**用dumpdecrypted给App砸壳**

在《iOS应用逆向工程》4.6.2节中，我们曾推荐使用iPhoneCake源的AppCrackr 1.7版给App砸壳。这种方式简单粗暴，省时省力，但正是因为它过于方便有木有，导致几乎所有iDevice用户都可轻松上手，随便亵玩，所以不少用户都拿它来破解程序，而不是学习《iOS应用逆向工程》，简直可以说是婶可忍叔不可忍！这也导致了iOS越狱开发社区普遍认为这个软件助长了盗版的气焰，没有脱离低级趣味，对iPhoneCake源进行了强烈谴责，责令其限期整改。

迫于压力，iPhoneCake在前段时间将AppCrackr下架，而书中提到的xsellize源中的AppCrackr则是1.5旧版，已不能在高级系统中使用。所以，为了响应业界反盗版的呼声，提倡毛主席“自己动手丰衣足食”的革命精神，让“砸壳”这件事恢复单纯的研究目的，在这里我们会使用更偏geek一些的dumpdecrypted方式来给App砸壳，不再推荐AppCrackr、Clutch、Crackulous等纯UI方式。由于dumpdecrypted刚经历过一次大升级，目前网上可以找到的使用教程均已过期，所以这里我们手把手带大家进行一次完整的“砸壳 + class-dump”，请大家准备板凳瓜子汽水，开始围观。如果能对着电脑，边看边做，善莫大焉！楼猪才疏学浅，如有纰漏，敬请斧正，洗耳恭听，污言秽语，免开尊口，感谢支持！

**1. 下载dumpdecrypted的源码**

源码下载地址是“<https://github.com/stefanesser/dumpdecrypted/archive/master.zip>”，下载后请将其解压至你习惯的位置，例如楼猪为“/Users/snakeninny/Code/”，解压后生成“/Users/snakeninny/Code/dumpdecrypted-master/”。

**2. 确认你的iDevice系统版本**

打开“设置”，点击“通用”……喂！这个还要我教吗？像话吗！自己解决！今天是2014年2月26日，我小声问一句，看到这里的朋友，你的iDevice上跑的应该是是iOS 5/6/7吧？没有3和4的了吧？有的请举手我看一下，好的，没有。

**3. 修改Makefile中的SDK字段**

dumpdecrypted必须使用与iOS版本相同的SDK版本编译，才能正常工作。打开“终端（Terminal）”，输入

xcrun --sdk iphoneos --show-sdk-path

查看输出，看看目前的默认SDK是什么版本，例如楼猪的是“/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS7.0.sdk”。这里分2种情况：

**i.** 如果你使用的iOS版本同以上输出的SDK版本符合（例如楼猪使用的正是iOS 7.0.4），则无需对dumpdecrypted工程中的文件作任何改动；

**ii.** 如果你使用的iOS版本同以上输出的SDK版本不符（例如楼猪的另一台iPhone 4S使用的是iOS 6.1.3，iPhone 4使用的是iOS 5.0.1，请问有谁想同土豪楼猪交朋友？仅限女生，请自拍一张素颜照片并@iOS应用逆向工程，择优录取哦亲），则需要更改Makefile中的 SDK=`xcrun --sdk iphoneos --show-sdk-path`

为iOS版本相同的SDK版本。什么，你只有一个SDK？去<https://developer.apple.com/downloads/index.action>下载旧版本的Xcode，然后把里面的SDK提取出来。什么，不会提取？自己去Google一下，罚你今晚表睡觉了！提取出了旧版的SDK后，楼猪习惯把它们放在新版SDK相同的目录下，然后将

SDK=`xcrun --sdk iphoneos --show-sdk-path`

改为

SDK=/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS6.X.sdk

或

SDK=/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS5.X.sdk

即iOS版本需要与SDK版本相同。注意，5.1版SDK编译出的dylib是向下兼容的，可以用于iOS5.0，6.1版SDK同理。

**4. 修改Makefile中的GCC\_UNIVERSAL字段和dumpdecrypted.c**

如果你的iOS/SDK版本是7.0，则goto "这里继续："。如果并不是7.0，则需要修改Makefile中的GCC\_UNIVERSAL字段和dumpdecrypted.c。将Makefile中的

GCC\_UNIVERSAL=$(GCC\_BASE) -arch armv7 -arch armv7s -arch arm64

改为

GCC\_UNIVERSAL=$(GCC\_BASE) -arch armv7 -arch armv7s

再将dumpdecrypted.c第76行的

if (lc->cmd ==LC\_ENCRYPTION\_INFO || lc->cmd == LC\_ENCRYPTION\_INFO\_64)

改成

if(lc->cmd == LC\_ENCRYPTION\_INFO)

保存。这里继续：接着直接cd到“/Users/snakeninny/Code/dumpdecrypted-master/”下，然后输入“make”并回车，生成“/Users/snakeninny/Code/dumpdecrypted-master/dumpdecrypted.dylib”。

