

目录

前言	1.1
静态分析概览	1.2
从ipa中找到二进制	1.3
静态分析内容	1.4
查看二进制信息	1.4.1
Mach-O格式	1.4.1.1
MachOView	1.4.1.1.1
jtool2	1.4.1.1.2
rabin2	1.4.1.1.3
导出字符串资源	1.4.2
strings	1.4.2.1
nm	1.4.2.2
otool	1.4.2.3
导出头文件	1.4.3
class-dump	1.4.3.1
分析代码逻辑	1.4.4
IDA	1.4.4.1
Hopper	1.4.4.2
静态分析实例	1.5
查看信息和导出字符串	1.5.1
经验心得	1.6
IDA vs Hopper	1.6.1
附录	1.7
参考资料	1.7.1

iOS逆向开发：静态分析

- 最新版本: v0.9
- 更新时间: 20221021

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

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。如有版权，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。

更多其他电子书

本人 `crifan` 还写了其他 100+ 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](#)

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: 2022-10-21 23:33:33

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

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：2022-10-21 20:48:31

从ipa中找到二进制

TODO:

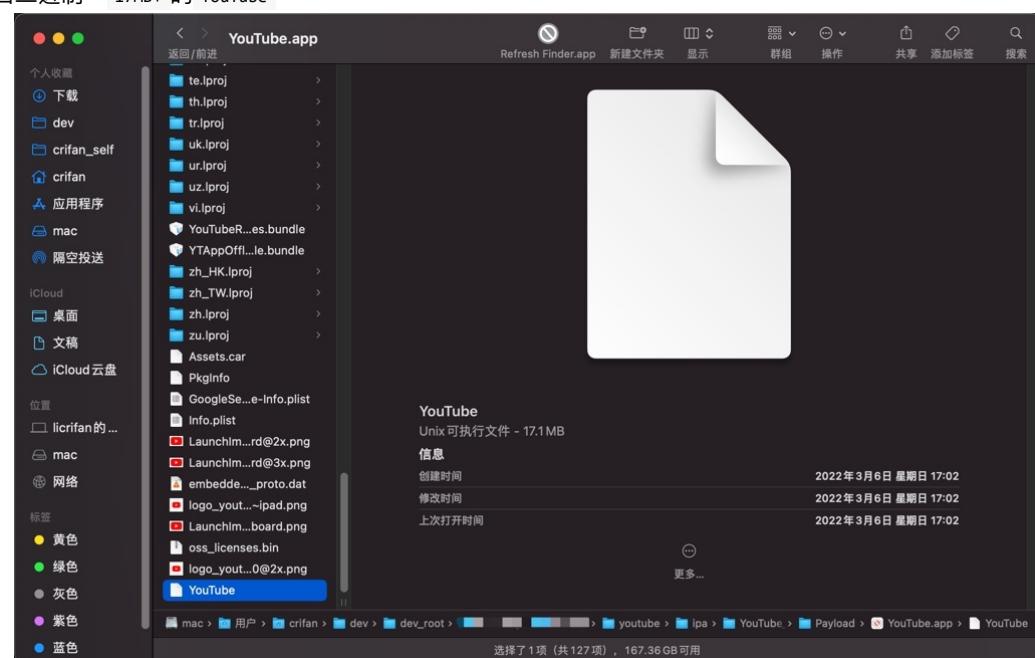
- 【未解决】静态分析抖音二进制寻找越狱检测手段
- 【已解决】如何从脱壳后的抖音的IPA文件中得到二进制文件
- 相关
 - 【已解决】越狱iPhone中抖音app的安装目录安装位置
 - 【已解决】研究iOS中app的目录的UUID类的值和app名称如何映射

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。

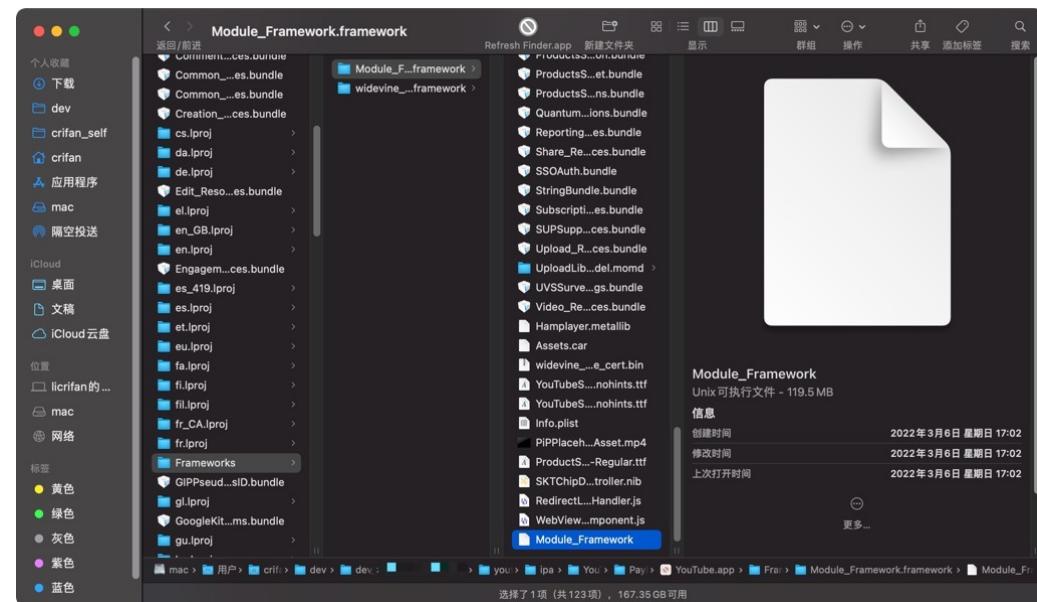
对应的就是，在前一步，[从app砸壳得到的ipa文件中](#)，找到对应的二进制文件，用于后续的静态分析。

