目录

[Android命令和IOS命令对照关系 3](#_Toc468127939)

[Mac/iOS环境配置 4](#_Toc468127940)

[分析工具 7](#_Toc468127941)

[Class-Dump用法 7](#_Toc468127942)

[IDA反汇编 8](#_Toc468127943)

[寻找oc函数调用栈 8](#_Toc468127944)

[正常显示unicode中文字符 11](#_Toc468127945)

[Radare2反汇编 12](#_Toc468127946)

[r2调试器 12](#_Toc468127947)

[rax 数字转换器 12](#_Toc468127948)

[rabin2 二进制解析 13](#_Toc468127949)

[radiff2 二进制及函数比较 14](#_Toc468127950)

[rafind2 字串搜索 15](#_Toc468127951)

[rahash2 基于块的哈希 15](#_Toc468127952)

[rasm2 反汇编 15](#_Toc468127953)

[Debug for mac&ios 21](#_Toc468127954)

[lldb调试 21](#_Toc468127955)

[搭建环境 21](#_Toc468127956)

[LLDB调试 22](#_Toc468127957)

[IDA调试 24](#_Toc468127958)

[Gikdbg调试 25](#_Toc468127959)

[CGDB调试 27](#_Toc468127960)

[Radare调试 28](#_Toc468127961)

[Trace/Hook for mac/ios 29](#_Toc468127962)

[系统支持 29](#_Toc468127963)

[OC支持 30](#_Toc468127964)

[Frida 31](#_Toc468127965)

[Snoop-it/IntroSpy 34](#_Toc468127966)

[Cycript 36](#_Toc468127967)

[基本功能 36](#_Toc468127968)

[OC运行时功能 36](#_Toc468127969)

[调试功能 36](#_Toc468127970)

[Hook功能 37](#_Toc468127971)

[其他模块 37](#_Toc468127972)

[JSPatch 38](#_Toc468127973)

[procexp-进程监视 39](#_Toc468127974)

[fsmon-文件系统监视 40](#_Toc468127975)

[iOS实例分析 41](#_Toc468127976)

[准备工作 41](#_Toc468127977)

[Inject 41](#_Toc468127978)

[Hook 43](#_Toc468127979)

[Jailbreak Development Tools 46](#_Toc468127980)

[Theos 46](#_Toc468127981)

[IOSOpenDev 46](#_Toc468127982)

[Mac&iOS file format analysis 47](#_Toc468127983)

[砸壳 48](#_Toc468127984)

[mach-o格式分析 49](#_Toc468127985)

[App目录和文件 51](#_Toc468127986)

[Objective C Reversing 52](#_Toc468127987)

[杂项 52](#_Toc468127988)

[Function 53](#_Toc468127989)

[Block 53](#_Toc468127990)

[Class 57](#_Toc468127991)

[内部结构体一览 57](#_Toc468127992)

[成员函数分析 60](#_Toc468127993)

[Objc\_msgSend调用流程 61](#_Toc468127994)

[Runtime Ability 63](#_Toc468127995)

[其他 65](#_Toc468127996)

[异常处理 67](#_Toc468127997)

[Reflection 68](#_Toc468127998)

[重要的库和函数 69](#_Toc468127999)

[iOS Attack&Defense 70](#_Toc468128000)

[AntiDebug - AntiAntiDebug 70](#_Toc468128001)

[sysctl P\_TRACED标志 检测调试 70](#_Toc468128002)

[ptrace PT\_DENY\_ATTACH 防止调试 70](#_Toc468128003)

[isatty检测调试 71](#_Toc468128004)

[\_RESTRICT节——防注入 72](#_Toc468128005)

[JailBreak Detect – Anti JailBreak Detect 73](#_Toc468128006)

[沙盒完整性检测 73](#_Toc468128007)

[文件系统检测 73](#_Toc468128008)

[URL Scheme检测 74](#_Toc468128009)

[系统内核环境变量检测 74](#_Toc468128010)

[运行进程检测 75](#_Toc468128011)

[Injection Detect ——检测注入 76](#_Toc468128012)

[Function Hook Sinature ——检测hook 76](#_Toc468128013)

[其他 78](#_Toc468128014)

[工具扩展 78](#_Toc468128015)

[mach-o补丁 78](#_Toc468128016)

[显示ios app表 78](#_Toc468128017)

[定位关键函数 78](#_Toc468128018)

[LLDB常用命令 78](#_Toc468128019)

[iOS app安全编码 89](#_Toc468128020)

[libimobiledevice工具集 91](#_Toc468128021)

[常用cycript函数 92](#_Toc468128022)

# Android命令和IOS命令对照关系

IOS工具位置：/Applications/Xcode.app/Contents/Developer/usr/bin/

|  |  |  |
| --- | --- | --- |
|  | Android命令 | iOS命令 |
| 安装应用 | adb install -r <apk path> | 真机安装：  fruitstrap  -b  UCWEB.app／XXX.ipa  模拟器安装：  xcrun simctl install booted <XXX.app/XXX.ipa>  ideviceinstaller -i |
| 卸载应用 | adb uninstall -k <package name> | xcrun simctl erase [device ID]  ideviceinstaller -u |
| 查看设备 | adb devices | instruments -s  devices  xcrun simctl list |
| 打开进程 | Adb shell ./?? | open\* |
| 端口转发 | Adb forward | Tcprelay.py |
|  |  |  |

\*ios使用framework的app，会访问保护服务，因此直接./方式执行会被杀死；open [bundleid]方式也经常会有失败

# Mac/iOS环境配置

安装brew

/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

安装wget

brew install binutils

brew install wget

brew install python

安装pip

wget https://bootstrap.pypa.io/get-pip.py

sudo su

python get-pip.py

pip install -U pip

安装应用

pip install frida

brew install gcc

brew install llvm

brew install automake

brew install cmake

brew install git

brew install gdbre

https://sourceware.org/gdb/wiki/BuildingOnDarwin

codesign -s gdb-cert /usr/local/bin/gdb

为方便window编译mac中得m文件：

1.mac中开启smb文件共享，使得win中可以以\\192.168.133.128访问

2.mac中开启远程登录，使得win中可以以ssh user@192.168.133.128登录

3.ssh连接后执行命令行编译

xcodebuild –showsdks列举sdk

OS X SDKs:

OS X 10.9 -sdk macosx10.9

OS X 10.10 -sdk macosx10.10

iOS SDKs:

iOS 8.4 -sdk iphoneos8.4

iOS Simulator SDKs:

Simulator - iOS 8.4 -sdk iphonesimulator8.4

单文件编译

Compile & Link for mac

gcc main.m -o main -ObjC -framework Foundation -framework CoreLocation or

clang main.m -o main -ObjC -framework Foundation -framework CoreLocation

Compile for iOS

clang -x objective-c -arch armv7 -fmessage-length=0 -fdiagnostics-print-source-range-info -fdiagnostics-show-category=id -fdiagnostics-parseable-fixits -std=gnu99 -Wno-trigraphs -fpascal-strings -O0 -Wmissing-prototypes -Wreturn-type -Wparentheses -Wswitch -Wno-unused-parameter -Wunused-variable -Wunused-value -DDEBUG=1 -isysroot /Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS4.3.sdk -gdwarf-2 -mthumb "-DIBOutlet=attribute((iboutlet))" "-DIBOutletCollection(ClassName)=attribute((iboutletcollection(ClassName)))" "-DIBAction=void)attribute((ibaction)" -miphoneos-version-min=3.2 -iquote [a bookkeeping file] -I[a list of headers] -iquote [more headers] -I[an include path] -fpch-preprocess -F[pointer to directory for debug files] -include [my prefix header] -c AppDelegate.m -o AppDelegate.o

Link for iOS

/Developer/Platforms/iPhoneOS.platform/Developer/usr/bin/clang -arch armv6 -isysroot /Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS4.3.sdk -L[something appropriate] -F[something appropriate] -filelist [a .LinkFileList] -dead\_strip -miphoneos-version-min=3.2 -framework SystemConfiguration -framework UIKit -framework Foundation -framework CoreGraphics -o [something appropriate]

多文件编译，写Makefile

安装CYDIA工具：

OpenSSH 基本命令

Tcpdump

AppSync 绕过系统验证，随意安装ipa

Apple File Conduit 安装后可在手机助手显示系统目录

samba windows 文件共享

syslog 日志存放在/var/log/syslog

安装python pip

Ncdu du command

Lsof lsof command

File file command

Less less command

Cyscript

Apt struct 提供apt-get

Adv-cmds finger fingerd lsvfs last md ps

File-cmds chflags compress ipcrm ipcs pax

Basic-cmds msg uudecode uuencode write

Shell-cmds killall mktemp renice time which

System-cmds iostat login passwd sync sysctl

Diskdev-cmds mount quota fsck fstyp fdisk tunefs

Network arp ifconfig netstat route traceroute

Syslog syslogon syslogoff /var/log/syslog

Wget

GNU Debugger ar nm objdump ranlib strip addr2line c++filt gdb objcopy objdump readelf

(compile your code with –mcpu=arm1176jzf-s)

CYDIA常用源：

http://apt.thebigboss.org

http://apt.saurik.com

http://apt.modmyi.com

http://repo666.ultrasn0wn.com

http://ctdua.zodttd.com

http://apt.weifeng.com

http://apt.feng.com

http://repo.feng.com

http://repo.xarold.com

http://julio.xarold.com

http://crak.cn/repo/

http://iphone.tgbus.com/cydia/

https://build.frida.re

ios刷机工具：爱思助手

ios越狱工具：盘古

越狱后连接iphone

python tcprelay –t 22:22 111:111

ssh root@127.0.0.1

# 分析工具

|  |  |  |  |
| --- | --- | --- | --- |
|  | Android | Mac | iOS |
| 跟踪工具 | strace ltrace introspy | dtruss dtrace | Frida cycript introspy |
| 文件操作 | adb push/pop |  | ssh scp |
| 日志 | Logcat | idevicesyslog | Syslog |
| 调试工具 | Gdb jdb ida gkidbg | Gdb ida lldb | Gdb Ida lldb gikdbg |
| Hook框架 | Cydia Xposed |  | Cydia |
| 静态分析 | Ida dex2jar apktool jadx jeb jd-gui | Ida classdump reveal iNalyzer hopper | Ida classdump reveal iNalyzer hopper |
| 其他 |  | fs\_usage sc\_usage vmmap | dumpencrypted clutch |

## Class-Dump用法

Class-dump是mac上的命令行工具用于解析Objective-C类接口，class-dump-z修复了一些bug并使用c++重写从而在mac, linux, win平台上运行，不支持x64 iphone，因此如果要解析mac os x程序的类，要用原始class-dump，而解析iphone的类使用class-dump-z

class-dump:

运行于mac的工具用于解析objectc 运行时信息，生成classes, categories protocols，和otool –ov的结果类似，以objectivec语法表示更可读

Usage: class-dump [options] <mach-o-file>

where options are:

-a show instance variable offsets

-A show implementation addresses

--arch <arch> choose a specific architecture from a universal binary (ppc, ppc64, i386, x86\_64)

-C <regex> only display classes matching regular expression

-f <str> find string in method name

-H generate header files in current directory, or directory specified with -o

-I sort classes, categories, and protocols by inheritance (overrides -s)

-o <dir> output directory used for -H

-r recursively expand frameworks and fixed VM shared libraries

-s sort classes and categories by name

-S sort methods by name

-t suppress header in output, for testing

--list-arches list the arches in the file, then exit

--sdk-ios specify iOS SDK version (will look in /Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS<version>.sdk

--sdk-mac specify Mac OS X version (will look in /Developer/SDKs/MacOSX<version>.sdk

--sdk-root specify the full SDK root path (or use --sdk-ios/--sdk-mac for a shortcut)

Class-dump-z：

速度快，便携且兼容各系统，修正ivar偏移处理，结构体名可读性高，属性化，隐藏继承和代理方法，参数名可读，修正头文件生成等

Usage: class-dump-z [<options>] <filename>

Options are:

分析：

-p 转换未声明的getters setters为属性

-h proto 隐藏protocol已有的method

-h super 隐藏继承method

-y <root> 指定sysroot，默认为iPhoneOS SDK

-u <arch> 指定架构(armv6, armv7,…)

格式化：

-a 打印成员变量偏移

-A 打印implementation VM地址

-k –K 附加注释

-R 指针显示方式为int \*a而不是int\* a

-N 保留原始结构体名\_为\_CFArray\*而不是 CFArrayRef

-b +/-号前空格

-i 读取更新signature hints文件

过滤：

-C <regex> 类型匹配正则表达式

-f <regex> 方法匹配正则表达式

-g 只显示导出类

-X <list> 忽略的类型列表

-h cats 隐藏categories

-h dogs 隐藏protocols

排序：

-S 类型按字母表排序

-s 方法按字母表排序

-z 方法按字母表排序，而类方法和-init…最前

输出：

-H 导出为不同头文件

-o <dir> 指定导出目录

若由于AppStore加密等原因无法直接用classdump导出类的情况下，可以用cycript脚本weak\_classdump在运行时进行同等操作，源码在最后一章

## IDA反汇编

逆向工具，用法不再赘述

### 寻找oc函数调用栈

对于OC语法由于是通过消息机制进行函数调用的，因此无法直接找到调用者，这里通过脚本解决：

# -\*- coding:utf-8 -\*-

################################################################

#

# Copyright (c) 2016 Baidu.com, Inc. All rights Reserved

#

################################################################

"""

this module add objc-xref for IDA in analysing macho-file

Authors: lichao(lichao26@baidu.com)

Date: 2016/09/23

"""

import idc

def addxref(x, y, z):

"""

add reference for objc\_meth\_addr <=> objc\_methname\_addr <=> msgsend\_call\_addr

:param x: msgsend\_call\_addr

:param y: objc\_meth\_addr

:param z: objc\_methname\_addr

:return: nothing

"""

AddCodeXref(x, y, XREF\_USER | fl\_F)

# AddCodeXref(y, x, XREF\_USER | fl\_F)

# AddCodeXref(x, z, XREF\_USER | fl\_F)

# AddCodeXref(z, x, XREF\_USER | fl\_F)

AddCodeXref(y, z, XREF\_USER | fl\_F)

# AddCodeXref(z, y, XREF\_USER | fl\_F)

def addobjcref():

"""

add reference for math-o file

:return: nothing

"""

objc\_meth\_map = {}

methnamebegin = 0

methnameend = 0

forbitmeth = [

"alloc",

"allocWithZone:",

"allowsWeakReference",

"autorelease",

"class",

"conformsToProtocol:",

"copy",

"copyWithZone:",

"dealloc",

"debugDescription",

"description",

"doesNotRecognizeSelector:",

"finalize",

"forwardingTargetForSelector:",

"forwardInvocation:",

"hash",

"init",

"initialize",

"instanceMethodForSelector:"

"instanceMethodSignatureForSelector:",

"instancesRespondToSelector:",

"isEqual",

"isKindOfClass:",

"isMemberOfClass:",

"isProxy",

"isSubclassOfClass:",

"load",

"methodForSelector:",

"methodSignatureForSelector:",

"mutableCopy",

"mutableCopyWithZone:",

"performSelector:",

"performSelector:withObject:",

"performSelector:withObject:withObject:",

"respondsToSelector:",

"release",

"resolveClassMethod:",

"resolveInstanceMethod:",

"retain",

"retainCount",

"retainWeakReference",

"superclass",

"zone",

".cxx\_construct",

".cxx\_destruct",

]

# find the segment which contains objc method names

curseg = FirstSeg()

while curseg != 0xffffffff:

if "\_\_objc\_methname" == SegName(curseg):

methnamebegin = SegStart(curseg)

methnameend = SegEnd(curseg)

break

curseg = NextSeg(curseg)

# get objc method names

if methnamebegin != 0:

while methnamebegin < methnameend:

funcname = GetString(methnamebegin)

objc\_meth\_map[funcname] = methnamebegin

methnamebegin = methnamebegin + len(funcname) + 1

# get objc func table

funcmap = {}

addr = PrevFunction(-1)

while addr != 0xffffffff:

curname = GetFunctionName(addr)

if -1 != curname.find('['):

curname = curname.replace("[", "").replace("]", "")

curname = curname.split(" ")[1]

# may be more than one function with same sel but differenct class

if curname not in funcmap:

funcmap[curname] = []

funcmap[curname].append(addr)

addr = PrevFunction(addr)

# make xref

for (k, v) in objc\_meth\_map.items():

# find corresponding func addr

if k in funcmap and k not in forbitmeth:

farr = funcmap[k]

# find xref to code and make xref for each

curref = DfirstB(v)

while curref != 0xffffffff:

for f in farr:

addxref(curref, f, v)

curref = DnextB(v, curref)

print "added xref for " + k

if \_\_name\_\_ == "\_\_main\_\_":

addobjcref()

### 正常显示unicode中文字符

由于ida使用python对中文支持不好，这里通过脚本解决一定程度的问题：

def find\_utf16\_string(addr):

start = SegStart(addr)

end = SegEnd(addr)

addr = start

while addr < end:

# get length

len = 1

while Name(addr + len) == "":

len = len + 1

totalstr = ""

for i in range(0, len, 2):

if Word(addr + i) > 0x100:

# read an unicode char

bytes = GetString(addr + i, 2)

try: # some chinese character not supported by python

comm = bytes.decode("utf-16")

if type(comm) == unicode:

comm = comm.encode("gbk")

else:

comm = '?'

except Exception as e:

comm = '?'

else:

# extract as ascii

comm = chr(Word(addr + i))

totalstr = totalstr + comm

MakeComm(addr, totalstr)

addr = addr + len

tofind = ["\_\_ustring"]

seg = FirstSeg()

while seg != 0xffffffff:

if SegName(seg) in tofind:

find\_utf16\_string(seg)

seg = NextSeg(seg)

## Radare2反汇编

功能：反汇编工具集

安装：添加radare源：http://cydia.radare.org 安装radare2-arm32即可

### r2调试器

### rax 数字转换器

Usage: rax2 [options] [expr ...]

=[base] ; rax2 =10 0x46 -> output in base 10

int -> hex ; rax2 10

hex -> int ; rax2 0xa

-int -> hex ; rax2 -77

-hex -> int ; rax2 0xffffffb3

int -> bin ; rax2 b30

int -> ternary ; rax2 t42

bin -> int ; rax2 1010d

float -> hex ; rax2 3.33f

hex -> float ; rax2 Fx40551ed8

oct -> hex ; rax2 35o

hex -> oct ; rax2 Ox12 (O is a letter)

bin -> hex ; rax2 1100011b

hex -> bin ; rax2 Bx63

hex -> ternary ; rax2 Tx23

raw -> hex ; rax2 -S < /binfile

hex -> raw ; rax2 -s 414141

-b bin -> str ; rax2 -b 01000101 01110110

-B str -> bin ; rax2 -B hello

-d force integer ; rax2 -d 3 -> 3 instead of 0x3

-e swap endianness ; rax2 -e 0x33

-E base64 encode ;

-f floating point ; rax2 -f 6.3+2.1

-F stdin slurp C hex ; rax2 -F < shellcode.c

-h help ; rax2 -h

-k keep base ; rax2 -k 33+3 -> 36

-K randomart ; rax2 -K 0x34 1020304050

-n binary number ; rax2 -n 0x1234 # 34120000

-N binary number ; rax2 -N 0x1234 # \x34\x12\x00\x00

-r r2 style output ; rax2 -r 0x1234

-s hexstr -> raw ; rax2 -s 43 4a 50

-S raw -> hexstr ; rax2 -S < /bin/ls > ls.hex

-t tstamp -> str ; rax2 -t 1234567890

-x hash string ; rax2 -x linux osx

-u units ; rax2 -u 389289238 # 317.0M

-w signed word ; rax2 -w 16 0xffff

-v version ; rax2 –v

Example:

rax2 10 => 0xa

rax2 33 0x41 0101b =>0x21 65 0x5

rax2 –s 4142434445 =>ABCDE

rax2 0x5\*101b+5 =>30

### rabin2 二进制解析

Usage: rabin2 [-AcdeEghHiIjlLMqrRsSvVxzZ] [-@ addr] [-a arch] [-b bits]

[-B addr] [-C F:C:D] [-f str] [-m addr] [-n str] [-N m:M]

[-o str] [-O str] [-k query] [-D lang symname] | file

-@ [addr] show section, symbol or import at addr

-A list sub-binaries and their arch-bits pairs

-a [arch] set arch (x86, arm, .. or <arch>\_<bits>)

-b [bits] set bits (32, 64 ...)

-B [addr] override base address (pie bins)

-c list classes

-C [fmt:C:D] create [elf,mach0,pe] with Code and Data hexpairs (see -a)

-d show debug/dwarf information

-D lang name demangle symbol name (-D all for bin.demangle=true)

-e entrypoint

-E globally exportable symbols

-f [str] select sub-bin named str

-F [binfmt] force to use that bin plugin (ignore header check)

-g same as -SMResiz (show all info)

-G [addr] load address . offset to header

-h this help message

-H header fields

-i imports (symbols imported from libraries)

-I binary info

-j output in json

-k [sdb-query] run sdb query. for example: '\*'

-K [algo] calculate checksums (md5, sha1, ..)

-l linked libraries

-L list supported bin plugins

-m [addr] show source line at addr

-M main (show address of main symbol)

-n [str] show section, symbol or import named str

-N [min:max] force min:max number of chars per string (see -z and -zz)

-o [str] output file/folder for write operations (out by default)

-O [str] write/extract operations (-O help)

-p show physical addresses

-P show debug/pdb information

-PP download pdb file for binary

-q be quiet, just show fewer data

-qq show less info (no offset/size for -z for ex.)

-Q show load address used by dlopen (non-aslr libs)

-r radare output

-R relocations

-s symbols

-S sections

-u unfiltered (no rename duplicated symbols/sections)

-v display version and quit

-V Show binary version information

-x extract bins contained in file

-X [fmt] [f] .. package in fat or zip the given files andbins contained in file

-z strings (from data section)

-zz strings (from raw bins [e bin.rawstr=1])

-zzz dump raw strings to stdout (for huge files)

-Z guess size of binary program

Environment:

RABIN2\_LANG: e bin.lang # assume lang for demangling

RABIN2\_NOPLUGINS: # do not load shared plugins (speedup loading)

RABIN2\_DEMANGLE=0:e bin.demangle # do not demangle symbols

RABIN2\_MAXSTRBUF: e bin.maxstrbuf # specify maximum buffer size

RABIN2\_STRFILTER: e bin.strfilter # r2 -qe bin.strfilter=? -c '' --

RABIN2\_STRPURGE: e bin.strpurge # try to purge false positives

RABIN2\_DMNGLRCMD: e bin.demanglercmd # try to purge false positives

RABIN2\_PREFIX: e bin.prefix # prefix symbols/sections/relocs with a specific string

Example:

rabin2 –I /bin/bash 打印导出表

…………………..

ordinal=319 plt=0x00061f1c bind=NONE type=FUNC name=wcsncmp

ordinal=320 plt=0x00061f28 bind=NONE type=FUNC name=wcsrtombs

ordinal=321 plt=0x00061f34 bind=NONE type=FUNC name=wctob

ordinal=322 plt=0x00061f40 bind=NONE type=FUNC name=wctype

ordinal=323 plt=0x00061f4c bind=NONE type=FUNC name=where\_history

ordinal=324 plt=0x00061f58 bind=NONE type=FUNC name=write

ordinal=325 plt=0x00061f64 bind=NONE type=FUNC name=write\_history

…………………………………

rabin2 –e /bin/bash 打印入口点

[Entrypoints]

vaddr=0x00008000 paddr=0x00004000 baddr=0x00004000 laddr=0x00000000 type=program

rabin2 –zz /bin/bash 打印字符串

……………………………

vaddr=0x000891ea paddr=0x000891ea ordinal=12304 sz=6 len=5 section=unknown type=ascii string=\_read

vaddr=0x000891f0 paddr=0x000891f0 ordinal=12305 sz=14 len=13 section=unknown type=ascii string=\_read\_history

vaddr=0x000891fe paddr=0x000891fe ordinal=12306 sz=20 len=19 section=unknown type=ascii string=\_read\_history\_range

vaddr=0x00089212 paddr=0x00089212 ordinal=12307 sz=9 len=8 section=unknown type=ascii string=\_readdir

vaddr=0x0008921b paddr=0x0008921b ordinal=12308 sz=10 len=9 section=unknown type=ascii string=\_readline

vaddr=0x00089225 paddr=0x00089225 ordinal=12309 sz=10 len=9 section=unknown type=ascii string=\_readlink

…………………….

### radiff2 二进制及函数比较

Usage: radiff2 [-abcCdjrspOxvV] [-g sym] [-t %] [file] [file]

-a [arch] specify architecture plugin to use (x86, arm, ..)

-A [-A] run aaa or aaaa after loading each binary (see -C)

-b [bits] specify register size for arch (16 (thumb), 32, 64, ..)

-c count of changes

-C graphdiff code (columns: off-A, match-ratio, off-B) (see -A)

-d use delta diffing

-D show disasm instead of hexpairs

-g [sym|off1,off2] graph diff of given symbol, or between two offsets

-j output in json format

-n print bare addresses only (diff.bare=1)

-O code diffing with opcode bytes only

-p use physical addressing (io.va=0)

-r output in radare commands

-s compute text distance

-ss compute text distance (using levenstein algorithm)

-S [name] sort code diff (name, namelen, addr, size, type, dist) (only for -C or -g)

-t [0-100] set threshold for code diff (default is 70%)

-x show two column hexdump diffing

-v show version information

-V be verbose (current only for -s)

### rafind2 字串搜索

Usage: D:\radare2-w32-1.0\rafind2 [-mXnzZhv] [-a align] [-b sz] [-f/t from/to] [-[m|s|S|e] str] [-x hex] file ..

-a [align] only accept aligned hits

-b [size] set block size

-e [regex] search for regular expression string matches

-f [from] start searching from address 'from'

-h show this help

-m magic search, file-type carver

-M [str] set a binary mask to be applied on keywords

-n do not stop on read errors

-r print using radare commands

-s [str] search for a specific string (can be used multiple times)

-S [str] search for a specific wide string (can be used multiple times)

-t [to] stop search at address 'to'

-v print version and exit

-x [hex] search for hexpair string (909090) (can be used multiple times)

-X show hexdump of search results

-z search for zero-terminated strings

-Z show string found on each search hit

### rahash2 基于块的哈希

### rasm2 反汇编

Usage: rasm2 [-CdDehLBvw] [-a arch] [-b bits] [-o addr] [-s syntax]

[-f file] [-F fil:ter] [-i skip] [-l len] 'code'|hex|-

-a [arch] Set architecture to assemble/disassemble (see -L)

-b [bits] Set cpu register size (8, 16, 32, 64) (RASM2\_BITS)

-c [cpu] Select specific CPU (depends on arch)

-C Output in C format

-d, -D Disassemble from hexpair bytes (-D show hexpairs)

-e Use big endian instead of little endian

-f [file] Read data from file

-F [in:out] Specify input and/or output filters (att2intel, x86.pseudo, ...)

-h Show this help

-i [len] ignore/skip N bytes of the input buffer

-k [kernel] Select operating system (linux, windows, darwin, ..)

-l [len] Input/Output length

-L List supported asm plugins

-o [offset] Set start address for code (default 0)

-O [file] Output file name (rasm2 -Bf a.asm -O a)

-s [syntax] Select syntax (intel, att)

-B Binary input/output (-l is mandatory for binary input)

-v Show version information

-w What's this instruction for? describe opcode

If '-l' value is greater than output length, output is padded with nops

If the last argument is '-' reads from stdin

Excample:

rasm2 –a x86 –b 32 ‘mov eax,33’ 得到汇编机器码

b821000000

rasm2 –a arm –b 16 1045 反汇编(thumb)

cmp r0, r2

rasm2 –L 枚举反编译插件

# Debug for mac&ios

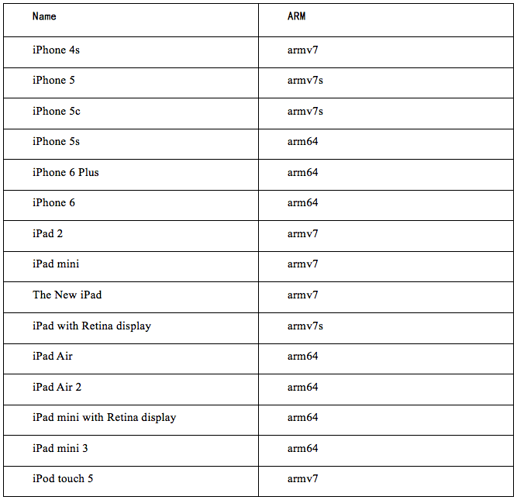
## lldb调试

### 搭建环境

server端：mac上的副本位于

/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/DeviceSupport/?.?/DeveloperDiskImage.dmg的/usr/bin/debugserver (ios的副本位于/Developer/usr/bin/debugserver)

Client端：lldb



下面是usb接口调试操作步骤：

1. 安装lldb和usb调试环境

brew install lldb libplist libusb usbmuxd ldid

wget http://cgit.sukimashita.com/usbmuxd.git/snapshot/usbmuxd-1.0.8.tar.bz2

tar xjfv usbmuxd-1.0.8.tar.bz2

cd usbmuxd-1.0.8/python-client/

python tcprelay.py -t 22:22 留作iphone命令行操作

python tcprelay.py –t 23946:23946 留作iphone调试

1. 签名debugserver使之可以附加

创建签名文件entitlements.plist (在mac文本编辑器中操作，\r会导致异常，plist相当于android manifest)，在上面结果中加入更多的调试权限得到：

<?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>get-task-allow</key>

<true/>

<key>task\_for\_pid-allow</key>

<true/>

<key>run-unsigned-code</key>

<true/>

</dict>

</plist>

查看已有签名：ldid –e debugserver

签名完成后送入终端：

codesign -s - --entitlements entitlements.plist -f debugserver

scp debugserver root@127.0.0.1:/bin/

上述过程在Xcode经历一次调试后自动完成，debugserver位于iOS /Developer/usr/bin

1. 拷贝ARMDisassembler，提升代码可读性

/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/DeviceSupport/?.?/DeveloperDiskImage.dmg的/Library/PrivateFrameworks/ARMDisassembler.framework

scp –r –p ARMDisassembler.framework root@127.0.0.1:/System/Library/PrivateFrameworks

1. 启动lldb-server

./bin/debugserver

debugserver host:port [program-name program-arg1 program-arg2 ...] 启动调试

debugserver host:port --attach=<pid> 进程id附加调试

debugserver host:port --attach=<process\_name> 进程名附加调试

1. 启动lldb-client

lldb -> process connect connect://127.0.0.1:23946

### LLDB调试

Debugserver命令行：

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

options:

-a process

Attach debugserver to process. The process can be a pid or executable name.

