
目录

前言	1.1
debugserver+lldb概览	1.2
debugserver+lldb调试	1.3
确保debugserver权限	1.3.1
iPhone运行debugserver	1.3.2
Mac运行lldb	1.3.3
debugserver	1.4
原始debugserver位置	1.4.1
原始entitlement权限	1.4.1.1
help语法	1.4.2
lldb	1.5
使用心得	1.6
获取iOS的app二进制路径	1.6.1
entitlement权限	1.6.2
查看entitlement权限	1.6.2.1
重签名	1.6.2.2
常见问题	1.7
运行崩溃killed	1.7.1
附录	1.8
参考资料	1.8.1

iOS逆向调试：debugserver+lldb

- 最新版本： `v1.0`
- 更新时间： `20231021`

简介

介绍iOS逆向中动态调试中的其中一种方式：debugserver+lldb。先是概览；然后再去详细介绍如何调试，首先是确保iPhone中debugserver有正确的权限，接着是iPhone中用debugserver启动app，最后是Mac中用lldb去调试app；接着单独介绍debugserver，包括原始的位置和来源，以及附录上原始entitlement权限供参考，以及help语法；然后介绍lldb；然后总结使用心得，包括获取iOS的app二进制路径，entitlement权限的如何查看权限、如何重签名；以及常见问题，包括运行崩溃killed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_re_debug_debugserver_lldb](#): iOS逆向调试：debugserver+lldb

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逆向调试：[debugserver+lldb book.crifan.org](#)
- iOS逆向调试：[debugserver+lldb crifan.github.io](#)

离线下载阅读

- iOS逆向调试：[debugserver+lldb PDF](#)
- iOS逆向调试：[debugserver+lldb ePub](#)
- iOS逆向调试：[debugserver+lldb 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。如发现侵权，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。

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：
2023-10-21 19:14:42

debugserver+lldb概览

- debugserver+lldb
 - 是什么：iOS逆向中动态调试的其中一种手段
 - 对比来说，其他手段还有：MonkeyDev、Frida等等
 - 交互方式：命令行
 - 优点：通用、兼容
 - 缺点：（相对GUI图形界面说）不够直观
 - 基本架构



- 相关文件
 - iPhone 中的 Server 端：`debugserver=lldb server`
 - Mac 中的 Client 端：`lldb=lldb client`
- 对比：非常类似的，通过Xcode中的lldb去调试时的架构



- 使用方式概述
 - iPhone端运行**debugserver**
 - 注：其中的 debugserver 是重签名后的
 - 核心操作：加了entitlement权限，再重新签名
 - 目的：支持任意进程可调试
 - 对比：原先版本 `/Developer/usr/bin/debugserver` 是只读版本，不可直接修改，所以无法直接重签名
 - Mac端运行**lldb**

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：
2023-10-21 11:55:08

debugserver+lldb调试

此处介绍用 debugserver + lldb 去调试iOS程序。

此处举例说明：

- iPhone机型： iPhone 7 Plus , iOS 13.4.1
- 被调试app： 抖音
- 调试方式： debugserver+lldb 的命令行

核心思路是：

- iPhone 中：确保 debugserver 有正确的权限entitlement
 - 先从 iPhone 中导出 debugserver
 - 再去 Mac 中，（用 codesign 或 ldid ）给 debugserver 加上合适的权限
 - 再把加了 entitlement 的 debugserver 拷贝回 iPhone 中
- iPhone 中：用 debugserver 启动(抖音)app
- Mac中：用 lldb 去调试(抖音)app

下面介绍详细过程。

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:
2023-10-21 18:53:53

确保iPhone中debugserver有正确的权限

Mac中：从iPhone中导出debugserver到Mac

- 前提
 - 已实现ssh免密登录，所以可以直接用scp去拷贝

```
scp root@192.168.0.58:/Developer/usr/bin/debugserver .
```

- 参数说明
 - root : ssh 的用户名
 - 192.168.0.58 : iPhone的IP
 - /Developer/usr/bin/debugserver : iPhone 中的 debugserver 的所在目录，原始的 debugserver
 - . : 当前文件夹

Mac中：给debugserver加上合适的权限

关于加上合适权限，很多人，其他人，都是说的思路是：

- 多步：先导出权限，再编辑，最后加上

但是后来确认，直接：

- 一步 = 直接写入合适的权限

即可。

具体步骤：

准备好entitlement文件

把下面内容保存为： debugserver.entitlements

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>com.apple.springboard.debugapplications</key>
  <true/>
  <key>com.apple.backboardd.launchapplications</key>
  <true/>
  <key>com.apple.backboardd.debugapplications</key>
  <true/>
  <key>com.apple.frontboard.launchapplications</key>
  <true/>
  <key>com.apple.frontboard.debugapplications</key>
  <true/>
</dict>
```

```

<key>com.apple.private.logging.diagnostic</key>
<true/>
<key>com.apple.private.memorystatus</key>
<true/>
<key>com.apple.private.cs.debugger</key>
<true/>
<key>get-task-allow</key>
<true/>
<key>task_for_pid-allow</key>
<true/>
<key>run-unsigned-code</key>
<true/>
</dict>
</plist>

```

说明：

(先导出原始的debugserver的entitlement权限，再经过如下处理)

- 已加上权限： `get-task-allow` 、 `task_for_pid-allow` 、 `run-unsigned-code`
 - 目的：允许debugserver调试其他app
- 已去掉权限： `com.apple.security.network.server` 、 `com.apple.security.network.client`
 - 目的：防止后续lldb调试报错 `Failed to get connection from a remote gdb process`
- 已去掉权限： `seatbelt-profiles`
 - 目的：方式后续 debugserver 加上 `-l` 的日志文件时报错： `Failed to open log file for writing errno = 1 Operation not permitted`

把entitlement权限加到debugserver中

- 概述
 - 推荐用 `codesign`

```
codesign -f -s - --entitlements debugserver.entitlements debugserver
```

- 详解
 - [重签名](#)

Mac中：把加了entitlement权限的 debugserver 拷贝回 iPhone 中

```
scp debugserver root@192.168.0.58:/usr/bin
```

说明：

- 为何没有拷贝回/覆盖原先的 `/Developer/usr/bin/debugserver` ?
 - 因为 `/Developer` 是 `ramdisk` 挂载的，是 `readonly` = 只读的，无法写入
- 为何选择路径 `/usr/bin` ?
 - 用于存放可执行文件工具的路径， `/usr/bin` 是常见之一，比较适合此处用途

- 其他目录，理论上也可以：
 - `/bin`
 - `/sbin`
 - `/usr/sbin`
 - 等
- 注意：确保iPhone中的 `PATH` 环境变量包含此处所用路径即可
 - 目的：便于后续其他任何位置都能找得到和能直接调用 `debugserver`
 - 如何查看当前环境变量值：`echo $PATH`
 - 举例

```
# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/bin/x11:/usr/games
```

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:
2023-10-21 18:01:19

iPhone中用debugserver启动app

此处通过例子来介绍，iPhone中用debugserver启动iOS的app：抖音

概述

iPhone中用debugserver启动（抖音）app：

- Attach模式
 - 用PID: `debugserver 0.0.0.0:20221 -a 10194`
 - 用app名称: `debugserver 0.0.0.0:20221 -a "Aweme"`
- Spawn模式
 - `debugserver -x auto 0.0.0.0:20221 /private/var/xxx/Aweme.app/Aweme`

详解

- Attach模式：先手动启动app，再去用 debugserver 挂载 attach

- -a 加上 PID 或 app名

- PID =进程ID

```
debugserver 0.0.0.0:20221 -a 10194
```

- app名

```
debugserver 0.0.0.0:20221 -a "Aweme"  
debugserver 0.0.0.0:1234 -a "YouTube"
```

- 其他额外参数

- 加日志 log

```
debugserver -l debug.log 0.0.0.0:20221 -a 10194
```

- 加详情 verbose

```
debugserver -v 0.0.0.0:20221 -a 10194
```

- 开启调试 debugging

```
debugserver -g 0.0.0.0:20221 -a 10194
```

- Spaw模式：直接用debugserver启动app

```
debugserver -x auto 0.0.0.0:20221 /private/var/containers/Bundle/Application/9AB25481-0AD3-435C-A02E-68F9623535BB/Aweme.app/Aweme
```

- 说明

- 关于如何获取到iOS的app二进制的完整路径，详见：[获取iOS的app二进制路径](#)

相关说明

- 0.0.0.0 : 比较好理解, 表示: 允许 (来自外部的) 任意IP访问
 - 此处指的是: 允许电脑端 (Mac) 中的lldb来访问
- 20221 : 端口号
 - 可以设置任意值, 只要不和其他端口号冲突即可
 - 注: 后续Mac中lldb连接时, 要用到此端口号
- 如果遇到反调试而启动失败, 则需要去处理 反反调试
 - 详见: [常见问题](#)

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:
2023-10-21 19:01:50

Mac中用lldb去调试app

然后去Mac中用lldb去调试app（抖音）

概述：

