第九章：无相关字符串

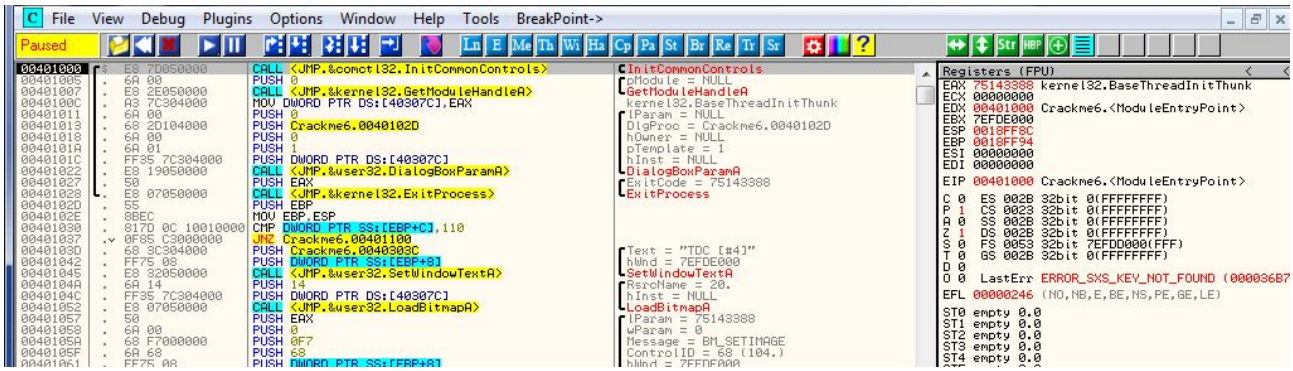
一、简介

此次教程中我将会向我们的武器库中加入一个新的装备。如果搜索二进制文件时发现没有可用的字符串你怎么办？我将会介绍一个新的 R.E.T.A.R.D. 规则。此次教程(下一章也是)我们将研究“TDC”写的一个 crackme 叫 Crackme6，相关下载里面包含有。总之，它不是一个硬骨头，不过我将会对其进行一些高级分析，好为将来的教程做准备。

你可以在教程页下载相关文件以及本教程的 PDF 版本。

那么，咱们开始吧.....

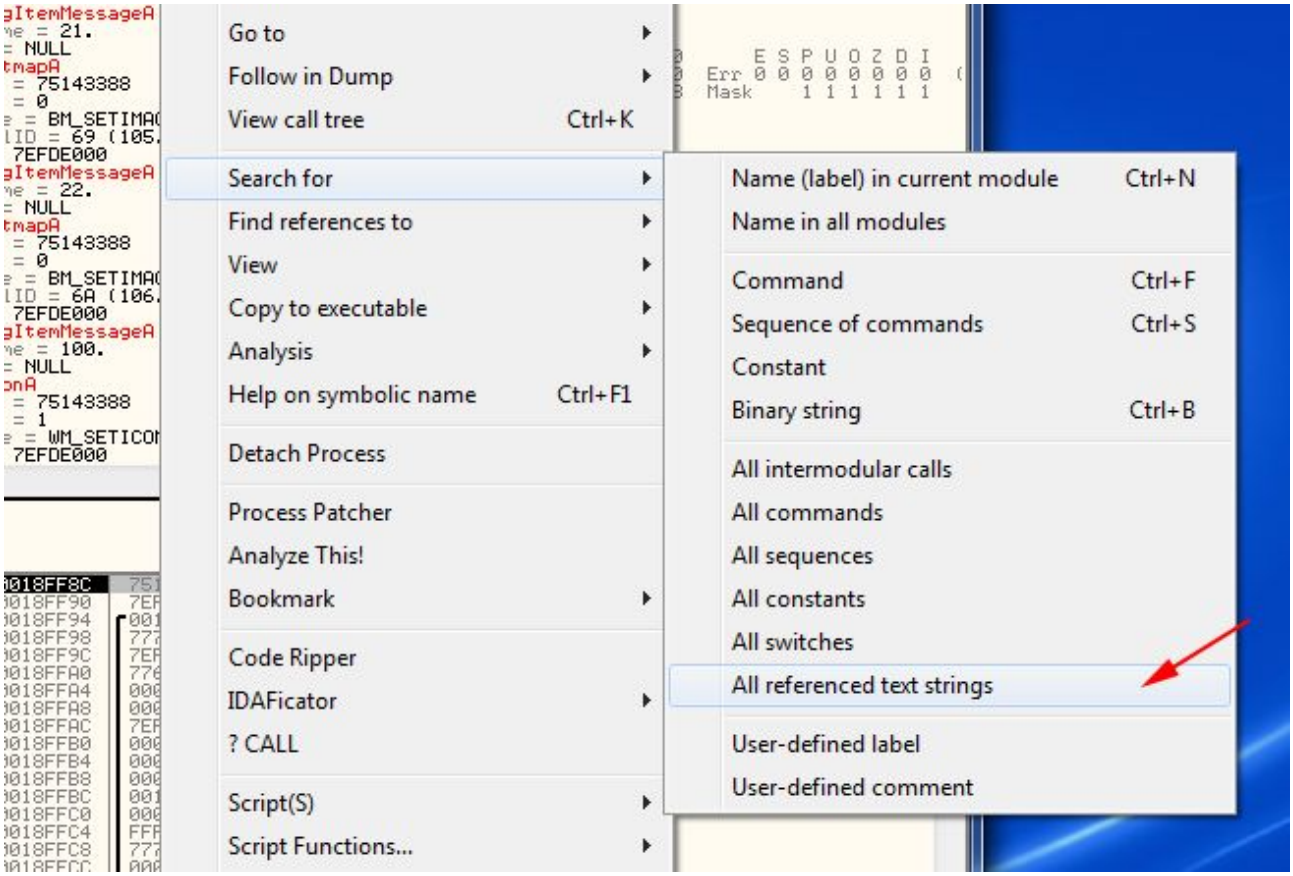
OllyDbg 载入 Crackme6: (p1)