-d integer

Assign the waitfor-duration.

-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

-x method

–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.

--lockdown

Obtain parameters from lockdown (?)

Lldb调试命令：

apropos -- Find a list of debugger commands related to a particular word/subject.

breakpoint -- A set of commands for operating on breakpoints. Also see \_regexp-break.

command -- A set of commands for managing or customizing the debugger commands.

disassemble -- Disassemble bytes in the current function, or elsewhere in the executable program as specified by the user.

expression -- Evaluate an expression (ObjC++ or Swift) in the current program context, using user defined variables and variables currently in scope.

frame -- A set of commands for operating on the current thread's frames.

gdb-remote -- Connect to a remote GDB server. If no hostname is provided, localhost is assumed.

gui -- Switch into the curses based GUI mode.

help -- Show a list of all debugger commands, or give details about specific commands.

kdp-remote -- Connect to a remote KDP server. udp port 41139 is the default port number.

log -- A set of commands for operating on logs.

memory -- A set of commands for operating on memory.

platform -- A set of commands to manage and create platforms.

plugin -- A set of commands for managing or customizing plugin commands.

process -- A set of commands for operating on a process.

quit -- Quit out of the LLDB debugger.

register -- A set of commands to access thread registers.

script -- Pass an expression to the script interpreter for evaluation and return the results. Drop into the interactive interpreter if no expression is given.

settings -- A set of commands for manipulating internal settable debugger variables.

source -- A set of commands for accessing source file information

target -- A set of commands for operating on debugger targets.

thread -- A set of commands for operating on one or more threads within a running process.

type -- A set of commands for operating on the type system

version -- Show version of LLDB debugger.

watchpoint -- A set of commands for operating on watchpoints.

## IDA调试

由于lldb扩展了RSP协议(gdb remote serial debug protocol)， Ida调试使用原始gdb调试协议，功能没有lldb和gikdbg强，对于app只能下断点跟踪，功能十分有限，目前还在研究协议转换中

由于默认编译的程序会有PIE标志，导致模块地址随机化，ida无法直接附加，因此使用010Editor删除PIE标志，上传到远程机器后chmod 777改为可执行即可

步骤如下：

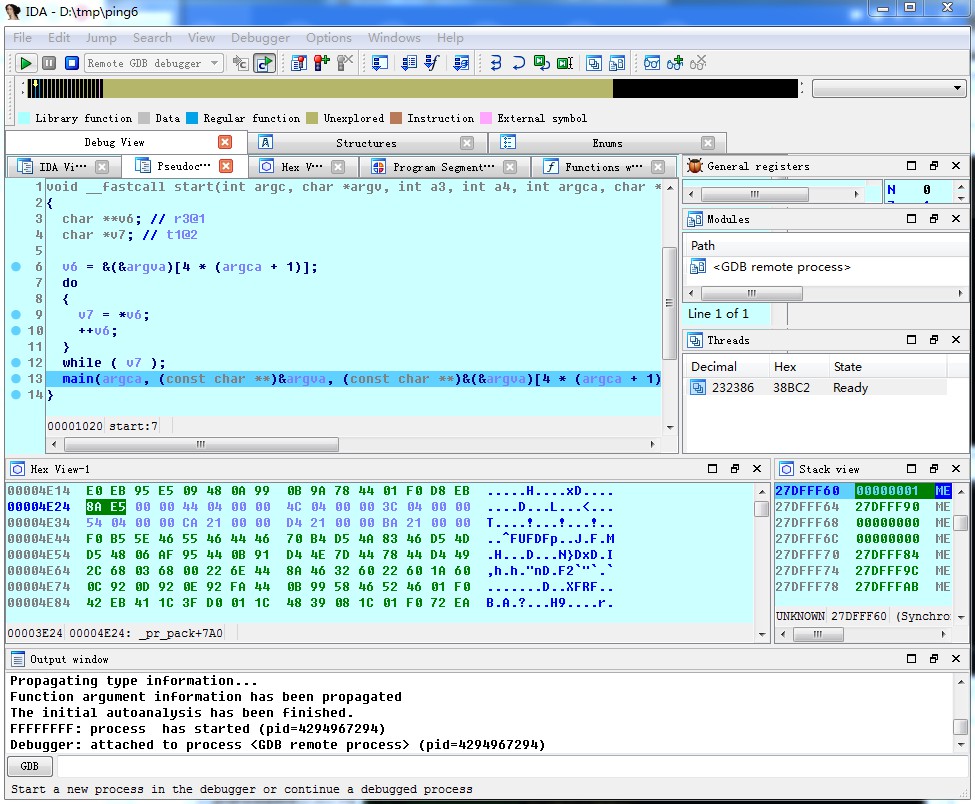
使用010editor等工具将可执行文件去除pie标志(mach\_header的flags MH\_PIE=0x200000，注意选择正确的架构)

拷贝可执行文件并载入到ida：scp root@127.0.01:/path/to/file /path/to/file

启动server端：debugserver –x backboard \*:1234 /path/to/file （或附加调试）

转发端口：python tcprelay.py –t 1234:1234

根据程序架构选择ida(x86 x64)设置ida为gdb调试，设置入口断点，设置调试地址和端口为127.0.0.1:1234，即可：



## Gikdbg调试

官网http://www.gikir.com/product.php，由ollydbg进一步开发的面向android和ios的汇编语言调试工具，支持静态分析elf/mach-o文件和动态调试android/iOS App，目前只支持arm系统，该软件运行在window上，适合调试dylib和可执行文件和简单的app

安装过程：

1. 配置服务器。

从官网下载gikdbg

scp $(GIKDBG)/iserver/gikir\_iserver.deb root@127.0.0.1:/var/tmp

ssh root@127.0.0.1

dpkg -i /var/tmp/gikir\_iserver.deb

重启后打开gikir\_server app（清除占用6080端口的进程）

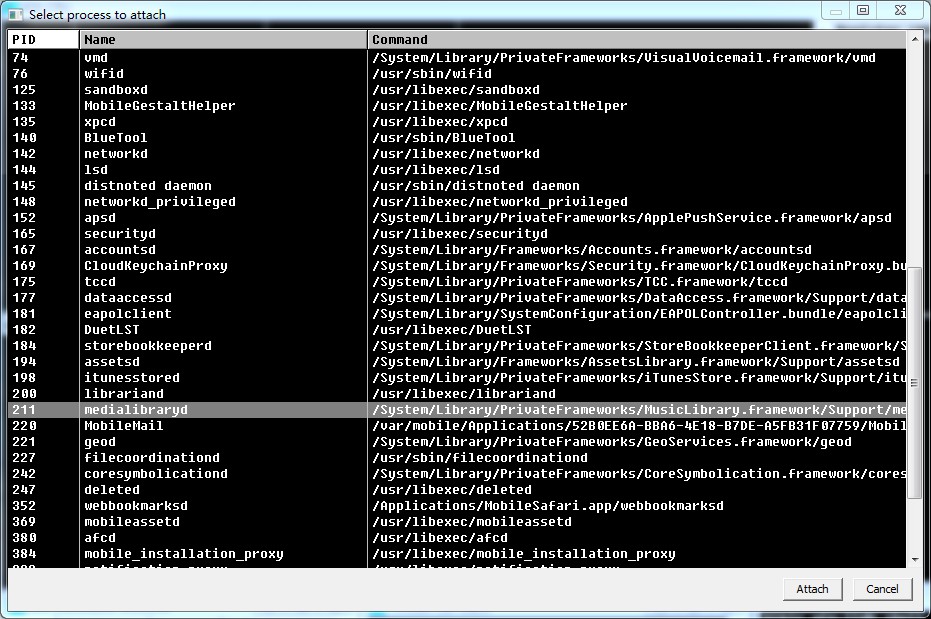
另一种安装方式是添加cydia源http://apt.feng.com/geekneo，安装gikir\_iserver

1. 启动客户端。

执行Gikdbg.exe

iDebug/Login(USB)登录

iDebug/File/Attach附加调试 Open启动调试



目前gikdbg可以调试控制台、动态库、app程序，支持usb/wifi，支持注入动态库，首次调试的程序需要打补丁：

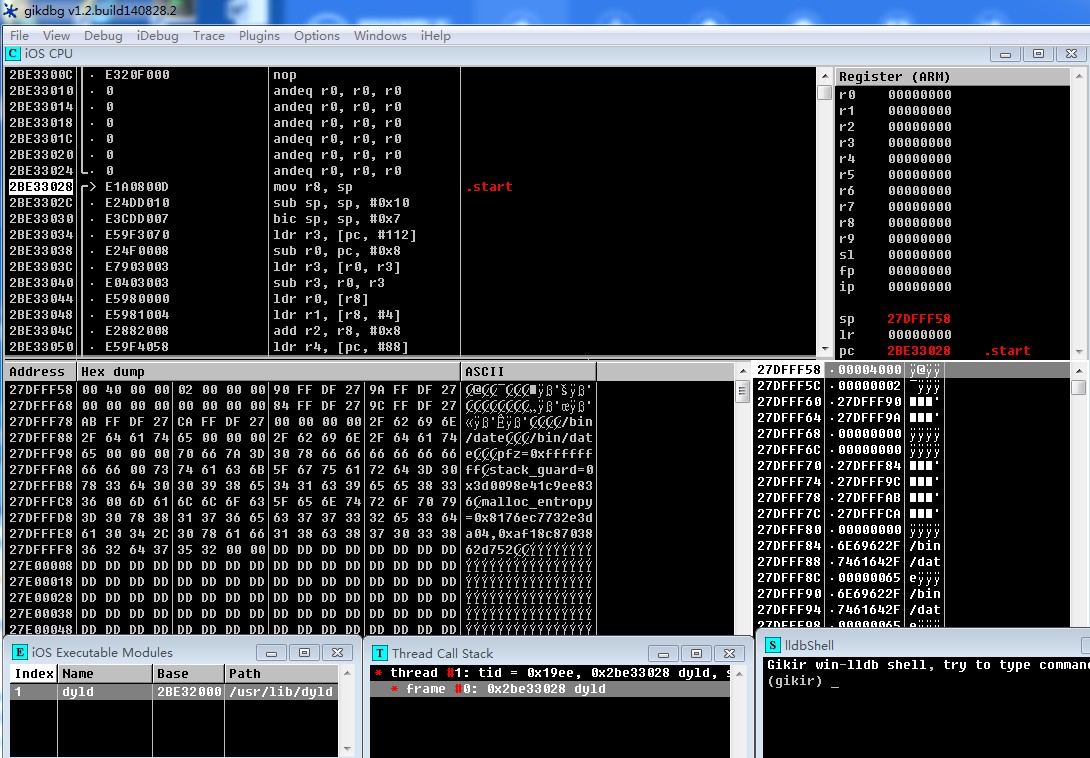
1） 删除MH\_PIE标志，让进程每次加载基址固定；

2） 记录App的UUID值；

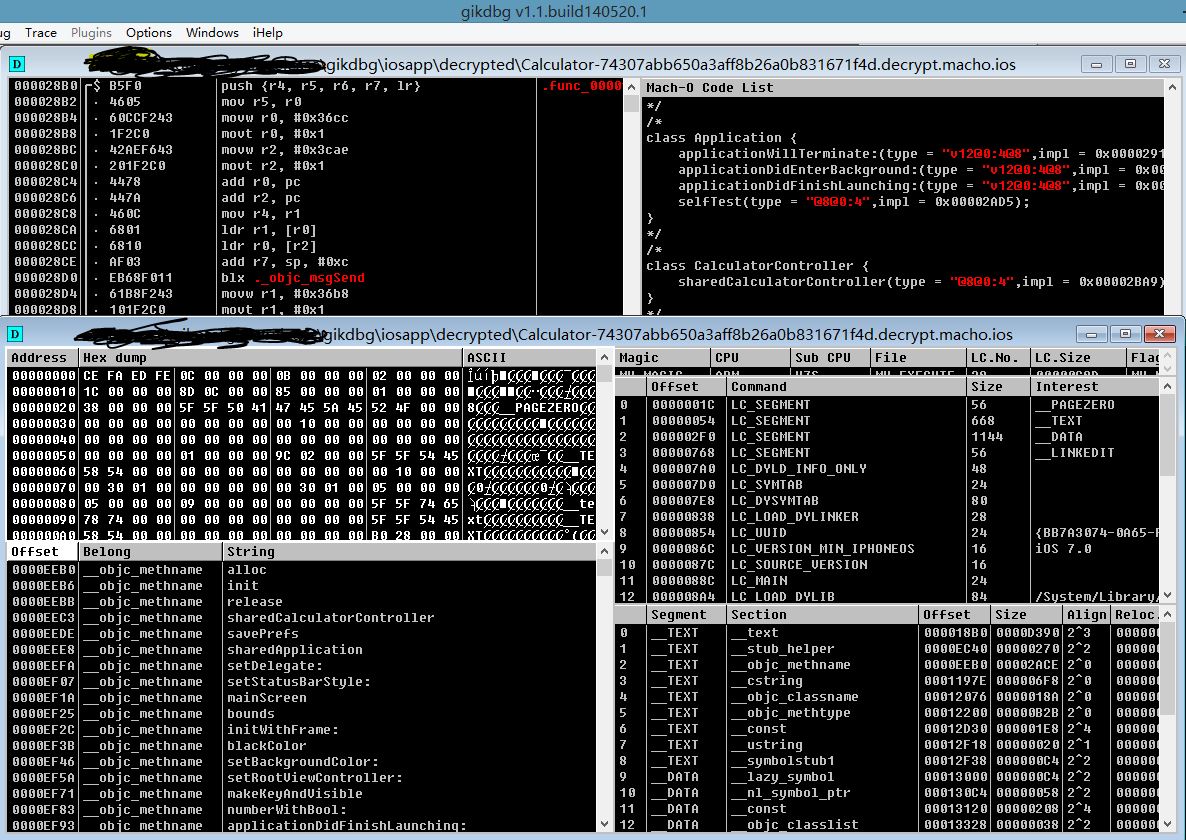
3） 如果是FAT格式的App则禁用最低以及最高的架构版本；

4） 如果是加密的App则解密该App；

5） 注入调试辅助动态库gikir\_iserver\_injecter.dylib；

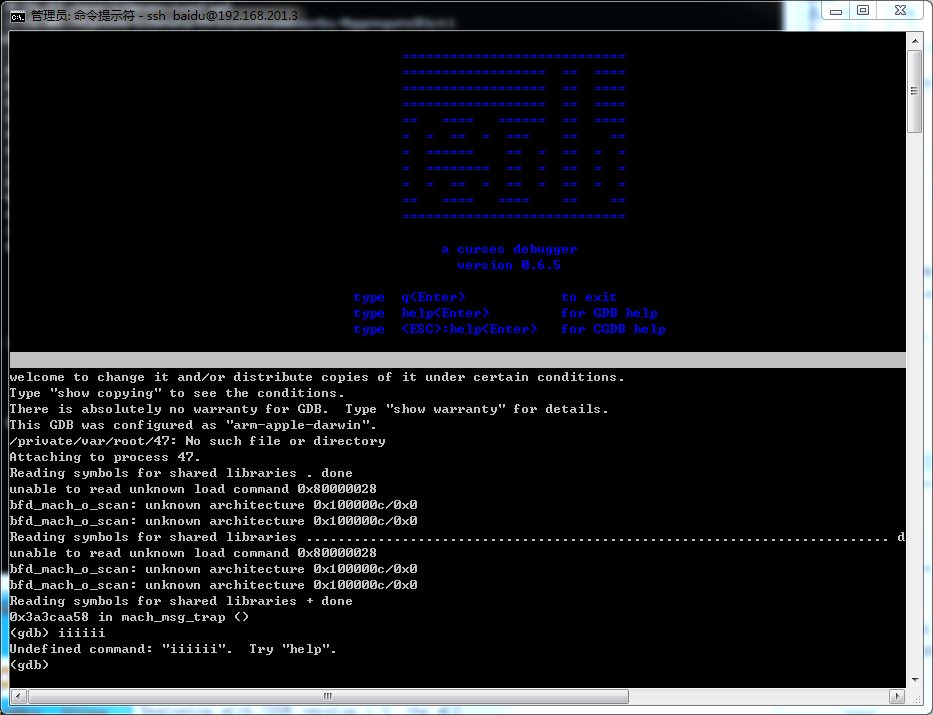


查看mach-o格式：



## CGDB调试

添加radare源：http://cydia.radare.org 安装The Curses Debugger即可



上面为代码区，下面为gdb调试区，”ESC”切换到源码区，”I”切换到调试区，源码区快捷键：

F5=run F6=continue F7=finish F8=next F9=step

## Radare调试

https://radare.gitbooks.io/radare2book/content/introduction/rax2.html

> d? ; get help on debugger commands

> ds 3 ; step 3 times

> db 0x8048920 ; setup a breakpoint

> db -0x8048920 ; remove a breakpoint

> dc ; continue process execution

> dcs ; continue until syscall

> dd ; manipulate file descriptors

> dm ; show process maps

> dmp A S rwx ; change page at A with size S protection permissions

> dr eax=33 ; set register value. eax = 33

# Trace/Hook for mac/ios

## 系统支持

mac&ios进程加载器dyld提供了设置环境变量DYLD\_INSERT\_LIBRARIES 的方式向目标进程注入动态库，另外mac&ios系统支持的hook为在mach-o的\_\_DATA \_\_interpose节数据，源码如下，编译成mac和ios的binary即可：

#include <unistd.h>

#include <fcntl.h>

typedef struct interpose\_s{

void\* new\_func;

void\* orig\_func;

} interpose\_t;

int my\_open(const char\*, int ,mode\_t);

\_\_attribute\_\_((used))

const interpose\_t interposers[] \_\_attribute\_\_ ((section("\_\_DATA, \_\_interpose"))) =

{

{(void\*)my\_open, (void\*)open},

};

int my\_open(const char\* path, int flags, mode\_t mode)

{

int ret = open(path, flags, mode);

printf("%d = open %s\n",ret, path);

return ret;

}

void init() \_\_attribute\_\_((constructor));

void init()

{

printf("im in\n");

}

//gcc -dynamiclib l.c -o 1.dylib -Wall // compile to dylib

// lichao26de-iPhone:/tmp root# DYLD\_INSERT\_LIBRARIES=interpose.dylib cat 1

//im in

//3 = open 1

该法适用于普通程序，不适用于app，因为app无法用命令行直接启动

## OC支持

Method Swizzling交换函数地址

#import <objc/runtime.h>

@implementation UIView(Loghiding)

- (BOOL)swizzled\_setHidden {

NSLog(@"We're calling setHidden now!");

BOOL result = [self swizzled\_setHidden];

return result;

}

+ (void)load {

Method original\_setHidden;

Method swizzled\_setHidden;

original\_setHidden = class\_getInstanceMethod(self, @selector(setHidden));

swizzled\_setHidden = class\_getInstanceMethod(self, @selector(swizzled\_setHidden));

method\_exchangeImplementations(original\_setHidden, swizzled\_setHidden);

}

@end

load重写使该类在第一次加载时交换swizzled\_setHidden和setHidden函数指针，导致调用swizzled\_setHidden实际调用的是setHidden，反之亦然

## Frida

著名的跨平台注入&跟踪工具，普通安装方式pip install frida，越狱ios上安装方式：

添加源http://build.frida.re，安装frida，确保27042 27043端口不被占用

启动frida-server ./usr/sbin/frida-server

转发端口 python tcprelay.py –t 27042:27042 27043:27043

(查看端口占用：netstat –nltp lsof –n –i4TCP:[port])

frida-ps –R

userdeMac:~ user$ frida-ps -Rai

PID Name Identifier

---- ------------ ---------------------------

3830 邮件 com.apple.mobilemail

- App Store com.apple.AppStore

- Cydia com.saurik.Cydia

- FaceTime com.apple.facetime

- Game Center com.apple.gamecenter

- HelloPython com.saurik.HelloPython

- Passbook com.apple.Passbook

- Safari com.apple.mobilesafari

- iTunes Store com.apple.MobileStore

- 信息 com.apple.MobileSMS

- 地图 com.apple.Maps

- 备忘录 com.apple.mobilenotes

- 天气 com.apple.weather

- 指南针 com.apple.compass

- 提醒事项 com.apple.reminders

- 日历 com.apple.mobilecal

- 时钟 com.apple.mobiletimer

- 照片 com.apple.mobileslideshow

- 爱思助手 com.diary.mood

- 电话 com.apple.mobilephone

- 相机 com.apple.camera

- 股市 com.apple.stocks

- 视频 com.apple.videos

- 计算器 com.apple.calculator

- 设置 com.apple.Preferences

- 语音备忘录 com.apple.VoiceMemos

- 通讯录 com.apple.MobileAddressBook

- 音乐 com.apple.Music

跟踪加密调用：

$ frida-trace -U -i "CCCryptorCreate\*" Twitter

Uploading data...

CCCryptorCreate: Auto-generated handler …/CCCryptorCreate.js

CCCryptorCreateFromData: Auto-generated handler …/CCCryptorCreateFromData.js

CCCryptorCreateWithMode: Auto-generated handler …/CCCryptorCreateWithMode.js

CCCryptorCreateFromDataWithMode: Auto-generated handler …/CCCryptorCreateFromDataWithMode.js

Started tracing 4 functions. Press Ctrl+C to stop.

跟踪libc调用：

$ frida-trace -U -f com.apple.MobileSMS -i "open\*"

Instrumenting functions...

openlog: Auto-generated handler at …/openlog.js

opendev: Auto-generated handler at …/opendev.js

opendir: Auto-generated handler at …/opendir.js

openpty: Auto-generated handler at …/openpty.js

openx\_np: Auto-generated handler at …/openx\_np.js

open: Auto-generated handler at …/open.js

open$NOCANCEL: Auto-generated handler at …/open\_NOCANCEL.js

open\_dprotected\_np: Auto-generated handler at …/open\_dprotected\_np.js

openat: Auto-generated handler at …/openat.js

openat$NOCANCEL: Auto-generated handler at …/openat\_NOCANCEL.js

openbyid\_np: Auto-generated handler at …/openbyid\_np.js

Started tracing 11 functions. Press Ctrl+C to stop.

/\* TID 0xb07 \*/

193 ms open(path=0x1988c4669, oflag=0x0, ...)

194 ms open(path=0x16fdeebc6, oflag=0x0, ...)

195 ms opendir()

195 ms | open$NOCANCEL()

195 ms opendir()

196 ms | open$NOCANCEL()

<http://www.frida.re/docs/javascript-api/>

Memory.readCString(address[, size = -1])

Usage: frida-trace [options] target

Options:

--version show program's version number and exit

-h, --help show this help message and exit

-D ID, --device=ID connect to device with the given ID

-U, --usb connect to USB device

-R, --remote connect to remote frida-server

-H HOST, --host=HOST connect to remote frida-server on HOST

-f FILE, --file=FILE spawn FILE

-n NAME, --attach-name=NAME

attach to NAME

-p PID, --attach-pid=PID

attach to PID

--debug enable the Node.js compatible script debugger

--disable-jit disable JIT

-I MODULE, --include-module=MODULE

include MODULE

-X MODULE, --exclude-module=MODULE

exclude MODULE

-i FUNCTION, --include=FUNCTION

include FUNCTION

-x FUNCTION, --exclude=FUNCTION

exclude FUNCTION

-a MODULE!OFFSET, --add=MODULE!OFFSET

add MODULE!OFFSET

-T, --include-imports

include program's imports

-t MODULE, --include-module-imports=MODULE

include MODULE imports

-m OBJC\_METHOD, --include-objc-method=OBJC\_METHOD

include OBJC\_METHOD

Usage: frida [options] target

Options:

--version show program's version number and exit

-h, --help show this help message and exit

-D ID, --device=ID connect to device with the given ID

-U, --usb connect to USB device

-R, --remote connect to remote frida-server

-H HOST, --host=HOST connect to remote frida-server on HOST

-f FILE, --file=FILE spawn FILE

-n NAME, --attach-name=NAME

attach to NAME

-p PID, --attach-pid=PID

attach to PID

--debug enable the Node.js compatible script debugger

--disable-jit disable JIT

-l SCRIPT, --load=SCRIPT

load SCRIPT

--no-pause automatically start main thread after startup

Usage: frida-ps [options]

Options:

--version show program's version number and exit

-h, --help show this help message and exit

-D ID, --device=ID connect to device with the given ID

-U, --usb connect to USB device

-R, --remote connect to remote frida-server

-H HOST, --host=HOST connect to remote frida-server on HOST

-a, --applications list only applications

-i, --installed include all installed applications

## Snoop-it/IntroSpy

IntroSpy

IntroSpy-Anayzer

python -m introspy -p ios -o output -f 192.168.1.12

http://repo.nesolabs.de//Snoop-it.deb

## Cycript

著名的注入&跟踪工具，支持iOS/Mac/Android，支持ObjC/JavaScript1.7/C++11语法

原理：远程线程注入

启动：cycript

（本文最后提供了常用的cycript函数库utils）

usage: cycript [-c] [-p <pid|name>] [-r <host:port>] [<script> [<arg>...]]

[-g <rename|bison|timing>] [-n <minify>]

-c 将表达式编译为js1.5语法

-r 远程连接cyript

-p 匹配进程并注入

远程连接cycript：

hcy=dlopen(”libcycript.dylib”,1) (可以使用各种方式将libcycript.dylib加载到进程中)

CYListenServer=(typedef void(short))(dlsym(hcy,”CYListenServer”))

CYListenServer(111) 调用函数

tcprelay –t 111:111 host上转发端口

cycript –r 127.0.0.1:111 host上连接server

编译JS

echo "[x\*x for each(x in [1,2,3])]" | cycript -c > x.js

cat x.js

(function($cyv,x){$cyv=[];(function($cys){$cys=[1,2,3];for(x in $cys){x=$cys[x];$cyv.push(x\*x)}})();return $cyv})()

?命令：

?bypass 忽略错误

?debug 调试输出开关

?lower

?exit

?reparse 显示换行等字符

?syntax 语法高亮

?gc 强制js垃圾回收

### 语法特点

JS/ObjC对象桥

某些js原生类型直接映射为oc类型：

|  |  |
| --- | --- |
| **JS type** | **ObjC type** |
| number | NSNumber (CFNumber) |
| boolean | NSNumber (CFBoolean) |
| string | NSString |
| Array | NSArray |
| object (Associative array) | NSDictionary |

cycript的null对应oc的nil，YES和NO已经在Cycript中定义

例如：

[[NSArray arrayWithObjects:

[NSNumber numberWithInt:41],

"foo",

[NSNumber numberWithBool:YES],

[NSArray arrayWithObjects:[NSNumber numberWithInt:8], [NSNumber numberWithInt:6], nil],

[NSDictionary dictionaryWithObjectsAndKeys:

[NSNumber numberWithInt:12], "a",

[NSNumber numberWithInt:46], "b",

nil],

[NSNumber numberWithInt:36],

nil] indexOfObject:"foo"]

可以直接简写为：[[41,"foo",true,[8,6],{a:12,b:46},36] indexOfObject:"foo"]

兼容OC语法

[obj msg:var]

@selector(selname)

obj->ivar

\*ptr 打印结构体或类成员

objc->[key] 获取实例的某成员

&var 获取Objc实例地址

@class classname : superclass {} 定义Objc类

+ methodname {function body}

- methodname {function body}

@end

new classname

@”str” 等价于”str”

[super …]

Selector(selname) 声明selector

Functor(function body, type encoding) 定义ObjC函数

new Functor(function(x,y){return (x+y).toString(16);}, "\*dd") (double, double) → char\*

block = ^ int (int value) { return value + 5; } 声明block

### 基本功能

导入js模块

import "/tmp/test.js"

导入cy模块

@import com.saurik.substrate.MS (对应/usr/lib/cycript0.9/com/saurik/substrate/MS.cy)

导入nodejs/cy模块

util=require(“util”)

utils=require(“/tmp/utils.cy”)

返回上一次执行结果

\_

获取可执行程序参数

system.args

指针类型转换

Pointer(address, type encoding) 函数地址转换为encoding指定类型

Instance(address) 对象地址转换为ObjC对象地址

pt=(typedef int\*)(oldpt) 强制类型转换

定义结构体

CGRect=(typedef struct {int x;int y;})

rect = new (struct CGRect)

rect.size 获取结构体大小

定义数组

arr=new (typedef char[10])

获取对象类型

obj.class

### OC运行时功能

获取所有类

ObjectiveC.classes

获取所有接口

ObjectiveC.protocols

获取某实例所有方法

[MYCLASS (tab-key)]

获取实例的所有变量

\*obj

由内存地址获取对象

[#0x18b6c8d0 show]

获取所有类实例

choose(SBIconView)

获取成员函数类型描述

selector.type(class)

selector(copyWithZone:).type(NSString) => @12@0:4^{\_NSZone=}8.