**5. 太麻烦？直接下载楼猪编译好的文件**

<https://github.com/iosre/Ready2Rock/blob/master/dumpdecrypted_5.dylib>

<https://github.com/iosre/Ready2Rock/blob/master/dumpdecrypted_6.dylib>

<https://github.com/iosre/Ready2Rock/blob/master/dumpdecrypted_7.dylib>

打开链接后，点击“Raw”即可下载。

**6. 将dumpdecrypted.dylib通过scp或iFunBox等工具拷贝至iOS**

楼猪习惯放在“/var/tmp/”下，这里大家随意就好。

**7. 定位倒霉催的App的可执行文件**

iFunBox，iFile，还是

1. find /var/mobile/Applications/ -name Victim

复制代码

？自由发挥，见仁见智，但我们的目标都是可执行文件，定位后记录其全路径，类似于“/var/mobile/Applications/XXXXXX/Victim.app/Victim”的形式。Today, I want to apologize to every...受害者……

**8. 开始砸壳**

cd到dumpdecrypted.dylib所在的目录，例如楼猪是“/var/tmp/”。然后输入

DYLD\_INSERT\_LIBRARIES=dumpdecrypted.dylib /var/mobile/Applications/XXXXXX/Victim.app/Victim

回车，坐等砸壳。如果出现如下报错：

dyld: could not load inserted library 'dumpdecrypted.dylib' because no suitable image found.  Did find:

dumpdecrypted.dylib: stat() failed with errno=1

Trace/BPT trap: 5

则说明因沙盒的原因，dumpdecrypted.dylib不能在当前目录下工作。以笔者的经验来看，将dumpdecrypted.dylib拷贝到App目录下的任一文件夹后，在那里重新运行砸壳命令即可，如下所示：

Touch5:/var/mobile/Applications/71C51582-9552-48DD-9811-B99C991F55BE/tmp root# DYLD\_INSERT\_LIBRARIES=dumpdecrypted.dylib /var/mobile/Applications/71C51582-9552-48DD-9811-B99C991F55BE/MicroMessenger.app/MicroMessenger

mach-o decryption dumper

DISCLAIMER: This tool is only meant for security research purposes, not for application crackers.

[+] detected 32bit ARM binary in memory.

[+] offset to cryptid found: @0x4abc(from 0x4000) = abc

[+] Found encrypted data at address 00004000 of length 23347200 bytes - type 1.

[+] Opening /private/var/mobile/Applications/71C51582-9552-48DD-9811-B99C991F55BE/MicroMessenger.app/MicroMessenger for reading.

[+] Reading header

[+] Detecting header type

[+] Executable is a plain MACH-O image

[+] Opening MicroMessenger.decrypted for writing.

[+] 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 abc

[+] Closing original file

[+] Closing dump file

如果出现如下报错：

dyld: could not load inserted library 'dumpdecrypted.dylib' because no suitable image found.  Did find:

dumpdecrypted.dylib: mach-o, but wrong architecture

Trace/BPT trap: 5

出现这种情况，一般是因为待砸壳的App处理器架构与dumpdecrypted.dylib的处理器架构不符。这句话是什么意思呢？我们都知道iPhone 4/4s/5的处理器是32位的（armv7/armv7/armv7s），iPhone 5s是64位的（arm64），因为32位和64位的指令长度不同，如果两者混合会完全乱掉，所以32/64位的dylib是不能被加载到64/32位进程中的。解决办法很简单，看看待砸壳的App是多少位的，然后重新编译一个相同位数的dumpdecrypted.dylib就好了。

当命令执行成功后，当前目录下会多出一个Victim.decrypted文件，表明这个App的壳被砸掉了。将它拷贝到OSX中，洗净备用。

**9. class-dump丫**

在崭新的21世纪，App的可执行文件一般都是fat binary，也就是说一个二进制文件里包含适合多个CPU架构使用的可执行文件，虽然CPU架构是向下兼容的（也就是说armv64可以执行armv7s的指令，反之则不行），但向下兼容一般会导致一部分性能的牺牲。这样如果一个binary既包含适用于armv7架构的可执行文件，又包含armv7s的，还包含arm64的，就意味着它可以在iPhone 4（armv7），4s（armv7），5（armv7s），5s（arm64）上都发挥100%的性能。虽然除了处女座以外的其他星座用户一般是感受不到这个性能的提升的~但是，机器比处女座还要处女座，它在执行一个fat binary时，会选择最适合自己CPU的那个可执行文件，其他的可执行文件其实是没有得到执行的。因此dumpdecrypted.dylib起作用的只有实际得到执行的那一个可执行文件，举个例子，如果Victim里含有armv7和armv7s这2种架构，而我们的操作机是一台iPhone5/4，那么dumpdecrypted砸掉的是armv7s/armv7那部分的壳，armv7/armv7s部分仍是有壳的。自然地，class-dump的作用对象必须是砸掉壳的binary，所以我们要在class-dump时指定目标，在本例中，就是

class-dump --arch armv7s Victim.decrypted

或

class-dump --arch armv7 Victim.decrypted

可以明白吧？没关系，慢一点，多看几遍，理一理，还是有问题的话，在下面留言啦