```
lldb

process connect connect://192.168.0.58:20221
```

参数说明：

- 192.168.0.58 : 是 iPhone 的 IP 地址
- 20221 : 是之前 debugserver 启动时设置的端口号

然后Mac中即可愉快的正常的调试了：

```
(lldb) b ptrace
Breakpoint 1: no locations (pending).
WARNING: Unable to resolve breakpoint to any actual locations.
(lldb) c
Process 10174 resuming
1 location added to breakpoint 1
...

```

注：

此时，iPhone的debugserver会输出：

```
Got a connection, launched process /private/var/containers/Bundle/Application/9AB25481-0AD3-435C-A02E-68F9623535BB/Aweme.app/Aweme (pid = 10211).
```

完整的2端的效果是：

```

12 +0.000026 sec [27ac/0303]: ::task_set_exception_ports ( task = 0x1407, exception_ma
sk = 0x00001bfe, new_port = 0x2903, behavior = 0x80000001, new_flavor = 0x00000005 ) e
rr = 0x00000000
13 +0.000123 sec [27ac/1703]: 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/containe
rs/Bundle/Application/9AB25481-0AD3-435C-A02E-68F9623535BB/Aweme.app/Aweme
debugserver-@(#)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
-0AD3-435C-A02E-68F9623535BB/Aweme.app/Aweme (pid = 10174).
Available completions:
connect -- Connect to a remote debug service.
continue -- Continue execution of all threads in the current process.
(lldb) process connect connect:192.168.0.58:20221
error: unsupported connection URL: 'connect:192.168.0.58:20221'
(lldb) process connect connect://192.168.0.58:20221
Process 10174 stopped
* thread #1, stop reason = signal SIGSTOP
  frame #0: 0x00000000100555000 dyld`_dyld_start
dyld`_dyld_start:
-> 0x100555000 <<0>: mov    x28, sp
0x100555004 <<4>: and    sp, x28, #0xffffffffffffff0
0x100555008 <<8>: mov    x0, #0x0
0x10055500c <<12>: mov   x1, #0x0
Target 0: (Aweme) stopped.
(lldb)

```

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: 2023-10-21 18:55:27

debugserver

对于 debugserver+lldb 的调试方案来说，应该先介绍背景知识：

LLDB的远程调试

- LLDB的远程调试
 - 涉及到2个端
 - lldb client
 - 运行在 local system
 - 比如PC端（Linux、Mac）的 lldb 命令行工具
 - lldb server
 - 不同平台
 - Linux 和 Android：lldb-server
 - 不依赖于 lldb
 - 因为：已静态链接包含了 LLDB 的核心功能
 - 对比：lldb 是默认是动态链接 liblldb.so
 - Mac 和 iOS：debugserver
 - 运行在 remote system
 - 典型例子
 - iPhone中的：debugserver
 - Android中的：lldb-server
 - 实现了remote-gdb的功能
 - 两者通讯
 - 用的是：gdb-remote 协议
 - 一般是在TCP/IP之上运行
 - 细节详见：
 - docs/lldb-gdb-remote.txt
 - 资料
 - 主页
 - Remote Debugging — The LLDB Debugger
 - <http://lldb.lvm.org/use/remote.html>

debugserver

然后再来详细介绍：debugserver 本身：

- debugserver
 - 是什么：一个终端的应用
 - 也是：xCode 去调试iOS设备中程序时候的进程名
 - 在哪里：iOS设备中
 - 位置：/Developer/usr/bin/debugserver
 - 注：iOS中默认没安装debugserver
 - iOS何时安装了 debugserver

- 在设备连接过一次 Xcode , 并在 Window -> Devices 中添加此设备后
 - debugserver 才会被 xcode 安装到 iOS 的 /Developer/usr/bin/ 下
- 作用: 作为服务端, 接受来自远端的 gdb 或 lldb 的调试
 - 可以理解为: lldb 的 server
- 为何需要
 - iOS中, 由于App运行检测到越狱后会直接退出, 所以需要通过 debugserver 来启动程序
- 通过 debugserver 来启动程序
 - 举例

```
debugserver -x backboard 0.0.0.0:1234 ./*
```

```
debugserver *:1234 -a "MoneyPlatListedVersion"
```

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:
2023-10-21 17:04:30

原始debugserver位置

我们在讨论iPhone中的 debugserver ，往往是会提到：

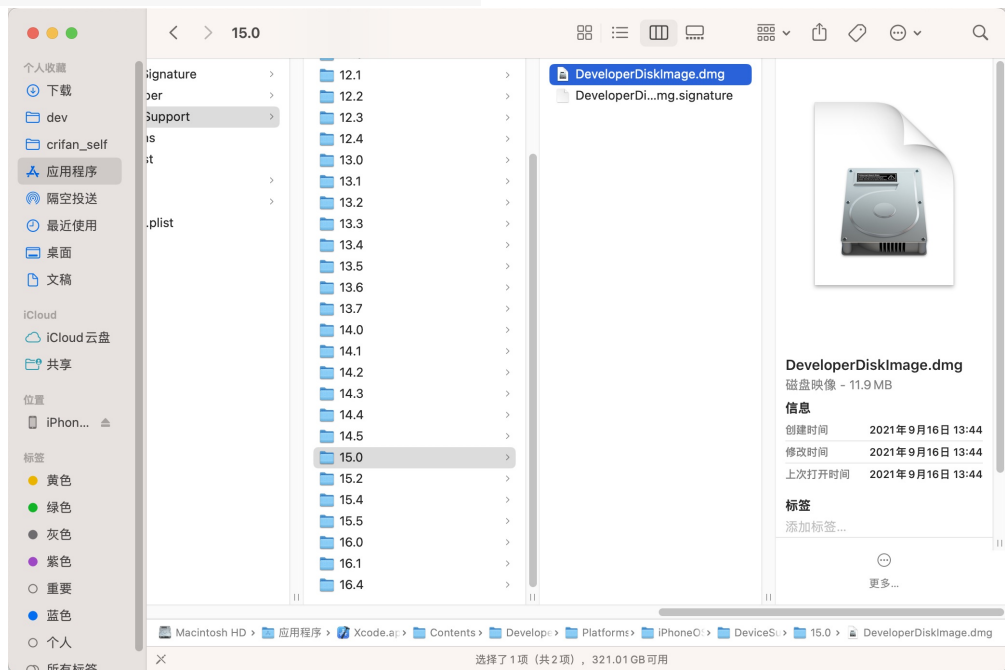
- iPhone中的，原始版本的 debugserver
 - 位置是： /Developer/usr/bin/debugserver
 - 特点是：只读，无法修改
 - 所以才会涉及到，拷贝到Mac中，添加可调试等权限，重新签名，写回iPhone（到另外的位置，一般是 /usr/bin/debugserver ）

但是其实该文件来自更早的地方：

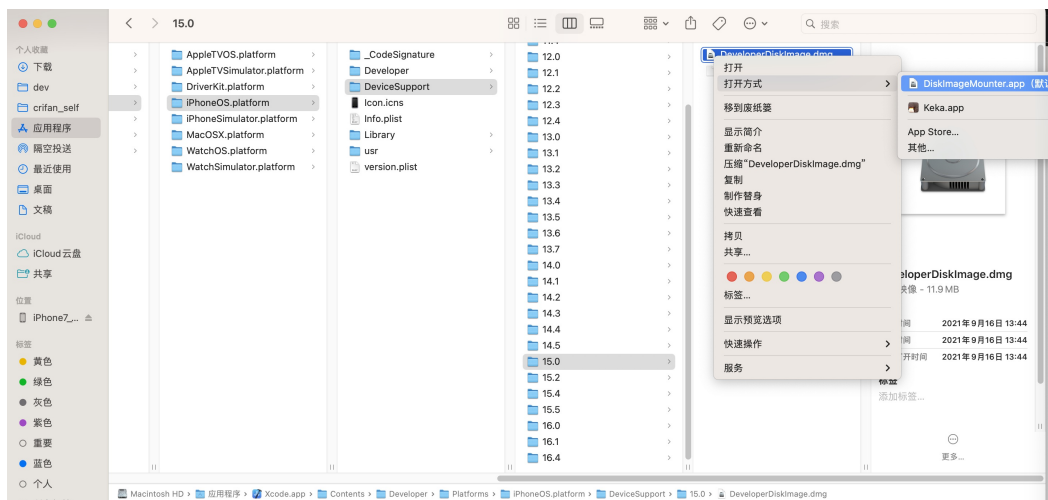
- DeveloperDiskImage.dmg 中的 /usr/bin/debugserver

DeveloperDiskImage.dmg

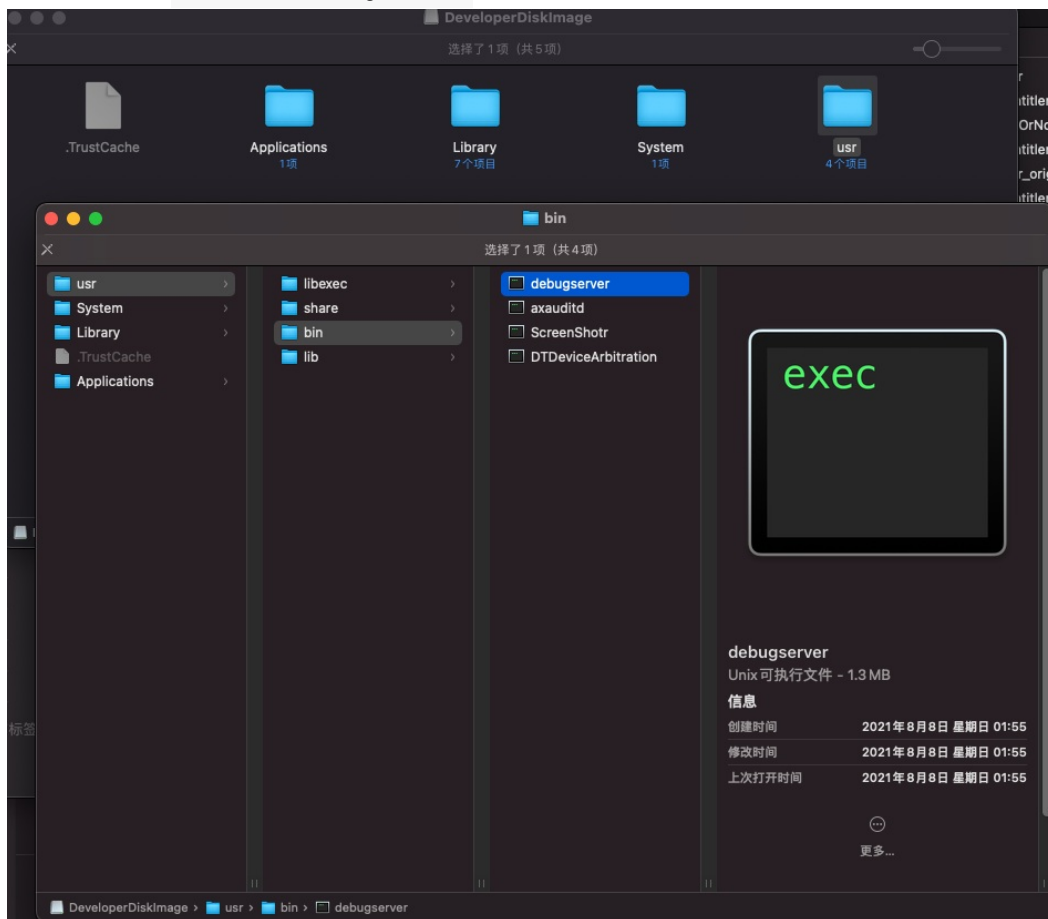
- 关于： DeveloperDiskImage.dmg
 - 位置：在Xcode中
 - 举例： iOS 15.0
 - /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/DeviceSupport/15.0/DeveloperDiskImage.dmg



- 如何打开
 - 方式1： 右键 -> DiskImageMounter.app



- 方式2:
 - 双击 DeveloperDiskImage.dmg , 而自动挂载打开
- 打开后=挂载后
 - 可以查看到其中的 /usr/bin/debugserver



-> 即: iPhone 中的 /Developer/usr/bin/debugserver , 最早就是来自于Xcode中的 DeveloperDiskImage.dmg 中的 /usr/bin/debugserver

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: 2023-10-21 18:37:19

原始debugserver的entitlement权限

从原始版本的debugserver中查看=导出的entitlement权限（基本）都是一样的。

此处列出

- iOS 13.4.1 的 iPhone7 Plus 中的
 - 原始的 debugserver (/Developer/usr/bin/debugserver)

的 entitlement 权限:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>com.apple.springboard.debugapplications</key>
  <true/>
  <key>com.apple.backboardd.launchapplications</key>
  <true/>
  <key>com.apple.backboardd.debugapplications</key>
  <true/>
  <key>com.apple.frontboard.launchapplications</key>
  <true/>
  <key>com.apple.frontboard.debugapplications</key>
  <true/>
  <key>seatbelt-profiles</key>
  <array>
    <string>debugserver</string>
  </array>
  <key>com.apple.private.logging.diagnostic</key>
  <true/>
  <key>com.apple.security.network.server</key>
  <true/>
  <key>com.apple.security.network.client</key>
  <true/>
  <key>com.apple.private.memorystatus</key>
  <true/>
  <key>com.apple.private.cs.debugger</key>
  <true/>
</dict>
</plist>
```

