iOS平台的应用程序调试与分析(openssh + gdb)

转载

标 题: 【原创】iOS平台的应用程序调试与分析
作 者: zhuliang
时 间: 2013-04-02,17:09:40
链 接: http://bbs.pediy.com/showthread.php?t=167398

iOS平台的应用程序调试与分析

作者：zhuliang

本文阐述如何在iOS平台上对应用程序进行调试与分析，旨在指导新手分析iOS程序，高手请无视。内容包括软件硬件的准备、代码的解密、符号信息的获取、用gdb调试等，最后以京东LeBook为例子进行演示。

1.为什么要进行调试与分析

研究iOS程序有很多用处，比如：
找bug或者漏洞，想知道某程序有没有漏洞或者bug。
某程序能实现某功能，我想知道如何实现，如ios6发短信功能，还有比较时髦的iPhone5通话录音功能。
对iOS程序进行DIY扩展功能，比如曾半仙前辈的作弊插件http://bbs.pediy.com/showthread.php?t=163435，就是一个例子。2011年的时候我也实现了类似的功能，做了一个iPhone版的“欢乐斗地主”的记牌器功能。有朋友问到如何实现的，主要就是hook了recv函数，从封包得到想要的信息，经过处理得知54张牌还剩什么牌。
总的来说对iOS程序进行调试与分析的目的和调试与分析Windows平台程序的目的是一样的。

2. 调试与分析的准备工作

iOS的封闭性极大地增大研究它的难度。为使系统更加安全，iOS引入了很多安全机制；如代码加密，使得分析前先要解密代码，SandBox（沙盒），code signing（代码签名），使得不能运行未签名的代码。这些安全机制使得研究它的难度比Linux/Windows要大。
尽管这样，并不能阻挡我们前进的脚步。

必要的硬件准备：一个越狱的设备，最好是iPhone

越狱是必要的，因为只有越狱才能运行非AppStore上的软件。

必要的软件准备：在Cydia里安装下面的软件

1.OpenSSH，OpenSSH是Linux下常用的服务，装上后设备可充当服务SSH服务端
2.GNU Debugger（gdb）调试工具
3.adv-cmds（ps命令）
4. darwin cc tools （otools）
5. Link Identity Editor （ldid）

PC端安装SSH Secure Shell Client或putty等SSH客户端软件，以便能通过SSH建立到设备的连接。
Ida pro，该软件搞过逆向的都熟悉，静态分析功能很强大，特别是大于或等于6.2的版本，能识别Objective-C结构，能把Objective-C函数名显示如下图：

Hopper Disassembler，该软件是刚出来没多久的，我还没用过，据说功能也很强大。不管怎么说ida pro是太贵了，一般个人很少买得起，这时候Hopper Disassembler是另一个选择。最后是class-dump-z，在没有上面两个工具的情况下用它能获得符号信息。

有了软硬件的准备，便可以开始调试了。
1.PC端运行SSH客户端，连接到设备，输入用户名：root，默认密码：alpine，第一次连接上去后建议用passwd命令改掉默认密码，以防黑客通过局域网入侵。
2.在设备上运行想调试的程序，如iRead，运行命令ps –ax可以看到所有运行的进程id，gdb –p pid这样就可以调试指定进程（gdb –p 10110）。
另外，可以通过otool –l | grep crypt 输出可知道已加密代码的位置。有这些准备后，我们可以先运行下面命令测试一下。

```
SSH Secure Shell 3.2.9 (Build 283)
Copyright (c) 2000-2003 SSH Communications Security Corp - http://www.ssh.com/
This copy of SSH Secure Shell is a non-commercial version.
This version does not include PKI and PKCS #11 functionality.
MMs-iPod:~ root# passwd
Changing password for root.
New password:
Retype new password:
MMs-iPod:~ root# ps -ax
  PID TTY          TIME CMD
……（省略部分无关记录）
12416 ??         0:02.52 /var/mobile/Applications/4DFD17D1-39AC-4F10-8AB8-
3A4CB99E9E77/iRead.app/iRead
……（省略部分无关记录）
MMs-iPod:~ root# gdb -p 12416
/usr/bin/gdb: line 55: awk: command not found
warning: unrecognized host cpusubtype , defaulting to host==armv7.
GNU gdb 6.3.50-20050815 (Apple version gdb-1708 + reverse.put.as patches v0.4) (Mon Apr
16 00:53:47 UTC 2012)
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "arm-apple-darwin".
/private/var/root/12416: No such file or directory
Attaching to process 12416.
Reading symbols for shared libraries . done
Reading symbols for shared libraries
.....
done
Reading symbols for shared libraries + done
0x39556eb4 in mach_msg_trap ()
(gdb) quit
The program is running.  Quit anyway (and detach it)? (y or n) y
Detaching from process 12416.
MMs-iPod:~ root# otool -l /var/mobile/Applications/4DFD17D1-39AC-4F10-8AB8-
3A4CB99E9E77/iRead.app/iRead | grep crypt
cryptoff  8192
cryptsize 8601600
cryptid   1
```

