

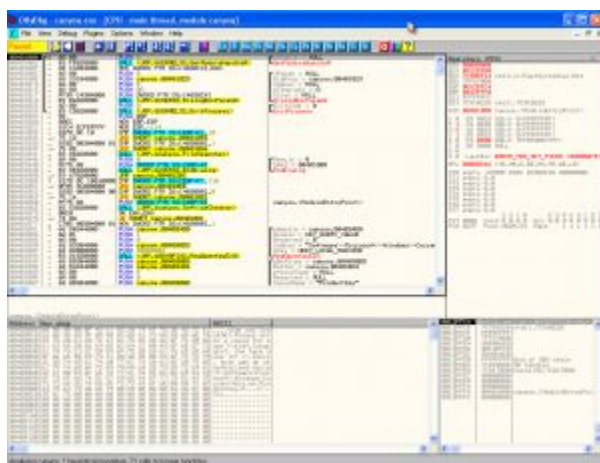
# 第七章：更多破解练习

## 一、简介

欢迎来到 R4ndom 逆向工程教程第七章。今天，我们破解两个 crackme：一个我们用来复习上一章的相关概念，另一个我打算用来做一些有趣的事😁。在本教程的相关下载中，你可以找到这两个 crackme，以及在第二个程序中要用到的软件“Resource Hacker”。你也可以在工具下载页面下载这些[工具](#)。你可以在本教程的[教程](#)页面下载相关文件，以及本文 PDF 版本（译者注：英文版）。

## 二、探究二进制文件

直入主题吧。Ollly 载入 canyou.exe（要确保 canyou.dll 在同一目录下）：  
(p1)



就像我以前说的，在开始之前的最重要的事情是运行程序看看情况。这可以给你大量的信息：有没有试用时间？是不是有些特性被禁用？是不是只能在有限次数内运行？有没有注册窗口让你输入注册码？

这些都是需要知道的很重要的东西，随着你在逆向领域做得越来越好，你会获得越来越多的经验让你知道应该找什么（需要多长时间来验证注册码？是不是强制你访问一个网站？.....）

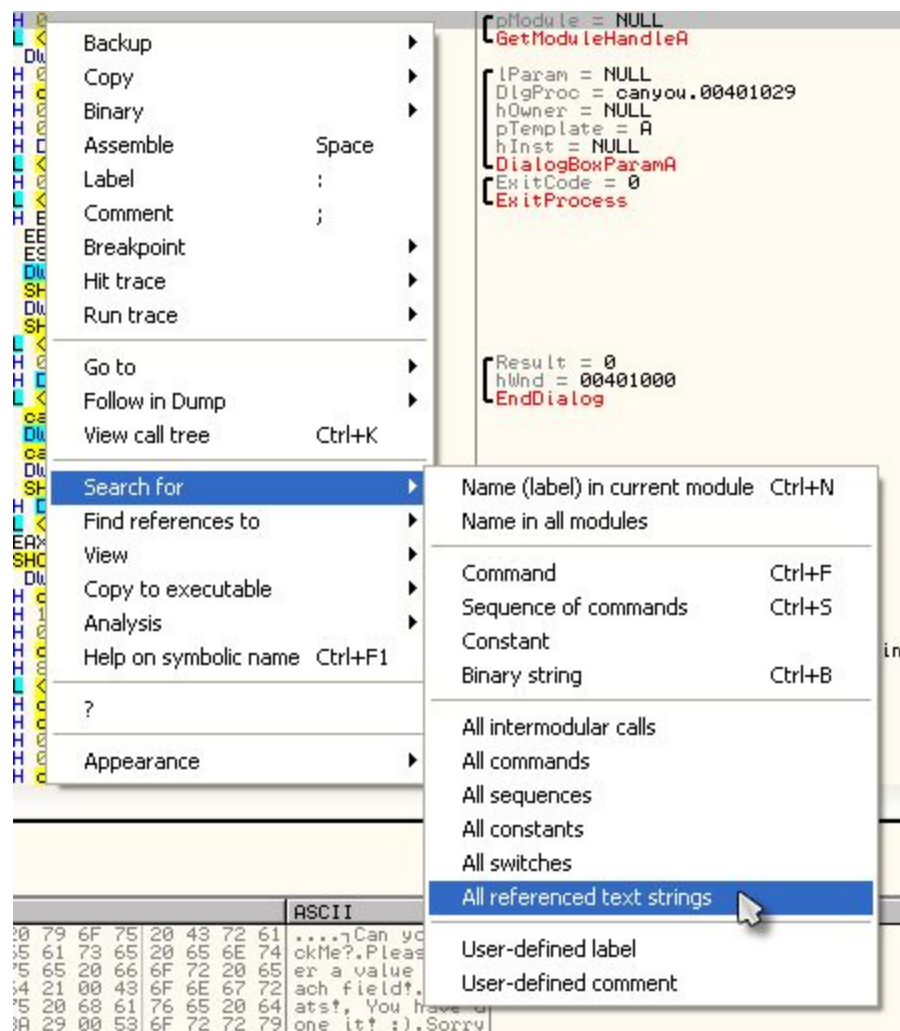
程序运行情况如下：(p2)



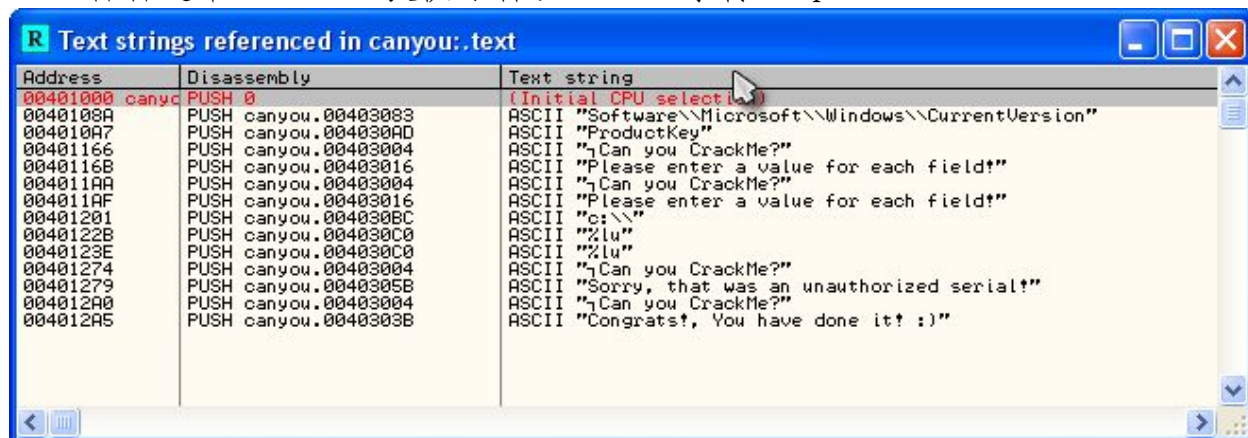
输入一些数据之后的结果： (p3)