修改函数

var oldm = NSObject.prototype.description

NSObject.prototype.description = function() { return oldm.call(this) + ' (of doom)'; }

[new NSObject init]

#"<NSObject: 0x100d11520> (of doom)"

### 调试功能

获取加载模块

utils.get\_dyld\_info()

ObjectiveC.images

修改内存权限

utils.mprotect(addr, size, utils.constants.PROT\_READ)

读写内存

var foo = new int

\*foo = 0x12345678

utils.hexdump(foo, 4)

获取当前回溯栈(用于hook)

function bt(){

return [NSThread callStackSymbols];

}

执行命令

utils.getOutputFromTask(“/bin/ls”, [])

调用函数

[obj msg: var] 调用oc函数

fopen(“/tmp”,”r”) 调用c函数

utils.apply("printf", ["%s %.3s, %d -> %c, float: %f\n", "foo", "barrrr", 97, 97, 1.5]) 反射调用c函数

反汇编

var method = class\_getInstanceMethod(NSNumber,@selector(intValue));

var imp = method\_getImplementation(method);

utils.disasm(imp, 10)

0x7fff83363b8c 1 55 push rbp

0x7fff83363b8d 3 4889e5 mov rbp, rsp

0x7fff83363b90 2 4157 push r15

0x7fff83363b92 2 4156 push r14

0x7fff83363b94 2 4155 push r13

汇编

var n = [NSNumber numberWithLongLong:10]

var method = class\_getInstanceMethod([n class], @selector(longLongValue));

var imp = method\_getImplementation(method);

utils.asm(imp, 'mov eax, 42; ret;')

### Hook功能

记录OC函数调用(需要substrate)

utils.logify(NSNumber, @selector(numberWithDouble:))

[NSNumber numberWithDouble:1.5] //触发logifyt

2014-07-28 02:26:39.805 cycript[71213:507] +[<NSNumber: 0x10032d0c4> numberWithDouble:1.5]

注意：对静态成员函数，第一参为object\_getClass(类名)；对普通成员函数，第一参为object\_getClass(实例)

底层实现：

cy# @import com.saurik.substrate.MS

cy# var oldm = {};

cy# MS.hookMessage(NSObject, @selector(description), function() {

return oldm->call(this) + " (of doom)";

}, oldm)

cy# [new NSObject init]

#"<NSObject: 0x100203d10> (of doom)"

记录C函数调用(需要substrate)

utils.logifyFunc("fopen", 2)

apply("fopen", ["/etc/passwd", "r"]); //触发logifyt

2015-01-14 07:01:08.009 cycript[55326:2042054] fopen(0x10040d4cc, 0x10040d55c)

底层实现：

cy# @import com.saurik.substrate.MS

cy# extern "C" void \*fopen(char \*, char \*);

cy# var oldf = {}

cy# var log = []

cy# MS.hookFunction(fopen, function(path, mode) {

var file = (\*oldf)(path, mode);

log.push([path.toString(), mode.toString(), file]);

return file;

}, oldf)

cy# fopen("/etc/passwd", "r");

(typedef void\*)(0x7fff774ff2a0)

cy# log

[["/etc/passwd","r",(typedef void\*)(0x7fff774ff2a0)]]

### 其他功能

获取所有控件元素

utils.find\_subviews()

获取所有viewcontroller

utils.find\_subview\_controllers()

获取cpu类型

utils.getCpuType()

获取坐标

manager=choose(CLLocationManager)[0]

[manager location]

获取bundleid

NSBundle.mainBunble.bundleIdentifier

模糊测试(需要iphone端radamsa和utils.cy)

function radamsa(input){

system(“echo ” + input + “ /tmp/radamsa.tmp”)

utils.getOutputFromTask(“/bin/radamsa”, [“/tmp/radamsa.tmp”])

}

//s=radamsa(“111”)

//@”11111\n”

### 其他CYCRIPT模块

cycript-utils https://github.com/Tyilo/cycript-utils/blob/master/utils.cy

weak-dump 运行时的class-dump

classdump-dyld weak-dump升级版，支持x64

### 实例

1. 记录所有http/https通信

修改hook函数为打印回溯栈

utils.logify = function(cls, sel) {

@import com.saurik.substrate.MS;

var selFormat = sel.toString().replace(/:/g, ":%s ").trim();

var logFormat = @"%s[<%@: %p> " + selFormat + "]";

var oldm = {};

MS.hookMessage(cls, sel, function() {

var args = [logFormat, class\_isMetaClass(cls)? "+": "-", cls, (typedef void \*)(this)];

for (arg of arguments) {

args.push(arg.toString());

}

NSLog.apply(null, args);

var s="";

s += NSThread.callStackSymbols;

NSLog(@"backtrace=%s", s.toString());

var r = oldm->apply(this, arguments);

if(r !== undefined) {

NSLog(@" = %s", r.toString());

}

return r;

}, oldm);

return oldm;

};

执行utils.logify(object\_getClass(NSURLRequest),@selector(requestWithURL:)) 即可



1. 定位窗口修改内容

先获取焦点所在控件：

var arr=utils.find\_subviews()

for(var i=0;i<arr.length;i++) {if(arr[i].isFirstResponder) i+” ”+arr[i]}

#"14 <UIFieldEditor: 0x1bef800; frame = (0 0; 210 21); text = ''; clipsToBounds = YES; opaque = NO; gestureRecognizers = <NSArray: 0x8009190>; layer = <CALayer: 0x800a150>; contentOffset: {0, 0}>"

设置内容：

arr[14][.text=@”test](mailto:.text=@)”

语句执行完，内容立显

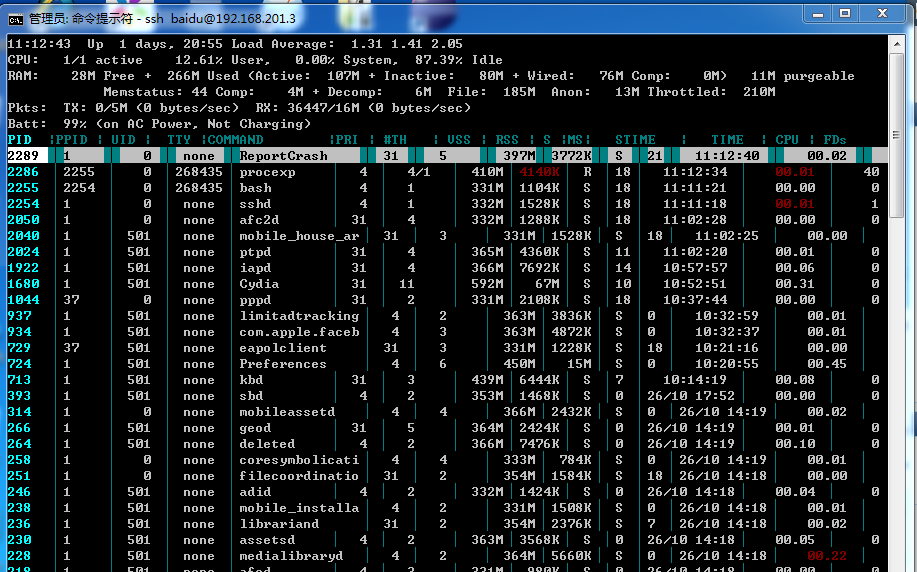
## JSPatch

弹警告窗：

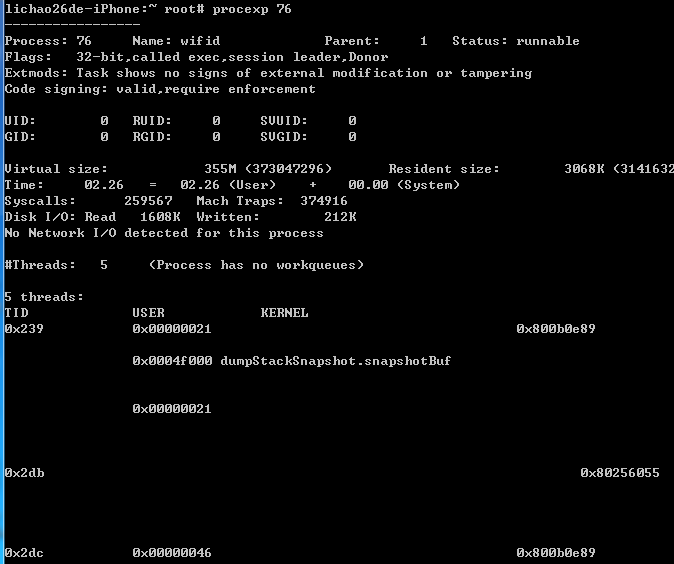
require('UIAlertView');return UIAlertView.alloc().initWithTitle\_message\_delegate\_cancelButtonTitle\_otherButtonTitles('a','b',null,'c',null).show()

## procexp-进程监视

添加radare源：<http://cydia.radare.org> 安装procexp即可

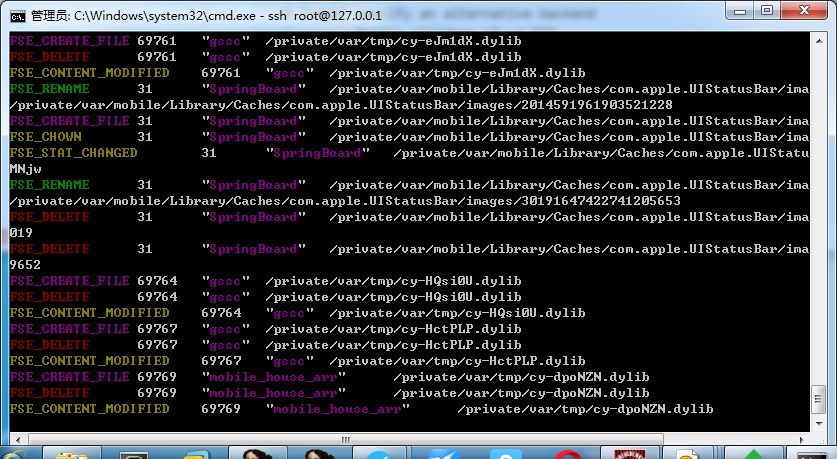


procexp [pid] 获取进程信息，包括线程栈，模块列表，打开文件



## fsmon-文件系统监视

跨平台文件系统监视工具，可以指定进程名或进程id https://github.com/nowsecure/fsmon



Usage: ./fsmon [-jc] [-a sec] [-b dir] [-B name] [-p pid] [-P proc] [path]

-a [sec] stop monitoring after N seconds (alarm)

-b [dir] backup files to DIR folder (EXPERIMENTAL)

-B [name] specify an alternative backend

-c follow children of -p PID

-f show only filename (no path)

-h show this help

-j output in JSON format

-L list all filemonitor backends

-p [pid] only show events from this pid

-P [proc] events only from process name

-v show version

[path] only get events from this path

backends:

inotify (linux / android)

fanotify (linux > 2.6.36 / android 5)

devfsev (osx /dev/fsevents - requires root)

kqueue (xnu - requires root)

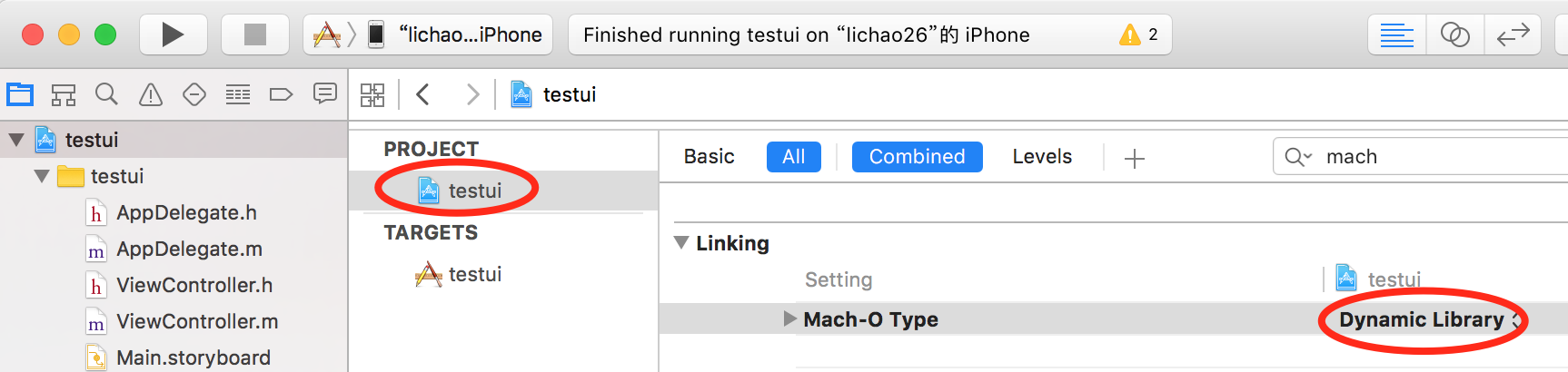
kdebug (bsd?, xnu - requires root)

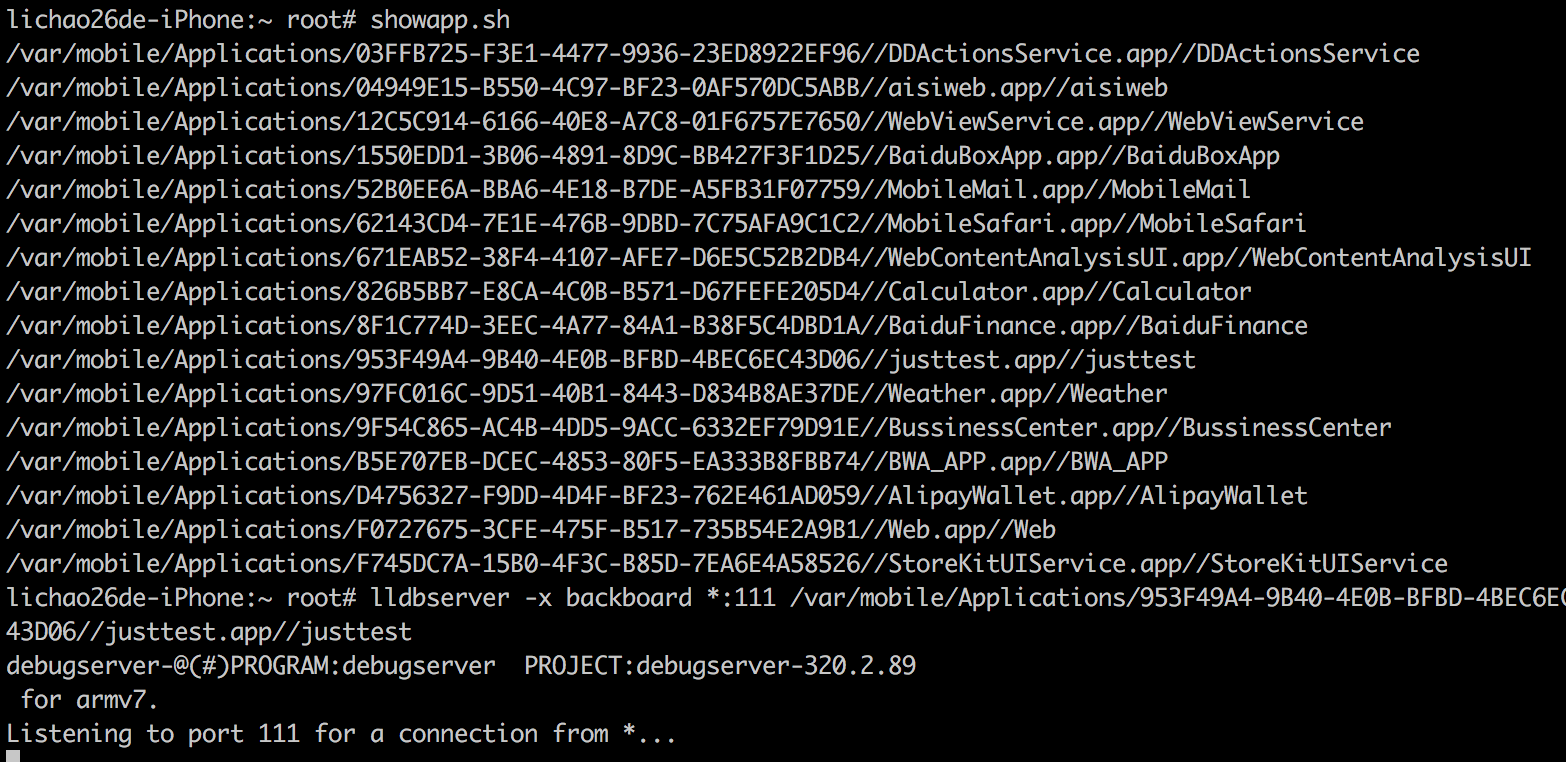
fsevapi (osx filesystem monitor api)

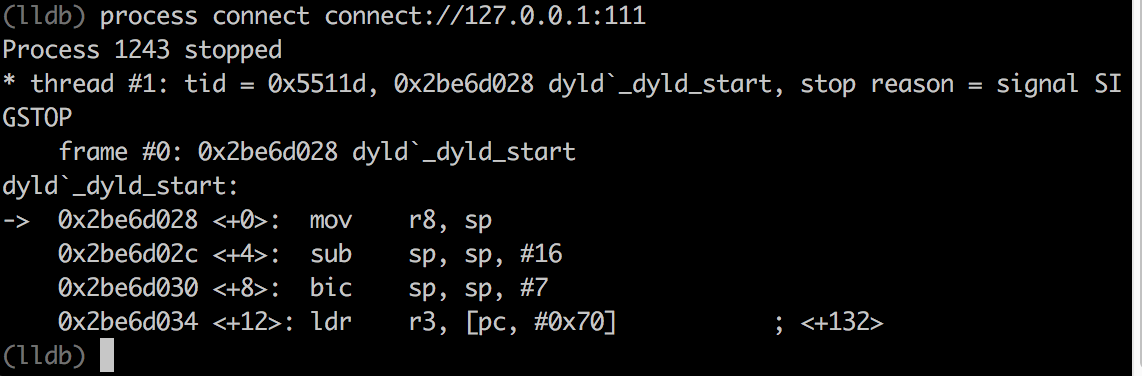
## iOS实例分析

### 准备工作

编译dylib：设置工程为Dynamic Library





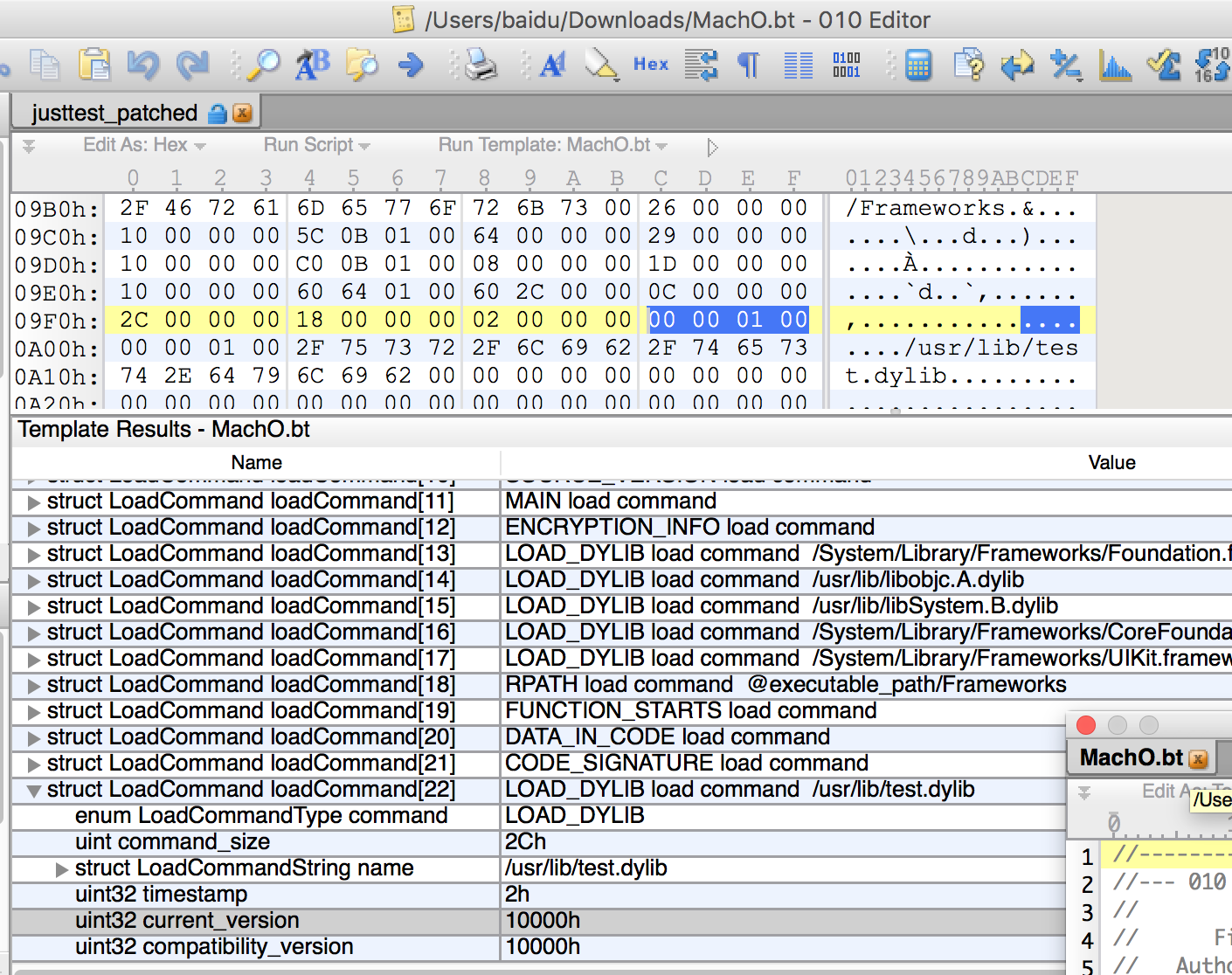
### Inject

**方式一：MachO格式注入**

sh-3.2# python extractmacho.py --injectdylib=/usr/lib/test.dylib justtest

update to justtest\_patched

放到app目录下即可，app会搜索@executable\_path



注意2个version设置为0即可，可执行文件需要有执行权限，dylib无需执行权限

(lldb) cont

Process 1894 resuming

2016-10-17 11:18:07.526 justtest[1894:60b] im in

**方式二：调试器(LLDB/GDB/Cycript等)注入**

LLDB/GDB：po dlopen("/usr/lib/test.dylib",1)

Cycript：dlopen("/usr/lib/test.dylib",1) 调试状态下也可使用cycript

**方式三：MobileLoader注入**

CydiaSubstrate的MobileLoader组件用于加载第三方dylib给指定程序，MobileLoader首先在启动时使用DYLD\_INSERT\_LIBRARIES加载自身，之后加载/Library/MobileSubstrate/DynamicLibraries下的所有动态库，由于是全局的默认会在所有程序中加载，可以采用过滤配置plist文件加载dylib(iOS9以后必须存在plist才准予加载)：

Bundle：必须匹配app(s)的bundle-id才准予加载

Classes：必须在目标进程中实现指定类(s)才准予加载

Executables：必须匹配可执行文件名才准予加载

例：

Filter = {

Executables = (“mediaserverd”);

Bundles = (“com.apple.sprintboard”, “net.whatsapp.WhatsApp”);

Mode = “Any”

};

plist文件名与dylib名相同

### Hook

**方式一：Cydia Hook框架**

Cydia提供的api：从终端/usr/include/substrate.h取最新版头文件

MSImageRef MSMapImage(const char\* file) 加载dylib

cont void\* MSImageAddress(MSImageRef image)

bool MSHookProcess(pid\_t pid, const char\* library) 远程线程方式(vm\_)注入dylib

MSImageRef MSGetImageByName(const char\* file) 获取模块基址，优于dlopen

Void\* MSFindSymbol(MSImageRef image, const char\* name) 获取函数地址，优于dlsym

char\* MSFindAddress(MSImageRef image, void\*\* address)

Void MSHookFunction(void\* symbol, void\* replace, void\*\* result) hook c函数

IMP MSHookMessage(Class \_class, SEL sel, IMP imp, const char\* prefix) hook oc消息

Void MSHookMessageEx(Class \_class, SEL sel, IMP imp, IMP\* result) hook oc消息

void MSHookClassPair(Class target, Class hook, Class old) 封装MSHookMessageEx

Hook c函数底层实现仍然是arm汇编的inline hook

Hook oc函数底层实现则是利用objective-c runtime function

对于 rettype funcname(type1 param1, type2 param2)的函数：

hook c function 方式1 -- MSHookFunction：

rettype (\*old\_funcname)(type1 param1, type2 param2);

rettype new\_funcname(type1 param1, type2 param2)

{

……….work before hook……….

old\_funcname(param1, param2);

……….work after hook…………

}

MSHookFunction((void\*) funcname, (void\*)&new\_funcname, (void\*\*)&old\_funcname);

hook c function 方式2 – MSHookFunction-MSHook-MSHack：

MSHook(rettype, funcname, type1 param1, type2 param2)

{

……….work before hook……….