供参考。

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:
2023-10-21 18:39:13

debugserver的help语法

语法概述

```
iPhone11-151:~ root# /Developer/usr/bin/debugserver --help
debugserver: unrecognized option '--help'
debugserver-@(#)PROGRAM:LLDB PROJECT:lldb-1300.2.10
for arm64.
Usage:
debugserver host:port [program-name program-arg1 program-arg2 ...]
debugserver /path/file [program-name program-arg1 program-arg2 ...]
debugserver host:port --attach=pid
debugserver /path/file --attach=pid
debugserver host:port --attach=process_name>
debugserver /path/file --attach=process_name>
```

语法详解

```
debugserver [ options ] host:port > [ prog-name > arg1 > arg2 > ... ]
```

参数含义:

- `-a process`
 - Attach debugserver to process. The process can be a pid or executable name
- `-d integer`
 - Assign the waitfor-duration
- `-f ?`
 - ?
- `-g`
 - Turn on debugging
- `-i integer`
 - Assign the waitfor-interval
- `-l filename`
 - Log to file. Set filename to stdout to log to standard output
- `-t`
 - Use task ID instead of process ID
- `-v`
 - Verbose
- `-w ?`
 - ?
- `-x method` `--launch=method`
 - 参数缩写的说明
 - 估计是: `x`, 表示 `execute = 执行 = 启动运行 = (此参数的全称) launch`
 - 所以对应的着: `--launch=method`

- How to launch the program. Can be one of:
 - `auto` : Auto-detect the best launch method to use.
 - `fork` : Launch program using `fork(2)` and `exec(3)`.
 - `posix` : Launch program using `posix_spawn(2)`.
 - `backboard` : Launch program via BackBoard Services.
 - The backboard option is only available in the closed-source version included in Xcode
- `--lockdown`
 - Obtain exit parameters from lockdown (?)

相关源码

[debugserver.cpp \(apple.com\)](https://opensource.apple.com/source/debugserver/debugserver.cpp)

```

void
show_usage_and_exit (int exit_code)
{
    RNLogSTDERR ("Usage:\n %s host:port [program-name program-arg1 program-arg2 ...]\n", DEBUGSERVER_PROGRAM_NAME);
    RNLogSTDERR (" %s /path/file [program-name program-arg1 program-arg2 ...]\n", DEBUGSERVER_PROGRAM_NAME);
    RNLogSTDERR (" %s host:port --attach=<pid>\n", DEBUGSERVER_PROGRAM_NAME);
    RNLogSTDERR (" %s /path/file --attach=<pid>\n", DEBUGSERVER_PROGRAM_NAME);
    RNLogSTDERR (" %s host:port --attach=<process_name>\n", DEBUGSERVER_PROGRAM_NAME);
    RNLogSTDERR (" %s /path/file --attach=<process_name>\n", DEBUGSERVER_PROGRAM_NAME);
    exit (exit_code);
}

//-----
// option descriptors for getopt_long()
//-----
static struct option g_long_options[] =
{
    { "attach",          required_argument, NULL, 'a' },
    { "arch",           required_argument, NULL, 'A' },
    { "debug",          no_argument, NULL, 'g' },
    { "verbose",        no_argument, NULL, 'v' },
    { "lockdown",       no_argument, &g_lockdown_opt, l }, // short option "-k"
    { "applist",        no_argument, &g_applist_opt, l }, // short option "-e"
    { "log-file",       required_argument, NULL, 'l' },
    { "log-flags",      required_argument, NULL, 'f' },
    { "launch",         required_argument, NULL, 'x' }, // Valid values are "auto", "posix-spawn", "fork-exec", "springboard" (arm only)
    { "waitfor",        required_argument, NULL, 'w' }, // Wait for a process whose name starts with ARG
    { "waitfor-interval", required_argument, NULL, 'i' }, // Time in usecs to wait between sampling the pid list when waiting for a process by name
    { "waitfor-duration", required_argument, NULL, 'd' }, // The time in seconds to wait for a process to show up by name
    { "native-regs",    no_argument, NULL, 'r' }, // Specify to use the native registers instead of the gdb defaults for the architecture.
    { "stdio-path",     required_argument, NULL, 's' }, // Set the STDIO path to be used when launching applications (STDIN, STDOUT and STDERR) (only if del
    { "stdin-path",     required_argument, NULL, 'I' }, // Set the STDIN path to be used when launching applications (only if debugserver launches the proc
    { "stdout-path",    required_argument, NULL, 'O' }, // Set the STDOUT path to be used when launching applications (only if debugserver launches the proc
    { "stderr-path",    required_argument, NULL, 'E' }, // Set the STDERR path to be used when launching applications (only if debugserver launches the proc
    { "no-stdio",       no_argument, NULL, 'n' }, // Do not set up any stdio (perhaps the program is a GUI program) (only if debugserver launches the
    { "setsid",         no_argument, NULL, 'S' }, // call setsid() to make debugserver run in its own session
    { "disable-aslr",   no_argument, NULL, 'D' }, // Use POSIX SPAWN_DISABLE_ASLR to avoid shared library randomization
    { "working-dir",    required_argument, NULL, 'W' }, // The working directory that the inferior process should have (only if debugserver launches the
    { "platform",       required_argument, NULL, 'p' }, // Put this executable into a remote platform mode
    { "unix-socket",    required_argument, NULL, 'u' }, // If we need to handshake with our parent process, an option will be passed down that specifies a
    { NULL,             0, NULL, 0 }
};