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

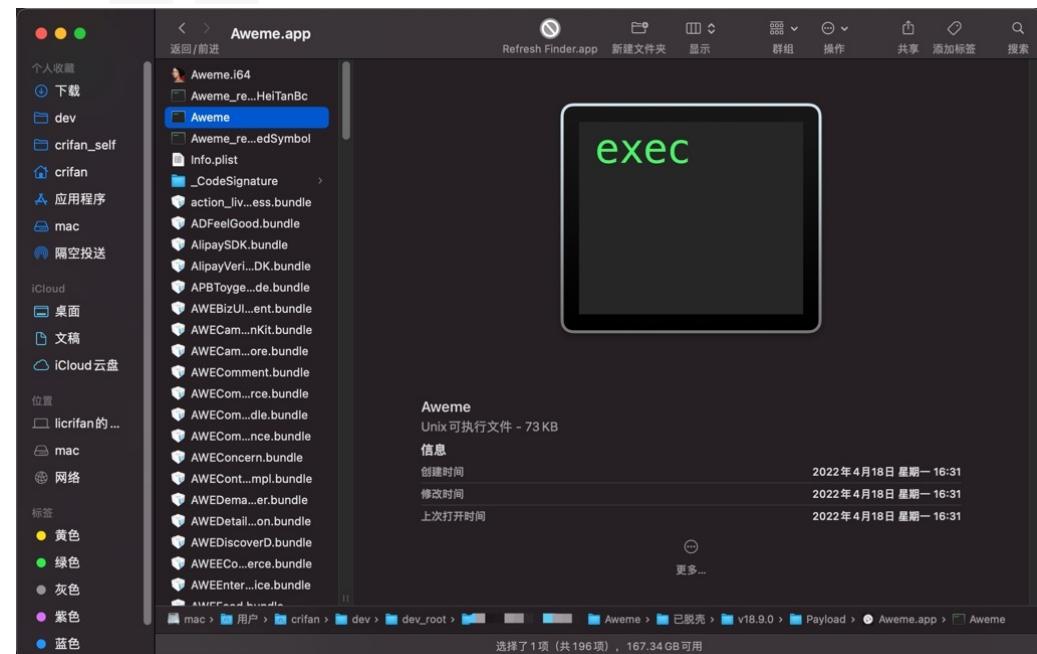


- 抖音

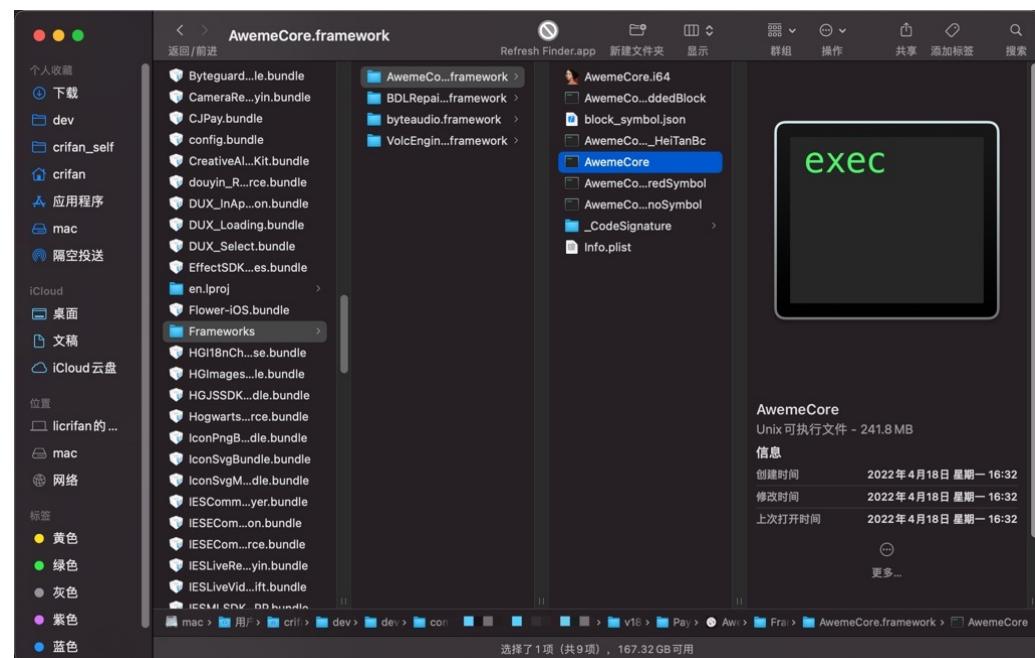
- v18.9.0

- 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



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: 2022-10-21 23:19:35

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

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: 2022-10-21 20:51:18

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。

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：2022-10-21 20:57:33

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

```
#define MH_OBJECT    0x1      /* relocatable object file */
#define MH_EXECUTE   0x2      /* demand paged executable file */
#define MH_FVMLIB   0x3      /* fixed VM shared library file */
#define MH_CORE     0x4      /* core file */
#define MH_PRELOAD   0x5      /* preloaded executable file */
