

目录

前言	1.1
iOS逆向概述	1.2
iOS逆向典型流程图	1.2.1
iOS逆向领域架构图	1.2.2
iOS逆向内容概述	1.2.3
iOS典型逆向开发流程	1.2.3.1
iOS逆向重点和难点	1.2.4
子教程	1.3
附录	1.4
参考资料	1.4.1

iOS逆向开发

- 最新版本: v1.9.0
- 更新时间: 20240305

简介

概述iOS逆向开发的各方面内容。包括整理了iOS逆向典型流程图，iOS逆向开发领域包含哪些内容的架构图，iOS的逆向开发的典型的流程，以及逆向的重点和难点。且把各个独立的部分整理出子教程，包括iPhone越狱、砸壳ipa、静态分析、动态调试、MonkeyDev调试，以及一些iOS逆向常涉及的领域，比如越狱插件开发、iOSOpenDev开发插件、越狱检测和反越狱检测、iOS底层机制，其中包括ObjC运行时和Block匿名函数；以及具体的实例，包括YouTube逆向、抖音逆向，以及YouTube中包含protobuf逆向；另外整理出相关的子教程，包括IDA、汇编asm、ARM汇编、Xcode开发、Xcode调试、LLDB调试器、iOS开发、C语言开发等内容。

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_reverse_dev: iOS逆向开发](#)

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 book.crifan.org](#)
- [iOS逆向开发 crifan.github.io](#)

离线下载阅读

- [iOS逆向开发 PDF](#)
- [iOS逆向开发 ePUB](#)
- [iOS逆向开发 Mobi](#)

版权和用途说明

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

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

鸣谢

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

其他

作者的其他电子书

本人 crifan 还写了其他 150+ 本电子书教程，感兴趣可移步至：

[crifan/crifan_ebook_readme: Crifan的电子书的使用说明](#)

关于作者

关于作者更多介绍，详见：

[关于CrifanLi李茂 – 在路上](#)

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2024-03-06 09:22:24

iOS逆向概述

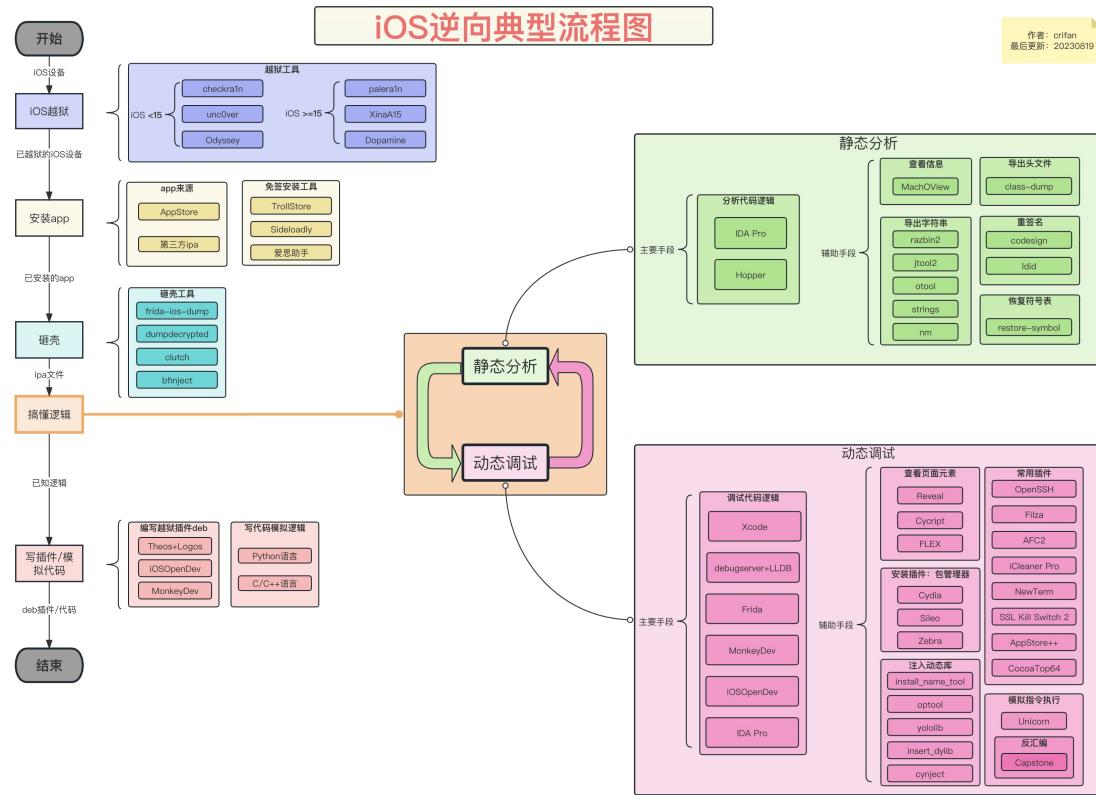
- 什么是iOS逆向
 - iOS开发，从 安全 领域的 攻防 角度， 可以分为：
 - 攻 : iOS逆向 = iOS破解
 - 防 : iOS安全 = iOS防护
- 为什么要iOS逆向
 - 为何要逆向
 - 从 需求 角度来说
 - 多数：灰产和黑产
 - 部分：正方的知己知彼百战不殆
 - 被破解的app厂商，从防的角度，要了解和实践黑灰产的攻击逆向破解技术，才能更好的提高防护技术
 - 偶尔：
 - 学习别人的技术，如何实现某些功能的方法和原理
 - 以及部分是为了做出功能一样但更好体验的产品
 - 技术极客技术上折腾好玩有趣的功能
 - 比如iPhone越狱后，写插件实现各种定制或破解功能，比如常见的微信抢红包、解锁VIP会员限制等等
 - 极个别：自己的程序，但丢了源码，要逆向破解找回大部分的代码
 - 从 学习技术 的角度来说
 - 搞懂逆向，技术更偏底层，更能学习到相对高深的底层技术
 - 包括但不限于：编译流程和原理、系统架构设计和实现细节等等
 - 此处：为何要iOS逆向
 - 需求角度：
 - 多数：为了黑灰产去赚钱，用iOS的iPhone去模拟各类app去刷流量等等
 - 部分：正方厂商防护自己的iOS的app之前要了解反方的逆向技术
 - 也有：是为了做出更好的替代品
 - 比如：YouTube的app，为了去广告和更好的体验，出现了替代品 `YouTube Vanced`，不过由于做的太好，以至于后来被google投诉而禁用了
 - 学习技术角度：能更加深刻学习和理解，iOS的ObjC和Swift语言的底层特性和实现机制，比如 `Block` 、 `Runtime` 运行时 等等

