IDA□□□□□

2014-12-10 02:23:38 by admin 2089 4 ☆

/*安全经典 如厕必备 此广告位长期招租*/

/*大家好我是 @无所不能的魂大人 ，求关注求微博粉丝过百 */

0x0000 背景

事情大概是这个样子的，在那啥那啥那啥的冬季，一坨CTF联赛正在如火如荼地进行中，上个月刚刚送走了溅得我一脸热翔的hctf，这两天又迎来了卖萌又坑爹的SCTF，连智商还有IQ还有智商直接被虐到了地里，算了，不吐槽了，说正题，首先背景就是有个SCTF，里面第二道题是个100分的MISC题，为啥是MISC而不是RE我也不太清楚，反正就是大概有那么一个ELF文件，也就是linux下的可执行文件，是个贪吃蛇，大概题目的内容是这样（尼玛刚说去上面截个图结果发现网站关了）。就是有这么个贪吃蛇程序，提示说吃到30的样子就可以知道答案了，但是，怎么控制它呢？

这种问题其实对于我这种机智的少年而言太简单了，上面不是有提示么，UP-'8'、DOWN-'2'啥啥的，像我这种IQ过百的人玩儿这种游戏不是分分钟搞定的事儿么。于是我按照提示，DOWN是2对吧，2，2，2，222222222222222222222krelkur984375493irdjlsjiwury 32-03alshg49540啪！啪！啪！

（此处省略换键盘买键盘等一系列过程）

然后聪明的我很快就发现了原来这个程序的提示是骗人的，可是那该怎么办呢。虽然我的技术很渣，但也不至于啥都不懂，逆向，反编译这俩词儿我还是知道的，不过以前都是粗略地看过一些用W32ASM暴力破解软件和用OD搞个啥的一些文章，可这是ELF文件不是PE文件啊，用这些东西到底能不能行该咋整呢。经常听痛经大牛说什么IDA，于是默默打开某厂主页，发现果然它可以用。

好，工欲善其事，必先利其器，工具找到了，这道题不就做出一半了么！

0x0001 初识

好的我们经过千辛万苦总算找到了一款相对比较新的IDA，分享地址给出。http://pan.baidu.com/s/1mgza77i，友情提示安装完后请务必把里面那个小rar里的东西解压覆盖到程序目录，目测那个好像就是F5神器。

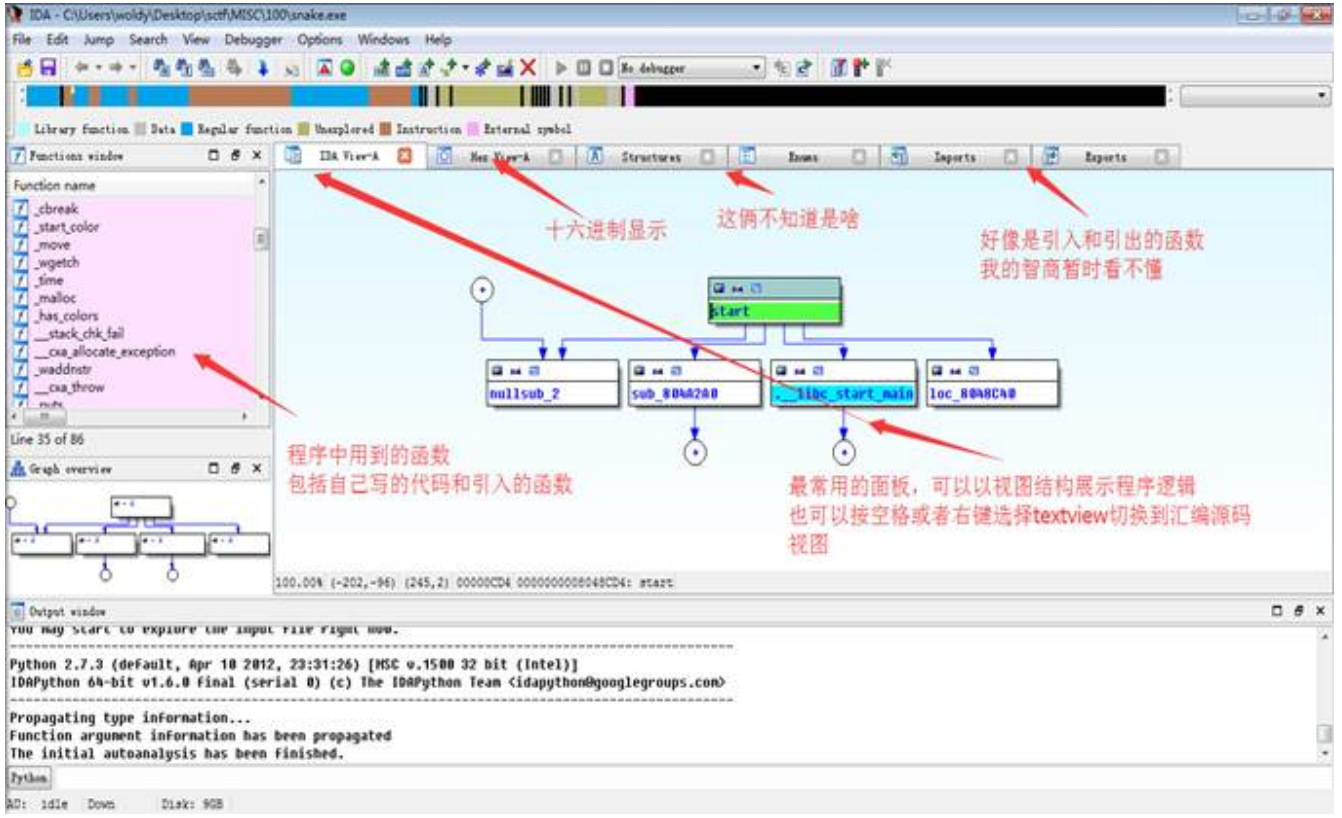
打开程序目录，很明显主程序就是这俩。

我是64位的win7，当然要用64位的那个了！哎，没办法，我就是如此的机智……

什么打开文件啥啥的我就不截图了，但凡智商比我差不了多少的人都会用，直接打开snake.exe（题目中的是snake-final.exe，痛经大神说那个有壳，然后现在的snake就是被脱壳的exe，为什么明明是个elf文件却要叫exe呢，我也不明白，我甚至不明白它为啥在xp下竟然可以运行）。

□□□□

新浪微博：@IDF实验室
投稿&建议：woldy@idf.cn



最后打开就是这个样子的，研究了好久才勉强研究明白主界面的这些东西大概都有啥用。然后接下来当然就是看代码了！

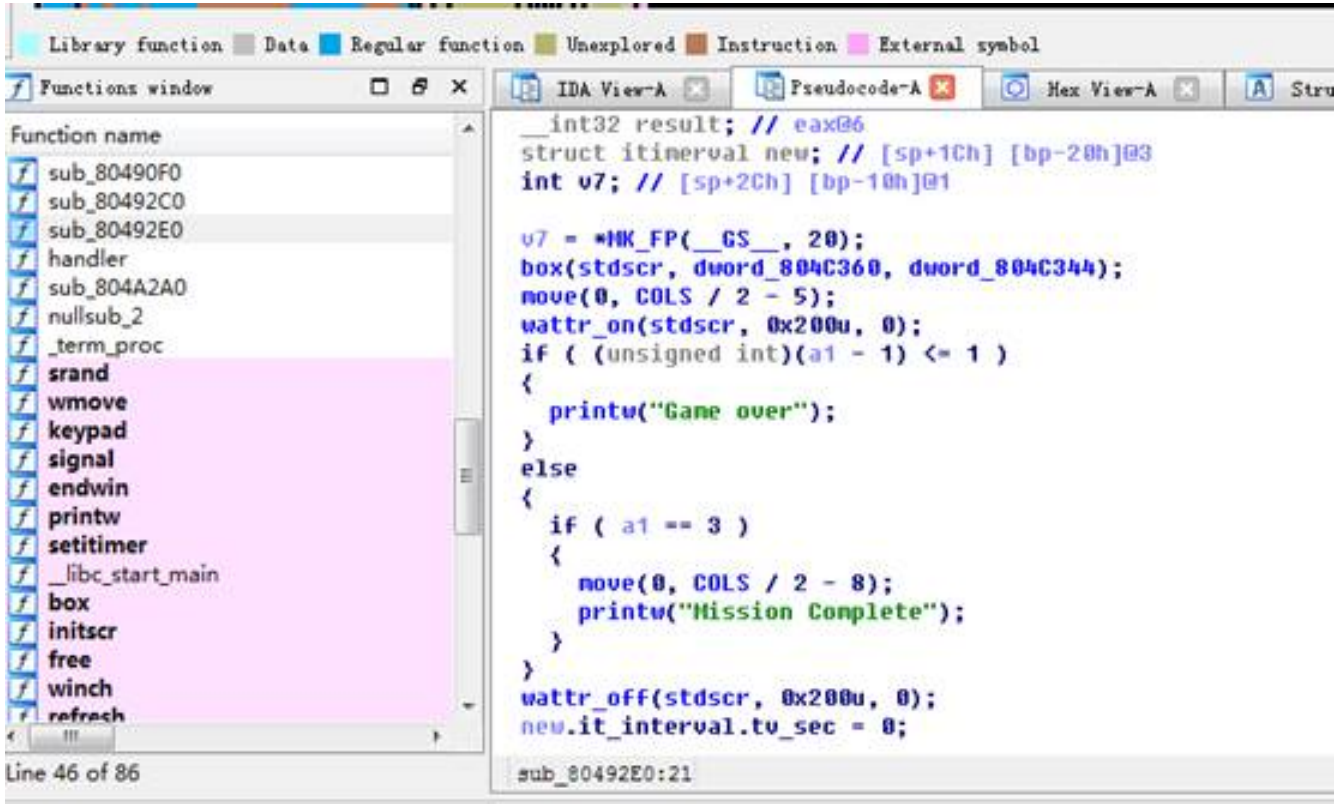
日啊！这尼玛都啥！再介绍下背景，其实我是一个WEB程序员，虽然我也会写代码，但是尼玛这都啥跟啥跟啥啊！智商明显跟不上了。

这个时候痛经牛又来拯救世界了，他说按F5可以把汇编代码还原成C语言，我一

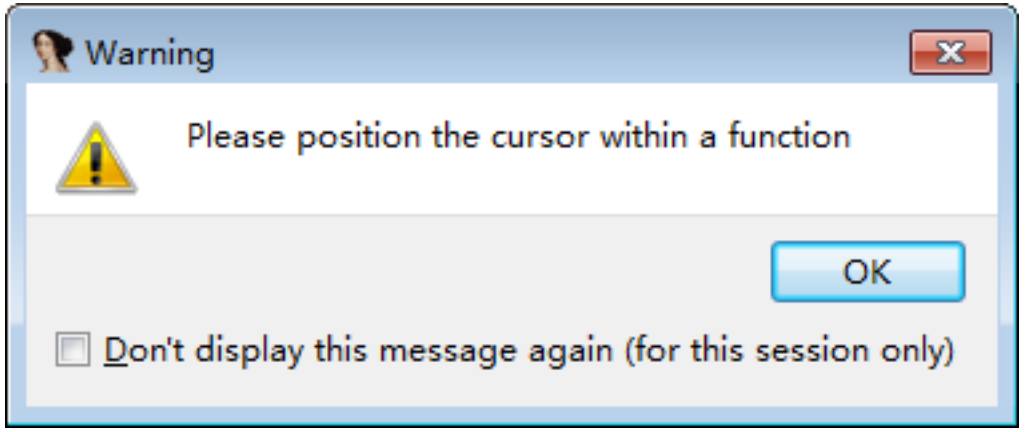
听卧槽牛逼啊，F5是吧？F5，F5，F5，F5F6F7F8riejrkioekjr45094roedowsjf3=3ir3e2啪！啪！啪！