现在你应该知道怎么搞了。回到 011y, 看看我们能够查找到哪些字符串:  
(p4)



看看这个 crackme 提供了什么 ASCII 字符: (p5)



好, 这里我们可以很明显的看到几个比较重要的。我们首先注意到的是, 我们必须在每个文本框中都要输入信息: (p6)

```

ASCII "Please enter a value for each field?"
ASCII "Can you CrackMe?"
ASCII "Please enter a value for each field?"
ASCII "c:\\"
```

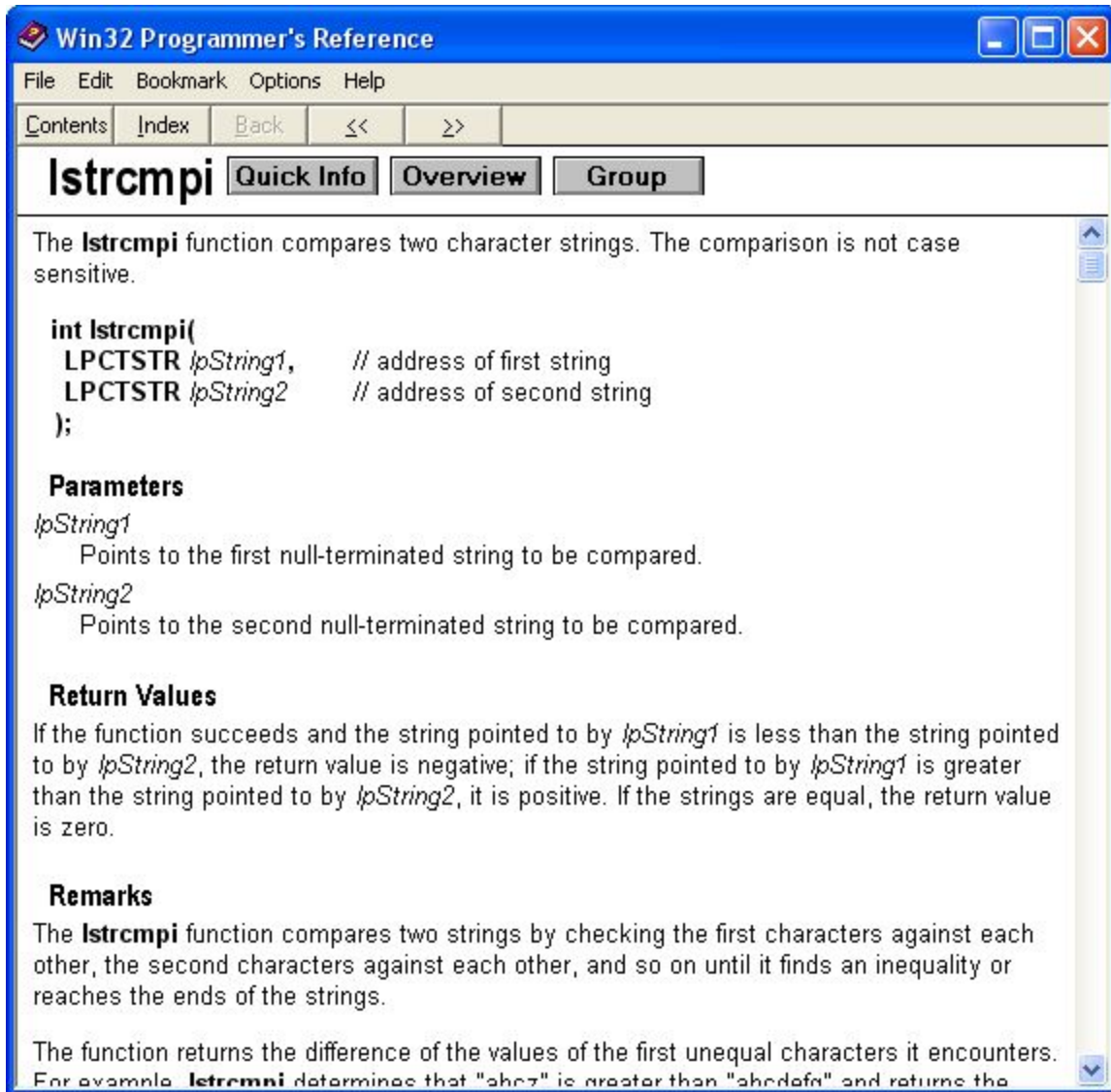
然后，我们才是真正重要的东西：(p7)

```
ASCII "Can you crackMe!"
ASCII "Sorry, that was an unauthorized serial!"
ASCII "Can you CrackMe?"
ASCII "Congrats!, You have done it! :)"
```

双击其中一个，看看我们会去哪：(p8)

00401240	. 83C4 0C	ADD ESP,0C	StringToAdd = ""
00401250	. 68 E83E4000	PUSH canyou.004038E8	ConcatString = ""
00401255	. 68 E83E4000	PUSH canyou.004038E8	lstrcatA
0040125A	. E8 A9000000	CALL <JMP.&KERNEL32.lstrcatA>	String2 = ""
0040125F	. 68 D0324000	PUSH canyou.004032D0	String1 = ""
00401264	. 68 E83E4000	PUSH canyou.004038E8	lstrcmpiA
00401269	. E8 A0000000	CALL <JMP.&KERNEL32.lstrcmpiA>	
0040126E	. 0BC0	OR EAX,EAX	
00401270	. 74 2C	JE SHORT canyou.0040129E	
00401272	. 6A 00	PUSH 0	Style = MB_OK!MB_APPLMODAL
00401274	. 68 04304000	PUSH canyou.00403004	Title = "Can you CrackMe?"
00401279	. 68 5B304000	PUSH canyou.0040305B	Text = "Sorry, that was an unauthorized serial!"
00401280	. 6A 00	PUSH 0	hOwner = NULL
00401280	. E8 65000000	CALL <JMP.&USER32.MessageBoxA>	MessageBoxA
00401285	. 6A 00	PUSH 0	lParam = 0
00401287	. 6A 00	PUSH 0	wParam = 0
00401289	. 6A 10	PUSH 10	Message = WM_CLOSE
0040128B	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	hWnd = 401000
0040128E	. E8 5D000000	CALL <JMP.&USER32.SendMessageA>	SendMessageA
00401293	. B8 00000000	MOV EAX,0	
00401298	. C9	LEAVE	
00401299	. C2 1000	RETN 10	
0040129C	. EB 13	JMP SHORT canyou.004012B1	
0040129E	. 6A 00	PUSH 0	Style = MB_OK!MB_APPLMODAL
004012A0	. 68 04304000	PUSH canyou.00403004	Title = "Can you CrackMe?"
004012A5	. 68 3B304000	PUSH canyou.0040303B	Text = "Congrats!, You have done it! :)"
004012A8	. 6A 00	PUSH 0	hOwner = NULL
004012AC	. E8 39000000	CALL <JMP.&USER32.MessageBoxA>	MessageBoxA
004012B1	. EB 09	JMP SHORT canyou.004012BC	
004012B3	. B8 00000000	MOV EAX,0	
004012B8	. C9	LEAVE	

