

IDA Pro Advanced_N0w 0r N3v3r

Author: _[kienmanowar]_

"Sóng gió cuộc đời chợt đến có ai chờ đợi.
Biết đâu một ngày phận người que diêm trước gió.
Lụi tàn trong một sớm không ngoài ai.
Nhưng với một triệu người
Hơi ấm sẽ chia từng người
Những đôi tay trần dầu nhau qua cơn khốn khó
Tâm hồn luôn rộng mở rất chân thành"

"Rock mang đến cho tôi sự hứng khởi, lòng đam mê, tình yêu cháy bỏng, sự cuồng nhiệt và niềm tin vào cuộc sống. Giúp tôi vững bước trước mọi khó khăn sóng gió cuộc đời."

I. Intro :

Chào tất cả các anh em REA, chúng ta lại gặp nhau trong phần tiếp theo tôi viết về IDA. Đây là bài viết của tác giả **Medardus**, có thể coi đây như là một "đơn đặt hàng" của anh Be, và tôi là người đứng ra nhận "bảo kê" cho "đơn đặt hàng" này. Thực tế tôi đã có ý định viết bài này từ rất lâu nhưng... hii lại có từ nhưng, vì thời gian bận bịu quá, sắp tới công việc lại có một số thay đổi mà chưa biết sẽ thế nào. Như đã hứa với anh em, bài viết trước tôi đã giới thiệu sơ qua một số tính năng của IDA, đủ để thuyết phục những người dù là khó tính nhất. Trong bài viết này tôi sẽ ứng dụng sử dụng IDA trong quá trình Reverse một Crackme. Hii trình độ tôi vẫn còn kém cỏi, không biết có đủ sức để hầu chuyện anh em không nữa ☺. Okie, N0w L3t's g0!!!!

II. IDA or W32Dasm.... N0w 0r N3v3r!!

Từ những ngày đầu tiên chập chững bước vào tìm hiểu thế giới Cracking & Reversing, những tut đầu tiên mà tôi đọc là của bậc đàn anh Ngô Vĩnh Hoàng hay còn được biết đến với nickname là NVH(c), nhớ lại cái thủa ban đầu ngơ ngác ấy tôi chẳng biết cái quái gì cả, chỉ biết đọc, down công cụ và làm theo một cách hoàn toàn dập khuôn, cứ 74->75, 75->74, không được nữa thì hehe Nop, nop, nop mà ít đầu tư vào suy nghĩ tại sao lại như thế. Mà hầu hết các tut thời đó đều sử dụng W32Dasm, Hiew, SoftIce v...v.. Sh1t!! nghĩ lại cái lúc hì hục cái SoftIce đến nhức, đọc và làm theo y như hướng dẫn mà lần nào Debug không treo máy thì lại bị Crash, hic cú quá tôi nghỉ chơi với SoftIce. Rồi sau này đọc các tut của lão nhỏ, tham gia vào REA ngay từ những ngày đầu tiên tôi dần dần tìm hiểu từ từ từng tí một. Các tut thời gian đó cũng hầu hết tập trung vào SoftIce, W32Dasm nhưng có sự xuất hiện của một Tool debugger mới mà hiện nay các bạn đang sử dụng đó, không nói chắc các bạn cũng biết đó chính là "em" Ollydbg. Hiii sân chơi của REA lúc đó đâu có được đông như bây giờ, chỉ có vài anh em, cầm đầu là anh Còm, anh Moon, lão Zom, bác RongchauA, bác Deux, chú Little, trong đó người có nhiều bài viết nhất là anh Moon, khà khà tôi, anh Be, lão Q là một trong những member đầu tiên và cũng là những người trụ lại lâu nhất trong REA. Có thể nói REA lúc đó là một tập thể tuy chưa hiểu hết về nhau nhưng đã có cảm giác như anh em một nhà, rất gần bó, trao đổi thẳng thắn, giúp đỡ nhau nhiệt tình và cho đến bây giờ vẫn vậy. Có thể tự hào mà nói rằng REA là một trong những 4room có "chất" nhất ở Việt Nam.