（此处省略换键盘买键盘等一系列过程）

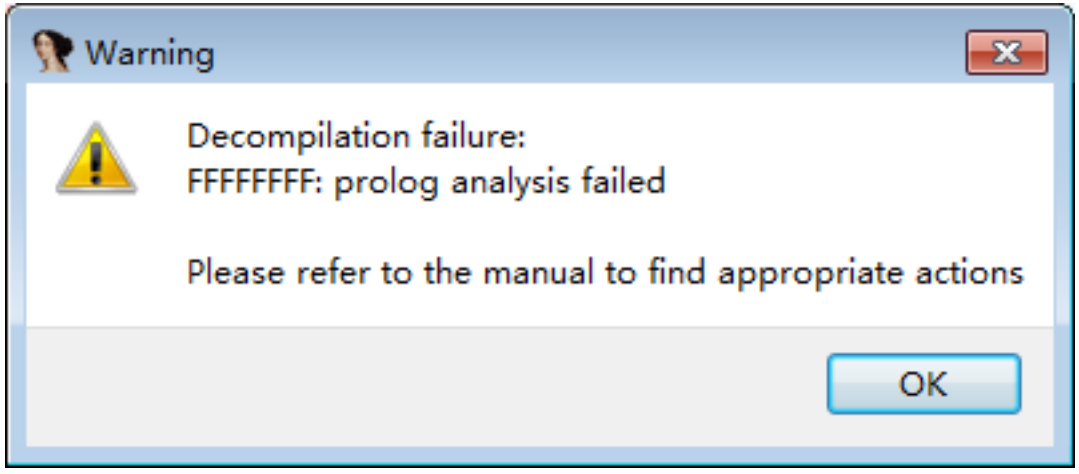
后来，发现貌似IDA64不支持F5欢迎成C这一功能。于是我默默打开了idaq.exe，随便找了个函数，然后F5。我靠果然好使！



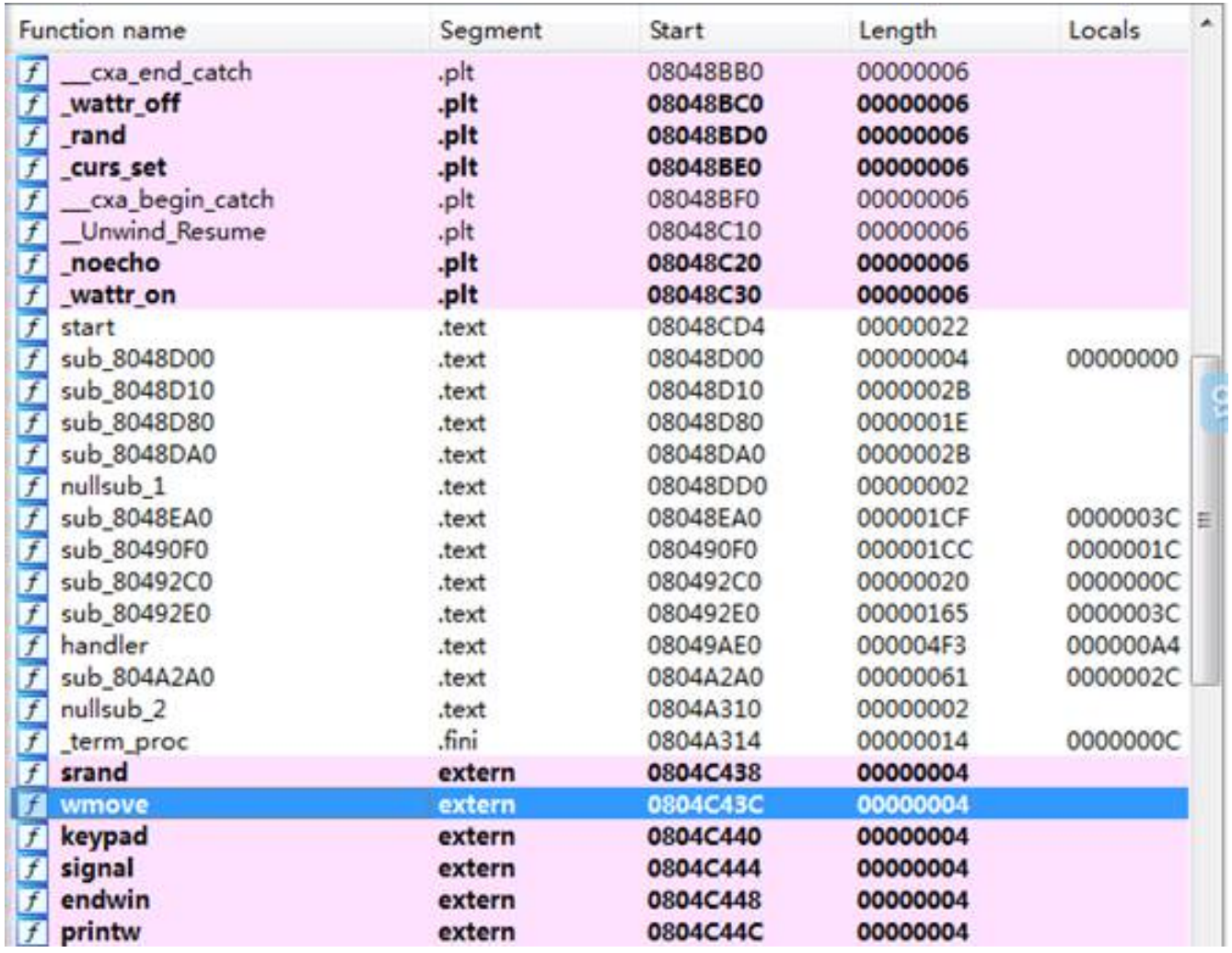
但是后来发现貌似不是所有的地方都能F5成C代码，比如会提示这个。



然后聪明的我马上就找到了其中的规律，嗯，只有函数是可以F5成C代码的。但是函数都有哪些呢？左边有。比如那个printw，我双击它，然后F5，他就会被反编译成……



额……后来，机智的我发现右边那个functions window可以拉伸！！



Function name	Segment	Start	Length	Locals
__cxa_end_catch	.plt	08048BB0	00000006	
_wattr_off	**.plt**	**08048BC0**	**00000006**	
_rand	**.plt**	**08048BD0**	**00000006**	
_curs_set	**.plt**	**08048BE0**	**00000006**	
__cxa_begin_catch	.plt	08048BF0	00000006	
_Unwind_Resume	.plt	08048C10	00000006	
_noecho	**.plt**	**08048C20**	**00000006**	
_wattr_on	**.plt**	**08048C30**	**00000006**	
start	.text	08048CD4	00000022	
sub_8048D00	.text	08048D00	00000004	00000000
sub_8048D10	.text	08048D10	0000002B	
sub_8048D80	.text	08048D80	0000001E	
sub_8048DA0	.text	08048DA0	0000002B	
nullsub_1	.text	08048DD0	00000002	
sub_8048EA0	.text	08048EA0	000001CF	0000003C
sub_80490F0	.text	080490F0	000001CC	0000001C
sub_80492C0	.text	080492C0	00000020	0000000C
sub_80492E0	.text	080492E0	00000165	0000003C
handler	.text	08049AE0	000004F3	000000A4
sub_804A2A0	.text	0804A2A0	00000061	0000002C
nullsub_2	.text	0804A310	00000002	
_term_proc	.fini	0804A314	00000014	0000000C
srand	**extern**	**0804C438**	**00000004**	
wmove	**extern**	**0804C43C**	**00000004**	
keypad	**extern**	**0804C440**	**00000004**	
signal	**extern**	**0804C444**	**00000004**	
endwin	**extern**	**0804C448**	**00000004**	
printw	**extern**	**0804C44C**	**00000004**	

它的segment属性可以分为plt，text和extern几个值，其中.plt和.extern我也忘了都是啥了之前百度过来着反正大概就是外部库引入的一些函数吧，只有.text是真正的程序代码，所以F5只针对.text的属性有效。

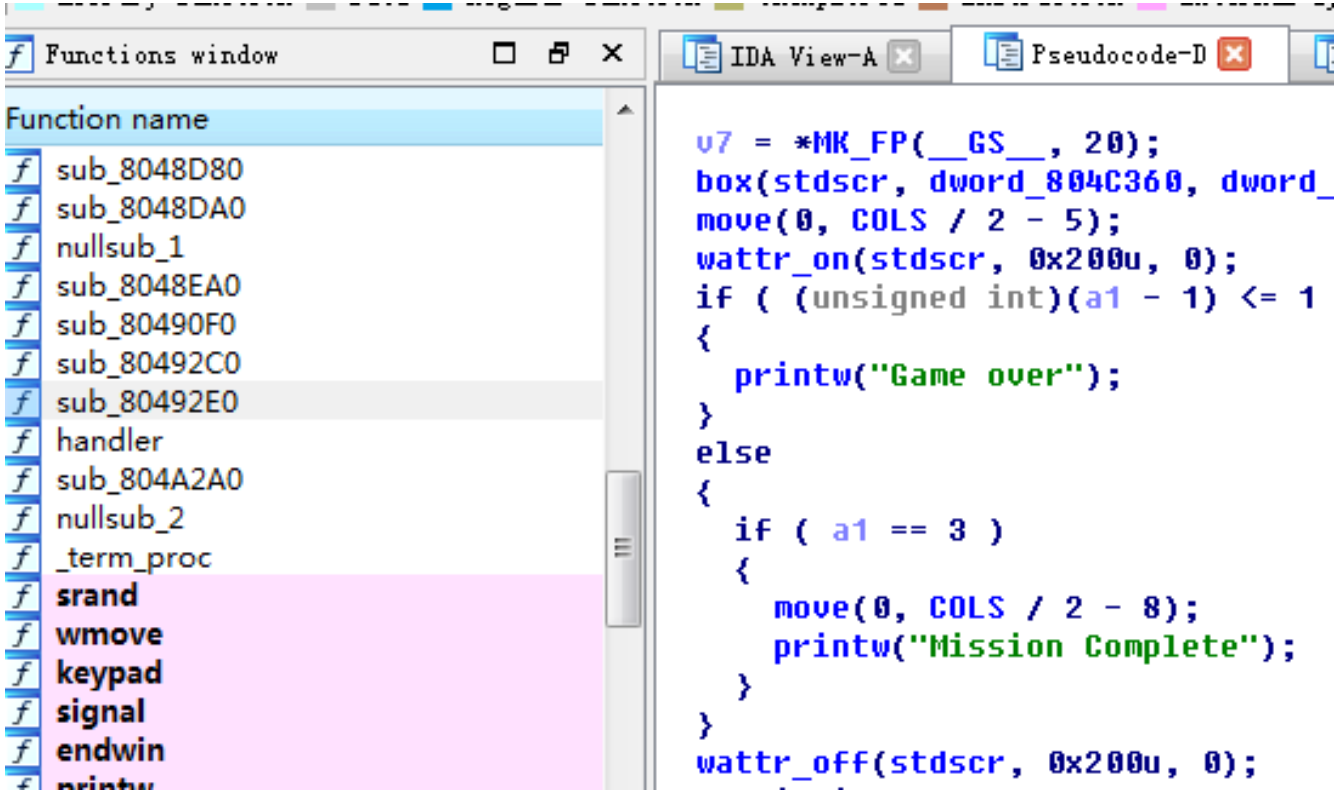
所以，理论上双击这里的任何text函数再F5都可以逆向成C代码。

嗯，有了这等神器，这题不就做出一半了么！

0x0002 爆破

//前面是7号写的。现在是12月9号18:46，最近时间不大够……

嗯，前面说到F5了对吧，确实挺牛逼的，于是我就在各个function上面F5，然后就定位到了这里。

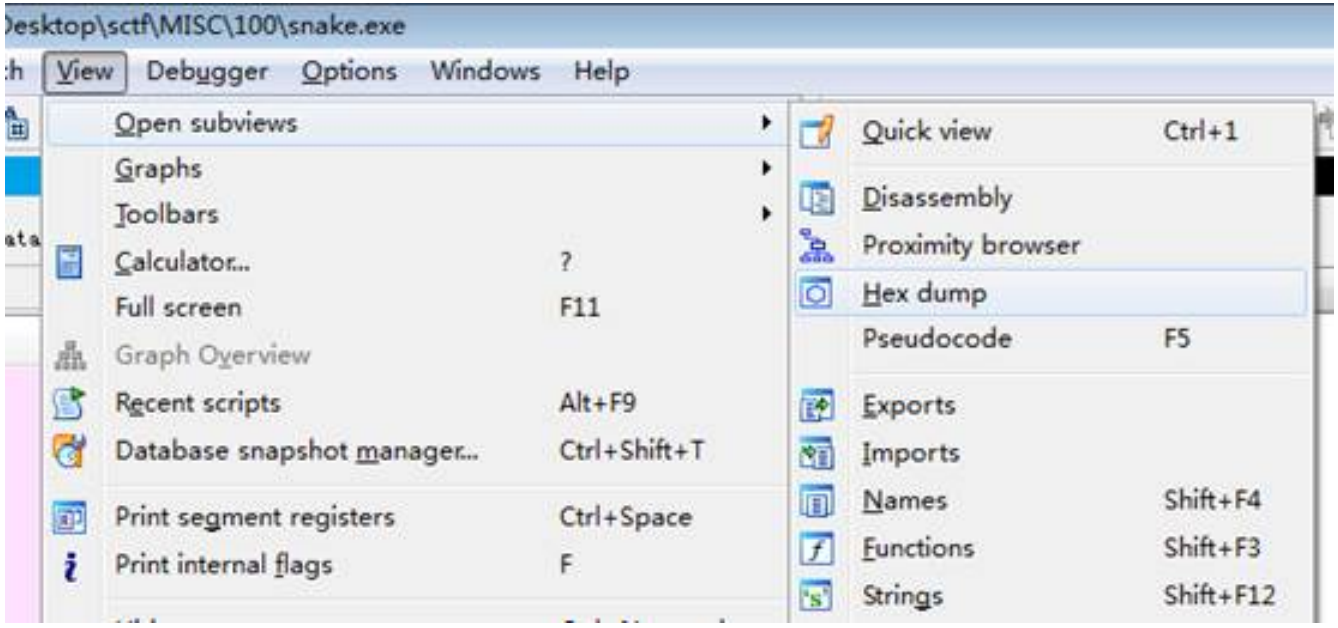


很明显，这里是判断游戏最终是成功还是失败的地方，主要的判断点有两部分，首先在if((unsigned int)(a1-1)<1)那里，把光标放在上面，按tab（我也不知道为啥是按tab但是反正我就是莫名其妙知道按tab可以在C和asm之间切换定位了），然后定位到了这里：

