目录

iOS逆向开发：iOS底层机制

- 最新版本：`v1.3.0`
- 更新时间：`20241026`

简介

介绍iOS逆向开发期间涉及到iOS底层机制方面的，主要是ObjC方面的内容。包括Block回调、ObjC的Runtime运行时、Apple苹果相关开发资料。以及一些逆向的动态调试中涉及到iOS底层机制的相关心得。

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_ios_internal: iOS逆向开发：iOS底层机制

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：iOS底层机制 book.crifan.org
- iOS逆向开发：iOS底层机制 crifan.github.io

离线下载阅读

- iOS逆向开发：iOS底层机制 PDF
- iOS逆向开发：iOS底层机制 ePub
- iOS逆向开发：iOS底层机制 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

iOS底层机制概览

iOS逆向中，尤其是动态调试期间，往往会涉及到iOS的，尤其是ObjC的，的底层机制和实现原理。

此处整理相关的内容。

Apple苹果相关开发资料

TODO：

- 【已解决】iOS中st_size的off_t是什么类型
- 【已解决】iOS或Linux或C中pid_t的定义

此处整理，Apple苹果的，和iOS逆向相关的，尤其是涉及到iOS底层机制方面的，开发资料。

- `Apple` = `苹果`
  - 相关开发资料
    - 源码 源代码
    - 官网文档

Apple苹果官网

源码+文档

概述=总入口：

Open Source - Apple Developer

详解：

- 开源代码Open Source Projects
  - Apple Open Source
    - https://opensource.apple.com
      - 文档
        - Kernel
          - https://developer.apple.com/library/mac/#documentation/Darwin/Conceptual/KernelProgramming/build/build.html
        - Frameworks
          - https://developer.apple.com/library/mac/#documentation/MacOSX/Conceptual/OSX_Technology_Overview/SystemFrameworks/SystemFrameworks.html
        - Security
          - https://developer.apple.com/library/archive/documentation/Security/Conceptual/Security_Overview/Introduction/Introduction.html
      - 源码
        - 总入口
          - https://opensource.apple.com/releases/
        - 离线下载 源码 总入口
          - https://opensource.apple.com/tarballs/
          - 子模块
            - ObjC Runtime
              - https://opensource.apple.com/tarballs/objc4/
            - xnu
              - https://opensource.apple.com/tarballs/xnu/
            - dyld
              - https://opensource.apple.com/tarballs/dyld/
            - cctools
              - https://opensource.apple.com/tarballs/cctools/
        - 在线浏览 源码 总入口

- https://opensource.apple.com/source/
- 子模块
  - xnu
    - https://opensource.apple.com/source/xnu/
    - kern
      - sysctl
        - https://opensource.apple.com/source/xnu/xnu-792/bsd/kern/kern_sysctl.c.auto.html
  - dlyd
    - https://opensource.apple.com/source/dyld/
  - system_cmds
    - https://opensource.apple.com/source/system_cmds/
    - sysctl
      - system_cmds-880.60.2
        - https://opensource.apple.com/source/system_cmds/system_cmds-880.60.2/sysctl.tproj/
          - sysctl的源码
            - https://opensource.apple.com/source/system_cmds/system_cmds-880.60.2/sysctl.tproj/sysctl.c.auto.html
  - Libc
    - posix_spawn
      - https://opensource.apple.com/source/Libc/Libc-825.25/sys/posix_spawn.c.auto.html
  - objc4 = Objc
    - https://opensource.apple.com/source/objc4/
    - 不同版本
      - https://opensource.apple.com/source/objc4/objc4-818.2/
      - https://opensource.apple.com/source/objc4/objc4-532
      - https://opensource.apple.com/source/objc4/objc4-646
    - 子模块
      - runtime
        - objc_release
          - https://opensource.apple.com/source/objc4/objc4-532/runtime/NSObject.mm.auto.html
        - objc_alloc
          - https://opensource.apple.com/source/objc4/objc4-646/runtime/objc-internal.h
        - objc_msgSendSuper2
          - https://opensource.apple.com/source/objc4/objc4-532/runtime/objc-abi.h.auto.html
        - objc_retainBlock
          - https://opensource.apple.com/source/objc4/objc4-493.9/runtime/objc-arr.mm.auto.html
  - libpthread
    - pthread_get_stackaddr_np
      - https://opensource.apple.com/source/libpthread/libpthread-105.10.1/src/pthread.c.auto.html

