylib > MaskDylib_otool_oV.txt

nm Mask.dylib > MaskDylib_nm.txt

strings Mask.dylib > MaskDylib_strings.txt

jtool2 -h Mask.dylib > MaskDylib_jtool2_h_header.txt
jtool2 -l Mask.dylib > MaskDylib_jtool2_l_list.txt
jtool2 -L Mask.dylib > MaskDylib_jtool2_L_library.txt
jtool2 -S Mask.dylib > MaskDylib_jtool2_S_symbol.txt
jtool2 --analyze Mask.dylib > MaskDylib_jtool2_analyze.txt

rabin2 -I Mask.dylib > MaskDylib_rabin2_I_identification.txt
rabin2 -i Mask.dylib > MaskDylib_rabin2_i_imports.txt
rabin2 -E Mask.dylib > MaskDylib_rabin2_E_exports.txt
rabin2 -l Mask.dylib > MaskDylib_rabin2_l_libraries.txt
rabin2 -z Mask.dylib > MaskDylib_rabin2_z_strings.txt
rabin2 -s Mask.dylib > MaskDylib_rabin2_s_symbols.txt
rabin2 -S Mask.dylib > MaskDylib_rabin2_S_sections.txt
```

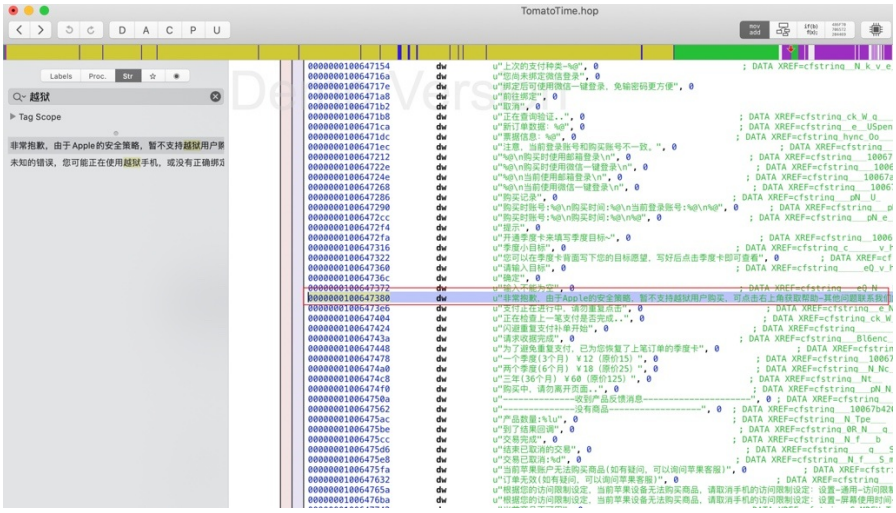
经验心得

此处整理，iOS逆向之静态分析方面的一些经验和心得。

IDA vs Hopper

TODO：

- 【整理】iOS逆向工具：IDA和 Hopper 对比

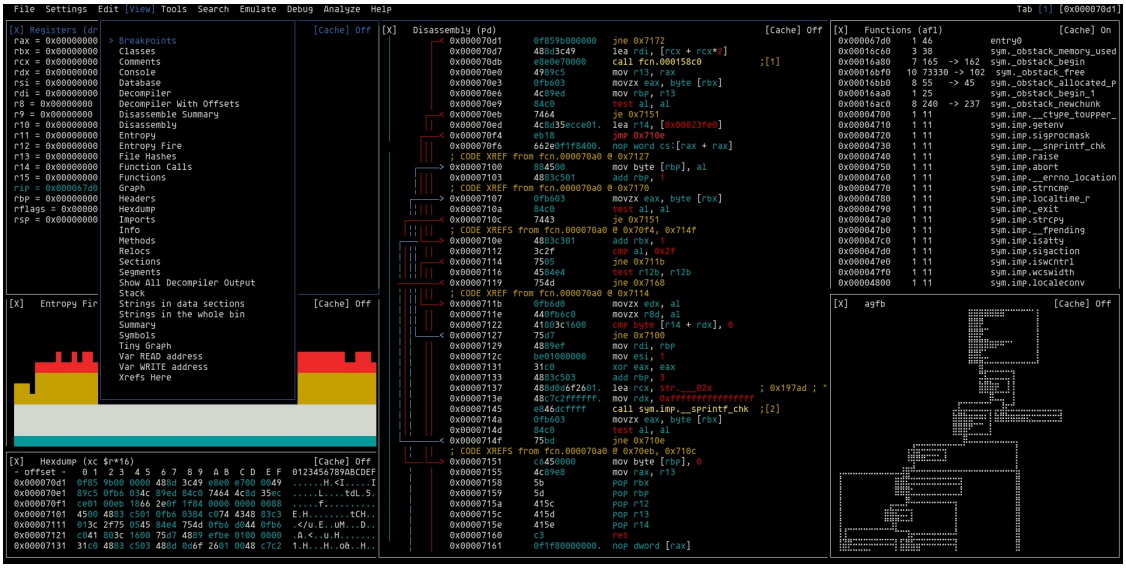
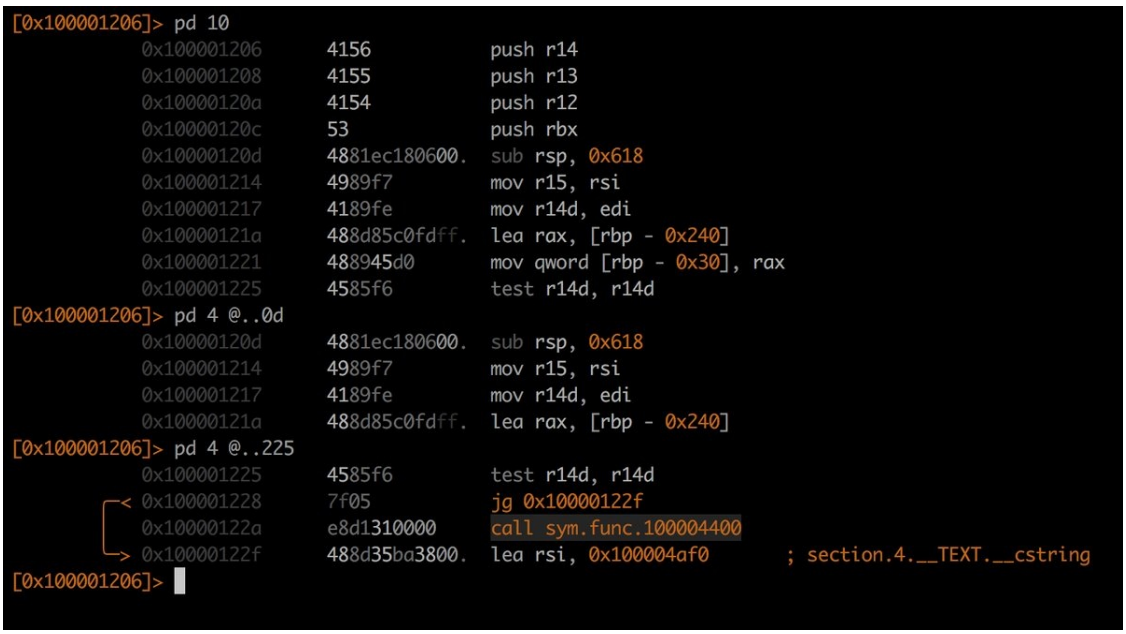
- Hopper vs IDA
  - 平台支持
    - Hopper：更倾向于 `Mac`
    - IDA：支持多平台： `Windows` 、 `Linux` 、 `Mac`
  - 功能支持
    - 总体上还是IDA更强大，Hopper相对较弱
      - 不过据说部分细节方面，有些Hopper支持更好?