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-21 22:11:34

iOS逆向典型流程图

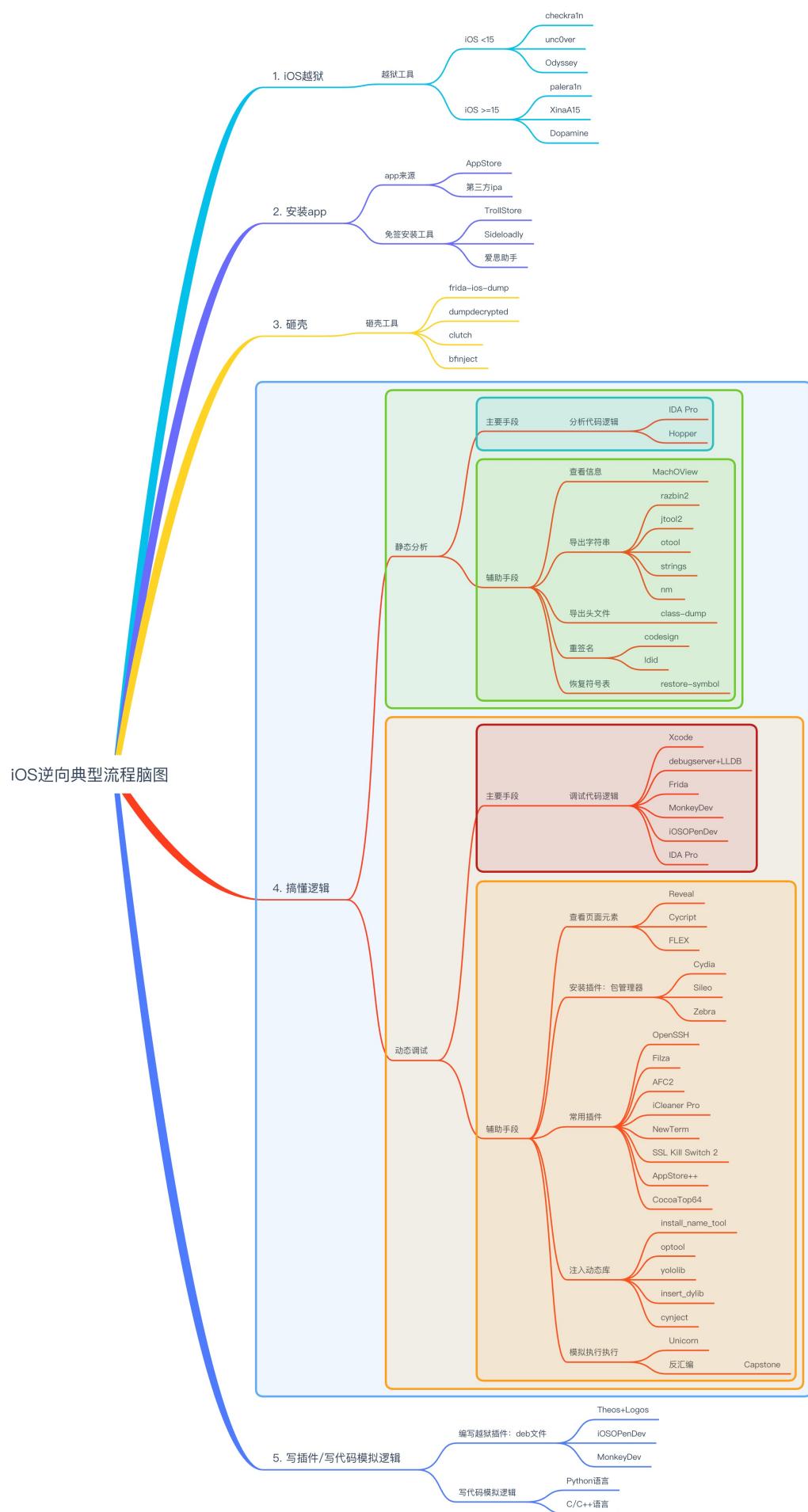
- iOS逆向典型流程图

- 在线浏览
 - [iOS逆向典型流程图| ProcessOn免费在线作图](#)
- 离线查看



- iOS逆向典型流程脑图

- 在线浏览
 - [iOS逆向典型流程脑图| ProcessOn免费在线作图](#)
- 离线查看



iOS逆向领域架构图

为了能对iOS逆向有个更全面和直观的了解，此处把iOS逆向开发所包含的各个领域的内容以及相互关系，整合到一张图：

iOS逆向领域架构图

- 在线浏览（支持缩放）
 - [iOS逆向开发内容架构图 | ProcessOn免费在线作图](#)
- 离线查看

◦

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 21:57:48

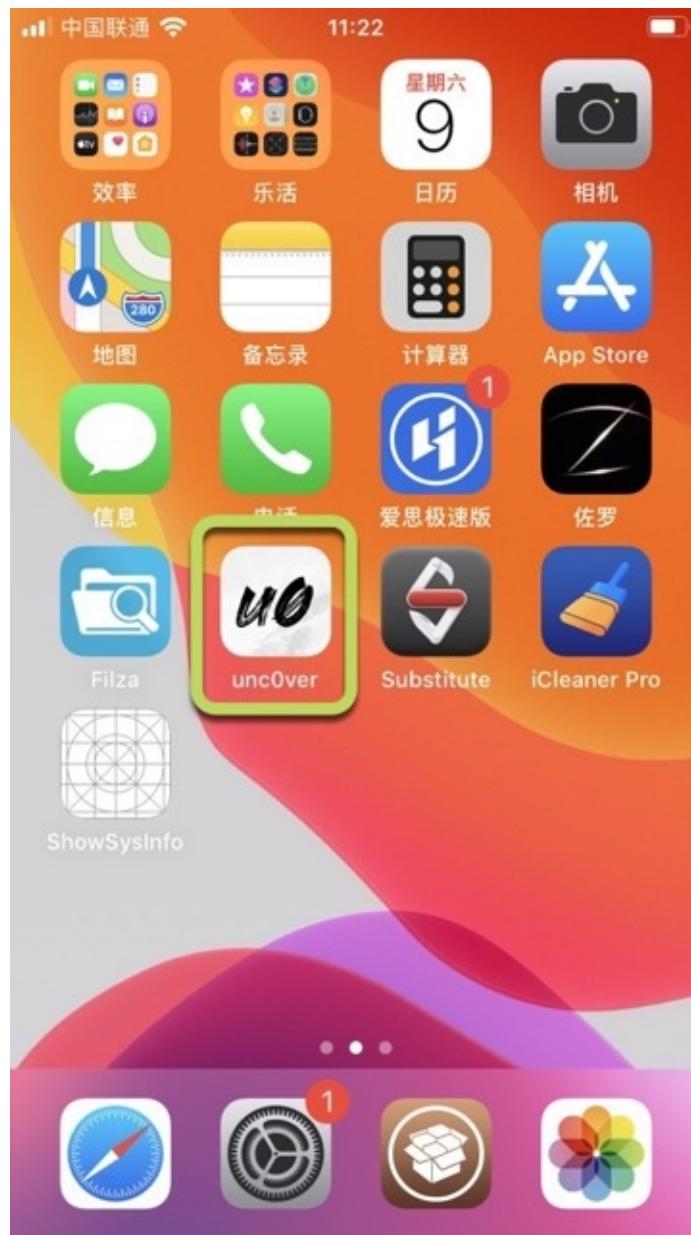
iOS逆向内容概述

- 先准备越狱iPhone

- 越狱工具

- unc0ver

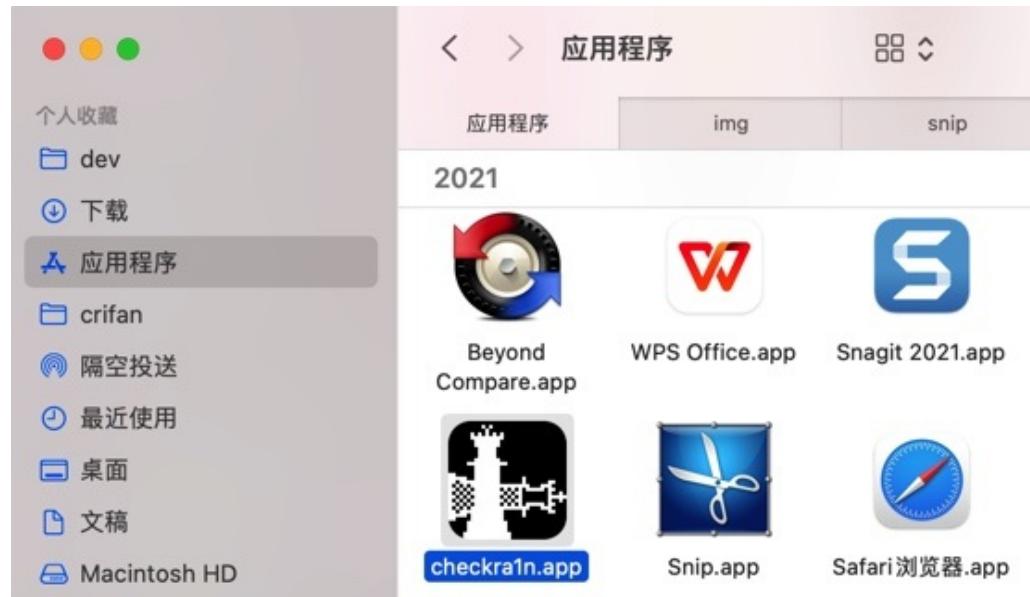
- 自己下载unc0ver到iPhone中



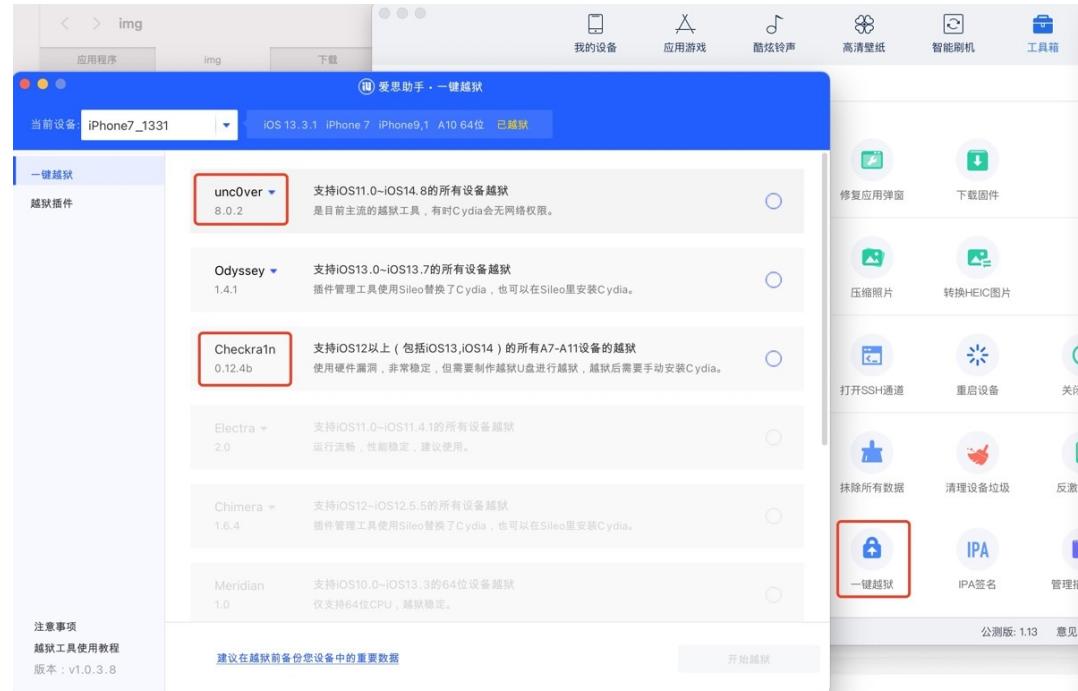
- 运行unc0ver去越狱

- checkra1n

- Mac中运行checkra1n，根据操作提示去越狱



- 也可以借助于 爱思助手 的 一键越狱 去越狱
 - 底层也是用unc0ver、checkra1n等工具



- 包括准备好常用越狱插件开发工具，比如
 - Filza : 文件管理
 - OpenSSH : ssh连接操作iPhone
 - AFC2 : 允许通过USB操作iPhone
 - iCleaner Pro : 临时禁止或启用插件
 - AppSync Unified : 免签名安装app
 - 等等
- 再从AppStore安装的正版app中砸壳出ipa

◦

○ 常用工具

■ frida-ios-dump

```
→ ~ iproxy 2222 22
Creating listening port 2222 for device port 22
waiting for connection
New connection for 2222->22, fd = 5
waiting for connection
Requesting connection to USB device handle 35 (serial: a
dab53e3250e8be1ee0db75bccdc2063df608b46), port 22
New connection for 2222->22, fd = 5
waiting for connection
Requesting connection to USB device handle 35 (serial: a
dab53e3250e8be1ee0db75bccdc2063df608b46), port 22
New connection for 2222->22, fd = 5
waiting for connection
Requesting connection to USB device handle 35 (serial: a
dab53e3250e8be1ee0db75bccdc2063df608b46), port 22
[frida-ios-dump]: Load widevine_cdm_secured_ios.framework success.
[frida-ios-dump]: Module_Framework.framework has been loaded.