- MacOS Forge
  - www.macosforge.org

函数和命令的文档

注：`man` = `manual` = `手册`

- Apple文档总入口：Documentation Archive (apple.com)
  - 函数 man手册 总入口：API Reference: iOS Manual Pages (apple.com)
    - 分很多大类
      - Section 2: system calls and error numbers
        - Section 2 of the manual contains documentation on UNIX system calls, error codes, and C library routines that wrap system calls. Most of these functions are described in headers that reside in /usr/include/sys.
        - For a detailed introduction, see intro(2)
          - Mac OS X Manual Page For intro(2) (apple.com)
      - Section 3: C libraries
        - Section 3 of the manual contains documentation on C library routines. This section excludes library routines that merely wrap UNIX system calls. Most of these functions are described in headers that reside in /usr/include or subdirectories therein.
        - For a detailed introduction, see intro(3)
          - Mac OS X Manual Page For intro(3) (apple.com)
      - Section 3cc: 加密 解密 算法 相关
        - API Reference: iOS Manual Pages (apple.com)
      - Section 3ssl
        - Section 3ssl of the manual contains documentation on OpenSSL library routines. These functions are described in headers that reside in /usr/include/openssl, and are split between the libssl and libcrypto libraries
        - For a detailed introduction, see crypto(3) and ssl(3)
          - Mac OS X Manual Page For crypto(3ssl) (apple.com)
            - crypto - OpenSSL cryptographic library
          - Mac OS X Manual Page For ssl(3ssl) (apple.com)
            - SSL - OpenSSL SSL/TLS library
      - Section 3x
        - Section 3x of the manual contains documentation on curses-related library routines used for general formatting on text terminals. These functions are described in headers that reside in /usr/include, and are located in the libncurses library.
        - For a detailed introduction, see ncurses(3)
          - Mac OS X Manual Page For ncurses(3x) (apple.com)
            - ncurses - CRT screen handling and optimization package
      - Section 5
        - Section 5 of the manual contains documentation on file formats and conventions. It includes documentation about file-system data structures, information about configuration files for various daemons, and information about data structures used in various binary and text file formats used by various parts of the operating system.
        - For a detailed introduction, see intro(5).
          - Mac OS X Manual Page For manpages(5) (apple.com)
            - manpages -- An introduction to manual pages

log日志的字符串格式化参数语法

String Format Specifiers (apple.com)

The Apple Wiki == The iPhone Wiki

- 旧
  - https://theiphonewiki.com/
- 新
  - https://theapplewiki.com/

syscall

- 新：Kernel Syscalls - The Apple Wiki
  - 旧：Kernel Syscalls - The iPhone Wiki

hw.machine 机型 映射

- Models - The iPhone Wiki

其他来源

内核原理和机制

OS Internals: (newosxbook.com)

->

- MAC OS X Internals: A Systems Approach: Singh, Amit: 9780321278548: Books: Amazon.com
- J's Entitlement DataBase (newosxbook.com)
  - OS X/iOS Entitlement Database - v0.8
  - As compiled by Jonathan Levin, @Morpheus__
  - Now with entitlements from iOS 9.0.2 through 15.2, MacOS 11.4 through 15.3

iOS逆向英文教程

据说是第一本英文书的专门详细介绍iOS逆向的书：

iosre/iOSAppReverseEngineering: The world’s 1st book of very detailed iOS App reverse engineering skills :) (github.com)

->

https://github.com/iosre/iOSAppReverseEngineering/blob/master/iOSAppReverseEngineering.pdf

有需要可以学习和参考。

iOS逆向常涉及内容

在iOS逆向期间，常涉及到很多Apple苹果相关的开发资料，整理如下供参考。

ObjC的类

UIDevice

UIDevice | Apple Developer Documentation

iOS中的基本类型的定义

关于iOS中的很多相关的底层的类型：

- __darwin_mode_t
- __darwin_off_t
- __darwin_pid_t

的定义是：

```
typedef __uint16_t   __darwin_mode_t;        /* [???] Some file attributes */
typedef __int64_t       __darwin_off_t;         /* [???] Used for file sizes */
typedef __int32_t       __darwin_pid_t;         /* [???] process and group IDs */
```

来源：

- Apple的opensource
  - https://opensource.apple.com/source/xnu/xnu-792/bsd/sys/_types.h
- 其他
  - apple/darwin-xnu: The Darwin Kernel (mirror)
    - https://github.com/apple/darwin-xnu/blob/main/bsd/sys/_types.h

头文件 errno.h

- 旧版本：`xnu-201`
  - https://opensource.apple.com/source/xnu/xnu-201/bsd/sys/errno.h

```
#define ENOTSUP      45              /* Operation not supported */
#ifndef _POSIX_SOURCE
#define EOPNOTSUPP       ENOTSUP               /* Operation not supported */
```

- 新版本：`xnu-792`
  - https://opensource.apple.com/source/xnu/xnu-792/bsd/sys/errno.h.auto.html

```
#define ENOTSUP               45             /* Operation not supported */
```

结论：

- `ENOTSUP` = `45`

man手册文档

前面介绍的

Apple 函数 man手册 总入口：API Reference: iOS Manual Pages (apple.com)

