2009

[Cracking with OllyDbg]

Based on OllyDbg tuts of Ricardo Narvaja (CrackLatinos Team)

www.reaonline.net

kienmanowar

12/28/2009

Mục Lục

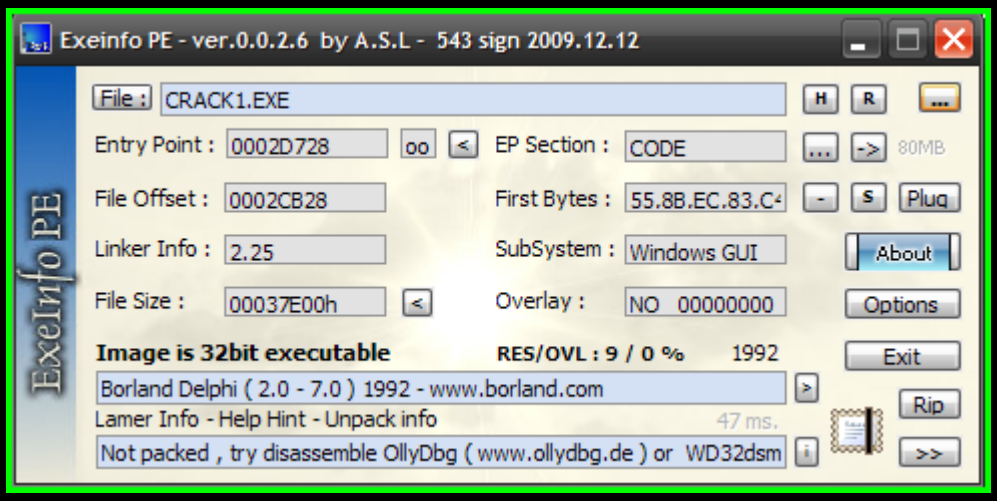
I. Giới thiệu chung

Không có gì để giới thiệu nhiều, ở phần 17 này chúng ta sẽ tập trung vào xử lý hai target: một crackme và một Share warez. Có thể nhiều bạn sẽ chê crackme, nhưng thú thực là ngày xưa khi bập bẹ Cracking tôi đâu có dám chơi Soft thật đâu. Tôi đi theo trường phái của đại ca Moon, ban đầu thì cứ crackme mà luyện dần dần mới thử sức trên các phần mềm thật. Đơn giản là vì crackme nhỏ nhẹ, có các level khác nhau, thường là không pack và thuật toán có đủ các thể loại trên trời dưới đất 😑. Còn Soft thật thì nhiều khi xử lý xong cũng có dùng mấy đâu ☺. Vài lời tâm sự thế là đủ rồi …. N0w let's g0……

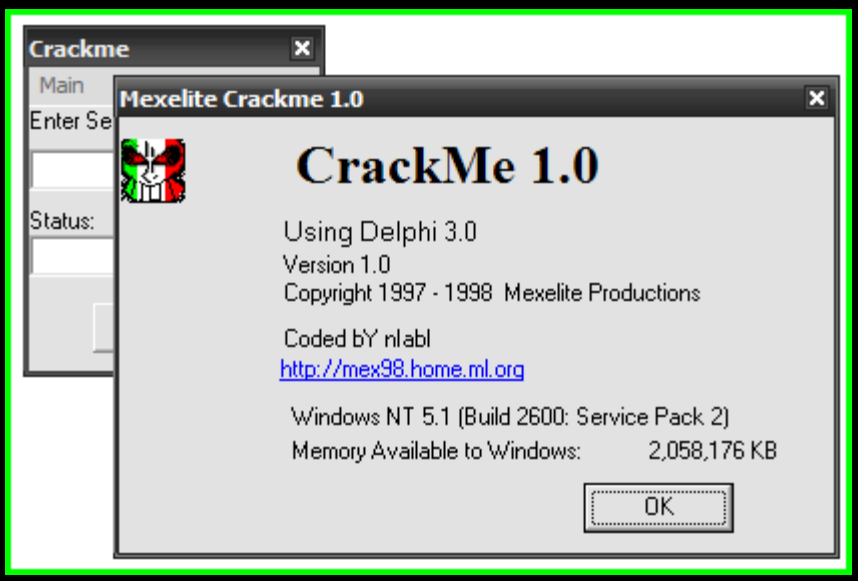
II. Phân tích và xử lý target

1. Xử lý Mexelite Crackme

Đầu tiên, ta kiểm tra thông tin sơ bộ về Crackme này :



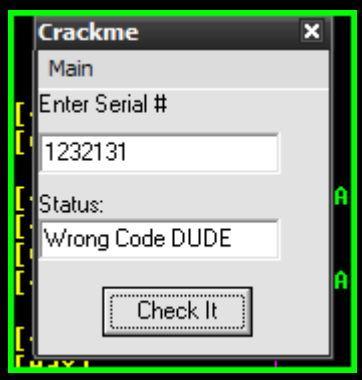
Chạy thử crackme và nhấn vào *Main* > *About* :



Kết luận : Crackme không bị pack và được code bằng Borland Delphi 3.0. Tiếp theo load crackme vào Olly, ta dừng lại tại EP của crackme :

```
0042D728  $  55            push    ebp
0042D729  .  8BEC          mov     ebp, esp
0042D72B  .  83C4 F4       add     esp, -0C
0042D72E  .  B8 40D64200   mov     eax, 0042D640
0042D733  .  E8 2078FDFF   call    00404F58
0042D738  .  A1 10EA4200   mov     eax, dword ptr [42EA10]
0042D73D  .  8B00          mov     eax, dword ptr [eax]
0042D73F  .  E8 BCB0FFFF   call    00428800
0042D744  .  8B0D 88EA420( mov     ecx, dword ptr [42EA88]      CRACK1.0042F748
0042D74A  .  A1 10EA4200   mov     eax, dword ptr [42EA10]
0042D74F  .  8B00          mov     eax, dword ptr [eax]
0042D751  .  8B15 5CD3420( mov     edx, dword ptr [42D35C]      CRACK1.0042D39C
0042D757  .  E8 BCB0FFFF   call    00428818
```

Nhấn **F9** để thực thi chương trình, nhập đại một giá trị Serial nào đó và nhấn nút ***Check It***. Ta nhận được thông tin sau :



Quay trở lại Olly, chuột phải chọn **Search for > All referenced text strings**. Ta tìm được thông tin quan trọng sau :

```
0042D4F6  ascii   "TForm1"
0042D507  ascii   "crackme"
0042D537  mov     edx, 0042D590                ASCII "Benadryl"
0042D543  mov     edx, 0042D5A4                ASCII "Wrong Code DUDE"
0042D555  mov     edx, 0042D5BC                ASCII "Thanks you made it"
0042D590  ascii   "Benadryl",0
0042D5A4  ascii   "Wrong Code DUDE",0
0042D5BC  ascii   "Thanks you made "
0042D5CC  ascii   "it",0
```

