2009

[Cracking with OllyDbg]

Based on OllyDbg tuts of Ricardo Narvaja (CrackLatinos Team)

www.reaonline.net

kienmanowar

12/30/2009

Mục Lục

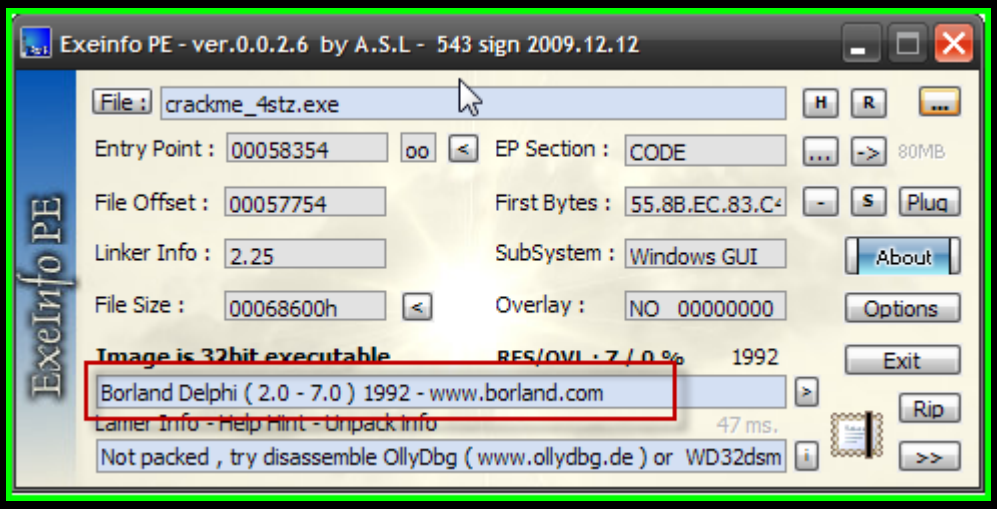
I. Giới thiệu chung

Ở phần 17, tôi có đề cập tới kĩ thuật tìm kiếm đoạn code quan trọng dựa vào Window Messages, cụ thể là sử dụng WM_KEYUP. Để có thể bắt được thông điệp WM_KEYUP thì chúng ta phải nhờ đến một hàm API qua trọng của Windows là **TranslateMessage**. Chức năng và công dụng của hàm này thế nào thì các bạn tự tìm hiểu nhé. Ở bài viết này, tôi sẽ cùng các bạn thực hành trên target - là một crackme được code bởi lão StzWei (trong CrackLatinos Team thì lão này cũng thuộc dạng tay to ☺). Về cơ bản kĩ thuật để cập trong bài này cũng không có gì mới, lý thuyết về nó nằm ở phần *Conditional Log BreakPoints* trong bài 11 và phần *Message BreakPoints* trong bài 12, chúng ta luyện lại để cho thuần thục hơn mà thôi. Ngoài ra có kèm thêm một target nữa để các bạn tự làm và tự đúc rút kinh nghiệm cho mình, đồng thời nó cũng là chìa khóa để unlock phần 19 kế tiếp lolz …. N0w let's g0……

II. Phân tích và xử lý target

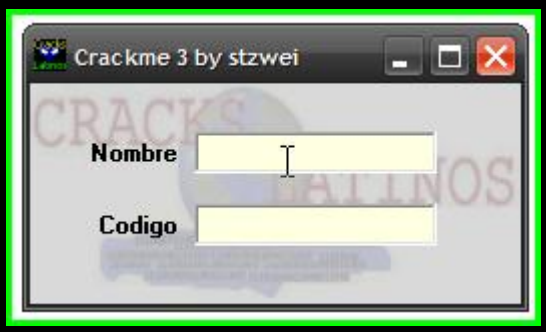
1. Xử lý Crackme 3 by StzWei

Trước tiên, cứ check info cái đã :



Lại là một target được code bằng Borland Delphi. Lão Ricardo có vẻ nghiện các target được code bằng ngôn ngữ này thì phải.

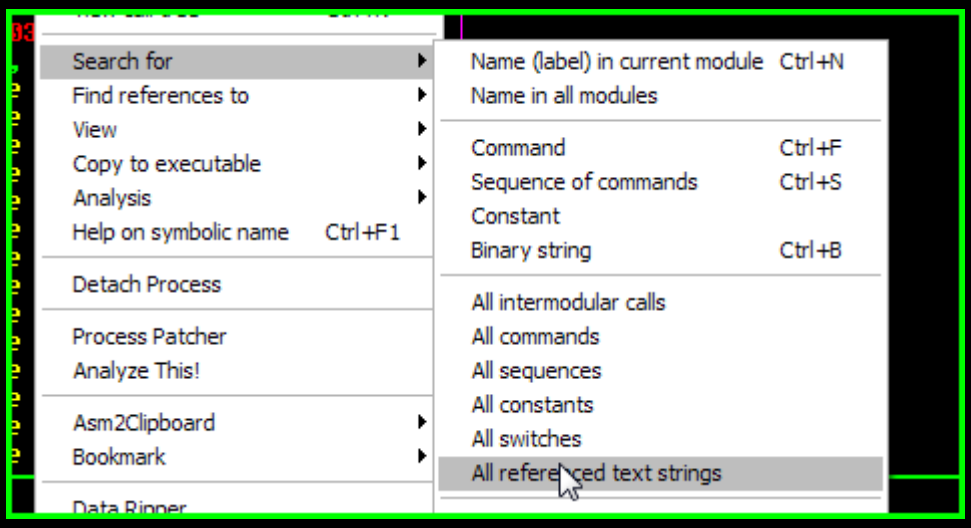
Chạy thử crackme này xem mặt mũi nó thế nào :



Chà chỉ thấy có hai text box cho phép nhập Name và Code, không thấy có nút Check gì cả ☺. Kiểu này thì biết đường nào mà mò đây!!. Ta cứ load vào Olly cái đã, sau khi load xong Olly dừng lại tại EP :