```

```

void
show_usage_and_exit (int exit_code)
{
    RNLogSTDERR ("Usage:\n %s host:port [program-name program-arg1 program-arg2 ...]\n",
    n", DEBUGSERVER_PROGRAM_NAME);
    RNLogSTDERR (" %s /path/file [program-name program-arg1 program-arg2 ...]\n", DEB
    UGSERVER_PROGRAM_NAME);
    RNLogSTDERR (" %s host:port --attach=<pid>\n", DEBUGSERVER_PROGRAM_NAME);
    RNLogSTDERR (" %s /path/file --attach=<pid>\n", DEBUGSERVER_PROGRAM_NAME);
    RNLogSTDERR (" %s host:port --attach=<process_name>\n", DEBUGSERVER_PROGRAM_NAME);

    RNLogSTDERR (" %s /path/file --attach=<process_name>\n", DEBUGSERVER_PROGRAM_NAME)

;
    exit (exit_code);
}

...

//-----
// option descriptors for getopt_long()

```

```

//-----
static struct option g_long_options[] =
{
  { "attach",          required_argument, NULL,          'a' },
  { "arch",           required_argument, NULL,          'A' },
  { "debug",          no_argument,      NULL,          'g' },
  { "verbose",        no_argument,      NULL,          'v' },
  { "lockdown",       no_argument,      &g_lockdown_opt, 1 }, // short op
tion "-k"
  { "applist",        no_argument,      &g_applist_opt,  1 }, // short op
tion "-t"
  { "log-file",       required_argument, NULL,          'l' },
  { "log-flags",      required_argument, NULL,          'f' },
  { "launch",         required_argument, NULL,          'x' }, // Valid va
lues are "auto", "posix-spawn", "fork-exec", "springboard" (arm only)
  { "waitfor",        required_argument, NULL,          'w' }, // Wait for
a process whose name starts with ARG
  { "waitfor-interval", required_argument, NULL,          'i' }, // Time in
usecs to wait between sampling the pid list when waiting for a process by name
  { "waitfor-duration", required_argument, NULL,          'd' }, // The time
in seconds to wait for a process to show up by name
  { "native-regs",    no_argument,      NULL,          'r' }, // Specify
to use the native registers instead of the gdb defaults for the architecture.
  { "stdio-path",     required_argument, NULL,          's' }, // Set the
STDIO path to be used when launching applications (STDIN, STDOUT and STDERR) (only if d
ebugserver launches the process)
  { "stdin-path",     required_argument, NULL,          'I' }, // Set the
STDIN path to be used when launching applications (only if debugserver launches the pr
ocess)
  { "stdout-path",    required_argument, NULL,          'O' }, // Set the
STDOUT path to be used when launching applications (only if debugserver launches the pr
ocess)
  { "stderr-path",    required_argument, NULL,          'E' }, // Set the
STDERR path to be used when launching applications (only if debugserver launches the pr
ocess)
  { "no-stdio",       no_argument,      NULL,          'n' }, // Do not s
et up any stdio (perhaps the program is a GUI program) (only if debugserver launches th
e process)
  { "setuid",         no_argument,      NULL,          'S' }, // call set
uid() to make debugserver run in its own session
  { "disable-aslr",   no_argument,      NULL,          'D' }, // Use _POS
IX_SPAWN_DISABLE_ASLR to avoid shared library randomization
  { "working-dir",    required_argument, NULL,          'W' }, // The work
ing directory that the inferior process should have (only if debugserver launches the p
rocess)
  { "platform",       required_argument, NULL,          'p' }, // Put this
executable into a remote platform mode
  { "unix-socket",    required_argument, NULL,          'u' }, // If we ne
ed to handshake with our parent process, an option will be passed down that specifies a
unix socket name to use
  { NULL,             0,                NULL,          0 }
};

```

2023-10-21 17:21:17

lldb

- `lldb` =Mac端的 `lldb` 的 Client
 - 具体用法详见
 - [主流调试器: LLDB](#)

Mac端: 自带 `lldb`

- 位置

```
→ ~ which lldb  
/usr/bin/lldb
```

- 版本

```
→ ~ lldb --version  
lldb-1403.0.17.67  
Apple Swift version 5.8.1 (swiftlang-5.8.0.124.5 clang-1403.0.22.11.100)
```

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:
2023-10-21 16:10:10

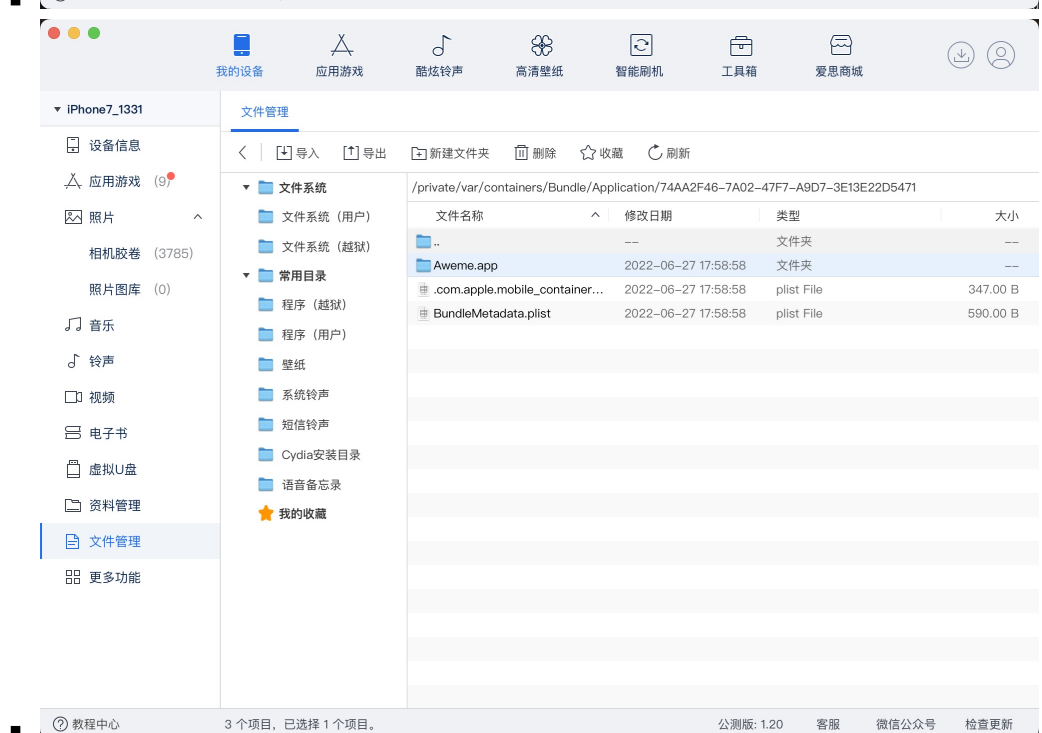
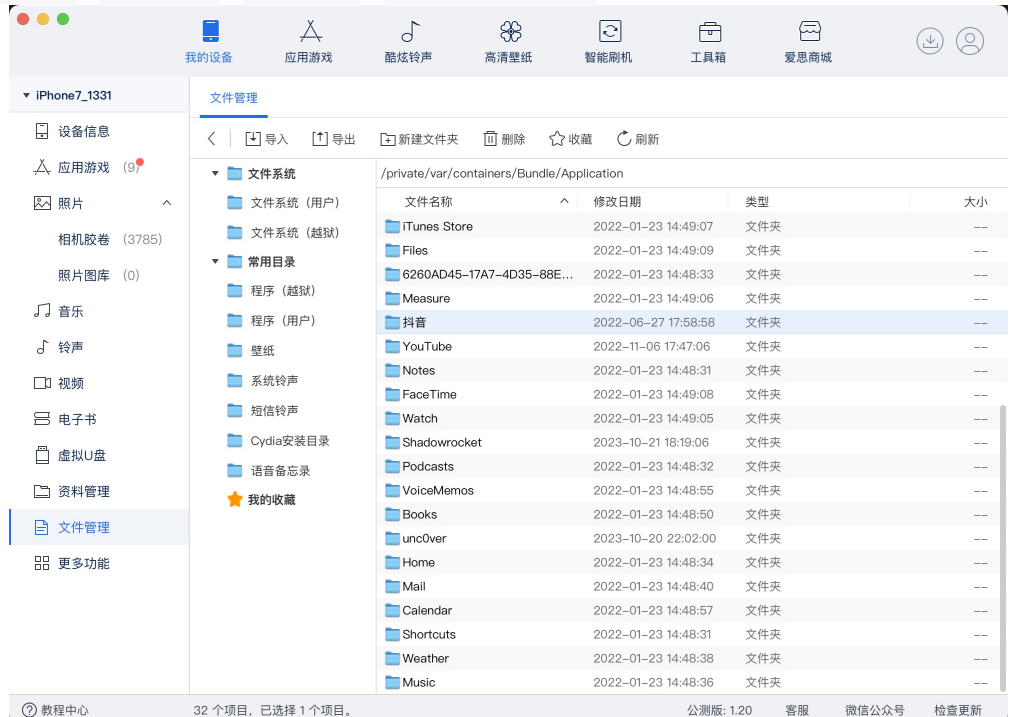
使用心得