3.解密代码并用IDA和gdb分析

按上文的步骤便能通过gdb来调试iOS平台的应用程序，但是在gdb下看代码是很不方便的，肯定没有在我们熟悉的ida pro上看得舒服。为了能在ida上进行静态分析，需要先去除DRM保护，iOS程序往往是以加密的状态存在于正版app的ipa文件中的，只有去除掉DRM保护（俗称ipa破解），才能用ida来进行静态分析。也许有人会说，用如维享应用汇、91助手、PP助手等

工具下载下来的ipa直接拖到ida里就能分析，需知道这样的ipa是经过破解了的，不是正版的ipa。

3.1DRM破解(代码解密)基本原理

1.定位到待破解软件在var/mobile/Applications下所在位置
2.复制软件目录下所有的文件到一个临时目录
3.用工具otool分析程序主文件，找出cryptsize和cryptid
4.运行待破解程序，用gdb附加到目标进程
5.因为程序已经运行，所以此时是解密状态。dump当前未加密的内存
6.将dunp结果输出到一个bin文件，退出gdb调试程序
7.将主程序文件的cryptid字段改为0，改加密状态为未加密
8.将dump出来的bin文件中的未加密内存的内容覆盖到原主程序文件的相应位置
9.对主程序签名
10.删除一些垃圾文件
11.打包IPA

下面以京东LeBook为例子，用gdb手工解密，具体操作如下：

先通过SSH客户端连接到iOS设备上，切换到对应程序的目录。依次输入下面的命令：

```
root# cd /var/mobile/Applications/4DFD17D1-39AC-4F10-8AB8-3A4CB99E9E77/iRead.app/
root# gdb -e ./iRead
/usr/bin/gdb: line 55: awk: command not found
warning: unrecognized host cpusubtype , defaulting to host==armv7.
/usr/bin/gdb: line 136: file: command not found
/usr/bin/gdb: line 171: file: command not found
/usr/bin/gdb: line 171: awk: command not found
GNU gdb 6.3.50-20050815 (Apple version gdb-1708 + reverse.put.as patches v0.4) (Mon Apr 16 00:53:47 UTC 2012)
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "arm-apple-darwin".Reading symbols for shared libraries .. done
(gdb) set sharedlibrary load-rules ".*" ".*" none
(gdb) set inferior-auto-start-dyld off
(gdb) set sharedlibrary preload-libraries off
(gdb) rb doModInitFunctions
Breakpoint 1 at 0x2fe0cece
<function, no debug info>
__dyld__ZN16ImageLoaderMachO18doModInitFunctionsERKN11ImageLoader11LinkContextE;
(gdb) r
Starting program: /private/var/mobile/Applications/4DFD17D1-39AC-4F10-8AB8-3A4CB99E9E77/iRead.app/iRead
Breakpoint 1, 0x2fe0cece in __dyld__ZN16ImageLoaderMachO18doModInitFunctionsERKN11ImageLoader11LinkContextE()
(gdb)  dump memory dec.bin 0x3000 0x837000
```