Nhấn đúp chuột tại một trong hai dòng thông báo *"Wrong Code.."* hoặc *"Thanks you.."* sẽ đưa chúng ta tới đoạn code mà ta quan tâm :

```
0042D529  .  8B83 DC01000( mov     eax, dword ptr [ebx+1DC]   *TForm1.Edit1:TEdit
0042D52F  .  E8 54CCFEFF   call    00410188                   ->Controls.TControl.GetText()
0042D534  .  8B45 FC       mov     eax, dword ptr [ebp-4]
0042D537  .  BA 90D54200   mov     edx, 0042D590              ASCII "Benadryl"
0042D53C  .  E8 8F63FDFF   call    004038D0                   ->System..LStrCmp()
0042D541  .v 74 12         je      short 0042D555
0042D543  .  BA A4D54200   mov     edx, 0042D5A4              ASCII "Wrong Code DUDE"
0042D548  .  8B83 E801000( mov     eax, dword ptr [ebx+1E8]   *TForm1.Edit2:TEdit
0042D54E  .  E8 65CCFEFF   call    004101B8                   ->Controls.TControl.SetText(System.AnsiString)
0042D553  .v EB 10         jmp     short 0042D565
0042D555  >  BA BCD54200   mov     edx, 0042D5BC              ASCII "Thanks you made it"
0042D55A  .  8B83 E801000( mov     eax, dword ptr [ebx+1E8]   *TForm1.Edit2:TEdit
0042D560  .  E8 53CCFEFF   call    004101B8                   ->Controls.TControl.SetText(System.AnsiString)
0042D565  >  33C0          xor     eax, eax
```

Tại vùng code này để ý ta sẽ thấy như sau :

```
0042D53C >|.  E8 8F63FDFF   call     004038D0  ;  ->System..LStrCmp()
0042D541  |.  74 12         je       short 0042D555
```

Tại địa chỉ `0x0042D53C` là một lệnh Call, không biết bên trong lệnh Call này sẽ làm gì nhưng theo gợi ý của Olly là `System..LStrCmp()` thì ta cũng có thể đoạn được lệnh Call này sẽ thực hiện công việc so sánh hai chuỗi. Kết quả so sánh thế nào sẽ tác động lên cờ ZF, dựa vào cờ ZF thì lệnh nhảy sẽ thực hiện hoặc không thực hiện. Vậy điểm mấu chốt chính là lệnh Call tại `0x0042D53C`, ta đặt BP tại lệnh call này. Nhấn **Ctrl+F2**, để restart lại Olly, **F9** để run chương trình. Sau đó, nhập Fake Serial vào và nhấn nút ***Check It***. Olly sẽ break tại địa chỉ mà ta thiết lập BP :

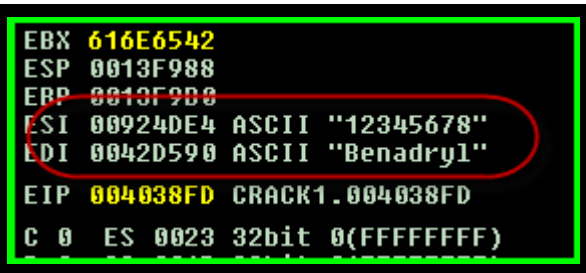

```
0042D534  .  8B45 FC       mov     eax, dword ptr [ebp-4]
0042D537  .  BA 90D54200   mov     edx, 0042D590              ASCII "Benadryl"
0042D53C  .  E8 8F63FDFF   call    004038D0                   ->System..LStrCmp()
0042D541  .v 74 12         je      short 0042D555
0042D543  .  BA A4D54200   mov     edx, 0042D5A4              ASCII "Wrong Code DUDE"
0042D548  .  8B83 E801000  mov     eax, dword ptr [ebx+1E8]   *TForm1.Edit2:TEdit
0042D54E  .  E8 65CCFEFF   call    0041A1B8                   ->Controls.TControl.SetText(System.AnsiString)
```

Để ý ở trước lệnh call là hai lệnh mov, trong đó ta thấy thanh ghi edx sẽ giữ địa chỉ chứa chuỗi mặc định (`0042D537  |.  BA 90D54200  mov  edx, 0042D590  ;  ASCII "Benadryl"`). Còn lệnh mov kia khỏi cần đoán ta cũng biết chắc đó là chuỗi Fake Serial. Để chắc chắn hơi ta nhấn **F7** để trace vào trong lệnh Call, ta dừng lại tại đây :

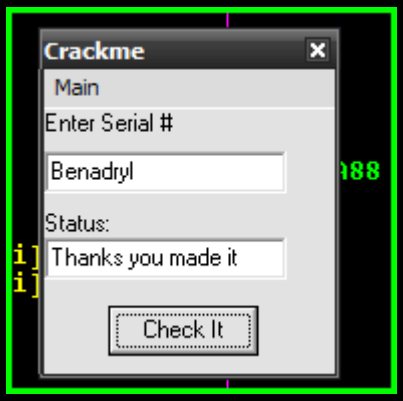
```
004038D0  $  53          push    ebx
004038D1  .  56          push    esi
004038D2  .  57          push    edi
004038D3  .  89C6        mov     esi, eax
004038D5  .  89D7        mov     edi, edx                   CRACK1.0042D590
004038D7  .  39D0        cmp     eax, edx                   CRACK1.0042D590
```

Nhấn **F8** để trace, ta sẽ bắt gặp đoạn code sau :

```
004038F9  >  8B0E        mov     ecx, dword ptr [esi]
004038FB  .  8B1F        mov     ebx, dword ptr [edi]
004038FD  .  39D9        cmp     ecx, ebx
004038FF  .v 75 58       jnz     short 00403959
```