#define MH_DYLIB     0x6      /* dynamically bound shared library */
#define MH_DYLINKER  0x7      /* dynamic link editor */
#define MH_BUNDLE    0x8      /* dynamically bound bundle file */
#define MH_DYLIB_STUB 0x9      /* shared library stub for static */
                           /* linking only, no section contents */
#define MH_DSYM      0xa      /* companion file with only debug */
                           /* sections */
#define MH_KEXT_BUNDLE 0xb      /* x86_64 kexts */
```

- 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 inputMacOFile -thin ArchType -output OutputFile
 - 合并多种架构: lipo inputMacOFile1 inputMacOFile2 -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：

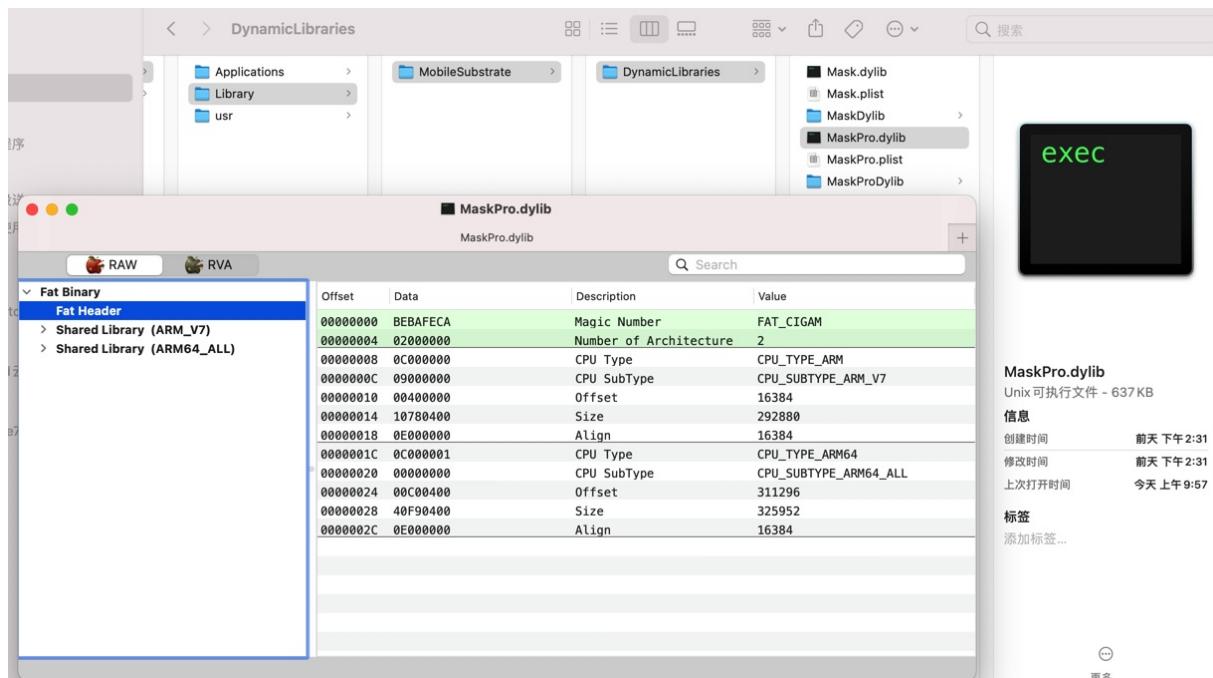
```
→ DynamicLibraries jtool2 -h MaskPro.dylib > MaskProDylib/MaskProDylib_jtool2_h_header.txt
Fat binary, little-endian, 2 architectures: armv7, arm64
Select an architecture setting the ARCH environment variable
```