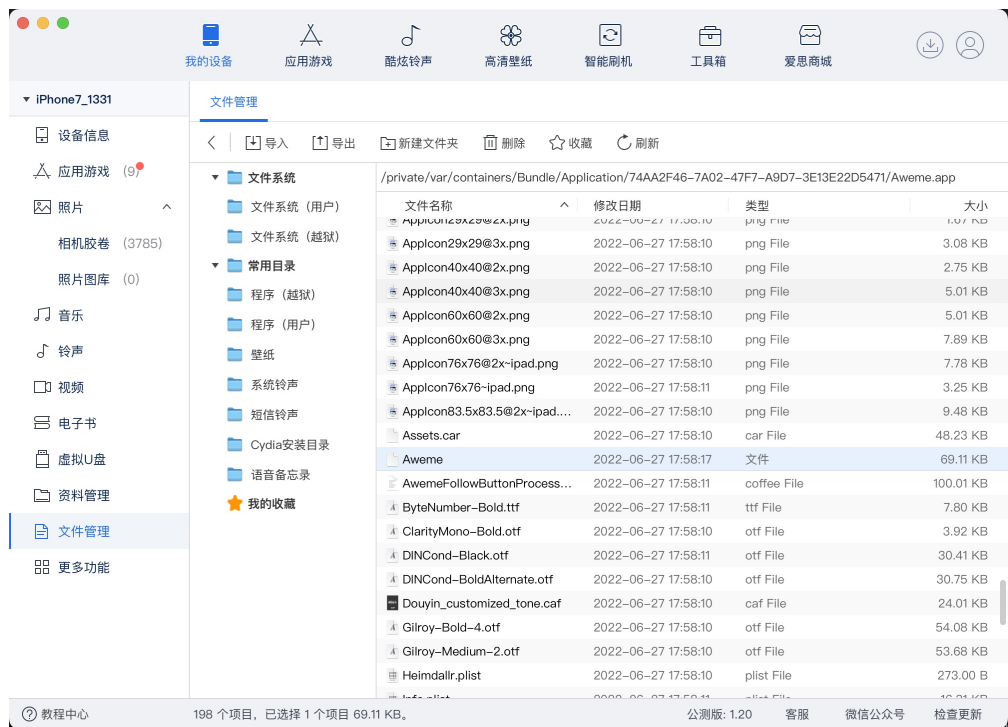
crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新:
2023-10-21 18:56:54

获取iOS的app二进制路径

关于iPhone以Spawn方式运行debugserver所涉及到的：

- iOS的app的二进制的完整路径，有多种方式可以得到
 - 方式1：通过爱思助手，查看已安装的app的相关路径
 - 抖音： `/private/var/containers/Bundle/Application/74AA2F46-7A02-47F7-A9D7-3E13E22D5471/Aweme.app/Aweme`
 - 爱思助手 -> 文件管理 -> 文件系统 -> 文件系统（越狱）





方式2: 通过前面的Attach模式调试期间, 通过 `image list -o -f` 查看得到

■ 举例

- WhatsApp : `/private/var/containers/Bundle/Application/CCFD22D2-32EE-4F23-9C81-226663100D40/WhatsApp.app/WhatsApp`

■ Attach模式调试时可查看到

```
(lldb) image list -o -f
[ 0] 0x000000004c6c000 /private/var/containers/Bundle/Application/CCFD22D2-32EE-4F23-9C81-226663100D40/WhatsApp.app/WhatsApp(0x00000000104c6c000)
...
```

方式3: (app正在运行时) 通过 CocoaTop 查看进程详情中的 Command Line 可以看到二进制完整路径

■ 举例

- WhatsApp

Column	Value▲
Command line	/var/containers/Bundle/Applica...
Process ID	2044
Parent PID	1
%CPU Usage	-
Process Time	0:05.62
Mach Task State	...
Raw Process Info	...
Resident Memory	...
Virtual Address	...
User Id	...
Group Id	...
Terminal	...
Thread Count	...
Mach Ports	162
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	5761

Command line

/var/containers/Bundle/Application/
02BED373-87F5-4B48-9F4B-70E853
B20267/WhatsApp.app/WhatsApp

Full command line with path
and arguments.

OK

entitlement权限

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:
2023-10-21 18:01:55

查看entitlement权限

如果想要 查看 = 导出 ，比如原始版本的 debugserver 的，entitlement权限，可以用：`codesign` 或 `ldid`

- 查看/导出entitlement权限
 - `ldid`

```
ldid -e debugserver
```

- `codesign`

```
codesign -d --entitlements - debugserver
```

- 参数说明

- `-d` : display显示
- `--entitlements` : 权限信息
- `-` : (把信息输出到) 当前默认 (stdout的) 终端=terminal

ldid对于FAT格式会输出多份entitlement权限信息

如果是 FAT 格式，`ldid -e debugserver` 则会输出 (多个架构所对应的) 多份entitlement权限信息

举例：

此处从iPhone8中导出的包含 arm64 和 arm64e 的 FAT 格式的 debugserver :

```
crifan@licrifandeMacBook-Pro ~/dev/dev_root/iosReverse/AppleStore/fromiPhone8/Developer/usr/bin ll
total 5240
-rw-r--r--  1 crifan  staff   832B  3  3 11:48 debugable_entitlement.xml
-rwxrwxr-x  1 crifan  staff  1.3M  8  8 2021 debugserver

crifan@licrifandeMacBook-Pro ~/dev/dev_root/iosReverse/AppleStore/fromiPhone8/Developer/usr/bin file debugserver
debugserver: Mach-O universal binary with 2 architectures: [arm64:Mach-O 64-bit executable arm64] [arm64e:Mach-O 64-bit executable arm64e]
debugserver (for architecture arm64):    Mach-O 64-bit executable arm64
debugserver (for architecture arm64e):   Mach-O 64-bit executable arm64e
```

用ldid查看时，会输出2份entitlement权限信息：

```
crifan@licrifandeMacBook-Pro ~/dev/dev_root/iosReverse/AppleStore/fromiPhone8/Developer/usr/bin ldid -e debugserver
<?xml version="1.0" encoding="UTF-8"?
< DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
```

```
key com.apple.springboard.debugapplications /key
true/
key com.apple.backboardd.launchapplications /key
true/
key com.apple.backboardd.debugapplications /key
true/
key com.apple.frontboard.launchapplications /key
true/
key com.apple.frontboard.debugapplications /key
true/
key seatbelt-profiles /key
array
  string debugserver /string
/array
key com.apple.private.logging.diagnostic /key
true/
key com.apple.security.network.server /key
true/
key com.apple.security.network.client /key
true/
key com.apple.private.memorystatus /key
true/
key com.apple.private.cs.debugger /key
true/
</dict>
</plist>
<?xml version="1.0" encoding="UTF-8"?>
< DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  key com.apple.springboard.debugapplications /key
  true/
  key com.apple.backboardd.launchapplications /key
  true/
  key com.apple.backboardd.debugapplications /key
  true/
  key com.apple.frontboard.launchapplications /key
  true/
  key com.apple.frontboard.debugapplications /key
  true/
  key seatbelt-profiles /key
  array
    string debugserver /string
  /array
  key com.apple.private.logging.diagnostic /key
  true/
  key com.apple.security.network.server /key
  true/
  key com.apple.security.network.client /key
  true/
  key com.apple.private.memorystatus /key
  true/
  key com.apple.private.cs.debugger /key
  true/
</dict>
```

```
</plist
```

对比：

后来去用lipo瘦身：

```
x crifan@licrifandeMacBook-Pro ~ /dev/dev_root/iosReverse/AppleStore/fromiPhone8/Developer/usr/bin lipo -thin arm64 debugserver -output debugserver_orig_arm64
crifan@licrifandeMacBook-Pro ~ /dev/dev_root/iosReverse/AppleStore/fromiPhone8/Developer/usr/bin ll
total 6504
-rw-r--r--  1 crifan  staff   832B  3  3 11:48 debugable_entitlement.xml
-rwxrwxr-x  1 crifan  staff  1.3M  8  8 2021 debugserver
-rwxr-xr-x  1 crifan  staff  1.3M  3  3 11:49 debugserver_debugable
-rwxr-xr-x  1 crifan  staff  632K  8  8 2021 debugserver_orig_arm64
crifan@licrifandeMacBook-Pro ~ /dev/dev_root/iosReverse/AppleStore/fromiPhone8/Developer/usr/bin cp debugserver_orig_arm64 debugserver_arm64_debugable
```

