master 1 branch 0 tags Go to file Code

4ch12dy Update README.md 0df134a on 18 Apr 169 commits

.vscode	v2.9	2 years ago
ida-block-search	添加支持IDA 7.4和python3的脚本	9 months ago
resource	update README	3 years ago
src	Merge branch 'master' of github.com:4ch12dy/xia0LLDB	last month
.gitignore	x	2 years ago
README.md	Update README.md	last month
install.sh	re write install.sh	2 years ago

About

LLDB python scripts for iOS arm64 reversing by xia0

reverse-engineering debug lldb

Readme
473 stars
16 watching
86 forks

Releases

No releases published

Packages

No packages published

Contributors 4

- 4ch12dy X!A0
- rozbo 键来!
- MustangYM Shelby
- xia0z xia0

Languages

Python 99.6% Shell 0.4%

README.md

xia0LLDB 😈

```
         https://github.com/4ch12dy/xia0LLDB
        Welcome to xia0LLDB - Python3 Edition
        ,--.           ,--.  ,--.   ,--.   ,------.  ,-----.
,--.  ,--.`--' ,--,--. /    \ |  |   |  |   |  .-.  \ |  |) /_
 \  `'  / ,--.' ,-.  ||  ()  ||  |   |  |   |  |  \  :|  .-.  \
 /  /.  \ |  |\ '-'  | \    / |  '--.|  '--.|  '--'  /|  '--' /
'--'  '--'`--' `--`--'  `--'  `-----'`-----'`-------' `------'

[xia0LLDB] * Version: v3.1
[xia0LLDB] + Loading all scripts from ~/xia0/iOSRE/LLDB/xia0LLDB
[xia0LLDB] * Finished
```

Notice(^_<)

~~There is a problem that lldb import xia0LLDB in last macOS Catalina, because the last macOS's lldb default use python3. Here is a way to change it to python2~~

~~`defaults write com.apple.dt.lldb DefaultPythonVersion 2`~~

Welcome to xia0LLDB - Python3 Edition

Thanks @Lakr so much for porting it to Python3 !

Install

Just open Terminal and run below command

`git clone https://github.com/4ch12dy/xia0LLDB.git && cd xia0LLDB && ./install.sh`

It highly recommend you to install issh/Tap2debug

Happy debugging ✔️

Commands

alias

Below is cmds just use alias in cmd.txt

- mload [dylib_in_the_iphone_device_path]

  Load a dylib into current process

- rr

  Fast show some important regiters

- pwindow

  Print current key windown

- xi [code_address]

  just show address disassmble +/- 8

- dfuc [addr_of_func]

  show function all disassemble by given address

- pclass [oc_object]

  print oc object class name

- pbcopy

  get string from iOS device pasteboard

- pbpaste [string]

  paste string to iOS device pasteboard

- data [object_of_NSData]

  print NSData object

- pcc

  It is just alias of `process connect connect://127.0.0.1:1234`

- wpc

  write pc register to control exe process

croc

👉👉👉 go to the env that can run oc script. This cmd is always used when backboard debug luanch app, debuger just attch on.The point is between app code not execute and can run lldb commands.So try use it when backboard debug luanch app.

ivars

print all ivars of OC object (iOS Only) and **macOS version will come soon!**

```
(lldb) ivars 0x2835c4d00
<CContactMgr: 0x2835c4d00>:
in CContactMgr:
        m_oLock (NSRecursiveLock*): <NSRecursiveLock: 0x2830aaca0>
        m_uiLoadedType (unsigned int): 0
        m_oContactDB (CContactDB*): <CContactDB: 0x2819b07e0>
        m_oNewContactDB (NewContactDB*): <NewContactDB: 0x28156b7e0>
        m_oContactOPLog (CContactOPLog*): <CContactOPLog: 0x2819b07f0>
        m_openImContactMgr (OpenImContactMgr*): <OpenImContactMgr: 0x281bc07a0>
        m_dicRemark (NSMutableDictionary*): <__NSDictionaryM: 0x281bc0a00>
        m_dicLastAccessTime (NSMutableDictionary*): <__NSDictionaryM: 0x281bc0a60>
        m_dicContacts (NSMutableDictionary*): <__NSDictionaryM: 0x281bc09e0>
...
```