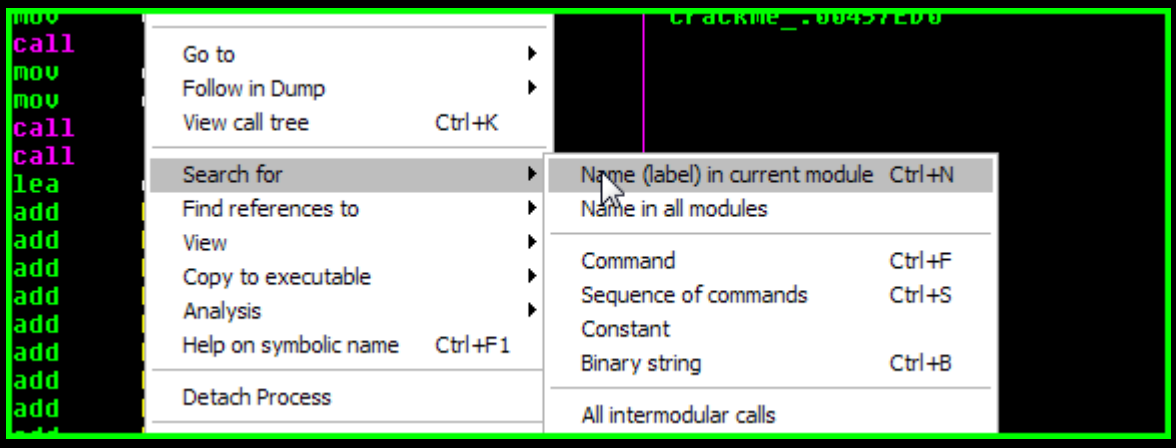
```
00458354  $  55            push    ebp
00458355  .  8BEC          mov     ebp, esp
00458357  .  83C4 F4       add     esp, -0C
0045835A  .  B8 04824500   mov     eax, 00458204
0045835F  .  E8 CCDBFAFF   call    00405F30
00458364  .  A1 CCA54500   mov     eax, dword ptr [45A5CC]
00458369  .  8B00          mov     eax, dword ptr [eax]
0045836B  .  E8 EC97FEFF   call    00441B5C
00458370  .  8B0D 9CA6450  mov     ecx, dword ptr [45A69C]      crackme_.0045B87C
00458376  .  A1 CCA54500   mov     eax, dword ptr [45A5CC]
0045837B  .  8B00          mov     eax, dword ptr [eax]
0045837D  .  8B15 847E450  mov     edx, dword ptr [457E84]      crackme_.00457ED0
00458383  .  E8 EC97FEFF   call    00441B74
00458388  .  A1 CCA54500   mov     eax, dword ptr [45A5CC]
0045838D  .  8B00          mov     eax, dword ptr [eax]
```

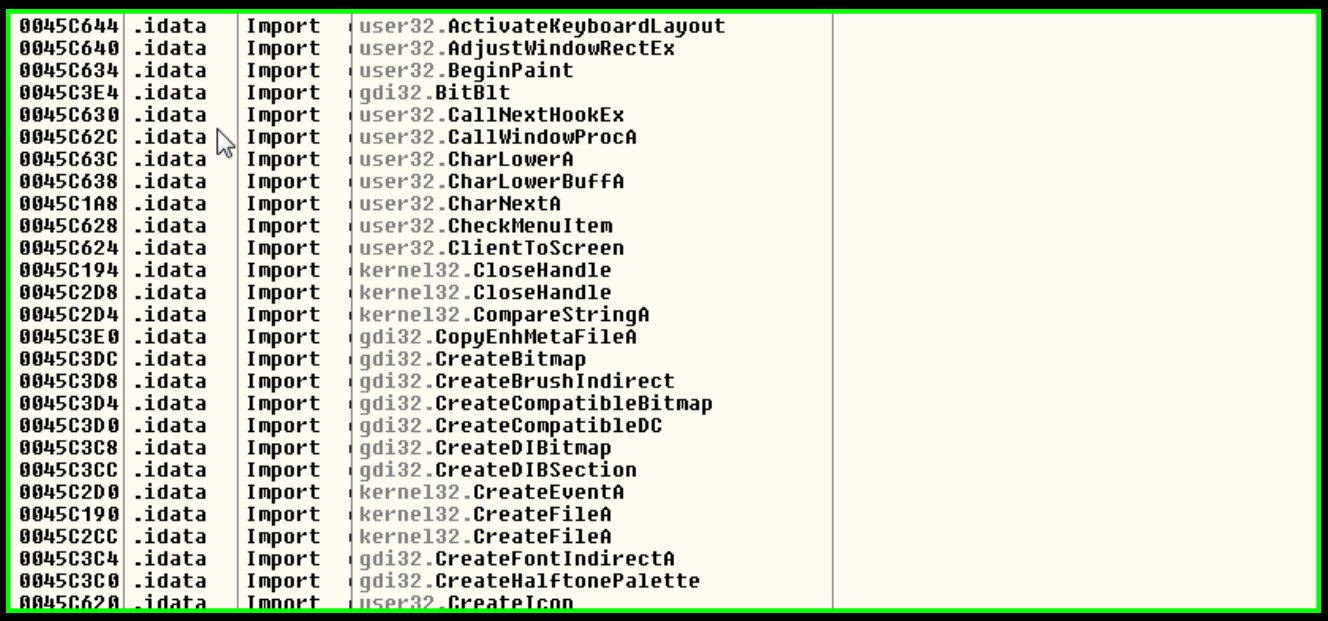
Thử tìm kiếm các Strings quan trọng xem có manh mối gì không :



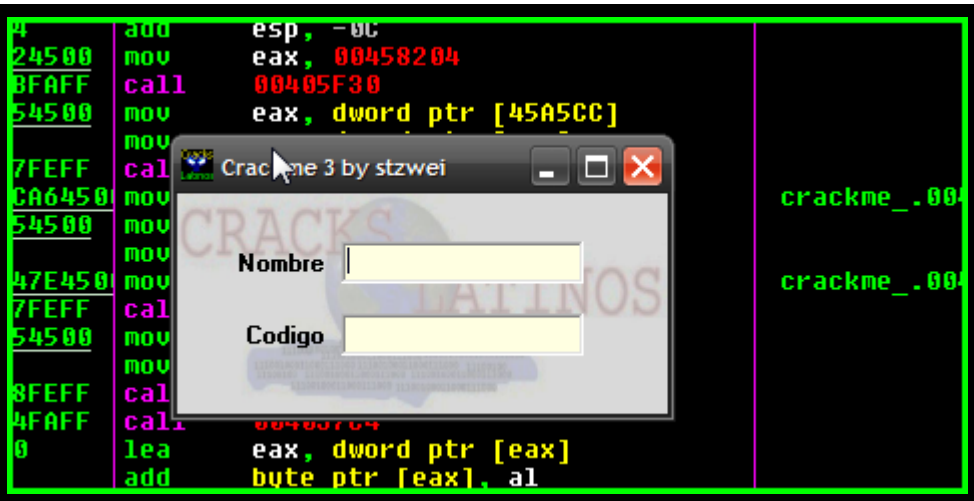
Address	Disassembly	Text string
00401006	ascii "Boolean"	
0040101B	ascii "False"	
00401021	ascii "True"	
0040102E	ascii "Char"	
00401042	ascii "Integer"	
0040105A	ascii "Byte"	
0040106E	ascii "Word"	
00401082	ascii "String"	
00401088	dd crackme_.004010D4	ASCII 07,"TObject"
004010A8	dd crackme_.004010D4	ASCII 07,"TObject"
004010D5	ascii "TObject"	
004010E2	ascii "TObject"	
004010E9	dd crackme_.004010D4	ASCII 07,"TObject"
004010F4	ascii "System"	
00401102	ascii "IUnknown"	
00401120	ascii "System"	
00401178	dd crackme_.004011C4	ASCII 11,"TInterfacedObject"
00401198	dd crackme_.004011C4	ASCII 11,"TInterfacedObject"
004011C5	ascii "TInterfacedObjec"	
004011D5	ascii "t"	
00402786	ascii "ÏÈÌÏÜØËÚÚÜÝÞßàáã"	
00402796	ascii 0	
00402B04	push 00402B84	ASCII "SOFTWARE\Borland\Delphi\RTL"
00402B38	push 00402BA0	ASCII "FPUMaskValue"
00402B84	ascii "SOFTWARE\Borland"	