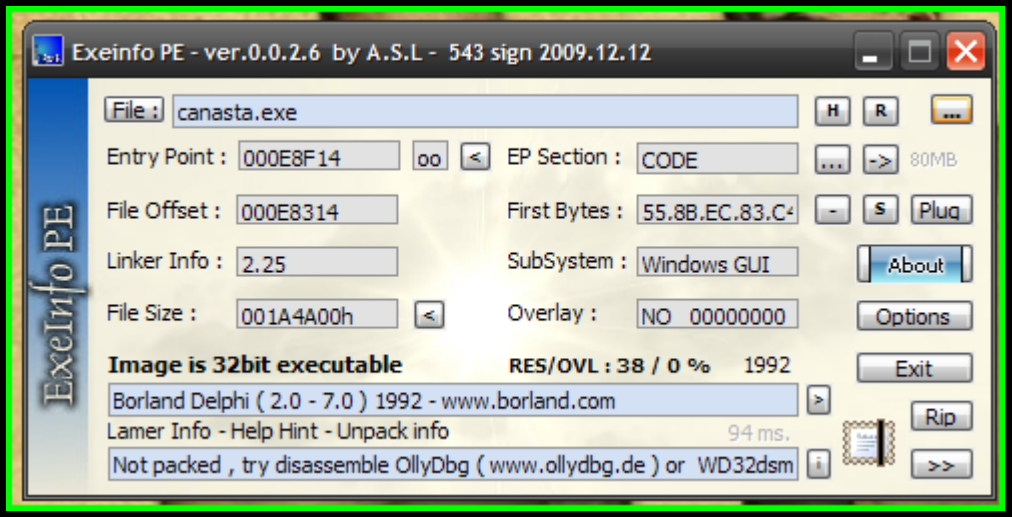
Chà đơn giản quá nhỉ, mọi thông tin hiển thị quá rõ ràng. Chuỗi Serial mà chúng ta cần tìm là : "Benadryl". Xóa BP mà chúng ta đã đặt, sau đó nhấn **F9** để run chương trình. Nhập Serial mà chúng ta vừa tìm được và nhấn nút **Check It** :



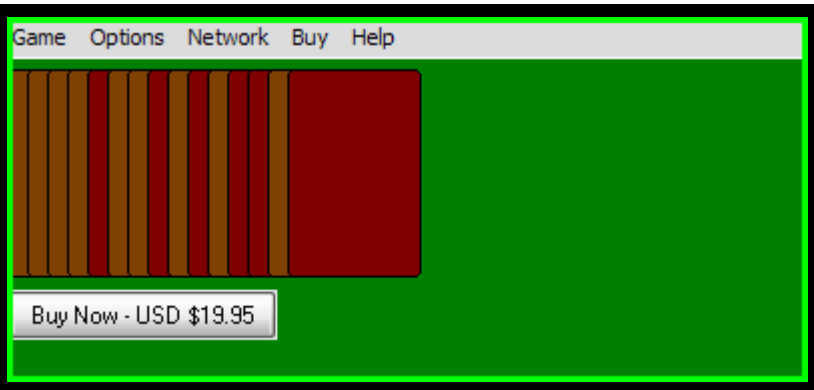
Như vậy là phần xử lý crackme đã xong ☺. Ta chuyển sang giải quyết target thứ hai.

2. *Xử lý chương trình Canasta 5.0*

Cài đặt chương trình. Sau đó kiểm tra thông tin về nó xem thế nào :



Cũng tương tự như Crackme trên, chương trình này không bị pack và được code bằng *Borland Delphi*. Chạy thử chương trình xem thế nào :



Oh chưa gì đã đập ngay vào mắt ta **Buy Now - USD $19.95** ☺. Chọn menu *Help* > *About* , sau đó nhấn vào nút **Enter License** :

Chà có vẻ target này khó nhằn đây, như các bạn thấy thì chỉ khi nào thông tin về License ta nhập vào là đúng thì nút OK mới được Enable. Vậy có nghĩa là trong quá trình

ta nhập Name và Key chương trình sẽ tự động gọi tới một hàm/hay một đoạn code (tạm gọi là hàm Validate) để xác nhận thông tin mà ta nhập vào là đúng hay sai. Sai thì giữ nguyên trang thái nút OK, còn đúng thì sẽ Enable nút OK. Tạm thời ta phán đoán thế đã, bước tiếp theo là load target vào Olly :

```
004E8F14  $  55           push    ebp
004E8F15  .  8BEC         mov     ebp, esp
004E8F17  .  83C4 F0      add     esp, -10
004E8F1A  .  B8 248B4E00  mov     eax, 004E8B24
004E8F1F  .  E8 78DEF1FF  call    00406D9C
004E8F24  .  BA F08F4E00  mov     edx, 004E8FF0                ASCII "Canasta for Windows"
004E8F29  .  A1 74884E00  mov     eax, dword ptr [4E8874]
004E8F2E  .  E8 A1F9FFFF  call    004E88D4
004E8F33  .  84C0         test    al, al
004E8F35  .v 75 14        jnz     short 004E8F4B
004E8F37  .  BA 0C904E00  mov     edx, 004E900C                ASCII "Canasta"
004E8F3C  .  A1 74884E00  mov     eax, dword ptr [4E8874]
```

Ta đang ở EP của chương trình. Tìm kiếm thông tin về các hàm APIs được sử dụng trong target :

Address	Section	Type	Name	Comment
004389D4	CODE	User	<-TShape@StyleChanged	<-TShape@StyleChanged
004F7850	.idata	Import	user32.ActivateKeyboardLayout	
004F784C	.idata	Import	user32.AdjustWindowRectEx	
004F7834	.idata	Import	user32.BeginDeferWindowPos	
004F7830	.idata	Import	user32.BeginPaint	
004F7598	.idata	Import	gdi32.BitBlt	
004F782C	.idata	Import	user32.CallNextHookEx	
004F7828	.idata	Import	user32.CallWindowProcA	
004F7840	.idata	Import	user32.CharLowerA	
004F783C	.idata	Import	user32.CharLowerBuffA	
004F7298	.idata	Import	user32.CharNextA	
004F7838	.idata	Import	user32.CharNextA	
004F7848	.idata	Import	user32.CharToOemA	
004F7844	.idata	Import	user32.CharUpperBuffA	
004F7824	.idata	Import	user32.CheckMenuItem	
004F7820	.idata	Import	user32.ChildWindowFromPoint	
004F7944	.idata	Import	comdlg32.ChooseColorA	
004F7940	.idata	Import	comdlg32.ChooseFontA	
004F781C	.idata	Import	user32.ClientToScreen	
004F7818	.idata	Import	user32.CloseClipboard	
004F7284	.idata	Import	kernel32.CloseHandle	
004F744C	.idata	Import	kernel32.CloseHandle	
004F7918	.idata	Import	winspool.ClosePrinter	
004F788C	.idata	Import	ole32.CoCreateInstance	
004F7894	.idata	Import	ole32.CoInitialize	
004F7448	.idata	Import	kernel32.CompareStringA	
004F7594	.idata	Import	gdi32.CopyEnhMetaFileA	
004F7888	.idata	Import	ole32.CoTaskMemAlloc	
004F7884	.idata	Import	ole32.CoTaskMemFree	
004F7890	.idata	Import	ole32.CoUninitialize	