start dump /private/var/containers/Bundle/Application/ECB295AB-1355-46D1-8580-273B2CE98802/
YouTube.app/YouTube
YouTube.fid: 100% [██████████] 16.3M/16.3M [00:00:00:00, 17.7MB/s]
start dump /private/var/containers/Bundle/Application/ECB295AB-1355-46D1-8580-273B2CE98802/
YouTube.app/Frameworks/widevine_cdm_secured_ios.framework/widevine_cdm_secured_ios
widevine_cdm_secured_ios.fid: 100% [██████████] 3.44M/3.44M [00:00:00:00, 24.2MB/s]
start dump /private/var/containers/Bundle/Application/ECB295AB-1355-46D1-8580-273B2CE98802/
YouTube.app/Frameworks/Module_Framework.framework/Module_Framework
Module_Framework.fid: 100% [██████████] 114M/114M [00:03:00:00, 36.6MB/s]
0.00B [00:00, 7B/s]
```

```
→ frida-ios-dump git:(master) ./dump.py com.google.ios.youtube
Start the target app com.google.ios.youtube
Dumping YouTube to /var/folders/2f/53mn2kn920dfq4ww2gdqfpvc0000gn/T
[frida-ios-dump]: Load widevine_cdm_secured_ios.framework success.
[frida-ios-dump]: Module_Framework.framework has been loaded.
start dump /private/var/containers/Bundle/Application/ECB295AB-1355-46D1-8580-273B2CE98802/
YouTube.app/YouTube
YouTube.fid: 100% [██████████] 16.3M/16.3M [00:00:00:00, 17.7MB/s]
start dump /private/var/containers/Bundle/Application/ECB295AB-1355-46D1-8580-273B2CE98802/
YouTube.app/Frameworks/widevine_cdm_secured_ios.framework/widevine_cdm_secured_ios
widevine_cdm_secured_ios.fid: 100% [██████████] 3.44M/3.44M [00:00:00:00, 24.2MB/s]
start dump /private/var/containers/Bundle/Application/ECB295AB-1355-46D1-8580-273B2CE98802/
YouTube.app/Frameworks/Module_Framework.framework/Module_Framework
Module_Framework.fid: 100% [██████████] 114M/114M [00:03:00:00, 36.6MB/s]
0.00B [00:00, 7B/s]
```