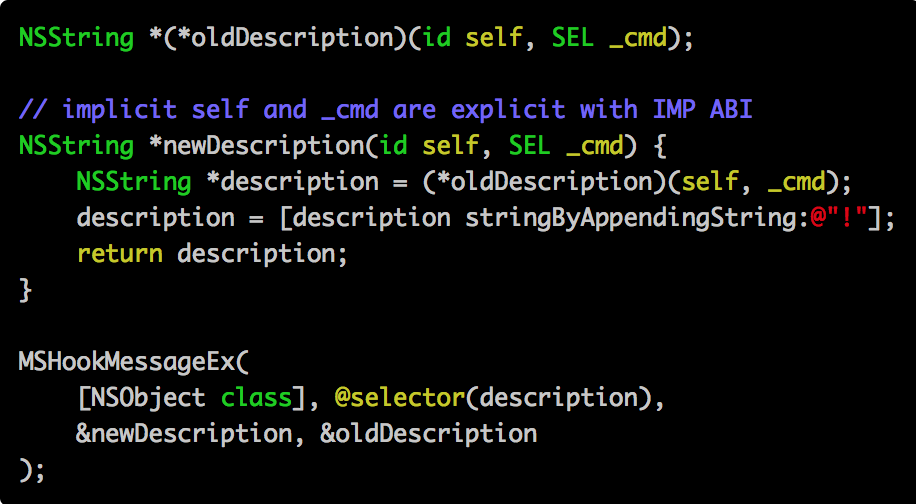
\_funcname(param1, param2);//注意前面加’\_’

……….work after hook…………

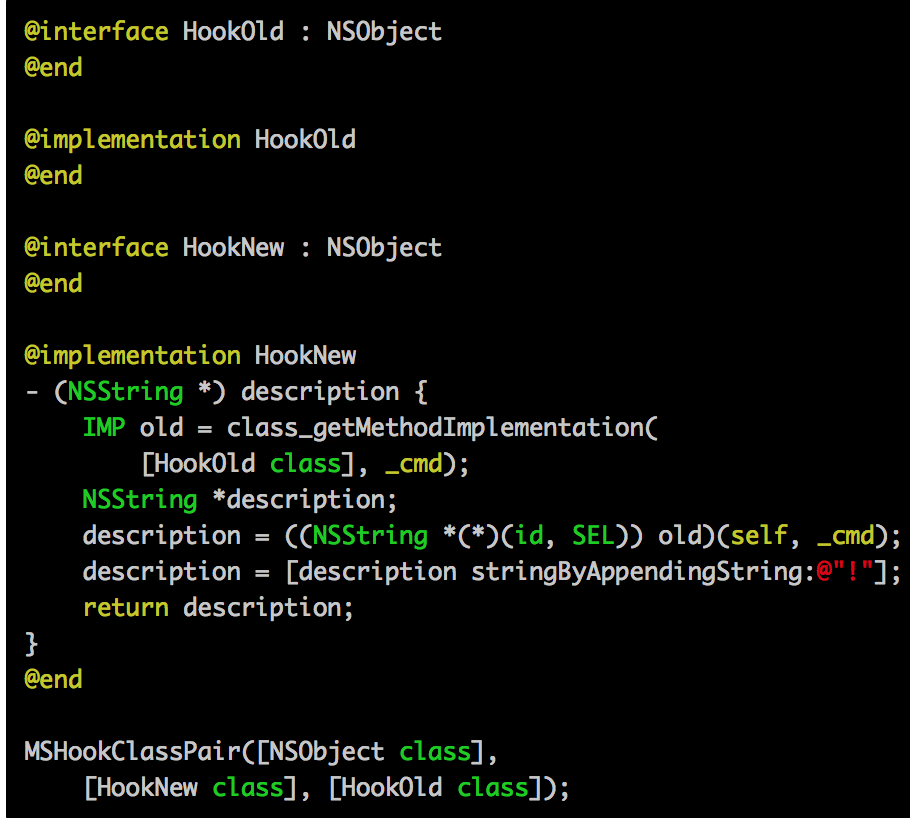
}

MSHookFunction(funcname, MSHake(funcname))

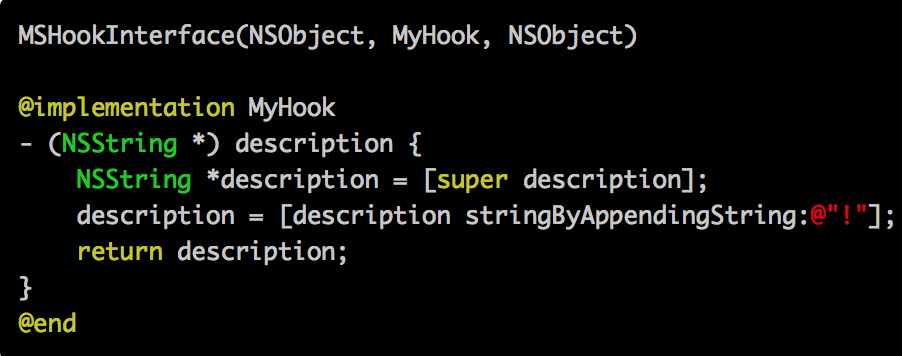
hook oc function 方式1 - MSHookMessageEx：



hook oc function 方式2 - MSHookClassPair：



hook oc function 方式3 - MSHookInterface：



1. Theos越狱框架开发

优点：方便，一键部署，缺点：调试麻烦

$THEOS/bin/nic.pl

iphone/tweak

export THEOS\_DEVICE\_IP=???

make package install

2. XCode开发

特点：和前者相反，调试方便，只需要如前述修改mach-o type为可执行程序即可调试

#include <CydiaSubstrate.h>

void\* handle = dlopen(“libsubstrate.dylib”, 1);

typedef void (\*HOOK)(void\*, void\*, void\*\*);

HOOK MSHookFunction = (HOOK)dlsym(handle, “MSHookFunction”);

MSHookFunction((void\*)funcname, (void\*)&oldfunc, (void\*\*)&newfunc);

**方式二：frida/frida-trace**

frida安装：mac/linux/win下执行pip install frida，iOS上从frida源安装服务端，安好后服务端每次开机启动，占用端口27042/27043，因此在客户端执行python tcprelay.pt –t 27042:27042 27043:27043

frida-ps –R 枚举所有进程

frida-ps –R –a 枚举所有app进程

frida-ps –R –a –i 枚举所有安装的app及其bundle name

frida-trace –R –p PID 附加到进程(按进程id)

frida-trace –R –n name 附加到进程(按进程名，例如百度商户)

frida-trace –R –f FILE 拉起进程并跟踪(例如com.baidu.bshoppush)

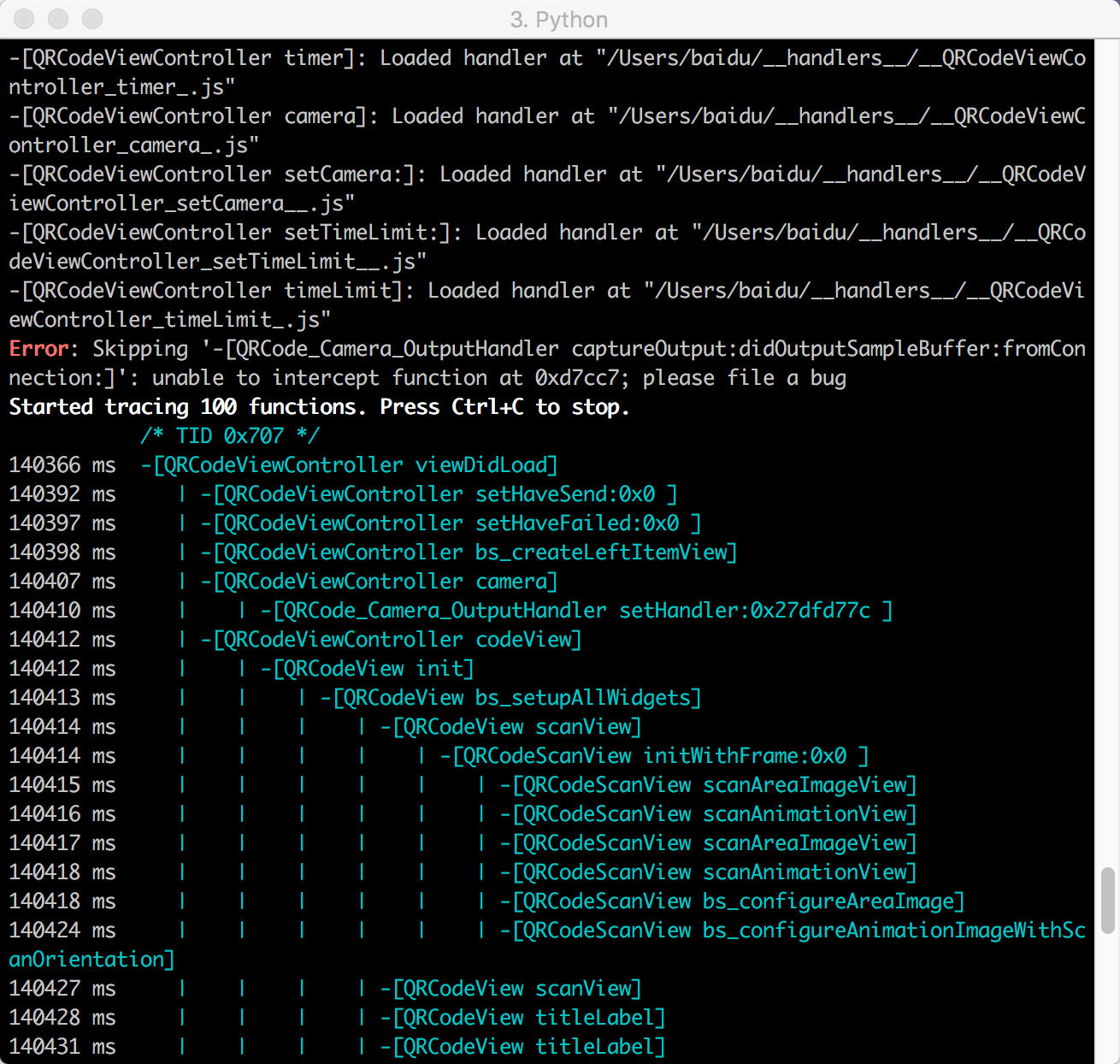
hook c function frida-trace –i “recv\*” –i “send\*” ….

hook oc function frida-trace –m “-[NS\* draw\*]” …

实例：跟踪商户app二维码操作

frida-trace -R -f com.baidu.bshoppush -m "-[QRCode\* \*]" -f com.baidu.bshoppush

对生成的js进行编辑，自定义输出数据可以在控制台得到相应显示



获取JSPatch下发代码：

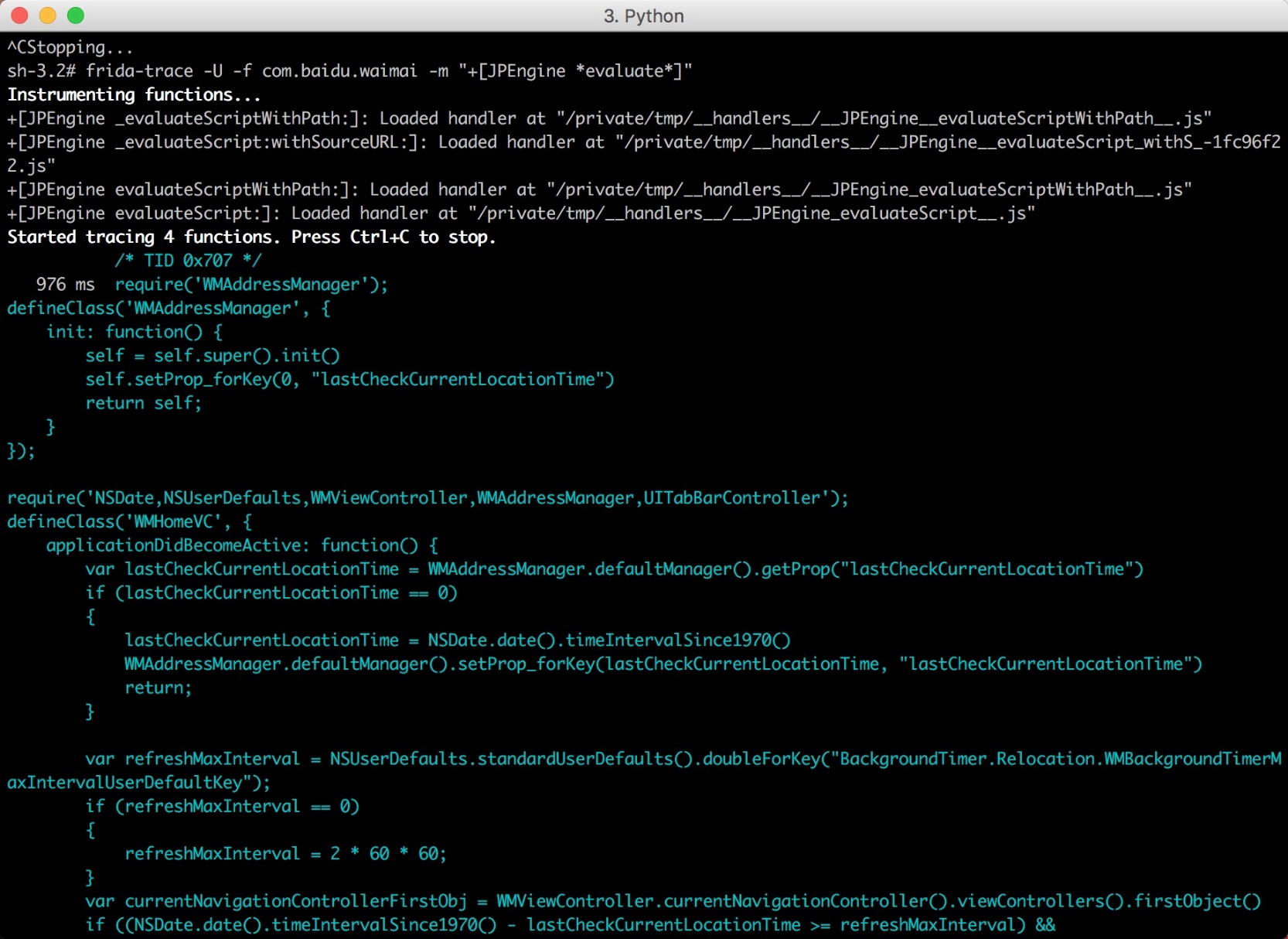
frida-trace –U –f com.baidu.waimai –m “+[JPEngine \*evaluate\*]”

js脚本内容

var data=new ObjC.Object(args[2]);

log(data.toString());

log(Thread.backtrace(this.context, Backtracer.ACCURATE).map(DebugSymbol.fromAddress).join("\n") + "\n");



**方式三：logify**

该工具从给定oc接口文件生成可以hook整个类的logos文件，同时输出到syslog，是方式一的自动化

# Jailbreak Development Tools

## Theos

著名的cydia插件开发环境，命令行方式

常用ios公有/私有api <https://github.com/nst/iOS-Runtime-Headers/tree/master/PrivateFrameworks>

$ $THEOS/bin/nic.pl

NIC 1.0 - New Instance Creator

------------------------------

[1.] iphone/application

[2.] iphone/library

[3.] iphone/preference\_bundle

[4.] iphone/tool

[5.] iphone/tweak

Choose a Template (required): 1

Project Name (required): iPhoneDevWiki

Package Name [com.yourcompany.iphonedevwiki]: net.howett.iphonedevwiki

Authour/Maintainer Name [Dustin L. Howett]:

Instantiating iphone/application in iphonedevwiki/...

Done.

$

### IOSOpenDev

用于XCode的越狱程序开发插件

<http://iphonedevwiki.net/index.php/IOSOpenDev>

# Mac&iOS file format analysis

工具：

otool 类似于objdump，可以解析objc类信息

class-dump objc类接口信息解析成可读objc头文件

OBJC\_HELP 环境变量打日志

OBJC\_HELP=1 ./build/Debug/HelloWorld

objc: OBJC\_HELP: describe Objective-C runtime environment variables

objc: OBJC\_PRINT\_OPTIONS: list which options are set

objc: OBJC\_PRINT\_IMAGES: log image and library names as the runtime loads

them

NSObjCMessageLoggingEnabled 环境变量用于打印objc\_msgSend调用日志

NSObjCMessageLoggingEnabled=Yes ./hello

Hello World!

-[dcbz@megatron:~/code/HelloWorld/build]$ cat /tmp/msgSends-6686

+ NSRecursiveLock NSObject initialize

+ NSRecursiveLock NSObject new

+ NSRecursiveLock NSObject alloc

....

+ Talker NSObject initialize

+ Talker NSObject alloc

+ Talker NSObject allocWithZone:

- Talker NSObject init

- Talker Talker say:

- Talker NSObject release

- Talker NSObject dealloc

machoview 查看格式的gui工具 https://github.com/gdbinit/MachOView.git

dtrace 跟踪mac上objective-c函数调用

<http://developer.apple.com/documentation/DeveloperTools/Conceptual/MachORuntime/Reference/reference.html>

分析iOS二进制文件的过程：

1. 如果是app store下载的app，需要先用工具砸壳，将代码数据区内存解密
2. 从手机将文件拷贝到主机使用ida分析
3. 将砸壳生成的文件修改PIE标志并重新签名，替换原始app，方便动态分析

## 砸壳

由于class-dump等工具的流行，App Store上发布的软件都经过加密处理(LC\_ENCRYPTION\_INFO所标志的区域)，加载器dyld对可执行文件校验，根据fat头选择合适的架构，处理所有的command，使用posix\_spawn函数启动进程。ios上所有第三方代码都需要使用developer id代码签名，而代码签名作为数据存储在mach-o格式command结构中，因此fat格式中得每个架构的文件都分别签名，并由内核验证，如果验证失败则会收到停止信号而退出。在越狱机上可以通过ldid进行伪签名通过签名校验。进行了加密后，无法直接用ida查看内部结构，脱壳软件如下：

dumpencrypted <https://github.com/stefanesser/dumpdecrypted/blob/master/dumpdecrypted.c>

该工具注入目标进程内存，利用解密后的内存转储数据得到脱壳文件，时机在dyld加载后，init(\_\_mod\_init\_func)节加载前

clutch <https://github.com/KJCracks/Clutch/releases>

命令行工具。该工具使用posix\_spawn函数以暂停态(POSIX\_SPAWN\_START\_SUSPENDED)和ASLR关闭模式创建目标程序子进程，从而使目标进程不执行任何代码而得到系统解密的内存，后使用task\_for\_pid从mach port得到目标进程内存，最后更新头部的LC\_ENCRYPTION\_COMMAND，合并成文件。

Usage: Clutch [OPTIONS]

-b --binary-dump <value> Only dump binary files from specified bundleID

-d --dump <value> Dump specified bundleID into .ipa file

-i --print-installed 枚举安装应用

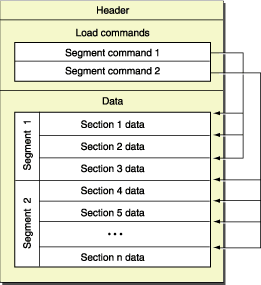
--clean Clean /var/tmp/clutch directory

--version Display version and exit

-? --help Display this help and exit

-n --no-color Print with colors disabled

## mach-o格式分析



相关数据结构定义在/Developer/SDKs/iPhoneOS.sdk/usr/include/mach-o/loader.h

总体结构包括：header结构、command表、数据区

header结构：用于指明cpu类型(x86?arm?...)，文件类型(动态库?可执行文件?...)，command表位置；如果文件中包含多个cpu的可执行文件，则会存在FAT header头指明每个cpu的文件位置，因此一个mach-o文件的开头可能是mach\_header结构，此时文件只包含一种cpu架构的可执行文件，也可能是fat\_header，存储不同mach\_header的偏移

struct mach\_header {

uint32\_t magic; /\* mach magic number identifier \*/

cpu\_type\_t cputype; /\* cpu specifier \*/

cpu\_subtype\_t cpusubtype; /\* machine specifier \*/

uint32\_t filetype; //静态库.a 目标文件.o 动态库.dylib 可执行文件 ………….

uint32\_t ncmds; /\* number of load commands \*/

uint32\_t sizeofcmds; /\* the size of all the load commands \*/

uint32\_t flags; /\* flags \*/

};

command表相当于pe的节表，描述文件和内存进行映射的表，包括\_\_PAGEZERO(标记可执行文件的第一个节)、\_\_TEXT、\_\_DATA、\_\_OBJC(objective-c运行库表用于描述类信息)、\_\_IMPORT、\_\_LINKEDIT(符号、字符串、重定位表)，常用的command如下：

LC\_SEGMENT/LC\_SEGMENT\_64 描述文件中得节和内存映射关系

struct segment\_command { /\* for 32-bit architectures \*/

uint32\_t cmd; /\* LC\_SEGMENT \*/

uint32\_t cmdsize; /\* includes sizeof section structs \*/

char segname[16]; /\* segment name \*/

uint32\_t vmaddr; /\* memory address of this segment \*/

uint32\_t vmsize; /\* memory size of this segment \*/

uint32\_t fileoff; /\* file offset of this segment \*/

uint32\_t filesize; /\* amount to map from the file \*/

vm\_prot\_t maxprot; /\* maximum VM protection \*/

vm\_prot\_t initprot; /\* initial VM protection \*/

uint32\_t nsects; /\* number of sections in segment \*/

uint32\_t flags; /\* flags \*/

};

LC\_LOAD\_DYLIB 要加载的动态库

struct dylib {

union lc\_str name; /\* library's path name \*/

uint32\_t timestamp; /\* library's build time stamp \*/

uint32\_t current\_version; /\* library's current version number \*/

uint32\_t compatibility\_version; /\* library's compatibility vers number\*/

};

struct dylib\_command {

uint32\_t cmd; /\* LC\_ID\_DYLIB, LC\_LOAD\_{,WEAK\_}DYLIB,

LC\_REEXPORT\_DYLIB \*/

uint32\_t cmdsize; /\* includes pathname string \*/

struct dylib dylib; /\* the library identification \*/

};

LC\_MAIN 描述入口点

struct entry\_point\_command {

uint32\_t cmd; /\* LC\_MAIN only used in MH\_EXECUTE filetypes \*/

uint32\_t cmdsize; /\* 24 \*/

uint64\_t entryoff; /\* file (\_\_TEXT) offset of main() \*/

uint64\_t stacksize;/\* if not zero, initial stack size \*/

};

LC\_LOAD\_DYLINKER 描述mach-o可执行文件加载器

struct dylinker\_command {

uint32\_t cmd; /\* LC\_ID\_DYLINKER, LC\_LOAD\_DYLINKER or

LC\_DYLD\_ENVIRONMENT \*/

uint32\_t cmdsize; /\* includes pathname string \*/

union lc\_str name; /\* dynamic linker's path name \*/

};

LC\_CODE\_SIGNATURE 用codesign和ldid(plist)签名生成的结构，用于突破沙盒等权限限制

struct linkedit\_data\_command {

uint32\_t cmd; /\* LC\_CODE\_SIGNATURE, LC\_SEGMENT\_SPLIT\_INFO,

LC\_FUNCTION\_STARTS, LC\_DATA\_IN\_CODE,

LC\_DYLIB\_CODE\_SIGN\_DRS or

LC\_LINKER\_OPTIMIZATION\_HINT. \*/

uint32\_t cmdsize; /\* sizeof(struct linkedit\_data\_command) \*/

uint32\_t dataoff; /\* file offset of data in \_\_LINKEDIT segment \*/

uint32\_t datasize; /\* file size of data in \_\_LINKEDIT segment \*/

};

LC\_ENCRYPTION\_INFO/LC\_ENCRYPTION\_INFO\_64 用于appstore加密程序

struct encryption\_info\_command {

uint32\_t cmd; /\* LC\_ENCRYPTION\_INFO \*/

uint32\_t cmdsize; /\* sizeof(struct encryption\_info\_command) \*/

uint32\_t cryptoff; /\* file offset of encrypted range \*/

uint32\_t cryptsize; /\* file size of encrypted range \*/

uint32\_t cryptid; /\* which enryption system,

0 means not-encrypted yet \*/

};

LC\_SYMTAB 符号表

LC\_UUID 文件唯一标识

## App目录和文件

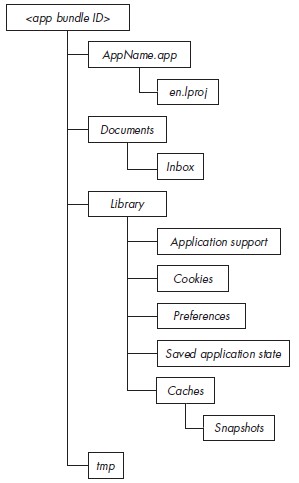
/var/mobile/Applications/[GUID]/=>

AppName.app 目录存放app静态数据和代码

Documents目录存放持久化数据，和iTunes同步；包括sql数据库

Library目录存放配置文件、缓存和cookie

tmp目录存放临时文件



AppName.app/Info.plist文件类似于android中得Androidmanifset.xml，在安装后以二进制形式存在，plutil用于转换原始xml和二进制xml

$ plutil -convert xml1 Info.plist -o -

$ plutil -convert xml1 Info.plist -o Info-xml.plist

$ plutil -convert binary1 Info-xml.plist -o Info-bin.plist

XCode可以解析文本和二进制形式的plist

# Objective C Reversing

研究方式：命令行编译+二进制对比+调试

Debug: clang/gcc –g -fobjc-arc -framework Foundation FKPerson.m main.m

Release: clang/gcc –O3 -fobjc-arc -framework Foundation FKPerson.m main.m

交叉编译arm： clang/gcc -x objective-c -arch armv7 -g -fobjc-arc -isysroot /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS9.3.sdk -framework Foundation main.m (arch=i386 x86\_64 armv7 arm64)

objective-c编译为c++源码：clang/gcc –rewrite-objc –framework Foundation main.m

一些名词：

meta-class 每个类都存在元类，

super-class 父类

root-class 根类

selector 选择器(存储为字符串，其内存位置与方法一一对应)

imp 普通函数指针

id 通用数据类型

## 杂项

字符串存储：

@”” => 实际编译为CFString结构

Class CFString : objc\_object

{

longlong info;

char\* data;//真正的字符串存储位置

longlong length;//字符串长度

}

synchronized锁：

@synchronized(expression1){

expression2;

}

实际编译生成为：

id lock = expression1

objc\_sync\_enter(lock)

expression2;

objc\_sync\_exit(lock)

选择器@selector：

@selector(x) => 实际编译为”x”

关键字@encode：

@encode(type) => 实际编译为 该类型的描述符

关键字@autorelease：

@autorelease{expression;} => 实际编译为

objc\_autoreleasePoolPush(…)

expression;

objc\_autoreleasePoolPop

快速枚举for：

for(type a in b) {expression;}=> 实际编译为

for(int i=0;i<b.selRef\_countByEnumeratingWithState\_objects\_count\_;i++){

expression;  
}

nil值：=> (void\*)0

## Function

传参所用寄存器，适用于普通函数和成员函数(id,sel)

|  |  |  |  |
| --- | --- | --- | --- |
| arm架构：  a1 R0  a2 R1  a3 R2  a4 R3  a5 [sp+0]  a6 [sp+4]  …. | arm64架构：  a1 W0  a2 W1  a3 W2  a4 W3  a5 W4  a6 W5  a7 W6  a8 [sp+0]  a9 [sp+8]  a10 [sp+16]  a11 [sp+24]  ……. | X86架构(默认调用约定)：  a1 [esp+0]  a2 [esp+4]  a3 [esp+8]  a4 [esp+8]  a5 [esp+12]  a6 [esp+16]  ……. | x86\_64架构：  a1 rdi  a2 rsi  a3 rdx  a4 rcx  a5 r8  a6 r9  a7 [rsp+0]  a8 [rsp+8]  …… |

## Block

用于定义匿名函数，等价于lambda表达式，形式如下：

^ [返回值类型] (类型1 形参1, 类型2 形参2, ...)

{

}

定义Block变量形式如下：

返回值类型 (^块变量名) (类型1, 类型2, ...);

   int (^hypot)(int, int) = ^(int num1, int num2)

        {

                 returnnum1 \* num1 + num2 \* num2;

        };

NSLog(@"%d",hypot(3,4));

编译得到：

  v3= ((int (\_\_fastcall \*)(\_QWORD, \_QWORD, \_QWORD))\*(&\_\_block\_literal\_global8 +2))(&\_\_block\_literal\_global8, 3LL, 4LL);

 NSLog(&cfstr\_D, (unsigned int)v3);

        其中\_\_block\_literal\_global8将函数等相关信息封装成类(这点和vs-win一致)，\_\_\_main\_block\_invoke\_2正是函数体实现：

\_\_const:0000000100001060 \_\_\_block\_descriptor\_tmp7dq 0           ; DATA XREF:\_\_const:0000000100001098o

\_\_const:0000000100001068                 dq 20h

\_\_const:0000000100001070                 dq offset aI16@?0i8i12  ; "i16@?0i8i12"

\_\_const:0000000100001078                 align 20h

\_\_const:0000000100001080\_\_\_block\_literal\_global8 dq offset \_\_NSConcreteGlobalBlock

\_\_const:0000000100001080                                         ; DATAXREF: \_main+87o

\_\_const:0000000100001088                 dq 50000000h

\_\_const:0000000100001090                 dq offset \_\_\_main\_block\_invoke\_2

\_\_const:0000000100001098                 dq offset\_\_\_block\_descriptor\_tmp7

从源码Block\_private.h可以得到构造的Block结构体

struct Block\_layout

{

   void \*isa;

   volatile int32\_t flags; // contains ref count

   int32\_t reserved;

   void (\*invoke)(void \*, ...);//实际调用的函数

   struct Block\_descriptor\_1 \*descriptor;

   // imported variables

};

struct Block\_descriptor\_1

{

   uintptr\_t reserved;

uintptr\_t size;

};

struct Block\_descriptor\_2

{

   void (\*copy)(void \*dst, const void \*src);

   void (\*dispose)(const void \*);

};

struct Block\_descriptor\_3

{

   const char \*signature;

   const char \*layout;     //contents depend on BLOCK\_HAS\_EXTENDED\_LAYOUT

};

从内部实现看，Block代码能生成3种类型：

NSGlobalBlock         代码中未操作外部变量或操作全局变量(如上例)

NSStackBlock          代码中操作外部栈变量

NSMallocBlock        代码中操作外部堆变量

下面分别讨论

第一种情况：

代码为最开始的例子，可见其中没有用到外部变量

实际产生的代码为：

int \_\_\_main\_block\_invoke(Block\_layout this,int num1, int num2)

{

        returnnum1 \* num1 + num2 \* num2;

}

Block\_layout \_\_block\_literal\_global =

{

        \_\_NSConcreteGlobalBlock,

        0x50000000,

        0,

        &\_\_\_main\_block\_invoke,

        &\_\_\_block\_descriptor\_tmp,

}

\_\_block\_literal\_global. \_\_\_main\_block\_invoke(&\_\_block\_literal\_global,3, 4);

第二种情况：

代码如下

        \_\_blockint my = argc;

        int(^hypot)(int, int) = ^(int num1, int num2)

        {

                 my+= 1;

                 returnnum1 \* num1 + num2 \* num2;

        };

        NSLog(@"%d%d", hypot(3, 4), my);

实际产生的代码为：

block\_descriptor \_\_\_block\_descriptor\_tmp =

{

        0,

        28,

        \_\_\_copy\_helper\_block\_,

        \_\_\_destroy\_helper\_block\_,

        "i16@?0i8i12",

        16

};

int \_\_\_main\_block\_invoke(Block\_layout this,int num1, int num2)

{

        this->\_\_\_stack\_variable->my+= 1;

        returnnum1 \* num1 + num2 \* num2;

}

Block\_layout \_\_block\_literal\_global =

{

        \_\_NSConcreteStackBlock,

        0xC2000000,

        0,

        &\_\_\_main\_block\_invoke,

        &\_\_\_block\_descriptor\_tmp,

        &\_\_\_stack\_variable//存放所有栈变量

};

void \_\_copy\_helper\_block\_()

{

        \_Block\_object\_assign(my,argc)

}

void \_\_destroy\_helper\_block\_()

{

        \_Block\_object\_dispose(my,argc)

}

\_\_block\_literal\_global.\_\_\_block\_descriptor\_tmp.\_\_\_copy\_helper\_block\_();

\_\_block\_literal\_global. \_\_\_main\_block\_invoke(&\_\_block\_literal\_global,3, 4);

..................