中有很多，iOS逆向期间，常常会涉及到的一些 `API` 或 `命令`，整理如下：

- Section 2: system calls and error numbers
  - ENOMEM 错误码定义
    - 12 = ENOMEM Cannot allocate memory
      - The new process image required more memory than was allowed by the hardware or by system-imposed mem-ory memory ory management constraints. A lack of swap space is normally temporary; however, a lack of core is not. Soft limits may be increased to their corresponding hard limits.
  - stat64
    - Mac OS X Manual Page For stat64(2)
- Section 3: C libraries
  - system
    - Mac OS X Manual Page For system(3)
  - sysctl
    - Mac OS X Manual Page For sysctl(3)
  - strlen
    - Mac OS X Manual Page For strlen(3)

iPhone iOS SDK 源码

- iPhoneOS13.0.sdk
  - iOS-SDKs/iPhoneOS13.0.sdk at master · xybp888/iOS-SDKs (github.com)
    - stat.h
      - iOS-SDKs/stat.h at master · xybp888/iOS-SDKs (github.com)
    - syscall.h
      - iOS-SDKs/syscall.h at master · xybp888/iOS-SDKs (github.com)
    - sysctl.h
      - iOS-SDKs/sysctl.h at master · xybp888/iOS-SDKs (github.com)
    - types.h
      - iOS-SDKs/types.h at master · xybp888/iOS-SDKs (github.com)

改机相关

sysctl相关

- 苹果官网文档
  - sysctlbyname
    - sysctlbyname | Apple Developer Documentation
  - sysctl
    - sysctl | Apple Developer Documentation
- man手册
  - SYSCTL
    - Mac OS X Manual Page For sysctlbyname(3) (apple.com)
- 源码
  - sysctl.c (apple.com)
- 其他
  - sysctlbyname (freebsd.org)
  - sysctlbyname(3) manual page (lemoda.net)
  - sysctl, sysctlbyname (qnx.com)
  - sysctlbyname.3 (daemon-systems.org)

其他

iOS中 属性列表 Property List = plist

- Introduction to Property Lists (apple.com)

C语言相关开发整理和心得

gcc

编译时-Wxxx的参数：

- Warning Options (Using the GNU Compiler Collection (GCC))

clang

编译时参数：

- Clang Compiler User's Manual — Clang 13 documentation (llvm.org)
- Clang command line argument reference — Clang 13 documentation (llvm.org)

通用逻辑

TODO：

- 【整理】iOS逆向心得：给函数加了hook同时加断点会导致EXC_BREAKPOINT的崩溃
- 【已解决】iOS逆向心得：类没有setXxx函数但是有属性xxx
- 【整理】iOS逆向调试心得：ObjC或ARM中从偏移量中取值的不同写法
- 【已解决】iOS中的caddr_t类型的定义
  - TODO：
    - 把各种iOS逆向期间，涉及的各种类型的定义，也整理过来

Prologue和Epilogue

TODO：

- 【整理】编程基础知识：函数Prologue开场白和函数Epilogue结尾

iOS内核和底层机制

TODO：

- iOS内核和底层机制
  - 【记录】研究XCode中iOS的app的ALSR相关配置
  - 【整理】Mac和iOS中的Sandbox沙箱
  - 【已解决】研究iOS中app的目录的UUID类的值和app名称如何映射
  - 【未解决】iOS的app的启动流程启动过程

NS和CF

iOS底层和内部：

底层有两套东西：

- `NS` 开头的 = `NextStep`
  - 比如
    - `NSString`
- `CF` 开头的 = `CoreFoundation`
  - 比如
    - `CFStringRef`

有些变量是可以相互互换使用的 == `toll-free bridged`

比如：

- `CFStringRef` == `NSString*`
- `CFURLRef` == `NSURL*`

详见：

- CFStringRef | Apple Developer Documentation
- CFURL | Apple Developer Documentation
- NSURL | Apple Developer Documentation

以及相关部分的变量的互相转换的写法是：

iOS中的：`NSString` / `NSString*` 和 `CFStringRef` 的互相转换：

- MRC

```
CFStringRef aCFString = (CFStringRef)aNSString;

NSString *aNSString = (NSString *)aCFString;
```

- ARC

```
CFStringRef aCFString = (__bridge CFStringRef)aNSString;

NSString *aNSString = (__bridge NSString *)aCFString;
```

常用小技巧

- 判断 `CFStringRef` 是否以某个字符串开头

```
CFStringRef prefixFile = (__bridge CFStringRef)@"file://";
CFStringHasPrefix(someCFStringRef, prefixFile)
```

- 判断 `CFURLRef` 是否为空

```
if (someCFURLRef != NULL){
```

- `CFURLRef` 转 `NSURL*`

```
NSURL *someNSURL = (NSURL *)someCFURLRef;
```

- 从NSURL获取url字符串NSString

```
NSString* someUrlNsStr = [someNSURL absoluteString];
```