这个看起来比较熟悉吧：有坏消息部分，紧跟其后的是好消息部分，并且在坏消息的前面有一个非常明显的跳转，想必是跳转到好消息的。这里我想让你注意的是，在跳转的前面有一个对 Windows API 函数 *lstrcmpi* 的 CALL。如果我们在其上右键，选择“Help on symbolic name”，会有如下显示：(p9)



如你所见，`lstrcmp` 是对两个字符串进行比较操作。这个函数在逆向工程领域非常的重要，你会一次又一次的看到。它被用于 注册码/密码 比较机制中，用于比较用户输入的字符串与程序内置的硬编码或被创建的字符串。如果字符串比较返回 0，说明用户的输入是正确的，意味着比较的两个字符串是一样的。如果返回非 0，说明两个字符串不匹配。本例中的 `crackme`，我们输入的字符串可能与一个内置的或动态生成的字符串进行核对，如果 `EAX` 返回的是 0，说明它们是相同的，否则就不相同。现在，01ly 不知道这些字符串是什么，因为我们还没有启动应用，也没有输入任何信息。不过一旦我们开始了，01ly 就会将 `String1=""`、`String2=""` 这两行替换成真正的字符串。如果我们在跳转那里设置一个 BP，然后运行程序，输入一个字符串（本例中是“12121212121212121212”），01ly 就会给我们显示被比较的字符串：(p10)



0040125F	68 00324000	PUSH canyou.004032D0	String2 = "1212121212121212"
00401264	68 E8364000	PUSH canyou.004036E8	String1 = "314216448336430"
00401269	E8 A0000000	CALL <JMP.&KERNEL32.lstrcmpiA>	lstrcmpiA
0040126E	0BC0	OR EAX,EAX	
00401270	74 2C	JE SHORT canyou.0040129E	
00401272	6A 00	PUSH 0	Style = MB_OK!MB_APPLMODAL
00401274	68 04304000	PUSH canyou.00403004	Title = "Can you CrackMe?"
00401279	68 5B304000	PUSH canyou.0040305B	Text = "Sorry, that was an unauthorized serial!"
0040127E	6A 00	PUSH 0	hOwner = NULL
00401280	E8 65000000	CALL <JMP.&USER32.MessageBoxA>	MessageBoxA
00401285	6A 00	PUSH 0	lParam = 0
00401287	6A 00	PUSH 0	wParam = 0
00401289	6A 10	PUSH 10	Message = WM_CLOSE
0040128B	FF75 08	PUSH DWORD PTR SS:[EBP+8]	hWnd = 69032E
0040128E	E8 5D000000	CALL <JMP.&USER32.SendMessageA>	SendMessageA
00401293	E8 00000000	MOV EAX,0	

如果你看跳转指令上面的那几行，你会看到我们的密码与“314216448336430”进行比较，无论它是什么都一样。在返回值上，如果它们相同 EAX 中就是 0，如果不相同就可能是任意值。很明显，本例中，它们不匹配。OR EAX, EAX 是一个判断 EAX 是否为 0 的非常巧妙的方法。如果 EAX 是 0 的话，“JE SHORT canyou.0040129E”就会跳到好消息部分。我之所以给你指出字符串比较部分，是因为在将来的教程中，我们需要找出这 15 个数字是如何被创建出来的。搜索 lstrcmp 能够引导我们找到它的创建过程。

不过现在，我们只做我们知道的。在 401270 处的 JE 指令处设置一个 BP，然后重启程序。输入一个用户名和序列号，01ly 会断在我们的 BP: (p11)

00401264	68 E8364000	PUSH canyou.004036E8	String1 = "303357474363752"
00401269	E8 A0000000	CALL <JMP.&KERNEL32.lstrcmpiA>	lstrcmpiA
0040126E	0BC0	OR EAX,EAX	
00401270	74 2C	JE SHORT canyou.0040129E	
00401272	6A 00	PUSH 0	Style = MB_OK!MB_APPLMODAL
00401274	68 04304000	PUSH canyou.00403004	Title = "Can you CrackMe?"
00401279	68 5B304000	PUSH canyou.0040305B	Text = "Sorry, that was an unauthorized serial!"
0040127E	6A 00	PUSH 0	hOwner = NULL
00401280	E8 65000000	CALL <JMP.&USER32.MessageBoxA>	MessageBoxA
00401285	6A 00	PUSH 0	lParam = 0
00401287	6A 00	PUSH 0	wParam = 0
00401289	6A 10	PUSH 10	Message = WM_CLOSE
0040128B	FF75 08	PUSH DWORD PTR SS:[EBP+8]	hWnd = 69032E
0040128E	E8 5D000000	CALL <JMP.&USER32.SendMessageA>	SendMessageA
00401293	E8 00000000	MOV EAX,0	

通过那个灰色的箭头，我们知道 01ly 不会跳到好消息部分，而是落在坏消息部分。所以咱们来帮帮它吧: (p12)

```

C 0  ES 0023
P 0  CS 001E
A 0  SS 0023
Z 1  DS 0023
S 0  FS 003E
T 0  GS 0000

```

现在，01ly 做对了: (p13)