即，一个Dylib中，包含了多种架构，此处是 armv7 和 arm64

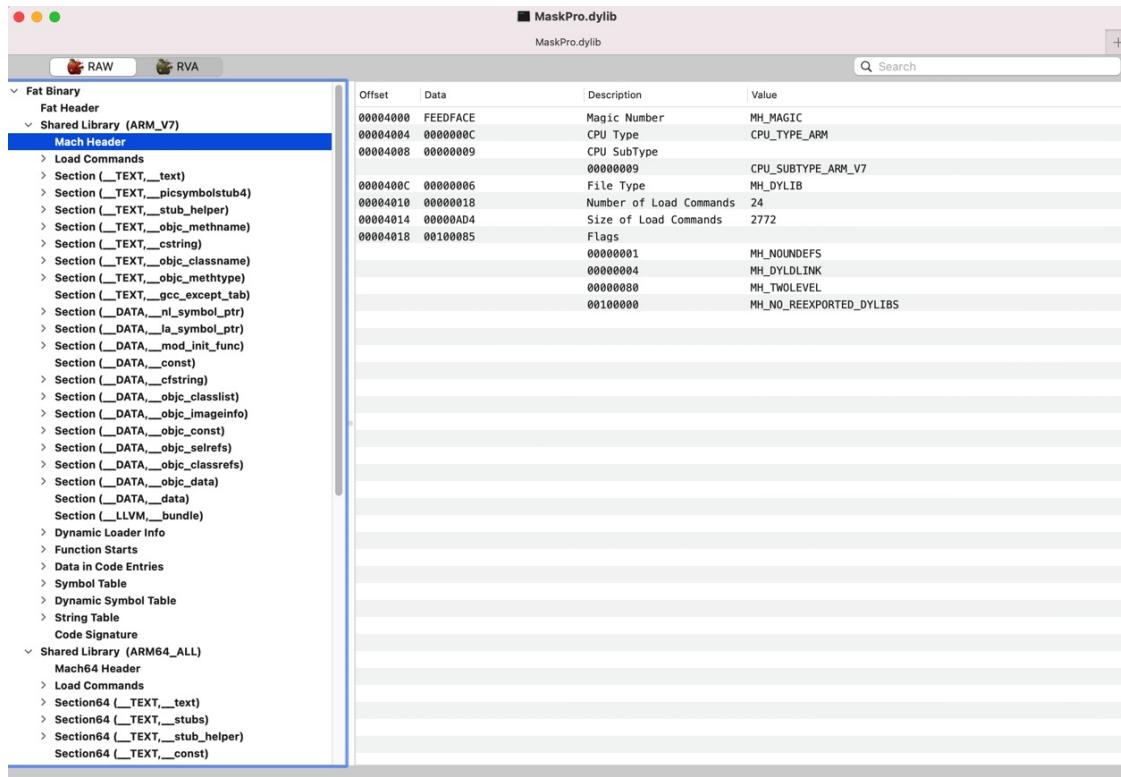
此处要指定具体架构，才能继续用 jtool2 查看信息：

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

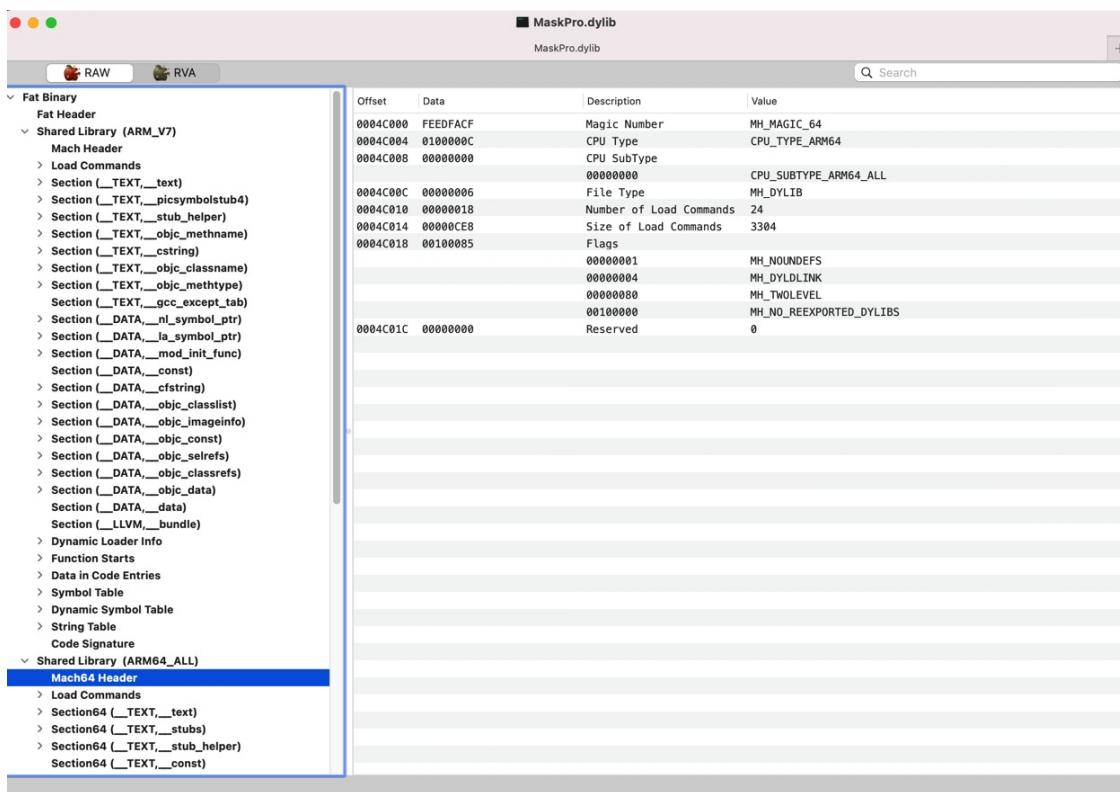
类似的，后续去用 MachOView 查看信息，也能看到是：FAT Binary