\_\_block\_literal\_global.\_\_\_block\_descriptor\_tmp.\_\_\_destroy\_helper\_block\_();

第三种情况：

需要开启arc，暂无研究

## Class

### 内部结构体一览

|  |  |
| --- | --- |
| Object描述通用对象，所有类继承自该类 | struct objc\_object{  Class isa; //描述类型  } |
| Class描述类，相当于模板，创建实例和使用静态方法时使用 | struct objc2\_class : objc2\_object{//runtime/  //Class isa; //Class对象的Class即meta class  Class superclass; //父类Class  cache\_t cache; //缓存调用过的成员函数  objc2\_class\_rw\* data; } |
| class\_rw动态类数据，内存中呈现形式 | struct objc2\_class\_rw{//runtime/objc-runtime-new.h class\_rw\_t  int flags; //标志位  int version;  objc2\_class\_ro\* ro;  method\_array\_t methods; //链表结构方便随时添加函数  property\_array\_t properties  protocol\_array\_t protocols;  Class firstSubclass;  Class nextSiblingClass;  char\* demangledName;  }  类属性flags  RW\_REALIZING 0x80000 class has started realizing  RW\_HAS\_INSTANCE\_SPECIFI |
| class\_ro静态类数据，二进制文件中呈现形式，在初始化后设置REALIZE转化成新结构class\_rw | struct objc2\_class\_ro{//runtime/objc-runtime-new.h class\_ro\_t  int flags; //标志位  int instanceStart; //在Instance中第一个ivar偏移  int instanceSize; //Instance大小  int reserved;  byte\* ivar\_layout;  char\* name; //对应类名  objc2\_meth\_list\* base\_meths; //类拥有的成员方法（静态成员在metaclass中）  objc2\_prot\_list\* base\_prots; //类遵守的接口  objc2\_ivar\_list\* ivars; //类拥有的成员变量  byte\* weak\_ivar\_layout;  objc2\_prop\_list\* base\_props; //使用@property定义的属性  }  类属性flags  RO\_META 1 meta-class  RO\_ROOT 2 root-class  RO\_HAS\_CXX\_STRUCTORS 4 has .cxx\_construct/destruct  RO\_HAS\_LOAD\_METHOD 8 has +load  RO\_HIDDEN 16 visibility=hidden  RO\_EXCEPTION 32 has attribute(objc\_exception)  RO\_REUSE\_ME 64 available for reassignment  RO\_IS\_ARR 128 class compiled with –fobjc-arc  RO\_HAS\_CXX\_DTOR\_ONLY 256 has .cxx\_destruct but no .cxx\_construct  RO\_FROM\_BUNDLE 0x20000000 class is in unloadable bundle  RO\_FUTURE 0x40000000 class is unrealized future  RO\_REALIZED 0x80000000 class is realized |
| Instance——实例结构，操作实例或类成员函数中使用 | struct objc\_instance : objc\_object{  //Class isa;  Member1; //成员变量1,2,3….  Member2;  } |
| Method List——描述类结构中包含的成员函数 | struct objc2\_method\_list{  int entrySize; //每个objc2\_method结构大小  int count; //后接count个objc2\_method  } |
| Method——描述单个成员函数 | struct objc2\_method {  SEL method\_name //方法名 setName:andAge:  char \*method\_types //方法类型 v28@0:8@16i24  IMP method\_imp //实际地址 ptr of setNameandAge  }  成员函数指针定义：typedef id (\*IMP)(id, SEL, ...);  函数修饰符method\_types runtime.h  ‘b’-bitfield  ‘B’-bool  ‘c’-char  ‘C’-uchar  ‘d’-double  ‘f’-float  ‘i’-int  ‘I’-uint  ‘l’-long  ‘L’-ulong  ‘n’-in for input  ‘N’-inout both for input and output  ‘o’-out for ouput  ‘O’-bycopy instead of using a proxy/NSDistantObject, pass or return a copy of the object  ‘q’-longlong  ‘Q’-ulonglong  ‘r’-const constant  ‘R’-byref use a proxy(default)  ‘s’-short  ‘S’-ushort  ‘v’-void  ‘V’-oneway 允许在不同线程和进程使用，不可阻塞调用线程直到返回  ‘^’-pointers  ‘@’-object  ‘[‘-array begin  ‘]’-array end  ‘{‘-structure begin  ‘}’-structure end  ‘(‘-union begin  ‘)’-union end  ‘#’-class  ‘:’-selector  ‘\*’-char pointer  ‘%’-atom  ‘!’-vector  ‘?’-undefine  Structure: returntype—stacksize—[argumenttype—bitoffset]\*  v28@0:8@16i24 -> void stacksize=28 (pointer, selector, pointer, int) |
| Ivar List | struct objc2\_ivar\_list{  int entrySize; //每个objc2\_ivar结构大小  int count; //后接count个objc2\_ivar  } |
| Ivar | struct objc2\_ivar{  int\* offset; //存储该变量相对Instance结构偏移  char\* name; //变量名  char\* type; //类型描述符  int alignment\_raw; //对齐  int size; //变量占用空间  } |
| Protocol List——描述遵守的接口 | struct objc2\_protocol\_list{  longlong count;//后接count个Protocol  } |
| Protocol——描述单个接口 | struct objc2\_protocol : objc\_object{  //Class isa;  char\* mangledName;  objc2\_protocol\_list\* protocols  objc2\_method\_list\* instanceMethods;  objc2\_method\_list\* classMethods;  objc2\_method\_list\* optionalInstanceMethods;  objc2\_method\_list\* optionalClassMethods;  objc2\_method\_list\* instanceProperties;  int size;  int flag;  char\*\* extendedMethodTypes;  char\* \_demangledName; } |
| Property List——描述使用@property关键字定义的成员变量（和普通成员变量分开存放） | struct objc2\_prop\_list{  int entrySize; //每个objc2\_prop结构大小  int count; //后接count个objc2\_prop  } |
| Property——描述单个@property成员变量 | struct objc2\_prop{  char\* name;  char\* attributes;// T@"NSString",&,V\_a1  } |

### 成员函数分析

[receiver, selector]的成员函数调用会生成如下伪代码([对象 class] == [类 class])：

|  |  |
| --- | --- |
| OBJC代码 | 伪代码 |
| 返回普通类型的静态成员函数调用  [FKPerson foo] | objc\_msgSend(**[**FKPerson class], **“**foo”)  void \_cdecl foo(FKPerson\* self, SEL selector) |
| 返回普通类型的普通成员函数调用  [person say] | objc\_msgSend(person, **“**say”)  void \_cdecl say(FKPerson \* self, SEL selector) |
| 返回普通类型的多参数成员函数调用  [person setName:@”1” andAge:500] | objc\_msgSend(person, **“**setName:andAge:”, @”1”, 500)  void \_cdecl setName:andAge:(FKPerson\* self, SEL selector, NSString\* name, int age) |
| 成员函数中调用父类函数，父函数返回普通类型  [super init] | objc\_msgSendSuper(make\_super super, “init”) |
| 返回栈结构体的成员函数调用  [person func] | objc\_msgSend\_stret(person, “func”) |
| 成员函数中调用父类函数，父函数返回栈结构体  [super func] | objc\_msgSendSuper\_stret(self, “func”) |
| 返回栈浮点数 | arm:不使用objc\_msgSend\_fpret  i386:float|double|long double使用objc\_msgSend\_fpret  x86-64:long double使用objc\_msgSend\_fpret |

成员函数分析：

1. 每增加一个成员函数，类模板会增加method，由于名称一一对应，同一个类不允许存在同名函数
2. 每个成员函数前两个参数分别是实例指针self和选择器SEL，之后才是用户为其定义的参数
3. 带(+)修饰的成员函数本质为静态成员，属于该类的meta-class类成员，因此位于meta-class函数表中，而普通成员函数位于该类的函数表中
4. 和c++不同的是，成员函数调用方式和普通函数相同，因此可以通过反射替换成普通函数

成员变量分析：

1． 每增加一个成员变量，类模板Class会增加ivar，以后使用该类模板创建的实例的对象结构也会增加该元素

2． 只要有一个实际使用的成员变量，就会产生”类名.cxx\_destruct”析构函数

3． 根据成员变量属性为weak/strong，在进行赋值操作时采用objc\_storeWeak/objc\_storeStrong，默认类型为strong

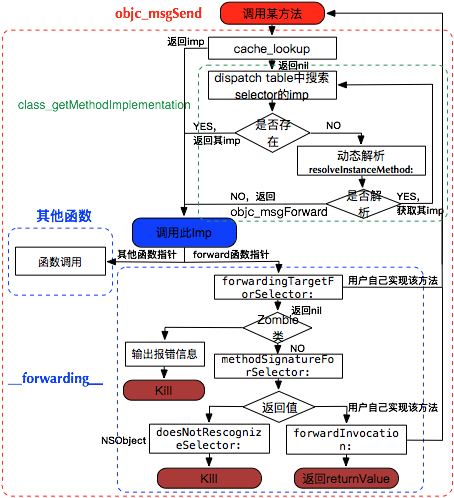
4. 对public成员变量的操作语法采用myclass->field形式，产生的逻辑也和c结构体相同，而更常规的方式是将成员变量写成@property中，这样编译器会自动为成员变量生成相应的getter和setter函数，使用kvc(键值编码)时会自动调用(msgsend)这些函数

meta-class存在的原因：

1. 直接从类对象进行的操作，例如调用静态成员函数，并不属于某个实例，因此需要存在于类类型中

2. 当自身被子类化(setsuperclass)时，父类并不等同于所属类(isa != superclass)，同理构造一个类要提供其isa

### Objc\_msgSend调用流程

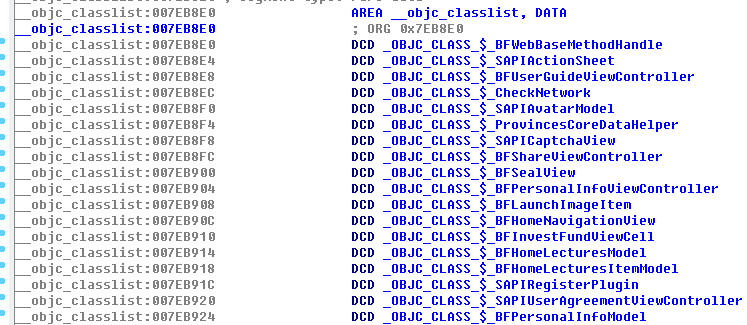


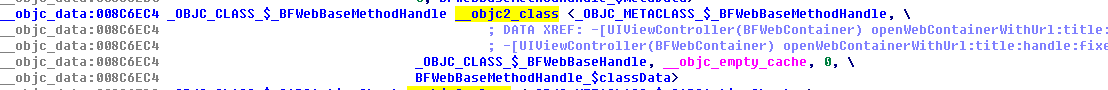
1. 根据对象的isa找到类，在类的dispatch table中查找selector
2. 如果未找到则找到该类的父类，并在父类的dispatch table中查找selector，直到NSObject类（该过程中优先查找cache）
3. 如果所有子类和父类都无法找到该函数，则进行msgForward，如果用户添加了动态实现(resolveInstanceMethod)则调用
4. 如果上一步失败，则尝试找到一个能响应该消息的对象(forwardingTargetForSelector)，如果能找到则转发给他
5. 如果上一步失败，则尝试获取一个方法签名(methodSignatureForSelector)，如果获取不到直接抛异常
6. 调用用户自己实现的forwardInvocation

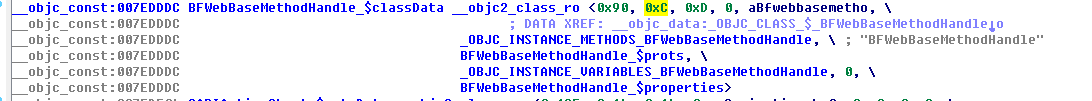


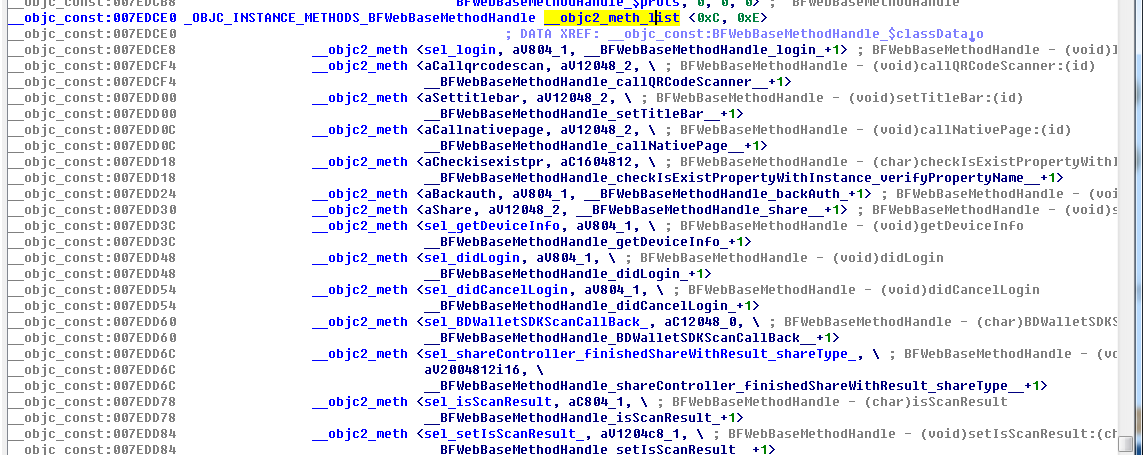
### Runtime Ability

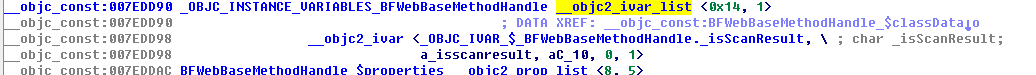
|  |  |
| --- | --- |
| Object | id object\_copy(id obj, size\_t size) 拷贝实例内容  Class object\_getClass(id obj) 返回Instance对应的Class  Class object\_setClass(id obj, Class cls) 绑定Instance的Class  BOOL object\_isClass(id obj) 判断所属  char\* object\_getClassName(id obj) 获取类名  void\* object\_getIvar(id obj, Ivar ivar) 获取成员变量值(按Ivar)  void object\_setIval(id obj, Ivarl ivar, id value) 设置成员变量值  Ivar object\_getInstanceVariable(id obj, char\* name, void\* value) 获取成员变量值(按变量名)  Ivar object\_setInstanceVariable(id obj, char\* name, void\* value) 设置成员变量值 |
| Class | Class objc\_getClass(char\* name) 返回指定类名的(类型)对象  Class objc\_getaMetaClass(char\* name) 返回指定类名的meta-class对象  Class objc\_lookUpClass(char\* name) 返回指定类名且已注册的的(类型)对象  int objc\_getClassList(Class\* buffer, int bufferCount) 返回所有已注册类  Class\* objc\_copyClassList(int\* outCount) 返回所有已注册类  char\* class\_getName(Class cls) 获取类名  BOOL class\_isMetaClass(Class cls) 是否meta-class  Class class\_getSuperClass(Class cls) 获取父类  Class class\_setSuperClass(Class cls, Class newSuper)设置父类  int class\_getVersion(Class cls) 获取版本  void class\_setVersion(Class cls, int version) 设置版本  size\_t class\_getInstanceSize(Class cls) 获取实例大小  Ivar class\_getInstanceVariable(Class cls, char\*name) 获取实例Ivar  Ivar\* class\_copyIvarList(Class cls, int\* outCount)  Method class\_getInstanceMethod(Class cls, SEL name) 获取非静态方法  Method class\_getClassMethod(Class cls, SEL name) 获取静态方法  IMP class\_getMethodImplementation(Class cls, SEL name) 获取方法实现  BOOL class\_conformsToProtocol(Class cls, Protocol\* protocol)是否遵守协议  Method\* class\_copyMethodList(Class cls, int\* outCount)  Protocol\* class\_copyProtocolList(Class cls, int\* outCount)  objc\_property\_t class\_getProperty(Class cls, char\* name)  objc\_property\_t class\_copyPropertyList(Class cls, int\* outCount)  uchar\* class\_getIvarLayout(Class cls)  BOOL class\_addMethod(Class cls, SEL name, IMP imp, char\* types) 增加函数(绑定普通函数)  BOOL class\_replaceMethod(Class cls, SEL name, IMP imp, char\* types)替换函数  BOOL class\_addIvar(Class cls, char\* name, size\_t size, uchar alignment, char\* types)添加变量  BOOL class\_addProtocol(Class cls, Protocol\* protocol) 增加协议  BOOL class\_addProperty(Class cls, char\* name, objc\_property\_attribute\_t\* attrib,int count)  BOOL class\_replaceProperty(Class cls, char\* name, objc\_property\_attribute\_t\* attrib,int count)  void class\_setIvarLayout(Class cls, uchar layout)  id class\_createInstance(Class cls, size\_t extrabytes) 创建实例  id objc\_constructInstance(Class cls, void\* bytes) 创建实例  void\* objc\_destructInstance(id obj)  Class objc\_allocateClassPair(Class superclass, char\* name, size\_t extrabytes) 创建类和元类  void objc\_registerClassPair(Class cls) 注册类  Class objc\_duplicateClass(Class original, char\* name, size\_t extraBytes) 复制类 |
| Method | SEL method\_getName(Method m) 获取函数名  int method\_getNumberOfArguments  char\* method\_getTypeEncoding 获取函数类型字段  void method\_getArgumentType 获取参数类型  void method\_getReturnType 获取返回值类型  IMP method\_getImplementation  IMP method\_setImplementation(Method m, IMP imp)设置函数实现  void method\_exchangeImplementations(Method m1, Method m2) |
| Ivar | char\* ivar\_getName(Ivar v) 获取Ivar名  char\* ivar\_getTypeEncodeing(Ivar v) 获取Ivar类型字段  ptrdiff\_t ivar\_getOffset(Ivar v) 获取该ivar在instance中得偏移 |
| Attribute | …… |
| Protocol | objc\_copyProtocolList  protocol\_getName  protocol\_copyProtocolList  protocol\_allocateProtocol  protocol\_registerProtocol  protocol\_addProtocol  protocol\_addProperty |
| 其他 | char\*\* objc\_copyImageNames(int\* outcount) 获取加载的动态库  char\* class\_getImageName(Class cls) 获取某类所属动态库  char\*\* objc\_copyClassNamesForImage(char\* image, int\* outCount)获取动态库中所有类  objc\_loadWeak 获取weak值  objc\_storeWeak 设置weak值  objc\_setAssociatedObject 设置关联  objc\_getAssociatedObject 获取关联  objc\_removeAssociatedObjects 移除所有关联，恢复对象到原始状态 |

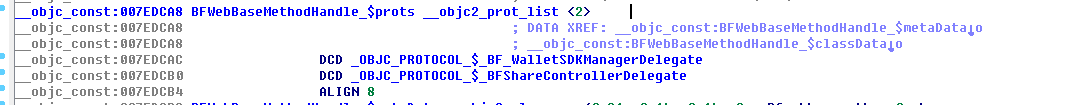


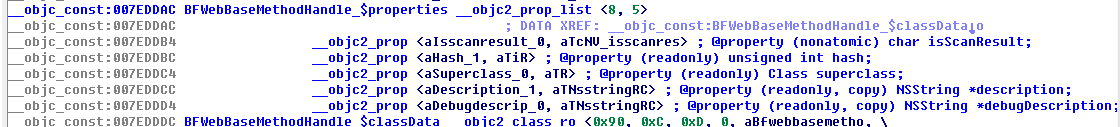












### 其他

arc类型转换：

普通指针和objc指针转换：（实现调试器中任意内存当作类操作）

id obj1 = [[class1 alloc] init];

void\* p = (\_\_bridge void\*)obj1;

id obj2 = (\_\_bridge id)p;

@property：

@property(?,?,…)用于快速生成类成员及getter setter，其修饰符如下：

atomic 原子操作，线程安全(默认)

objc\_getProperty objc\_setProperty\_atomic

nonatomic 非线程安全 ‘N’

readwrite 具有setter getter(默认)

readonly 具有getter ‘R’

assign 简单赋值(默认)

copy setter方法中深度复制传入对象 ‘C’

objc\_getProperty objc\_setProperty\_atomic\_copy

retain setter方法中对传入对象引用计数加一 ‘&’

strong 强引用(默认)，和retain相似 ‘&’

初始化/赋值= 销毁objc\_storeStrong

weak 对象消失后指针置nil ‘W’

初始化objc\_initWeak 赋值objc\_loadWeakRetained 销毁objc\_destroyWeak/objc\_autoreleaseReturnValue

\_\_unsafe\_unretain 对象引用计数不加一，对象释放后不置nil

autorelease 对象加入自动释放池 对应objc\_autorelease

## 异常处理

objc提供异常处理机制

@try{

expr1;

}

@catch(NSException\* ex){

expr2;

}  
@finally{

expr3;

}  
产生的流程如下：

......

flag = 0

expr1

label1:

expr3

...

if(flag & 1)

objc\_exception\_rethrow()

return

tail:

if(..)

{

expr2;

}

goto label1;

@throw语句层产生：objc\_exception\_throw()

## Reflection

Objective-C是一种反射型语言，可以在运行时获取和修改自身状态，其中的实现存在于libobjc.A.dylib库中，这些“运行时”能力源于objective-c类结构组织较为灵活，并提供了操作自身结构的接口，同时在生成的可执行文件(mach-o)中存在\_OBJC节，这些节中提供了足够的类构成信息，而Mac端gdb可以解析这些结构，而正由于objc提供了如此多的信息，因此也比c++在同等情况下逆向难度低一些。

LC\_SEGMENT.\_\_OBJC.\_\_cat\_cls\_meth

LC\_SEGMENT.\_\_OBJC.\_\_cat\_inst\_meth

LC\_SEGMENT.\_\_OBJC.\_\_string\_object

LC\_SEGMENT.\_\_OBJC.\_\_cstring\_object

LC\_SEGMENT.\_\_OBJC.\_\_message\_refs

LC\_SEGMENT.\_\_OBJC.\_\_sel\_fixup

LC\_SEGMENT.\_\_OBJC.\_\_cls\_refs

LC\_SEGMENT.\_\_OBJC.\_\_class

LC\_SEGMENT.\_\_OBJC.\_\_meta\_class

LC\_SEGMENT.\_\_OBJC.\_\_cls\_meth

LC\_SEGMENT.\_\_OBJC.\_\_inst\_meth

LC\_SEGMENT.\_\_OBJC.\_\_protocol

LC\_SEGMENT.\_\_OBJC.\_\_category

LC\_SEGMENT.\_\_OBJC.\_\_class\_vars

LC\_SEGMENT.\_\_OBJC.\_\_instance\_vars

LC\_SEGMENT.\_\_OBJC.\_\_module\_info

LC\_SEGMENT.\_\_OBJC.\_\_symbols

java与objc反射对比：

|  |  |  |
| --- | --- | --- |
|  | objc | java |
| 获取类 | NSClassFromString  myClass.class [myClass class] | Class.forName  myClass.class |
| 检查继承 | isKindOfClass 是否为该类或其子类的实例  isMemberOfClass 是否为该类实例  conformsToProtocol是否遵循接口 | class.isAssignableFrom 是否有继承关系  instanceOf 是否为该类实例  class.isInstance是否为该类实例 |
| 获取函数 | @selector  NSSelectorFromString | getMethod |
| 调用函数 | perfromSelector  objc\_msgSend | invoke |
| 其他 | 见Runtime Ability |  |

https://developer.apple.com/library/mac/documentation/Cocoa/Reference/ObjCRuntimeRef/#//apple\_ref/doc/constant\_group/Associative\_Object\_Behaviors

## 重要的库和函数

dyld 进程加载器

libobjc.A.dylib runtime

# iOS Attack&Defense

## AntiDebug - AntiAntiDebug

### sysctl P\_TRACED标志 检测调试

可以检测调试器和跟踪器，但是不能检测注入和cycript：

#include <sys/types.h>

#include <sys/sysctl.h>

static int check\_debugger( ) \_\_attribute\_\_((always\_inline));

int check\_debugger( )

{

size\_t size = sizeof(struct kinfo\_proc);

struct kinfo\_proc info;

int ret,name[4];

memset(&info, 0, sizeof(struct kinfo\_proc));

name[0] = CTL\_KERN;

name[1] = KERN\_PROC;

name[2] = KERN\_PROC\_PID;

name[3] = getpid();

if((ret = (sysctl(name, 4, &info, &size, NULL, 0)))){

return ret; //sysctl() failed for some reason

}

return (info.kp\_proc.p\_flag & P\_TRACED) ? 1 : 0;

}

### ptrace PT\_DENY\_ATTACH 防止调试

可以阻止调试器附加：

#import <dlfcn.h>

#import <sys/types.h>

typedef int (\*ptrace\_ptr\_t)(int \_request, pid\_t \_pid, caddr\_t \_addr, int \_data);

#if !defined(PT\_DENY\_ATTACH)

#define PT\_DENY\_ATTACH 31

#endif // !defined(PT\_DENY\_ATTACH)

void disable\_gdb() {

void\* handle = dlopen(0, RTLD\_GLOBAL | RTLD\_NOW);

ptrace\_ptr\_t ptrace\_ptr = dlsym(handle, "ptrace");

ptrace\_ptr(PT\_DENY\_ATTACH, 0, 0, 0);

dlclose(handle);

}

相同效果的还有syscall(26,31,0,0)和汇编版本的代码：

#ifdef \_\_arm\_\_

asm volatile(

“mov r0,#31\n”

“mov r1,#0\n”

“mov r2,#0\n”

“mov r12,#26\n”

“svc #80\n”

#endif

#ifdef \_\_arm64\_\_

asm volatile(

“mov x0,#26\n”

“mov x1,#31\n”

“mov x2,#0\n”

“mov x3,#0\n”

“mov x16,#0\n”

“svc #128\n”

#endif

直接附加调试器会产生segmentation fault:11 启动调试程序会在ptrace执行后退出

//调试已经被调试的进程：直接失败产生日志：(os/kern) invalid task Exiting

反反调试：hook相应函数

#import <substrate.h>

#if !defined(PT\_DENY\_ATTACH)

#define PT\_DENY\_ATTACH 31

#endif

static int (\*\_ptraceHook)(int request, pid\_t pid, caddr\_t addr, int data);

static int $ptraceHook(int request, pid\_t pid, caddr\_t addr, int data) {

if (request == PT\_DENY\_ATTACH) {

request = -1;

}

return \_ptraceHook(request,pid,addr,data);

}

%ctor {

MSHookFunction((void \*)MSFindSymbol(NULL,"\_ptrace"), (void \*)$ptraceHook, (void \*\*)&\_ptraceHook);

}

### isatty检测调试

isatty函数在给定文件描述符被附加到调试器控制台时返回1，否则返回0

http://sourceware.org/gdb/onlinedocs/gdb/isatty.html

if(isatty(1)){

NSLog(@”Being Debugged isatty”);

}

else{

NSLog(@”isatty() bypassed”);

}

task\_get\_exception\_ports检测调试

调试器通常会监听异常端口，因此可以用task\_get\_exception\_ports循环遍历以校验该端口是否设置

struct ios\_execp\_info{

exception\_mask\_t masks[EXC\_TYPES\_COUNT];

mach\_port\_ports[EXC\_TYPES\_COUNT];

exception\_behavior\_t behaviors[EXC\_TYPES\_COUNT];

thread\_state\_flavor\_t flavors[EXC\_TYPES\_COUNT];

mach\_msg\_type\_number\_t count;

}

struct ios\_execp\_info\* info = malloc(sizeof(struct ios\_execp\_info));

kern\_return\_t kr = task\_get\_exception\_ports(mach\_task\_self(),EXC\_MASK\_ALL,info->masks,&info->count,info->ports

,info->behaviors,info->flavors);

for(int i=0;i<info->count;i++){

if(info->ports[i] != 0 || info->flavors[i] == THREAD\_STATE\_NONE){

NSLog(@”Beging debugged”);

}

else{

NSLog(@“bypassed”);

}

}

### \_RESTRICT节——防注入

加载器dyld(ios7.0以后)源码中关于DYLD\_环境变量的逻辑pruneEnvironmentVariables

switch (sRestrictedReason) {

case restrictedNot:

break;

case restrictedBySetGUid:

dyld::log("main executable (%s) is setuid or setgid\n", sExecPath);

break;

case restrictedBySegment:

dyld::log("main executable (%s) has \_\_RESTRICT/\_\_restrict section\n", sExecPath);

break;

case restrictedByEntitlements:

dyld::log("main executable (%s) is code signed with entitlements\n", sExecPath);

break;

}

3种情况下DYLD环境变量会被忽视：

1. 可执行文件设置了setuid setgid位
2. 可执行文件有\_\_restrict节
3. 可执行文件有特殊代码签名

由于受app store的限制，1和3都不能实现，而2可以设置Other linker flags为-Wl,-sectcreate,\_\_RESTRICT,\_\_restrict,/dev/null

使用该方法可以禁止dylib的注入，在生成的mach-o文件中会多出一个\_\_RESTRICT节

这种方式是可以防止启动注入和运行时注入的，尝试用dumpdecrypted脱壳时会产生类似如下的日志：

dyld: warning, LC\_RPATH @executable\_path/Frameworks in /var/mobile/Applications/[id]/?.app/\* being ignored in restricted program because of @executable\_path

尝试用cycript附加会产生如下输入：

dlopen(/usr/lib/libcript.lib, 5): Library not loaded: /System/Library/PrivateFrameworks/JavaScriptCore.framework/JavaScriptCore

referenced from: /usr/lib/libcript.dylib

reason: image not found

\*\*\* \_assert(status == 0):../Inject.cpp(143):InjectLibrary

而使用lldb则可以正常附加调试

anti-anti-debug：修改restrict节名，重签名(ldid –S)即可

## JailBreak Detect – Anti JailBreak Detect

iOS的越狱相当于Android上的root，获取root权限

### 沙盒完整性检测

iOS设备上，用户app安装在/var/mobile/Application中受沙盒限制，而系统app安装在/Application中不受沙盒限制。越狱设备上很多第三方app也安装在/Application下从而不受沙盒限制而拥有更多权限。一些越狱工具会移除沙盒限制以允许特定行为(如fork vfork popen)

int result = fork();

if(!result)

exit(0);

if(result >= 0)//jail broken

{sandbox\_is\_compromised = 1};

监测点2：在沙盒中，执行opendir(“/dev”)会返回NULL

监测点3：system() getgid() ??