就只有一个架构arm64了

再去查看entitlement，就只有一份entitlement信息了：

```
crifan@licrifandeMacBook-Pro ~ /dev/dev_root/iosReverse/AppleStore/fromiPhone8/Developer/usr/bin ldid -e debugserver_arm64_debugable
<?xml version="1.0" encoding="UTF-8"?
< DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key com.apple.springboard.debugapplications /key>
  <true/>
  <key com.apple.backboardd.launchapplications /key>
  <true/>
  <key com.apple.backboardd.debugapplications /key>
  <true/>
  <key com.apple.frontboard.launchapplications /key>
  <true/>
  <key com.apple.frontboard.debugapplications /key>
  <true/>
  <key com.apple.private.logging.diagnostic /key>
  <true/>
  <key com.apple.private.memorystatus /key>
  <true/>
  <key com.apple.private.cs.debugger /key>
  <true/>
  <key get-task-allow /key>
  <true/>
  <key task_for_pid-allow /key>
  <true/>
  <key run-unsigned-code /key>
  <true/>
</dict>
</plist %
```

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:
2023-10-21 18:33:45

重签名

如前所述，把debugserver的原始entitlement权限做了改动后，需要去：`重签名 = 重新签名`

- 推荐方式：`codesign`
 - 优点：适用于 `iOS 15.0+` 和 `iOS < 15.0`
 - 具体方式
 - 推荐方式：（最省事的）直接一步

```
codesign -f -s - --entitlements debugserver.entitlements debugserver
```

- 等价于=简写为

```
codesign -fs- --entitlements debugserver.entitlements debugserver
```

- 次优方式：分两步
 - 第一步：先找到自己当前的有效的 `sign identity`

```
security find-identity -p codesigning
```

- 得到：`sign identity: CDDC8C0D2F3A79EB17C183E14F799F75815E294E`

- 第二步：再加上完整的参数，去用codesign重签名

```
codesign --force --sign CDDC8C0D2F3A79EB17C183E14F799F75815E294E --entitlements debugable_entitlement.xml --timestamp=none --generate-entitlement-der debugserver
```

- 当 `iOS < 15.0` ，也可以用：`ldid`

```
ldid -Sdebugserver.entitlements debugserver
```

- 说明
 - `-s` 和 参数（即权限文件：`debugserver.entitlements`）之间没有空格
- 注意：当 `iOS > 15.0` 时，此方式会导致：运行崩溃killed
 - 详见：[运行崩溃killed](#)

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:
2023-10-21 18:23:30

常见问题

Failed to get connection from a remote gdb process

- 现象

用 `debugserver` 去调试抖音：

```
debugserver -x auto 0.0.0.0:20221 /private/var/containers/Bundle/Application/9AB25481-0AD3-435C-A02E-68F9623535BB/Aweme.app/Aweme
```

报错：

```
iPhone7P-1341:~/forDebug root# debugserver -x auto 0.0.0.0:20221 /private/var/containers/Bundle/Application/9AB25481-0AD3-435C-A02E-68F9623535BB/Aweme.app/Aweme
debugserver-@(#)PROGRAM:LLDB PROJECT:lldb-900.3.104
for arm64.
Listening to port 20221 for a connection from 0.0.0.0...
Failed to get connection from a remote gdb process.
Exiting.
```

```
iPhone7P-1341:~/forDebug root# debugserver -x auto 0.0.0.0:20221 /private/var/containers/Bundle/Application/9AB25481-0AD3-435C-A02E-68F9623535BB/Aweme.app/Aweme
debugserver-@(#)PROGRAM:LLDB PROJECT:lldb-900.3.104
for arm64.
Listening to port 20221 for a connection from 0.0.0.0...
Failed to get connection from a remote gdb process.
```

- 原因

(iPhone7P中的) `debugserver` ，多了额外的权限：

- `com.apple.security.network.server`
- `com.apple.security.network.client`
- `seatbelt-profiles`

导致，不允许连接进程（去调试）

- 解决办法：去掉权限
- 具体步骤：用去掉了上述权限：

```
<key>seatbelt-profiles</key>
<array>
  <string>debugserver</string>
</array>
...
<key>com.apple.security.network.server</key>
<true/>
<key>com.apple.security.network.client</key>
<true/>
```

的entitlements文件:

- debugserver_noSecurity.entitlements

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>com.apple.springboard.debugapplications</key>
  <true/>
  <key>com.apple.backboardd.launchapplications</key>
  <true/>
  <key>com.apple.backboardd.debugapplications</key>
  <true/>
  <key>com.apple.frontboard.launchapplications</key>
  <true/>
  <key>com.apple.frontboard.debugapplications</key>
  <true/>
  <key>com.apple.private.logging.diagnostic</key>
  <true/>
  <key>com.apple.private.memorystatus</key>
  <true/>
  <key>com.apple.private.cs.debugger</key>
  <true/>
  <key>get-task-allow</key>
  <true/>
  <key>task_for_pid-allow</key>
  <true/>
  <key>run-unsigned-code</key>
  <true/>
</dict>
</plist>
```

加到debugserver中 == 重新给debugserver签名:

- 推荐用 codesign

```
codesign -f -s - --entitlements debugserver_noSecurity.entitlements debugserver
```

- 或 iOS < 15 也可以用 ldid

```
ldid -Sdebugserver_noSecurity.entitlements debugserver
```

再放到iPhone中==拷贝到iPhone中

```
scp debugserver root@192.168.0.58:/usr/bin
```

即可。

Failed to open log file for writing: errno = 1 (Operation not permitted)

- 现象

用debugserver调试程序，带日志参数：

```
debugserver -l debugservr_20220107_1050.log 0.0.0.0:20221 -a 8829
```

报错：

```
iPhone7P-1341:~ root# debugserver -l debugservr_20220107_1050.log 0.0.0.0:20221 -a 8829
Failed to open log file 'debugservr_20220107_1050.log' for writing: errno = 1 (Operation not permitted)debugserver-@(#)PROGRAM:LLDB PROJECT:lldb-900.3.104
for arm64.
...
```

- 原因：

- 表面原因：debugserver没有写入（日志文件的）权限
- 深层次原因：当前 debugserver 的权限entitlements中有：

```
<key>seatbelt-profiles</key>
<array>
  <string>debugserver</string>
</array>
```

- 意思是，给debugserver开启了Sandbox
 - 导致：没有（各种的，包括文件）写入的权限

- 解决办法：把debugserver的entitlements权限中去掉：

```
<key>seatbelt-profiles</key>
<array>
  <string>debugserver</string>
</array>
```

- 注：再重签名debugserver，写回iPhone，即可。

Segmentation fault: 11

- 现象

debugserver调试抖音，报错：

```
iPhone7P-1341:~ root# debugserver 0.0.0.0:20221 -a 8829
debugserver-@(#)PROGRAM:LLDB PROJECT:lldb-900.3.104
for arm64.
Attaching to process 8829...
Segmentation fault: 11
```

- 原因：此处的iOS的app，抖音，内部做了反调试，使得此处调试中断，无法继续调试
- 解决办法：深入逆向对应app，找到反调试的逻辑，进行反反调试
 - 概述

- 此处抖音的反调试手段是：二进制 AwemeCore 中用 svc 0x80 的内联汇编实现的 syscall 的 ptrace 的 PT_DENY_ATTACH
- 此处反反调试=破解反调试的手段是：把 AwemeCore 中的 svc 0x80 指令替换成 空指令 = NOP 指令
- 具体详见
 - [反调试和反反调试 · iOS逆向开发：动态调试](#)

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:
2023-10-21 18:16:11

运行崩溃killed

重签名后的debugserver运行崩溃：killed

现象

表面现象

之前用：

```
ldid -Sdebugable_entitlement.xml debugserver_debugable
```

给debugserver重签名，加上了额外的可以被调试的权限。

注：此方法，之前在 iOS 14 中，是有效的。

但是此处 iOS 15 中，报错了：运行崩溃killed

```
iPhone8-150:/Developer/usr/bin root# debugserver --version
zsh: killed      debugserver --version


iPhone8-150:/usr/bin root# /usr/bin/debugserver --help
zsh: killed      /usr/bin/debugserver --help
```