■ Clutch

■ dumpdecrypted

■ bfinject

● 接着才是逆向开发

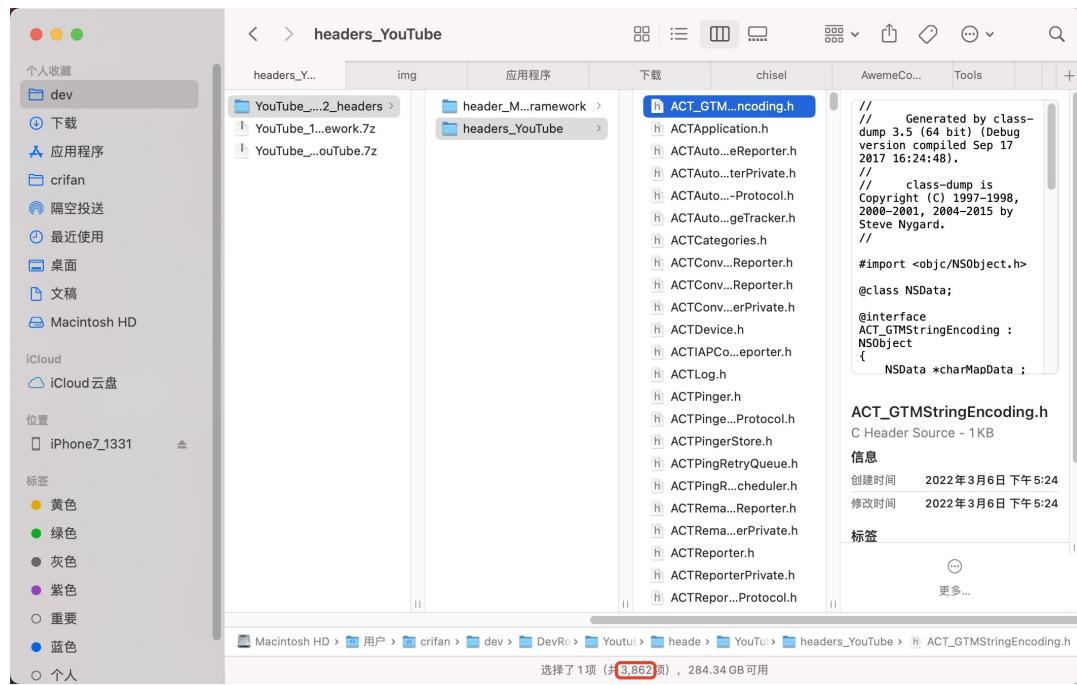
○ 主要分两类

- 静态分析
- 动态调试

○ 下面分别详细解释：

● 静态分析

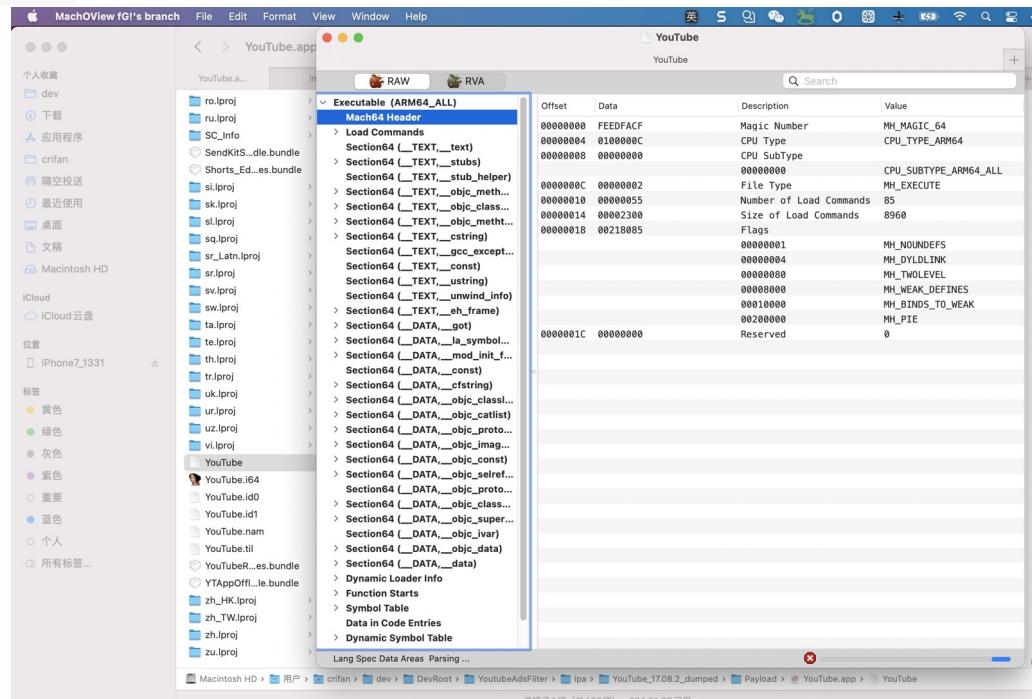
- 从脱壳ipa解压得到app包
 - 用 class-dump 导出头文件



■ 以搞懂包含哪些类

■ 查看二进制信息

■ MachOView

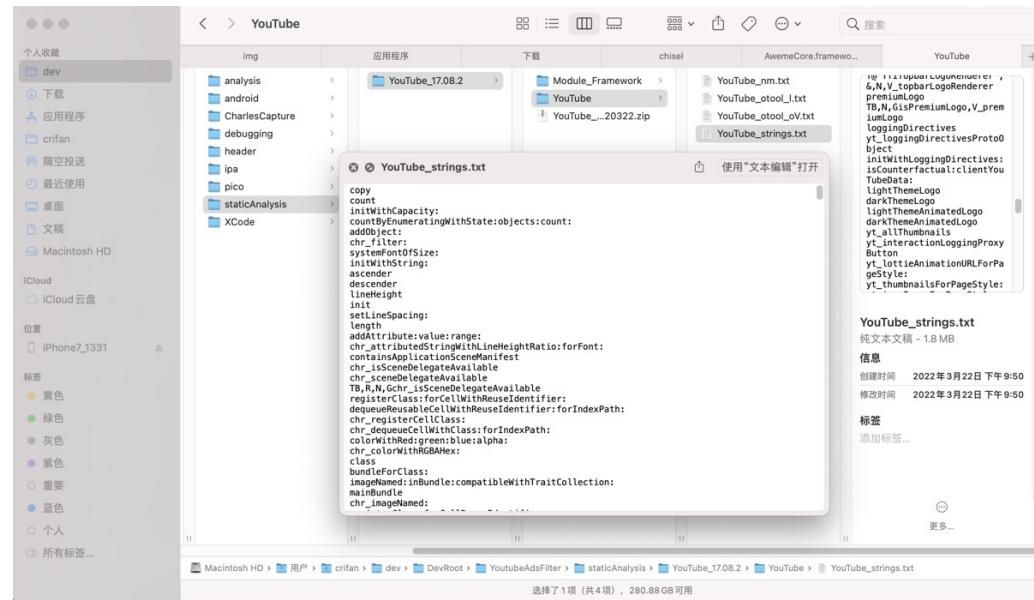


■ jtool2

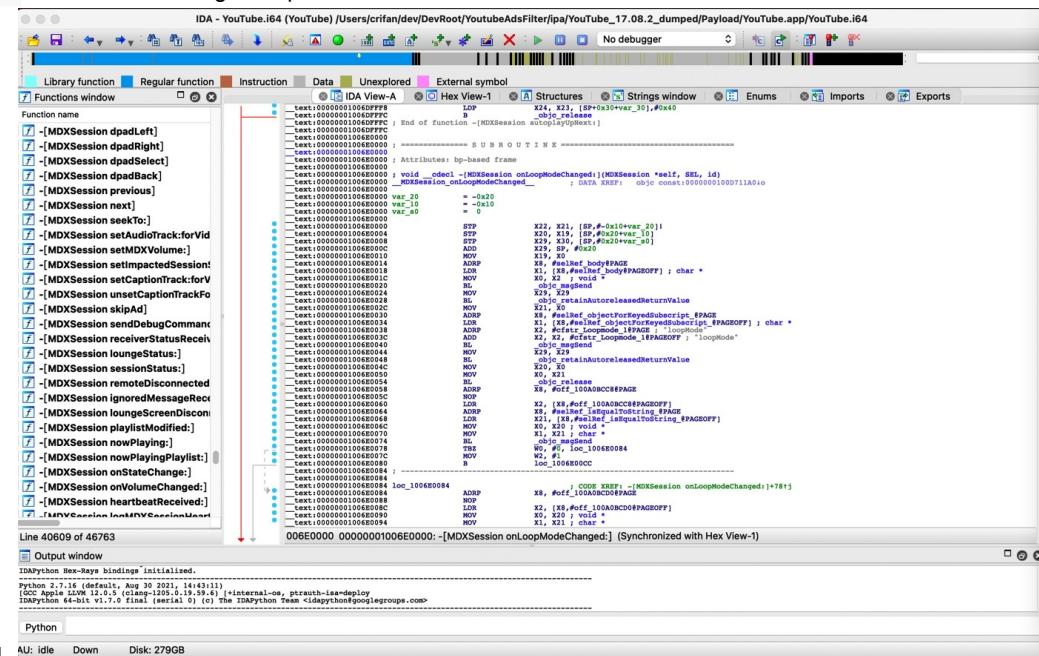
■ rabin2

■ 导出字符串等资源

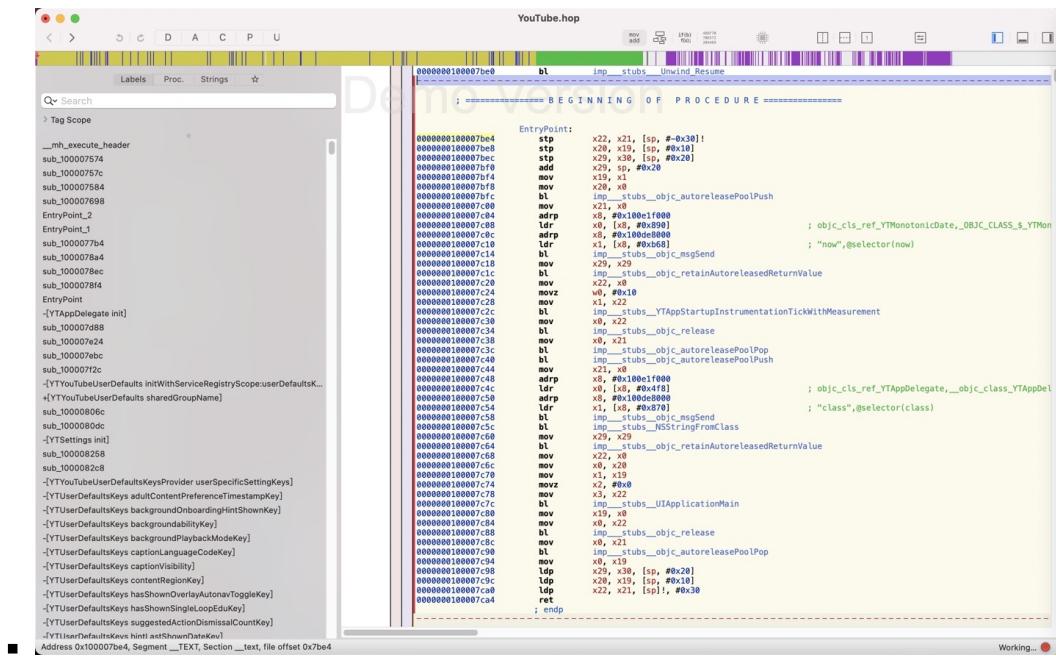
■ strings



- nm
- otool
- jtool2
- rabin2
- 恢复符号表
 - restore-symbol
- 分析代码逻辑
 - IDA : Functions、Strings、Imports、F5伪代码、导出全部伪代码等好用的功能和模块