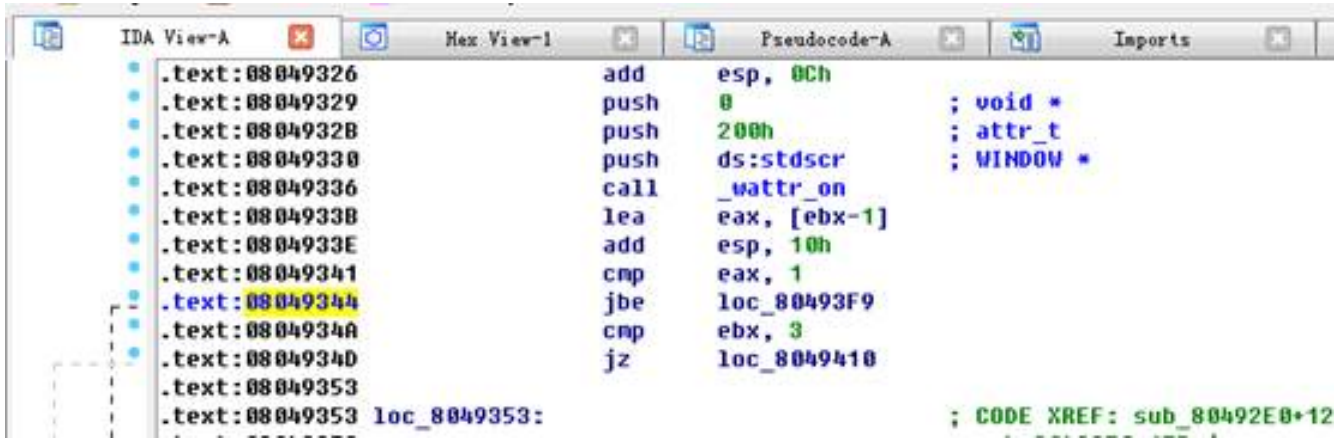
Cmp肯定是compare的意思，就是个判断语句，下面的jbe虽然我不知道是啥意

思，但是以我多年遭到的潜移默化的知识经验告诉我，凡是带j开头的都是和jump相关的，这句代码大部分都是代表if(条件) jump 到某个程序段，不同的jxx代表不同的条件，所以我们可以查一下jbe是啥意思，http://ctf.idf.cn/index.php?g=portal&m=article&a=index&id=31 然后找到了这么一篇文章，搜JBE的时候可能会搜到两条，第一条是8位的，虽然不知道是干啥的但是貌似我们的是32位的，找到下一个搜索结果，16位和32位的那里，可以看到说明是什么小于等于则跳，对应的机器码是0F86，跟它相反的指令肯定是JNBE ，不大于等于则跳，虽然我屡不清这其中的逻辑，但是大概应该是没问题的，机器码是0F87。

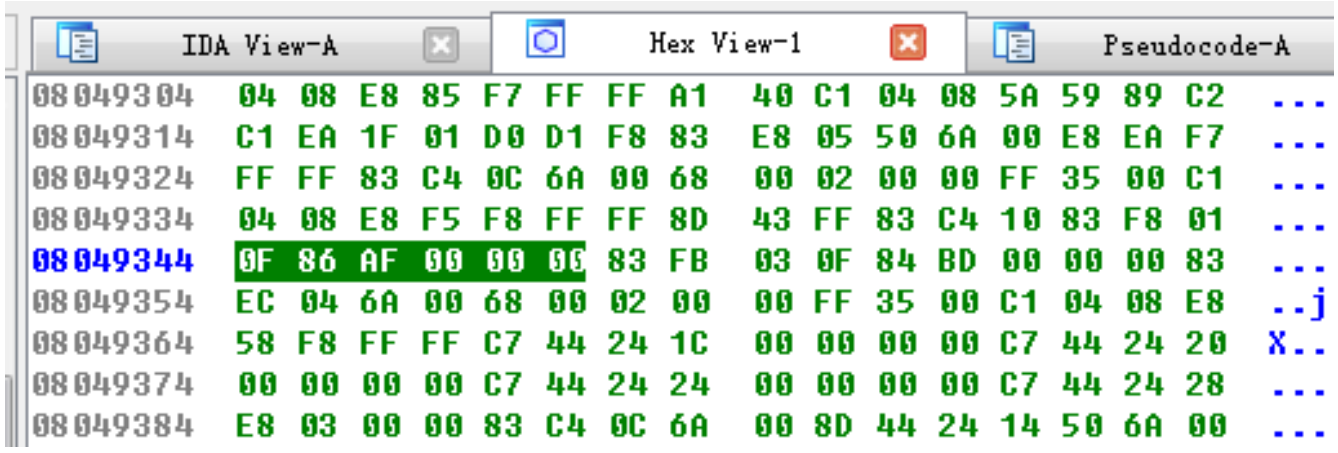
根据我零散的知识，所有的文件本质都是以2进制方式存储的，都有自己的格式。可执行文件的执行指令代码也一样，存在文件的某一块区域中，这些代码就是传说中的机器码，而机器码和ASM代码又基本是一一对应的。所以，下一步就是，从ASM定位到机器码，然后把文件对应的机器码部分修改掉。点开view菜单，open subviews，打开Hexdump：



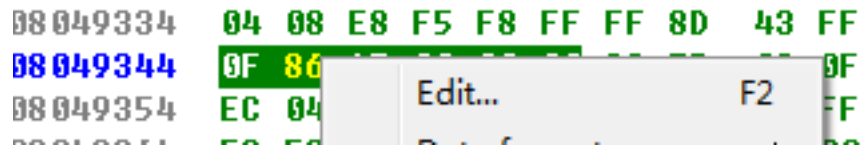
这样我们就会得到反汇编和机器码相对应的视图：



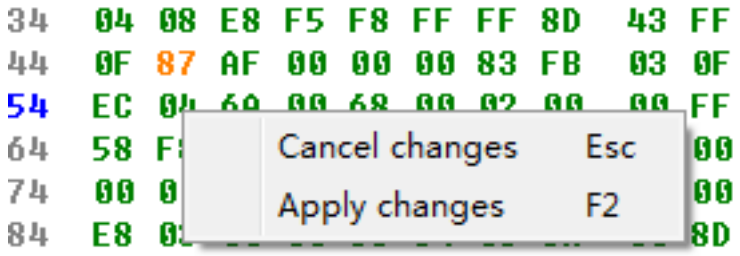
可以看到刚刚反汇编的跳转地址是0809344，点到Hex view的视图，就会自动跳转到这里。



0F86AF000000就是jbe loc_80493F9所对应的十六进制机器码，（PS：我也不知道为什么有时候直接切换视图就会自动定位并选中代码，有时候就不行，反正大概就是切换视图的时候还是要看看偏移是否对应的），刚才说了0F86就是jbe，我们要把它改成0f87就会反向原来程序中的条件。也就是刚才反出来的那段 if((unsigned int)(a1-1)<1) 会变成 if((unsigned int)(a1-1)>=1)，在上面右键，点EDIT：