### 文件系统检测

检测常见的越狱工具目录和文件是否存在

struct stat s;

int is\_jailbroken = stat(“/Applications/Cydia.app”, &s) == 0;

常见的目录和文件

/Applications/MxTube.app

/Applications/blackra1n.app

/Applications/RockApp.app

/Applications/WinterBoard.app

/Applications/SBSettings.app

/Library/LaunchDaemons/com.openssh/sshd.plist

/Applications/IntelliScreen.app

/Library/MobileSubstrate/DynamicLibraries/Veency.plist

/Applications/FakeCarrier.app

/private/var/mobile/Library/SBSettings/Themes

/System/Library/LaunchDaemons/com.saurik.Cydia.Startup.plist

/Library/MobileSubstrate/DynamicLibraries/LiveClock.plist

/System/Library/LaunchDaemons/com.ikey.bbot.plist

/Applications/Icy.app

/Applications/Loader.app

/private/var/tmp/cydia.log

/Library/MobileSubstrate/MobileSubstrate.dylib

/private/var/stash

/private/var/lib/apt

/private/var/lib/cydia

/usr/libexec/cydia

/usr/libeec/sftp-server

/var/cache/apt

/var/lib/apt

/var/lib/cydia

/var/log/syslog

/var/tmp/cydia.log

/var/tmp/cydia.log

/bin/bash

/bin/sh

/usr/sbin/sshd

/bin/mv

/usr/libexec/ssh-keysign

/etc/ssh/sshd\_config

/etc/apt

检测装载点

越狱工具会替换/etc/fstab文件导致变小，IOS5上正常为80字节

struct stat s;

stat(“/etc/fstab”, &s);

return s.st\_size;

第二个监测点：statfs函数检测

在非越狱机上statfs(“/”)应该返回如下标志：buf->f\_flags = MNT\_RDONLY + MNT\_ROOTFS + MNT\_DOVOLFS + MNT\_JOURNALED + MNT\_MULTILABEL，同时statfs(“/var/mobile/Container/Data/Application/<APP\_GUID>”)应该返回如下标志：buf->f\_flags = MNT\_NOSUID + MNT\_NODEV + MNT\_DOVOLFS + MNT\_JOURNALED + MNT\_MULTILABEL

检测软链接

检测/Appliations软链接，越狱工具会将其替换到/var/stash/…下

/Appliations软链接，越狱工具会将其替换到/var/stash/…下

struct stat s;

if(lstat(“/Applications”, &s) != 0)[  
 if(s.st\_mode & S\_IFLNK)

exit(-1);

}

其他软链接

/Library/Ringtones

/Library/WallPaper

/usr/arm-apple-darwin9

/usr/include

/usr/libexec

/usr/share

/var/stash/Library/Ringtones

/var/stash/usr/include

/var/stash/Library/WallPaper

/var/stash/usr/libexec

/var/stash/usr/share

/var/stash/usr/arm-apple-darwin9

### URL Scheme检测

在越狱机上Cydia会创建一个cydia://的URL scheme，因此如果调用该Scheme返回成功则机器越狱

[NSURL URLWithString @”cydia://package/com.example.package”]

### 系统内核环境变量检测

越狱时会增加2个内核环境变量用于绕过iOS代码签名机制，sysctlbyname函数用于检测系统信息，在非越狱机上，下面值应该为1

sysctlbyname(security.mac.proc\_enforce)

sysctlbyname(security.mac.vnode\_enforce

检测DYLD\_INSERT\_LIBRARIES是否存在，越狱环境下会出现”/Library/MobileSubstrate/MobileSubstrate.dylib”

getenv(“DYLD\_INSERT\_LIBRARIES”) ，检测返回NULL和”\0”

### 运行进程检测

@try{

NSArray\* processes = [self runningProcesses]

for(NSDictionary\* dict in processes){

NSString\* process = [dict objectForKey:@”ProcessName”];

if([process isEqualToString:@”MobileCydia”]){

return true;

}

else if([process isEqualToString:”Cydia”]){

return true

}

}

}

@catch(NSException\* exception){

return 0  
}

+ (NSArray\*)runningProcesses{

int mib[4] = {CTL\_KERN,KERN\_PROC,KERN\_PROC\_ALL,0};

size\_t miblen =4;

size\_t size;

int st = sysctl(mib,miblen,NULL,&size,NULL,0);

struct kinfo\_proc\* process = NULL;

struct kinfo\_proc\* newprocess = NULL;

do{

size += size/10

newprocess = realloc(process,size);

if(!newprocess){

if(process){

free(process);

}

return nil;

}

int st = sysctl(mib,miblen,NULL,&size,NULL,0);

st = sysctl

}while(st == -1 && errno == ENOMEM);

}

if(st == 0){

if(size % sizeof(struct kinfo\_proc) == 0){

int nprocess = size/sizeof(struct kinfo\_proc);

if(nprocess){

NSMutableArray\* array = [[NSMutableArray alloc] init];

for(int I = nprocess – 1;I >= 0;i--){

NSString\* processID = [[NSString alloc] initWithFormat:@”%d”,process[i].kp\_proc.p\_pid];

NSString\* processName = [[NSString alloc] initWithFormat:@”%d”,process[i].kp\_proc.p\_comm];

NSString\* processPriority = [[NSString alloc] initWithFormat:@”%d”,process[i].kp\_proc.p\_priority];

NSDate\* processStartDate = [NSDate dateWithTimeInternvalSince1970:process[i].kp\_proc.p\_un.\_\_p\_starttime.tv\_sec];

NSDictionary\* dict = [[NSDictionary alloc] initWithObjects:[NSArray arrayWithObjects:processID, processPriority, processName, processStartDate, nil] forKeys:[NSarray arrayWithObject:@”ProcessID”, @”ProcessPriority”, @”ProcessName”, @”ProcessStartDate”, nil]];

[array addObject:dict];

}

free(process);

return array;

}

}

return nil;

}

反检测方式：hook

## Injection Detect ——检测注入

void dylibCheck(){

uint32\_t count = \_dyld\_image\_count();

char\* substrate = “/Library/MobileSubstrate/MobileSubstrate.dylib”;

for(uint32\_t i = 0;i<count;i++){

const char\* dyld = \_dyld\_get\_image\_name();

if(strcmp(dyld.substrate) == 0)

NSLog(@”Substrate found”);

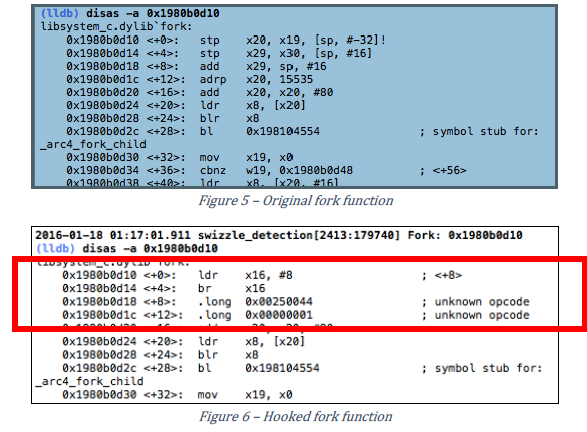
}

}

anti-injection detect hook dyld\_image\_? func

## Function Hook Sinature ——检测hook

用于检测inline hook，下图为fork函数在hook之前和之后的机器码



# 其他

## 工具扩展

功能：

1. 利用hook框架在所有objc\_msgSend消息中筛选并显示指定selector的调用，用于检测代码覆盖，对于单一消息的详细检测，配合frida使用
2. 利用运行时函数打印oc对象结构
3. 打印模块
4. 去除各种形式的反调试
5. 动态敏感函数检测
6. 自动化测试

utility.h

utility.m

utility.mm

使用方式：XCode编译dylib，使用各种方式注入到macho即可

## mach-o补丁

功能：

1. 打印macho格式关键点
2. 解除模块随机化
3. 解除反注入
4. 注入
5. 提取armv7版本文件

使用方式：python extractmacho.py [–dump-macho] [–remove-pie] [–remove-restrict] [–injectdylib=/??/test1.dylib] [-injectdylib=/??/test2.dylib] …. /path/to/BaiduFinance

## 显示ios app表

在ios上获取app对应可执行文件的路径比较麻烦，这里通过脚本定位

showapp.sh

使用方式：上传到iOS设备/usr/local/bin执行

## 定位关键函数

字符串引用

关键函数回溯栈

oc消息跟踪

## LLDB常用命令

|  |  |
| --- | --- |
| **GDB** | **LLDB** |
| Launch a process no arguments. | |
| **(gdb)** run **(gdb)** r | **(lldb)** process launch **(lldb)** run **(lldb)** r |
| Launch a process with arguments <args>. | |
| **(gdb)** run <args> **(gdb)** r <args> | **(lldb)** process launch -- <args> **(lldb)** r <args> |
| Launch a process for with arguments **a.out 1 2 3** without having to supply the args every time. | |
| **%** gdb --args a.out 1 2 3 **(gdb)** run ... **(gdb)** run ... | **%** lldb -- a.out 1 2 3 **(lldb)** run ... **(lldb)** run ... |
| Or: | |
| **(gdb)** set args 1 2 3 **(gdb)** run ... **(gdb)** run ... | **(lldb)** settings set target.run-args 1 2 3 **(lldb)** run ... **(lldb)** run ... |
| Launch a process with arguments in new terminal window (Mac OS X only). | |
|  | **(lldb)** process launch --tty -- <args> **(lldb)** pro la -t -- <args> |
| Launch a process with arguments in existing terminal /dev/ttys006 (Mac OS X only). | |
|  | **(lldb)** process launch --tty=/dev/ttys006 -- <args> **(lldb)** pro la -t/dev/ttys006 -- <args> |
| Set environment variables for process before launching. | |
| **(gdb)** set env DEBUG 1 | **(lldb)** settings set target.env-vars DEBUG=1 **(lldb)** set se target.env-vars DEBUG=1 **(lldb)** env DEBUG=1 |
| Unset environment variables for process before launching. | |
| **(gdb)** unset env DEBUG | **(lldb)** settings remove target.env-vars DEBUG **(lldb)** set rem target.env-vars DEBUG |
| Show the arguments that will be or were passed to the program when run. | |
| **(gdb)** show args Argument list to give program being debugged when it is started is "1 2 3". | **(lldb)** settings show target.run-args target.run-args (array of strings) = [0]: "1" [1]: "2" [2]: "3" |
| Set environment variables for process and launch process in one command. | |
|  | **(lldb)** process launch -v DEBUG=1 |
| Attach to a process with process ID 123. | |
| **(gdb)** attach 123 | **(lldb)** process attach --pid 123 **(lldb)** attach -p 123 |
| Attach to a process named "a.out". | |
| **(gdb)** attach a.out | **(lldb)** process attach --name a.out **(lldb)** pro at -n a.out |
| Wait for a process named "a.out" to launch and attach. | |
| **(gdb)** attach -waitfor a.out | **(lldb)** process attach --name a.out --waitfor **(lldb)** pro at -n a.out -w |
| Attach to a remote gdb protocol server running on system "eorgadd", port 8000. | |
| **(gdb)** target remote eorgadd:8000 | **(lldb)** gdb-remote eorgadd:8000 |
| Attach to a remote gdb protocol server running on the local system, port 8000. | |
| **(gdb)** target remote localhost:8000 | **(lldb)** gdb-remote 8000 |
| Attach to a Darwin kernel in kdp mode on system "eorgadd". | |
| **(gdb)** kdp-reattach eorgadd | **(lldb)** kdp-remote eorgadd |
| Do a source level single step in the currently selected thread. | |
| **(gdb)** step **(gdb)** s | **(lldb)** thread step-in **(lldb)** step **(lldb)** s |
| Do a source level single step over in the currently selected thread. | |
| **(gdb)** next **(gdb)** n | **(lldb)** thread step-over **(lldb)** next **(lldb)** n |
| Do an instruction level single step in the currently selected thread. | |
| **(gdb)** stepi **(gdb)** si | **(lldb)** thread step-inst **(lldb)** si |
| Do an instruction level single step over in the currently selected thread. | |
| **(gdb)** nexti **(gdb)** ni | **(lldb)** thread step-inst-over **(lldb)** ni |
| Step out of the currently selected frame. | |
| **(gdb)** finish | **(lldb)** thread step-out **(lldb)** finish |
| Return immediately from the currently selected frame, with an optional return value. | |
| **(gdb)** return <RETURN EXPRESSION> | **(lldb)** thread return <RETURN EXPRESSION> |
| Backtrace and disassemble every time you stop. | |
|  | **(lldb)** target stop-hook add Enter your stop hook command(s). Type 'DONE' to end. > bt > disassemble --pc > DONE Stop hook #1 added. |
| Run until we hit line **12** or control leaves the current function. | |
| **(gdb)** until 12 | **(lldb)** thread until 12 |

BREAKPOINT COMMANDS

|  |  |
| --- | --- |
| **GDB** | **LLDB** |
| Set a breakpoint at all functions named **main**. | |
| **(gdb)** break main | **(lldb)** breakpoint set --name main **(lldb)** br s -n main **(lldb)** b main |
| Set a breakpoint in file **test.c** at line **12**. | |
| **(gdb)** break test.c:12 | **(lldb)** breakpoint set --file test.c --line 12 **(lldb)** br s -f test.c -l 12 **(lldb)** b test.c:12 |
| Set a breakpoint at all C++ methods whose basename is **main**. | |
| **(gdb)** break main *(Hope that there are no C functions named****main****)*. | **(lldb)** breakpoint set --method main **(lldb)** br s -M main |
| Set a breakpoint at and object C function: **-[NSString stringWithFormat:]**. | |
| **(gdb)** break -[NSString stringWithFormat:] | **(lldb)** breakpoint set --name "-[NSString stringWithFormat:]" **(lldb)** b -[NSString stringWithFormat:] |
| Set a breakpoint at all Objective C methods whose selector is **count**. | |
| **(gdb)** break count *(Hope that there are no C or C++ functions named****count****)*. | **(lldb)** breakpoint set --selector count **(lldb)** br s -S count |
| Set a breakpoint by regular expression on function name. | |
| **(gdb)** rbreak regular-expression | **(lldb)** breakpoint set --func-regex regular-expression **(lldb)** br s -r regular-expression |
| Ensure that breakpoints by file and line work for #included .c/.cpp/.m files. | |
| **(gdb)** b foo.c:12 | **(lldb)** settings set target.inline-breakpoint-strategy always **(lldb)** br s -f foo.c -l 12 |
| Set a breakpoint by regular expression on source file contents. | |
| **(gdb)** shell grep -e -n pattern source-file **(gdb)** break source-file:CopyLineNumbers | **(lldb)** breakpoint set --source-pattern regular-expression --file SourceFile **(lldb)** br s -p regular-expression -f file |
| Set a conditional breakpoint | |
| **(gdb)** break foo if strcmp(y,"hello") == 0 | **(lldb)** breakpoint set --name foo --condition '(int)strcmp(y,"hello") == 0' **(lldb)** br s -n foo -c '(int)strcmp(y,"hello") == 0' |
| List all breakpoints. | |
| **(gdb)** info break | **(lldb)** breakpoint list **(lldb)** br l |
| Delete a breakpoint. | |
| **(gdb)** delete 1 | **(lldb)** breakpoint delete 1 **(lldb)** br del 1 |

WATCHPOINT COMMANDS

|  |  |
| --- | --- |
| **GDB** | **LLDB** |
| Set a watchpoint on a variable when it is written to. | |
| **(gdb)** watch global\_var | **(lldb)** watchpoint set variable global\_var **(lldb)** wa s v global\_var |
| Set a watchpoint on a memory location when it is written into. The size of the region to watch for defaults to the pointer size if no '-x byte\_size' is specified. This command takes raw input, evaluated as an expression returning an unsigned integer pointing to the start of the region, after the '--' option terminator. | |
| **(gdb)** watch -location g\_char\_ptr | **(lldb)** watchpoint set expression -- my\_ptr **(lldb)** wa s e -- my\_ptr |
| Set a condition on a watchpoint. | |
|  | **(lldb)** watch set var global **(lldb)** watchpoint modify -c '(global==5)' **(lldb)** c ... **(lldb)** bt \* thread #1: tid = 0x1c03, 0x0000000100000ef5 a.out`modify + 21 at main.cpp:16, stop reason = watchpoint 1 frame #0: 0x0000000100000ef5 a.out`modify + 21 at main.cpp:16 frame #1: 0x0000000100000eac a.out`main + 108 at main.cpp:25 frame #2: 0x00007fff8ac9c7e1 libdyld.dylib`start + 1 **(lldb)** frame var global (int32\_t) global = 5 |
| List all watchpoints. | |
| **(gdb)** info break | **(lldb)** watchpoint list **(lldb)** watch l |
| Delete a watchpoint. | |
| **(gdb)** delete 1 | **(lldb)** watchpoint delete 1 **(lldb)** watch del 1 |

EXAMINING VARIABLES

|  |  |
| --- | --- |
| **GDB** | **LLDB** |
| Show the arguments and local variables for the current frame. | |
| **(gdb)** info args and **(gdb)** info locals | **(lldb)** frame variable **(lldb)** fr v |
| Show the local variables for the current frame. | |
| **(gdb)** info locals | **(lldb)** frame variable --no-args **(lldb)** fr v -a |
| Show the contents of local variable "bar". | |
| **(gdb)** p bar | **(lldb)** frame variable bar  **(lldb)** fr v bar  **(lldb)** p bar |
| Show the contents of local variable "bar" formatted as hex. | |
| **(gdb)** p/x bar | **(lldb)** frame variable --format x bar  **(lldb)** fr v -f x bar |
| Show the contents of global variable "baz". | |
| **(gdb)** p baz | **(lldb)** target variable baz  **(lldb)** ta v baz |
| Show the global/static variables defined in the current source file. | |
| n/a | **(lldb)** target variable  **(lldb)** ta v |
| Display the variables "argc" and "argv" every time you stop. | |
| **(gdb)** display argc **(gdb)** display argv | **(lldb)** target stop-hook add --one-liner "frame variable argc argv" **(lldb)** ta st a -o "fr v argc argv" **(lldb)** display argc **(lldb)** display argv |
| Display the variables "argc" and "argv" only when you stop in the function named **main**. | |
|  | **(lldb)** target stop-hook add --name main --one-liner "frame variable argc argv" **(lldb)** ta st a -n main -o "fr v argc argv" |
| Display the variable "\*this" only when you stop in c class named **MyClass**. | |
|  | **(lldb)** target stop-hook add --classname MyClass --one-liner "frame variable \*this" **(lldb)** ta st a -c MyClass -o "fr v \*this" |

EVALUATING EXPRESSIONS

|  |  |
| --- | --- |
| **GDB** | **LLDB** |
| Evaluating a generalized expression in the current frame. | |
| **(gdb)** print (int) printf ("Print nine: %d.", 4 + 5) or if you don't want to see void returns:  **(gdb)** call (int) printf ("Print nine: %d.", 4 + 5) | **(lldb)** expr (int) printf ("Print nine: %d.", 4 + 5) or using the print alias: **(lldb)** print (int) printf ("Print nine: %d.", 4 + 5) |
| Creating and assigning a value to a convenience variable. | |
| **(gdb)** set $foo = 5 **(gdb)** set variable $foo = 5 or using the print command  **(gdb)** print $foo = 5 or using the call command  **(gdb)** call $foo = 5 and if you want to specify the type of the variable: **(gdb)** set $foo = (unsigned int) 5 | In lldb you evaluate a variable declaration expression as you would write it in C: **(lldb)** expr unsigned int $foo = 5 |
| Printing the ObjC "description" of an object. | |
| **(gdb)** po [SomeClass returnAnObject] | **(lldb)** expr -o -- [SomeClass returnAnObject] or using the po alias: **(lldb)** po [SomeClass returnAnObject] |
| Print the dynamic type of the result of an expression. | |
| **(gdb)** set print object 1 **(gdb)** p someCPPObjectPtrOrReference only works for C++ objects. | **(lldb)** expr -d 1 -- [SomeClass returnAnObject] **(lldb)** expr -d 1 -- someCPPObjectPtrOrReference or set dynamic type printing to be the default:**(lldb)** settings set target.prefer-dynamic run-target |
| Calling a function so you can stop at a breakpoint in the function. | |
| **(gdb)** set unwindonsignal 0 **(gdb)** p function\_with\_a\_breakpoint() | **(lldb)** expr -i 0 -- function\_with\_a\_breakpoint() |
| Calling a function that crashes, and stopping when the function crashes. | |
| **(gdb)** set unwindonsignal 0 **(gdb)** p function\_which\_crashes() | **(lldb)** expr -u 0 -- function\_which\_crashes() |

EXAMINING THREAD STATE

|  |  |
| --- | --- |
| **GDB** | **LLDB** |
| Show the stack backtrace for the current thread. | |
| **(gdb)** bt | **(lldb)** thread backtrace **(lldb)** bt |
| Show the stack backtraces for all threads. | |
| **(gdb)** thread apply all bt | **(lldb)** thread backtrace all **(lldb)** bt all |
| Backtrace the first five frames of the current thread. | |
| **(gdb)** bt 5 | **(lldb)** thread backtrace -c 5 **(lldb)** bt 5 (*lldb-169 and later*) **(lldb)** bt -c 5 (*lldb-168 and earlier*) |
| Select a different stack frame by index for the current thread. | |
| **(gdb)** frame 12 | **(lldb)** frame select 12 **(lldb)** fr s 12 **(lldb)** f 12 |
| List information about the currently selected frame in the current thread. | |
|  | **(lldb)** frame info |
| Select the stack frame that called the current stack frame. | |
| **(gdb)** up | **(lldb)** up **(lldb)** frame select --relative=1 |
| Select the stack frame that is called by the current stack frame. | |
| **(gdb)** down | **(lldb)** down **(lldb)** frame select --relative=-1 **(lldb)** fr s -r-1 |
| Select a different stack frame using a relative offset. | |
| **(gdb)** up 2 **(gdb)** down 3 | **(lldb)** frame select --relative 2 **(lldb)** fr s -r2  **(lldb)** frame select --relative -3 **(lldb)** fr s -r-3 |
| Show the general purpose registers for the current thread. | |
| **(gdb)** info registers | **(lldb)** register read |
| Write a new decimal value '123' to the current thread register 'rax'. | |
| **(gdb)** p $rax = 123 | **(lldb)** register write rax 123 |
| Skip 8 bytes ahead of the current program counter (instruction pointer). Note that we use backticks to evaluate an expression and insert the scalar result in LLDB. | |
| **(gdb)** jump \*$pc+8 | **(lldb)** register write pc `$pc+8` |
| Show the general purpose registers for the current thread formatted as **signed decimal**. LLDB tries to use the same format characters as **printf(3)** when possible. Type "help format" to see the full list of format specifiers. | |
|  | **(lldb)** register read --format i **(lldb)** re r -f i  *LLDB now supports the GDB shorthand format syntax but there can't be space after the command:* **(lldb)** register read/d |
| Show all registers in all register sets for the current thread. | |
| **(gdb)** info all-registers | **(lldb)** register read --all **(lldb)** re r -a |
| Show the values for the registers named "rax", "rsp" and "rbp" in the current thread. | |
| **(gdb)** info all-registers rax rsp rbp | **(lldb)** register read rax rsp rbp |
| Show the values for the register named "rax" in the current thread formatted as **binary**. | |
| **(gdb)** p/t $rax | **(lldb)** register read --format binary rax **(lldb)** re r -f b rax  *LLDB now supports the GDB shorthand format syntax but there can't be space after the command:* **(lldb)** register read/t rax **(lldb)** p/t $rax |
| Read memory from address 0xbffff3c0 and show 4 hex uint32\_t values. | |
| **(gdb)** x/4xw 0xbffff3c0 | **(lldb)** memory read --size 4 --format x --count 4 0xbffff3c0 **(lldb)** me r -s4 -fx -c4 0xbffff3c0 **(lldb)** x -s4 -fx -c4 0xbffff3c0  *LLDB now supports the GDB shorthand format syntax but there can't be space after the command:* **(lldb)** memory read/4xw 0xbffff3c0 **(lldb)** x/4xw 0xbffff3c0 **(lldb)** memory read --gdb-format 4xw 0xbffff3c0 |
| Read memory starting at the expression "argv[0]". | |
| **(gdb)** x argv[0] | **(lldb)** memory read `argv[0]` ***NOTE:****any command can inline a scalar expression result (as long as the target is stopped) using backticks around any expression:* **(lldb)** memory read --size `sizeof(int)` `argv[0]` |
| Read 512 bytes of memory from address 0xbffff3c0 and save results to a local file as **text**. | |
| **(gdb)** set logging on **(gdb)** set logging file /tmp/mem.txt **(gdb)** x/512bx 0xbffff3c0 **(gdb)** set logging off | **(lldb)** memory read --outfile /tmp/mem.txt --count 512 0xbffff3c0 **(lldb)** me r -o/tmp/mem.txt -c512 0xbffff3c0 **(lldb)** x/512bx -o/tmp/mem.txt 0xbffff3c0 |
| Save binary memory data starting at 0x1000 and ending at 0x2000 to a file. | |
| **(gdb)** dump memory /tmp/mem.bin 0x1000 0x2000 | **(lldb)** memory read --outfile /tmp/mem.bin --binary 0x1000 0x2000 **(lldb)** me r -o /tmp/mem.bin -b 0x1000 0x2000 |
| Get information about a specific heap allocation (available on Mac OS X only). | |
| **(gdb)** info malloc 0x10010d680 | **(lldb)** command script import lldb.macosx.heap **(lldb)** process launch --environment MallocStackLogging=1 -- [ARGS] **(lldb)** malloc\_info --stack-history 0x10010d680 |
| Get information about a specific heap allocation and cast the result to any dynamic type that can be deduced (available on Mac OS X only) | |
|  | **(lldb)** command script import lldb.macosx.heap **(lldb)** malloc\_info --type 0x10010d680 |
| Find all heap blocks that contain a pointer specified by an expression EXPR (available on Mac OS X only). | |
|  | **(lldb)** command script import lldb.macosx.heap **(lldb)** ptr\_refs EXPR |
| Find all heap blocks that contain a C string anywhere in the block (available on Mac OS X only). | |
|  | **(lldb)** command script import lldb.macosx.heap **(lldb)** cstr\_refs CSTRING |
| Disassemble the current function for the current frame. | |
| **(gdb)** disassemble | **(lldb)** disassemble --frame **(lldb)** di -f |
| Disassemble any functions named **main**. | |
| **(gdb)** disassemble main | **(lldb)** disassemble --name main **(lldb)** di -n main |
| Disassemble an address range. | |
| **(gdb)** disassemble 0x1eb8 0x1ec3 | **(lldb)** disassemble --start-address 0x1eb8 --end-address 0x1ec3 **(lldb)** di -s 0x1eb8 -e 0x1ec3 |
| Disassemble 20 instructions from a given address. | |
| **(gdb)** x/20i 0x1eb8 | **(lldb)** disassemble --start-address 0x1eb8 --count 20 **(lldb)** di -s 0x1eb8 -c 20 |
| Show mixed source and disassembly for the current function for the current frame. | |
| n/a | **(lldb)** disassemble --frame --mixed **(lldb)** di -f -m |
| Disassemble the current function for the current frame and show the opcode bytes. | |
| n/a | **(lldb)** disassemble --frame --bytes **(lldb)** di -f -b |
| Disassemble the current source line for the current frame. | |
| n/a | **(lldb)** disassemble --line **(lldb)** di -l |

EXECUTABLE AND SHARED LIBRARY QUERY COMMANDS

|  |  |
| --- | --- |
| **GDB** | **LLDB** |
| List the main executable and all dependent shared libraries. | |
| **(gdb)** info shared | **(lldb)** image list |
| Look up information for a raw address in the executable or any shared libraries. | |
| **(gdb)** info symbol 0x1ec4 | **(lldb)** image lookup --address 0x1ec4 **(lldb)** im loo -a 0x1ec4 |
| Look up functions matching a regular expression in a binary. | |
| **(gdb)** info function <FUNC\_REGEX> | This one finds debug symbols: **(lldb)** image lookup -r -n <FUNC\_REGEX>  This one finds non-debug symbols: **(lldb)** image lookup -r -s <FUNC\_REGEX>  Provide a list of binaries as arguments to limit the search. |
| Find full source line information. | |
| **(gdb)** info line 0x1ec4 | This one is a bit messy at present. Do:  **(lldb)** image lookup -v --address 0x1ec4  and look for the LineEntry line, which will have the full source path and line range information. |
| Look up information for an address in **a.out** only. | |
|  | **(lldb)** image lookup --address 0x1ec4 a.out **(lldb)** im loo -a 0x1ec4 a.out |
| Look up information for for a type Point by name. | |
| **(gdb)** ptype Point | **(lldb)** image lookup --type Point **(lldb)** im loo -t Point |
| Dump all sections from the main executable and any shared libraries. | |
| **(gdb)** maintenance info sections | **(lldb)** image dump sections |
| Dump all sections in the **a.out** module. | |
|  | **(lldb)** image dump sections a.out |
| Dump all symbols from the main executable and any shared libraries. | |
|  | **(lldb)** image dump symtab |
| Dump all symbols in **a.out** and **liba.so**. | |
|  | **(lldb)** image dump symtab a.out liba.so |

MISCELLANEOUS

|  |  |
| --- | --- |
| **GDB** | **LLDB** |
| Echo text to the screen. | |
| **(gdb)** echo Here is some text\n | **(lldb)** script print "Here is some text" |
| Remap source file pathnames for the debug session. If your source files are no longer located in the same location as when the program was built --- maybe the program was built on a different computer --- you need to tell the debugger how to find the sources at their local file path instead of the build system's file path. | |
| **(gdb)** set pathname-substitutions /buildbot/path /my/path | **(lldb)** settings set target.source-map /buildbot/path /my/path |
| Supply a catchall directory to search for source files in. | |
| **(gdb)** directory /my/path | (*No equivalent command - use the source-map instead.*) |

## iOS app安全编码

https://developer.apple.com/library/content/documentation/Security/Conceptual/SecureCodingGuide

常见场景

缓冲区溢出

计算内存对象的偏移和大小时使用unsigned值，并检查整数溢出

避免底层c函数strcat, strcpy, strncpy, sprintf, vsprintf, gets等，使用strlcat,strlcpy,snprintf,vsnprintf,fgets