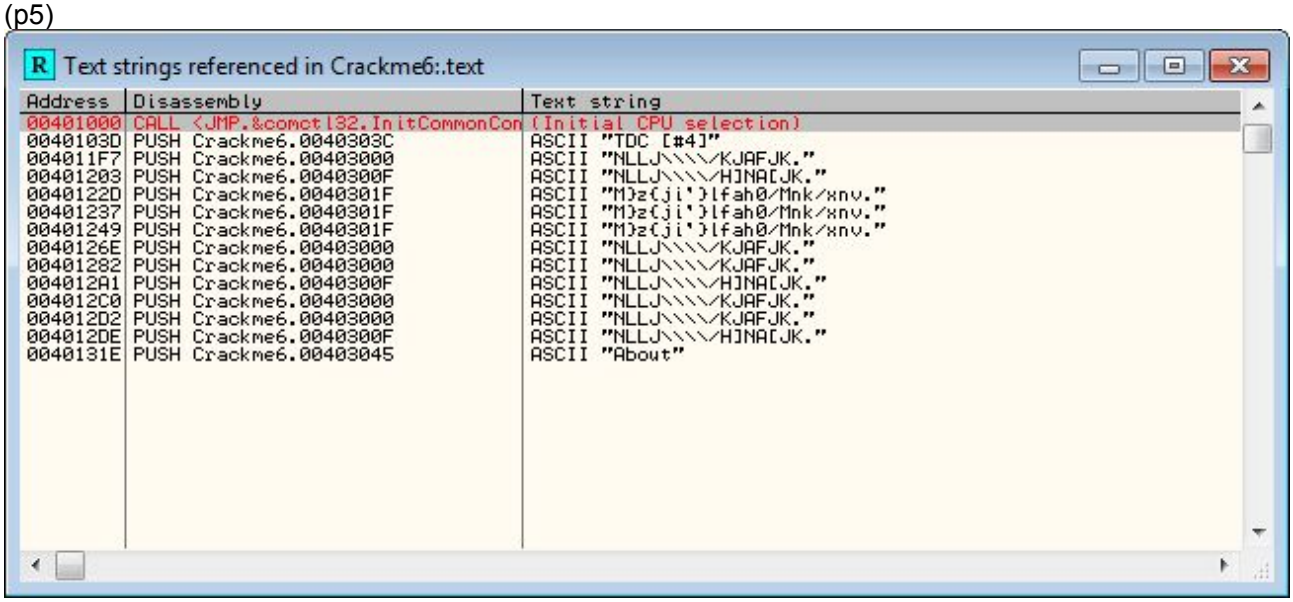
现在，我们已经知道操作程序了。运行程序看看情况: (p2)

嗯，看起来挺简单的。我输入了一个密码 1212121212，下面就是返回的情况: (p3)

相当直接呀。试试我们拿手的“字符串搜索”，看看有什么：(p4)



(p5)



搞什么鬼这是!!! 这些一点用也没有啊😟。我们可以拿这些字符串干啥！？！？明显，这个 crackme 将字符串加密了（或者是作者说一种很奇怪的语言:D）。好，是个好时候介绍