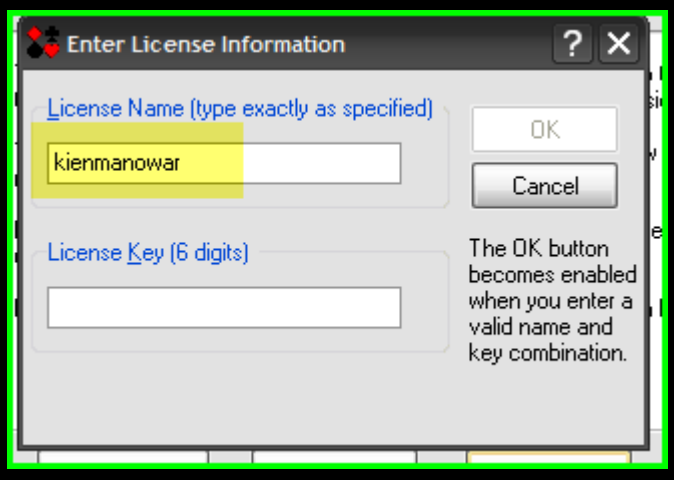
Chà nhiều quá 😐, thử tìm kiếm thông tin về các strings quan trọng :

```
004B47EB mov     edx, 004B4880                  ASCII "Settings\DateFormat"
004B4804 push    004B489C                       ASCII "Registration Key"
004B4813 mov     ecx, 004B48B8                  ASCII "User Name"
004B4823 push    004B48CC                       ASCII "CodePageEx"
004B4832 mov     ecx, 004B48E0                  ASCII "CodePage"
004B4880 ascii   "Settings\DateFor"
004B4890 ascii   "mat",0
004B489C ascii   "Registration Key"
004B48AC ascii   0
004B48B8 ascii   "User Name",0
004B48CC ascii   "CodePageEx",0
004B48E0 ascii   "CodePage",0
004B493E mov     edx, 004B49C4                  ASCII "Settings\DateFormat"
004B495C push    004B49E0                       ASCII "Registration Key"
004B496B mov     ecx, 004B49FC                  ASCII "User Name"
004B497B push    004B4A10                       ASCII "CodePageEx"
004B498A mov     ecx, 004B4A24                  ASCII "CodePage"
004B49C4 ascii   "Settings\DateFor"
004B49D4 ascii   "mat",0
004B49E0 ascii   "Registration Key"
004B49F0 ascii   0
004B49FC ascii   "User Name",0
004B4A10 ascii   "CodePageEx",0
004B4A24 ascii   "CodePage",0
004B4AC6 mov     eax, 004B4B24                  ASCII "LICENSE INVALID"
004B4B24 ascii   "LICENSE INVALID",0
004B4BB8 mov     edx, 004B4D70                  ASCII "http://www.canasta.net/keychecker/"
004B4BE0 mov     eax, 004B4D9C                  ASCII "000"
004B4C0A mov     eax, 004B4DA8                  ASCII "yyyy/mm/dd"
004B4C2D mov     eax, 004B4DA8                  ASCII "yyyy/mm/dd"
004B4C49 mov     eax, 004B4DBC                  ASCII "app=%s&name=%s&postfix=%s&key=%s&installdate=%s&currentdate=%s&submit=SUBMIT"
```

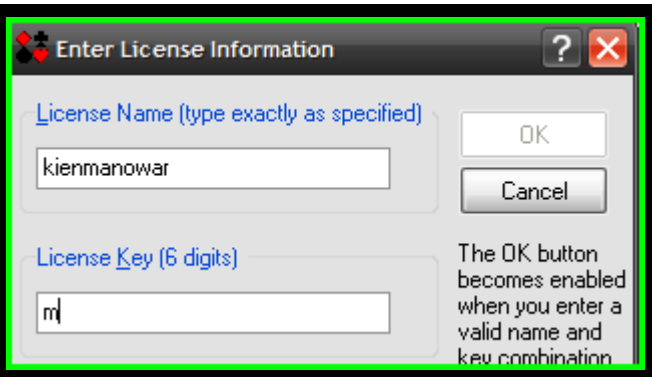
Theo trên, ta thấy xuất hiện những string quan trọng để ta có thể lần ngược về đoạn code thực hiện để check Lic. Tuy nhiên, việc lần ngược lại có vẻ hơi rắc rối và mất thời gian một chút, ngoài ra ta để ý rằng nút OK chỉ được enable lên khi ta nhập đúng Lic, cho nên ở đây ta sẽ đi mò lic theo một hướng khác mà không dựa vào những thông tin ta tìm kiếm được ở trên.

Như tôi đã nói ở trên, khi ta nhập lic vào ô text box : *License Key* thì chương trình sẽ tự động thực hiện gọi một hàm/hay một đoạn code nào đó (ta đã tạm gọi là Validate) để xác thực thông tin ta nhập vào. Nếu đúng sẽ Enable nút OK, vậy ta sẽ bám theo dấu vết này để xem mỗi lần ta nhập một kí tự vào ô text box thì nó sẽ được lưu ở vùng nhờ nào, sau khi nhập thì nó sẽ được so sánh với cái gì ? Có phải là so sánh với Valid License hay không ☺ ?

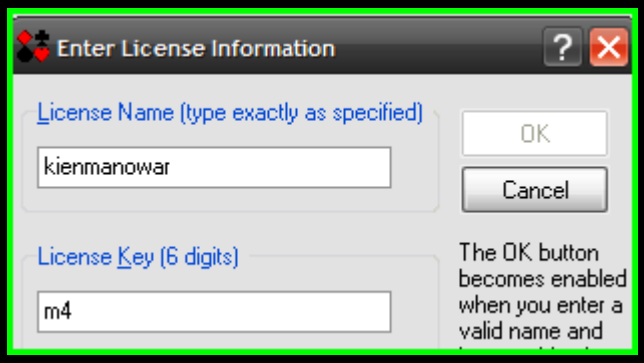
Nhấn F9 để thực thi chương trình, nhập thông tin vào phần License. Đầu tiên ta điền một Name bất kì, ví dụ ở đây của tôi là *kienmanowar* :



Tiếp theo ta nhập phần License Key, để ý ta thấy nó yêu cầu nhập 6 con số nhưng ở đây tôi nhập đại là "m4n0w4" . Trước tiên tôi nhập chữ "m" trước :

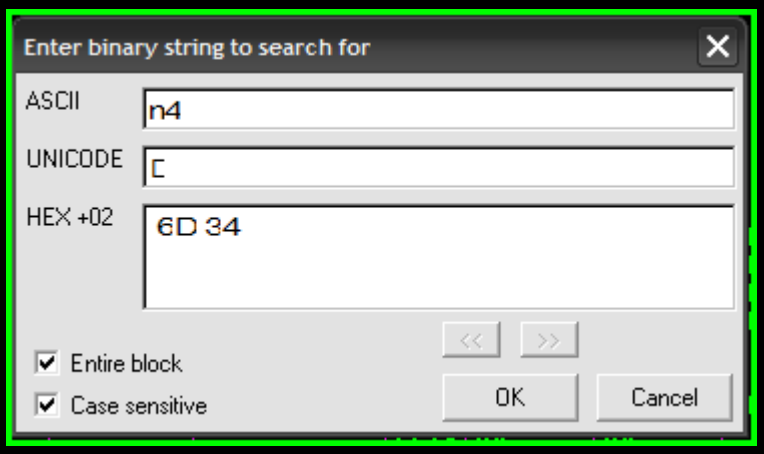


Tiếp theo tôi nhập số "4" :