使用malloc,new等函数避免溢出

在NSString转换为c字符串时避免溢出

使用CFString API代替c字符串

使用c++ string类时避免溢出

避免硬编码缓冲区大小

避免整数溢出

开启MH\_PIE MH\_NO\_HEAP\_EXECUTION (mprotect)

输入数据未验证

输入长度检查

格式化字符串攻击，包括如下函数：

标准c：printf sscanf syslog vsyslog等

Carbon：AEBuildDesc vAEBuildDesc AEBuildParameters vAEBuildParameters

Core Foundation：CFStringCreateWithFormat CFStringCreateWithFormatAndArguments CFStringAppendFormat CFStringAppendFormatAndArguments

Cocoa：

NSLog

NSString的stringWithFormat: initWithFormat: stringByAppendingFormat等

NSMutableString的appendFormat:

NSAlert的informativeTextWithFormat: alertWithMessageText

NSPredicate的predicateWithFormat:

NSException的format

NSRunAlertPanel等

如果注册了URL scheme(Info.plist CFBundleURLTypes)，需要在收到命令时验证UIApplication:handleOpenURL/openURL，后者可以验证发起者

注入攻击

持久化数据验证：使用initWIthCoder: decodeObjectForKey: 需要验证文件系统数据，否则使用可用于不可信域的decodeObjectOfClass:forKey:，并实现NSSecureCoding协议；某些对象(NSImage)会在反持久化时(NSKeyedUnarchiverDelegate unarchiveer:didDecodeObject:)返回不同的对象，或在收到消息awakeAfterUsingCoder:时，会自己注册一个名字取代app使用的image，攻击者可以利用这点插入恶意代码；nib文件要存放在可信位置

进程间通信

使用Mach消息通信时，不要传递自身进程的Mach task port，mach消息只应用于和client通信，iOS不支持

使用Distributed Objects通信时，NSDistantObject不遵守NSSecureCoding协议因此没有内建认证机制，因此跨进程通信只应该用connectionWithReceivePort:sendPort方法在一个进程的多个线程中使用

使用XPC服务通信是最安全的方式，正确的实现NSSecureCoding协议，

使用共享内存通信，shmget shmat shmat shmctl，不可共享非完整页的块内存，不可共享栈空间；某些线程竞争条件下内存映射文件可被其他文件替换；使用命名共享内存不安全，设置为IPC\_PRIVATE

使用信号通信，signal handler中只有部分库函数和系统调用时安全的，因此不要做过多工作(可以将这些工作放在main loop)；攻击者可以向你的app发送信号从而在任意时刻执行，如果存在线程竞争可能产生未定义行为(CVE-2004-0794)。在handler中使用非阻塞函数，否则可导致拒绝服务攻击

文件目录

在正确目录创建临时文件，获取临时目录：cocoa api层使用NSTemporaryDirectory， POSIX层使用confstr \_CS\_DARWIN\_USER\_TEMP\_DIR；创建文件：使用open mkstep；避免使用NSString/NSData的writeToFile:atomically

避免在/tmp, /var/tmp, /var/run, /var/db, ~/Library/Caches/TemporaryItems操作临时文件

检查文件是否为hard link或symbolic link

重要数据明文存储，NSData/NSFileManager提供四个层级的文件系统保护

key chain安全

认证、授权、加密存在漏洞

不在本地存储、验证、修改密码，验证后清除内存中的密码数据

使用可信随机数生成器，iOS中使用Randomization Services接口，不要使用rand函数

使用安全的网络通信协议TLS/SSL而不要自定义协议

使用安全的加密算法而不要自定义加密算法

安装和加载

不要直接下载和执行代码

不要在/Library/StartupItems和/System/Library/Extensions中加载组件

不要自定义安装脚本

从安全位置加载插件和库

使用外部工具和库

不使用shell脚本

不使用popen system，若使用要确保命令中不含特殊符号从而执行恶意命令

使用execlp, execvp,popen,system等使用PATH环境变量的函数时，使用绝对路径

验证参数

已经被攻击利用过的：

dyld:LD\_LIBRARY\_PATH DYLD\_LIBRARY\_PATH

libc:MallocLogFile

CoreFoundation:CF\_CHARSET\_PATH

未开启arc进行手动引用计数，可能导致objc类结构覆盖

Use After free引用失效内存

Double Free 双重释放

Stack Overflow 缓冲区溢出

Format bug 格式化漏洞 %n %@

## libimobiledevice工具集

市面上常见的iOS助手都是基于这套工具开发

git目录：<http://github.com/libimobiledevice>

包括工程：

libimobiledevice ios交互工具，包括：

idevice\_id 枚举ios设备 => adb devices

idevicedebug 指定bundle id拉起并调试进程，可以执行环境变量

ideviceinfo 设备信息

idevicescreenshot 截屏

idevicebackup 备份/还原设备

idevicedebugserverproxy

idevicename

idevicesyslog

idevicediagnostics 设备关机重启

idevicedate

ideviceimagemounter 映射dmg设备分区

libplist 依赖库，解析plist文件

idevicerestore 还原/升级固件

sbmanager 管理ios桌面图标

libirecovery

usbmuxd sub转socket

libusbmuxd

ideviceinstaller 管理app

libideviceactivation

ifuse 映射iphone到本地

编译过程：

1安装依赖库

brew/port install openssl automake autoconf libtool pkg-config libzip

1编译libplist

./autogen.sh

make install

2编译libusbmuxd

./autogen.sh

make install

3编译libimobiledevice

PKG\_CONFIG\_PATH=/path/to/libusbmuxd ./autogen.sh

make install

其他工程类似

ideviceinstaller需要1.0.0版本的libimobiledevice编译

## 常用cycript函数

来自https://github.com/Tyilo/cycript-utils/blob/master/utils.cy

(function(utils) {

// Expose C constants to cycript's global scope

var shouldExposeConsts = true;

// Expose functions defined here (in utils) to cycript's global scope

var shouldExposeFuncs = true

// Various constants

utils.constants = {

// <sys/sysctl.h>

CTL\_MAXNAME: 12,

// <mach/machine.h>

// These are builtin in cycript, but they are broken

CPU\_TYPE\_X86: 7,

CPU\_TYPE\_ARM: 12,

};

var c = utils.constants;

c.CPU\_TYPE\_X86\_64 = c.CPU\_TYPE\_X86 | c.CPU\_ARCH\_ABI64;

c.CPU\_TYPE\_ARM64 = c.CPU\_TYPE\_ARM | c.CPU\_ARCH\_ABI64;

// Capital D

$cysDl\_info = (struct dl\_info);

extern "C" mach\_port\_t mach\_task\_self();

extern "C" size\_t malloc\_size(const void \*);

extern "C" struct dyld\_all\_image\_infos \*\_dyld\_get\_all\_image\_infos();

extern "C" int sysctl(int \*, u\_int, void \*, size\_t \*, void \*, size\_t);

extern "C" int sysctlnametomib(const char \*, int \*, size\_t \*);

var log = x => NSLog(@"%s", x.toString());

/\*

Aligns the pointer downwards, aligment must be a power of 2

Useful for mprotect

Example:

cy# utils.align(0x100044, 0x1000).toString(16)

"100000"

\*/

utils.align = function(ptr, alignment) {

var high = Math.floor(ptr / Math.pow(2, 32));

var low = ptr | 0;

low = (low & ~(alignment - 1));

if(low < 0) {

low = Math.pow(2, 32) + low;

}

return low + high \* Math.pow(2, 32);

};

/\*

Sets the protection of the memory starting at addr

with length len, to prot

Example:

cy# var foo = new int;

&0

cy# utils.mprotect(foo.valueOf(),

foo.type.size,

utils.constants.PROT\_READ)

cy# \*a = 1

\*\*\* \_assert(CYRecvAll(client, &size, sizeof(size))):../Console.cpp(142):Run

\*/

utils.mprotect = function(addr, len, prot) {

addr = utils.getPointer(addr);

var pagesize = getpagesize();

var aligned = utils.align(addr, pagesize);

var mprotect\_size = addr - aligned + len;

if(mprotect(aligned, mprotect\_size, prot)) {

throw "mprotect failed.";

}

};

/\*

Returns an array of all values associated with an object

Example:

cy# utils.getValues({a: 1, b: 2, c: 2})

[1,2,2]

\*/

utils.getValues = function(obj) {

return Object.keys(obj).map(o => obj[o]);

};

/\*

Runs an external program with arguments

Returns the program's stdout

Example:

cy# utils.getOutputFromTask("/bin/date", ["+%s"])

@"1419918861\n"

\*/

utils.getOutputFromTask = function(path, args) {

var task = [new NSTask init];

task.launchPath = path;

task.arguments = args;

var pipe = [NSPipe pipe];

task.standardOutput = pipe;

[task launch];

[task waitUntilExit];

var data = [[pipe fileHandleForReading] readDataToEndOfFile];

return [new NSString initWithData:data encoding:c.NSUTF8StringEncoding];

};

utils.hex = function(ptr) {

return '0x' + utils.getPointer(ptr).toString(16);

};

/\*

Logs a specific message sent to an instance of a class like logify.pl in theos

Requires Cydia Substrate (com.saurik.substrate.MS) and NSLog (org.cycript.NSLog) modules

Returns the old message returned by MS.hookMessage (Note: this is not just the old message!)

FIXME: For certain combinations of arguments, the process will crash

Example:

cy# var oldm = utils.logify(object\_getClass(NSNumber), @selector(numberWithDouble:))

...

cy# var n = [NSNumber numberWithDouble:1.5]

2014-07-28 02:26:39.805 cycript[71213:507] +[<NSNumber: 0x10032d0c4> numberWithDouble:1.5]

2014-07-28 02:26:39.806 cycript[71213:507] = 1.5

@1.5

\*/

utils.logify = function(cls, sel) {

@import com.saurik.substrate.MS;

var selFormat = sel.toString().replace(/:/g, ":%s ").trim();

var logFormat = @"%s[<%@: %p> " + selFormat + "]";

var oldm = {};

MS.hookMessage(cls, sel, function() {

var args = [logFormat, class\_isMetaClass(cls)? "+": "-", cls, (typedef void \*)(this)];

for (arg of arguments) {

args.push(arg.toString());

}

NSLog.apply(null, args);

var r = oldm->apply(this, arguments);

if(r !== undefined) {

NSLog(@" = %s", r.toString());

}

return r;

}, oldm);

return oldm;

};

/\*

Similar to utils.logify, but for functions instead of messages

Example:

cy# logifyFunc("fopen", 2);

...

cy# apply("fopen", ["/etc/passwd", "r"]);

2015-01-14 07:01:08.009 cycript[55326:2042054] fopen(0x10040d4cc, 0x10040d55c)

2015-01-14 07:01:08.010 cycript[55326:2042054] = 0x7fff754fc070

0x7fff754fc070

\*/

utils.logifyFunc = function(nameOrPointer, argCount) {

@import com.saurik.substrate.MS;

var name = "" + nameOrPointer;

var ptr = nameOrPointer;

if(typeof nameOrPointer === "string") {

ptr = dlsym(RTLD\_DEFAULT, nameOrPointer);

if(!ptr) {

throw "Couldn't find function with name using dlsym!";

}

}

var oldf = {};

var f = function() {

var logFormat = @"%s(";

for(var i = 0; i < arguments.length; i++) {

logFormat += (i > 0? ", ": "") + "%s";

}

logFormat += ")";

var args = [];

for (var arg of arguments) {

args.push(utils.hex(arg));

}

NSLog.apply(null, [logFormat, name].concat(args));

var r = (\*oldf).apply(null, arguments);

if(r !== undefined) {

NSLog(@" = %s", utils.hex(r));

}

return r;

};

var voidPtr = (typedef void \*);

var argTypes = [];

for(var i = 0; i < argCount; i++) {

argTypes.push(voidPtr);

}

var fType = voidPtr.functionWith.apply(voidPtr, argTypes);

MS.hookFunction(fType(ptr), fType(f), oldf);

return oldf;

};

/\*

Calls a C function by providing its name and arguments

Doesn't support structs

Return value is always a void pointer

Example:

cy# utils.apply("printf", ["%s %.3s, %d -> %c, float: %f\n", "foo", "barrrr", 97, 97, 1.5])

foo bar, 97 -> a, float: 1.500000

0x22

\*/

utils.apply = function(fun, args) {

if(!(args instanceof Array)) {

throw "utils.apply: Args needs to be an array!";

}

var argc = args.length;

var voidPtr = (typedef void \*);

var argTypes = [];

for(var i = 0; i < argc; i++) {

var argType = voidPtr;

var arg = args[i];

if(typeof arg === "string") {

argType = (typedef char \*);

}

if(typeof arg === "number" && arg % 1 !== 0) {

argType = (typedef double);

}

argTypes.push(argType);

}

var type = voidPtr.functionWith.apply(voidPtr, argTypes);

if(typeof fun === "string") {

fun = dlsym(RTLD\_DEFAULT, fun);

}

if(!fun) {

throw "utils.apply: Function not found!";

}

return type(fun).apply(null, args);

};

/\*

Converts a string (char \*) to a void pointer (void \*)

You can't cast to strings to void pointers and vice versa in cycript. Blame saurik.

Example:

cy# var voidPtr = utils.str2voidPtr("foobar")

0x100331590

cy# utils.voidPtr2str(voidPtr)

"foobar"

\*/

utils.str2voidPtr = function(str) {

return (typedef void \*)(strdup(str));

};

/\*

The inverse function of str2voidPtr

\*/

utils.voidPtr2str = function(voidPtr) {

return (typedef char \*)(p).toString()

};

/\*

Converts a double into a void pointer

This can be used to view the binary representation of a floating point number

Example:

cy# var n = utils.double2voidPtr(-1.5)

0xbff8000000000000

cy# utils.voidPtr2double(n)

-1.5

\*/

utils.double2voidPtr = function(n) {

var doublePtr = new double;

\*doublePtr = n;

var voidPtrPtr = (typedef void \*\*)(doublePtr);

return \*voidPtrPtr;

};

/\*

The inverse function of double2voidPtr

\*/

utils.voidPtr2double = function(voidPtr) {

var voidPtrPtr = new (typedef void \*\*);

\*voidPtrPtr = voidPtr;

var doublePtr = (typedef double \*)(voidPtrPtr);

return \*doublePtr;

};

/\*

Determines in a safe way if a memory location is readable

Example:

cy# utils.isMemoryReadable(0)

false

cy# utils.isMemoryReadable(0x1337)

false

cy# utils.isMemoryReadable(NSObject)

true

cy# var a = malloc(100); utils.isMemoryReadable(a)

true

\*/

utils.isMemoryReadable = function(ptr) {

if(typeof ptr === "string") {

return true;

}

var fds = new (typedef int[2]);

pipe(fds);

var result = write(fds[1], ptr, 1) == 1;

close(fds[0]);

close(fds[1]);

return result;

};

/\*

Returns the pointer to an object as a number

Returns null if the input isn't a number or doesn't represent a memory location

Example:

cy# utils.getPointer(0)

0

cy# utils.getPointer(1234)

1234

cy# utils.getPointer(NSObject)

140735254495472

cy# utils.getPointer([])

null

\*/

utils.getPointer = function(obj) {

if(obj === 0 || obj === null) {

return 0;

}

var p = (typedef void \*)(obj);

if(p === null) {

return null;

}

return p.valueOf();

};

/\*

Determines if two object has the same pointer value

Example:

cy# var ptr = utils.getPointer(NSObject)

140735254495472

cy# utils.pointerCompare(ptr, NSObject)

true

cy# utils.pointerCompare(ptr, NSString)

false

\*/

utils.pointerCompare = function(o1, o2) {

if(o1 === o2) {

return true;

}

return utils.getPointer(o1) === utils.getPointer(o2);

};

/\*

Determines in a safe way if the memory location is a registered Objective-C class or metaclass

Example:

cy# utils.isClass(0x1337)

false

cy# utils.isClass(NSObject)

true

cy# utils.isClass(object\_getClass(NSObject))

true

\*/

utils.isClass = function(obj) {

var ptr = utils.getPointer(obj);

if(!ptr) {

return false;

}

var classes = utils.getValues(ObjectiveC.classes);

for(var i = 0; i < classes.length; i++) {

var c = classes[i];

if(utils.pointerCompare(ptr, c)) {

return true;

}

var metaclass = object\_getClass(c);

if(utils.pointerCompare(ptr, metaclass)) {

return true;

}

}

return false;

};

/\*

Determines in a safe way if the memory location contains an Objective-C object

Example:

cy# utils.isObject(0x1337)

false

cy# utils.isObject(NSObject)

true

cy# utils.isObject(object\_getClass(NSObject))

true

cy# utils.isObject([new NSObject init])

true

cy# var a = malloc(100); utils.isObject(a)

false

cy# \*@encode(void \*\*)(a) = NSObject; utils.isObject(a)

true

\*/

utils.isObject = function(obj) {

function safe\_objc\_isa\_ptr(ptr) {

if(!utils.isMemoryReadable(ptr)) {

return false;

}

var isa = utils.getPointer(\*@encode(void \*\*)(ptr));

// See http://www.sealiesoftware.com/blog/archive/2013/09/24/objc\_explain\_Non-pointer\_isa.html

var objc\_debug\_isa\_class\_mask = 0x00000001fffffffa;

isa = (isa & 1)? (isa & objc\_debug\_isa\_class\_mask): isa;

if((isa & (@encode(void \*).size - 1)) != 0) {

return null;

} else {

return isa;

}

}

var ptr = utils.getPointer(obj);

if(!ptr) {

return false;

}

if(utils.isClass(ptr)) {

return true;

}

var c = safe\_objc\_isa\_ptr(ptr);

if(!utils.isClass(c)) {

return false;

}

var msize = malloc\_size(ptr);

var isize = class\_getInstanceSize(new Instance(c));

return msize >= isize;

};

/\*

Dumps all UI/NS Image instances to a temporary folder

Optionally takes a filter function to filter which images to dump

Example:

cy# utils.dumpImages()

"43 images written to /tmp/cycript-images-rdIbcB"

cy# utils.dumpImages(img => img.size.width == 16)

"5 images written to /tmp/cycript-images-8oso44"

\*/

utils.dumpImages = function(filter\_fun) {

var image\_class = ObjectiveC.classes["UIImage"] || ObjectiveC.classes["NSImage"];

var images = choose(image\_class);

if(filter\_fun) {

images = images.filter(filter\_fun);

}

if(images.length === 0) {

throw "utils.dumpImages: No images found!"

}

var template = utils.str2voidPtr("/tmp/cycript-images-XXXXXX");

mkdtemp(template);

var dir = utils.voidPtr2str(template);

for(var i of images) {

data = [i TIFFRepresentation];

[data writeToFile:dir + "/0x" + utils.getPointer(i).toString(16) + ".tiff" atomically:YES];

}

return images.length + " images written to " + dir;

}

var app\_class = ObjectiveC.classes["UIApplication"] || ObjectiveC.classes["NSApplication"];

var app = app\_class && [app\_class sharedApplication];

/\*

Uses a heuristic method to determine if the object's class is an Apple provided one

Example:

cy# @implementation TestClass : NSObject {} @end

#"TestClass"

cy# utils.is\_not\_standard\_class([new TestClass init])

true

cy# utils.is\_not\_standard\_class([new NSObject init])

false

\*/

utils.is\_not\_standard\_class = function(obj) {

var classname = [obj className];

while(classname[0] == "\_") {

classname = classname.substr(1);

}

return !([classname hasPrefix:"UI"] || [classname hasPrefix:"NS"]);

};

/\*

Internal function used utils.find\_subviews and utils.find\_subview\_controllers

\*/

function find\_subviews\_internal(view, predicate, transform) {

var arr = [];

var o = transform(view);

if(o && predicate(o)) {

arr.push(o);

}

return arr.concat.apply(arr, view.subviews.map(x => find\_subviews\_internal(x, predicate, transform)));

}

/\*

Recusirvely finds all subviews satisfying a predicate

By default returns all subviews from the app's keyWindow

Example:

cy# utils.find\_subviews().length

421

cy# utils.find\_subviews(utils.is\_not\_standard\_class).length

48

cy# utils.find\_subviews(x => true, choose(UINavigationItemView)[0]).length

2

\*/

utils.find\_subviews = function(predicate, view) {

predicate = predicate || (x => true);

view = view || app.keyWindow;

return find\_subviews\_internal(view, predicate, x => x);

};

/\*

Like utils.find\_subviews but for viewcontrollers instead of views

\*/

utils.find\_subview\_controllers = function(predicate, view) {

predicate = predicate || (x => true);

view = view || app.keyWindow;

return find\_subviews\_internal(view, predicate, x => x.viewDelegate || x.delegate);

};

/\*

Finds all classes with only one instance in the app's keyWindow's subviews

Also filter outs classes which are provided by Apple

Example:

cy# utils.find\_interesting\_view\_classes().length

9

\*/

utils.find\_interesting\_view\_classes = function() {

var views = utils.find\_subviews();

var classes = views.map(x => x.className.toString());

var interesting\_classes = classes.filter(x => classes.indexOf(x) === classes.lastIndexOf(x));

return interesting\_classes;

};

/\*

Like utils.find\_interesting\_view\_classes but for viewcontroller classes

\*/

utils.find\_interesting\_viewcontroller\_classes = function() {

var views = utils.find\_subview\_controllers();

var classes = views.map(x => x.className.toString());

var interesting\_classes = classes.filter(x => classes.indexOf(x) === classes.lastIndexOf(x));

return interesting\_classes;

};

/\*

Recursively returns the superviews of the view

Example:

cy# utils.view\_hierarchy(UIApp.keyWindow.subviews[0].subviews[0])

[#"<UIWindow: ...>",#"<UILayoutContainerView: ...>",#"<UINavigationTransitionView: ...>"]

\*/

utils.view\_hierarchy = function(view) {

var arr = [];

do {

arr.unshift(view);

} while(view = view.superview);

return arr;

};

/\*

Determines if an UI/NS View is on the screen

Example:

cy# utils.is\_on\_screen(UIApp.keyWindow)

true

cy# utils.is\_on\_screen([new UIView init])

false

\*/

utils.is\_on\_screen = function(view) {

var hierarchy = utils.view\_hierarchy(view);

return !![hierarchy[0] isEqual:app.keyWindow];

};

/\*

Returns the common superview of the two views

and two integers with the distance between the views and the superview

Example:

cy# rootview = [new UIView init]

...

cy# subview1 = [new UIView init]; [rootview addSubview:subview1];

cy# subview2 = [new UIView init]; [rootview addSubview:subview2];

cy# subview22 = [new UIView init]; [subview2 addSubview:subview22];

cy# utils.view\_relation(subview1, subview22)

[#"<UIView: ...>",1,2]

cy# utils.view\_relation(subview1, subview2)

[#"<UIView: ...>",1,1]

cy# utils.view\_relation(rootview, [new UIView init])

null

\*/

utils.view\_relation = function(view1, view2) {

var view\_hierarchy1 = utils.view\_hierarchy(view1);

var view\_hierarchy2 = utils.view\_hierarchy(view2);

var i;

for(i = 0; [view\_hierarchy1[i] isEqual:view\_hierarchy2[i]]; i++) {

}

if(i === 0) {

return null;

}

return [view\_hierarchy1[i - 1], view\_hierarchy1.length - i, view\_hierarchy2.length - i];

};

/\*

Returns an array with two integers specifying

the CPU\_TYPE and CPU\_SUB\_TYPE of the current running process

If the executable is fat, this returns the value for the active slice

Example:

cy# utils.getCpuType() // x86\_64

[16777223,0]

cy# utils.getCpuType() // i368

[7,0]

cy# utils.getCpuType() // arm 32 bit

[12,0]

\*/

utils.getCpuType = function() {

var mibLen = c.CTL\_MAXNAME;

var mib = new (typedef int[mibLen]);

var mibLenPtr = new uint64\_t(mibLen);

var err = sysctlnametomib("sysctl.proc\_cputype", mib, mibLenPtr);

if(err !== 0) {

throw "utils.getCpuType: Error calling sysctlnametomib!";

}

mibLen = \*mibLenPtr;

(\*mib)[mibLen] = getpid();

mibLen++;

var current\_arch = (typedef struct {cpu\_type\_t type; cpu\_subtype\_t subtype;});

var archType = new current\_arch;

var archTypeSizePtr = new uint64\_t(current\_arch.size);

err = sysctl(mib, mibLen, archType, archTypeSizePtr, 0, 0);

if(err !== 0) {

throw "utils.getCpuType: Error calling sysctl!";

}

return [archType->type, archType->subtype];

};

/\*

Pads a hex number with zeros so it represents a certain number of bytes

Example:

cy# utils.hexpad(1, 4)

"00000001"

cy# utils.hexpad(0xffffff, 4)

"00ffffff"

\*/

utils.hexpad = function(num, bytes) {

if(typeof num === "string") {

num = Number(num);

}

var hex = num.toString(16);

var padded = Array(bytes \* 2 + 1).join('0') + hex;

return padded.slice(-Math.max(2 \* bytes, hex.length));

};

/\*

Returns a string containing the address and path to every loaded image in the process

Example:

cy# ?expand

expand == true

cy# utils.get\_dyld\_info()

"

0x0000000100000000: /Users/Tyilo/bin/nsrunlooper

0x00007fff89ff4000: /usr/lib/libobjc.A.dylib

0x00007fff9325f000: /System/Library/Frameworks/Foundation.framework/Versions/C/Foundation

0x00007fff9324f000: /usr/lib/libSystem.B.dylib

0x00007fff8cec6000: /System/Library/Frameworks/CoreFoundation.framework/Versions/A/CoreFoundation

..."

\*/

utils.get\_dyld\_info = function() {

var all\_image\_infos = \_dyld\_get\_all\_image\_infos();

var image\_count = all\_image\_infos->infoArrayCount;

var info\_array = all\_image\_infos->infoArray;

var log = "";

for(var i = 0; i < image\_count; i++) {

var info = info\_array[i];

var base = info.imageLoadAddress.valueOf();

log += "\n0x" + utils.hexpad(base, @encode(void \*).size) + ": " + info.imageFilePath;

}

return log;

};

/\*

Returns a string of hexpairs representing the memory

starting at addr with length len

Example:

cy# var foo = new int;

cy# \*foo = 0x12345678

305419896

cy# utils.gethex(foo, 4)

"78563412"

\*/

utils.gethex = function(addr, len) {

addr = utils.getPointer(addr);

var res = "";

var p = (typedef uint8\_t \*)(addr);

for(var i = 0; i < len; i++) {

res += utils.hexpad(p[i], 1);

}

return res;

};

/\*

Returns a hexdump of a memory range with similar format as xxd

Example:

cy# var foo = new int;

cy# \*foo = 0x12345678

305419896

cy# ?expand

expand == true

cy# utils.hexdump(foo, 4)

"

01005015a0: 78 56 34 12 xV4.