o ARMV7



o ARM64

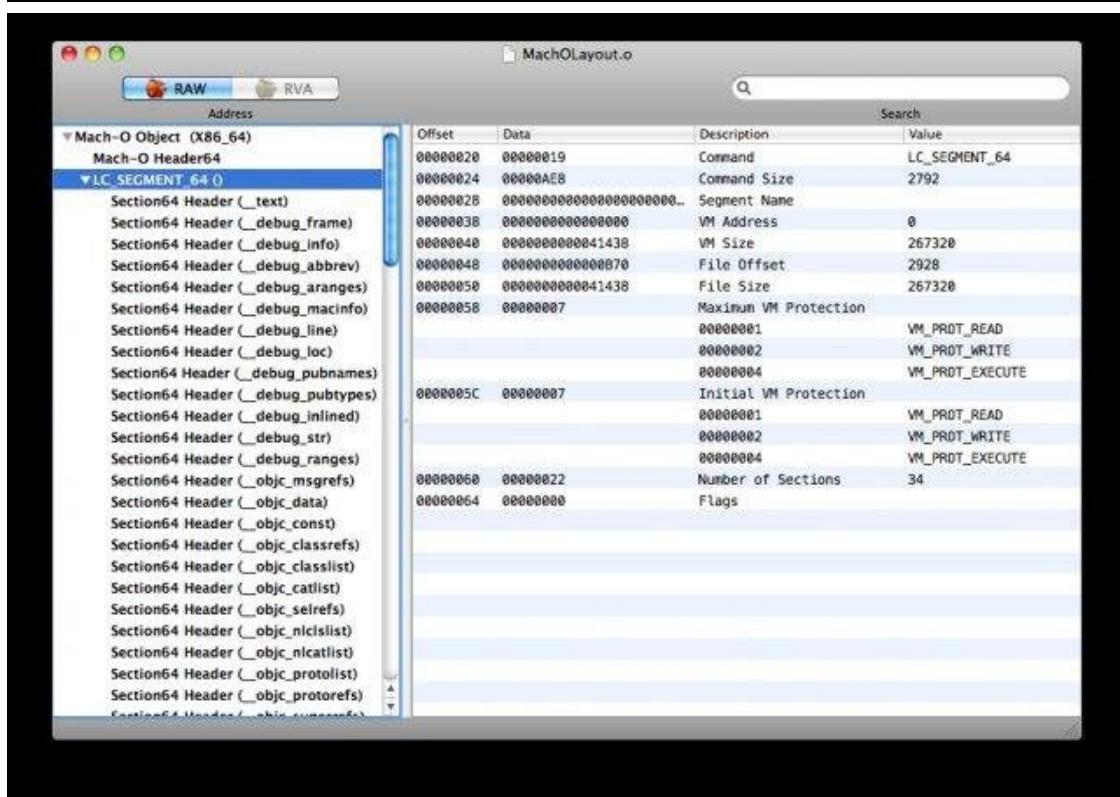
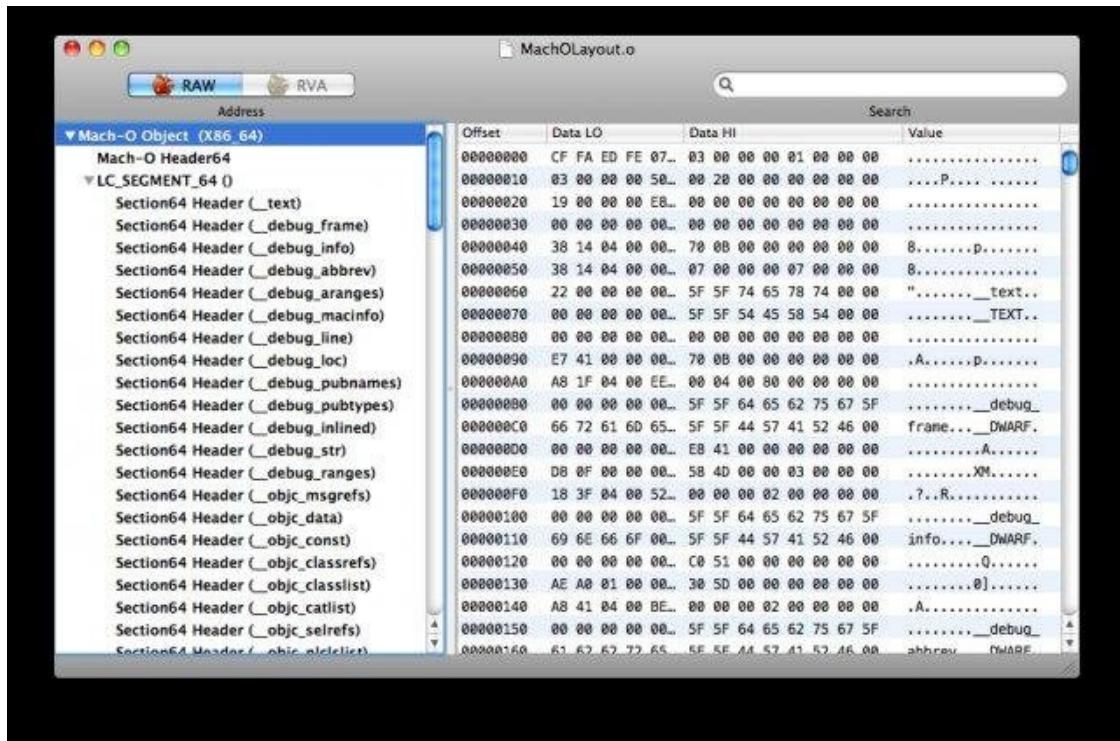


crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook 最后更新: 2022-10-21 23:29:33