该命令是把已解密的内存dump到dec.bin文件中，0x3000为开始地址，0x837000为结束地址，为什么是这两个地址呢？

从上文otool的输出知道cryptoff=8192(0x2000)，cryptsize=8601600(0x834000)，而当地址空间被加载的时候，开始地址是0x1000，而不是0x000。0x2000+0x1000等于0x3000，所以开始地址为0x3000，结束地址为开始地址加上被加密的映像大小，为0x3000+0x834000=0x837000。

```
(gdb) kill
Kill the program being debugged? (y or n) y
(gdb) q
```

到此，我们得到解密后代码，只要用它来替换掉原来被加密的代码即可，
root#cp iRead iRead.bak
root# dd seek=8192 bs=1 conv=notrunc if=./dec.bin of=./iRead
上面这条命令在iOS设备上执行完要大大几分钟，用WinHex来进行替换会快很多，从文件偏移8192(0x2000)开始替换，大小为加密块大小0x834000字节。在可执行程序有两个或多于两个Fat header的情况下，开始替换的偏移地址还要加上Architecture Offset，Architecture Offset可以用命令“otool –f ./iRead”得到。可参考《Hacking and Securing iOS Applications》第七章，解释得很详细具体。另喜欢看中文的朋友可以参考论坛里肖兄(Claud)的文章http://bbs.pediy.com/showthread.php?t=152843 ，只是要注意的是，要在doModInitFunctions下断，而最好不要在程序入口点下断。因为如果断在程序入口点，动态库的入口函数已被调用，有些程序会被inline hook，有可能造成dump出来的内容是错的。
接下来把cryptid由1改为0，表示不加密。再用“ldid –S iRead”重新签一下。而cryptid可以这样来定位到，从刚才替换区域开始处往上拉，一直拉到能看到很多.dylib的路径，放慢速度仔细看，在最后一个dylib的地方，再往上大概0x28的地方有个01，把它改为00就行，如果不是很确定，可以通过“otool -l iRead | grep crypt”来看cryptid是否变为0来确定。
至此完成代码解密，再打包成ipa文件便完成app的破解。
另外还可以在Cydia里装上Crackulous软件来破解，Crackulous软件是全自动化的图形界面的破解程序。只要点几下就能生成破解后的ipa文件，方便易用快捷。但是也有个缺点，就是有些软件不能用它来破解。相对于第二个是全自动化的方法，第一个方法虽然比较麻烦，但便于理解过程。

3.2获取符号信息便于调试

获取到尽可能多的符号信息对分析调试有很大的帮助，就像在茫茫大海中航行的船找到了灯塔，不至于迷失方向，不知所措。有了足够多的符号，我们就可以迅速地定位目标程序的关键点，不会迷失在二进制的海洋中，因找不到方向而无从下手。
前文提到了，利用ida pro可以帮助我们获取符号，特别是大于或等于6.2的版本，能识别Objective-C结构，能把Objective-C函数全都识别出来，如下图，为本人利用ida 6.4打开京东LeBook-1.0.4的截图。

可以从左边Functions window看出来，没有一个函数是以“sub_”开头的，也就说所有的函数都识别出来了。
没有ida pro的朋友可以使用Hopper Disassembler，也能把Objective-C函数识别出来。在版权方要求删除之前，可以从看雪论坛下载到。据说该软件很好用，但因本人习惯了ida pro没使用过它，不便进行过多的评论。
上面说到的两个软件都是商业软件，不是人人都愿意买或买得起，这时还可以利用class-dump-z获取符号信息，建议不能使用盗版、不愿意使用盗版而又没钱买正版的朋友使用，它的命令行如下：

```
class-dump-z.exe -u armv7 -A -a iRead  >  iRead.txt
   class-dump-z.exe -u armv7 -A -a iRead -H -o C:\
```

其中-u armv7这个参数比较有用，而class-dump和class-dump-x 不支持-u armv7参数。两年前，ida pro还没有更新到6.2，Hopper Disassembler还没有问世，class-dump-z是唯一的选择。本人就是使用它来获取符号信息的。
上面的命令行生成的iRead.txt文件的部分内容如下图：