methods

print all methods of OC object (iOS Only) and **macOS version will come soon!**

if the objc class name contains space like " m" or other odd characters. you can use "methods -n the_odd_class_name."

```
(lldb) methods CContactMgr
<CContactMgr: 0x1071caa28>:
in CContactMgr:
        Properties:
                @property (readonly) unsigned long hash;
                @property (readonly) Class superclass;
                @property (readonly, copy) NSString* description;
                @property (readonly, copy) NSString* debugDescription;
        Instance Methods:
                - (void) MessageReturn:(id)arg1 Event:(unsigned int)arg2; (0x1005cb338)
                - (id) getContactByName:(id)arg1; (0x1000f4e74)
                - (void) OnGetNewXmlMsg:(id)arg1 Type:(id)arg2 MsgWrap:(id)arg3; (0x1001de380)
                - (void) onServiceReloadData; (0x102d10934)
...

(lldb) methods -n " m"
[*] will get methods for class:" m"
< m: 0x10d6f86f0>:
in  m:
        Properties:
                @property (retain, nonatomic)  N*  kManager;  (@synthesize  kManager = _configMan
                @property (retain, nonatomic)  h* payloadStore;  (@synthesize payloadStore = _pay
                @property (retain, nonatomic)  5* sensorAgent;  (@synthesize sensorAgent = _senso
                @property (retain, nonatomic) NSObject<OS_dispatch_queue>* scriptMsgQueue;  (@syn
  ...
        Instance Methods:
                - (void) setConfigManager:(id)arg1; (0x10d65b68c)
                - (void) setSensorAgent:(id)arg1; (0x10d5c86d0)
                - (void)  lb; (0x10d60aa04)
                - (void) setKernelCode:(id)arg1; (0x10d6d9330)
                - (void) setIsBaseKernel:(BOOL)arg1; (0x10d606168)
  ...
```

freshxlldb

Re import xia0LLDB from lldbinit

sbt [2018/08/04]

the replacement of `bt` , it can restore frame OC symbol on stackframe. if you want to restore block symbol, you can use the ida python script provided to get block symbol json file. then input `sbt -f block_json_file_path` in lldb. Beside it can show more infomation: mem address, file address

```
// also you can spcail -f block_json_file to restore block symbol
(lldb) sbt
==========================================xia0LLDB==========================================
  BlockSymbolFile    Not Set The Block Symbol Json File, Try 'sbt -f'
============================================================================================
  frame #0: [file:0x100009740 mem:0x100fb1740] WeChat`-[MMServiceCenter getService:] + 0
  frame #1: [file:0x100017cd4 mem:0x100fbfcd4] WeChat`+[SettingUtil getMainSetting] + 88
  frame #2: [file:0x10004eef0 mem:0x100ff6ef0] WeChat`-[CDownloadVoiceMgr TimerCheckDownloadQueue
  frame #3: [file:0x1800a3604 mem:0x1ccb33604] libobjc.A.dylib`-[NSObject performSelector:withObj
  frame #4: [file:0x10002e92c mem:0x100fd692c] WeChat`-[MMNoRetainTimerTarget onNoRetainTimer:] +
  frame #5: [file:0x1819750bc mem:0x1ce4050bc] Foundation`__NSFireTimer + 88
  frame #6: [file:0x180e3d0a4 mem:0x1cd8cd0a4] CoreFoundation`__CFRUNLOOP_IS_CALLING_OUT_TO_A_TIM
  frame #7: [file:0x180e3cdd0 mem:0x1cd8ccdd0] CoreFoundation`__CFRunLoopDoTimer + 884
  frame #8: [file:0x180e3c5c4 mem:0x1cd8cc5c4] CoreFoundation`__CFRunLoopDoTimers + 252
  frame #9: [file:0x180e37284 mem:0x1cd8c7284] CoreFoundation`__CFRunLoopRun + 1832
  frame #10: [file:0x180e36844 mem:0x1cd8c6844] CoreFoundation`CFRunLoopRunSpecific + 452
  frame #11: [file:0x1830e5be8 mem:0x1cfb75be8] GraphicsServices`GSEventRunModal + 104
  frame #12: [file:0x1ae78431c mem:0x1fb21431c] UIKitCore`UIApplicationMain + 216
  frame #13: [file:0x10022ee88 mem:0x1011d6e88] WeChat`main + 556
  frame #14: [file:0x1808ec020 mem:0x1cd37c020] libdyld.dylib`start + 4
```

