THE SH3LLCOD3R'S BLOG

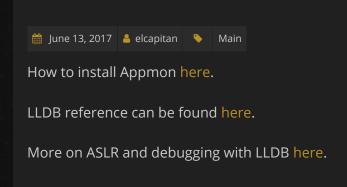
HOME CONTACT CTF WALKTHROUGHS EXPLOIT DEVELOPMENT MOBILE SECURITY NETWORK

SECURITYTUBE - LINUX ASSEMBLY EXPERT 32-BIT SECURITYTUBE - OFFENSIVE IOT EXPLOITATION SECURITYTUBE EXAMS

CISCO EMBEDDED

Home / Main / How to debug an iOS application with Appmon and LLDB

How to debug an iOS application with Appmon and LLDB



This blog is dedicated to my research and experimentation on ethical hacking. The methods and techniques published on this site should not be used to do illegal things.

If we install Appmon, then we have an LLDB window, where we can debug the IOS application. Debugging is one way to examine/manipulate an application.

The best way to debug the application is to load it into IdaPro first. Then let us find an address, where we want to set breakpoint. For this blog post, I am using the <code>DamnVulnerablelosApp</code>. Here is the <code>[JailbreakDetectionVC jailbreakTest1Tapped:]</code> method in IdaPro.

```
text:000000010001974C :
text - 8888888818881974C
text<mark>:00000010001974C</mark>; JailbreakDetectionUC - (void)jailbreakTest1Tapped:(id)
text: 88888888188819746 :
                           Attributes: bp-based frame
text:000000010001974C
__text:000000010001974C ; void __cdecl -[JailbreakDetectionUC jailbreakTest1Tapped:](struct JailbreakDetectionUC *self, SEL, id)
text:000000010001974C
                         __JailbreakDetectionVC_jailbreakTest1Tapped_
                                                                      ; DATA XREF: __objc_const:00000001001FA1A81o
text:0000000100019740
text:0000000100019740
text:000000010001974C var 10
                                           = -8x18
text:000000010001974C var_s0
                                           = 0
text: AAAAAAAA1AAA1974C
text: AAAAAAAA1AAA1974C
                                           STP
                                                            X20, X19, [SP,#-0x10+var_10]!
text:0000000100019750
                                                            X29, X30, [SP,#0x10+var_s0]
                                           STP
text:0000000100019754
                                           ADD
                                                            X29, SP, #0x10
                                                            X8, #classRef_DamnUulnerableAppUtilities@PAGE
X19, [X8,#classRef_DamnUulnerableAppUtilities@PAGEOFF]
X8, #selRef_isJailbroken@PAGE
                                           ADRP
text:0000000100019758
text - 0000000100019750
                                           LDR
text - 0000000100019760
                                           ADRP
text:0000000100019764
                                           NNP
                                                            X1, [X8,#selRef_isJailbroken@PAGEOFF]
text - 0000000100019768
                                           IDR
text:000000010001976C
                                                            _objc_msgSend
X2. X0
                                           BL
text:0000000100019770
                                           MOU
text:0000000100019774
                                                            X8, #selRef_showAlertForJailbreakTestIsJailbroken_@PAGE
                                           ADRP
text:0000000100019778
                                           NNP
text:0000000100019770
                                          I DR
                                                            X1, [X8, #selRef_showAlertForJailbreakTestIsJailbroken_@PAGEOFF]
                                                            X0, X19
text:0000000100019780
                                           MOU
                                                            X29, X30, [SP,#0x10+var s0]
text:0000000100019784
                                          LDP
text:0000000100019788
                                          LDP
                                                            X20, X19, [SP+0x10+var_10],#0x20
__text:000000010001978C B __objc_msgSend
_text:000000010001978C ; End of function -[JailbreakDetectionUC jailbreakTest1Tapped:]
text:000000010001978C
```

The address is 0x10001974C. However the real address in memory is different, because the whole TEXT segment is shifted with a certain value (ASLR). In order to determine the shift value, execute the following command in LLDB:

(IIdb) image dump sections <APP_NAME>

I do not take responsibility for acts of other people.

RECENT POSTS

Androguard usage

How to debug an iOS application with Appmon and LLDB

OWASP Uncrackable – Android Level3

OWASP Uncrackable – Android Level2

How to install Appmon and Frida on a Mac

CATEGORIES

Android (5)

Fusion (2)

IoT (13)

Main (3)

Mobile (6)

Protostar (24)

SLAE32 (8)

The TEXT segment starts at 0x1000a8000, so the shift value is 0xa8000. We have to add this value to every address, that comes from IdaPro. The following command disassemble a few bytes from the passed address:

VulnServer (6)

Windows Reverse Shell (2)