可以很清楚地看到类叫什么名，是如何定义的，有哪些成员，继承了哪些接口，还有函数的实现地址。这些信息都是对逆向很有用的信息。

3.3用gdb进行调试

有了足够的符号信息，通过静态分析，我们可以初步猜出程序的逻辑，哪里是关键点，再通过动态跟踪来证实猜想，结合上面给出的调试的步骤，下面开始用gdb进行调试：

```
ps –au                  注意输出信息中目标进程的pid
gdb –p pid              从上面的输出信息得到进程pid
info sh                 注意目标的基地址，记为base
break *(0xAABBCC+base)  0xAABBCC为你想下断点的地址
```

```
display /x $r0
display / $pc | $cpsr.t
ni
```
在实际的环境，命令行如下：
```
MMs-iPod:~ root# ps -ax
  PID TTY           TIME CMD
……（省略部分无关记录）
12416 ??         0:02.52 /var/mobile/Applications/4DFD17D1-39AC-4F10-8AB8-3A4CB99E9E77/iRead.app/iRead
……（省略部分无关记录）
MMs-iPod:~ root# gdb -p 12416
/usr/bin/gdb: line 55: awk: command not found
warning: unrecognized host cpusubtype , defaulting to host==armv7.
GNU gdb 6.3.50-20050815 (Apple version gdb-1708 + reverse.put.as patches v0.4) (Mon Apr 16 00:53:47 UTC 2012)
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "arm-apple-darwin".
/private/var/root/12416: No such file or directory
Attaching to process 12416.
Reading symbols for shared libraries . done
Reading symbols for shared libraries
.....
done
Reading symbols for shared libraries + done
0x39556eb4 in mach_msg_trap ()
(gdb) info sh
info sh
The DYLD shared library state has not yet been initialized.
                                 Requested State Current State
Num Basename              Type Address        Reason | | Source
  | |                     | |                  | | | |
  1 iRead                   - 0x26000          exec C C /private/var/mobile/Applications/4DFD17D1-39AC-4F10-8AB8-3A4CB99E9E77/iRead.app/iRead at 0x26000 (offset 0x25000)
                                      (objfile is) [memory object "/private/var/mobile/Applications/4DFD17D1-39AC-4F10-8AB8-3A4CB99E9E77/iRead.app/iRead" at 0x26000]
```
这条命令info sh很重要，特别强调一下。很多网友在QQ群小窗问我这个问题，ida里看到某个关键函数的地址为0x1C4038，用gdb成功附加进程后，break *0x1C4038下断点，老是断不下来，为什么呢。出现这样的情况是因为断点没下对，程序在编译时启用了PIE(Position Independent Executables)，该安全机制是从ios4.3开始引进的，和Windows下的ASLR类似，这时候下断点的地址要加上基地址才对。注意到括号里的“(offset 0x25000)”，故这样下断(gdb) break *(0x25000+0x1C4038)
```
Breakpoint 1 at 0x1e9038
```
如果不肯定断点下得对不对，可用下面的gdb命令来确认，
```
(gdb) x/10i (0x25000+0x1C4038)+1
0x1e9039:  b5 03                   push   {r4, r5, r6, r7, lr}
0x1e903b:  af 4d                   add    r7, sp, #12
0x1e903d:  f8 04 8d 84               str.w  r8, [sp, #-4]!
```

```
0x1e9041:  b0 05                    sub    sp, #16
0x1e9043:  46 00                    mov    r5, r0
0x1e9045:  20 0c                    movs   r0, #0
0x1e9047:  46 03                    mov    r4, r1
0x1e9049:  90 02                    str    r0, [sp, #12]
0x1e904b:  90 00                    str    r0, [sp, #8]
0x1e904d:  2a 18                    cmp    r2, #0
```