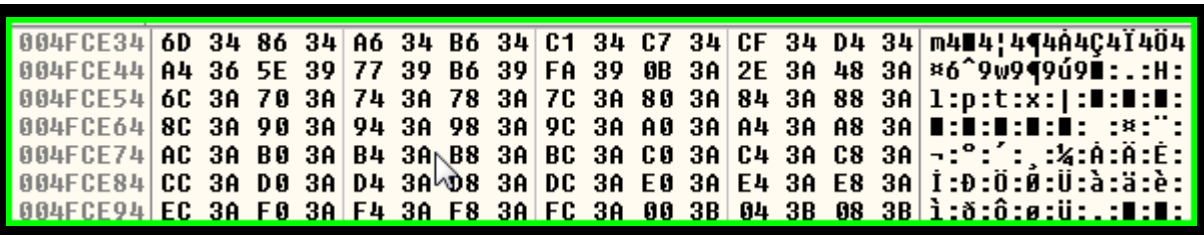


Ok, tạm thời tôi nhập thế thôi đã. Quay trở lại Olly tôi sẽ tiến hành tìm kiếm xem chuỗi tôi vừa nhập vào ở trên sẽ được lưu ở vùng nhớ nào ☺ . Nhấn vào **M** để tới cửa sổ Memory :

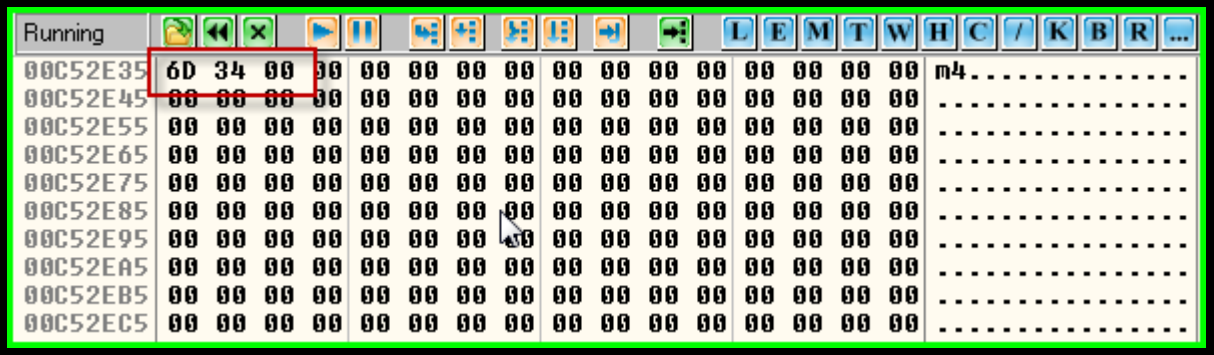
Cuộn chuột lên trên đầu của cửa sổ Memory và tiến hành tìm kiếm chuỗi như sau :



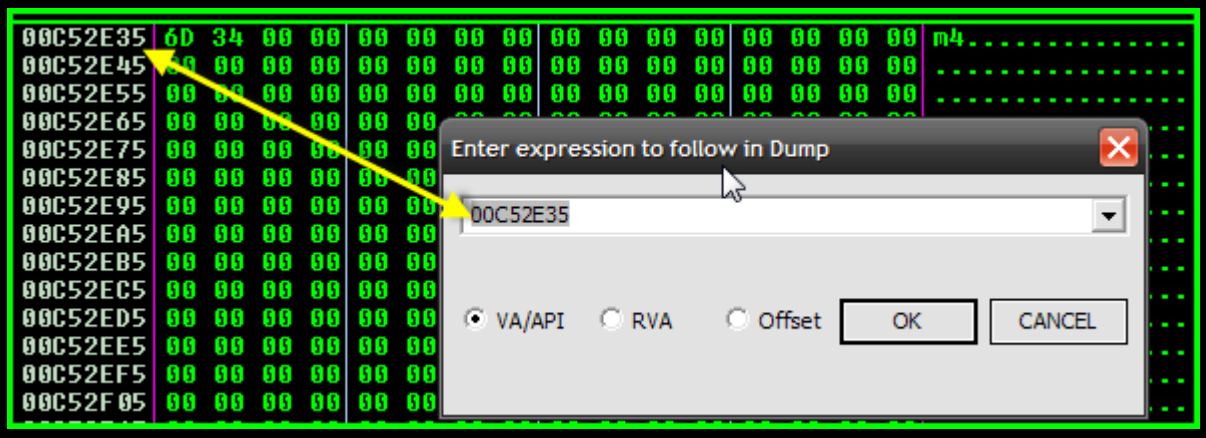
Hehe nếu các bạn nhìn trên hình thì có thể tưởng tôi nhập sai, nhưng thực ra cái Olly của tôi bị sao đó ở cái chức năng Search này, gõ chữ m mà thấy nó toàn hiện n. Nhưng check lại mã hexa thì 0x6d tương ứng với m. Sau đó nhấn OK ta sẽ tới đây :



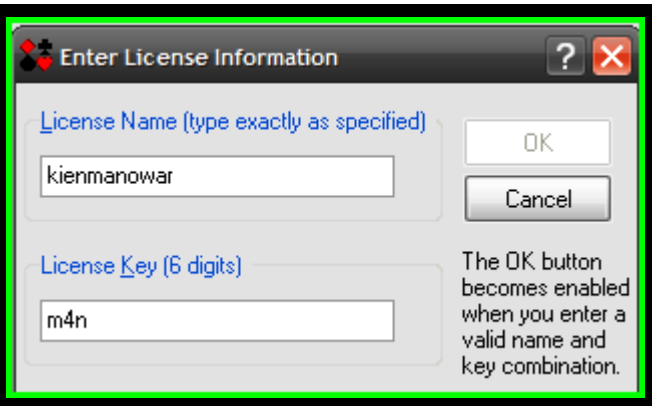
Vùng này thuộc section .rsrc nên không quan tâm. Ta tìm kiếm tiếp, nhấn **Ctrl + L** để Olly tiếp tục tìm ta sẽ dừng lại tại đây :



Đây có thể là vùng nhớ mà ta cần quan tâm. Để khẳng định chắc chắn hơn tôi thử tìm kiếm tiếp, nhưng những thông tin có được hoàn toàn không phù hợp. Như vậy vùng địa chỉ mà tôi tìm kiếm được ở trên là chính xác, ghi nhớ vùng địa chỉ này (có thể trên máy các bạn sẽ khác máy tôi). Quay trở về cửa sổ Code của Olly, vào cửa sổ Dump và nhập địa chỉ `0x00C52E35` vào :