"

\*/

utils.hexdump = function(addr, len, bytes\_per\_line) {

addr = utils.getPointer(addr);

if(!len) {

len = 0x100;

}

if(!bytes\_per\_line) {

bytes\_per\_line = 0x10;

}

function isprint(c) {

return 0x20 <= c && c <= 0x7E;

}

var p = (typedef uint8\_t \*)(addr);

var res = "\n";

var addr\_len = Math.ceil((addr + len - 1).toString(16).length / 2);

for(var i = 0; i < len; i += bytes\_per\_line) {

var cols = [utils.hexpad(addr + i, addr\_len) + ":", "", ""];

for(var j = i; j < i + bytes\_per\_line && j < len; j++) {

var n = p[j];

cols[1] += utils.hexpad(n, 1) + " ";

cols[2] += isprint(n)? String.fromCharCode(n): ".";

}

res += cols.join("\t") + "\n";

}

return res;

};

function getRasm2Args(addr) {

addr = utils.getPointer(addr);

var cpuType = utils.getCpuType();

var bits = "32";

if(cpuType[0] & CPU\_ARCH\_ABI64) {

bits = "64";

}

var arch;

if(cpuType[0] & c.CPU\_TYPE\_X86) {

arch = "x86";

} else if(cpuType[0] & c.CPU\_TYPE\_ARM) {

arch = "arm";

} else {

throw "Unknown arch for cpu: " + cpuType.join(", ");

}

var args = ["-a", arch, "-b", bits, "-o", addr.toString()];

return args;

}

function getRasm2Output(args) {

var rasm2\_path = utils.getOutputFromTask("/bin/bash", ["-c", "which rasm2"]).trim();

if(!rasm2\_path) {

throw "rasm2 command not found in /bin/bash's $PATH";

}

return utils.getOutputFromTask(rasm2\_path, args);

}

/\*

Disassembles a memory range using rasm2

Example:

cy# ?expand

expand == true

cy# var method = class\_getInstanceMethod(NSNumber,

@selector(intValue));

...

cy# var imp = method\_getImplementation(method);

...

cy# utils.disasm(imp, 10)

"

0x7fff83363b8c 1 55 push rbp

0x7fff83363b8d 3 4889e5 mov rbp, rsp

0x7fff83363b90 2 4157 push r15

0x7fff83363b92 2 4156 push r14

0x7fff83363b94 2 4155 push r13

"

\*/

utils.disasm = function(addr, len) {

addr = utils.getPointer(addr);

if(!len) {

len = 0x40;

}

var args = getRasm2Args(addr);

var hex = utils.gethex(addr, len);

args.push("-D", hex);

return "\n" + getRasm2Output(args);

};

/\*

Assembles some instructions to a memory address using rasm2

Example:

cy# var n = [NSNumber numberWithLongLong:10]

@10

cy# [n intValue]

10

cy# var method = class\_getInstanceMethod([n class], @selector(longLongValue));

...

cy# var imp = method\_getImplementation(method);

...

cy# utils.asm(imp, 'mov eax, 42; ret;')

6

cy# [n longLongValue]

42

\*/

utils.asm = function(addr, ins) {

addr = utils.getPointer(addr);

var args = getRasm2Args(addr);

args.push("--", ins);

var output = getRasm2Output(args).trim();

if(!output) {

throw "Couldn't assemble instructions with rasm2.";

}

utils.mprotect(addr, output.length / 2, PROT\_READ | PROT\_WRITE | PROT\_EXEC);

var p = @encode(uint8\_t \*)(addr);

for(var i = 0; i < output.length; i += 2) {

p[i / 2] = parseInt(output[i] + output[i + 1], 16);

}

return output.length / 2;

};

if(shouldExposeConsts) {

for(var k in utils.constants) {

Cycript.all[k] = utils.constants[k];

}

}

if(shouldExposeFuncs) {

for(var k in utils) {

if(utils.hasOwnProperty(k)) {

var f = utils[k];

if(typeof f === 'function') {

Cycript.all[k] = f;

}

}

}

}

})(exports);

来自https://github.com/limneos/weak\_classdump/blob/master/weak\_classdump.cy

NSLog\_ = dlsym(RTLD\_DEFAULT, "NSLog");

WCDLog = function() { var types = 'v', args = [], count = arguments.length; for (var i = 0; i != count; ++i) { types += '@'; args.push(arguments[i]); } new Functor(NSLog\_, types).apply(null, args); }

\_method\_copyReturnType=new Functor(dlsym(RTLD\_DEFAULT,"method\_copyReturnType"),"\*^{objc\_method=}");

\_method\_copyArgumentType=new Functor(dlsym(RTLD\_DEFAULT,"method\_copyArgumentType"),"\*^{objc\_method=}I");

\_\_sysctlbyname=new Functor(dlsym(-2,"sysctlbyname"),"v\*^?^i^?i");

function is64Bit(){

size=new int;

\_\_sysctlbyname("hw.cpu64bit\_capable",NULL,size,NULL,0);

is64Bit=new BOOL;

\_\_sysctlbyname("hw.cpu64bit\_capable",is64Bit,size,NULL,0);

return \*is64Bit;

}

NSNotFound=is64Bit() ? 9.2233720368547758e+18 : 2147483647;

function commonTypes(type){

isPointer=NO;

if ([type containsSubstring:@"^"]){

isPointer=YES;

type=[type stringByReplacingOccurrencesOfString:@"^" withString:""];

}

switch (type.toString()){

case "d": type = "double"; break;

case "i": type = "int"; break;

case "f": type = "float"; break;

case "c": type = "BOOL"; break;

case "s": type = "short"; break;

case "I": type = "unsigned"; break;

case "l": type = "long"; break;

case "q": type = "long long"; break;

case "L": type = "unsigned long"; break;

case "C": type = "unsigned char"; break;

case "S": type = "unsigned short"; break;

case "Q": type = "unsigned long long"; break;

case "B": type = "BOOL"; break; //replacing \_Bool , change it here if you wish

case "v": type = "void"; break;

case "\*": type = "char\*"; break;

case ":": type = "SEL"; break;

case "#": type = "Class"; break;

case "@": type = "id"; break;

case "@?": type = "id"; break;

case "Vv": type = "void"; break;

case "rv": type = "const void\*"; break;

default: type = type;

}

return isPointer ? type.toString()+"\*" : type.toString();

}

function getProtocolLines(protocol){

var protocolsMethodsString="";

currentProtocol=protocol;

protocolName=protocol\_getName(currentProtocol);

protocolsMethodsString=protocolsMethodsString.toString()+"\n@protocol "+protocolName.toString()+"\n";

protPropertiesString="";

protPropertiesCount=new int;

protPropertyList=protocol\_copyPropertyList(currentProtocol,protPropertiesCount);

for (xi=0; xi<\*protPropertiesCount; xi++){

propname=property\_getName(protPropertyList[xi]);

attrs=property\_getAttributes(protPropertyList[xi]);

newString=propertyLineGenerator(attrs,propname).toString();

if (![protPropertiesString containsSubstring:newString]){

protPropertiesString=protPropertiesString.toString()+newString.toString();

}

}

protocolsMethodsString=protocolsMethodsString.toString()+protPropertiesString;

free(protPropertyList);

for (acase=0; acase<5; acase++){

protocolMethodsCount=new int;

isRequiredMethod=acase<2 ? NO : YES;

isInstanceMethod=(acase==0 || acase==2) ? NO : YES;

protMeths=protocol\_copyMethodDescriptionList(currentProtocol, isRequiredMethod, isInstanceMethod, protocolMethodsCount);

for (gg=0; gg<\*protocolMethodsCount; gg++){

if (acase<2 && ![[NSString stringWithString:protocolsMethodsString] containsSubstring:@"@optional"]){

protocolsMethodsString=protocolsMethodsString.toString()+"@optional\n";

}

if (acase>1 && ![[NSString stringWithString:protocolsMethodsString] containsSubstring:@"@required"]){

protocolsMethodsString=protocolsMethodsString.toString()+"@required\n";

}

startSign=isInstanceMethod==NO ? "+" : "-";

protSelector=protMeths[gg][0].toString();

protTypes=protMeths[gg][1];

methodSign=[NSMethodSignature signatureWithObjCTypes:protTypes];

returnType=constructTypeAndName(commonTypes([methodSign methodReturnType].toString()),"",0);

finString="";

if ([methodSign numberOfArguments]>2){

selectorsArray=[[NSString stringWithString:protSelector] componentsSeparatedByString:@":"];

for (i=2; i<[methodSign numberOfArguments]; i++){

ad=i-2;

argCount=ad+1;

finString=finString.toString()+selectorsArray[ad].toString()+":("+constructTypeAndName(commonTypes([methodSign getArgumentTypeAtIndex:i].toString()),"",0).toString()+")"+"arg"+argCount.toString()+" ";

}

finString=[finString substringToIndex:finString.length-1];

}

else{

finString=protSelector.toString();

}

finString=finString.toString()+";";

protocolsMethodsString=protocolsMethodsString.toString()+startSign.toString()+"("+returnType.toString()+")"+finString.toString()+"\n";

}

free(protMeths);

}

return protocolsMethodsString.toString()+"@end\n";

}

function constructTypeAndName(aType,IvarName,isIvar){

//NSNotFound=2147483647;

space=isIvar ? " " : "";

compareString1=[NSString stringWithString:aType];

compareString2=[[[NSString stringWithString:aType] stringByReplacingOccurrencesOfString:"^" withString:""] stringByAppendingString:@"\*"];

if (![[NSString stringWithString:commonTypes(aType)] isEqual:compareString1] && ![[NSString stringWithString:commonTypes(aType)] isEqual:compareString2]){

return commonTypes(aType).toString()+" "+IvarName.toString();

}

charSet=[NSCharacterSet characterSetWithCharactersInString:"@^\"{}="];

structCharSet=[NSCharacterSet characterSetWithCharactersInString:"?:{}="];

if ([aType rangeOfString:"]"].location!=NSNotFound && [aType rangeOfString:"^{"].location==NSNotFound){

aType=[aType stringByRemovingCharactersFromSet: [NSCharacterSet punctuationCharacterSet ]];

arrayCount=[[aType copy] stringByRemovingCharactersFromSet: [NSCharacterSet letterCharacterSet ]];

arrayType=[aType stringByRemovingCharactersFromSet: [NSCharacterSet decimalDigitCharacterSet ]];

return commonTypes(arrayType).toString()+"["+arrayCount.toString()+"]"+space+IvarName.toString();

}

if ([aType rangeOfString:"{?"].location!=NSNotFound && [aType rangeOfString:"{?"].length>0 && isIvar){

aType=[aType stringByRemovingCharactersFromSet:structCharSet];

structValues=[aType componentsSeparatedByString:@"\""];

structValues =[NSMutableArray arrayWithArray:structValues ];

firstEntry=[structValues removeObjectAtIndex:0];

[structValues removeObject:firstEntry];

newString=[NSString stringWithString:"struct {\n"];

namesArray=[NSMutableArray array];

typesArray=[NSMutableArray array];

for (d=0; d<[structValues count] ; d++){

if ((d % 2)==0){

[namesArray addObject:structValues[d]];

}

else{

[typesArray addObject:structValues[d]];

}

}

for (e=0; e<[typesArray count]; e++){

newString=newString.toString()+"\t\t"+constructTypeAndName(typesArray[e],namesArray[e],0).toString()+";\n";

}

return newString.toString()+"\t} "+IvarName.toString();

}

if ([aType rangeOfString:"{"].location!=NSNotFound && [aType rangeOfString:"{"].length>0){

returnValue="struct ";

range=[aType rangeOfString:@"="];

if ([aType containsSubstring:@"^{?="]){

aStruct=aType;

structString="";

someType="";

aStruct=[NSString stringWithString:aStruct];

aStruct=[aStruct stringByReplacingOccurrencesOfString:@"^{?=" withString:""];

aStruct=[aStruct stringByReplacingOccurrencesOfString:@"}" withString:""];

for (var f=0; f<[aStruct.toString() length]; f++){

currentLetter=[aStruct substringWithRange:[f,1]];

someType=constructTypeAndName(currentLetter,"",0);

someType=[someType stringByRemovingWhitespace];

structString=structString.toString()+"\t"+someType.toString()+" value"+(f+1).toString()+";\n";

}

structName="WCStruct\_"+aStruct.toString();

if (![structsString containsSubstring:structName]){

structString="typedef struct{\n"+structString.toString();

structString=structString.toString()+"} "+structName.toString()+";\n\n";

structsString=structsString.toString()+structString.toString();

}

structName=structName.toString()+"\*";

return structName+" "+IvarName.toString();

}

aType=[aType stringByReplacingCharactersInRange:[range.location,aType.toString().length-range.location] withString:"" ];

returnValue=returnValue.toString()+[aType stringByRemovingCharactersFromSet:structCharSet].toString();

if ([returnValue containsSubstring:@"GSEvent"] || [returnValue containsSubstring:@"CTCall"]){

returnValue=[returnValue stringByReplacingOccurrencesOfString:"\_\_" withString:""];

returnValue=[returnValue stringByReplacingOccurrencesOfString:"struct " withString:""];

returnValue=[returnValue stringByReplacingOccurrencesOfString:"^" withString:""];

returnValue=[returnValue stringByAppendingString:@"Ref"];

}

if ([returnValue containsSubstring:@"NSZone"]){

returnValue="NSZone\*";

}

if ([returnValue containsSubstring:@"CGPoint"] || [returnValue containsSubstring:@"CGRect"] || [returnValue containsSubstring:@"CGSize"] ){

returnValue=[returnValue stringByReplacingOccurrencesOfString:@"struct " withString:@""];

}

return commonTypes(returnValue).toString()+space+IvarName.toString();

}

if ([aType rangeOfString:@"^"].location!=NSNotFound && [aType rangeOfString:@"^"].length>0){

range=[aType rangeOfString:"^" options: NULL range: [2,aType.toString().length-2]];

if (range.length>0){

aType=[aType stringByReplacingCharactersInRange:[range.location-1,aType.toString().length-range.location+1] withString:"" ];

}

aType=[aType stringByRemovingCharactersFromSet:charSet];

//aType=[aType stringByReplacingOccurrencesOfString:@"\_\_" withString:""];

return aType.toString()+"\* "+IvarName.toString();

}

if ([aType rangeOfString:@"@\""].location!=NSNotFound && [aType rangeOfString:@"@\""].length>0){

if ([aType rangeOfString:"<"].location==2){

aType="id"+aType.toString();

return [aType stringByRemovingCharactersFromSet:charSet].toString()+" "+IvarName.toString();

}

strippedString=[aType stringByRemovingCharactersFromSet:charSet];

return strippedString.toString()+ "\* "+IvarName.toString();

}

if ([aType rangeOfString:@"b"].location!=NSNotFound && [aType rangeOfString:@"b"].length>0 && [aType rangeOfString:":{"].length<1){

string=[aType stringByReplacingOccurrencesOfString:@"b" withString:""];

return "unsigned int "+IvarName.toString()+":"+string.toString();

}

return aType.toString() + space + IvarName.toString();

}

function propertyLineGenerator(attributes,name){

parSet=[NSCharacterSet characterSetWithCharactersInString:@"()"];

attributes=[attributes stringByRemovingCharactersFromSet:parSet];

attrArr=[attributes componentsSeparatedByString:@","];

type=attrArr[0];

type=[type stringByReplacingCharactersInRange:[0,1] withString:""];

type=constructTypeAndName(type,"",0);

type=[type stringByRemovingWhitespace];

attrArr=[NSMutableArray arrayWithArray:attrArr];

[attrArr removeObjectAtIndex:0];

propertyString="@property ";

newPropsArray=[NSMutableArray array];

synthesize=[NSString stringWithString:""];

for each (attr in attrArr){

vToClear=nil;

if ([attr rangeOfString:@"V\_"].location==0){

vToClear=attr;

attr=[attr stringByReplacingCharactersInRange:[0,2] withString:""];

synthesize="\t\t\t\t//@synthesize "+attr.toString()+"=\_"+attr.toString()+" - In the implementation block";

}

if ([attr.toString() length]==1){

switch (attr.toString()){

case "R" : translatedProperty = "readonly";

case "C" : translatedProperty = "copy"; break;

case "&" : translatedProperty = "retain"; break;

case "N" : translatedProperty = "nonatomic"; break;

case "D" : translatedProperty = "@dynamic"; break;

case "W" : translatedProperty = "\_\_weak"; break;

case "P" : translatedProperty = "t<encoding>"; break;

default: translatedProperty = attr;

}

[newPropsArray addObject:translatedProperty];

}

if ([attr rangeOfString:@"G"].location==0){

attr=[attr stringByReplacingCharactersInRange:[0,1] withString:""];

attr="getter="+attr.toString();

[newPropsArray addObject:attr];

}

if ([attr rangeOfString:@"S"].location==0){

attr=[attr stringByReplacingCharactersInRange:[0,1] withString:""];

attr="setter="+attr.toString();

[newPropsArray addObject:attr];

}

}

if ([newPropsArray containsObject:@"nonatomic"] && ![newPropsArray containsObject:@"assign"] && ![newPropsArray containsObject:@"readonly"] && ![newPropsArray containsObject:@"copy"] && ![newPropsArray containsObject:@"retain"]){

[newPropsArray addObject:@"assign"];

}

newPropsArray=[newPropsArray reversedArray];

rebuiltString=[newPropsArray componentsJoinedByString:","];

attrString=newPropsArray.length>0 ? "("+rebuiltString.toString()+")" : "(assign)";

propertyString=propertyString.toString()+attrString.toString()+" "+type.toString()+" "+name.toString()+"; "+synthesize.toString()+"\n";

return propertyString;

}

function methodLinesGenerator(methodList,methodsCount,isClassMethod){

methodLines="";

for (n=0; n<\*methodsCount;n++){

method=methodList[n];

methodName=method\_getName(method);

if (methodName==".cxx\_destruct"){

continue;

}

returnType=\_method\_copyReturnType(method);

returnType=[constructTypeAndName(returnType.toString(),[NSString stringWithString:@""],0).toString() stringByRemovingWhitespace];

argNum=method\_getNumberOfArguments(method);

methodBrokenDown=[methodName.toString() componentsSeparatedByString:@":"];

methodString=[NSString stringWithString:""];

if ([methodBrokenDown count]>1){

for (x=0; x<[methodBrokenDown count]-1; x++){

anIndex=x+2;

argumentType=\_method\_copyArgumentType(method,anIndex);

var hadType=YES;

if (!argumentType){

hadType=NO;

argumentType="id";

}

typeName=constructTypeAndName(argumentType.toString(),[NSString stringWithString:""],0);

//if (hadType){

//free(argumentType); //cannot free since its a \*

//}

typeName=[typeName substringToIndex:typeName.toString().length-1];

methodString=methodString.toString()+methodBrokenDown[x].toString()+":("+typeName.toString()+")arg"+(x+1)+" ";

}

methodString=[methodString substringToIndex:methodString.length-1];

}

else{

methodString=methodName;

}

symbol=isClassMethod ? "+" : "-"; symbol=[NSString stringWithString:symbol.toString()];

newMethod=symbol.toString()+"("+returnType.toString()+")"+methodString.toString()+";\n";

cappedMethod=[[NSString stringWithString:methodName.toString()] capitalizedString];

setterMethod="set"+cappedMethod.toString();

//free(returnType);

if (![methodsArray containsObject:newMethod.toString()] && ![propertiesString containsSubstring:methodName.toString()] && ![methodsString containsSubstring:setterMethod.toString()]){

[methodsArray addObject:newMethod];

methodLines=methodLines.toString()+newMethod.toString();

}

}

return methodLines;

}

function weak\_classdump(classname,alsoDumpSuperclasses,outputdir){

//[[NSString stringWithString:classname.toString()] writeToFile:"/tmp/LASTCLASS" atomically:YES];

//NSLog(@"weak\_classdump: Dumping class %@",classname);

if (!classname){

return "Cannot find class";

}

if (typeof(alsoDumpSuperclasses) == 'undefined' || !alsoDumpSuperclasses){

alsoDumpSuperclasses=0;

}

structsString="";

interfaceString="";

version = [NSProcessInfo processInfo ].operatingSystemVersionString;

loc=[NSLocale localeWithLocaleIdentifier: "en-us"];

date=[NSDate.date descriptionWithLocale: loc];

classString = "/\*\n \* This header is generated by weak\_classdump 0.2\n \* on "+date.toString()+"\n \* Operating System: "+version.toString()+"\n \* weak\_classdump is Freeware by Elias Limneos.\n \*\n \*/\n\n";

if ( [[classname description] containsSubstring:@"<Protocol:"]){

classString=classString.toString()+getProtocolLines(classname).toString();

if (typeof(outputdir) == 'undefined'){

outputdir = "/tmp";

}

if (typeof(alsoDumpSuperclasses) == 'string'){

outputdir=alsoDumpSuperclasses;

}

outputdir=outputdir.toString()+"/";

if (![NSFileManager.defaultManager fileExistsAtPath:outputdir]){

try{

[NSFileManager.defaultManager createDirectoryAtPath:outputdir withIntermediateDirectories:YES attributes:nil error:nil];

}catch(e){}

}

classString = [NSString stringWithString:classString ];

if ([classString writeToFile:outputdir.toString()+protocol\_getName(classname).toString()+".h" atomically:YES]){

return "Wrote /PROTOCOL/ header file to "+outputdir.toString()+protocol\_getName(classname).toString()+".h";

}

else {

NSSearchPathForDirectoriesInDomains=new Functor(dlsym(RTLD\_DEFAULT,"NSSearchPathForDirectoriesInDomains"),"@ccc");

writeableDir=NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);

writeableDir=writeableDir[0];

return "Failed to write to "+outputdir.toString()+protocol\_getName(classname).toString()+".h - Check file path and permissions? Suggested writeable directory: "+writeableDir.toString();

}

}

methodsArray=[NSMutableArray array];

propertiesString=@"";

methodsString=@"";

ivarsString=@"";

classMethodsString=@"";

startingClassname=classname;

superclass=classname.superclass;

protocolsCount=new int;

protocolArray=class\_copyProtocolList(classname,protocolsCount);

protocolsMethodsString=classString;

var protocolName;

allProtocols="";

if (\*protocolsCount>0){

for (iter=0; iter<\*protocolsCount; iter++){

if (class\_conformsToProtocol(classname,protocolArray[iter])){

protocolName=protocol\_getName(protocolArray[iter]);

protocolsMethodsString=protocolsMethodsString.toString()+getProtocolLines(protocolArray[iter]).toString();

}

protocolsMethodsString = [NSString stringWithString:protocolsMethodsString ];

classString=classString.toString()+"#import <"+protocolName.toString()+".h>\n";

if (typeof(outputdir)=="undefined" || outputdir==null){

outputdir="/tmp";

}

lastChar=[outputdir.toString() substringFromIndex:outputdir.toString().length-1];

if (![lastChar isEqual:@"/"]){

outputdir=outputdir+"/";

}

allProtocols=allProtocols.toString()+", "+outputdir.toString()+protocolName.toString()+".h";

if ([protocolsMethodsString writeToFile:outputdir.toString()+protocolName.toString()+".h" atomically:YES]){

//NSLog(@"Found Protocol %@, wrote to %@%@.h",protocolName,outputdir,protocolName);

}

}

}

protocolsString="";

if (\*protocolsCount>0){

protocolsString=@" <".toString();

for (i=0; i<\*protocolsCount; i++){

if (class\_conformsToProtocol(classname,protocolArray[i])){

comma=@"".toString();

if (i<\*protocolsCount-1){

comma=@", ".toString();

}

protocolsString=protocolsString.toString()+protocol\_getName(protocolArray[i]).toString()+comma;

}

}

protocolsString=protocolsString+@">".toString();

}

free(protocolArray);

if (classname.superclass!=nil && classname.superclass!="nil"){

interfaceString = [NSString stringWithString:@"\n@interface "+classname.toString()+" : "+classname.superclass.toString()].toString() + protocolsString.toString();

}

else{

interfaceString = [NSString stringWithString:@"\n@interface "+classname.toString()].toString() + protocolsString.toString();

}

while (classname!=NSObject && (classname.superclass!="nil" && classname.superclass!=NSObject) ) {

// Get Ivars

classIvarCount=new int;

superclassIvarCount=new int;

list=class\_copyIvarList(classname,classIvarCount);

superlist=class\_copyIvarList(superclass,superclassIvarCount);

superClassIvars=[NSMutableArray array];

for (i=0; i<\*superclassIvarCount;i++){

if (ivar\_getName(superlist[i])){

[superClassIvars addObject:ivar\_getName(superlist[i])];

}

}

free(superlist);

for (i=0; i<\*classIvarCount;i++){

classIvar=ivar\_getName(list[i]);

appendString="";

if (classIvar && ![superClassIvars containsObject:classIvar]){

ivarType=ivar\_getTypeEncoding(list[i]).toString();

ivar=constructTypeAndName([NSString stringWithString:ivarType],[NSString stringWithString:ivar\_getName(list[i])],1);

newString="\n\t"+ivar.toString()+"; ";

if (![ivarsString containsSubstring:newString]){

ivarsString=ivarsString.toString()+newString.toString();

}

}

}

free(list);

// Get Properties

propertiesCount=new int;

propertyList=class\_copyPropertyList(classname,propertiesCount);

for (i=0; i<\*propertiesCount; i++){

propname=property\_getName(propertyList[i]);

attrs=property\_getAttributes(propertyList[i]);

newString=propertyLineGenerator(attrs,propname).toString();

if (![propertiesString containsSubstring:newString]){

propertiesString=propertiesString.toString()+newString.toString();

}

}

free(propertyList);

// Get Methods

methodsCount=new int;

classMethodsCount=new int;

classMethodList=class\_copyMethodList(object\_getClass(classname),classMethodsCount);

methodList=class\_copyMethodList(classname,methodsCount);

classMethodsString=classMethodsString.toString()+methodLinesGenerator(classMethodList,classMethodsCount,1).toString();

methodsString=methodsString.toString()+methodLinesGenerator(methodList,methodsCount,0).toString();

free(methodList);

free(classMethodList);

if (!alsoDumpSuperclasses)

break;

classname=classname.superclass ? classname.superclass : NSObject;

}

classString= classString.toString()+structsString.toString()+interfaceString.toString();

classString = classString.toString()+" {"+ivarsString.toString()+"\n}\n"+propertiesString.toString()+classMethodsString.toString()+methodsString.toString();

classString = classString.toString()+"@end";

if (typeof(outputdir) == 'undefined'){

outputdir = "/tmp";

}

if (typeof(alsoDumpSuperclasses) == 'string'){

outputdir=alsoDumpSuperclasses;

}

isDir= new boolean;

dirExists=[[NSFileManager defaultManager ] fileExistsAtPath:outputdir isDirectory: isDir] ;

if (!dirExists || !isDir){

createDirSucceeded = [[NSFileManager defaultManager ] createDirectoryAtPath:outputdir attributes: nil];

}

outputdir=outputdir.toString()+"/";

if (![NSFileManager.defaultManager fileExistsAtPath:outputdir]){

try{

[NSFileManager.defaultManager createDirectoryAtPath:outputdir withIntermediateDirectories:YES attributes:nil error:nil];

}catch(e){}

}

classString = [NSString stringWithString:classString ];

if ([classString writeToFile:outputdir.toString()+startingClassname.toString()+".h" atomically:YES]){

return "Wrote file to "+outputdir.toString()+startingClassname.toString()+".h"+allProtocols.toString();

}

else {

NSSearchPathForDirectoriesInDomains=new Functor(dlsym(RTLD\_DEFAULT,"NSSearchPathForDirectoriesInDomains"),"@ccc");

writeableDir=NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);

writeableDir=writeableDir[0];

return "Failed to write to "+outputdir.toString()+startingClassname.toString()+".h - Check file path and permissions? Suggested writeable directory: "+writeableDir.toString();

}

}

function mweak\_loadwdc\_class(){

@implementation WCDBundleDumper : NSObject {}

+(id)dumpBundle:(id)infoDictionary{

var bundle=[infoDictionary objectForKey:@"bundle"];

var outputdir=[infoDictionary objectForKey:@"outputdir"];

for (d = 0; d<[ObjectiveC.classes allKeys].length; d++) {

name=[ObjectiveC.classes allKeys][d].toString();

if ([[NSBundle bundleForClass:objc\_getClass(name.toString())] isEqual:bundle]){

try {

weak\_classdump(objc\_getClass(objc\_getClass(name.toString())), false, outputdir);

}

catch (e) {

}

}

}

if (typeof(\_\_AudioServicesPlaySystemSound)=="undefined"){

dlopen("/System/Library/Frameworks/AudioToolbox.framework/AudioToolbox",RTLD\_LAZY);

\_\_AudioServicesPlaySystemSound=new Functor(dlsym(-2,"AudioServicesPlaySystemSound"),"vi");

}

\_\_AudioServicesPlaySystemSound(1005); // comment out to not produce any sound on finish

//NSLog(@"weak\_classdump: Finished dumping bundle %@. Check output dir %@",bundle,outputdir);

}

@end

}

function weak\_classdump\_bundle(bundle, outputdir) {

if (typeof(bundle)=="undefined"){

bundle=[NSBundle mainBundle];

}

if (![bundle isLoaded]){

//NSLog(@"weak\_classdump: Bundle %@ is not loaded,attempting to load it",bundle);

[bundle load];

}

if (typeof(outputdir)=="undefined"){

outputdir="/tmp/";

}

var infoDict=[NSMutableDictionary dictionary];

[ infoDict setObject:bundle forKey:@"bundle"];

[ infoDict setObject:outputdir forKey:@"outputdir"];

try {

[WCDBundleDumper class];

} catch (e){

mweak\_loadwdc\_class();

}

[objc\_getClass("WCDBundleDumper") performSelectorInBackground:@selector(dumpBundle:) withObject:infoDict ];

return "Dumping bundle... Will play alert sound when done. Check output directory for changes";

}

if (! [NSString instancesRespondToSelector:@selector(containsSubstring:)]){

@implementation NSString (lim)

-(BOOL)containsSubstring:(NSString\*)string{

return [self rangeOfString:string].location!=NSNotFound;

}

@end

}

if ( ! [NSString instancesRespondToSelector:@selector(stringByRemovingCharactersFromSet:)] ){

@implementation NSString (weakclassdump\_compatibility)

- (void)removeCharactersInSet:(id)set{

length = [this length];

matchRange = [this rangeOfCharacterFromSet:set options:2 range:[0, length]];

while(matchRange.length > 0){

replaceRange = matchRange;

searchRange=[0,0];

searchRange.location = replaceRange.location + replaceRange.length;

searchRange.length = length - searchRange.location;

for(;;){

matchRange = [this rangeOfCharacterFromSet:set options:2 range:searchRange];

if((matchRange.length == 0) || (matchRange.location != searchRange.location))

break;

replaceRange.length += matchRange.length;

searchRange.length -= matchRange.length;

searchRange.location += matchRange.length;

}

[this deleteCharactersInRange:replaceRange];

matchRange.location -= replaceRange.length;

length -= replaceRange.length;

}

}

- (id)stringByRemovingCharactersFromSet:(id)set{

if([this rangeOfCharacterFromSet:set options:2].length == 0)

return this;

temp = [[this mutableCopyWithZone:[this zone]] autorelease];

[temp removeCharactersInSet:set];

temp=[temp stringByReplacingOccurrencesOfString: @"\"" withString: @""];

return temp;

}

@end

}

if ( ! [NSString instancesRespondToSelector:@selector(stringByRemovingWhitespace)] ){

@implementation NSString (weakclassdump\_compatibility)

-(id)stringByRemovingWhitespace{

return [this stringByRemovingCharactersFromSet:[NSCharacterSet whitespaceCharacterSet]];

}

@end

}

// Usage example : weak\_classdump(SBAwayController);

// (will write to default path "/tmp/SBAwayController.h"

// example 2: weak\_classdump(UIApplication,"/var/mobile/");

// will write to "/var/mobile/UIApplication.h"

// example 3: weak\_classdump\_bundle([NSBundle bundleWithPath:"/System/Library/Frameworks/iAd.framework"]);

// will dump all classes in the defined bundle to default dir "/tmp"

// example 4: weak\_classdump\_bundle([NSBundle bundleWithPath:"/System/Library/Frameworks/iAd.framework"],"/tmp/iAD.framework/Headers/");

// will dump all classes in the defined bundle to "/tmp/iAD.framework/"

"Added weak\_classdump to \""+NSProcessInfo.processInfo .processName.toString()+"\" ("+NSProcessInfo.processInfo .processIdentifier.toString()+")";