        - 比如
          - 中文字符搜索
            - IDA 7.0+：不支持
            - Hopper：支持
            - 

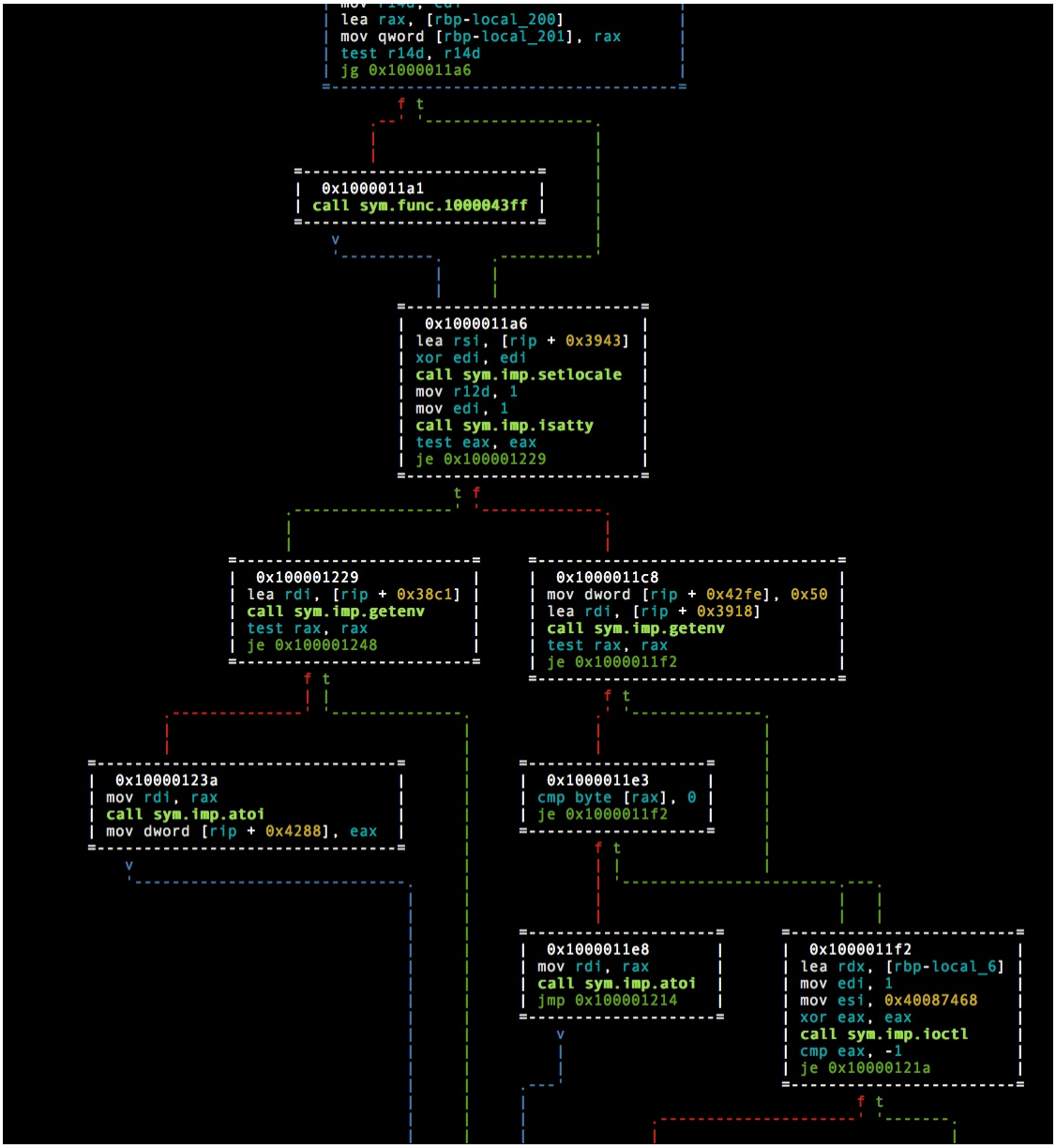
相关工具

对于iOS逆向中的静态分析，其实还有其他一些相关工具，值得介绍。

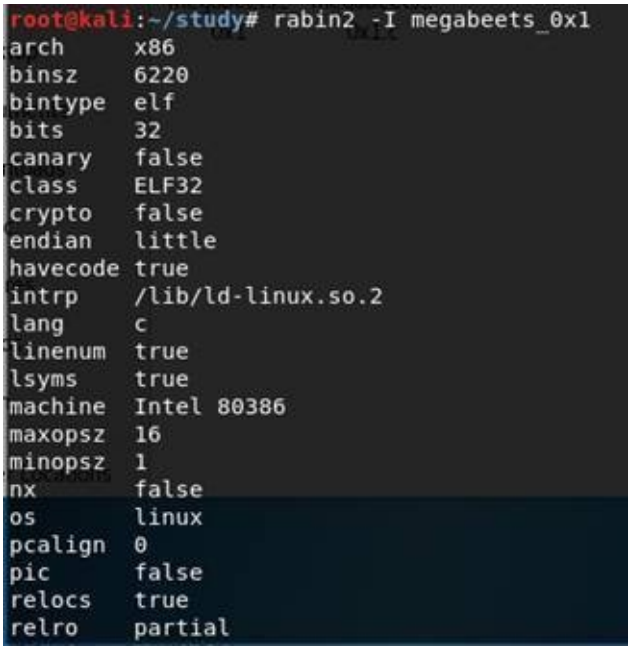
- `radare2`：成套的逆向工具
  - 包含多个命令行的工具，前面提到的rabin2就是其中之一
- `Cutter`：radare2的GUI版本
- Miasm
  - 一句话介绍：用Python实现的逆向工程框架
    - Reverse engineering framework in Python
  - 用途：分析、修改、生成 二进制程序
    - analyze / modify / generate binary programs
  - 特性
    - Opening / modifying / generating PE / ELF 32 / 64 LE / BE
    - Assembling / Disassembling X86 / ARM / MIPS / SH4 / MSP430
    - Representing assembly semantic using intermediate language
    - Emulating using JIT (dynamic code analysis, unpacking, ...)