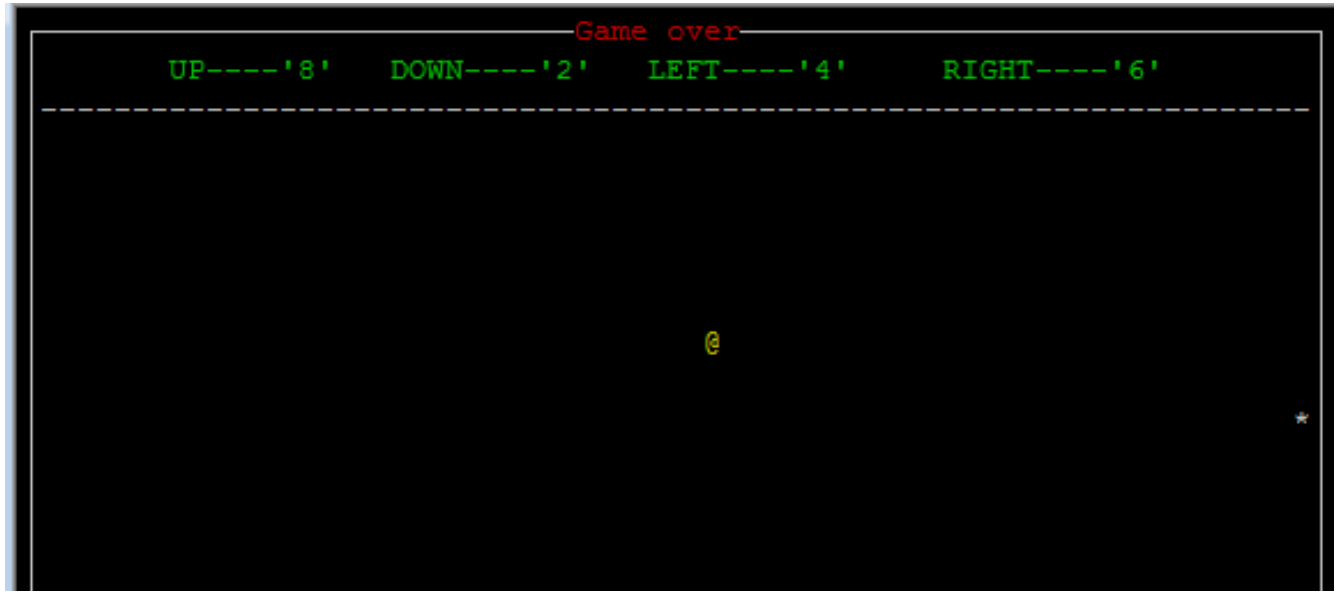
然后把86改成87，然后继续右键，apply changes，当然也可以F2。



然后我们再回到ida view，就是主视图去看我们改过的汇编代码，不出意外就会从jbe变成jnbe了，看！

卧槽？ja是个啥？后来又回到那篇文章看了看，jnbe是不小于等于则跳，ja是大于则跳，机器码都是87，我读书少，但是发明汇编的人你这么逗我眞的好吗……

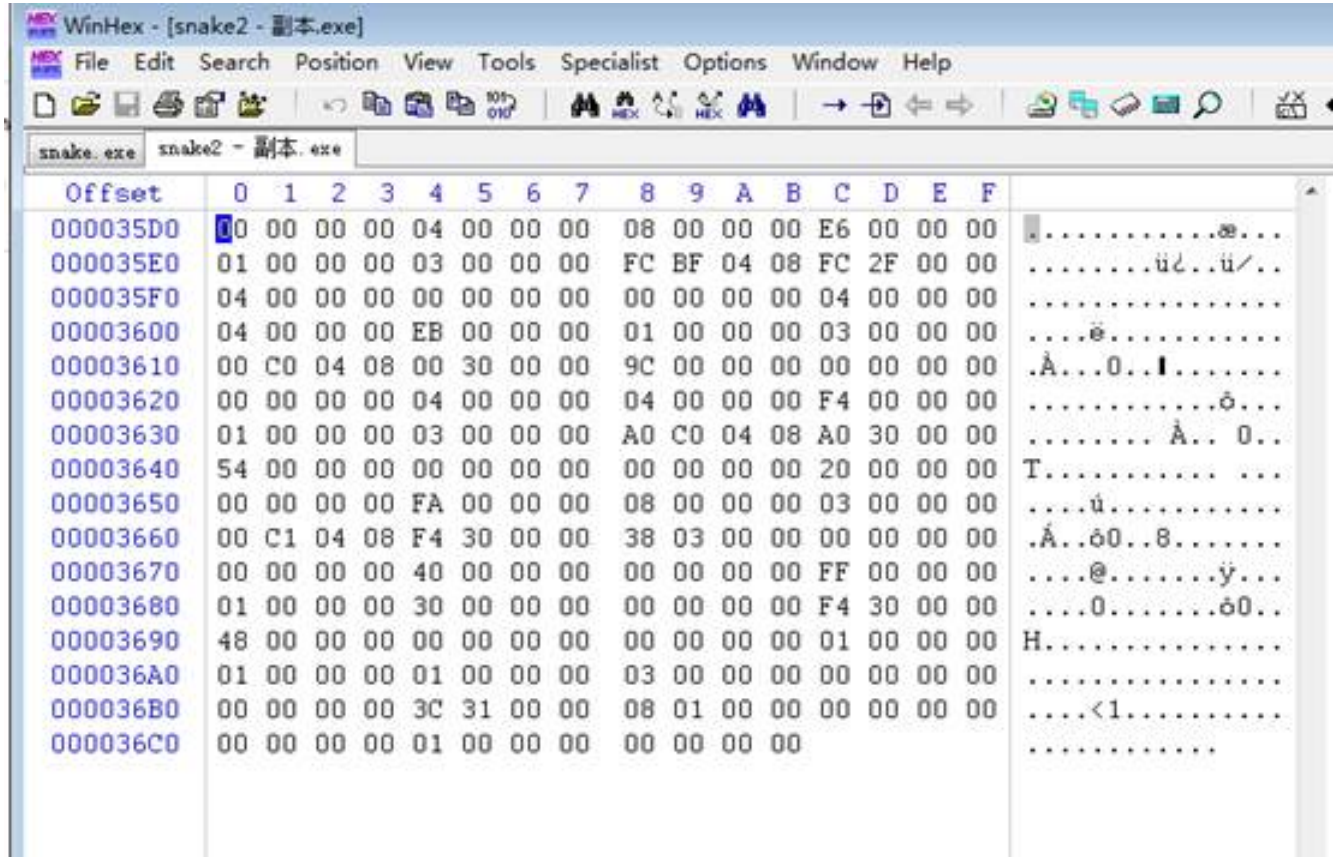
因为刚才那段C代码明显看出是有两段对比，所以同理，在下面那个if也要改一下，就是0804934D那个jz，等于则跳，改成jnz，不等于则跳。机器码从84改成85，F2生效。好，保存我们的工程。理论上来讲只要运行我们的程序，当我们游戏结束的时候，就会出现mission complate字样，我简直太机智了，看！



一定是我打开的方式不对.....

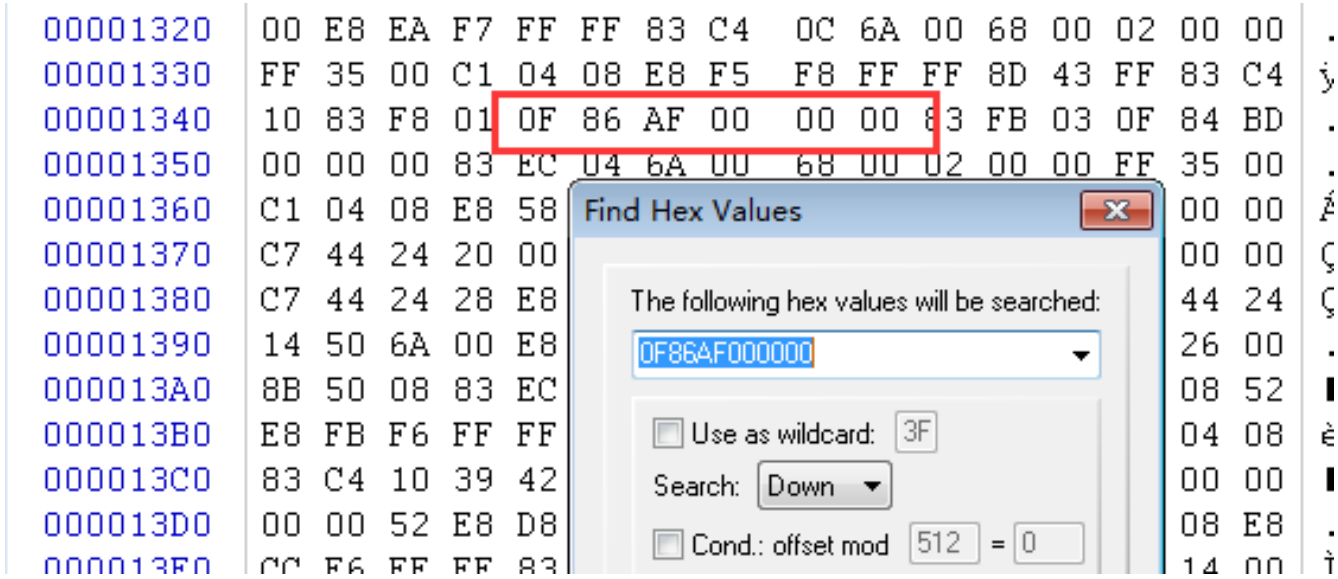
咦？为啥修改日期还是没变？卧槽IDA你这不坑爹么我都把十六进制改了你居然没保存到文件！！

好吧，拖出一个副本，直接扔到WinHex中，08049344是吧，我在winhex里面改一下不完了么。

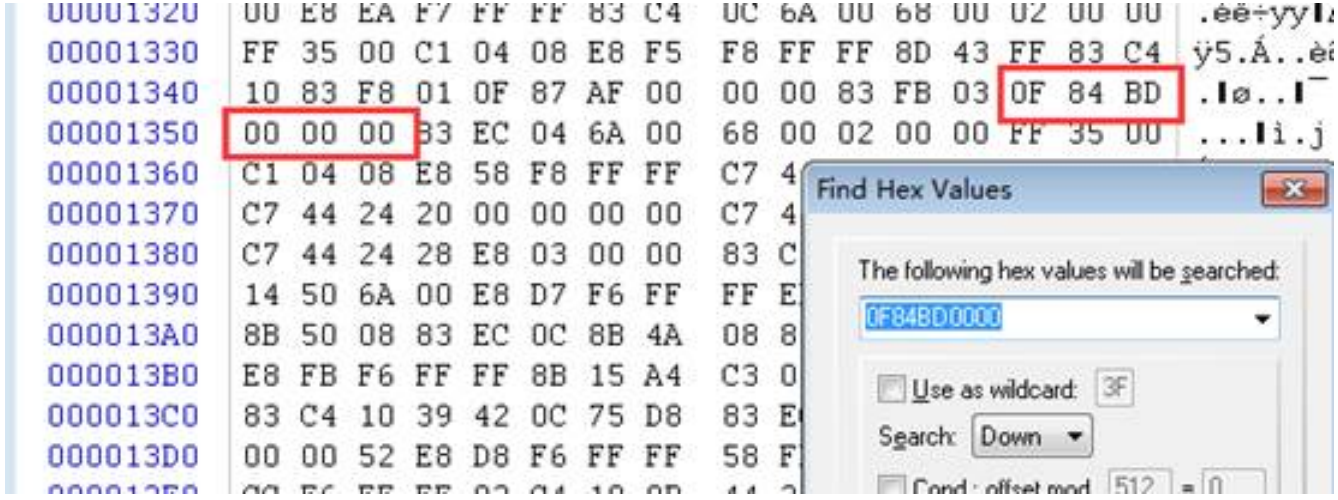


怎么感觉少了点什么，一定是我复制的不完整……

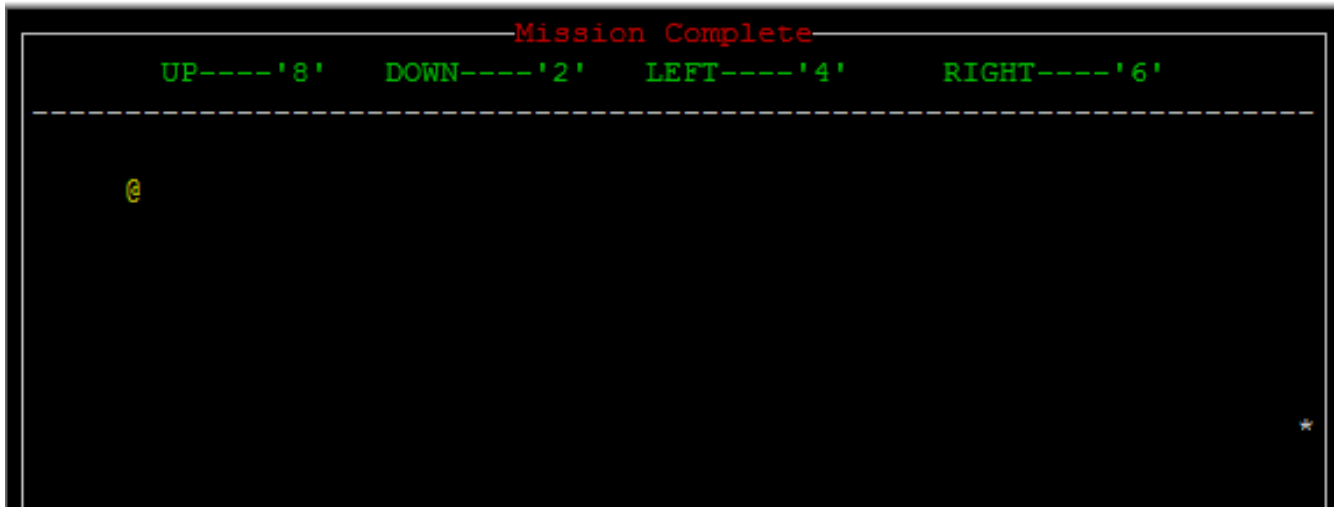
不过这难不倒机智的我，那段机器码是啥来着？0f86af000000是吧。



0f84bd000000是吧？



全部改完，走你！



mission complate！肿么样，我是不是很厉害~这题太弱了！有能耐你们整四岁的！

说好的flag呢……

不过没关系，都已经搞出mission complate了，这题不就做出一半了么！

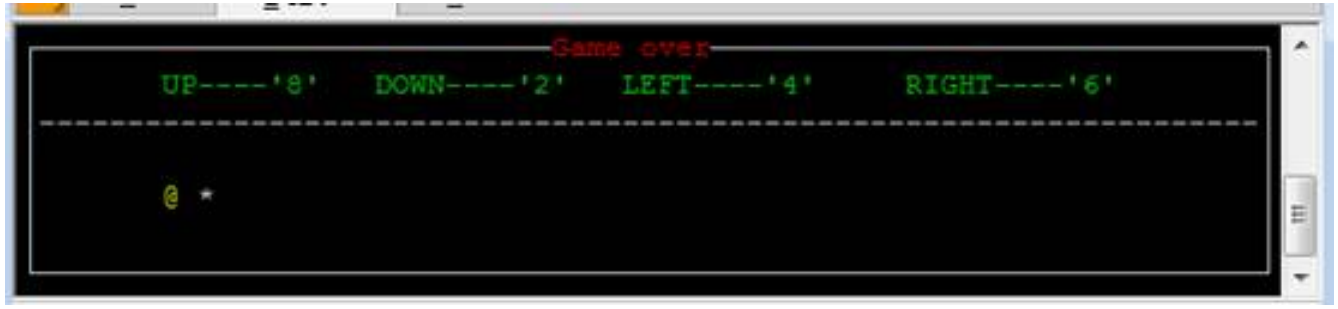
0x0003 篡改

（此处省略无数探索过程）

一般遇到这种情况，我都会想静静。

嗯，在11上被虐了无数盘之后，我又回到了贪吃虫上来，然后机智的我很快就想到，这个牛逼的贪吃蛇居然可以根据窗口的大小自动调整，我要把它高度改成1那

岂不就是直接过关？走你~



它咋不动呢……咋不动就gameover了呢。后来机智的我又发现一个秘密，就是每次的*号出现的位置都是固定的，都是大概第六七八行左右我数学不好反正就是那块儿吧，如果窗口过矮就会不动。嗯，咦？那@的位置是咋写的呢？