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

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：2022-10-21 22:54:52

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>

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： 2022-10-21 22:55:46

rabin2

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: 2022-10-21 21:00:17

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，供后续研究分析。

常用的导出字符串等资源的工具有：

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

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: 2022-10-21 22:56:48

strings

TODO:

- 【记录】用strings查看dylib库中包含的字符串
-

nm

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: 2022-10-21 21:04:26

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] [-fah1LDt 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 disassemble 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

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：2022-10-21 23:21:17

class-dump

TODO:

- 【基本解决】砸壳抖音ipa后导出iOS抖音头文件
- 常见问题
 - 【已解决】class-dump导出砸壳后抖音ipa的头文件为空

-
- 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://stevenyngard.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/XCTTestDriver.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。

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       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)
```

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: 2022-10-21 22:58:47

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

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: 2022-10-21 21:14:50

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

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：

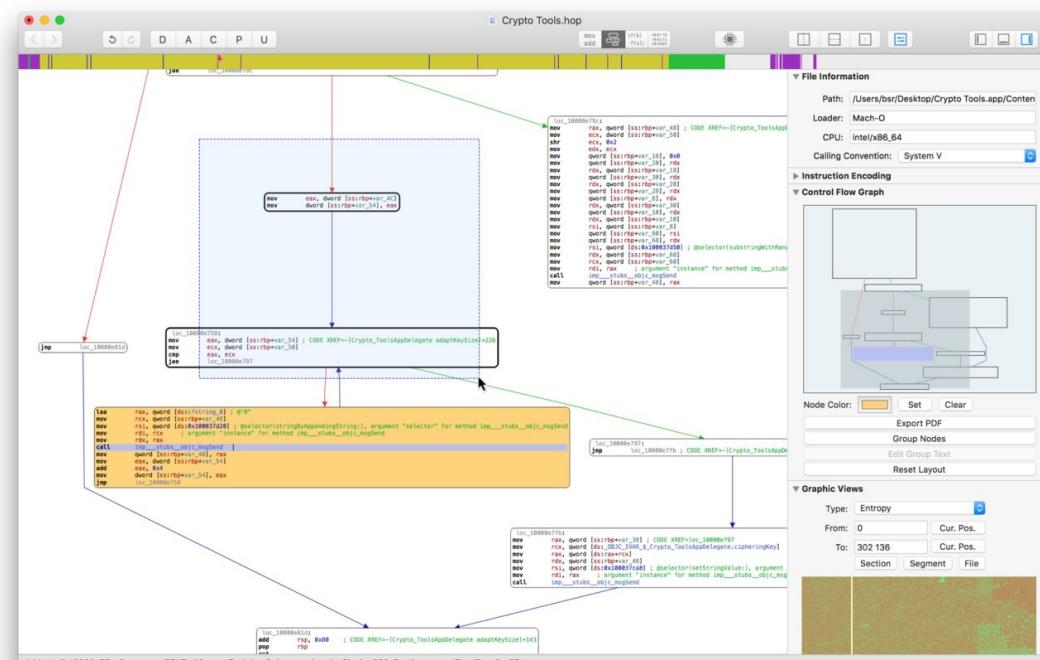
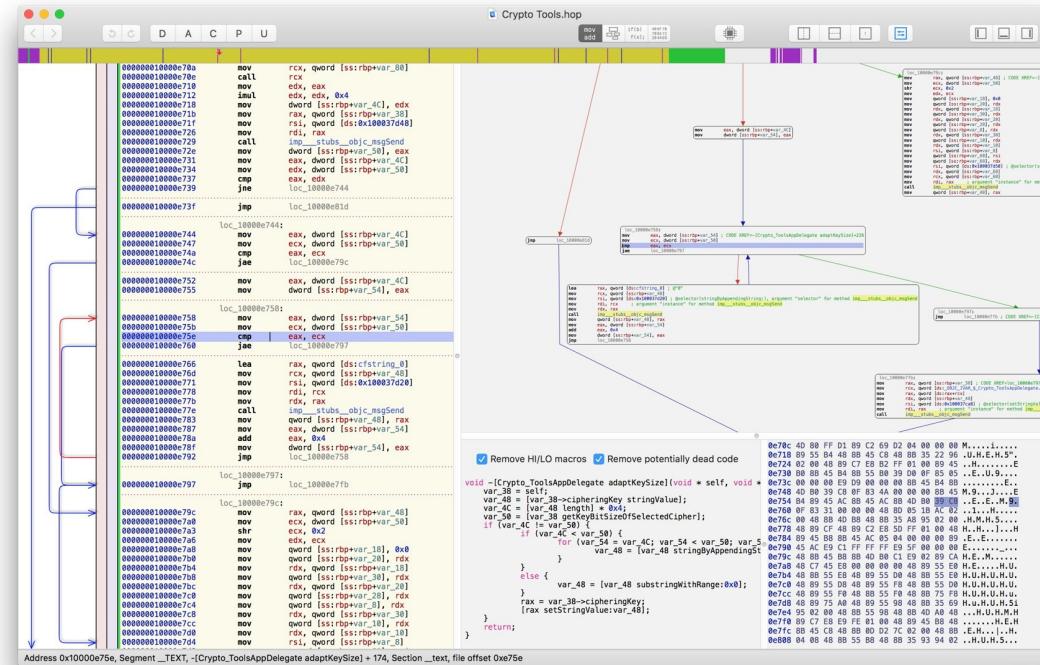
```
do
{
    v34 = 0LL;
    . . .
    v81 = v31;
    do
    {
        if ( *(__QWORD *)v98 != v32 )
            objc_enumerationMutation(v28);
        v86 = v34;
        v35 = *(__QWORD *)((__QWORD *)v97 + 1) + 8 * v34;
        v85 = jmp_objc_msgSend_x19tox2(v26[83], objectForKeyedSubscript__);
    }
    while ( v86 + 1 < (unsigned __int64)v31 );
    v31 = objc_msgSend(v28, v73, v97, v109, 16LL);
}
while ( v31 );
```

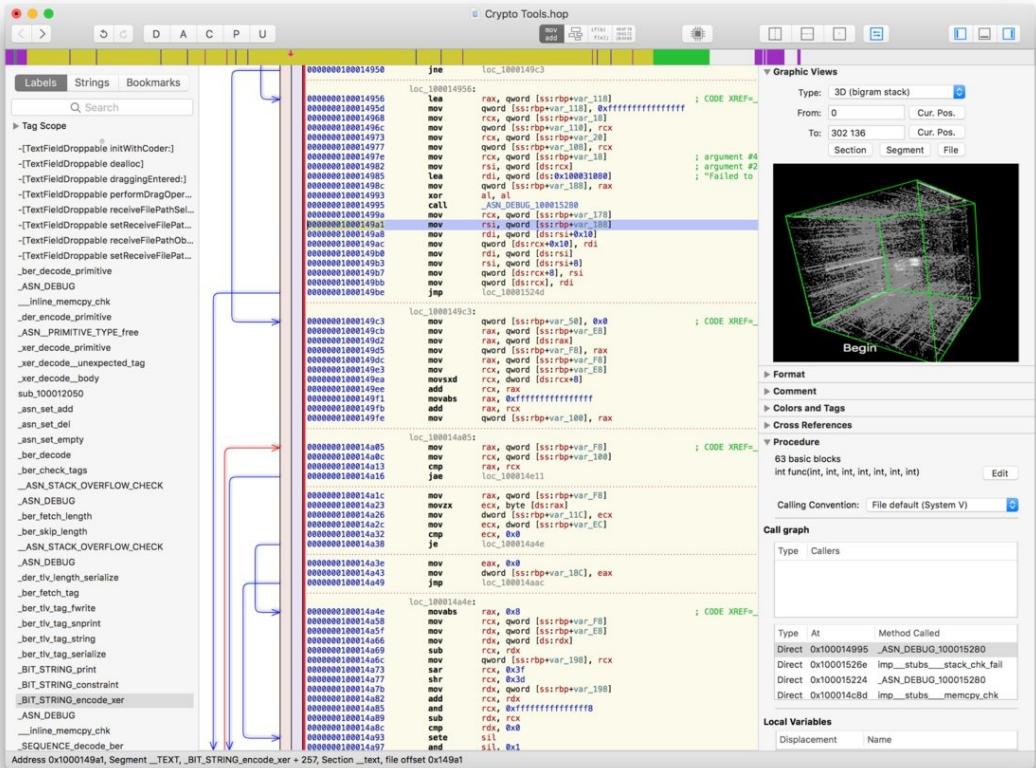

Hopper

- 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

The screenshot shows the Hopper Disassembler interface with the following details:

- File Information:** Path: /Users/bsr/Desktop/Crypto Tools.a, Loader: Mach-O, CPU: intel/x86_64, Calling Convention: System V.
- Instruction Encoding:** Shows assembly code with comments and memory dump options.
- Graphic Views:** Entropy heatmap showing the distribution of entropy across the file.
- Format:** Options for arguments, signed/negative/leading zeroes, type, and field path.
- Analysis Sections:** Includes __const, __unwind_info, __eh_frame, __DATA, __program_vars, and __NL_symbol_ptr.





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: 2022-10-21 22:16:55

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。

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：2022-10-21 21:42:43

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.164
```