    - Expression simplification for automatic de-obfuscation
  - 资料
    - GitHub
      - cea-sec/miasm: Reverse engineering framework in Python
        - https://github.com/cea-sec/miasm
    - 官网
      - Home — Miasm's blog
        - https://miasm.re/blog/

radare2

- `radare2`
  - 是什么：一个著名的开源逆向工程平台
    - Unix-like reverse engineering framework and commandline tools
  - 评价
    - 可谓是一大神器
      - 支持包括反汇编、分析数据、打补丁、比较数据、搜索、替换、虚拟化等等，同时具备超强的脚本加载能力，并且可以运行在几乎所有主流的平台
  - 竞品
    - `IDA`
  - 截图
    - 
    - 



- 支持平台
  - Mac
  - Windows
  - Linux
  - Solaris
  - Android
  - iOS
  - Haiku
- 历史
  - Radare project started as a forensics tool, a scriptable commandline hexadecimal editor able to open disk files but later support for analyzing binaries, disassembling code, debugging programs, attaching to remote gdb servers
- 功能：Radare is a portable reversing framework that can
  - Disassemble (and assemble for) many different architectures
  - Debug with local native and remote debuggers (gdb, rap, webui, r2pipe, winedbg, windbg)
  - Run on Linux, *BSD, Windows, OSX, Android, iOS, Solaris and Haiku
  - Perform forensics on filesystems and data carving
  - Be scripted in Python, Javascript, Go and more
  - Support collaborative analysis using the embedded webserver

        - Visualize data structures of several file types
        - Patch programs to uncover new features or fix vulnerabilities
        - Use powerful analysis capabilities to speed up reversing
        - Aid in software exploitation
    - 特性
        - Batch, commandline, visual and panels interactive modes
        - Embedded webserver with js scripting and webui
        - Assemble and disassemble a large list of CPUs
        - Runs on Windows and any other UNIX flavour out there
        - Analyze and emulate code with ESIL
        - Native debugger and GDB, WINDBG, QNX and FRIDA
        - Navigate ascii-art control flow graphs
        - Ability to patch binaries, modify code or data
        - Search for patterns, magic headers, function signatures
        - Easy to extend and modify
        - Commandline, C API, script with r2pipe in any language