R4ndom's Essential Truths About Reversing Data #3:
R4ndom 关于逆向数据的必备真理#3：

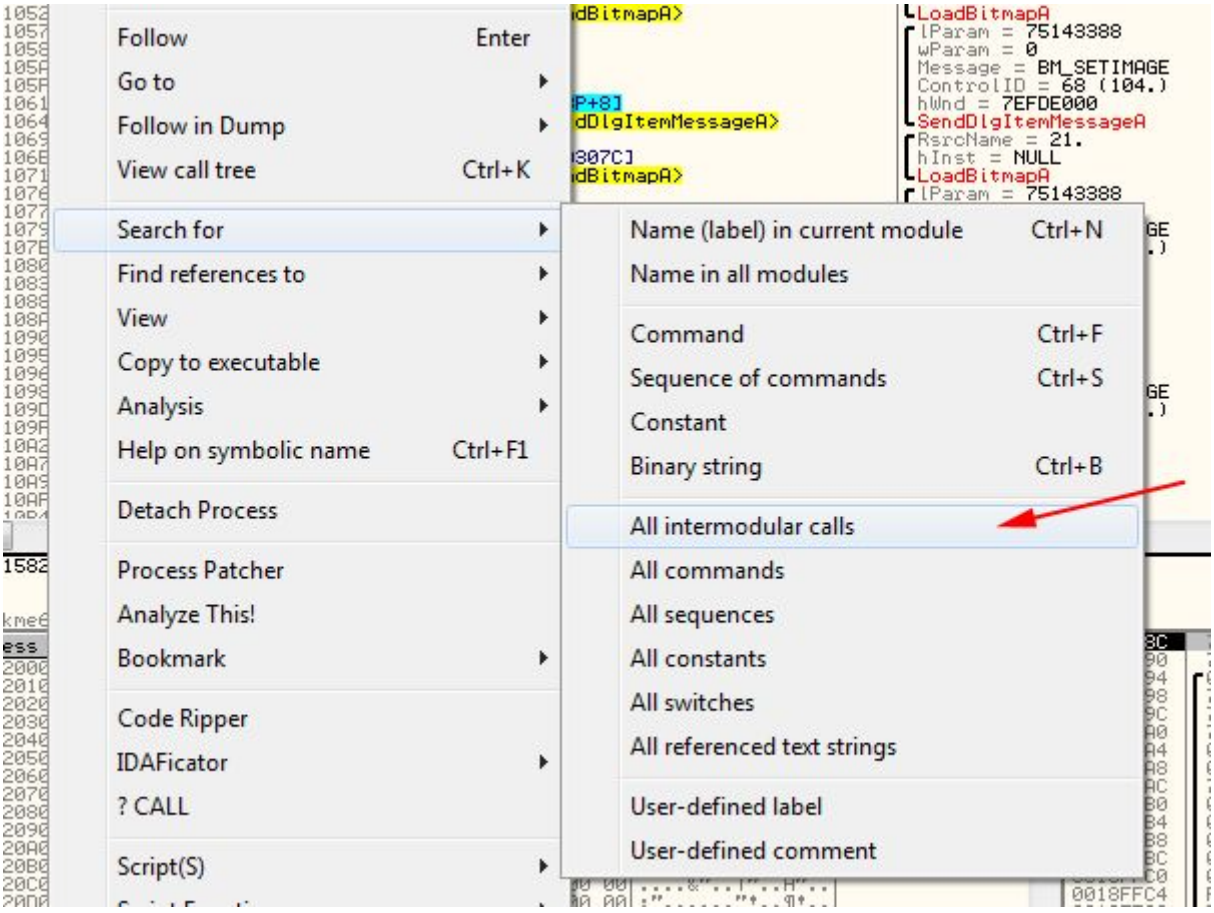
#3.不要依赖二进制文件当前已有的字符串。

不幸的是，在你开始研究真正的二进制文件（比如商业产品）时，它们中的大部分被以某种方式打包 以及/或 保护。干扰逆向工程师的一个最明显的方法是加密字符串。坦率地说，在逆向工程领域当我第一次研究一个感兴趣的新的二进制文件时，如果我搜索字符串并且搜出来了，我能够假定那个二进制文件很可能没有多少挑战。所以，你不能够依赖于那些东西（如果有当然更好🙂）。

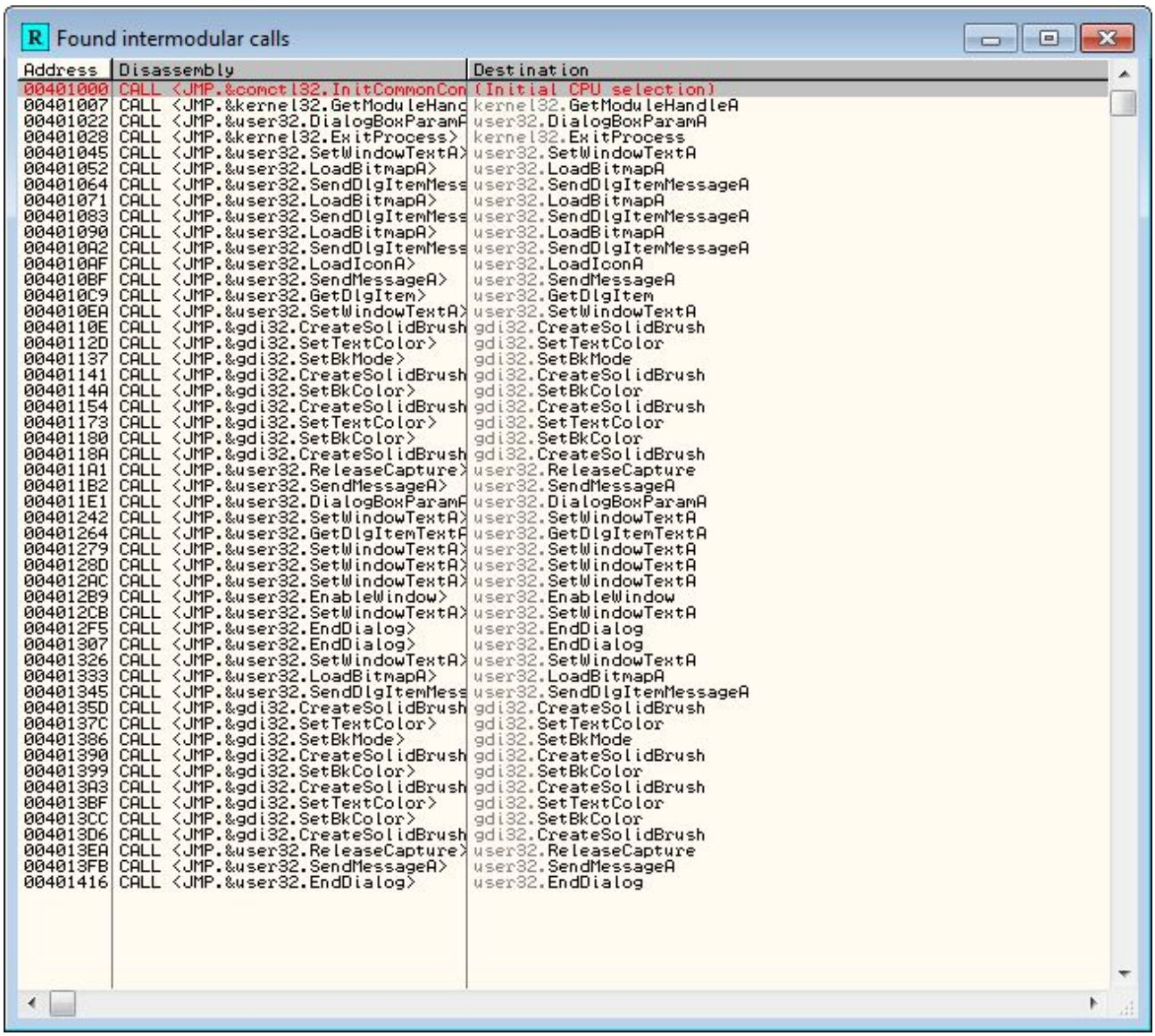
二、模块间的调用

有鉴于此，我向你展示一个新的在没有字符串的情况下的技巧。大部分的 Windows 应用程序使用一个标准的 API 集来完成特定的动作。例如，如果需要一个简单的消息框的话就调用 MessageBoxA，当程序想要退出的时候就调用 TerminateProcess。因为大部分的应用都使用这些相同的 API，我们可以用这个获利。例如，有些 API 可以用于从对话框的输入框（类似用户名和序列号）获取文本。有可以被调用用来比较两个字符串的字符串比较函数（输入的密码和程序中存储的密码相同吗？）。有读写注册表的 API（存储和读取你的注册状态）。

Olly 提供了一种搜索所有被调用的 API 的方法。在反汇编窗口右键，选择“Search for” -> “All intermodular calls”：(p6)