继续一点一点读代码。

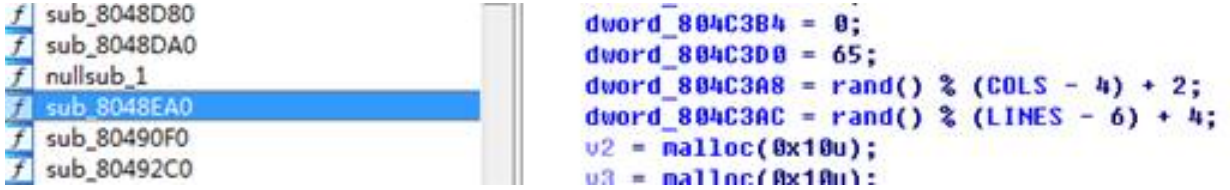
然后我在handler的函数中看到了这一段代码。

聪明的我马上就猜到@的位置是根据那个啥啥 804C3AC和804C3A8决定的，一个是x一个是y。那么这俩变量是哪里的呢。

于是又在hanlder里找到了这个。

```
}
do
{
  dword_804C3A8 = rand() % (COLS - 4) + 2;
  dword_804C3AC = rand() % (LINES - 6) + 4;
}
```

在sub_804AEA0里找到了这个，这个函数应该是初始化用的，即首次位置。



于是我突然想，去尼玛我想办法每次把@的位置放到*号前面不就完了么我眞是太机智了我靠我自己都佩服自己！

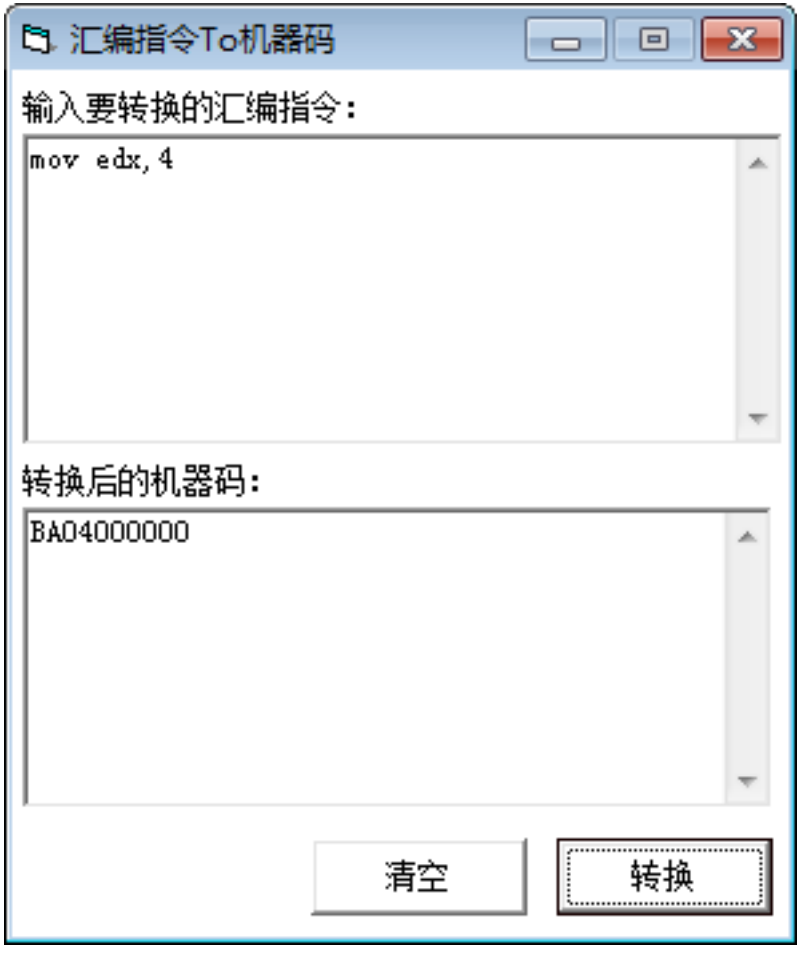
804C3AC肯定是y，来我先把它固定。dword_804C3AC=rand()%(LINES-6)+4，

```
.text:08048FBE                call    _rand
.text:08048FC3                mov     esi, ds:LINES
.text:08048FC9                cdq
.text:08048FCA                mov     [esp+3Ch+timer], 10h ; size
.text:08048FD1                lea     ecx, [esi-6]
.text:08048FD4                idiv    ecx
.text:08048FD6                add     edx, 4
```

对应的汇编代码是这里，虽然我不懂汇编，但是add eax,4就是最后的那一步加4我还是能看出来的，然后我就机智地想到，把加4变成等于4不就可以了么，改成mov edx,4，嗯，汇编代码编出来了，再来改对应的机器码。

```
08049F11   FC F7 F9 83 C2 02 89 15   A8 C3
08049F21   FF 8B 35 04 C1 04 08 99   83 EC
08049F31   83 C2 04 52 FF 35 A8 C3   04 08
08049F41   89 15 AC C3 04 08 E8 C4   EA FF
08049F51   FF 0F 84 60 FE FF FF 83   EC 0C
08049F61   E8 5A EB FF FF 83 C4 10   83 F8
```

这个时候我突然想起来，小时候我好像是看过汇编的，好像什么从哪mov到哪事儿挺多的，刚才是add我改成mov会不会有些问题，虽然我还不知道mov的机器码是啥，于是找啊找啊找的终于找到了这个。

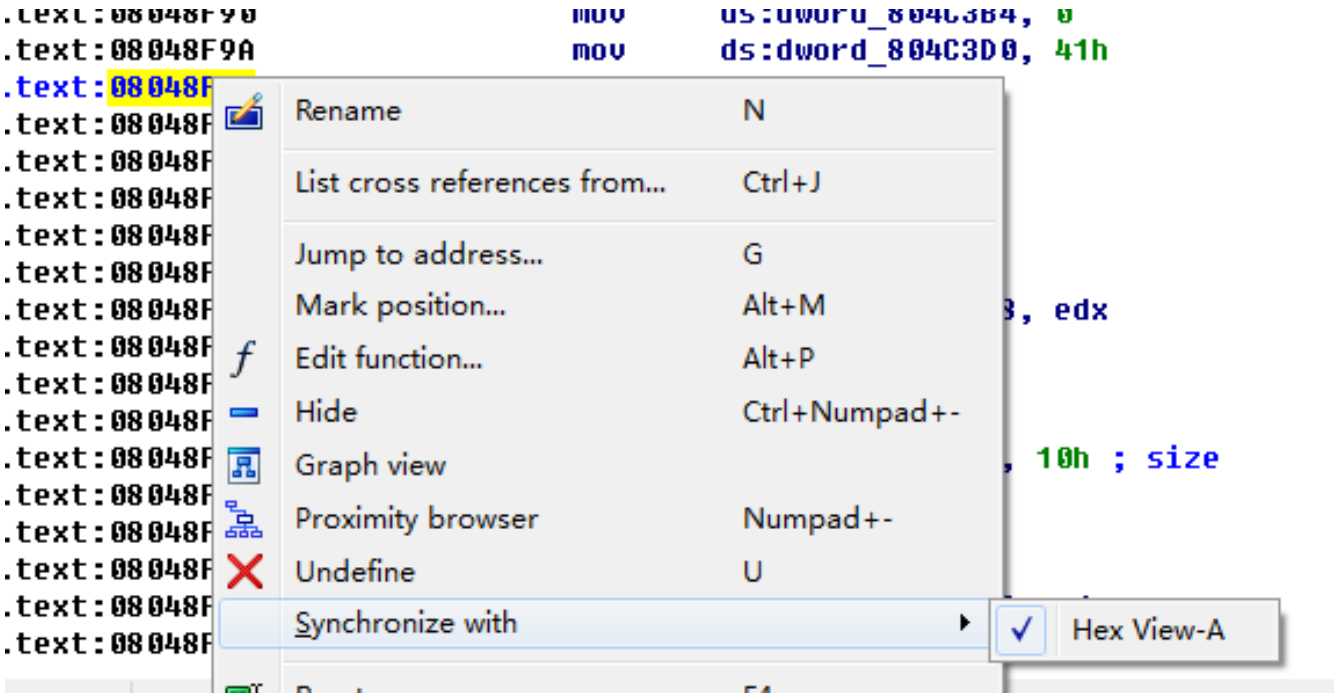


卧槽神器啊，but，为啥这么长……

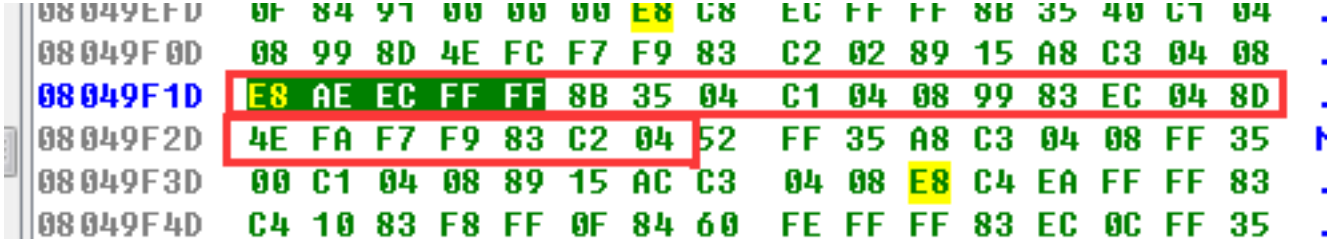
好吧我们再来看看源码。

```
.text:08048FBE                call    _rand
.text:08048FC3                mov     esi, ds:LINES
.text:08048FC9                cdq
.text:08048FCA                mov     [esp+3Ch+timer], 10h ; size
.text:08048FD1                lea     ecx, [esi-6]
.text:08048FD4                idiv    ecx
.text:08048FD6                add     edx, 4
```

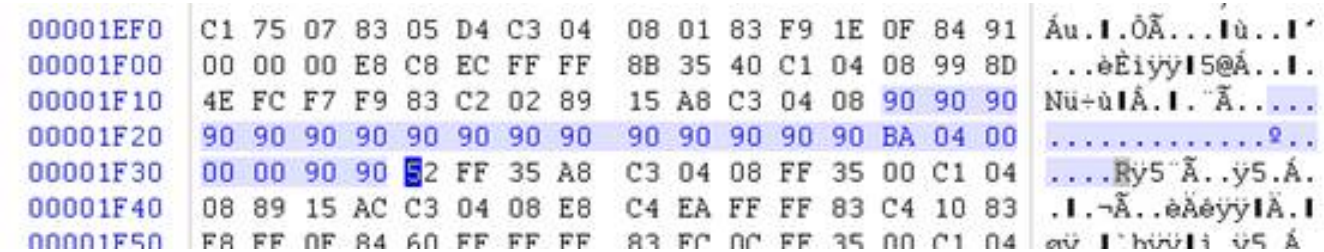
整个的计算流程大概是从08048FBE一直到08048FD6，好，反正他们是连着的，只要我把任何一个地方改成B804000000然后两边nop掉不就可以了，咦？卧槽我在写这篇文章的时候突然学会怎么同步反汇编窗口和hex窗口了，这样。



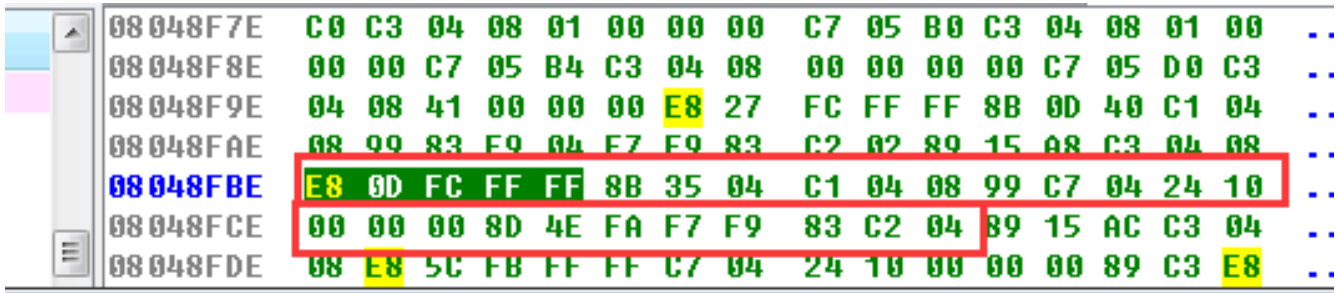
嗯，然后继续说。我要改代码了，首先用winhex先打开snake.exe，搜一下这两段之间的代码，大概就是下面这一段。