Kéo lên, kéo xuống, nhìn ngược và nhìn xuôi, chẳng thấy có thông tin nào mang lại cho ta một chút gí đó gọi là manh mối. Tìm thêm thông tin về các hàm APIs xem thế nào :

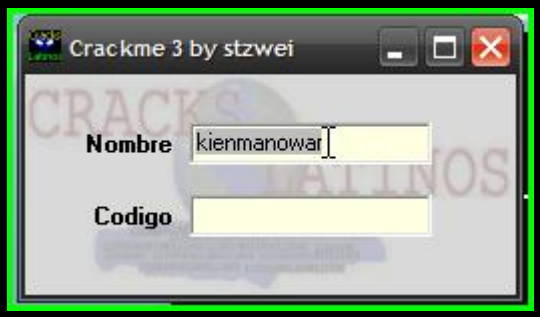




Oạch quá nhiều luôn, mắt mũi tôi tèm nhèm nhìn một lúc là thấy hoa lá cành rồi. Kiểu này tìm kiếm không ăn thua gì rồi, ta áp dụng luôn kĩ thuật mới vậy. Nhấn **F9** để thực thi chương trình :

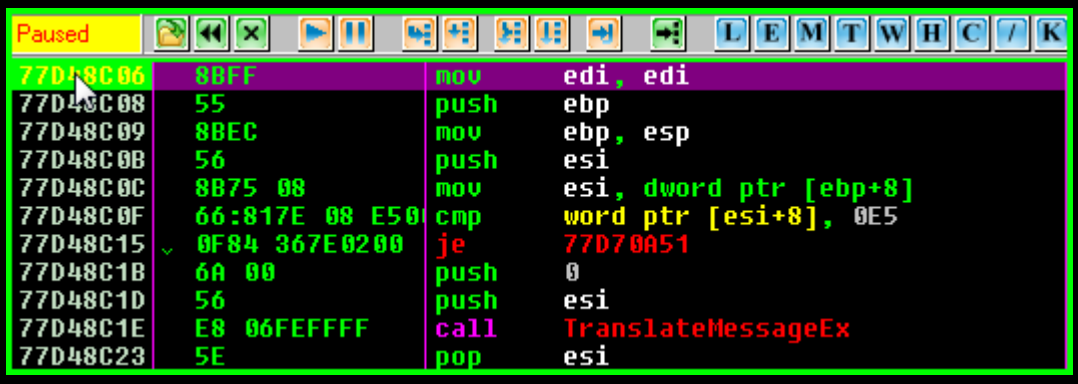


Nhập tên vào text box Name :

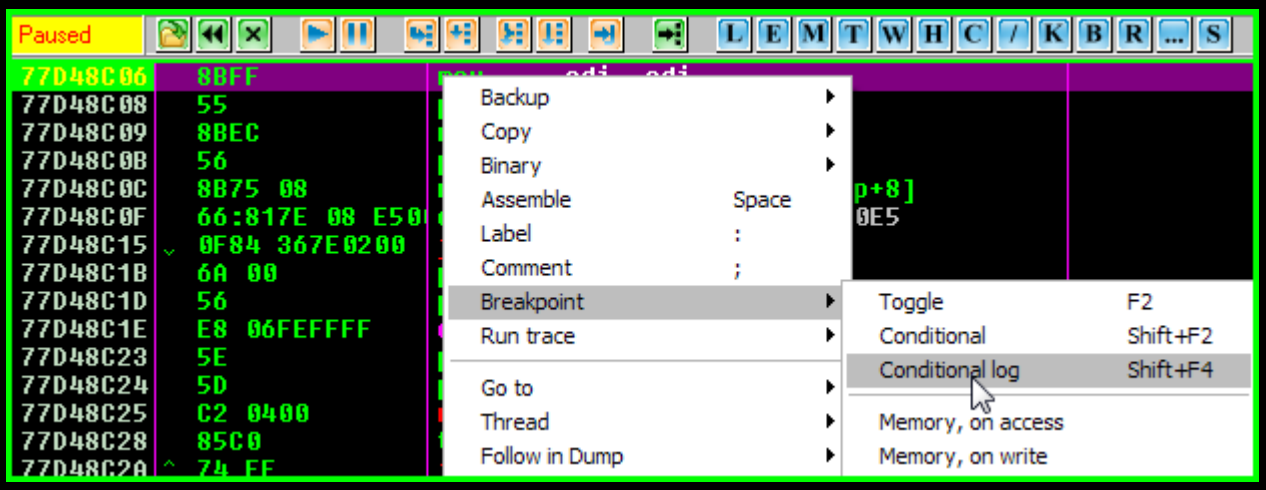


Tiếp theo, tôi sẽ đặt một Conditional Log BP tại hàm API là `TranslateMessage`, nhưng trước tiên tôi đặt BP tại `TranslateMessage` đã :

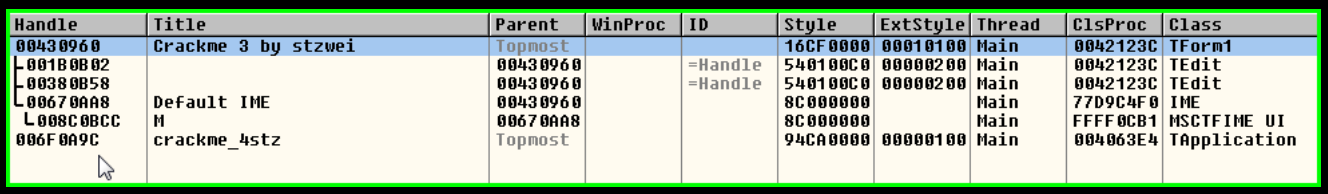
Sau khi đặt BP trên xong, ta chuyển qua Crackme đang thực thi thì ngay lập tức Olly sẽ dừng lại tại BP ta vừa đặt :