00401270	74 2C	JE SHORT canyou.0040129E	
00401272	6A 00	PUSH 0	Style = MB_OK!MB_APPLMODAL
00401274	68 04304000	PUSH canyou.00403004	Title = "Can you CrackMe?"
00401279	68 5B304000	PUSH canyou.0040305B	Text = "Sorry, that was an unauthorized serial!"
0040127E	6A 00	PUSH 0	hOwner = NULL
00401280	E8 65000000	CALL <JMP.&USER32.MessageBoxA>	MessageBoxA
00401285	6A 00	PUSH 0	lParam = 0
00401287	6A 00	PUSH 0	wParam = 0
00401289	6A 10	PUSH 10	Message = WM_CLOSE
0040128B	FF75 08	PUSH DWORD PTR SS:[EBP+8]	hWnd = 69032E
0040128E	E8 5D000000	CALL <JMP.&USER32.SendMessageA>	SendMessageA
00401293	E8 00000000	MOV EAX,0	
00401296	C2 1000	RETN 10	
00401299	EB 13	JMP SHORT canyou.004012B1	
0040129E	6A 00	PUSH 0	Style = MB_OK!MB_APPLMODAL
004012A0	68 04304000	PUSH canyou.00403004	Title = "Can you CrackMe?"
004012A5	68 3B304000	PUSH canyou.0040303B	Text = "Congrats!, You have done it! :)"
004012A9	6A 00	PUSH 0	hOwner = NULL
004012AC	E8 39000000	CALL <JMP.&USER32.MessageBoxA>	MessageBoxA
004012B1	EB 09	JMP SHORT canyou.004012B0	
004012B3	B8 00000000	MOV EAX,0	
004012B8	C9	LEAVE	

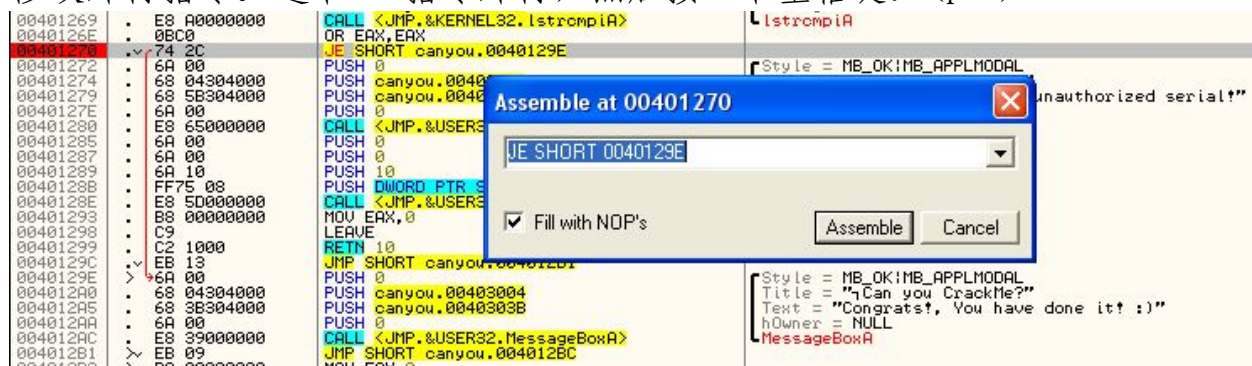
为了确定下，咱们运行程序看看: (p14)



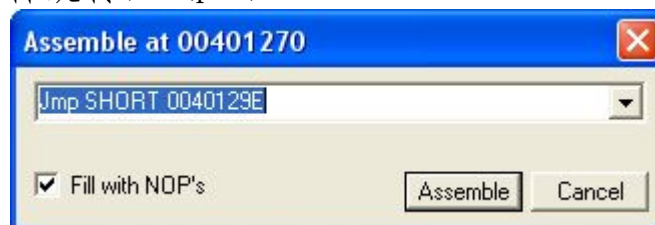
现在，让我们.....

### 三、给程序打补丁

这回我不打算将跳转 NOP 掉，因为这样会让程序每一次都显示坏消息。相反，我想要确保跳转每一次都成功，跳转到我们好消息部分。转到设置 BP 的那行（如果你找不到的话，打开“Breakpoint Window”，然后在 BP 上双击），修改那行指令。选中 JE 指令那行，然后按一下空格键：(p15)



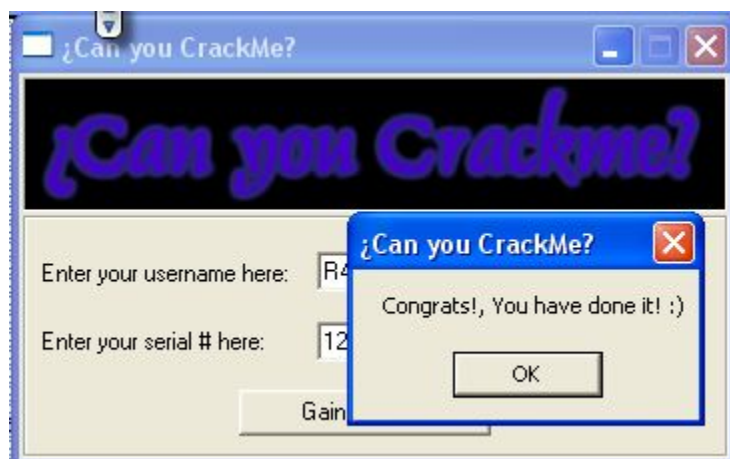
注意我们选中的指令已经在文本框中了。现在，我们将 JE (Jump on Equal) 修改成 JMP (无条件跳转)：(p16)



点击那个 Assemble 按钮，然后点 Cancel 按钮。你就会发现我们的修改已经放到了代码中：(p17)

00401269	:	E8 A0000000	CALL <JMP.&KERNEL32.lstrcmpiA>
0040126E	:	0BC0	OR EAX,EAX
00401270	✓	EB 2C	JMP SHORT canyou.0040129E
00401272	:	6A 00	PUSH 0
00401274	:	68 04304000	PUSH canyou.00403004
00401279	:	68 5B304000	PUSH canyou.0040305B

现在，运行下程序以确保没什么问题：(p18)



现在，咱们将打过补丁的程序保存到磁盘。要记住，如果你重启应用的话，你需要重新启用补丁（Patch 窗口中，选中补丁再按一下空格键），不过我们的程序还在运行，只需要点一下 Ollly，右键反汇编窗口，选择“Copy to executable” -> “All modifications”：(p19)