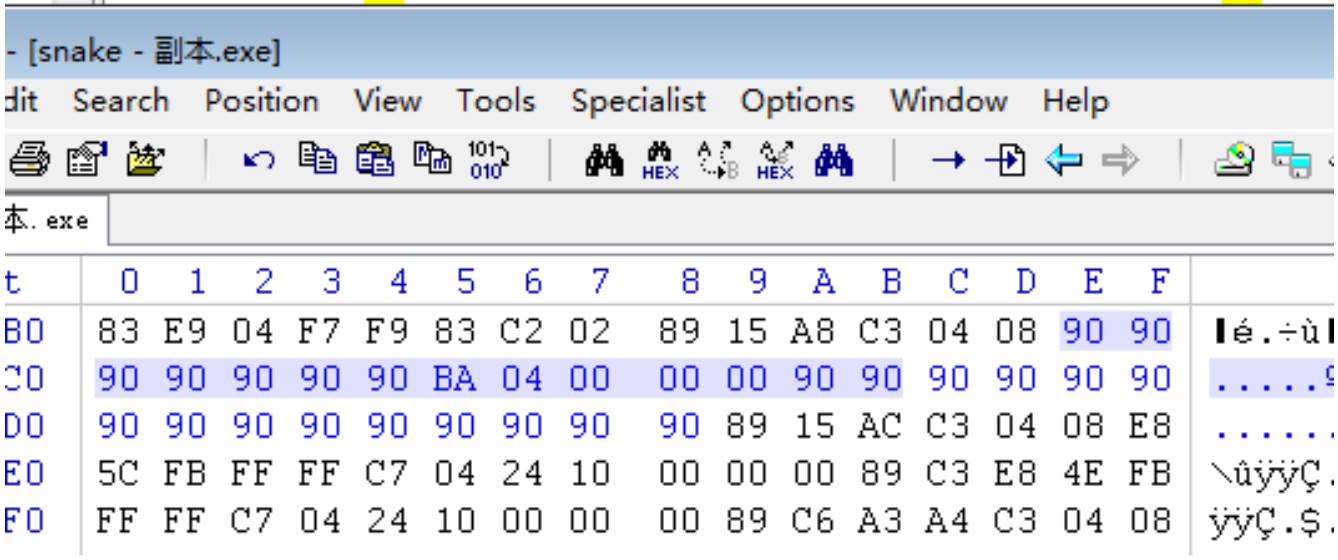


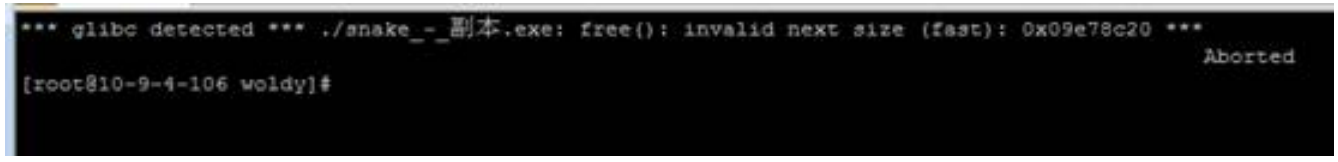
然后到winhex中，全他妈的改成90，然后再加一个BA04000000。



刚才说过这里有两段，一个是每次吃完@后重新生成的，这个在hanlder中，另一个是初始化的，在sub_804AEA0里面，修改同理。







卧槽哪里不对么.....

对比了好几遍，还是不明白为啥会这样，不管那么多了，回去再看一看代码：

```
.text:08048FB3                 idiv    ecx
.text:08048FB5                 add     edx, 2
.text:08048FB8                 mov     ds:dword_804C3A8, edx
```

从idiv到add这两行的代码正好对应五个字节：

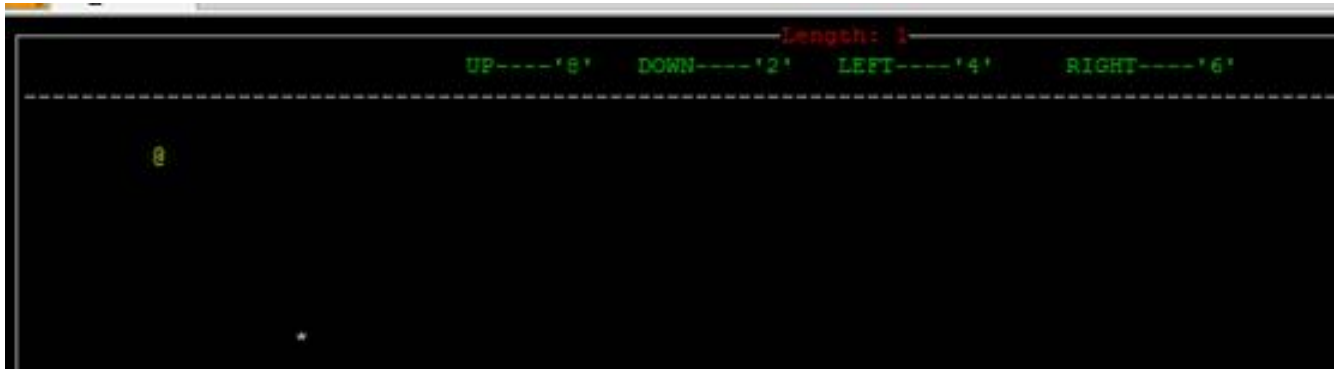
```
08048FA3  00 E8 27 FC FF FF 8B 0[
08048FB3  F7 F9 83 C2 02 89 15 A[
08048FC3  8B 35 04 C1 04 08 99 C7
```

直接把F7F983C204改成BA0400000。

切回代码，可以看到，dword_804C3AC变成4了，卧槽我好厉害！

```
dword_804C3C0 = 1;
dword_804C3B0 = 1;
dword_804C3B4 = 0;
dword_804C3D0 = 65;
dword_804C3A8 = rand() % (COLS - 4) + 2;
rand();
dword_804C3AC = 4;
v2 = malloc(0x10u);
```

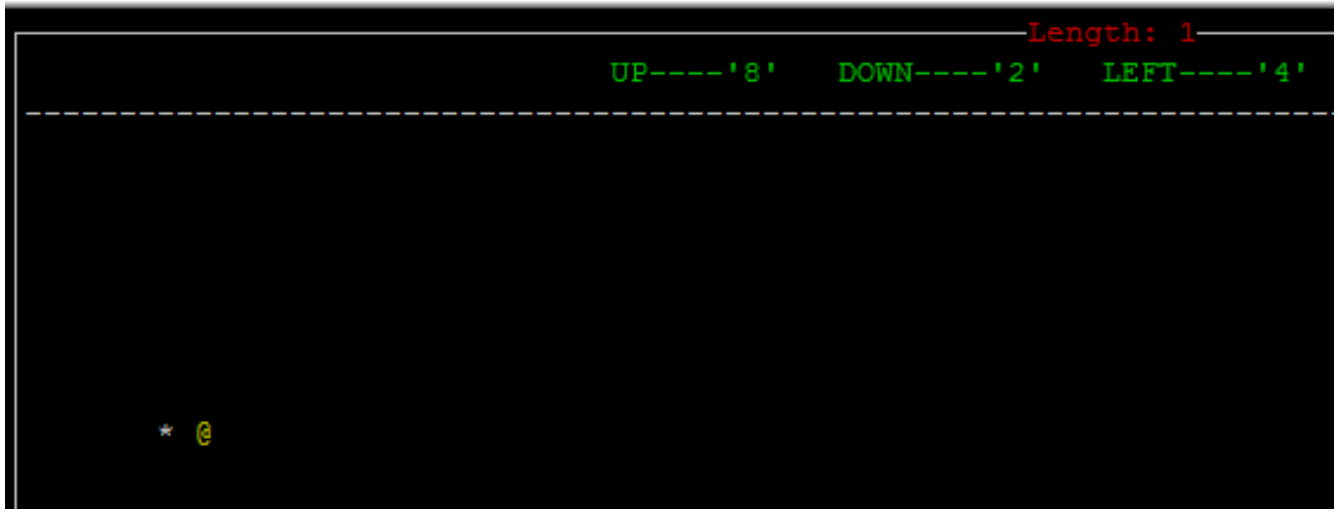
跑一下看看，看到那个@了没，果然是在第四行！！



现在我把04改成0A,也就是每次都出现在第10行，当然前面那个BA0400000也要

改。

OK，没问题了，这样每次的高度就固定住了，我们再来研究研究x轴坐标，首先初始化的坐标我们可以改一下，嗯，就改成10好了。方法很简单，和上面同理，在sub_804AEA0里面找到dword_804c3a8对应的汇编代码，也是idiv啥的，对应的机器码是F7F983C202，改成ba0a000000，不过直接在winhex里搜的话肯定会搜到两个结果，因为两个地方代码是相同的，我们只要改初始化的那里就行了，区分方法也很简单，搜索前面的机器码就可以了，04089983E904，然后把后面改成BA0A000000。



成功！，但是吃完了这个@之后，后面的@的x位置就不受控制了，有时候甚至会

跑到*的前面，咋整呢？想了想，让dword_804c3a8每次的值都自动+1不就完了！

但是，卧槽汇编代码咋写……

不过ida帮我逆出来了这么多的汇编代码，难道我抄还不会么……咦？这里有一段。

```
if ( v5 > 60 )
{
  ++dword_804C3CC;
  dword_804C3C8 = 0;
}
```

看看汇编代码咋写的：

```
.text:08049187                  jle     short loc_804919A
.text:08049189                  add     ds:dword_804C3CC, 1
.text:0804919A                  mov     ds:dword_804C3C8, 0
```

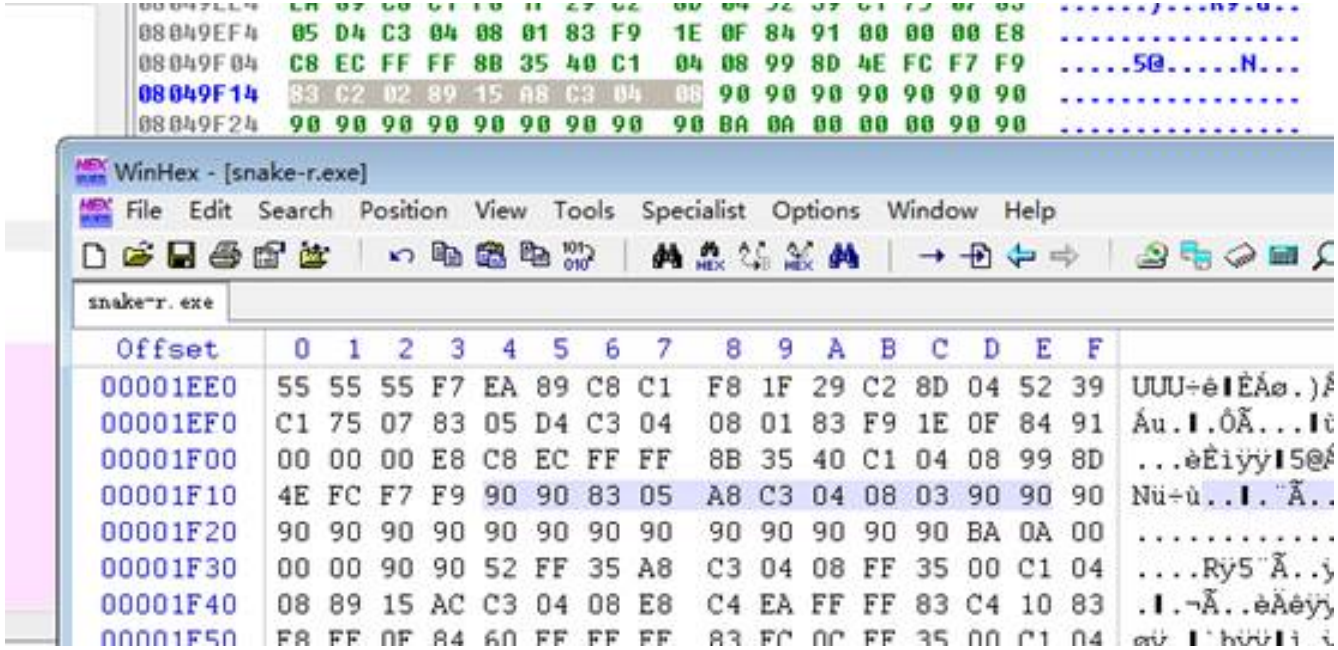
再看看机器码是啥。

```
08049179  0F 8F 21 01 00 00 A1 C8
08049189  83 05 CC C3 04 08 01 C7
08049199  00 A1 40 C1 04 08 83 EC
```