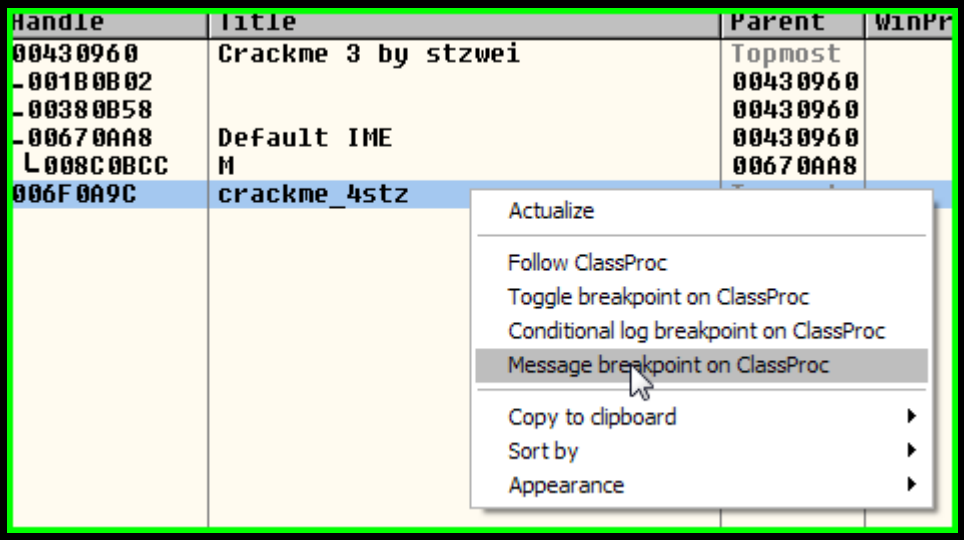
Nhấn chuột phải và chọn như sau :



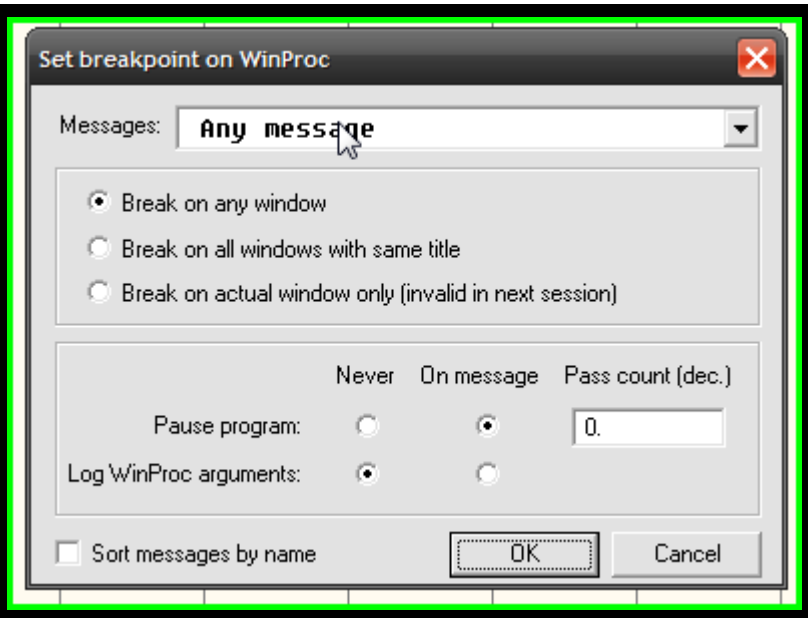
Để đặt được Conditional Log BP với trường hợp cụ thể ở đây là *WM_KEYUP* thì ta cần biết con số gắn liền với Message ta cần log lại là gì. Nếu bạn không nhớ được giá trị đó thì có thể làm theo cách sau để tìm lại, đầu tiên nhấn vào nút **W** để tới cửa sổ Windows:



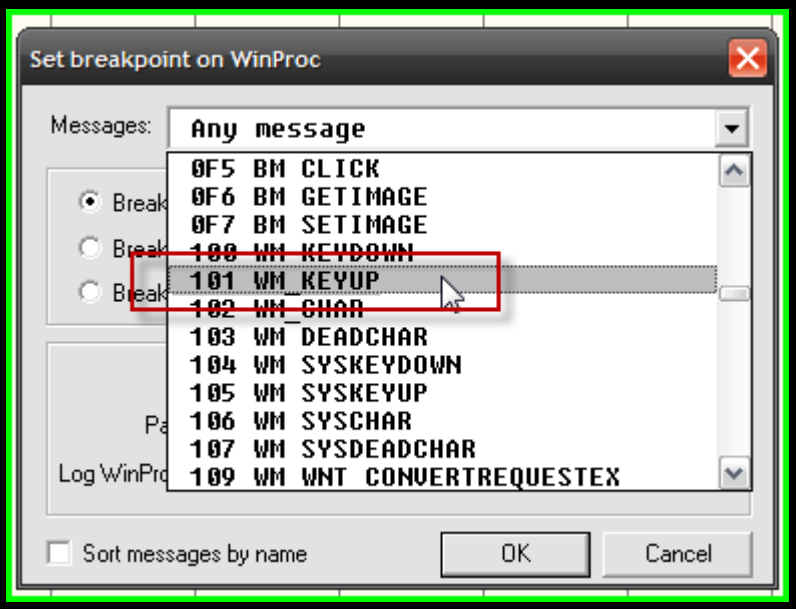
Ta thấy có một danh sách các Handle, nhấn chuột phải và chọn như sau :



Cửa sổ Message breakpoint hiện ra, kèm theo đó là một danh sách sổ xuống chứa thông tin liên quan đến Window Messages mà ta muốn tìm kiếm :



Ta tìm con số liên quan tới *WM_KEYUP* :