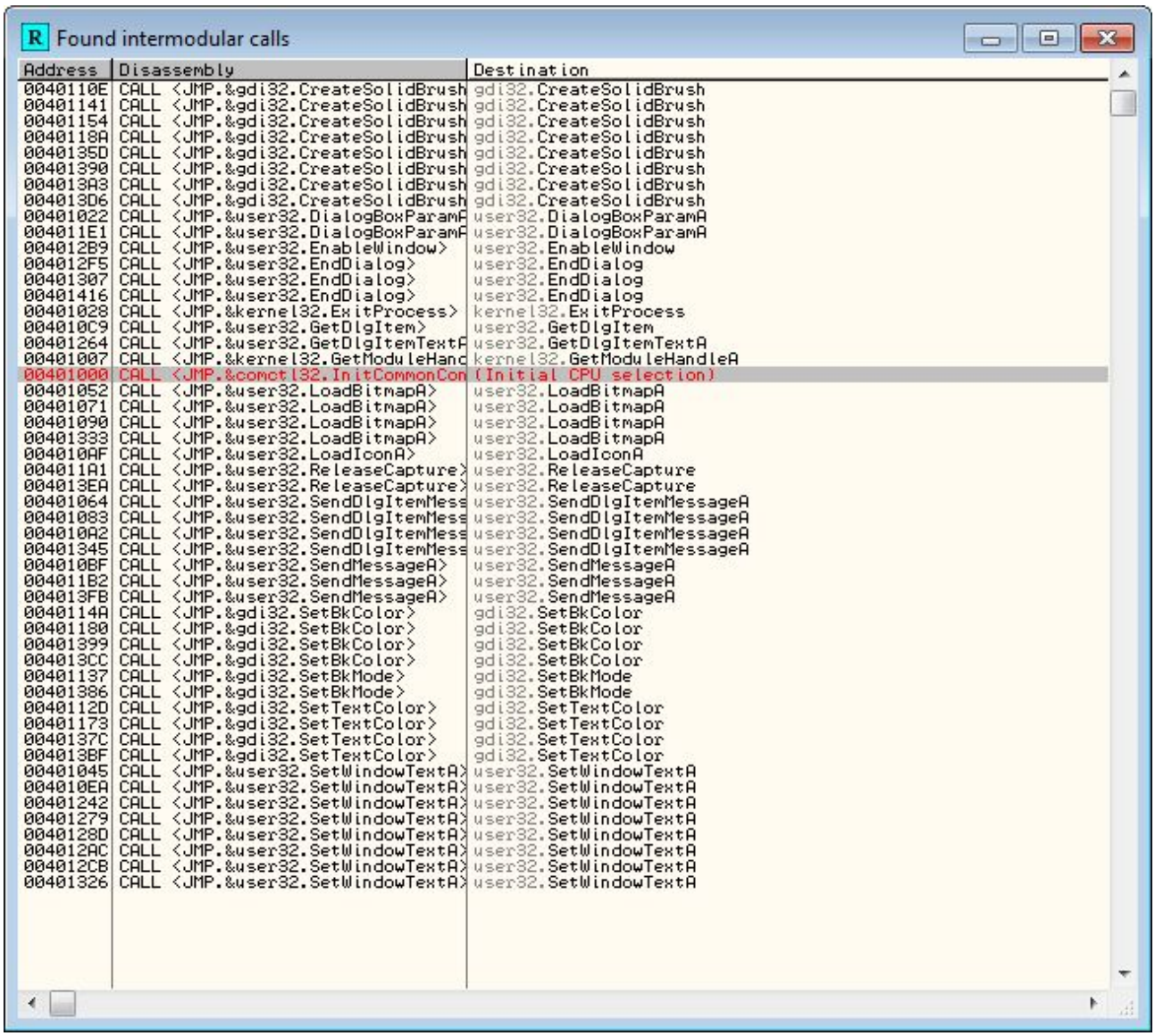


Olly 会弹出 Found intermodular calls 窗口：(p7)



通常我做的第一件事是点一下“Destination”，将列出的函数按字母顺序进行排序（而不是按地址排序）：(p8)

现在，如果你看第三列的话，你可以看到该 crackme 调用的所有 API：(p9)



这是一个小程序，所以调用的函数不是那么多。大部分的程序都有数百个。不过通过这个列表，你可以了解到一个二进制文件的很多信息。你可以发现它用一个对话框作为主窗口。它载入了一个自定义的位图。它修改了对话框中的一些颜色。

在更大些的应用中，这个窗口的价值更高，因为它能告诉你这些事情：1）是否有注册表相关 API 被调用用来存储和获取信息？是不是有 API 呼叫网站来验证我们确实注册了？3）有没有读写一个可能存有注册码文件的 API？当我们研究一个加壳的二进制文件时，这个窗口将更加重要（这个后面讨论）。

尽管如此，有几个特定 API 逆向工程师总是会留意，因为这几个在保护机制中用的比较多。包括：

DialogBoxParamA
GetDlgItem
GetDlgItemInt
GetDlgTextA

GetWindowTextA
GetWindowWord

LoadStringA
lstrcmpA

wsprintfA

MessageBeep
MessageBoxA
MessageBoxExA
SendMessageA
SendDlgItemMessageA

ReadFile
WriteFile
CreateFileA

GetPrivateProfileIntA
WritePrivateProfileStringA
GetPrivateProfileStringA

不幸的是，这里没有包括你可能遇到的所有 API，不过幸运的是，大部分应用使用下面的其中一个:

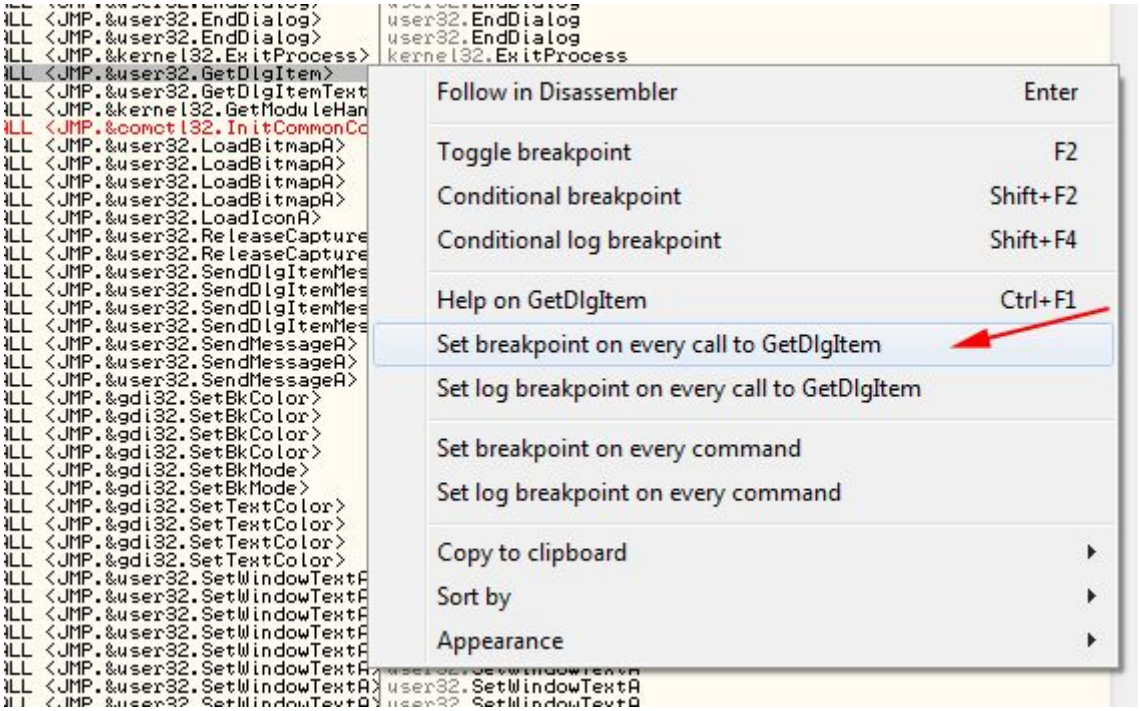
GetDlgItemTextA
GetWindowTextA
lstrcmpA
GetPrivateProfileStringA
GetPrivateProfileIntA
RegQueryValueExA
WritePrivateProfileStringA
GetPrivateProfileIntA

如果你专注这 8 个 API 的调用，你就可以处理绝大多数的实例。还有别忘了，“Get help on symbolic name”是可以给你提供帮助的。

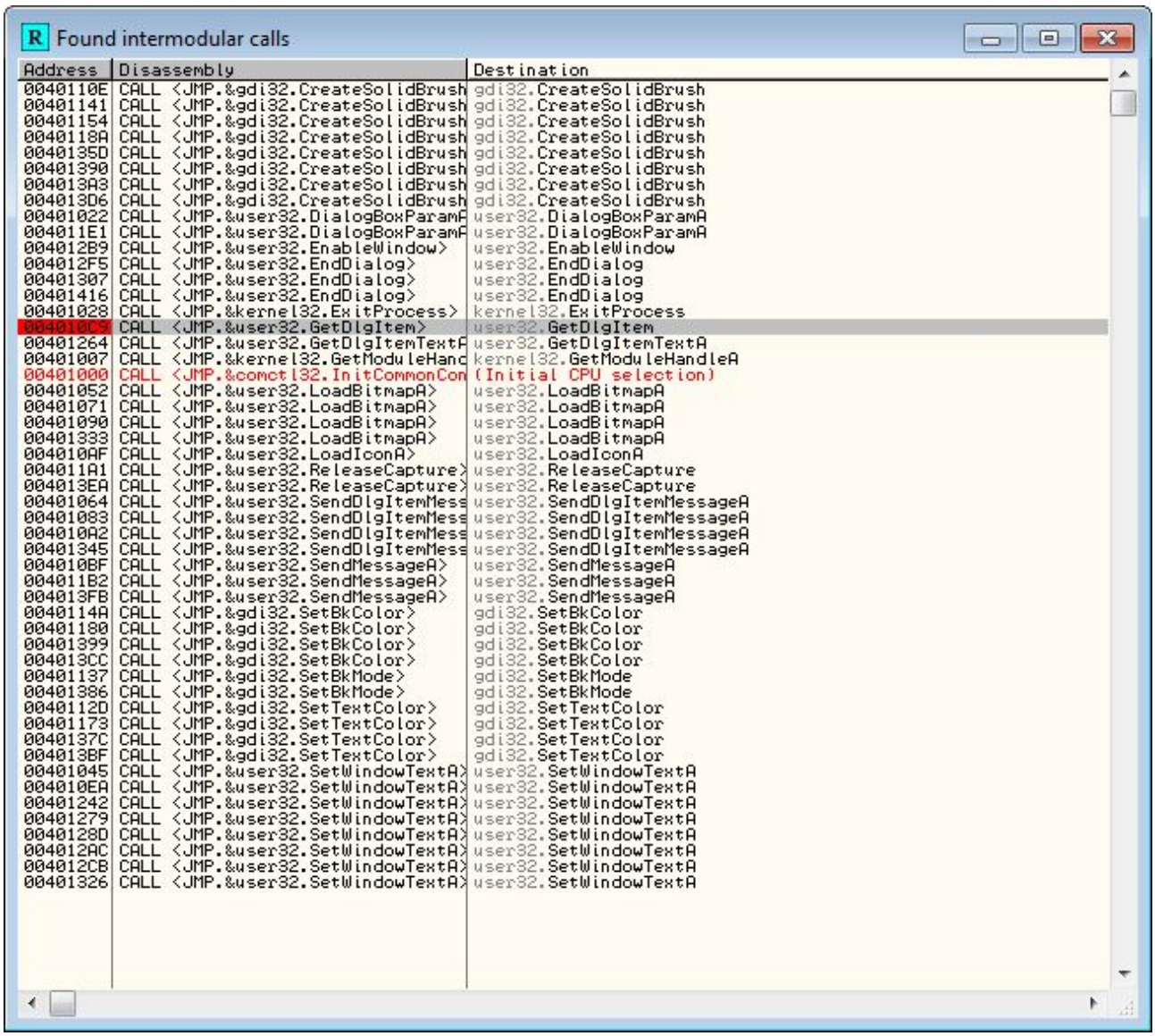
现在，在 Olly 查找出的 crackme 的调用列表中往下看，在那个简短的列表中有两个 API:

GetDlgItem 和 **GetDlgItemTextA**

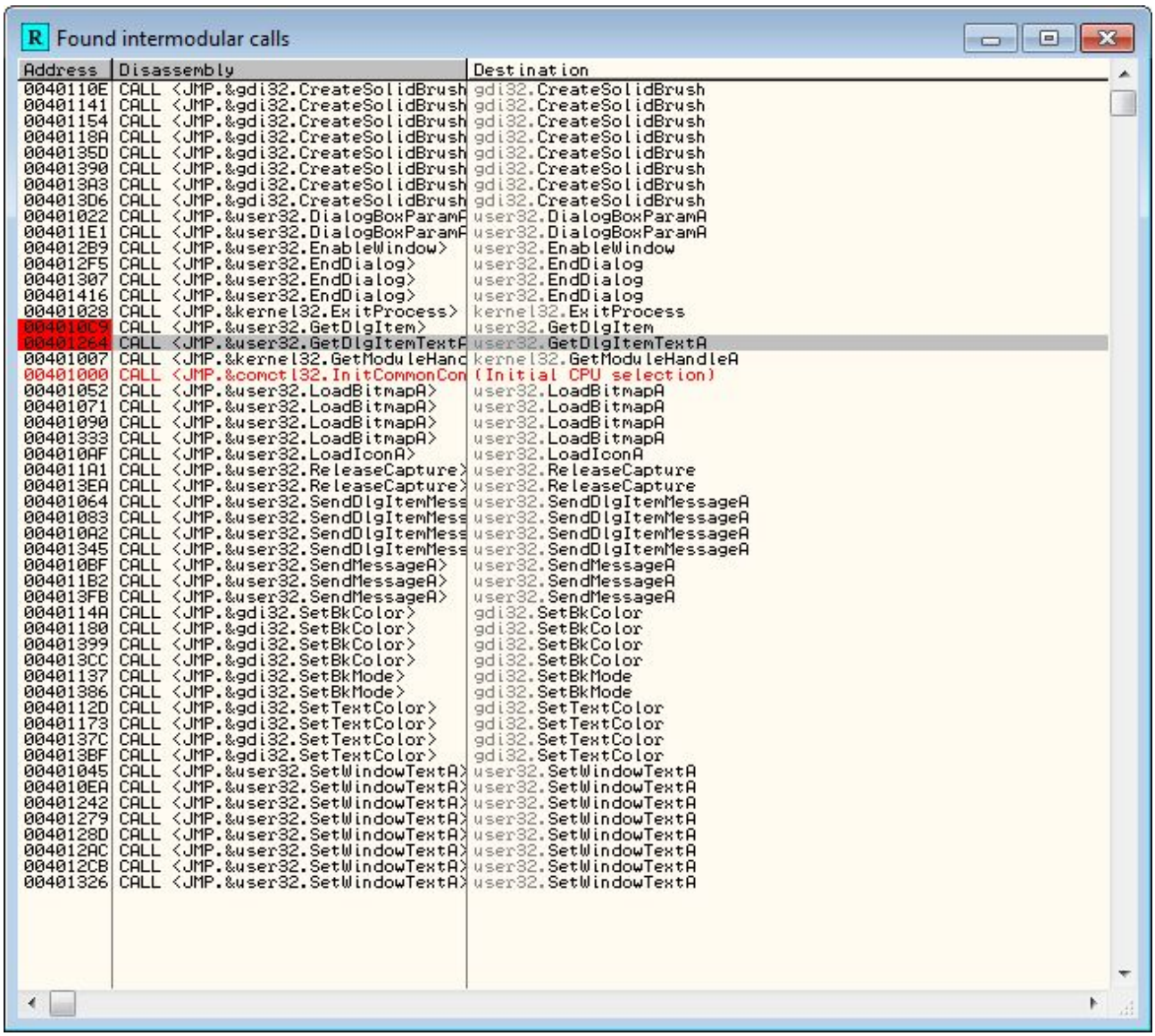
这两个函数是用来获取输入到对话框中文本框的文本。好吧，在我们的教程中，这只可能说明一件事，获取我们输入的密码。我们想要做的是，不管什么时候只要 Olly 遇到两者中的一个就暂停。方法是，选中你想要关注的 API 那行，右键然后选择 “Set breakpoint on every call to ____”，这里的____是 API 的名称（这里是 GetDlgItem）：（p10）



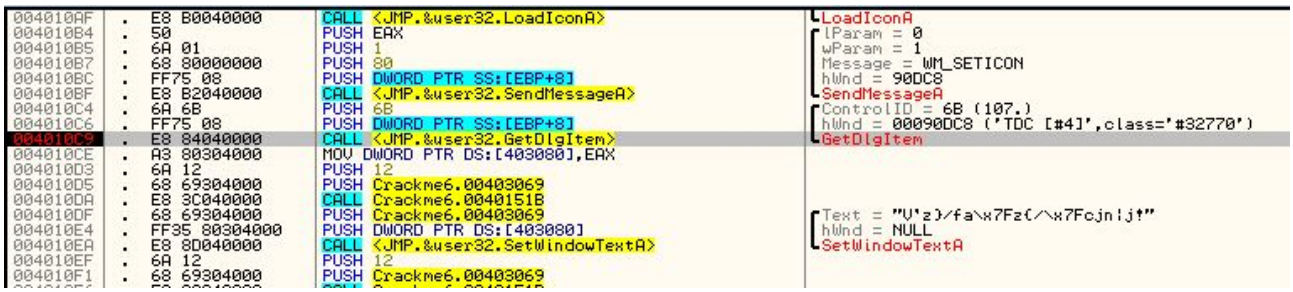
现在，我们看到 Olly 已经在该行设置了一个 BP：（p11）



我们也想在另一个 API GetDlgItemTextA 那暂停，那么点击选中它，右键然后和前面一样进行操作: (p12)