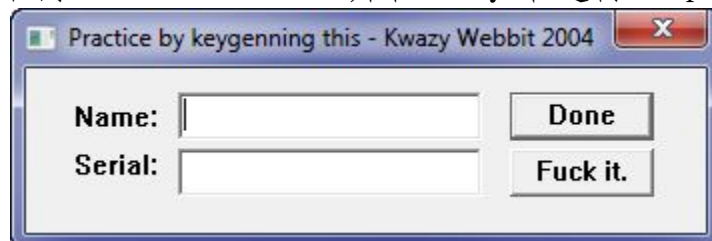




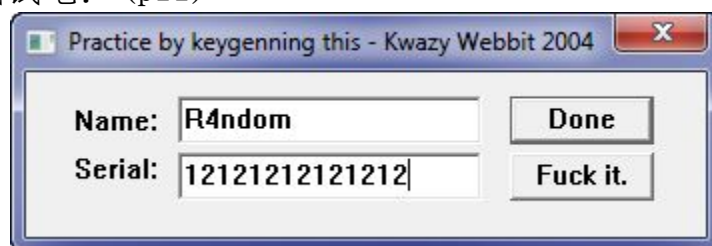
011y 并允许。如果你不想这么做的话，事实上你再也不用将其载入 011y 了。因为补丁已经被保存到磁盘，你可以从任何地方运行它。你要你运行的是打过补丁的就行😁。现在，无论你输入什么名字和序列号，都会弹出好消息窗口😁。

#### 四、另一个 crackme

载入第二个程序 Crackme8.exe，并在 011y 中运行：(p21)



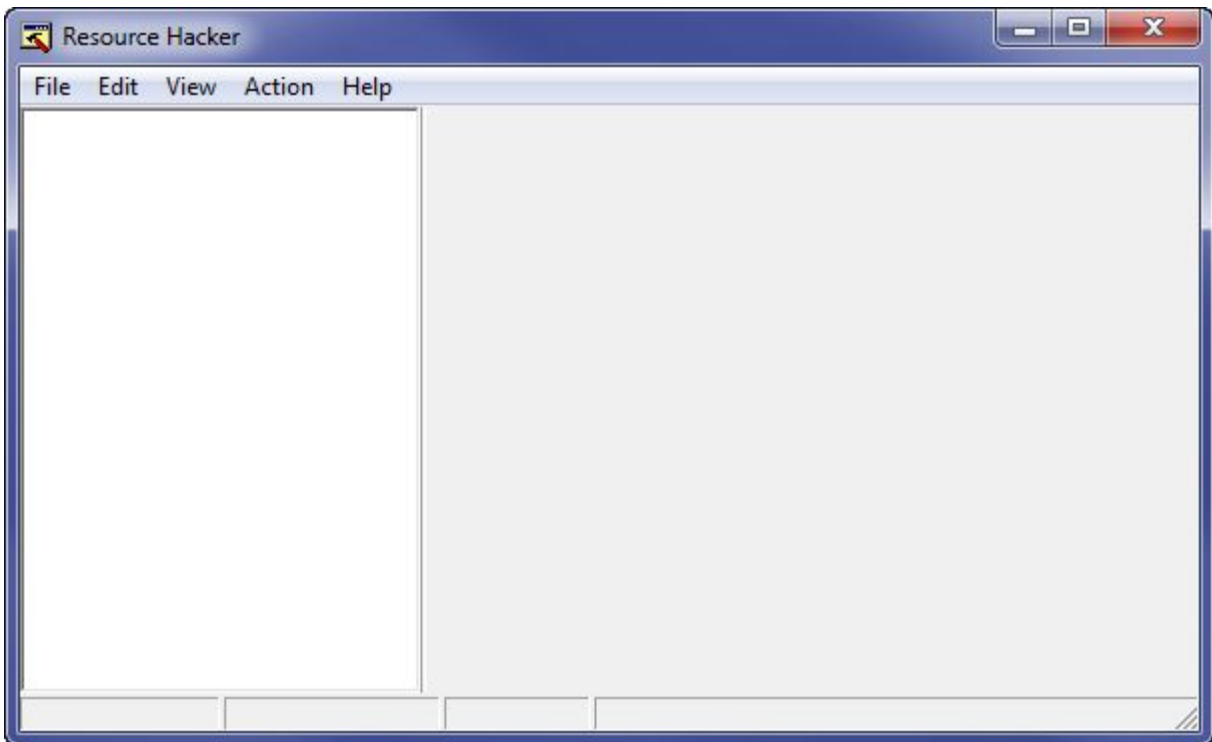
好吧，这里有点点疑惑：0。嗯，在输入了用户名和密码后，我该点哪个按钮呢？好吧，试试吧：(p22)



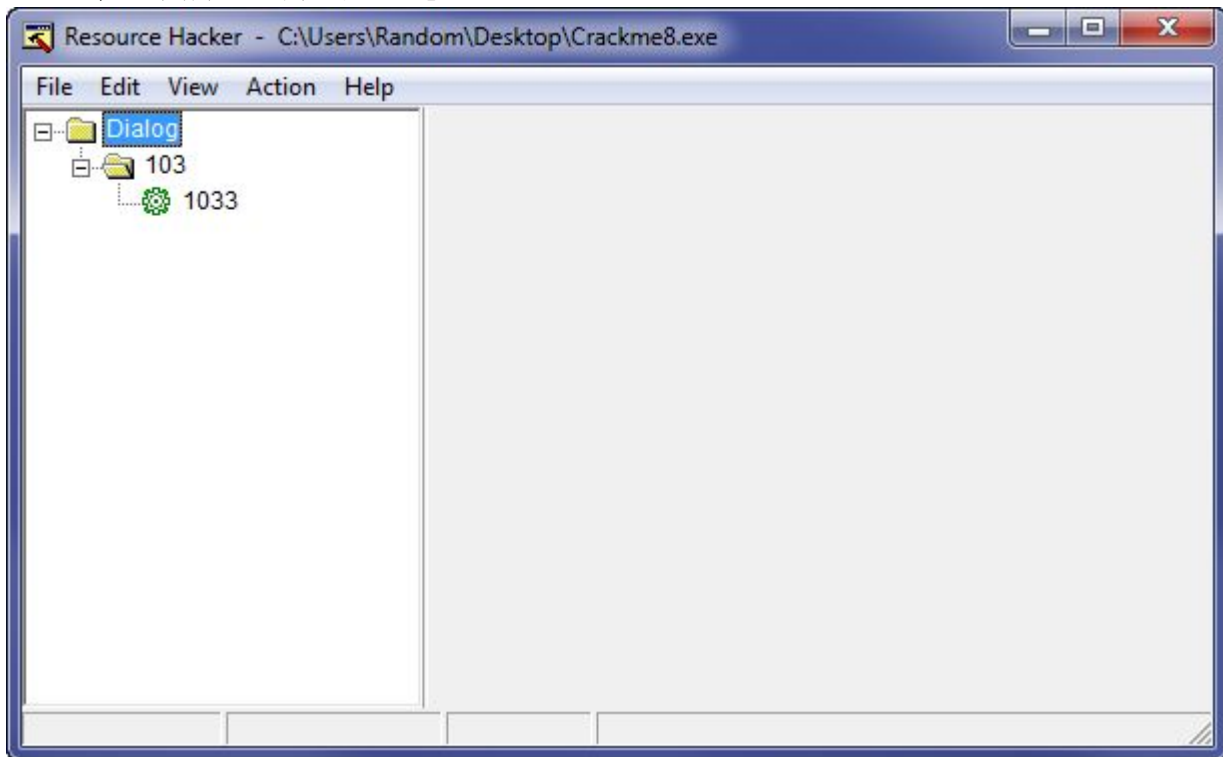
这里，Done 通常意味着退出，所以我试试另一个。嗯.....，程序退出了。很明显我应该点 Done 的（？）。不管了，借此机会咱们改改程序，做些有趣的事。咱们将按钮改成更加有意义的“Check”和“Done”，或者是任何你喜欢的都行😁。