choose [2019/07/21]

get instance object of given class name, a lldb version of cycript's choose command

```
(lldb) choose CContactMgr
====>xia0LLDB NSArray Address: 0x2815a8540      size: 0x1
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
V  V  V  V  V  V  V  V  V  V  V  V  V  V  V  V  V  V  V  V
======>xia0LLDB Object Address: 0x2835c4d00
<CContactMgr: 0x2835c4d00>
```

xbr [2019/08/11]

xia0 super set breakpoint command:set breakpoint at OC class method although strip symbol and so on

```
// set breakpoint at oc methold even symbol stripped
(lldb) xbr "-[MMServiceCenter getService:]"
[*] className:MMServiceCenter methodName:getService:
[+] found class address:0x10803d208
[+] found selector address:0x106425b4c
[+] found method address:0x100fb1740
Breakpoint 1: where = WeChat`___lldb_unnamed_symbol50$$WeChat, address = 0x0000000100fb1740

// set breakpoint at address of ida, auto add slide
(lldb) xbr 0x100009740
[*] you not specail the module, default is main module
[*] ida's address:0x100009740 main module slide:0xfa8000 target breakpoint address:0x100fb1740
Breakpoint 3: where = WeChat`___lldb_unnamed_symbol50$$WeChat, address = 0x0000000100fb1740

// set breakpoint at memory address
(lldb) xbr -a 0x100fb1740
[*] breakpoint at address:0x100fb1740
Breakpoint 4: where = WeChat`___lldb_unnamed_symbol50$$WeChat, address = 0x0000000100fb1740

// set breakpoint at main function
(lldb) xbr -E main
[*] breakpoint at main function:0x1011d6c5c
Breakpoint 5: where = WeChat`___lldb_unnamed_symbol7390$$WeChat, address = 0x00000001011d6c5c

// set breakpoint at first mod_init function
(lldb) xbr -E init
[*] breakpoint at mod int first function:0x1044553dc
Breakpoint 6: where = WeChat`___lldb_unnamed_symbol143513$$WeChat, address = 0x00000001044553dc

//  set breakpoint at adresses of all methods of given class name
(lldb) xbr UPLivePlayerVC
Breakpoint 1: where = TestPaly`-[UPLivePlayerVC progressSliderSeekTime:] at UPLivePlayerVC.m:205,
Breakpoint 2: where = TestPaly`-[UPLivePlayerVC progressSliderTouchDown:] at UPLivePlayerVC.m:197
Breakpoint 3: where = TestPaly`-[UPLivePlayerVC progressSliderValueChanged:] at UPLivePlayerVC.m:
...
Breakpoint 45: where = TestPaly`-[UPLivePlayerVC setUrl:] at UPLivePlayerVC.h:13, address = 0x000
Breakpoint 46: where = TestPaly`-[UPLivePlayerVC play] at UPLivePlayerVC.m:124, address = 0x00000
Breakpoint 47: where = TestPaly`-[UPLivePlayerVC pause] at UPLivePlayerVC.m:132, address = 0x0000
Set 47 breakpoints of UPLivePlayerVC

// set breakpoint at all +[* load] methods
(lldb) xbr -E load
[*] will set breakpoint at all +[* load] methold, count:2
Breakpoint 2: where = TestAPP`+[OCTest load] at OCTest.m:19, address = 0x00000001042df674
[+] set br at:0x1042df674
Breakpoint 3: where = TestAPP`+[OCClassDemo load] at OCClassDemo.m:19, address = 0x00000001043027
[+] set br at:0x10430272c
```

debugme [2019/08/13]

bypass anti-debug: can hook ptrace and inlinehook svc to kill anti debug. it is so strong ever!!!

```
[*] start patch ptrace funtion to bypass antiDebug
[+] success ptrace funtion to bypass antiDebug
[*] start patch svc ins to bypass antiDebug
[+] get text segment start address:0x100017430 and end address:0x10001a398
[+] found svc address:0x100017528
[*] start hook svc at address:0x100017528
[+] success hook svc at address:0x100017528
[+] found svc address:0x100017540
[*] start hook svc at address:0x100017540
[+] success hook svc at address:0x100017540
[*] all patch done
[x] happy debugging~ kill antiDebug by xia0@2019
```

info [2019/08/20]