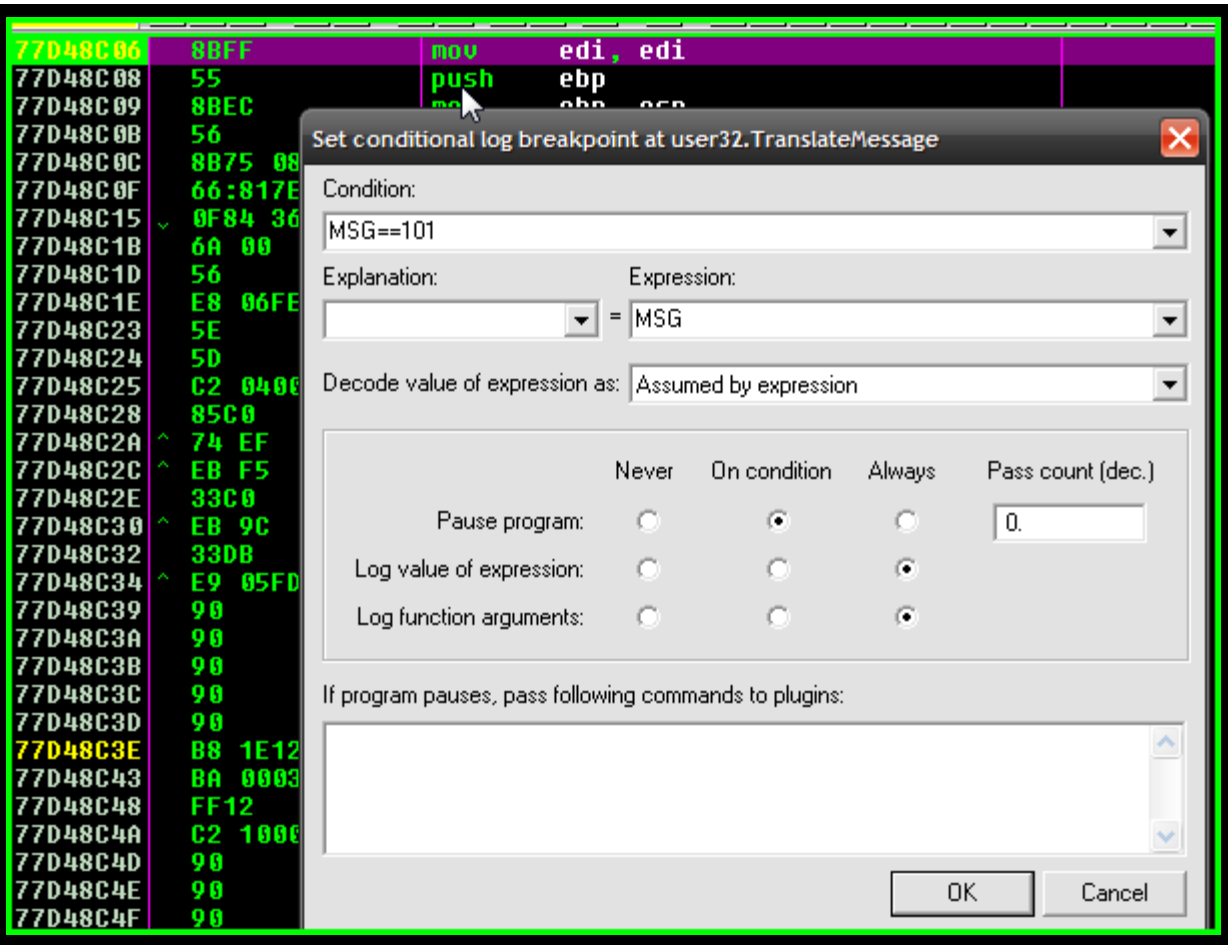


Thông tin thêm về WM_KEYUP :

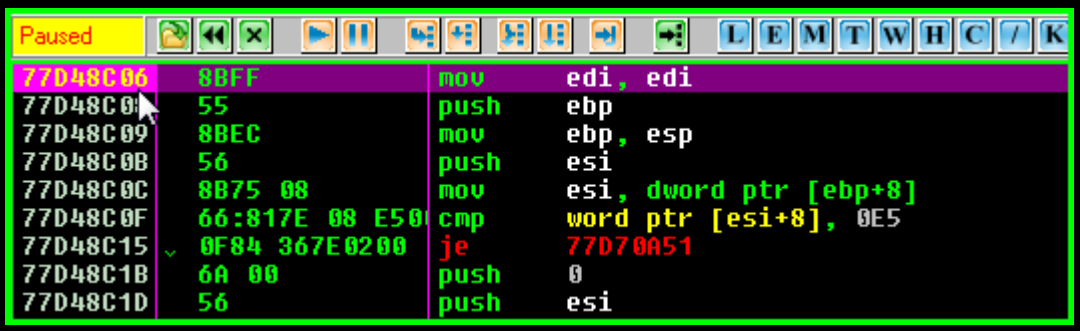
```
The WM_KEYUP message is posted to the window with the keyboard focus when
a nonsystem key is released. A nonsystem key is a key that is pressed when
the ALT key is not pressed, or a keyboard key that is pressed when a
window has the keyboard focus.

WM_KEYUP
nVirtKey = (int) wParam;     // virtual-key code
lKeyData = lParam;           // key data
```

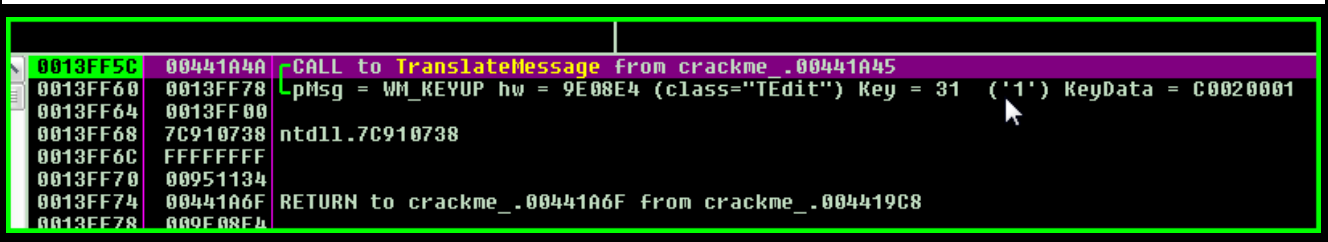
OK như vậy ta có được *WM_KEYUP* tương ứng với mã là 101. Giờ ta đóng màn hình trên lại và quay trở lại chỗ thiết lập Conditional Log BP. Tiến hành thiết lập BP như sau :



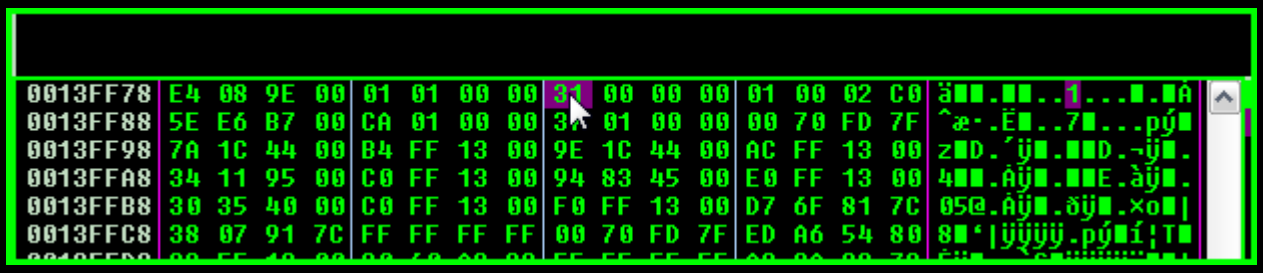
Ở đây, các bạn có thể đặt điều kiện là `MSG==WM_KEYUP` nhưng tôi thích sử dụng số hơn. Ngoài ra thông tin về cách thức đặt một Conditional Log BP như thế nào đã nói trong bài 11 và 12, các bạn tìm đọc lại nhé. Sau khi đặt xong thì BP mà ta thiết lập tại hàm `TranslateMessage` giờ đã được chuyển sang màu hồng ☺ :