去hook某个函数时的相关完整代码

```
CFURLRef CFURLCreateWithString(CFAllocatorRef allocator, CFStringRef URLString, CFURLRef baseURL);

%hookf(CFURLRef, CFURLCreateWithString, CFAllocatorRef allocator, CFStringRef URLString, CFURLRef baseURL){
    CFStringRef prefixFile = (__bridge CFStringRef)@"file://";
    CFStringRef prefixXCoredata = (__bridge CFStringRef)@"x-coredata://";
    bool shouldOmit = false;
    if (CFStringHasPrefix(URLString, prefixFile)){
        shouldOmit = true;
    } else if (CFStringHasPrefix(URLString, prefixXCoredata)) {
        shouldOmit = true;
    }
//    iosLogInfo("shouldOmit=%d for URLString=%{public}@", shouldOmit, URLString);

    if (baseURL != NULL){
        //    NSString* absUrlStr = [baseURL absoluteString];
        NSURL* baseNsurl = (NSURL*)baseURL;
        NSString* baseAbsUrlStr = [baseNsurl absoluteString];
        CFStringRef baseAbsUrlStrRef = (CFStringRef)baseAbsUrlStr;
        if(!shouldOmit) {
            if (CFStringHasPrefix(baseAbsUrlStrRef, prefixFile)){
                shouldOmit = true;
            } else if (CFStringHasPrefix(baseAbsUrlStrRef, prefixXCoredata)) {
                shouldOmit = true;
            }
//            iosLogInfo("shouldOmit=%d for baseAbsUrlStrRef=%{public}@", shouldOmit, baseAbsUrlStrRef);
        }
    }

    CFURLRef newUrl = %orig;
    if(!shouldOmit) {
        iosLogInfo("allocator=%{public}@, URLString=%{public}@, baseURL=%{public}@ -> urlStr=%{public}@", allocator,
URLString, baseURL, newUrl);
    }
    return newUrl;
}
```

函数签名

TODO：

- 【整理】iOS逆向心得：Block的invoke函数的签名signature的含义
  - 把如何动态调试查看函数签名的内容整理过来
- 【已解决】iOS逆向：__block_literal_global

- `函数签名` = `function signature`
  - 举例
    - `v8@?0`
      - 函数定义：`void ^();`
    - `v12@0:4@8`
      - 函数定义：`- (void) setSomething:(id) anObject;`

举例详解

调试某个ObjC的 `block` 时：

```
(lldb) reg r x0
      x0 = 0x000000014726ef70
(lldb) po 0x000000014726ef70
__NSMallocBlock__  0x14726ef70
 signature  "v32@?0@"NSError"8@16@"TTHttpResponse"24"
 invoke     0x111735758 ( private var containers Bundle Application 1FFDC079 CC8A 4219 955A E01C73207969 Aweme app Fr
ameworks AwemeCore framework AwemeCore` [MKMapView(AWEMap) awe_screenScope])
 copy       0x108c97674 ( private var containers Bundle Application 1FFDC079 CC8A 4219 955A E01C73207969 Aweme app Fr
ameworks AwemeCore framework AwemeCore` [AWELaunchMainPlaceholder _generateBootLoaderLogs])
 dispose    0x108c9767c ( private var containers Bundle Application 1FFDC079 CC8A 4219 955A E01C73207969 Aweme app Fr
ameworks AwemeCore framework AwemeCore` [AWELaunchMainPlaceholder _generateBootLoaderLogs])
```

可以看到：

- 函数签名：`v32@?0@"NSError"8@16@"TTHttpResponse"24`
  - 对应的函数定义其实是：`void ^(NSError *, id, TTHttpResponse *);`

继续查看Block详情中，也有Signature相关属性：

```
(lldb) po _Block_has_signature(0x14726ef70)
0x0000000000000001

(lldb) po _Block_signature(0x14726ef70)
0x0000000107067dd6
```