■ Hopper



● 动态调试

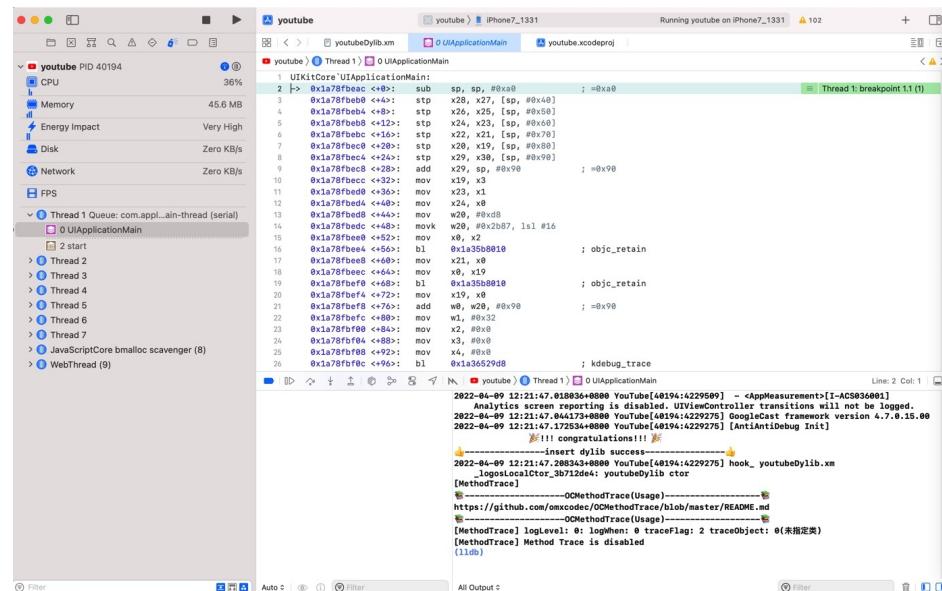
- 用各种调试方式和工具去调试app的逻辑

- 常用调试方式

- 图形界面

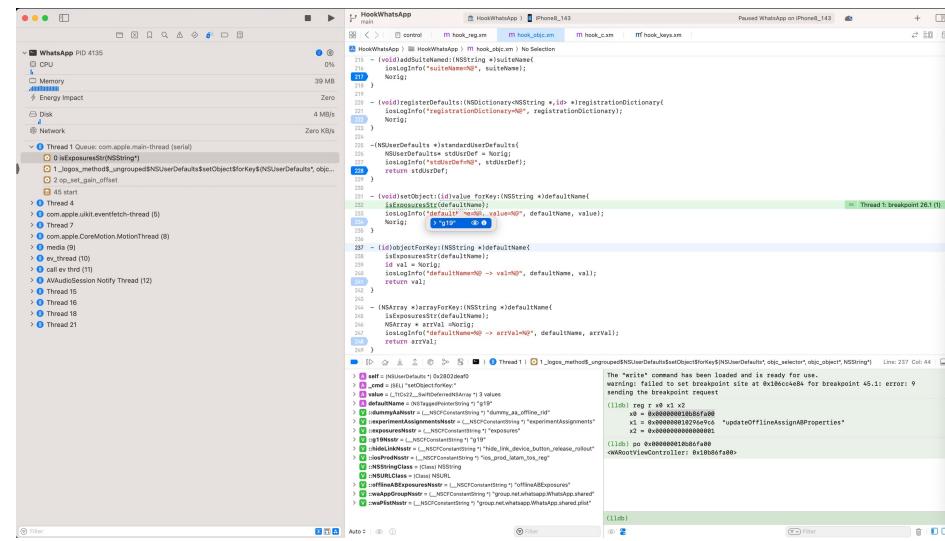
- 入门时可用: Xcode + MonkeyDev

- 效果



- 熟练时推荐: Xcode+iOSOpenDev

- 效果



■ 好处

- Xcode内部有（好用的）lldb
- 可以写hook代码hook函数
- Xcode+lldb内置对于iOS的ObjC（甚至Swift）有比较好的支持
 - 比如：po可以查看到类的信息等等
- 图形化界面方便调试

■ 命令行： debugserver + lldb

```

root@192.168.0.58 (ssh)
1 -0.000000 sec [27ac/0x303]: RNBRLoopLaunchAttaching Attaching to pid 10146...
2 -0.003597 sec [27ac/0x303]: MachProcess::MachProcess()
3 -0.000022 sec [27ac/0x303]: (DebugHub) attaching to pid 10146...
4 -0.000014 sec [27ac/0x303]: MachProcess::SetState(Unloaded) ignoring redundant state change...
5 -0.000023 sec [27ac/0x303]: MachProcess::SetState(Attaching) updating state (previous state was Unloaded), eventMask = 0x00000001
6 -0.000023 sec [27ac/0x303]: MachTask::StartExceptionThread()
7 -0.000035 sec [27ac/0x303]: ::task_for pid ( target_tport = 0x0103, pid = 10146, &task ) => err = 0x00000000 (success) err = 0x00000000
8 -0.000029 sec [27ac/0x303]: ::task_info { target_task = 0x1407, flavor = TASK_BASIC_INFO, task_info.out => 0x1d855fe8, task_info.outcnt => 10 } err = 0x00000000
9 -0.000021 sec [27ac/0x303]: task_basic_info = { suspend_count = 0, virtual_size = 0x151a3c000, resident_size = 0x17108000, user_time = 4.664964, system_time = 4.664964 }
10 -0.000028 sec [27ac/0x303]: MachException::PortInfo::Save ( task = 0x1407 )
11 -0.000025 sec [27ac/0x303]: ::task_get_exception_ports ( task = 0x1407, mask = 0x1bf, e, maskBit => 3, ports, behaviors, flavors ) err = 0x00000000
12 -0.000026 sec [27ac/0x303]: ::task_set_exception_ports ( task = 0x1407, exception_mask = 0x000001bf, new_port = 0x2903, behavior = 0x80000001, new_flavor = 0x00000005 ) e
rr = 0x00000000
13 -0.000023 sec [27ac/1073]: MachTask::ExceptionThread ( arg = 0x10400b668 ) starting thread...
iPhone7P:1341:~/forDebug root# pwd
/var/root/forDebug
iPhone7P:1341:~/forDebug root# debugserver -x auto 0.0.0.0:20221 /private/var/containers/Bundle/Application/9AB25481-0A03-435C-A02E-68F9623535B8/Aweme.app/Aweme
debugserver -q #PROGRAM:LLDB PROJECT:lldb-900.3.104
for arm64.
Listening to port 20221 for a connection from 0.0.0.0...
Got a connection, launched process /private/var/containers/Bundle/Application/9AB25481-0A03-435C-A02E-68F9623535B8/Aweme.app/Aweme (pid = 10174).
Exiting.
iPhone7P:1341:~/forDebug root# []