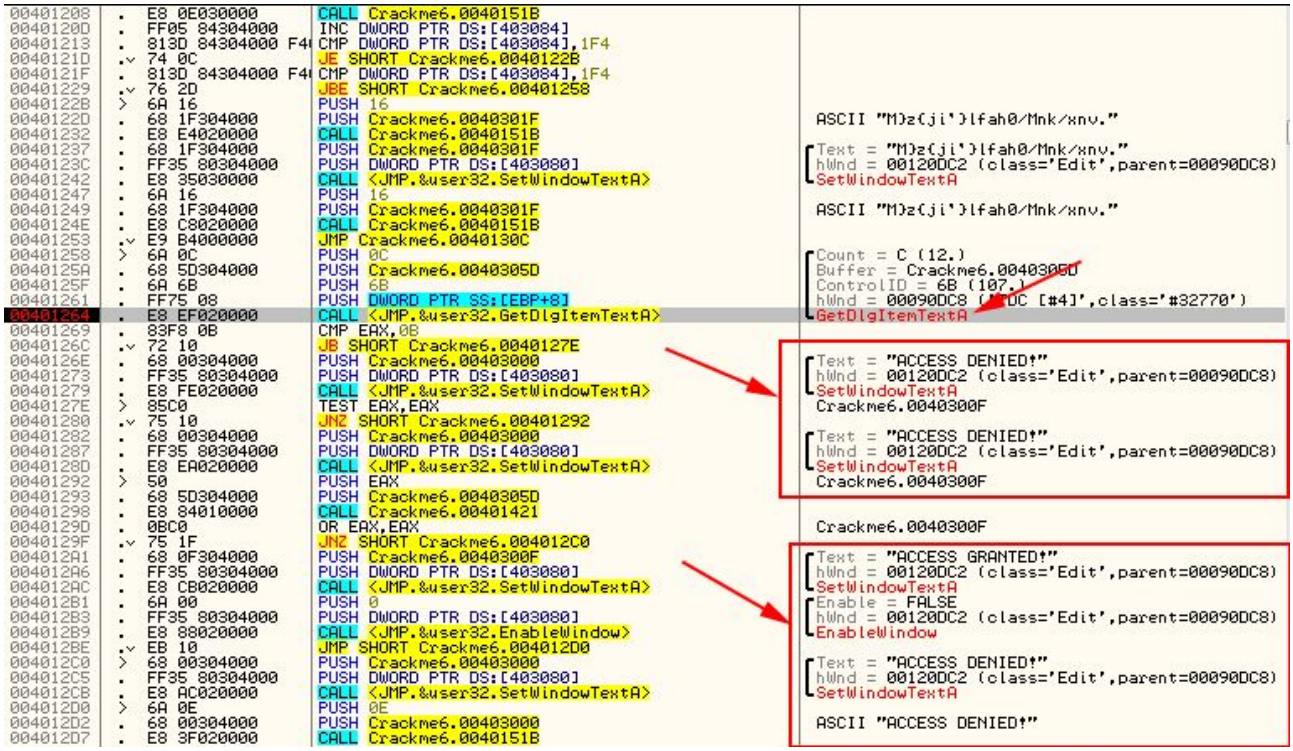
现在，不管什么时候只要 O11y 遇到了对这两个 API 的调用，它都会断下来（在调用被执行前）。咱们来试试看。重启 crackme 并运行。O11y 会断在对 GetDlgItem 的调用处：（p13）



现在，因为我们还没有输入任何内容，我们对 GetDlgItem 取到了什么东西不感兴趣，好咱们继续（F9）：（p14）

现在输入一个密码，然后点“check”：(p15)

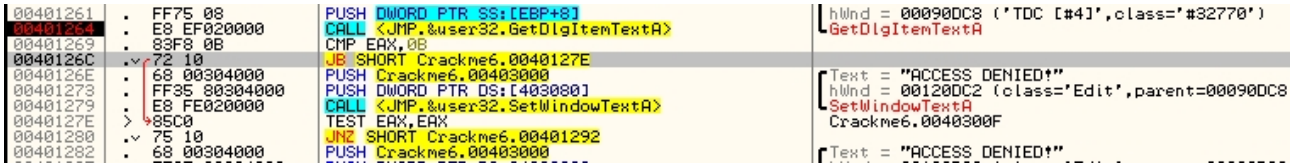
Olly 会再次断下来，这次是在 GetDlgItemTextA：(p16)



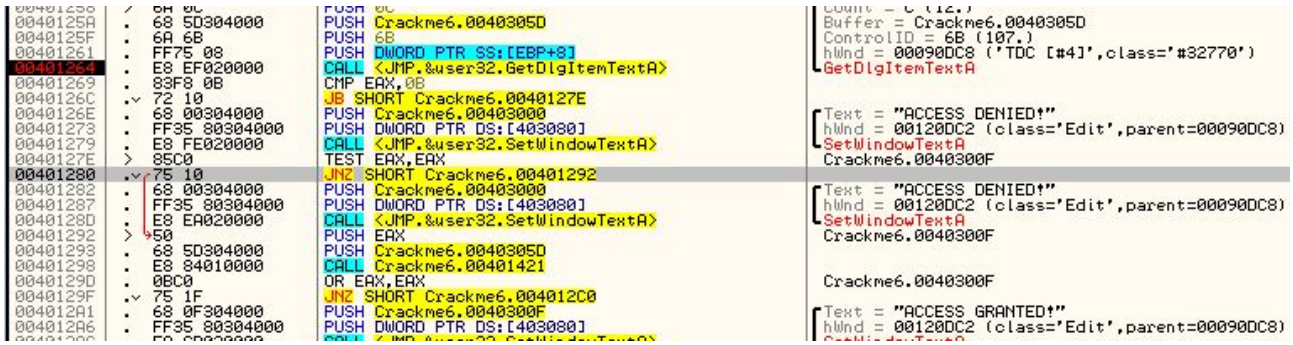
如果你看看周围，你会注意到我们已经来到正确的位置。搞笑的是，我们起初搜索字符串的时候，没有一个是这些字符串中的。

三、破解应用

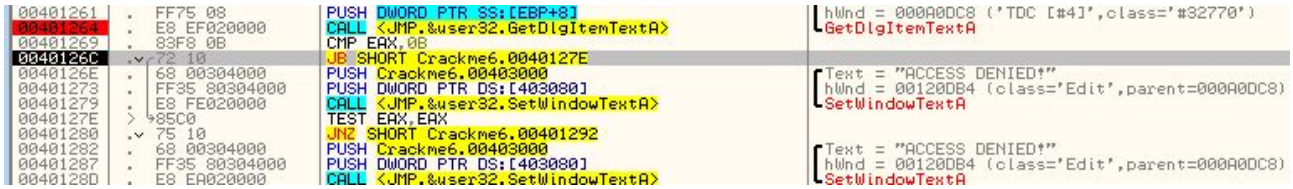
咱们快速浏览下附近的....。我们注意到有一个跳转（JB）跳过了第一个“ACCESS DENIED”，所以我们关注一下它：(p17)