Process进程

进程flag

进程flag的定义

- iOS中process进程的flag的定义

来源1

- cs_blobs.h (apple.com)

```
#ifndef _KERN_CODESIGN_H_
#define _KERN_CODESIGN_H_


/* code signing attributes of a process */
#define CS_VALID                    0x00000001  /* dynamically valid */
#define CS_ADHOC                    0x00000002  /* ad hoc signed */
#define CS_GET_TASK_ALLOW           0x00000004  /* has get-task-allow entitlement */
#define CS_INSTALLER                0x00000008  /* has installer entitlement */


#define CS_FORCED_LV                0x00000010  /* Library Validation required by Hardened System Policy */
#define CS_INVALID_ALLOWED          0x00000020  /* (macOS Only) Page invalidation allowed by task port policy */


#define CS_HARD                     0x00000100  /* don't load invalid pages */
#define CS_KILL                     0x00000200  /* kill process if it becomes invalid */
#define CS_CHECK_EXPIRATION         0x00000400  /* force expiration checking */
#define CS_RESTRICT                 0x00000800  /* tell dyld to treat restricted */


#define CS_ENFORCEMENT              0x00001000  /* require enforcement */
#define CS_REQUIRE_LV               0x00002000  /* require library validation */
#define CS_ENTITLEMENTS_VALIDATED   0x00004000  /* code signature permits restricted entitlements */
#define CS_NVRAM_UNRESTRICTED       0x00008000  /* has com.apple.rootless.restricted-nvram-variables.heritable entitl
ement */


#define CS_RUNTIME                                  0x00010000  /* Apply hardened runtime policies */


#define CS_ALLOWED_MACHO            (CS_ADHOC | CS_HARD | CS_KILL | CS_CHECK_EXPIRATION | \
                                     CS_RESTRICT | CS_ENFORCEMENT | CS_REQUIRE_LV | CS_RUNTIME)


#define CS_EXEC_SET_HARD            0x00100000  /* set CS_HARD on any exec'ed process */
#define CS_EXEC_SET_KILL            0x00200000  /* set CS_KILL on any exec'ed process */
#define CS_EXEC_SET_ENFORCEMENT     0x00400000  /* set CS_ENFORCEMENT on any exec'ed process */
#define CS_EXEC_INHERIT_SIP         0x00800000  /* set CS_INSTALLER on any exec'ed process */


#define CS_KILLED                   0x01000000  /* was killed by kernel for invalidity */
#define CS_DYLD_PLATFORM            0x02000000  /* dyld used to load this is a platform binary */
#define CS_PLATFORM_BINARY          0x04000000  /* this is a platform binary */
#define CS_PLATFORM_PATH            0x08000000  /* platform binary by the fact of path (osx only) */


#define CS_DEBUGGED                 0x10000000  /* process is currently or has previously been debugged and allowed t
o run with invalid pages */
#define CS_SIGNED                   0x20000000  /* process has a signature (may have gone invalid) */
#define CS_DEV_CODE                 0x40000000  /* code is dev signed, cannot be loaded into prod signed code (will g
o away with rdar://problem/28322552) */
#define CS_DATAVAULT_CONTROLLER     0x80000000  /* has Data Vault controller entitlement */


#define CS_ENTITLEMENT_FLAGS        (CS_GET_TASK_ALLOW | CS_INSTALLER | CS_DATAVAULT_CONTROLLER | CS_NVRAM_UNRESTRICT
ED)


/* executable segment flags */
```

```
#define CS_EXECSEG_MAIN_BINARY          0x1                     /* executable segment denotes main binary */
#define CS_EXECSEG_ALLOW_UNSIGNED       0x10            /* allow unsigned pages (for debugging) */
#define CS_EXECSEG_DEBUGGER                     0x20            /* main binary is debugger */
#define CS_EXECSEG_JIT                          0x40            /* JIT enabled */
#define CS_EXECSEG_SKIP_LV                      0x80            /* OBSOLETE: skip library validation */
#define CS_EXECSEG_CAN_LOAD_CDHASH      0x100           /* can bless cdhash for execution */
#define CS_EXECSEG_CAN_EXEC_CDHASH      0x200           /* can execute blessed cdhash */
'''
```

来源2

- codesign.h (apple.com)

```
#ifndef _SYS_CODESIGN_H_
#define _SYS_CODESIGN_H_


/* code signing attributes of a process */
#define       CS_VALID          0x0000001       /* dynamically valid */
#define CS_ADHOC                0x0000002       /* ad hoc signed */
#define CS_GET_TASK_ALLOW       0x0000004       /* has get-task-allow entitlement */
#define CS_INSTALLER            0x0000008       /* has installer entitlement */


#define       CS_HARD                   0x0000100       /* don't load invalid pages */
#define       CS_KILL                   0x0000200       /* kill process if it becomes invalid */
#define CS_CHECK_EXPIRATION     0x0000400       /* force expiration checking */
#define CS_RESTRICT             0x0000800       /* tell dyld to treat restricted */
#define CS_ENFORCEMENT          0x0001000       /* require enforcement */
#define CS_REQUIRE_LV           0x0002000       /* require library validation */
#define CS_ENTITLEMENTS_VALIDATED       0x0004000       /* code signature permits restricted entitlements */


#define       CS_ALLOWED_MACHO  (CS_ADHOC | CS_HARD | CS_KILL | CS_CHECK_EXPIRATION | CS_RESTRICT | CS_ENFORCEMENT |
CS_REQUIRE_LV)


#define CS_EXEC_SET_HARD        0x0100000       /* set CS_HARD on any exec'ed process */
#define CS_EXEC_SET_KILL        0x0200000       /* set CS_KILL on any exec'ed process */
#define CS_EXEC_SET_ENFORCEMENT 0x0400000       /* set CS_ENFORCEMENT on any exec'ed process */
#define CS_EXEC_SET_INSTALLER   0x0800000       /* set CS_INSTALLER on any exec'ed process */


#define CS_KILLED               0x1000000       /* was killed by kernel for invalidity */
#define CS_DYLD_PLATFORM        0x2000000       /* dyld used to load this is a platform binary */
#define CS_PLATFORM_BINARY      0x4000000       /* this is a platform binary */
#define CS_PLATFORM_PATH        0x8000000       /* platform binary by the fact of path (osx only) */
#define CS_DEBUGGED             0x10000000  /* process is currently or has previously been debugged and allowed to ru
n with invalid pages */
#define CS_SIGNED           0x20000000  /* process has a signature (may have gone invalid) */
#define CS_DEV_CODE         0x40000000  /* code is dev signed, cannot be loaded into prod signed code (will go away w
ith rdar://problem/28322552) */

#define CS_ENTITLEMENT_FLAGS    (CS_GET_TASK_ALLOW | CS_INSTALLER)


/* MAC flags used by F_ADDFILESIGS_* */
#define MAC_VNODE_CHECK_DYLD_SIM 0x1   /* tells the MAC framework that dyld-sim is being loaded */


/* csops  operations */
#define       CS_OPS_STATUS             0       /* return status */
#define       CS_OPS_MARKINVALID        1       /* invalidate process */
#define       CS_OPS_MARKHARD           2       /* set HARD flag */
#define       CS_OPS_MARKKILL           3       /* set KILL flag (sticky) */
#ifdef KERNEL_PRIVATE
/* CS_OPS_PIDPATH               4       */
#endif
#define       CS_OPS_CDHASH             5       /* get code directory hash */
#define CS_OPS_PIDOFFSET        6       /* get offset of active Mach-o slice */
#define CS_OPS_ENTITLEMENTS_BLOB 7      /* get entitlements blob */
#define CS_OPS_MARKRESTRICT     8       /* set RESTRICT flag (sticky) */
#define CS_OPS_SET_STATUS       9       /* set codesign flags */
```

```
#define CS_OPS_BLOB             10      /* get codesign blob */
#define CS_OPS_IDENTITY         11      /* get codesign identity */
#define CS_OPS_CLEARINSTALLER   12      /* clear INSTALLER flag */
```