```

■ 常用逆向工具

- Frida : hook对应函数，调试输入参数和返回值

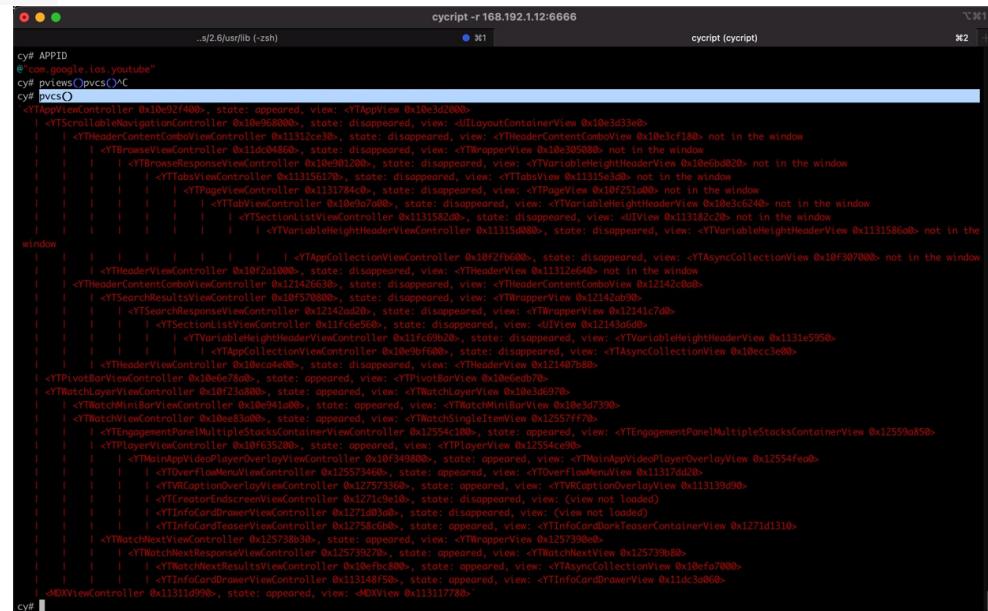
```

~ frida -U -f com.ss.iphone.ugc.Aweme
    _ _ |  Frida 15.1.14 - A world-class dynamic instrumentation toolkit
| _\_| |
> _ | Commands:
/_\_| help      -> Displays the help system
. . . object?   -> Display information about 'object'
. . . exit/quit -> Exit
. . .
. . . More info at https://frida.re/docs/home/
Spawned 'com.ss.iphone.ugc.Aweme'. Use %resume to let the main thread start executing!
stdout: obj[20744]: Class PodsDummy_EffectPlatformSDK is implemented in both /private/var/containers/Bundle/Application/0C35E4DD-5B68-40DE-BB14-1F1EE9266003/Aweme.app/Frameworks/bytaudio.framework/bytaudio (0x104f77f90) and /private/var/containers/Bundle/Application/0C35E4DD-5B68-40DE-BB14-1F1EE9266003/Aweme.app/Frameworks/VoLcEngineRTC.framework/VoLcEngineRTC (0x11827cb10). One of the two will be used. Which one is undefined.
stdout: obj[20744]: Class IESEAlgorithmModelCleaner is implemented in both /private/var/containers/Bundle/Application/0C35E4DD-5B68-40DE-BB14-1F1EE9266003/Aweme.app/Frameworks/bytaudio.framework/bytaudio (0x104f77f90) and /private/var/containers/Bundle/Application/0C35E4DD-5B68-40DE-BB14-1F1EE9266003/Aweme.app/Frameworks/VoLcEngineRTC.framework/VoLcEngineRTC (0x11827cb10). One of the two will be used. Which one is undefined.
stdout: obj[20744]: Class IESEAlgorithmModelConfig is implemented in both /private/var/containers/Bundle/Application/0C35E4DD-5B68-40DE-BB14-1F1EE9266003/Aweme.app/Frameworks/bytaudio.framework/bytaudio (0x104f78008) and /private/var/containers/Bundle/Application/0C35E4DD-5B68-40DE-BB14-1F1EE9266003/Aweme.app/Frameworks/VoLcEngineRTC.framework/VoLcEngineRTC (0x11827cb8). One of the two will be used. Which one is undefined.
stdout: obj[20744]: Class IESEAlgorithmModelDownloadQueue is implemented in both /private/var/containers/Bundle/Application/0C35E4DD-5B68-40DE-BB14-1F1EE9266003/Aweme.app/Frameworks/bytaudio.framework/bytaudio (0x104f78058) and /private/var/containers/Bundle/Application/0C35E4DD-5B68-40DE-BB14-1F1EE9266003/Aweme.app/Frameworks/VoLcEngineRTC.framework/VoLcEngineRTC (0x11827cc28). One of the two will be used. Which one is undefined.
stdout: obj[20744]: Class IESEAlgorithmModelManager is implemented in both /private/var/containers/Bundle/Application/0C35E4DD-5B68-40DE-BB14-1F1EE9266003/Aweme.app/Frameworks/bytaudio.framework/bytaudio (0x104f780f8) and /private/var/containers/Bundle/Application/0C35E4DD-5B68-40DE-BB14-1F1EE9266003/Aweme.app/Frameworks/VoLcEngineRTC.framework/VoLcEngineRTC (0x11827cc8).
■

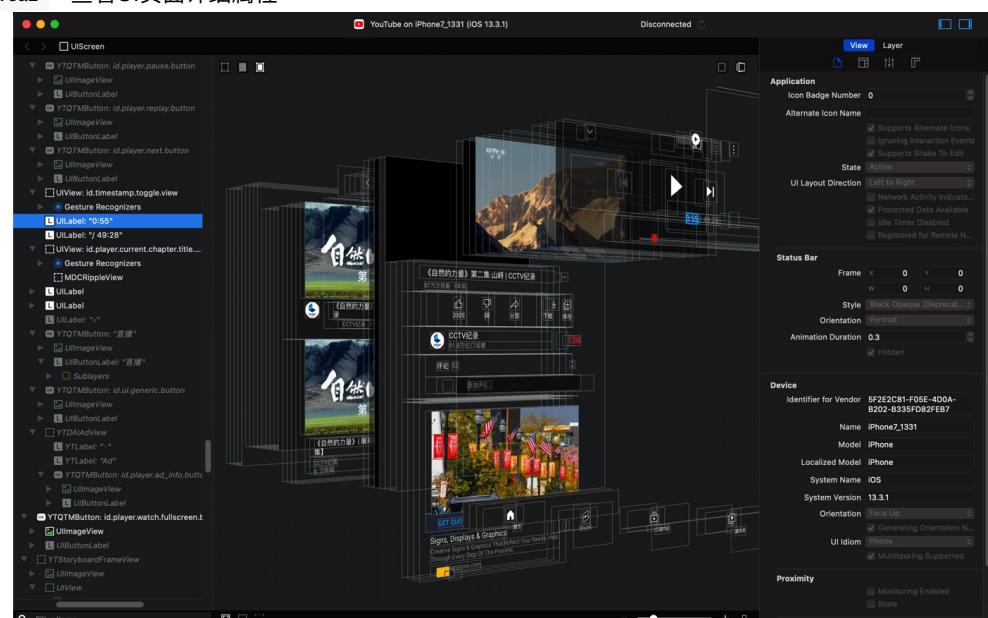
```