very useful command to get info of address/function/module and so on

```
// get info of image
(lldb) info -m WeChat
=======
Module Path : /var/containers/Bundle/Application/747A9704-6252-45A9-AE55-59690DAD60BB/WeChat.app/
Module Silde: 0x7d4000
Module base : 0x1007d4000
=======

// get info of address of function
(lldb) info -a 0x00000001cd4ca3b8
Module Path: /usr/lib/system/libsystem_kernel.dylib
Module base: 0x1cd4a8000
Symbol name: __getpid
Symbol addr: 0x1cd4ca3b8

// get info of function
(lldb) info -f getpid
Func   name: getpid
Func   addr: 0x1cd4ca3b8
Module Path: /usr/lib/system/libsystem_kernel.dylib
Module base: 0x1cd4a8000
Symbol name: __getpid
Symbol addr: 0x1cd4ca3b8
```

dumpdecrypted [2019/09/22]

dump macho image in lldb, default dump all macho image.

👇👇👇 very important!!!

Notice: if app crash at launch like detect jailbreak, you should use -x backboard launch app, and just input `dumpdecrypted -X` see more: http://4ch12dy.site/2020/02/26/lldb-how-to-dump-gracefully/lldb-how-to-dump-gracefully/

```
(lldb) dumpdecrypted
[*] start dump image:/var/containers/Bundle/Application/701B4574-1606-41F3-B0DB-92D34F92E886/com_

[+] Dumping com_kwai_gif
[+] detected 64bit ARM binary in memory.
[+] offset to cryptid found: @0x100014980(from 0x100014000) = 980
[+] Found encrypted data at address 00004000 of length 16384 bytes - type 1.
[+] Opening /private/var/containers/Bundle/Application/701B4574-1606-41F3-B0DB-92D34F92E886/com_k
[+] Reading header
[+] Detecting header type
[+] Executable is a plain MACH-O image
[+] Opening /var/mobile/Containers/Data/Application/23C75F90-C42D-4F43-83D9-5DCCA36FE2D5/Document
[+] Copying the not encrypted start of the file
[+] Dumping the decrypted data into the file
[+] Copying the not encrypted remainder of the file
[+] Setting the LC_ENCRYPTION_INFO->cryptid to 0 at offset 980
[+] Closing original file
[+] Closing dump file
[*] This mach-o file decrypted done.
[+] dump macho file at:/var/mobile/Containers/Data/Application/23C75F90-C42D-4F43-83D9-5DCCA36FE2


[*] start dump image:/private/var/containers/Bundle/Application/701B4574-1606-41F3-B0DB-92D34F92E

[+] Dumping gifIMFramework
[+] detected 64bit ARM binary in memory.
[+] offset to cryptid found: @0x100064bd0(from 0x100064000) = bd0
[+] Found encrypted data at address 00004000 of length 2752512 bytes - type 1.
[+] Opening /private/var/containers/Bundle/Application/701B4574-1606-41F3-B0DB-92D34F92E886/com_k
[+] Reading header
[+] Detecting header type
[+] Executable is a plain MACH-O image
[+] Opening /var/mobile/Containers/Data/Application/23C75F90-C42D-4F43-83D9-5DCCA36FE2D5/Document
[+] Copying the not encrypted start of the file
[+] Dumping the decrypted data into the file
[+] Copying the not encrypted remainder of the file
[+] Setting the LC_ENCRYPTION_INFO->cryptid to 0 at offset bd0
[+] Closing original file
[+] Closing dump file
[*] This mach-o file decrypted done.
[+] dump macho file at:/var/mobile/Containers/Data/Application/23C75F90-C42D-4F43-83D9-5DCCA36FE2

...
[*] Developed By xia0@2019
```

patcher [2019/10/17]