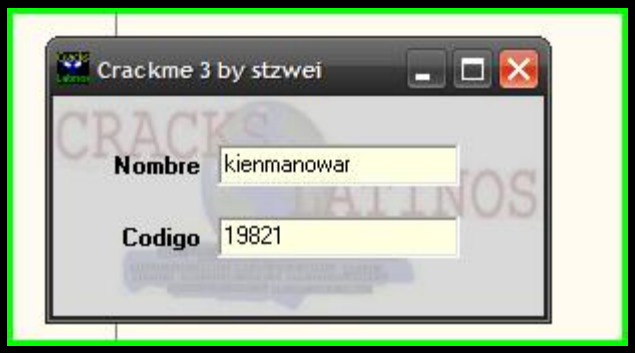
Công việc đặt BP đã xong, giờ ta nhấn **F9** để thực thi chương trình. Tiến hành nhập Fake Code vào, khi ta nhập kí tự đầu tiên và nhả tay khỏi phím thì Olly ngay lập tức sẽ break:



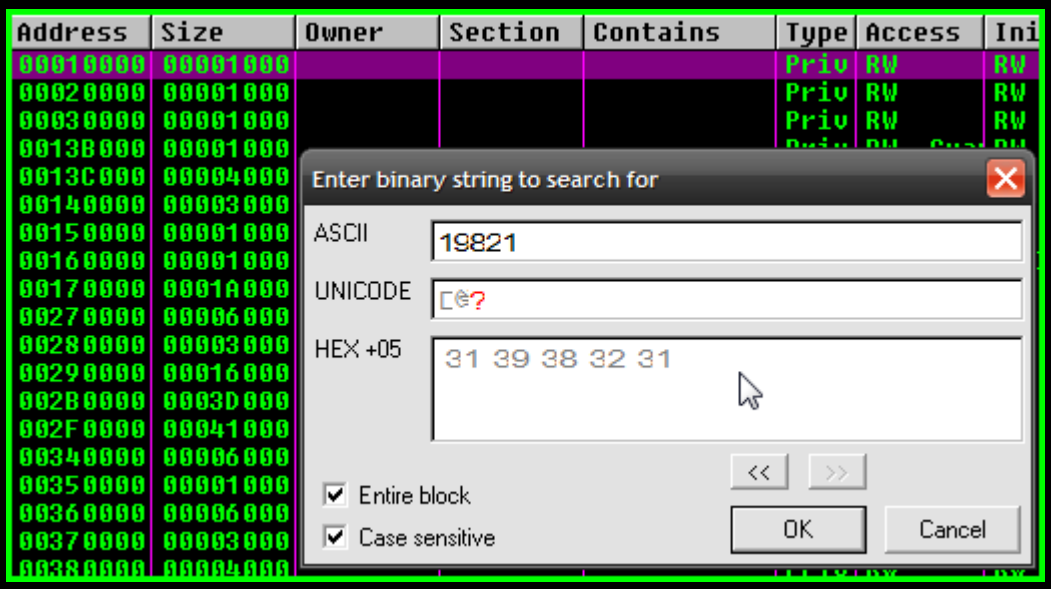
Như trên hình minh hoa, ta thấy tại địa chỉ `0x0013FF78` sẽ lưu các tham số truyền cho hàm `TranslateMessage`. Đó là một cấu trúc đã được quy định sẵn, ở đây các bạn thấy rằng giá trị tôi nhập vào là Key=31 tương ứng với số "1". Chuột phải và chọn **Follow in Dump**:



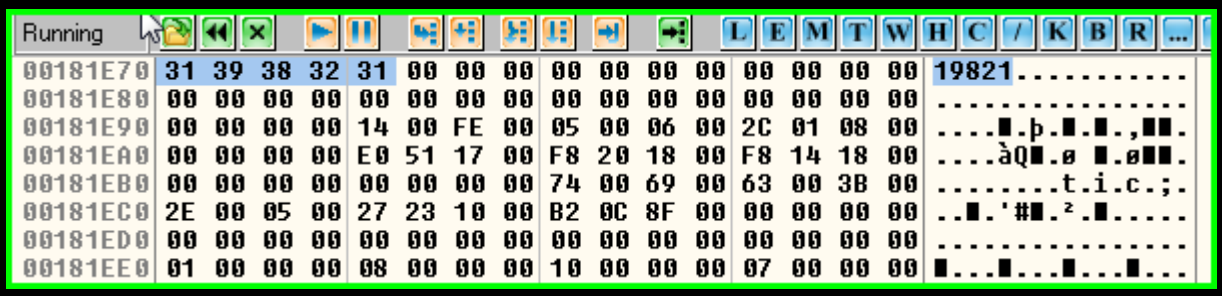
Như vậy, ta đã biết được chỗ lưu Fake Code nhập vào từ bàn phím (thực chất là Message Structure) , để làm tham số truyền vào cho hàm `TranslateMessage`. Tuy nhiên nếu ta máy móc áp dụng cách đặt BP *Memory on Access* tại Byte này để đi tìm đoạn code truy cập vào byte đó, thì ta sẽ vấp phải vô số đoạn code có truy cập tới byte trên. Đọc tài liệu của lão Ricardo viết thì lão bảo đó là một Problem của các chương trình code bằng Delphi (tôi cũng không rõ lắm, vì chuyển từ TBN – Eng nó không được chuẩn, nên đọc hơi khó hiểu). Do đó để tiếp cận mục tiêu ta lại áp dụng cách mà ở bài 17 đã thực hiện, Disable Conditional Log BP đã đặt nhé, nhấn **F9** để run chương trình. Nhập thêm 4 số nữa vào phần Fake :



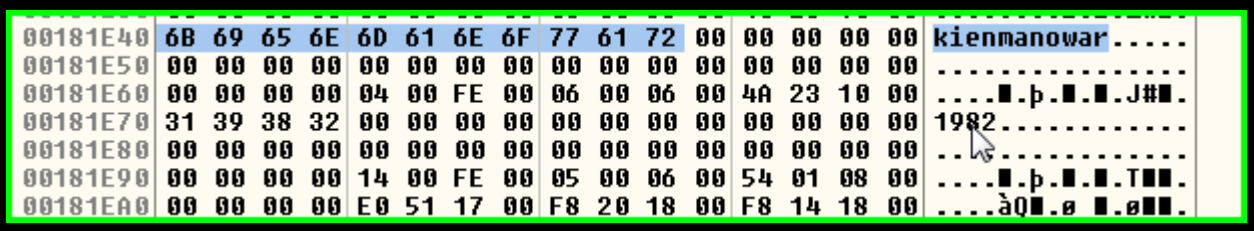
Sau khi nhập thêm xong, vào cửa sổ Memory để tìm kiếm xem toàn bộ Fake code sẽ được lưu ở đâu :



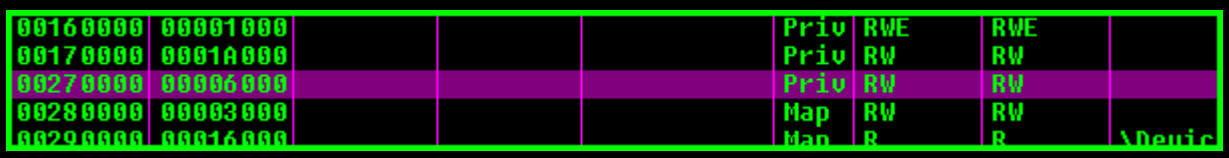
Nhấn OK để thực hiện tìm kiếm. Ta có được vị trí đầu tiên xuất hiện chuỗi Fake code :