进程flag应用场景举例

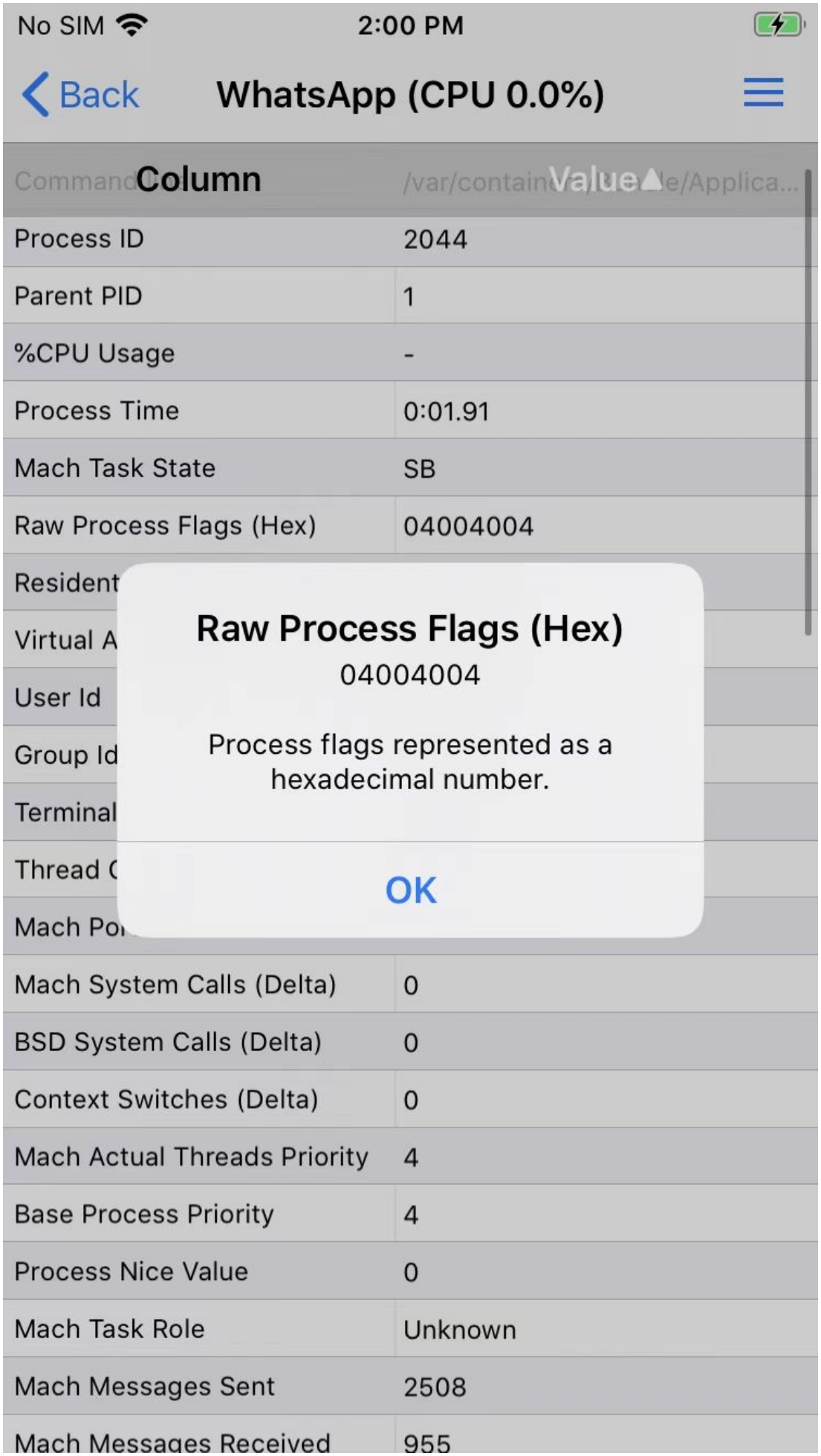
akd

iOS中（通过工具：`CocoaTop64`）查看出akd进程的：

- `flag` = `0x4004004`
  - 含义是
    - `CS_PLATFORM_BINARY` = 1
      - `#define CS_PLATFORM_BINARY 0x4000000 /* this is a platform binary */`
    - `CS_ENTITLEMENTS_VALIDATED` = 1
      - `#define CS_ENTITLEMENTS_VALIDATED 0x0004000 /* code signature permits restricted entitlements */`
    - `CS_GET_TASK_ALLOW` = 1
      - `#define CS_GET_TASK_ALLOW 0x0000004 /* has get-task-allow entitlement */`