runtime patch instrument in lldb

```
// -a patch_address -i patch_instrument{nop/ret/mov0/mov1} -s instrument_count
(lldb) patcher -a 0x0000000100233a18 -i nop -s 8
[*] start patch text at address:0x100233a18 size:8 to ins:"nop" and data:0x1f, 0x20, 0x03, 0xd5
[*] make ins data:
{0x1f, 0x20, 0x03, 0xd5 ,0x1f, 0x20, 0x03, 0xd5 ,0x1f, 0x20, 0x03, 0xd5 ,0x1f, 0x20, 0x03, 0xd5 ,
[+] patch done
[x] power by xia0@2019
(lldb) x/12i 0x0000000100233a18
    0x100233a18: 0xd503201f   nop
    0x100233a1c: 0xd503201f   nop
    0x100233a20: 0xd503201f   nop
    0x100233a24: 0xd503201f   nop
    0x100233a28: 0xd503201f   nop
    0x100233a2c: 0xd503201f   nop
    0x100233a30: 0xd503201f   nop
    0x100233a34: 0xd503201f   nop
    0x100233a38: 0xf941ac14   ldr    x20, [x0, #0x358]
    0x100233a3c: 0xf9419c15   ldr    x21, [x0, #0x338]
    0x100233a40: 0xf941a400   ldr    x0, [x0, #0x348]
    0x100233a44: 0xf9400008   ldr    x8, [x0]

// 2019-10-27 update: -i option can receive raw instrument data like: "{0x20, 0x00, 0x80, 0xd2}"
(lldb) patcher -a 0x183a40fd8 -i "{0x20, 0x00, 0x80, 0xd2}"
[*] detect you manual set ins data:{0x20, 0x00, 0x80, 0xd2}
[*] start patch text at address:0x183a40fd8 size:1 to ins data:{0x20, 0x00, 0x80, 0xd2}
[x] power by xia0@2019
(lldb) x/12i $pc
->  0x183a40fd8: 0xd2800020   mov    x0, #0x1
    0x183a40fdc: 0x928003f0   mov    x16, #-0x20
    0x183a40fe0: 0xd4001001   svc    #0x80
    0x183a40fe4: 0xd65f03c0   ret
    0x183a40fe8: 0x92800410   mov    x16, #-0x21
    0x183a40fec: 0xd4001001   svc    #0x80
    0x183a40ff0: 0xd65f03c0   ret
    0x183a40ff4: 0x92800430   mov    x16, #-0x22
    0x183a40ff8: 0xd4001001   svc    #0x80
    0x183a40ffc: 0xd65f03c0   ret
    0x183a41000: 0x92800450   mov    x16, #-0x23
    0x183a41004: 0xd4001001   svc    #0x80
```

TODO

- Anti-anti-debug：bypass anti debug in lldb （done at 2019/09/11)
- OCHOOK：hook ObjectC function in lldb
- NetworkLog：minitor network info
- UI Debug：some useful command for UI debug
- xbr: set breakpoint at address of methods of class（done at 2019/08/11)
- traceOC: trace ObjectC call by inlinehook msg_send stub code
- ...

Update

- [2019/07/04] Update for **sbt -x / xutil** : xutil cmd and sbt -x to disable color output in Xcode
- [2019/07/21] Update for **choose** : lldb's choose command version of cycript's choose command
- [2019/08/07] Fix critical bugs in **choose** : Fix critical bugs
- [2019/08/11] Update for **xbr** : `xbr className` can set breakpoint at adresses of all methods of class
- [2019/08/13] New **debugme**: kill anti debug in lldb
- [2019/08/20] New **info**: get info of address/function/module and so on
- [2019/09/11] **debugme** update: hook ptrace and inlinehook svc ins done.
- [2019/09/22] new **dumpdecrypted**: dump macho image in lldb
- [2019/09/27] **dumpdecrypted** update: can dump all image in app dir
- [2019/10/17] new **patcher** :runtime patch instrument in lldb
- [2022/04/18] add xivars/xmethods/xprotocol to enable dump class when ivars/methods not support like in macOS or iOS system process.

Document

- About_this_project
- sbt command for frida

Credits

- http://blog.imjun.net/posts/restore-symbol-of-iOS-app/ thanks to the ida_block_json.py script
- https://github.com/DerekSelander/LLDB Special thanks to DerekSelander's LLDB provide the code framework
- https://lldb.llvm.org/tutorial.html
- https://github.com/hankbao/Cycript/blob/bb99d698a27487af679f8c04c334d4ea840aea7a/ObjectiveC/Library.mm choose command in cycript
- https://opensource.apple.com/source/lldb/lldb-179.1/examples/darwin/heap_find/heap.py.auto.html

  Apple lldb opensource about heap

- https://blog.0xbbc.com/2015/07/%e6%8a%bd%e7%a6%bbcycript%e7%9a%84choose%e5%8a%9f%e8%83%bd/