#### 五、使用 Resource Hacker

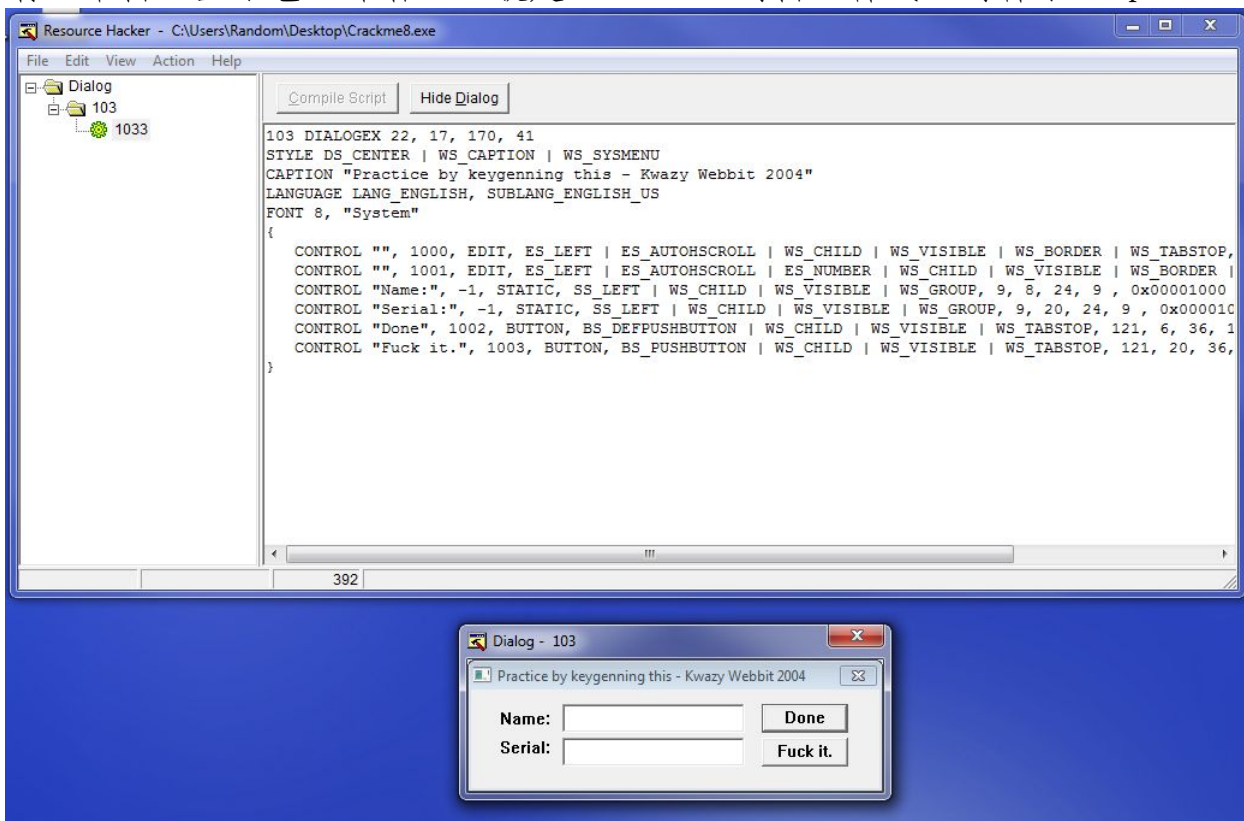
如果你还没准备好，先安装 Resource Hacker。第一次运行如下所示：(p23)



将 Crackme8 载入到 Resource Hacker，你就会看到一个叫 Dialog 的文件夹，它旁边有个+号。展开+号，点一下下一个文件夹（103）边上的+号，你会看到如下所示的内容：（p24）



现在，点那个 1033，然后右边面板就会显示对话框的相关数据，同时会有一个窗口显示它（译者注：就是 crackme8 的窗口样式）的样子：(p25)



在右侧面板的顶部，你可以看到一些关于窗口的数据，比如字体、标题、类型等等：(p26)

```
103 DIALOGEX 22, 17, 170, 41
STYLE DS_CENTER | WS_CAPTION | WS_SYSMENU
CAPTION "Practice by keygenning this - Kwazy Webbit 2004"
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
FONT 8, "System"
```

在下面你可以看到对话框中所有元素的细节，包括“Name”、“Serial”标签和两个按钮。咱们把这个对话框修改成我们喜欢的，好不好？首先将两个按钮的名字修改成“Check”和“Exit”：(p27)

```
CONTROL "Serial:", -1, STATIC, SS_
CONTROL "Check", 1002, BUTTON, BS_
CONTROL "Exit.", 1003, BUTTON, BS_
```

现在，我们修改顶部的标题：(p28)

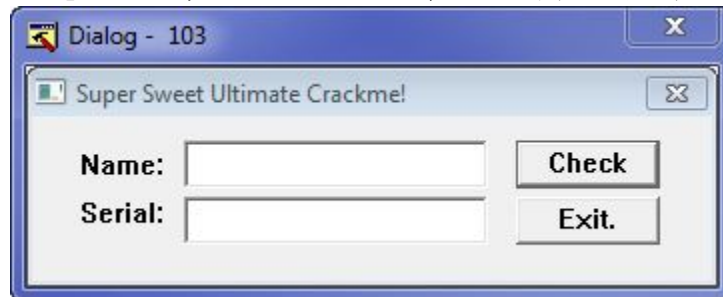


```

103 DIALOGEX 22, 17, 170, 41
STYLE DS_CENTER | WS_CAPTION | WS_SYSMENU
CAPTION "Super Sweet Ultimate Crackme!"
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
FONT 8, "System"
{
    CONTROL "", 1000, EDIT, ES_LEFT | ES_AUT

```

现在点击“Compile”按钮，就会看到我们的窗口更新了：(p29)