    - 包含工具
        - `rabin2`
            - 获取 `ELF`，`PE`，`Mach-O`，`Java CLASS` 文件的区段、头信息、导入导出表、字符串相关、入口点等等
                - 包括打印出二进制文件的系统属性、语言、字节序、框架、以及使用了哪些加固技术
            - 支持多种格式的输出文件
            - 截图
            - 

        - `radiff2`：比较文件不同的
        - `rahash2`：各种密码算法，hash算法集成
        - `rasm2`：汇编和反汇编
        - `ragg2`：开发shellcode工具(radare2自己编写的编译器)
        - `radare2`：整合了所有工具
    - 资料
        - 官网
            - radare
                - https://rada.re/n/radare2.html
        - GitHub
            - radareorg/radare2: UNIX-like reverse engineering framework and command-line toolset
                - https://github.com/radareorg/radare2
        - 教程

- The Official Radare2 Book
  - https://book.rada.re/index.html

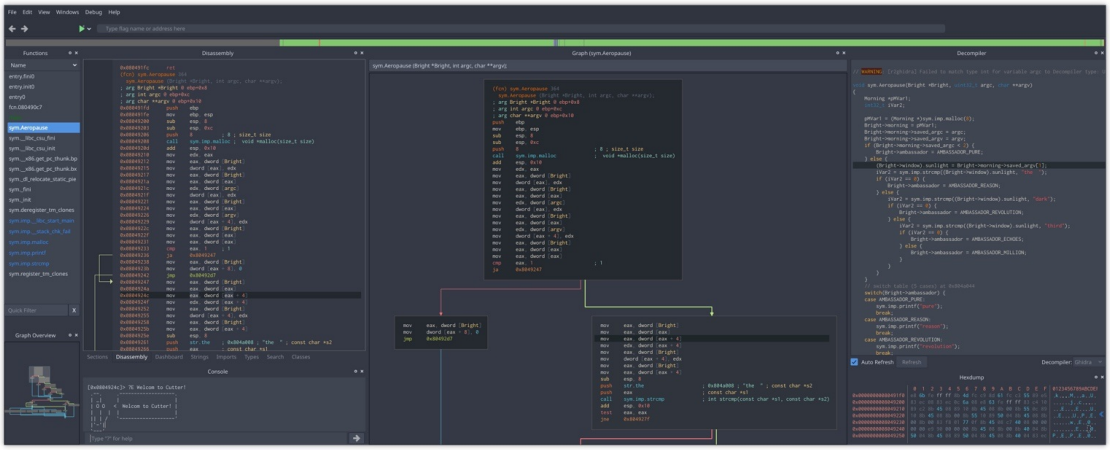
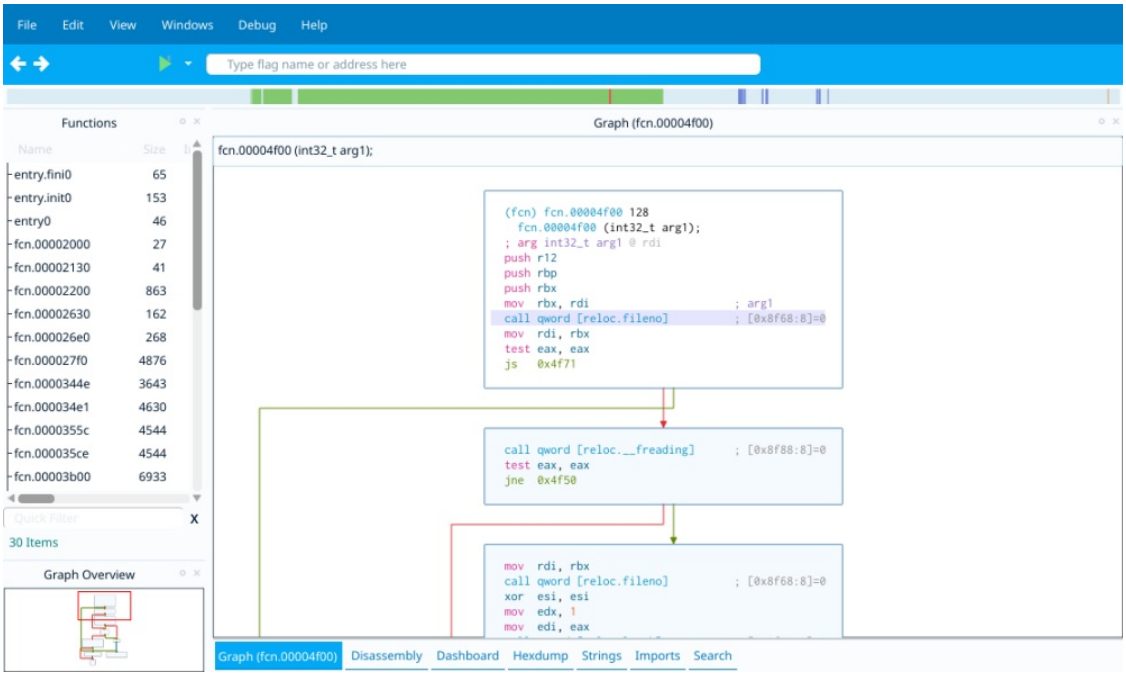
help帮助语法

```
$ radare2 -h
Usage: r2 [-ACdfLMnNqStuvwzX] [-P patch] [-p prj] [-a arch] [-b bits] [-i file]
          [-s addr] [-B baddr] [-m maddr] [-c cmd] [-e k=v] file|pid|-|--
 --            run radare2 without opening any file
 -             same as 'r2 malloc://512'
               read file from stdin (use -i and -c to run cmds)
 -=            perform =! command to run all commands remotely
 -0            print \x00 after init and every command
 -2            close stderr file descriptor (silent warning messages)
 -a [arch]     set asm.arch
 -A            run 'aaa' command to analyze all referenced code
 -b [bits]     set asm.bits
 -B [baddr]    set base address for PIE binaries
 -c 'cmd..'    execute radare command
 -C            file is host:port (alias for -c+=http://%s/cmd/)
 -d            debug the executable 'file' or running process 'pid'
 -D [backend]  enable debug mode (e cfg.debug=true)
 -e k=v        evaluate config var
 -f            block size = file size
 -F [binplug]  force to use that rbin plugin
 -h, -hh       show help message, -hh for long
 -H ([var])    display variable
 -i [file]     run script file
 -I [file]     run script file before the file is opened
 -k [OS/kern]  set asm.os (linux, macos, w32, netbsd, ...)