分析界面元素

- Crypt : 查看页面元素的类和属性、实时修改各种值



- Reveal : 查看UI页面详细属性



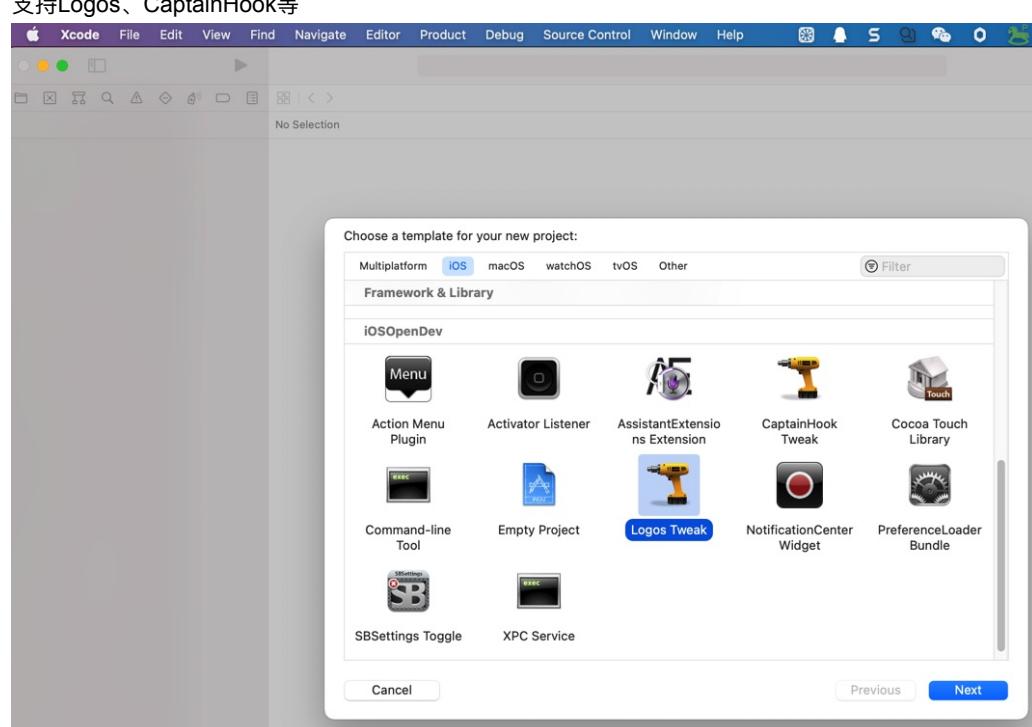
- 目的：搞懂我们所关心的app内部相关逻辑
 - 用于后续去写hook代码去修改成我们要的逻辑和值
- 最后才是：Tweak插件开发
 - 常见插件开发框架
 - 基于 CydiaSubstrate
 - Theos / Logos
 - fishhook
 - CaptainHook
 - 具体开发方式
 - 命令行
 - Theos / Logos : 写hook源码.xm, 基于Makefile去编译

```

Tweak.x -- altsys
Tweak.x - altsys
TweakX
1 // #import <SpringBoard/SpringBoard.h>
2 // #import <Preferences/Preferences.h>
3 #import <sys/log.h>
4
5 #import <CoreTelephony/CTTelephonyNetworkInfo.h>
6 #import <CoreTelephony/CTCarrier.h>
7
8 #import <sys/types.h>
9 #import <sys/sysctl.h>
10 #import <sys/errno.h>
11
12 int sysctlbyname(const char *name, void *oldp, size_t *oldlenp, void *newp, size_t newlen);
13
14 hookf(int, sysctlbyname, const char *name, void *oldp, size_t *oldlenp, void *newp, size_t newlen);
15 os_log(OS_LOG_DEFAULT, "AltSys hook sysctlbyname: name=%s", name);
16 NSString *hwMachineKey = @"hw.machine";
17 os_log(OS_LOG_DEFAULT, "sysctlbyname: hwMachineKey=%@", hwMachineKey);
18
19 // NSString *retValue = @"";
20 NSString *name = [NSString stringWithFormat:@"%@", name];
21
22 // NSString *fakeModel = @"Fake_hw.machine";
23 // const char fakeModel[] = "Fake_hw.machine";
24 // const char fakeModel[] = "iPhone14,5"; // emulate 'iPhone 13'
25 // const char fakeModel = "Fake_hw.machine";
26 const char *fakeModel = "iPhone14,5"; // emulate 'iPhone 13'
27 // NSString *fakeModel = [NSString stringWithFormat:@"%@", fakeModel];
28 os_log(OS_LOG_DEFAULT, "sysctlbyname: fakeModel=%s", fakeModel);
29
30 // if (ndName == hwMachineKey)
31
+ altsys git:(master) ✘ make do
swift-driver version: 1.26.21 => Notice: Build may be slow as Theos isn't using all available CPU cores on this computer. Consider upgrading GNU Make: https://github.com/theos/wiki/Parallel-Build
> Making stage for tweak AltSys...
swift-driver version: 1.26.21 swift-driver version: 1.26.21 make[2]: Nothing to be done for 'internal-library-compile'.
> Making stage for tweak AltSys...
swift-driver version: 1.26.21 dm.pl: building package 'com.crifan.altsys:iphones-arm' in './packages/com.crifan.altsys_0.1.5-5+debug_iphones-arm.deb'
-- Installing...

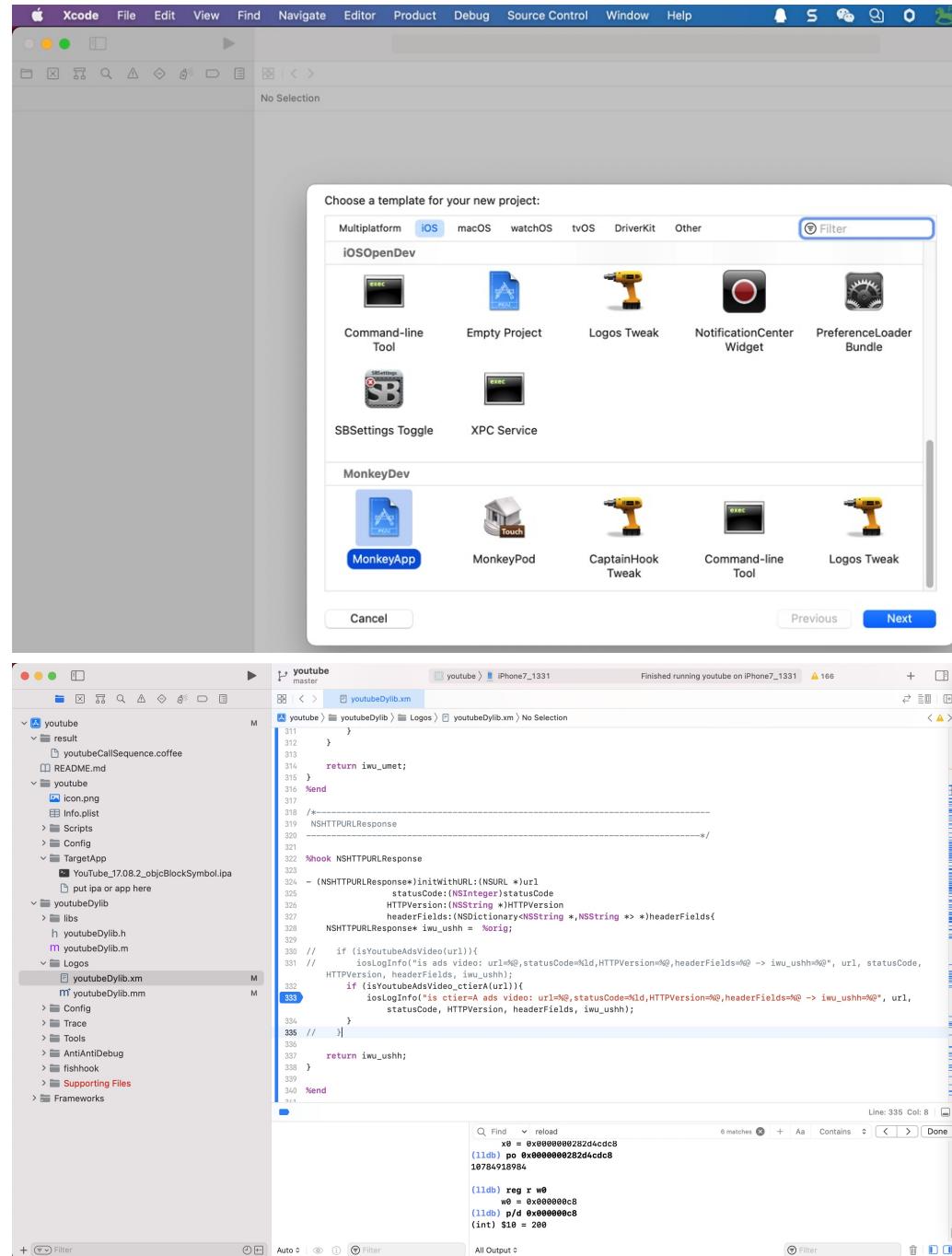
```

- 图形界面
 - iOSOpenDev : 把Logos等开发集成进了Xcode
 - 支持Logos、CaptainHook等



- MonkeyDev : iOSOpenDev的改进版

- 支持Logos、CaptainHook等



- 去开发插件=写hook代码

- 前提：通过静态分析的头文件和动态调试，已知的app内部的类的属性和函数

- 核心逻辑：去hook对应类的函数和属性

- 实现对应的效果，比如：

- 调试：输出函数的输入参数和输出结果
- 修改逻辑：屏蔽原先逻辑，重写自己想要的逻辑

- 开发出的插件常用于

- 逆向破解特定app

- 绕过ssl证书校验，实现Charles抓包https可看到明文数据

- 修改app原有逻辑，实现特定的功能

- 支付宝：修改显示的余额
- 微信：抢红包
- 抖音：点赞关注

- 反反调试

- 反越狱检测

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2024-03-05 22:40:06

iOS典型逆向开发流程

典型的iOS逆向开发的流程是：

- 先：iPhone越狱
 - 越狱工具
 - unc0ver
 - checkra1n
- 再：逆向破解某iOS的app
 - 对于要研究的某个iOS的app来说，从工作内容角度，主要分：
 - 从AppStore中搜索和安装正版app
 - ipa
 - 砸壳得到ipa
 - frida-ios-dump
 - 或者直接从网上搜索某版本的ipa
 - 再去研究
 - 静态分析
 - 字符串分析
 - strings
 - nm
 - otool
 - 查看详情
 - MachOView
 - jtool2
 - rabin2
 - 导出头文件
 - class-dump
 - 恢复符号表
 - restore-symbol
 - 分析代码逻辑
 - IDA
 - 汇编代码
 - 伪代码
 - 动态调试
 - 调试代码逻辑
 - 命令行方式
 - debugserver + lldb
 - GUI图形界面方式
 - Xcode + MonkeyDev
 - lldb
 - 插件
 - chisel
 - 其他调试工具
 - frida
 - 分析界面元素
 - Reveal
 - Cycript
 - 再去调试和验证
 - 写hook插件
 - iOSOpenDev
 - 加断点调试，验证代码是否运行到

- 如此反复静态分析和动态调试的过程
- 最后结果：得到一个插件tweak，实现了你要的目的
 - 比如反越狱检测、改机、hook某个app的某些特定功能（比如微信抢红包）等等

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2022-11-06 16:41:03

iOS逆向的重点和难点

对于iOS逆向开发涉及到众多方面的内容，其中属于重点和难点的是：

- 先搞懂你想要干啥
 - 比如 微信抢红包
- 重点和难点
 - 用静态分析工具和动态调试，共同配合，找出要hook的类
 - 如何用好各种静态分析工具（[MachOView](#)、[rabin2](#)、[jtoo2](#)、[otool](#)、[IDA](#)、[Hopper](#)等），如何进行动态调试（[Xcode+MonkeyDev](#)、[debugserver+lldb](#)、[frida](#)、[Cycrypt](#)、[Reveal](#)等），以及利用好已有的[class-dump](#)导出的头文件，从而找到要分析的类和代码运行逻辑等等，才是重点和难点
 - 再去写hook代码，开发出插件，实现对应的功能