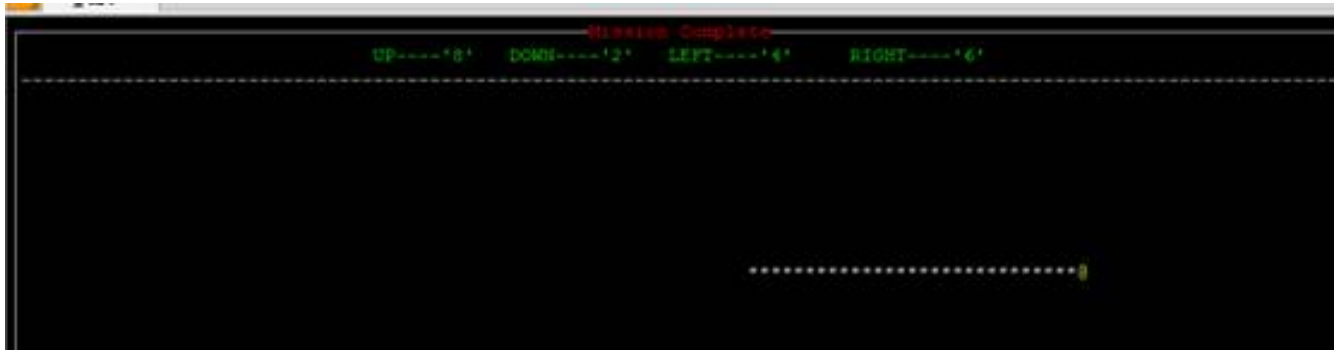
嗯，大概七个字节，8305肯定是add啥啥的了，后面是地址，最后一位是1。

找到hanlder的function，找到这个add和mov啥啥的，直接跳转到机器码，把它改成8305啥啥啥的就ok了。

```
.text:08049F0F                  lea     ecx, [esi-4]
.text:08049F12                  idiv    ecx
.text:08049F14                  add     edx, 2
.text:08049F17                  mov     ds:dword_804C3A8, edx
.text:08049F1D                  pop
```



最终看看效果：

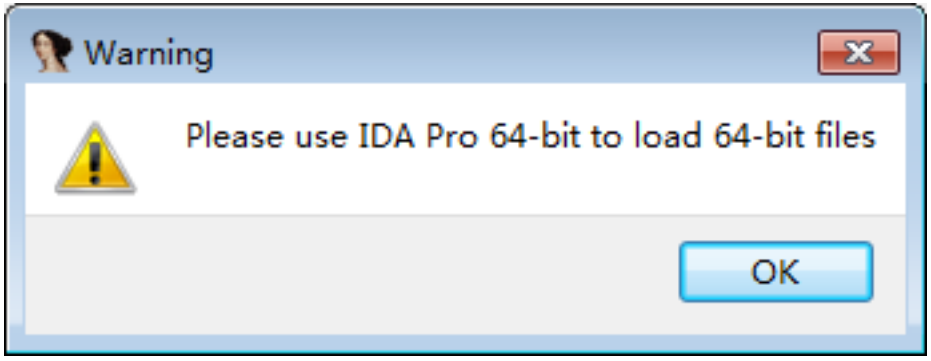


成功！不过，说好的flag呢……

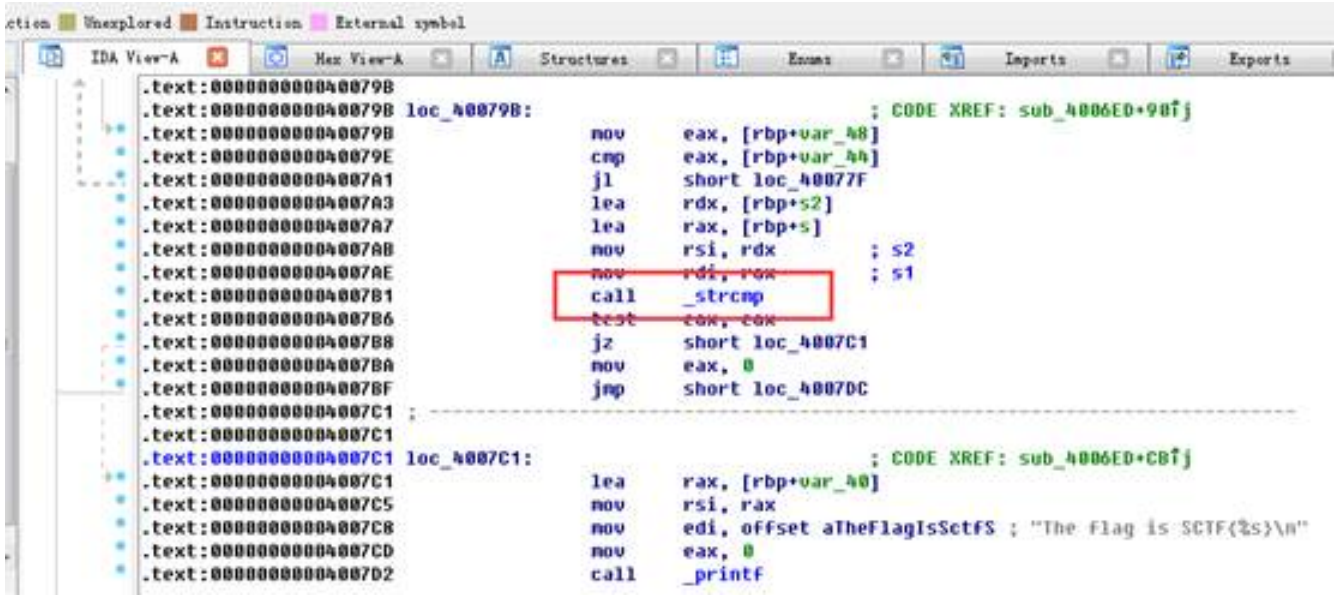
0x0004 RE50

话说这个贪吃蛇严重打击了我的自信心，让我对CTF这个比赛彻底失去了信心，就在这个时候，一道RE50的题横空出世，就像那啥中的那啥啥给找指明了方向，100分的题我不会，50分的题还能难道我吗！

直接下载文件，打开。

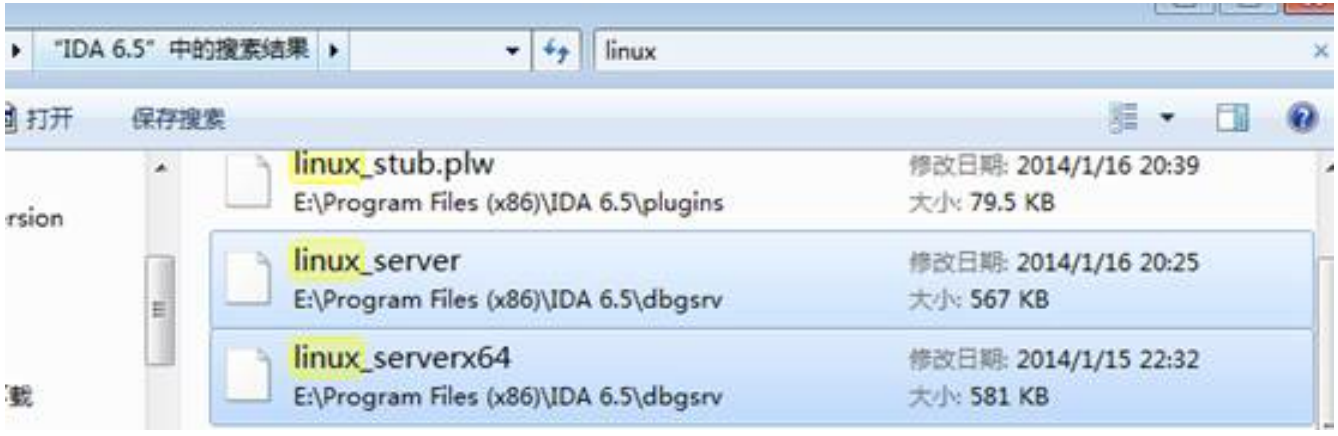


一定是我打开的方式不对，换64位打开。



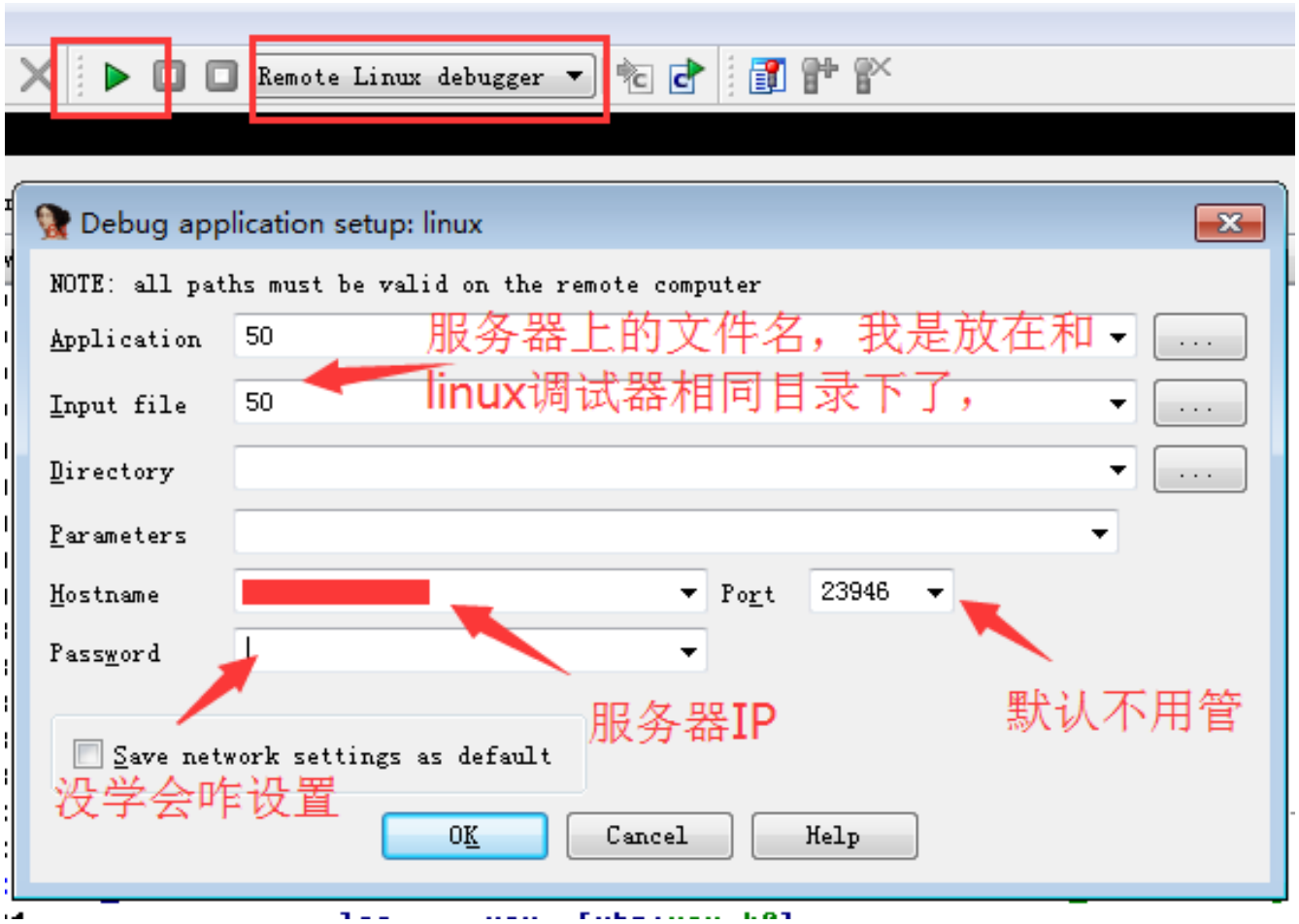
首先映入眼帘的就是这一坨东西，虽然在64位下没办法F5，不过strcmp我还是认识的，字符串比较么！这个时候要是能动态调试的话直接看看他们的值就好了……嗯？动态调试？

百度了一下，神器的IDA果然是可以动态调试的，不过都没咋看明白，最后艰难地学会了一个最简单的方法。就是在ida安装目录下找到这俩文件，分别对应x86和x64版，拖到linux服务器上运行就OK了。