(IIdb) disassemble -start-address 0xa8000+0x10001974

```
(lldb) disassemble --start-address 0xa8000+0x10001974c

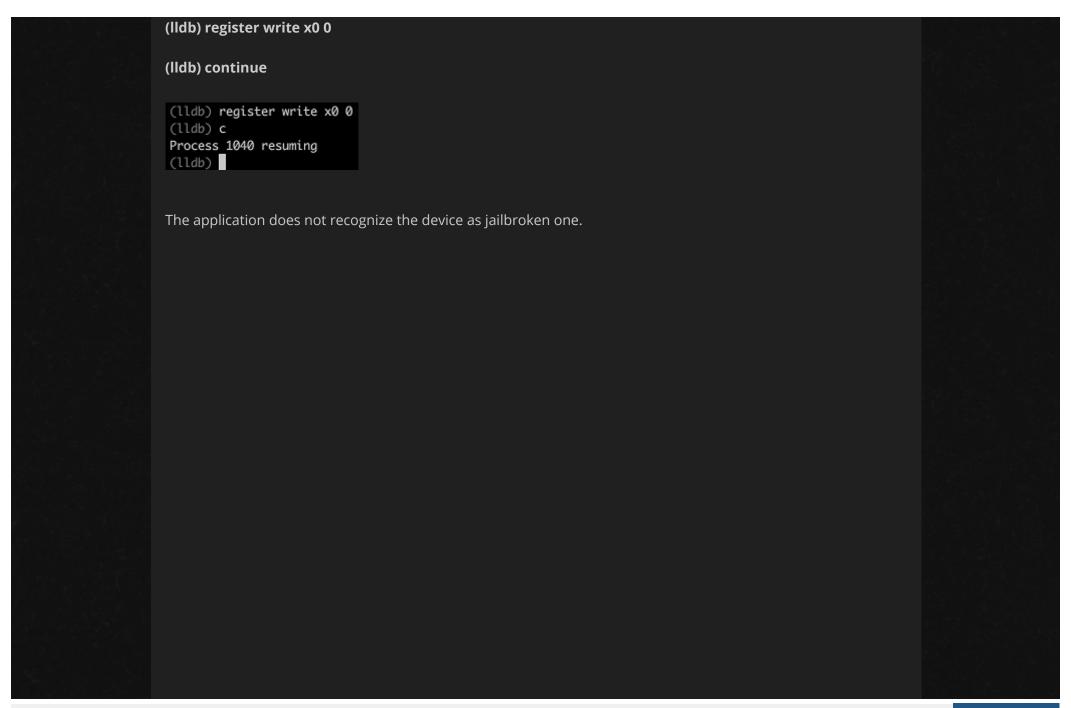
DamnVulnerableIOSApp`__lldb_unnamed_symbol578$$DamnVulnerableIOSApp:
0x1000c174c <+0>: stp x20, x19, [sp, #-0x20]!
0x1000c1750 <+4>: stp x29, x30, [sp, #0x10]
0x1000c1754 <+8>: add x29, sp, #0x10 ; =0x10
0x1000c1758 <+12>: adrp x8, 510
0x1000c175c <+16>: ldr x19, [x8, #0xcf8]
0x1000c1760 <+20>: adrp x8, 507
0x1000c1764 <+24>: nop
0x1000c1768 <+28>: ldr x1, [x8, #0xdc0]
```

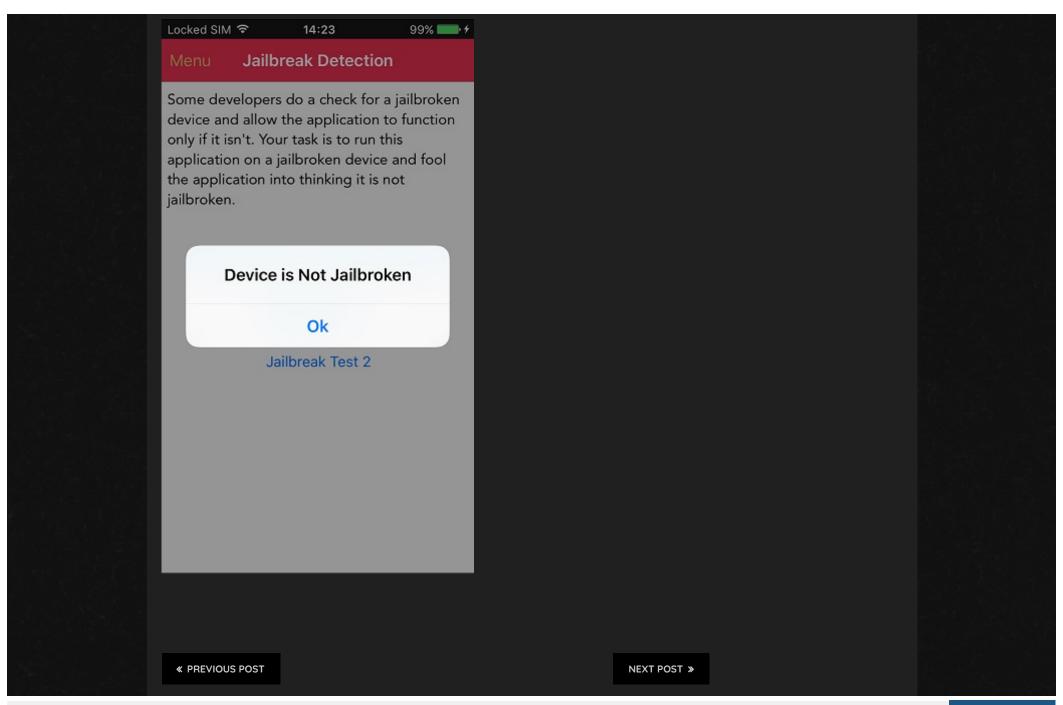
I set a breakpoint at 0x100019770.

(lldb) breakpoint set -a 0xa8000+0x100019770

This is the next instruction after the [JailbreakDetectionVC jailbreakTest1Tapped:] call in jailbreakTestTapped1 method. The return value is in X0 register. 0x1 means, that the device is jailbroken.

This can be modified with ...





Copyright © 2019, The sh3llc0d3r's blog. Proudly powered by

WordPress. Blackoot design by Iceable Themes.

WordPress. Blackoot design by Iceable Themes.

SecurityTube – Linux Assembly Expert 32-bit

SecurityTube – Offensive IoT Exploitation SecurityTube exams

CISCO Embedded