 -l [lib]      load plugin file
 -L            list supported IO plugins
 -m [addr]     map file at given address (loadaddr)
 -M            do not demangle symbol names
 -n, -nn       do not load RBin info (-nn only load bin structures)
 -N            do not load user settings and scripts
 -q            quiet mode (no prompt) and quit after -i
 -Q            quiet mode (no prompt) and quit faster (quickLeak=true)
 -p [prj]      use project, list if no arg, load if no file
 -P [file]     apply rapatch file and quit
 -r [rarun2]   specify rarun2 profile to load (same as -e dbg.profile=X)
 -R [rr2rule]  specify custom rarun2 directive
 -s [addr]     initial seek
 -S            start r2 in sandbox mode
 -t            load rabin2 info in thread
 -u            set bin.filter=false to get raw sym/sec/cls names
 -v, -V        show radare2 version (-V show lib versions)
 -w            open file in write mode
 -x            open without exec-flag (asm.emu will not work), See io.exec
 -X            same as -e bin.usextr=false (useful for dyldcache)
 -z, -zz       do not load strings or load them even in raw
```

R2Pipe

- `R2Pipe`
  - 是什么： `R2Pipe` 是一个可以调用 `radare2` 的 `Python` 脚本库
  - 示例代码
    - https://github.com/radareorg/radare2-r2pipe/tree/master/python/examples

Cutter

- `Cutter`
  - 一句话描述：radare2的GUI版本
    - Free and Open Source RE Platform powered by radare2
    - Cutter is the official UI for radare2 for Linux, macOS and Windows, it's written in C++ and uses the Qt
  - 支持多平台
    - `Linux`
    - `Mac`
    - `Windows`
  - 实现细节
    - `C++` 语言写的
    - 前端：`QT`
  - 截图
    - 
    - 

  - 特点
    - 内置Ghidra decompiler
      - 无需额外安装Java
  - 核心功能和特点
    - 开源 Open Source
      - Completely FREE and licensed under GPLv3

        - Decompiler
            - Native integration of Ghidra's decompiler in Cutter releases
        - Graph View
            - Fully featured graph view as well as mini-graph for fast navigation
        - Debugger
            - Multiplatform native and remote debugger for dynamic analysis
        - Disassembly
            - Linear disassembly view
        - Hex Editor
            - View and modify any file with a rich and powerful Hex View
        - Python Scripting Engine
            - Quickly write python scripts to automate tasks
        - Plugins
            - Use Native or Python plugins to extend Cutter's core functionality
        - Binary Patching
            - Add, remove and modify bytes and instructions
        - Emulation
            - Great for automation, crypto algorithms and malware analysis
        - Theme Editor
            - Fully featured theme editor for easy and user-friendly customization of Cutter
        - Modern & Customizable UI
            - Built using Qt C++ and design best practices
        - Integrated Radare2 Console
        - Multi Language
        - Binary Searching
        - Types & Structs
        - Syntax Highlighting
        - STDIO Redirection
        - Remote Debugging
        - Kernel Debug
        - Graph Overview
    - 资料
        - 官网
            - Cutter
                - https://cutter.re
        - GitHub
            - radareorg/cutter: Free and Open Source Reverse Engineering Platform powered by radare2
                - https://github.com/rizinorg/cutter

附录

下面列出相关参考资料。

参考资料

- iOS逆向开发
- iOS逆向开发：砸壳ipa
-
- OS Internals: (newosxbook.com)
- IOS逆向初探 · 浮萍's Blog (fuping.site)
- iOS逆向之Reveal、Hopper、MachOView等逆向工具的安装使用 - 简书 (jianshu.com)
- Mach-O Programming Topics Introduction
- IOS逆向初探 · 浮萍's Blog (fuping.site)
- iOS逆向之Reveal、Hopper、MachOView等逆向工具的安装使用 - 简书 (jianshu.com)
- poboke/Class-Decompile: Class Decompile is a python script for Hopper Disassembler. This script can export pseudo code of the classes. (github.com)
-