```
root@192.168.2.13 (ssh) 361 ..:/OS/palera1n (-zsh) 362 ..S/debugserver (-zsh) 363 ..loper/usr/bin (-zsh) 364 +
60% 9.6 GB 0.0 kB↓ 0.0 kB↑ 5/05, 17:32
-rwxr-xr-x 1 root wheel 69K Feb 23 11:58 tset*
-rwxr-xr-x 1 root wheel 105K May 5 15:03 tsort*
-rwxr-xr-x 1 root wheel 104K May 5 15:03 tty*
-rwxr-xr-x 1 root wheel 70K May 5 15:03 uicache*
lrwxr-xr-x 1 root wheel 19 May 5 15:03 uiduid -> /usr/bin/deviceinfo*
-rwxr-xr-x 1 root wheel 68K May 5 15:03 uiopen*
-rwxr-xr-x 1 root wheel 70K Sep 16 2021 umtool*
-rwxr-xr-x 1 root staff 104K May 5 15:03 unname*
-rwxr-xr-x 1 root wheel 105K May 5 15:03 unexpand*
-rwxr-xr-x 1 root wheel 105K May 5 15:03 uniq*
-rwxr-xr-x 1 root wheel 105K May 5 15:03 unlink*
-rwxr-xr-x 1 root wheel 86K Feb 21 04:56 update-alternatives*
-rwxr-xr-x 1 root wheel 8.9K May 5 15:03 updatedb*
-rwxr-xr-x 1 root wheel 106K May 5 15:03 uptime*
-rwxr-xr-x 1 root wheel 105K May 5 15:03 users*
-rwxr-xr-x 1 root staff 193K May 5 15:03 vdir*
lrwxr-xr-x 1 root wheel 23 May 5 15:03 vedit -> /etc/alternatives/vedit*
lrwxr-xr-x 1 root wheel 20 May 5 15:03 vi -> /etc/alternatives/vi*
lrwxr-xr-x 1 root wheel 22 May 5 15:03 view -> /etc/alternatives/view*
-rwxr-xr-x 1 root wheel 34K Sep 16 2021 vm_stat*
-rwxr-xr-x 1 root wheel 68K Nov 16 11:17 w*
-rwxr-xr-x 1 root wheel 122K May 5 15:03 wc*
-rwxr-xr-x 1 root wheel 51K Nov 16 11:17 what*
-rwxr-xr-x 1 root wheel 52K Nov 16 11:17 whereis*
-rwxr-xr-x 1 root wheel 51K Nov 16 11:17 which*
-rwxr-xr-x 1 root wheel 106K May 5 15:03 who*
-rwxr-xr-x 1 root wheel 104K May 5 15:03 whoami*
-rwxr-xr-x 1 root wheel 107K May 5 15:03 xargs*
-rwxr-xr-x 1 root wheel 52K Nov 23 02:15 xattr*
-rwxr-xr-x 1 root wheel 104K May 5 15:03 yes*
-rwxr-xr-x 1 root wheel 69K Sep 16 2021 zprint*
-rwxr-xr-x 1 root staff 689K May 5 15:03 zsh*
iPhone8-150:/usr/bin root# mkdir test
mkdir: cannot create directory 'test': File exists
iPhone8-150:/usr/bin root# mkdir testDir
iPhone8-150:/usr/bin root# rm -rf testDir
iPhone8-150:/usr/bin root# ls -lh | grep debugserver
-rwxr-xr-x 1 root wheel 9.4M May 5 17:13 debugserver*
iPhone8-150:/usr/bin root# /usr/bin/debugserver --version
zsh: killed /usr/bin/debugserver --version
iPhone8-150:/usr/bin root# rm -rf ./debugserver
iPhone8-150:/usr/bin root# /usr/bin/debugserver --version
zsh: killed /usr/bin/debugserver --version
iPhone8-150:/usr/bin root# /Developer/usr/bin/debugserver --version
debugserver-@(#)PROGRAM:LLDB PROJECT:Lldb-1300.2.10
for arm64.
iPhone8-150:/usr/bin root# ls -lh /Developer/usr/bin/debugserver
-rwxr-xr-x 1 root admin 1.3M Aug 8 2021 /Developer/usr/bin/debugserver*
iPhone8-150:/usr/bin root# rm -rf /usr/bin/debugserver
iPhone8-150:/usr/bin root# /usr/bin/debugserver --version
zsh: killed /usr/bin/debugserver --version
iPhone8-150:/usr/bin root# /usr/bin/debugserver --help
zsh: killed /usr/bin/debugserver --help
```

深层次现象

debugserver 的崩溃日志 .ips 文件:

崩溃日志1

无SIM卡 

上午11:58



[分析与改进](#)

数据

awdd-2023-03-02-144026-39.con... >

awdd-2023-03-02-144026-40.con... >

bird.cpu_resource-2023-02-28-102... >

bird.diskwrites_resource-2023-02-... >

CloudServicesTopic-2023-02-27-0... >

CloudServicesTopic-2023-03-02-16... >

debugserver-2023-03-03-115432.ips >

DifferentialPrivacy_2023-02-27-145... >

ExcUserFault_webbookmarksd-202... >

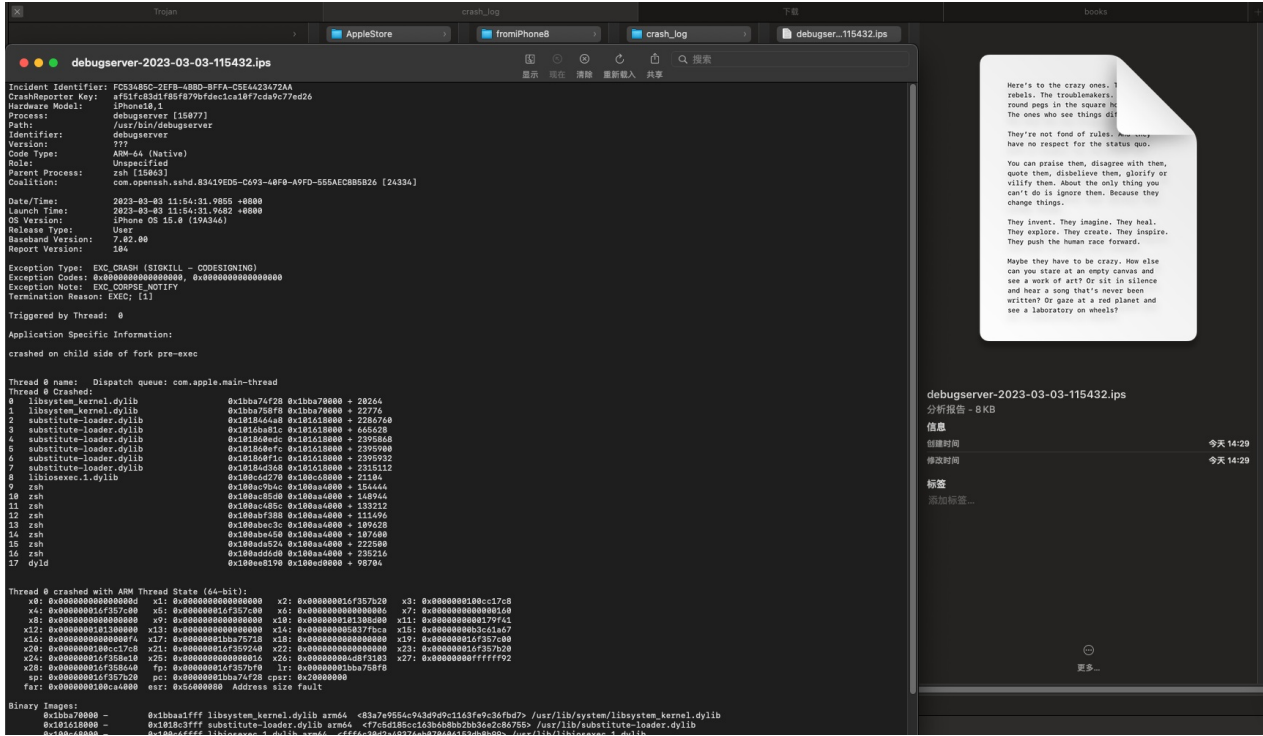
ExcUserFault_webbookmarksd-202... >

ExcUserFault_webbookmarksd-202... >

ExcUserFault_wifid-2023-02-28-10... >

ExcUserFault_wifid-2023-02-28-10... >

可以看到具体崩溃原因是：



```

Incident Identifier  FC53485C-2EFB-4BBD-BFFA-C5E4423472AA
CrashReporter Key   af51fc83d1f85f879bfdec1ca10f7cda9c77ed26
Hardware Model      iPhone10,1
Process             debugserver [15077]
Path                /usr/bin/debugserver
Identifier          debugserver
Version             ???
Code Type           ARM-64 (Native)
Role                Unspecified
Parent Process      zsh [15063]
Coalition           com.openssh.sshd.83419ED5-C693-40F0-A9FD-555AEC8B5B26 [24334]

Date/Time           2023-03-03 11 54 31.9855 +0800
Launch Time         2023-03-03 11 54 31.9682 +0800
OS Version          iPhone OS 15.0 (19A346)
Release Type        User
Baseband Version    7.02.00
Report Version      104

Exception Type      EXC_CRASH (SIGKILL - CODESIGNING)
Exception Codes     0x0000000000000000, 0x0000000000000000
Exception Note      EXC_CORPSE_NOTIFY
Termination Reason  EXEC; [1]