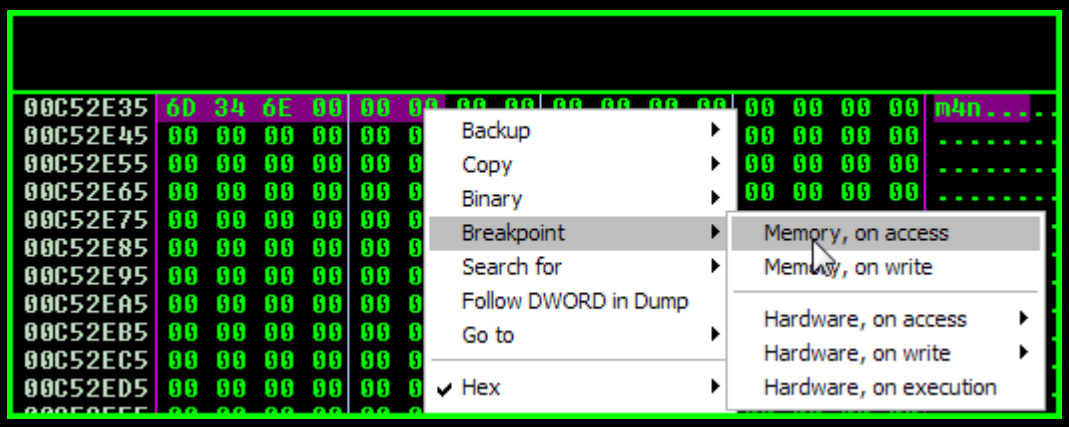


Một lần nữa, để khẳng định vùng nhớ ta tìm được hoàn toàn chính xác là nơi lưu giữ Fake License, chúng ta thử nhập vào kí tự tiếp theo là "n" :



Ngay lập tức ta thấy vùng nhớ thay đổi và chữ "n" nhập vào xuất hiện ngay lập tức :

OK, như vậy là đúng vùng nhớ cần tìm rồi. Trở lại vấn đề về License key, chương trình yêu cầu chúng ta nhập vào 6 số, vậy ở vùng nhớ này sẽ lưu 6 kí tự cho nên tôi sẽ bôi đen 6 bytes tại vùng nhớ này và đặt BP như sau :



Nhớ lại một chút :

BreakPoint Memory on access: Với kiểu đặt bp này lên một vùng nhớ sẽ cho phép chúng ta dừng sự thực thi của chương trình khi có bất kì một sự thực thi, đọc hay ghi lên vùng nhớ mà ta đã đặt bp.

Sau khi đặt BP xong, ta nhập tiếp kí tự tiếp theo là "0" vào, ngay lập tức Olly sẽ break khi có sự truy cập (ở đây là ghi lên vùng nhớ) vùng nhớ mà ta đã thiết lập BP. Ta dừng lại ở đoạn code sau :

```
Paused
004029A1  .  F3:A5          rep     movs dword ptr es:[edi], dword ptr [esi]
004029A3  .  89C1           mov     ecx, eax
004029A5  .  83E1 03        and     ecx, 3
004029A8  .  83C6 03        add     esi, 3
004029AB  .  83C7 03        add     edi, 3
004029AE  .  F3:A4          rep     movs byte ptr es:[edi], byte ptr [esi]
004029B0  .  FC             cld
004029B1  >  5F             pop     edi                                     00C25ACC
004029B2  .  5E             pop     esi                                     00C25ACC
004029B3  L. C3             retn
```

Quan sát cửa sổ chứa thông tin về các thanh ghi :

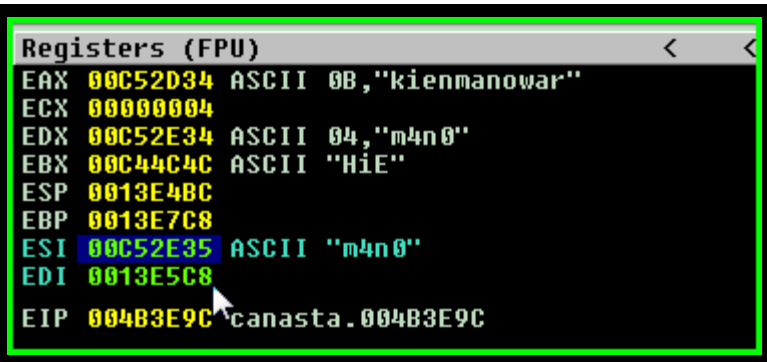
```
Registers (FPU)
EAX 00000004
ECX 00000001
EDX 00C52E35 ASCII "m4n"
EBX 00000004
ESP 0013E7D0
EBP 0013E804
ESI 00C46008 ASCII "m4n0"
EDI 00C52E35 ASCII "m4n"

EIP 004029A1 canasta.004029A1

C 0  ES 0023 32bit 0(FFFFFFFF)
```

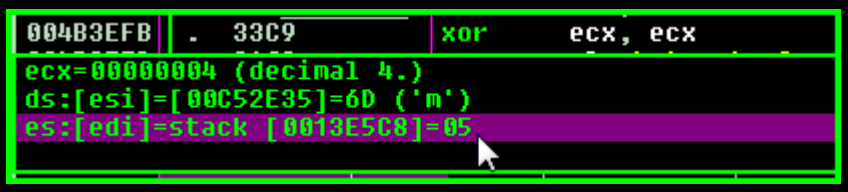
Ta nhận thấy rằng đoạn code này sẽ copy dữ liệu từ ESI vào EDI. Nhấn F**8** để thực thi đoạn code và quan sát vùng nhớ :

```
00C52E35  6D 34 6E 30 00 00 00 00  00 00 00 00  00 00 00 00  m4n0............
00C52E45  00 00 00 00 00 00 00 00  00 00 00 00  00 00 00 00  ................
00C52E55  00 00 00 00 00 00 00 00  00 00 00 00  00 00 00 00  ................
00C52E65  00 00 00 00 00 00 00 00  00 00 00 00  00 00 00 00  ................
00C52E75  00 00 00 00 00 00 00 00  00 00 00 00  00 00 00 00  ................
00C52E85  00 00 00 00 00 00 00 00  00 00 00 00  00 00 00 00  ................
```