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2023-10-08 10:33:06

子教程

上述所有和iOS逆向开发相关的内容，分别整理到多个独立的子教程以及相关内容：

- iOS逆向的成套教程 = 系列教程 = 子教程
 - iOS逆向基本流程
 - iOS逆向之前，先要给iOS设备越狱
 - [iOS逆向开发：iPhone越狱](#)
 - 如果是 iOS 15+，则需要最新的越狱工具
 - [iOS逆向：iOS15越狱](#)
 - 越狱后，常会涉及到
 - [iOS逆向开发：越狱包管理器](#)
 - [iOS逆向：安装app和插件](#)
 - 然后再去砸壳出ipa文件
 - [iOS逆向开发：砸壳ipa](#)
 - 然后再去静态分析
 - [iOS逆向开发：静态分析](#)
 - 常用静态分析工具
 - [逆向利器：IDA](#)
 - [iOS逆向工具：Hopper](#)
 - 以及动态调试
 - [iOS逆向开发：动态调试](#)
 - [iOS逆向开发：MonkeyDev调试](#)
 - [逆向调试利器：Frida](#)
 - 移动端调试利器：Objection
 - [主流调试器：LLDB](#)
 - [iOS逆向调试：debugserver+lldb](#)
 - Xcode
 - [XCode开发心得](#)
 - [Xcode开发：调试心得](#)
 - [推荐：iOS逆向调试：Xcode+iOSOpenDev](#)
- iOS逆向常涉及领域/子项目
 - 插件
 - 常用插件
 - [iOS越狱开发：常用越狱插件](#)
 - 插件开发：如何用 Theos / iOSOpenDev / MonkeyDev 开发越狱插件，实现特定功能
 - [iOS逆向开发：越狱插件开发](#)
 - [iOS逆向开发：iOSOpenDev开发插件](#)
 - 符号表
 - [iOS逆向分析：恢复符号表](#)
 - iOS底层机制和原理
 - [iOS逆向开发：iOS底层机制](#)
 - [iOS逆向开发：ObjC运行时](#)
 - [iOS逆向开发：Block匿名函数](#)
 - 正向的越狱检测和逆向的反越狱检测
 - [iOS逆向开发：越狱检测和反越狱检测](#)
 - 签名和权限
 - [iOS逆向开发：签名和权限](#)
 - 断点
 - [iOS逆向之动态调试：断点](#)
 - 文件格式

- 可执行文件格式
 - 可执行文件格式: Mach-O
- 心得
 - iOS逆向: 心得集锦
- 举例
 - iOS逆向: 举例详解
- 高级逆向
 - 代码模拟
 - CPU模拟利器: Unicorn
 - 反汇编利器: Capstone
 - Swift
 - iOS逆向: Swift逆向
- iOS逆向具体实例
 - iOS逆向开发: YouTube逆向
 - iOS逆向YouTube: protobuf逆向
 - 【整理Book】iOS逆向开发: 抖音逆向
- 相关
 - 语言
 - 汇编
 - 通用
 - 【整理Book】底层编程语言: 汇编语言asm
 - ARM
 - 最流行汇编语言: ARM crifan.org
 - C
 - C语言开发心得
 - iOS
 - iOS开发心得
 - 代码
 - iOS正向
 - 越狱检测
 - crifan/iOSJailbreakDetection: iOS的ObjC的app, 实现iOS越狱检测
 - 配合测试(iOS逆向改机)的app
 - crifan/ShowSystemInfo: iOS的app, 检测并显示iOS的iPhone的系统信息
 - iOS逆向
 - 反越狱检测
 - crifan/iOSBypassJailbreak: 越狱iOS的hook插件, 实现反越狱检测
 - 逆向YouTube
 - crifan/iOSYouTubeAdsFilter: MonkeyDev+Xcode项目, iOS逆向YouTube, 尝试实现广告过滤功能

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

- [crifan \(Crifan Li\)](#)
- [What is Reverse-engineering? How Does It Work?](#)
- [iOS安全-看雪论坛-安全社区|安全招聘|bbs.pediy.com](#)
- [iOS 的系统安全性比Android 系统要高! // 解读iOS安全机制 - 知乎 \(zhihu.com\)](#)
- [iOS安全杂谈 \(inforsec.org\)](#)
- [iOS应用安全开发概述 · 唐巧的博客 \(devtang.com\)](#)
- [安卓和iOS谁更安全 2021年比较报告 \(ganbey.com\)](#)
- [iOS开发安全-InfoQ](#)
- [不想iPhone被黑？赶紧试试这个-iPhone,被黑,Apple ID,验证, ——快科技\(驱动之家旗下媒体\)--科技改变未来 \(mydrivers.com\)](#)
- [iPhone可以被黑客入侵吗？如果是，您必须采取什么行动？ — iStarTips](#)
- [Low Level Virtual Machine \(LLVM\)](#)
- [LLDB \(debugger\) - Wikipedia](#)
- [LLDB Homepage — The LLDB Debugger](#)
- [Tutorial — The LLDB Debugger](#)
- [深入了解GDB和LLDB - 简书](#)
- [LLDB 知多少 - 掘金](#)
- [iOS \(十六\) 一次通过lldb绕过越狱检测&反反调试实践 ~ gandalf](#)
- [debugserver - iPhone Development Wiki](#)
- [使用lldb+debugserver动态调试iOS应用 | LaOs](#)
- [iOS 逆向指南：动态分析 – 小专栏](#)
- [iOS逆向, 基础工具之LLDB和debugserver - 简书](#)
- [一步一步用debugserver + lldb代替gdb进行动态调试 - Blog | 干货分享 - iOSRE](#)
- [Remote Debugging — The LLDB Debugger](#)
- [OLLVM代码混淆移植与使用 | Heroims的博客](#)
- [利用ollvm进行代码混淆 | m4bln](#)
- [Macho文件浏览器---MachOView - 简书](#)
- [iOS \(十四\) 高版本越狱的坑 & killed 9 ~ gandalf](#)
- [macos - Editing assembly on Mac OS X - Stack Overflow](#)
- [iOS逆向 \(八\) 逆向工具 otool 介绍 - 掘金](#)
- [otool 一些用途 - 简书](#)
- [otool命令查看App动态库 - 简书](#)
- [iOS 逆向---otool命令入门_嵌入式_ParadiseDuo-CSDN博客](#)
- [iOS App的加固保护原理 - 知乎](#)
- [iOS App 安全加固方案调研 - iOS - 掘金](#)
- [对 iOS app 进行安全加固 - 我的学习历程](#)
- [iOS 应用加固方法 - 简书](#)
- [为了保护公司的 App 安全, 我用遍了市面上的加固产品 - V2EX](#)
- [iOS逆向工程 介绍 | iOS 安全 Wiki](#)
- [iOS \(十五\) 几种App砸壳工具对比 ~ gandalf](#)
- [iOS逆向工具之砸壳工具\(MacOS&iOS\)介绍 - 简书](#)
- [iOS攻防（四）：使用Dumpdecrypted 砸壳 & class-dump 导出头文件 | 曹雪松de博客|CoderBoy's Blog](#)
- [iOS攻防（六）：使用Cyclicrypt一窥运行程序的神秘面纱\(入门篇\) | 曹雪松de博客|CoderBoy's Blog](#)
- [class-dump的安装和使用 - 简书](#)
- [iOS攻防——（四）class-dump 与 Dumpdecrypted 使用 | 周小鱼のCODE_HOME](#)
- [class-dump的安装和使用 - 简书](#)
- [class dump使用方式和原理 - 简书](#)
- [Class-dump: class-dump-x和class-dump-z如何分析dylib等文件？ - Discussion | 技术讨论 - iOSRE](#)
- [iOS逆向之旅（进阶篇） — 工具\(class-dump\) - 掘金](#)

- [iOS逆向之class-dump - LeeGof - 博客园](#)
- [Tutorial · KJCracks/Clutch Wiki](#)
- [iOS攻防 - \(六\) iOS应用使用Clutch脱壳_移动开发_VictorZhang-CSDN博客](#)
- [iOS逆向工程之Clutch砸壳\(图文多\) - 简书](#)
- [iOS逆向之Clutch砸壳 - 简书](#)
- [iOS \(十五\) 几种App砸壳工具对比 ~ gandalf](#)
- [sqlcipher/sqlcipher: SQLCipher is an SQLite extension that provides 256 bit AES encryption of database files.](#)
- [iOS安全攻与防\(总篇\) - 简书](#)
- [tianjifou/iOS-security-attack-and-prevent: iOS安全攻与防,详细的列出了, 在iOS开发中, 项目会存在的安全漏洞以及解决办法。](#)
- [《iOS 应用逆向与安全》读后感 - 掘金](#)
- [iOS \(十四\) 高版本越狱的坑 & killed 9 ~ gandalf](#)
-

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2023-10-08 10:22:48