WhatsApp

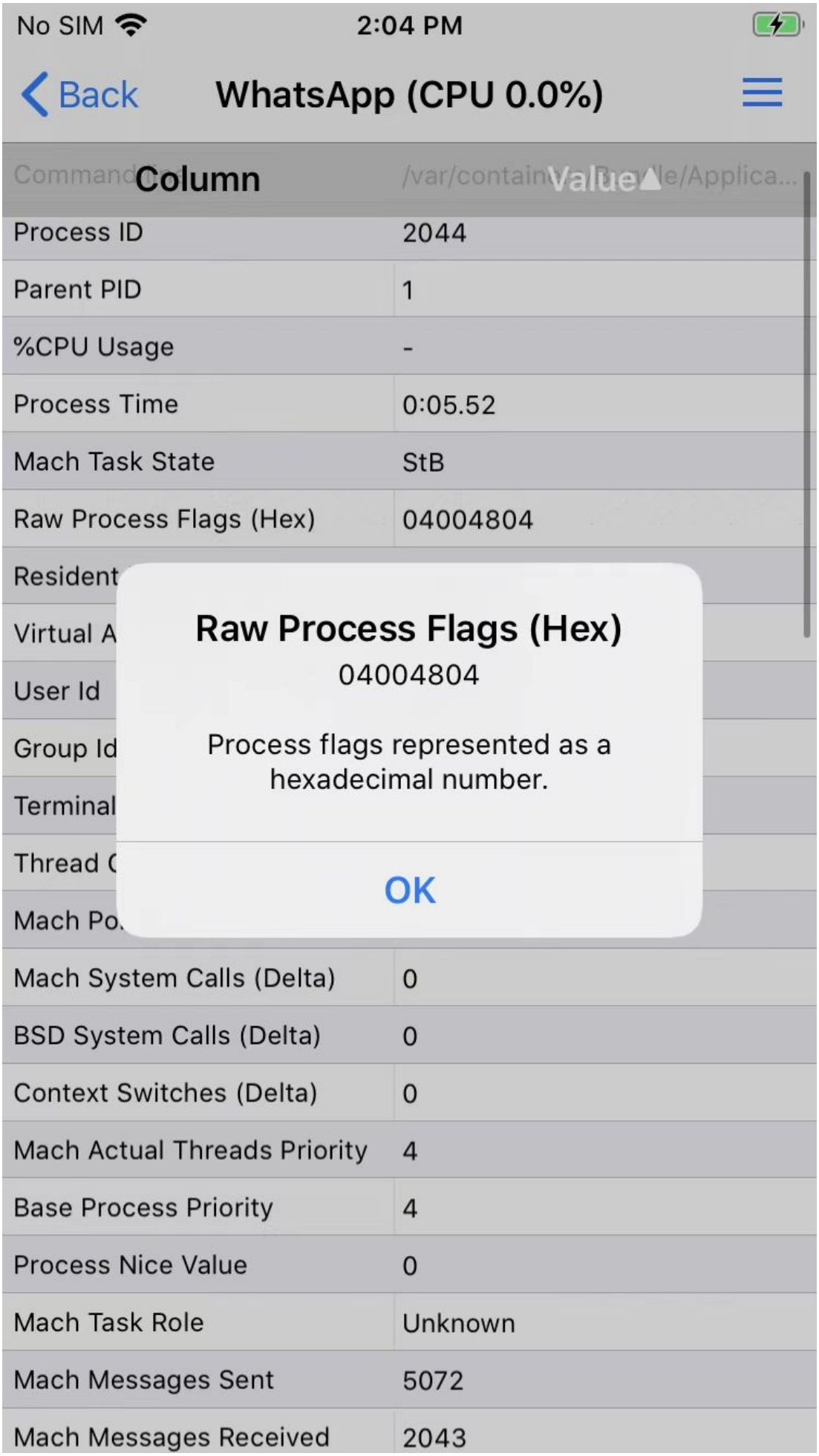
用CocoaTop查看WhatsApp进程的flag属性：


No SIM
2:00 PM
Back
WhatsApp (CPU 0.0%)
Column
Value
Process ID
2044
Parent PID
1
%CPU Usage
-
Process Time
0:01.91
Mach Task State
SB
Raw Process Flags (Hex)
04004004
Raw Process Flags (Hex)
04004004
Process flags represented as a hexadecimal number.
OK
Mach System Calls (Delta)
0
BSD System Calls (Delta)
0
Context Switches (Delta)
0
Mach Actual Threads Priority
4
Base Process Priority
4
Process Nice Value
0
Mach Task Role
Unknown
Mach Messages Sent
2508
Mach Messages Received
955

- 含义解释
  - `0x4000000` -> `CS_PLATFORM_BINARY` =1
  - `0x0004000` -> `CS_ENTITLEMENTS_VALIDATED` =1
  - `0x0000004` -> `CS_GET_TASK_ALLOW` =1

以及，给改进程去调试后：

- Frida调试：flag没变
- Xcode调试：flag变了
  - 变成了
    -


No SIM
2:04 PM
Back
WhatsApp (CPU 0.0%)
Column
Value
Process ID
2044
Parent PID
1
%CPU Usage
-
Process Time
0:05.52
Mach Task State
StB
Raw Process Flags (Hex)
04004804
Raw Process Flags (Hex)
04004804
Process flags represented as a hexadecimal number.
OK
Mach System Calls (Delta)
0
BSD System Calls (Delta)
0
Context Switches (Delta)
0
Mach Actual Threads Priority
4
Base Process Priority
4
Process Nice Value
0
Mach Task Role
Unknown
Mach Messages Sent
5072
Mach Messages Received
2043

- 新增了：`CS_RESTRICT` = `0x0000800`

子教程

此处列出，相关的子教程，和相关其他教程：

- ObjC
  - 底层机制
    - Block
      - iOS逆向开发：Block匿名函数
    - Runtime
      - iOS逆向开发：ObjC运行时
    - Framework