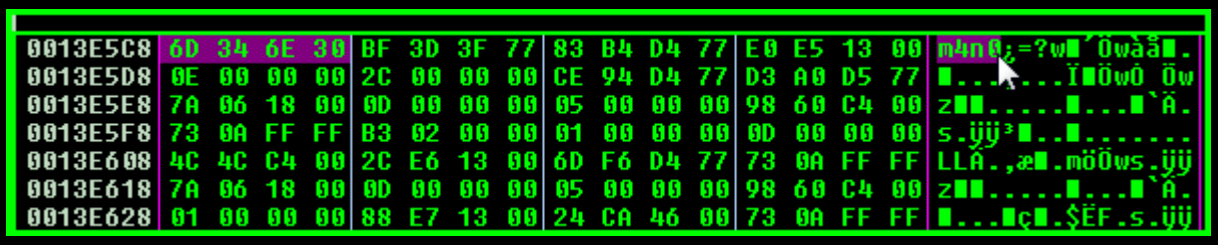
Nhấn **F9** để run chương trình, ta thấy Olly lại break tại một đoạn code khác như sau :

```
004B3E8E  .  57             push    edi
004B3E8F  .  8BF2           mov     esi, edx
004B3E91  .  8DBD FFFDFFFF  lea     edi, dword ptr [ebp-201]
004B3E97  .  33C9           xor     ecx, ecx
004B3E99  .  8A0E           mov     cl, byte ptr [esi]
004B3E9B  .  41             inc     ecx
004B3E9C  .  F3:A4          rep     movs byte ptr es:[edi], byte ptr [esi]
004B3E9E  .  8BF0           mov     esi, eax
004B3EA0  .  8DBD FFFEFFFF  lea     edi, dword ptr [ebp-101]
004B3EA6  .  33C9           xor     ecx, ecx
004B3EA8  .  8A0E           mov     cl, byte ptr [esi]
```

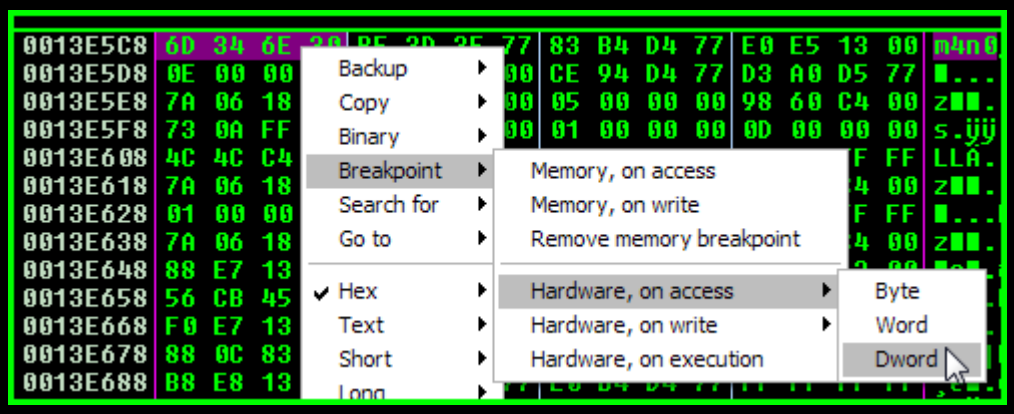




Qua toàn bộ thông tin có được ở trên, ta thấy rằng chương trình sẽ thực hiện copy toàn bộ 4 bytes mà ta nhập vào lên vùng Stack , cụ thể ở đây nó copy vào địa chỉ `0x0013E5C8` . ***Follow in Dump*** tại vùng nhớ này và nhấn **F8** để thực thi đoạn code copy ta thấy được kết quả như sau :



Ok, như vậy nó copy sang vùng mới thì chắc chắn một điều nó sẽ sử dụng vùng nhớ này để làm cái gì đó. Ta đặt tiếp một BP khác như sau :



Tôi sử dụng tới HWBP là vì Memory BP tôi đã sử dụng rồi và tôi muốn linh hoạt hơn trong việc sử dụng các BP khác nhau ☺. Nếu ta trace thêm một chút nữa ta cũng sẽ nhận thấy Name mà ta nhập vào cũng sẽ được copy lên một vùng nhớ khác của Stack :

```
004B3EA0  .  8DBD FFFEFFFF lea     edi, dword ptr [ebp-101]
004B3EA6  .  33C9          xor     ecx, ecx
004B3EA8  .  8A0E          mov     cl, byte ptr [esi]
004B3EAA  .  41            inc     ecx
004B3EAB  .  F3:A4         rep     movs byte ptr es:[edi], byte ptr [esi]
004B3EAD  .  80BD FFFEFFFF cmp     byte ptr [ebp-101], 0
004B3EB4  .v 0F84 F6000000 je      004B3FB0
```

```
Registers (FPU)
EAX 00C52D34 ASCII 0B,"kienmanowar"
ECX 0000000C
EDX 00C52E34 ASCII 04,"m4n0"
EBX 00C44C4C ASCII "HiE"
ESP 0013E4BC
EBP 0013E7C8
ESI 00C52D34 ASCII 0B,"kienmanowar"
EDI 0013E6C7

EIP 004B3EAB canasta.004B3EAB

C 0  ES 0023 32bit 0(FFFFFFFF)
```

```
ecx=0000000C (decimal 12.)
ds:[esi]=[00C52D34]=0B
es:[edi]=stack [0013E6C7]=00
```

Follow in Dump tại địa chỉ mà thanh ghi EDI đang giữ. Nhấn **F8** để trace qua đoạn code copy, qua sát vùng nhớ ta có thông tin về Name được copy tới như sau :

```
0013E6C7  0B 6B 69 65 6E 6D 61 6E 6F 77 61 72 00 C0 91 18  ■kienmanowar.À‘■
0013E6D7  00 14 00 00 00 01 00 00 00 D1 75 F1 77 20 30 F5  .■...■...Ñuñw 0õ
0013E6E7  77 00 00 00 00 00 00 00 00 40 6B F1 77 00 00 00  w........@kñw...
0013E6F7  00 14 E7 13 00 58 6B F1 77 99 0E 01 67 C0 91 18  .■ç■.Xkñw■■■gÀ‘■
0013E707  00 14 00 00 00 38 E7 13 00 01 00 00 00 48 E7 13  .■...8ç■.■...Hç■
```

Giá trị `0x0B` cho ta biết đó là chiều dài của chuỗi Name. Giá trị này sẽ được lấy ra so sánh để với 0x0 thông qua đoạn code bên dưới lệnh copy :

```
004B3EAD  |.  80BD FFFEFFFF>cmp     byte ptr [ebp-101], 0
```