看到gdb反汇编出来的代码和ida里显示的一样，继续运行

```
(gdb) c
Continuing.
```

程序运行后，在iOS设备上点击一本书畅读，断点被触发中断下来，这时可以用如下display命令显示和ni或si一步一步跟进去分析。

```
Breakpoint 1, 0x001e9038 in dyld_stub_pthread_key_create ()
(gdb) display /x $r0
1: /x $r0 = 0x1fdae580
(gdb) display /i $pc | $cpsr.t
2: x/i $pc | $cpsr.t  0x1e9038:  f0 b5  push   {r4, r5, r6, r7, lr}
```

补充一点，如果觉得每次下断的地址都要加上基地址比较麻烦，可以像我们在Windows下调试启用了ASLR功能的程序那样操作，通过把PE头中表示ASLR的bit清零来禁用ASLR功能。对于iOS系统，mach_header是如下定义的，

```
struct mach_header
{
uint32_t magic;
cpu_type_t cputype;
cpu_subtype_t cpusubtype;
uint32_t filetype;
uint32_t ncmds;
uint32_t sizeofcmds;
uint32_t flags;
};
```

根据头文件里的“#define MH_PIE 0x200000”，我们只要把flags里bit21清零即可。对程序进行了改动，重新签名一下“ldid –S iRead”，再重新运行，就会发现程序基地址不会再变化了。

还可以用removePIE(https://github.com/peterfillmore/removePIE)程序来帮我们做相同的工作。可以从代码看出工作原理是完全一样的。

最后，如何觉得gdb界面不够友好，用它来调试得不爽，还可以参考论坛obaby的帖子(http://bbs.pediy.com/showthread.php?t=138472)，用ida pro来进行图形化界面的调试。

4. 京东LeBook不严谨逻辑的绕过的演示

4.1背景交待

京东商城(www.360buy.com)上10元的畅读卡，可以下载京东LeBook软件后，用该软件下载1000本电子书来阅读，但是有两个限制，一是要联网才能看，二是只能在一个月内看。还有下载的电子书以加密形式存在。

经过分析发现京东LeBook验证逻辑不是很严谨，既然解密pdf所需要的certificate都设计为从网络传送过来了，是不是畅读却简单地通过本地数据库来标识，这使得上面两个限制可以去掉。通过替换sqlite3数据库(jdreader.db)即可实现。

4.2绕过步骤

解压附件中的injectHook1.zip（会员区有bin）到以下目录，/Library/MobileSubstrate/DynamicLibraries，并执行命令：chmod +x injectHook1.dylib给它加上可执行属性。该dylib对iRead进行了两处Hook，一处是让iRead认为设备的udid为“4A696E67646F6E67283336306275792E636F6D29”(下文的random就是由它计算得到的)，另一处是把解密所需要的certificate写到数据库。

注册A、B两账号，A账号出10元买一个畅读月卡，用A账号从设备登录然后注销；B账号从设备登录然后注销，最后用A再次登录。

在iOS设备上下载你所想要的电子书(按照游戏规则最多可下载1000本)，在设备上点畅读，每本书打开一次。这样才能确保解密所需要的certificate存到数据库里。
结束京东LeBook程序，把它的数据库文件复制到电脑上，路径为：/Library/Application Support/Root/jdreader.db，对该数据库文件用SQLite Database Browser程序(http://sqlitebrowser.sourceforge.net/)执行下面几条SQL命令

update LocalBook set userid=2 where isJoyRead=1 and hasdownload=1
update LocalBook set isJoyRead=0 where isJoyRead=1 and hasdownload=1
update User set random="0001X2NNi+0gEX6kKfArTVWJCBZnWjRBeWFpWmZBZHpiYVQ=" where id=2
把数据库文件复制回去，再次启动京东LeBook程序，接下来

4.3 让我们一起来见证奇迹的时刻

可以看到，畅读标志没有了，断网情况点击能顺利阅读了，两个限制成功地去掉了。

参考文献：

1. 《Hacking and Securing iOS Applications》
2. https://developer.apple.com/library/...reference.html

最后

感谢cd-team的各位朋友，DarkMage(dm557)，Windknown，Daniel，s1mbily，Danniez等。
感谢kanxue，Albert_liuwei等各位朋友！