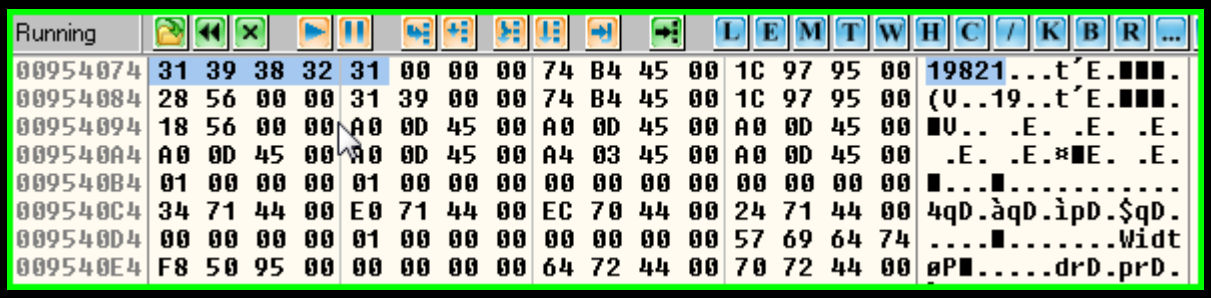
Cuộn chuột lên một chút để xem có thông tin gì khác không, ta có được như sau :



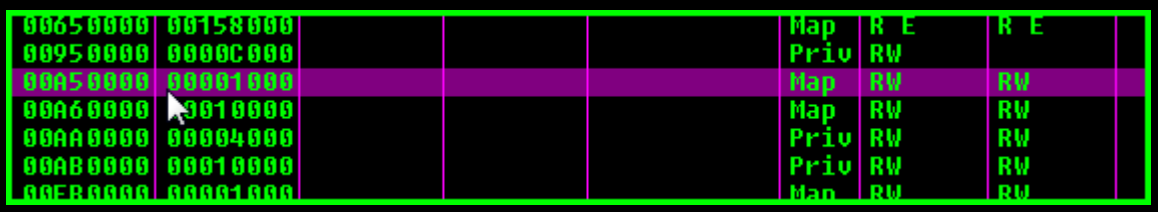
Đây có vẻ là vùng nhớ mà ta cần. Nhấn tiếp **Ctrl+L** để xem Fake Code nó có xuất hiện ở đâu khác tại vùng nhờ này không. Kết quả, chỉ có đúng một lần xuất hiện tại `0x00181E70` (có thể khác trên máy bạn). Nhấn vào **M** để quay trở lại cửa số Memory, ta đang đứng tại section sau :



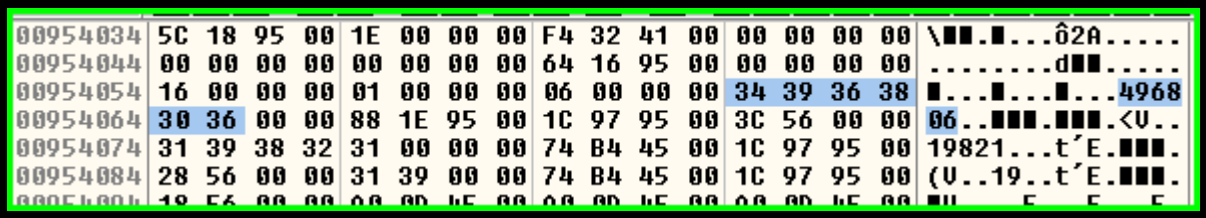
Nhấn tiếp **Ctrl+L** để tìm kiếm trong các section khác, tôi có thêm vị trí sau :



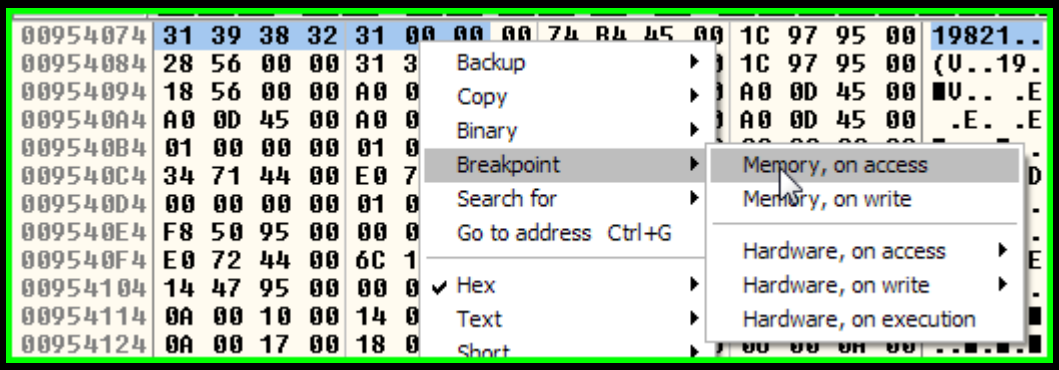
Tiếp tục nhấn **Ctrl+L** một lần nữa để xem section này còn vị trí nào lưu nữa không. Với tôi chỉ có đúng vị trí trên ☺. Tương ứng với Section này :



Tìm kiếm tiếp thì thấy không còn section nào lưu chuỗi Fake Code nữa. Vậy kết luận chúng ta có hai vị trí quan trọng đã tìm được là `0x00181E70` và `0x00954074` . Không biết vị trí nào mới được sử dụng cho quá trình so sánh với Real code đây. Các bạn có thể đặt BP Memory on Access với từng vị trí để kiểm tra, nhưng tôi đặt sự quan tâm lớn ở vùng nhớ thứ hai mà ta tìm được :



Do tôi nghi ngờ đoạn bôi đen chính là Real code, nhưng chưa dám khẳng định chắc chắn cho nên mới nói là quan tâm tới vùng nhờ này hơn. Để cụ thể xem thực hư thế nào, ta đặt BP như sau :



Đặt BP xong, quay trở lại Crackme và nhập thêm một số nữa vào Fake Code. Olly sẽ break tại đoạn code sau :

```
004039CD   .  83C0 08      add     eax, 8
004039D0   .  5A           pop     edx
004039D1   .  8950 FC      mov     dword ptr [eax-4], edx
004039D4   .  C740 F8 0100 mov     dword ptr [eax-8], 1
004039DB   .  C60410 00    mov     byte ptr [eax+edx], 0
004039DF   .  C3           retn
004039E0   >  31C0         xor     eax, eax
004039E2   L. C3           retn
004039E3      90           nop
```