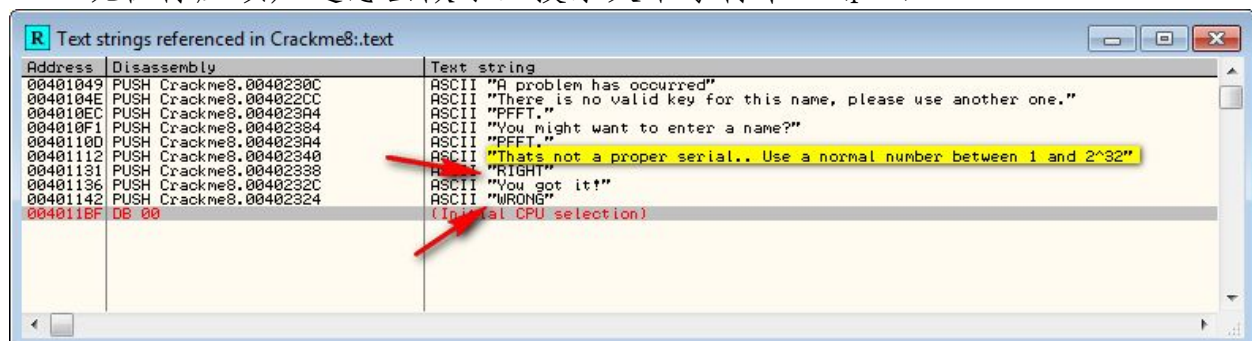
好，相当不错。将其保存 (“File” -> “Save”), 将新的 crackme 载入 Olly (原始的 crackme 被 Resource Hacker 以 Crackme8-original 名字保存), 运行它：(p30)



啊，相当好。现在我们正式开始...

## 六、破解程序

现在你应该知道怎么做了。搜索文本字符串：(p31)



我们了解了两件事：1) 序列号必须是一个 1 到非常大的数字；2) 我们知道了好消息和坏消息生成的地方。咱们转到好消息那：(p32)

```
*G.P.U* - main thread, module Crackme8
0040110F . 56 PUSH ESI
0040110F . 68 E9030000 PUSH 3E9
0040110F . 57 PUSH EDI
0040110F . FF15 14204000 CALL DWORD PTR DS:[<&USER32.GetDlgItemInt>]
0040110F . 3BC5 CMP EAX,ESI
0040110F . 75 14 JNS SHORT Crackme8.00401120
0040110F . 56 PUSH ESI
0040110F . 68 A4234000 PUSH Crackme8.00402304
0040110F . 68 40234000 PUSH Crackme8.00402340
0040110F . 57 PUSH EDI
0040110F . FF15 10204000 CALL DWORD PTR DS:[<&USER32.MessageBoxA>]
0040110F . EB A9 JMP SHORT Crackme8.0040110C9
0040110F . 50 PUSH EAX
0040110F . 8D45 CC LEA EAX,[LOCAL_13]
0040110F . 50 PUSH EAX
0040110F . 68 06FEFFFF CALL Crackme8.00401000
0040110F . 85C0 TEST EAX,EAX
0040110F . 59 POP ECX
0040110F . 59 POP ECX
0040110F . 56 PUSH ESI
0040110F . 74 0C JE SHORT Crackme8.0040113D
0040110F . 68 38234000 PUSH Crackme8.00402338
0040110F . 68 2C234000 PUSH Crackme8.0040232C
0040110F . EB DA JMP SHORT Crackme8.00401117
0040110F . A1 84234000 MOV EAX,DWORD PTR DS:[4023B4]
0040110F . 68 06234000 PUSH Crackme8.00402324
0040110F . 50 PUSH 0
0040110F . 59 POP ECX
0040110F . 59 POP ECX
0040110F . 50 PUSH 0
0040110F . F7F9 IDIV ECX
0040110F . FF3495 28204000 PUSH DWORD PTR DS:[EDX*4+402028]
0040110F . 57 PUSH EDI
0040110F . FF15 10204000 CALL DWORD PTR DS:[<&USER32.MessageBoxA>]
0040110F . FF05 B4234000 INC DWORD PTR DS:[4023B4]
0040110F . 83D0 B4234000 CMP DWORD PTR DS:[4023B4],0B
0040110F . 0F85 5BFFFFFF JNB Crackme8.004010C9
0040110F . 56 PUSH ESI
0040110F . 57 PUSH EDI
0040110F . E9 4EFFFFFF JMP Crackme8.004010C3
0040110F . 8B45 08 MOV EAX,[ARG_0]
0040110F . A3 0C234000 MOV DWORD PTR DS:[4023AC],EAX
0040110F . E9 47FFFFFF JMP Crackme8.004010C9
0040110F . 6A 00 PUSH 0
0040110F . 68 83104000 PUSH Crackme8.00401083
0040110F . 50 PUSH 0
0040110F . 6A 00 PUSH 0
0040110F . 6A 00 PUSH 0
0040110F . FF15 04204000 CALL DWORD PTR DS:[<&KERNEL32.GetModuleHandleA>]
0040110F . 50 PUSH EAX
0040110F . FF15 20204000 CALL DWORD PTR DS:[<&USER32.DialogBoxParamA>]
0040110F . 6A 00 PUSH 0
0040110F . FF15 00204000 CALL DWORD PTR DS:[<&KERNEL32.ExitProcess>]
0040110F . CC INT3

pSuccess = 7EFD0000
ControlID = SE9 (1001.)
hWnd = 0018EE6C
GetDlgItemInt

ASCII "PFFT."
ASCII "That's not a proper serial... Use a normal number between 1
hOwner = 0018EE6C
MessageBoxA

0018EF28
0018EF28

ASCII "RIGHT"
ASCII "You got it!"

Title = "WRONG"
0018EF28

Text = "Now now, that wasn't even close.\nYou gotta try harder."
hOwner = 0018EE6C
MessageBoxA

Case 110 (WM_INITDIALOG) of switch 0040108C

iParam = NULL
DlgProc = Crackme8.00401083
hOwner = NULL
pTemplate = 67
Module = NULL
GetModuleHandleA
hInst = NULL
DialogBoxParamA
ExitCode = 0
ExitProcess
```