      - iOS逆向：Framework动态库
    - Apple OS
      - iOS逆向：Apple操作系统
- 其他相关
  - Apple苹果
    - 苹果相关开发总结
  - 可执行文件
    - 可执行文件格式：Mach-O
      - 概述：`Mach-O`是Apple平台（iOS和macOS等）中的底层二进制（可执行文件、库等）的格式

附录

下面列出相关参考资料。

参考资料

- 可执行文件格式：Mach-O
-
- 【整理】iOS中的Frameworks框架
- 【整理】dyld相关：dyld_shared_cache动态库共享缓存
- 【记录】研究黑豹dylib文件zzzzHeiBaoLib.dylib中字符串和反越狱相关内容
- 【已解决】dyld_shared_cache相关：update_dyld_shared_cache
- 【规避解决】Mac M2 Max中jtool2运行崩溃：killed
- 【已解决】用jtool2查看Mach-O的二进制akd找代码段相关信息
- 【已解决】iOS中进程的flag定义
- 【已解决】iOS逆向中CocoaTop64中查看进程详情中的Flags的具体含义
- 【记录】用CocoaTop查看WhatsApp进程的flag属性
- 【已解决】iOS中CFStringRef转换为NSString和互相转换
- 【已解决】iOS中CFStringRef和CFURLRef字符串判断和过滤以某个子字符串开头
- 【已解决】iOS中从otool输出的B16@0:8搞懂函数类型和参数定义
-
- 再看CVE-2016-1757---浅析mach message的使用 | mrh的学习分享 (turingh.github.io)
- 从RunLoop来看iOS内核中消息的发送: mach_msg - 掘金 (juejin.cn)
- 谈谈 iOS 中的 `dyld_shared_cache` · Issue #55 · kingcos/Perspective (github.com)
- update_dyld_shared_cache(1) [osx man page]
- Extracting libraries from dyld_shared_cache | Worth Doing Badly
- dyld_shared_cache - iPhone Development Wiki
- OS Internals: (newosxbook.com)
- MAC OS X Internals: A Systems Approach: Singh, Amit: 9780321278548: Books: Amazon.com
- Open Source - Apple Developer
- Documentation Archive (apple.com)
- Mac OS X Manual Page For stat64(2) (apple.com)
- Kernel Syscalls - The iPhone Wiki
- 越狱检测抖音逻辑??? - Cydia | 微信公众号文章阅读 - WeMP
- Frameworks - iPhone Development Wiki
- XLsn0w/Cydia
-