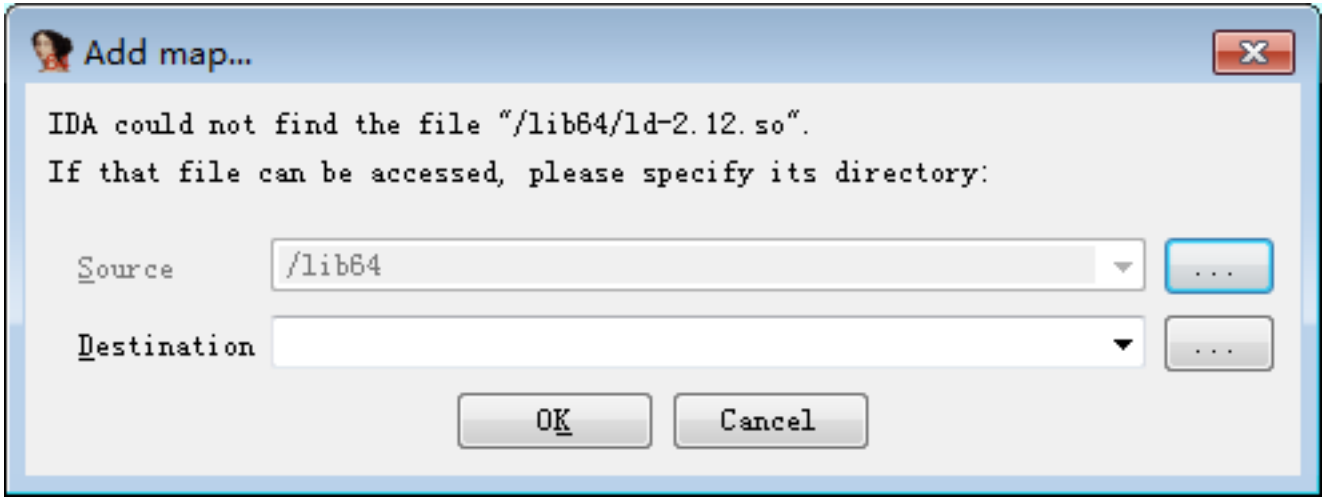


```
[root@10-9-4-106 woldy]# ./linux_serverx64
IDA Linux 64-bit remote debug server(ST) v1.17. Hex-Rays (c) 2004-2013
Listening on port #23946...
```

然后在对应的IDA客户端上选择好调试方式，点那个绿色的按钮。

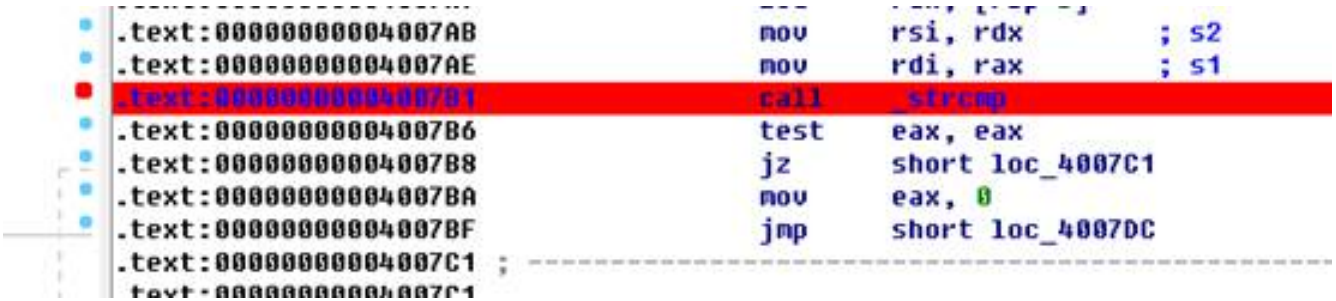


OK之后走你，会弹出对话框让你设置一堆东西。然后不停取消，好像也没啥影响。



接着，查看远程shell，程序在远程成功执行了。

随便输入了一坨东西，然后就结束了。好，现在我们在strcmp那里打个断点，虽然不太懂但是多年的程序员生涯还是让我直觉地感到这是有用的。



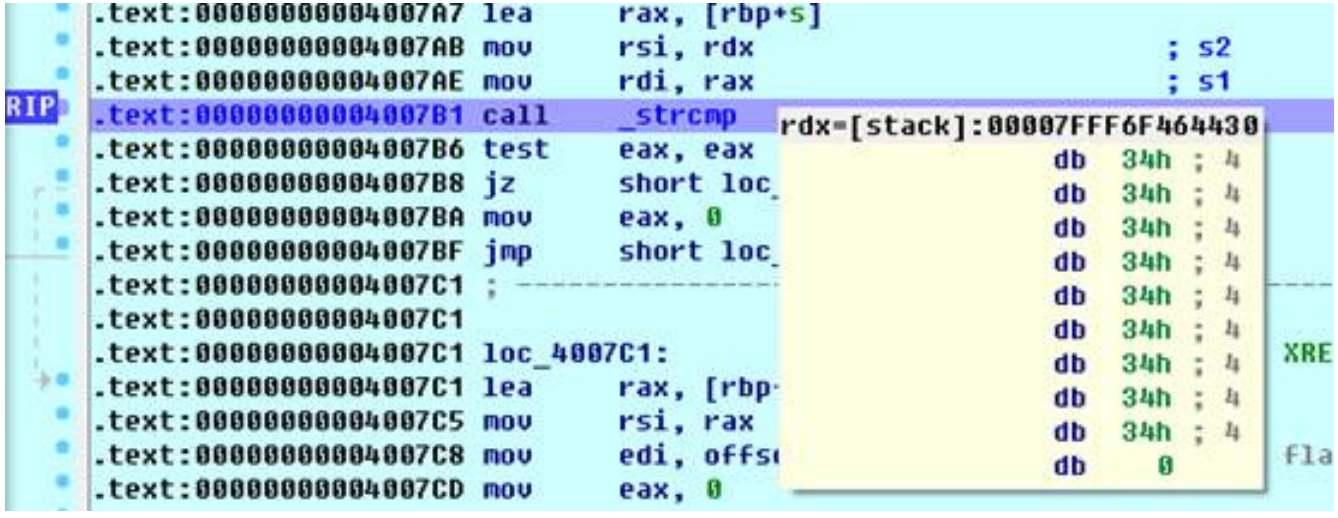
输入一坨东西，这个时候我们就会来到断点处去看他们的字符串对比，卧槽咋又退出了？

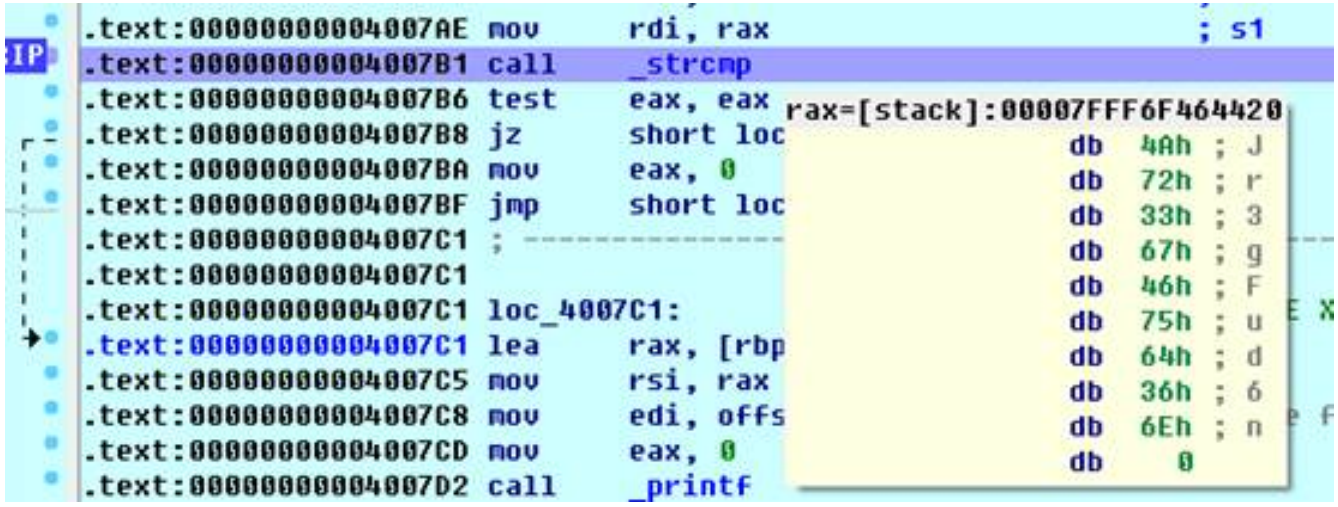
往上翻代码，看到了这里。

```
.text:000000000040071F          mov     edi, offset format ; "input your password:"
.text:0000000000400724          mov     eax, 0
.text:0000000000400729          call    _printf
.text:000000000040072E          lea     rax, [rbp+var_40]
.text:0000000000400732          mov     rsi, rax
.text:0000000000400735          mov     edi, offset aS  ; "%s"
.text:000000000040073A          mov     eax, 0
.text:000000000040073F          call    __isoc99_scanf
.text:0000000000400744          lea     rax, [rbp+s]
.text:0000000000400748          mov     rdi, rax        ; s
.text:000000000040074B          call    _strlen
.text:0000000000400750          mov     [rbp+var_44], eax
.text:0000000000400753          cmp     [rbp+var_44], 9
.text:0000000000400757          jz      short loc_400760
.text:0000000000400759          mov     eax, 0
```

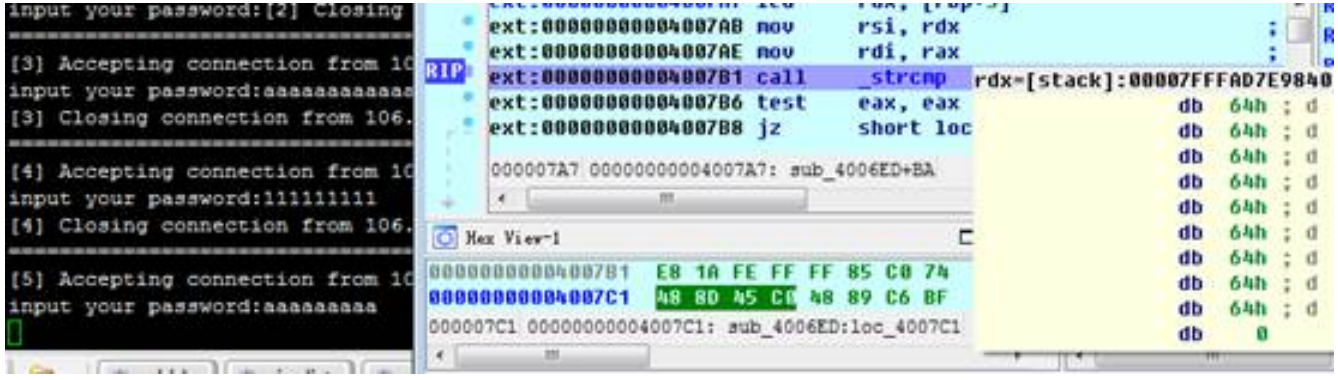
输完密码后有个啥啥cmp 9的，难道长度是九位？我试试。

输入了9个1，果然断到strcmp处了，看看是咋cmp的。



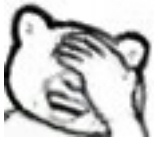


看了看离得最近的两个变量，一个是9个4，另一个是Jr3gFud6n，很明显那9个4就是我之前那9个1，尼玛，谁让你给我变的，1变成4，我要是输入9个a你岂不是还要给我变成9个d？



我读书少，这是凯撒加密么？

最后，我又花了半个小时的时间来破解ascii码加3过的Jr3gFud6n……



0xffff 后记

卧槽真不敢相信我废话居然这么多，一个贪吃蛇我都能BB将近五千字，主要是第一次接触IDA这么高大上的东西玩儿激动了，真是第一次用，所以文章写的哪里不对请不要骂我，有能耐你拿辣条来抽我啊！我键盘两千多呢……看我BB了这么久多辛苦，一百块钱都没有人给我，还打我，我跟你们什么怨什么仇啊……

Made in China by woldy，@无所不能的魂大人

公元2014年12月10日2:00

PS：除IDA外本文所提到文件地址在此→点击下载

（全文完）

Write your comment here

发表评论

友情链接： IDF实验室-博译有道 合天网安实验室 乂魂域ζ Exp1oit 皓仁

Made in China by woldy