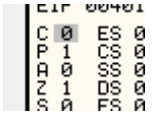
有一个跳转（JNZ）跳过了第二个坏消息，所以我们也将其加入关注名单。然后我们就会直接穿过到达好消息，所以基本上我们想要确保我们跳过了这两个跳转：(p18)



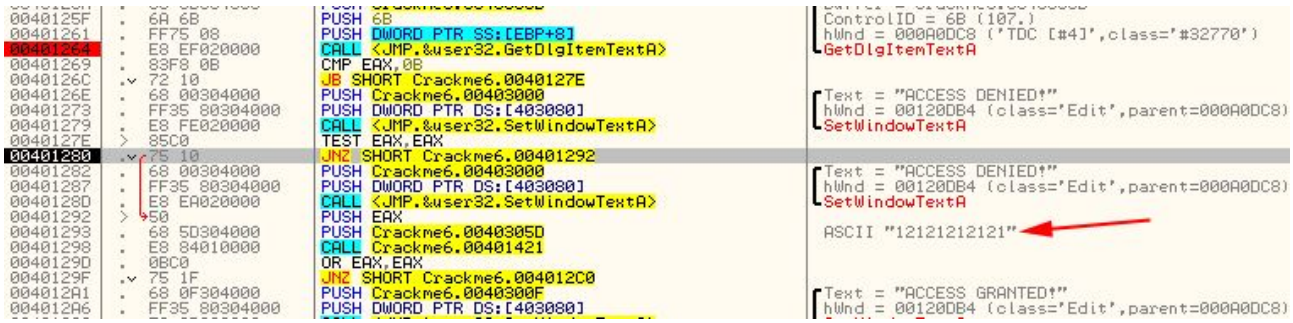
咱们试试，看看咱们是不是对的。再一次运行程序，我们应该断在 GetDlgItemTextA 指令处（记住绕过第一个断点）：(p19)



因为这是一个 JB 跳转，所以我们需要翻转进位标志位：(p20)



这样就会强制跳转。现在我们将做另一个 TEST，停在了 401280 处的跳转那。注意，我们的密码已经出现了注释列😀：(p21)



我们想要那个跳转实现，因为它跳过了第二个坏消息，所以我们只需要继续单步直到到达 40129F 的 JNZ 指令：(p22)

```
0040125F   .  6A 6B         PUSH 6B                                  ControlID = 6B (107.)
00401261   .  FF75 08       PUSH DWORD PTR SS:[EBP+8]                hWnd = 000A0DC8 ('TDC [#4]',class='#32770')
00401264   .  E8 EF020000   CALL <JMP.&user32.GetDlgItemTextA>       GetDlgItemTextA
00401269   .  83F8 0B       CMP EAX,0B
0040126C   .v 72 10         JB SHORT Crackme6.0040127E
0040126E   .  68 00304000   PUSH Crackme6.00403000                   Text = "ACCESS DENIED!"
00401273   .  FF35 80304000 PUSH DWORD PTR DS:[403080]               hWnd = 00120DB4 (class='Edit',parent=000A0DC8)
00401279   .  E8 FE020000   CALL <JMP.&user32.SetWindowTextA>       SetWindowTextA
0040127E   >  85C0          TEST EAX,EAX
00401280   .v 75 10         JNZ SHORT Crackme6.00401292
00401282   .  68 00304000   PUSH Crackme6.00403000                   Text = "ACCESS DENIED!"
00401287   .  FF35 80304000 PUSH DWORD PTR DS:[403080]               hWnd = 00120DB4 (class='Edit',parent=000A0DC8)
0040128D   .  E8 EA020000   CALL <JMP.&user32.SetWindowTextA>       SetWindowTextA
00401292   >  50            PUSH EAX
00401293   .  68 5D304000   PUSH Crackme6.0040305D
00401298   .  E8 84010000   CALL Crackme6.00401421
0040129D   .  0BC0          OR EAX,EAX
0040129F   .v 75 1F         JNZ SHORT Crackme6.004012C0
004012A1   .  68 0F304000   PUSH Crackme6.0040300F                   Text = "ACCESS GRANTED!"
004012A6   .  FF35 80304000 PUSH DWORD PTR DS:[403080]               hWnd = 00120DB4 (class='Edit',parent=000A0DC8)
004012AC   .  E8 CB020000   CALL <JMP.&user32.SetWindowTextA>       SetWindowTextA
004012B1   .  6A 00         PUSH 0                                   Enable = FALSE
004012B3   .  FF35 80304000 PUSH DWORD PTR DS:[403080]               hWnd = 00120DB4 (class='Edit',parent=000A0DC8)
004012B9   .  E8 88020000   CALL <JMP.&user32.EnableWindow>          EnableWindow
004012BE   .v EB 10         JMP SHORT Crackme6.004012D0
004012C0   >  68 00304000   PUSH Crackme6.00403000                   Text = "ACCESS DENIED!"
004012C5   .  FF35 80304000 PUSH DWORD PTR DS:[403080]               hWnd = 00120DB4 (class='Edit',parent=000A0DC8)
004012CB   .  E8 AC020000   CALL <JMP.&user32.SetWindowTextA>       SetWindowTextA
004012D0   >  6A 0E         PUSH 0E
004012D2   .  68 00304000   PUSH Crackme6.00403000                   ASCII "ACCESS DENIED!"
004012D7   .  E8 3F020000   CALL Crackme6.0040151B
```

好，这条指令将会跳过我们的好消息，所以我们想要阻止它跳。你知道该怎么做：(p23)

```
C 0  ES
P 0  CS
A 0  SS
Z 1  DS
S 0  FS
```

现在运行程序（F9），看看我们已经成功的破解了程序：(p24)

四、家庭作业

作为一个挑战，试着给这个 crackme 打补丁，基于那个我们已经修改的标志位。在将打过补丁的程序保存后，你应该可以运行它，输入任何密码（少于 11 个数字）它都会提示 “Access Granted”。记住有几个补丁可以完成这个任务，所以如果一个不起作用，那就找下一个。

加分题：给 crackme 打补丁，让它接受任意长的密码。