Qua khỏi đoạn code so sánh ta sẽ bắt gặp một đoạn code thú vị sau :

```
004B3EAD  .  80BD FFFEFFFF cmp     byte ptr [ebp-101], 0
004B3EB4  .v 0F84 F6000000 je      004B3FB0
004B3EBA  .  8D85 FFFEFFFF lea     eax, dword ptr [ebp-101]
004B3EC0  .  BA 5C404B00   mov     edx, 004B405C                ASCII 03,"TNO"
004B3EC5  .  33C9          xor     ecx, ecx
004B3EC7  .  8A08          mov     cl, byte ptr [eax]
004B3EC9  .  41            inc     ecx
004B3ECA  .  E8 E9F2F4FF   call    004031B8
004B3ECF  .v 0F84 DB000000 je      004B3FB0
004B3ED5  .  8D85 FFFEFFFF lea     eax, dword ptr [ebp-101]
004B3EDB  .  BA 60404B00   mov     edx, 004B4060                ASCII 05,"afdad"
004B3EE0  .  33C9          xor     ecx, ecx
004B3EE2  .  8A08          mov     cl, byte ptr [eax]
004B3EE4  .  41            inc     ecx
004B3EE5  .  E8 CEF2F4FF   call    004031B8
004B3EEA  .v 0F84 C0000000 je      004B3FB0
004B3EF0  .  8D85 FFFEFFFF lea     eax, dword ptr [ebp-101]
004B3EF6  .  BA 68404B00   mov     edx, 004B4068                ASCII 05,"Gauss"
004B3EFB  .  33C9          xor     ecx, ecx
004B3EFD  .  8A08          mov     cl, byte ptr [eax]
004B3EFF  .  41            inc     ecx
004B3F00  .  E8 B3F2F4FF   call    004031B8
004B3F05  .v 0F84 A5000000 je      004B3FB0
004B3F0B  .  8D85 FFFEFFFF lea     eax, dword ptr [ebp-101]
004B3F11  .  BA 70404B00   mov     edx, 004B4070                ASCII 16,"STaRDoGG CHaMPioN PC97"
004B3F16  .  33C9          xor     ecx, ecx
```

Toàn bộ đoạn code mà ta thấy dùng để kiểm tra Name nhập vào có nằm trong danh sách Black list mà chương trình liệt kê ra không (chắc mấy cái tên này thường xử lý chương trình này nên bị ghét và đưa vào Black list lolz ☺). Do tên của tôi không thuộc dạng nằm trong bảng vàng thành tích nên cứ thế mà nhấn **F9** để mò tiếp, sau khi nhấn **F9** ta sẽ dừng lại tại đoạn code sau :

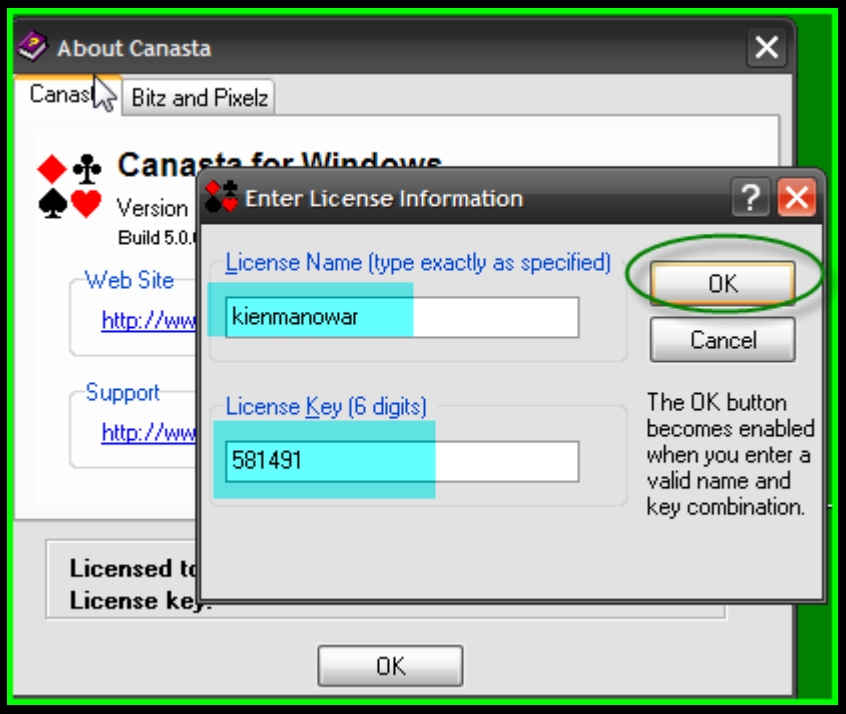
```
004031C2  >  8B08          mov     ecx, dword ptr [eax]
004031C4  .  8B1A          mov     ebx, dword ptr [edx]
004031C6  .  39D9          cmp     ecx, ebx
004031C8  .v 75 45         jnz     short 0040320F
004031CA  .  4E            dec     esi
004031CB  .v 74 15         je      short 004031E2
004031CD  .  8B48 04       mov     ecx, dword ptr [eax+4]
004031D0  .  8B5A 04       mov     ebx, dword ptr [edx+4]
004031D3  .  39D9          cmp     ecx, ebx
004031D5  .v 75 38         jnz     short 0040320F
004031D7  .  83C0 08       add     eax, 8
004031DA  .  83C2 08       add     edx, 8
004031DD  .  4E            dec     esi
004031DE  .^ 75 E2         jnz     short 004031C2
```

Ngó sang bên cửa sổ **Registers** để xem ta có thông tin gì nào :

```
Registers (FPU)
EAX 0013E5C7 ASCII 04,"m4n0"
ECX 6E346D04
EDX 0013E4C7 ASCII 06,"581491"
EBX 00C44C4C ASCII "HiE"
ESP 0013E4F8
EBP 0013E7C8
ESI 00000001
EDI 0013E6D3
```

Ồ ta có gì đây, một chuỗi gồm 6 số lưu tại vùng nhớ `0x0013E4C7`, chuỗi này sau đó được sao chép vào thanh ghi ebx để thực hiện việc so sánh với giá trị hexa của Fake License lưu tại thanh ghi ECX : `004031C6  |.  39D9          |cmp     ecx, ebx`

Vậy qua đó ta kết luận Valid License mà chúng ta cần tìm tương ứng Name nhập vào là : "**581491**". Giờ ta xóa hết các BP đã đặt, sau đó nhấn F9 để thực thi chương trình. Nhập license hợp lệ mà ta vừa tìm được và ta có được kết quả như sau :



Nút OK đã được Enable ☺. Như vậy là ta đã xử lý xong chương trình này rồi.

III. Kết luận

Như các bạn thấy sẽ có nhiều cách để tiếp cận một mục tiêu, khuôn khổ bài viết có hạn cho nên tôi sẽ không trình bày các cách khác ở đây, có thể các bạn sẽ tìm ra cho mình một cách làm hay hơn tôi, nhưng đường nào rồi cũng tới một đích mà thôi ☺. Ở bài viết sau tôi sẽ trình bày thêm một kĩ thuật mới đó là đặt BP dựa vào Windows Message, cụ thể là WM_KEYUP (có thể dùng để áp dụng với bài viết này). Tuy nhiên, như tôi đã nói không có gì phải vội vàng cả, chúng ta cứ từ từ tìm hiểu. Dần dần sẽ học thêm được nhiều mẹo hay. Hẹn gặp lại các bạn ở bài 18.

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