双击进到相关领域。我们看到好消息的路径是从 401131 开始的，坏消息从 40113D 开始。我们看到那个跳转指令（JE SHORT Crackme8.0040113D）在 401131 处，比较指令（TEST EAX, EAX）在 40112A 处。咱们在 40112F 处设置 BP，然后运行程序。输入用户名和序列号后点击“Check”。01ly 随后断在了我们的断点处：(p33)

```
*G.P.U* - main thread, module Crackme8
00401124 . 56 PUSH EAX
00401124 . E8 06FEFFFF CALL Crackme8.00401000
00401124 . 85C0 TEST EAX,EAX
00401124 . 59 POP ECX
00401124 . 59 POP ECX
00401124 . 56 PUSH ESI
00401124 . 74 0C JE SHORT Crackme8.0040113D
00401124 . 68 38234000 PUSH Crackme8.00402338
00401124 . 68 2C234000 PUSH Crackme8.0040232C
00401124 . EB DA JMP SHORT Crackme8.00401117
00401124 . A1 84234000 MOV EAX,DWORD PTR DS:[4023B4]
00401124 . 68 06234000 PUSH Crackme8.00402324
00401124 . 50 PUSH 0
00401124 . 59 POP ECX
00401124 . 59 POP ECX
00401124 . 50 PUSH 0
00401124 . F7F9 IDIV ECX
00401124 . FF3495 28204000 PUSH DWORD PTR DS:[EDX*4+402028]
00401124 . 57 PUSH EDI
00401124 . FF15 10204000 CALL DWORD PTR DS:[<&USER32.MessageBoxA>]
00401124 . FF05 B4234000 INC DWORD PTR DS:[4023B4]
00401124 . 83D0 B4234000 CMP DWORD PTR DS:[4023B4],0B
00401124 . 0F85 5BFFFFFF JNB Crackme8.004010C9
00401124 . 56 PUSH ESI
00401124 . 57 PUSH EDI
00401124 . E9 4EFFFFFF JMP Crackme8.004010C3
00401124 . 8B45 08 MOV EAX,[ARG_0]
00401124 . A3 0C234000 MOV DWORD PTR DS:[4023AC],EAX
00401124 . E9 47FFFFFF JMP Crackme8.004010C9
00401124 . 6A 00 PUSH 0
00401124 . 68 83104000 PUSH Crackme8.00401083
00401124 . 50 PUSH 0
00401124 . 6A 00 PUSH 0
00401124 . 6A 00 PUSH 0
00401124 . FF15 04204000 CALL DWORD PTR DS:[<&KERNEL32.GetModuleHandleA>]
00401124 . 50 PUSH EAX
00401124 . FF15 20204000 CALL DWORD PTR DS:[<&USER32.DialogBoxParamA>]
00401124 . 6A 00 PUSH 0
00401124 . FF15 00204000 CALL DWORD PTR DS:[<&KERNEL32.ExitProcess>]
00401124 . CC INT3

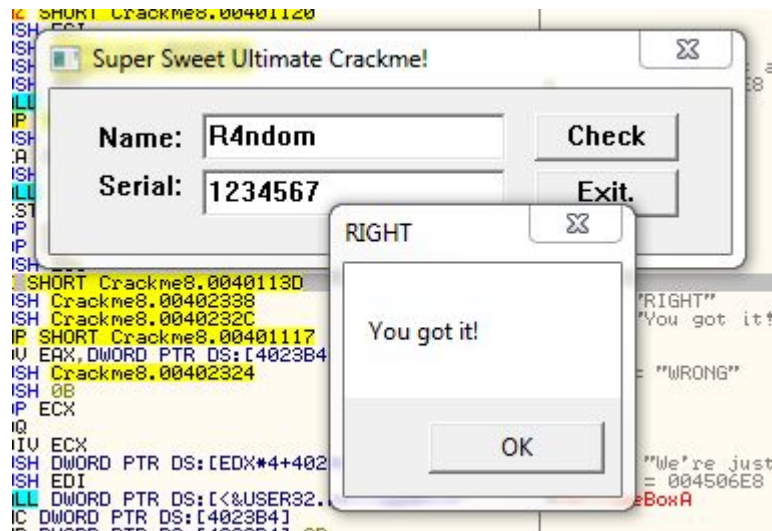
pSuccess = 7EFD0000
ControlID = SE9 (1001.)
hWnd = 0018EE6C
GetDlgItemInt

ASCII "RIGHT"
ASCII "You got it!"

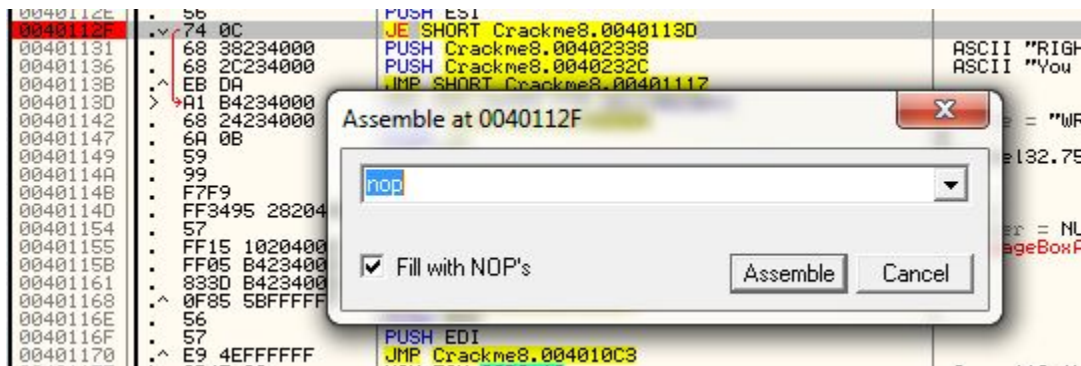
Title = "WRONG"

Text = "We're just as far as when we started :("
hOwner = 004506E8 ('Super Sweet Ultimate Crackme!',class='#
MessageBoxA
```

我们可以看到，01ly 依旧要跳过好消息部分，直达坏消息部分。你知道了那个路径...，清除 0 标志位运行程序：(p34)



成功了！现在咱们快速的创建一个补丁。重启应用，找到断点（通过断点窗口），在 JE 指令上点一下，再按一下空格键，NOP 掉跳转指令，这样我们就能够一直的直达好消息部分：(p35)



先点 “Assemble” 然后是 “Cancel”。右键然后选择 “Save to executable” -> “All modifications”，再选择 “Copy all”。右键弹出的窗口，选择 “Save file” 保存它。现在你有了一个打过补丁的并且修改过资源的 crackme，你输入的任何序列号都会让他显示好消息 😊。

## 七、值得思考的事

我想说的是，Resource Hacker 是一个有意思的非常有用的小程序。通过它你不仅仅可以修改一个文件的许多东西（字符串、图标、标签、按钮、标题），你也可以用它修改 Windows 自身的许多东西（开始按钮、上下文菜单、计算机的 “关于” 对话框等）。事实上，Resource Hacker 正是我的版本的 01ly 的图标修改工具！