Triggered by Thread 0

Application Specific Information:
crashed on child side of fork pre-exec

Thread 0 name       Dispatch queue: com.apple.main-thread

```

Thread 0 Crashed

```

0  libsystem_kernel.dylib          0x1bba74f28 0x1bba70000 + 20264
1  libsystem_kernel.dylib          0x1bba758f8 0x1bba70000 + 22776
2  substitute-loader.dylib         0x1018464a8 0x101618000 + 2286760
3  substitute-loader.dylib         0x1016ba81c 0x101618000 + 665628
4  substitute-loader.dylib         0x101860edc 0x101618000 + 2395868
5  substitute-loader.dylib         0x101860efc 0x101618000 + 2395900
6  substitute-loader.dylib         0x101860f1c 0x101618000 + 2395932
7  substitute-loader.dylib         0x10184d368 0x101618000 + 2315112
8  libioexec.1.dylib              0x100c6d270 0x100c68000 + 21104
9  zsh                              0x100ac9b4c 0x100aa4000 + 154444
10 zsh                              0x100ac85d0 0x100aa4000 + 148944
11 zsh                              0x100ac485c 0x100aa4000 + 133212
12 zsh                              0x100abf388 0x100aa4000 + 111496
13 zsh                              0x100abec3c 0x100aa4000 + 109628
14 zsh                              0x100abe450 0x100aa4000 + 107600
15 zsh                              0x100ada524 0x100aa4000 + 222500
16 zsh                              0x100add6d0 0x100aa4000 + 235216
17 dyld                            0x100ee8190 0x100ed0000 + 98704

```

Thread 0 crashed with ARM Thread State (64-bit)

```

x0 0x000000000000000d  x1 0x0000000000000000  x2 0x000000016f357b20  x3 0x00
00000100cc17c8
x4 0x000000016f357c00  x5 0x000000016f357c00  x6 0x0000000000000006  x7 0x00
00000000000160
x8 0x0000000000000000  x9 0x0000000000000000  x10 0x0000000101308d00  x11 0x00
00000000179f41
x12 0x0000000101300000  x13 0x0000000000000000  x14 0x000000005037fbc8  x15 0x00
000000b3c61a67
x16 0x00000000000000f4  x17 0x00000001bba75718  x18 0x0000000000000000  x19 0x00
0000016f357c00
x20 0x0000000100cc17c8  x21 0x000000016f359240  x22 0x0000000000000000  x23 0x00
0000016f357b20
x24 0x000000016f358e10  x25 0x0000000000000016  x26 0x000000004d8f3103  x27 0x00
000000ffffff92
x28 0x000000016f358640  fp 0x000000016f357bf0  lr 0x00000001bba758f8
sp 0x000000016f357b20  pc 0x00000001bba74f28  cpsr 0x20000000
far 0x0000000100ca4000  esr 0x56000080  Address size fault

```

Binary Images

```

0x1bba70000 - 0x1bbaa1fff libsystem_kernel.dylib arm64 <83a7e9554c943d9d
9c1163fe9c36fbd7> /usr/lib/system/libsystem_kernel.dylib
0x101618000 - 0x1018c3fff substitute-loader.dylib arm64 <f7c5d185cc163b6
b8bb2bb36e2c86755> /usr/lib/substitute-loader.dylib
0x100c68000 - 0x100c6ffff libioexec.1.dylib arm64 <fff6c30d2a49376eb070
606153db8b99> /usr/lib/libioexec.1.dylib
0x100aa4000 - 0x100b33fff zsh arm64 <5f9852c7fc8c37dc81250e39a1bd688b> /
usr/bin/zsh
0x100ed0000 - 0x100f23fff dyld arm64 <d7a0282e93de3a1e981327e84517cc96>
/usr/lib/dyld

```

EOF

其中的 Exception Type: EXC_CRASH (SIGKILL - CODESIGNING) 能看出是：代码签名方面的问题

崩溃日志2

无SIM卡 

下午 2:54



debugserver_arm64-2023-03-...



```

{"app_name":"debugserver_arm64","timestamp":"2023-03-03
14:53:45.00 +0800","app_version":"","slice_uuid":"5f9852c7-
fc8c-37dc-8125-0e39a1bd688b","build_version":"","platform":2,"share_wit
h_app_devs":1,"is_first_party":1,"bug_type":"309","os_version":"iPhone
OS 15.0
(19A346)","incident_id":"A7F99E43-12D0-4D0C-9122-66F43AC3BD7D","n
ame":"debugserver_arm64"}
{
  "uptime" : 210000,
  "procLaunch" : "2023-03-03 14:53:45.3586 +0800",
  "procRole" : "Unspecified",
  "version" : 2,
  "userID" : 0,
  "deployVersion" : 210,
  "modelCode" : "iPhone10,1",
  "procStartAbsTime" : 5268037881478,
  "coalitionID" : 24334,
  "osVersion" : {
    "isEmbedded" : true,
    "train" : "iPhone OS 15.0",
    "releaseType" : "User",
    "build" : "19A346"
  },
  "captureTime" : "2023-03-03 14:53:45.3775 +0800",
  "incident" : "A7F99E43-12D0-4D0C-9122-66F43AC3BD7D",
  "bug_type" : "309",
  "pid" : 15156,
  "procExitAbsTime" : 5268038223259,
  "cpuType" : "ARM-64",
  "procName" : "debugserver_arm64",
  "procPath" : "\usr\bin\debugserver_arm64",
  "parentProc" : "zsh",
  "parentPid" : 15063,
  "coalitionName" : "com.openssh.sshd.83419ED5-C693-40F0-
A9FD-555AEC8B5B26",
  "crashReporterKey" : "af51fc83d1f85f879bfdec1ca10f7cda9c77ed26",
  "basebandVersion" : "7.02.00",
  "isCorpse" : 1,
  "exception" : {"codes":"0x0000000000000000",
0x0000000000000000","rawCodes":
[0,0],"type":"EXC_CRASH","signal":"SIGKILL - CODESIGNING"},
  "termination" : {"flags":64,"code":1,"namespace":"EXEC","indicator":"Bad
Mach-O"},
  "asi" : {"libsystem_c.dylib":["crashed on child side of fork pre-
exec"],"dyld":[]},
  "faultingThread" : 0,
  "threads" : [{"triggered":true,"id":814594,"threadState":{"x":{"value":13},
{"value":0},{"value":6160743200},{"value":4308342752},
{"value":6160743424},{"value":6160743424},{"value":3},{"value":352},
{"value":0}.{"value":0}.{"value":4314927312}.{"value":1548097}.

```

```
"exception" : {"codes":"0x0000000000000000, 0x0000000000000000", "rawCodes":[0,0], "type":"EXC_CRASH", "signal":"SIGKILL - CODESIGNING"},
"termination" : {"flags" 64, "code":1, "namespace":"EXEC", "indicator":"Bad Mach-O"},
"asi" : {"libsystem_c.dylib":["crashed on child side of fork pre-exec"], "dyld":[]},
```

其中的:

- "signal":"SIGKILL - CODESIGNING"} : 确定是: 代码签名方面的问题, 导致的崩溃
- "indicator":"Bad Mach-O" : 具体指示器 ~ = 出错的地方/来源: 是由于 Bad Mach-O =Mach-O二进制文件是坏的
 - 后续证明: 是签名方面有问题 -> 所以是: 坏的Mach-O文件

原因

iOS <15 时的旧的重签名的方式=ldid重签名, 不满足此处 (arm64 的 A11 的 iPhone8 , iOS 15.0) 新的 iOS 15+ 的要求

解决办法

用 codesign 重新签名

具体步骤

- 概述

```
codesign -f -s - --entitlements debugable_entitlement.xml debugserver
```

- 详解
 - [重签名](#)

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: 2023-10-21 18:27:48

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

- 【已解决】 debugserver调试iOS抖音报错： Failed to get connection from a remote gdb process
- 【未解决】 iOS中debugserver调试报错： Failed to get connection from a remote gdb process
- 【已解决】 Mac中如何用lldb调试iPhone中的app
- 【已解决】 debugserver启动iOS的app抖音报错： Segmentation fault 11
- 【已解决】 debugserver带日志运行报错： Failed to open log file for writing errno 1 Operation not permitted
- 【已解决】 用debugserver启动iPhone中抖音app
- 【已解决】 iOS的debugserver的语法和作用
- 【已解决】 ldid加了entitlement权限后debugserver运行崩溃： Bad Mach-O EXC_CRASH SIGKILL CODESIGNING
- 【已解决】 ldid查看二进制的entitlement时输出2份相同的信息
- 【已解决】 debugserver加了权限后放回iPhone8无法运行而崩溃报错killed
-
- [使用LLDB远程调试iOS程序 · 大专栏 \(dazhuanlan.com\)](#)
- [实战：干掉高德地图7.2.0版iOS客户端的反动态调试保护 - 干货分享 | Blog - iOSRE](#)
- [debugserver.cpp \(apple.com\)](#)
-

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:
2023-10-21 18:28:12