Nhấn **F9**, ta sẽ dừng lại tại đây :

```
0040280E   .v 78 11        js      short 00402821
00402810   .  FD           std
00402811   .  F3:A5        rep     movs dword ptr es:[edi], dword ptr [esi]
00402813   .  89C1         mov     ecx, eax
00402815   .  83E1 03      and     ecx, 3
00402818   .  83C6 03      add     esi, 3
0040281B   .  83C7 03      add     edi, 3
0040281E   .  F3:A4        rep     movs byte ptr es:[edi], byte ptr [esi]
00402820   .  FC           cld
00402821   >  5F           pop     edi
00402822   .  5E           pop     esi
00402823   L. C3           retn
```

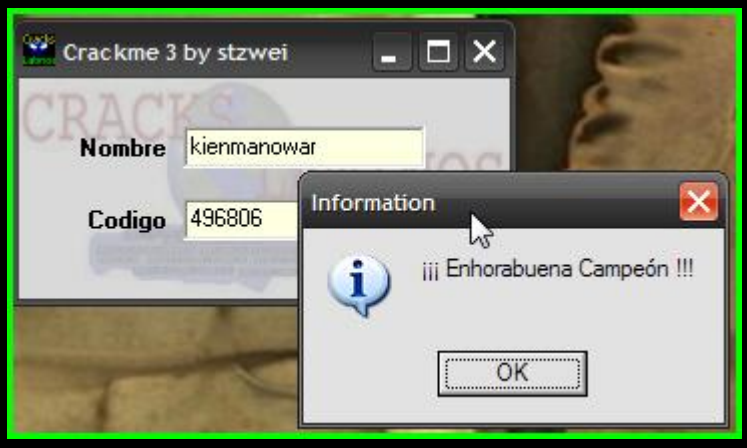
F9 thêm một lần nữa ta sẽ dừng lại tại đoạn code so sánh sau :

```
00403CB5   >  8B0E         mov     ecx, dword ptr [esi]
00403CB7   .  8B1F         mov     ebx, dword ptr [edi]
00403CB9   .  39D9         cmp     ecx, ebx
00403CBB   .v 75 58        jnz     short 00403D15
00403CBD   .  4A           dec     edx
00403CBE   .v 74 15        je      short 00403CD5
00403CC0   .  8B4E 04      mov     ecx, dword ptr [esi+4]
00403CC3   .  8B5F 04      mov     ebx, dword ptr [edi+4]
00403CC6   .  39D9         cmp     ecx, ebx
00403CC8   .v 75 4B        jnz     short 00403D15
00403CCA   .  83C6 08      add     esi, 8
00403CCD   .  83C7 08      add     edi, 8
00403CD0   .  4A           dec     edx
00403CD1   .^ 75 E2        jnz     short 00403CB5
```

Nhìn sang cửa sổ Register xem ta có gì nào ☺ :

```
Registers (FPU)                                <
EAX 00000000
ECX 32383931
EDX 00000001
EBX 0095185C
ESP 0013F3A8
EBP 0013F3E4
ESI 00954060 ASCII "198213"
EDI 00954074 ASCII "496806"
EIP 00403CB7 crackme_.00403CB7
```

Thanh ghi ESI lưu địa chỉ chứa chuỗi Fake Code, còn EDI lưu địa chỉ chứa Real Code mà ta nghi ngờ ở trên. Vậy là xong, chúng ta đã tìm được Real code tương ứng với Name nhập vào. Test thử cái xem sao nhé :



Quá chuẩn lolz….dành cho những bạn nào quan tâm tới việc tại sao ta lại có được con số của Real code và muốn code keygen :

```
00458100  .  8B75 FC      mov     esi, dword ptr [ebp-4]
00458103  .  8B7D F8      mov     edi, dword ptr [ebp-8]
00458106  .  31C9         xor     ecx, ecx
00458108  .  31DB         xor     ebx, ebx
0045810A  >  0FBE040F     movsx   eax, byte ptr [edi+ecx]
0045810E  .  41           inc     ecx
0045810F  .  31C8         xor     eax, ecx
00458111  .  01C3         add     ebx, eax
00458113  .  3B4D FC      cmp     ecx, dword ptr [ebp-4]
00458116  .^ 75 F2        jnz     short 0045810A
00458118  .  C1C0 0C      rol     eax, 0C
0045811B  .  01D8         add     eax, ebx
0045811D  .  8945 F4      mov     dword ptr [ebp-C], eax
00458120  .  61           popad
```

Bài 18 xin phép được kết thúc tại đây!

III. Kết luận

OK, toàn bộ bài 18 đến đây là kết thúc. Có thể các bạn sẽ cảm thấy hơi thất vọng, cá nhân tôi cũng thế, vì cách thức lần theo WM_KEYUP mà lão Ricardo giới thiệu dường như không mang lại nhiều thông tin mà ta mong muốn. Thậm chí có thể làm ta thấy khó khăn và rối hơn ☺ khi muốn mò người trở lại các đoạn code cần quan tâm. Tuy nhiên, cốt yếu của vấn đề ở đây là ta vận dụng linh hoạt các kiểu đặt BP, tìm kiếm thông tin trong memory v..v.. để giải quyết bài toán, sao cho bằng cách nhanh nhất ta tìm ra được lời giải.

Trong bài 18 này có thêm một target để các bạn luyện tập tay nghề, đó là : CrackMe v2.0 by CrueHead. Chuỗi Serial bạn tìm ra sẽ là password để mở khóa bài viết thứ 19 nhé. Hẹn gặp lại các bạn ở bài 19!

PS: Tài liệu này chỉ mang tính tham khảo, tác giả không chịu trách nhiệm nếu người đọc sử dụng nó vào bất kì mục đích nào.

Best Regards
[Kienmanowar]

--++--==[Greatz Thanks To]==--++--
My family, Computer_Angel, Moonbaby , Zombie_Deathman, Littleboy, Benina, QHQCrker, the_Lighthouse, Merc, Hoadongnoi, Nini ... all REA's members, TQN, HacNho, RongChauA, Deux, tlandn, light.phoenix, dqtln, ARTEAM .... all my friend, and YOU.

--++--==[Thanks To]==--++--
iamidiot, WhyNotBar, trickyboy, dzungltvn, takada, hurt_heart, haule_nth, hytkl, moth, XIANUA, nhc1987, 0xdie, Unregistered!, akira, mranglex v..v.. các bạn đã đóng góp rất nhiều cho REA. Hi vọng các bạn sẽ tiếp tục phát huy ☺

I want to thank **Teddy Roggers** for his great site, Reversing.be folks(especially **haggar**), Arteam folks(**Shub-Nigurrath, MaDMAn_H3rCuL3s**) and all folks on crackmes.de, thank to all members of **unpack.cn** (especially **fly** and **linhanshi**). Great thanks to **lena151**(I like your tutorials). And finally, thanks to **RICARDO NARVAJA** and all members on **CRACKSLATINOS.**

>>>> If you have any suggestions, comments or corrections email me:
kienmanowar[at]reaonline.net