去导出字符串等资源：

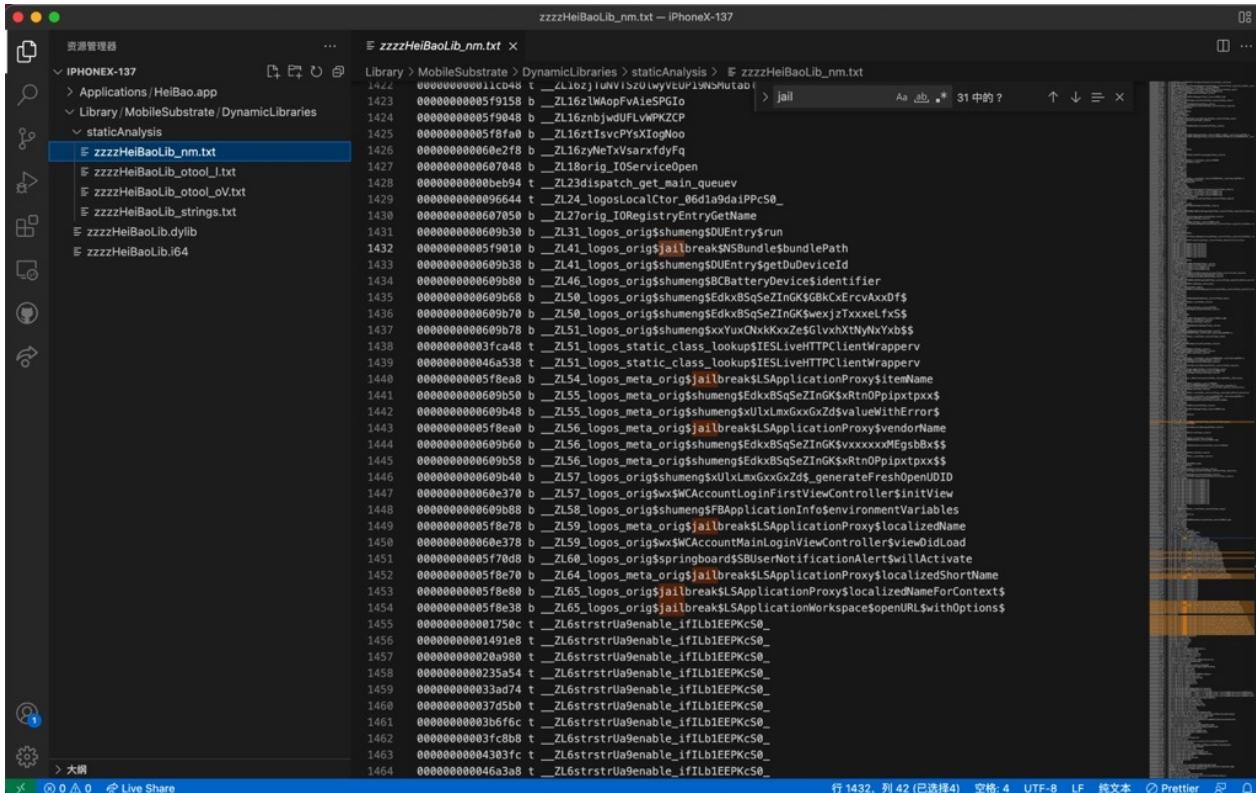
```
→ DynamicLibraries otool -l zzzzHeiBaoLib.dylib > HeiBaoLib_otool_l.txt
→ DynamicLibraries otool -oV zzzzHeiBaoLib.dylib  HeiBaoLib_otool_oV.txt
→ DynamicLibraries nm zzzzHeiBaoLib.dylib  HeiBaoLib_nm.txt
→ DynamicLibraries strings zzzzHeiBaoLib.dylib > HeiBaoLib_strings.txt

→ 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.164
```

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
```

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：2022-10-21 22:59:48

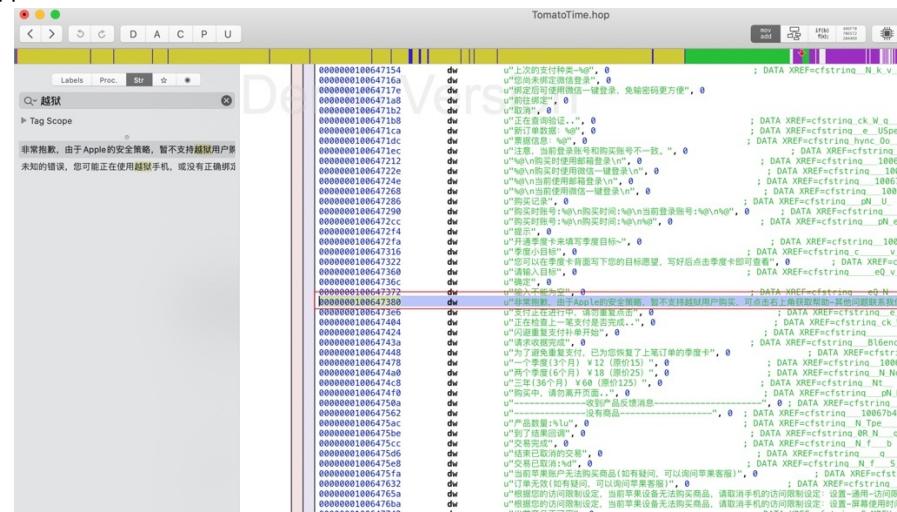
经验心得

此处整理，iOS逆向之静态分析方面的一些经验和心得。

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：2022-10-21 22:05:29

IDA vs Hopper

- 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



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: 2022-10-21 22:18:44

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: 2022-03-17 20:39:28

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 \(jianshu.com\)](#)
-

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: 2022-10-21 22:19:38