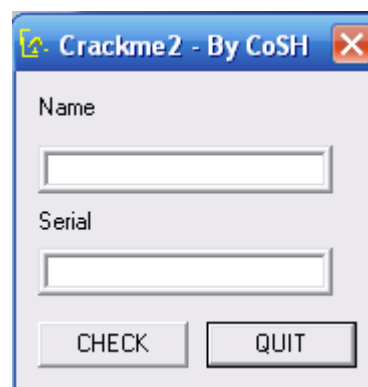
Hehe hơi lạc đề thì phải, như các bạn thấy thời đó công cụ Disassembly được ưa chuộng nhất là W32Dasm, có thể nói nó là công cụ của mọi Newbies, đơn giản, dễ học và dễ sử dụng. Nhưng sau này khi được tiếp xúc nhiều hơn với Cracking & Reversing, tôi thấy có một công cụ được nhắc đến khá nhiều đó là IDA được dùng cho cả hai giới Reversing lẫn Security. Lúc đó tôi tự hỏi "IDA là công cụ gì mà kinh thế?" nhưng rất tiếc không có câu trả lời. Sau này khi W32Dasm không còn được phát triển nữa, nó không đáp ứng được những chương trình có mức độ Protect cao cũng như những chương trình bị Packed, hic hic không còn cách nào khác tôi đành nhắm mắt đưa chân, download em IDA về "ngịch" thử xem thế nào. Phải nói rằng cảm giác đầu tiên khi mở IDA lên là một cảm giác choáng ngợp đến khó tả, vừa khó dùng lại vừa lăm chực năng, rối rắm và phức tạp ☺.

IDA là một chương trình Disassembler được phát triển bởi hãng DataRescue. Những ai đã từng sử dụng qua W32Dasm thì chắc hẳn sẽ biết Disassembler là như thế nào. Đối với những người chưa biết gì thì có thể nói: **Disassembler là một chương trình, mà theo đó nhờ vào sự giúp đỡ, hỗ trợ của chương trình này nó sẽ chuyển đổi code của một file (exe hay dll) về dạng mã assembler, được sắp xếp lại theo một khuôn dạng dễ hiểu và dễ đọc hơn.** IDA hỗ trợ đầy đủ

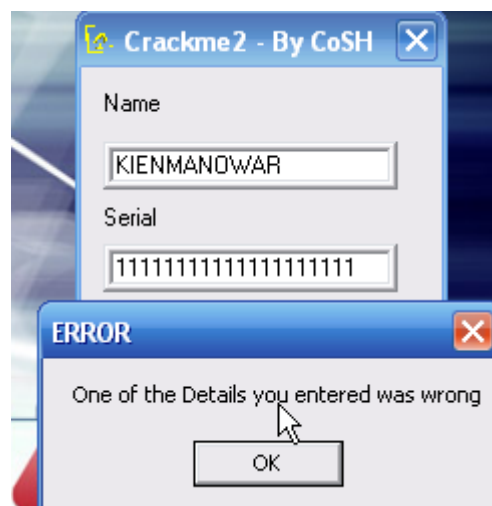
các tính năng có trên W32Dasm, thêm vào đó nó còn hỗ trợ thêm những tính năng cao cấp khác mà W32Dasm không có, ví dụ như : **FLIRT (Nearly Library Identification and Recognition Technology)**, giúp đỡ cho việc nhận diện các hàm phù hợp từ các thư viện khác nhau và hỗ trợ các chú thích cung cấp cho việc mô tả. Sẽ rất ít người trả lời cho bạn tại sao IDA lại được sử dụng nhiều đến thế, chỉ còn mỗi một cách là tự mình tìm hiểu và chất lọc. Vậy còn chần chừ gì nữa mà không tìm hiểu về IDA.

III. Play with Crackme

Oki, tốn bao giấy mực mới đến phần quan trọng nhất của bài viết này, đó là sử dụng IDA để thực hành với 1 Crackme. Tại sao lại là Crackme vì đơn giản nó có kích thước nhỏ và dễ dàng trong việc Demo những kiến thức, thêm nữa là IDA sẽ disassembly rất nhanh còn nếu không bạn phải chờ dài cổ khà khà. Trong bài ngày hôm nay tôi sẽ sử dụng Crackme là **CoSH Crackme #2.exe**. Run thử Crackme này lên xem thế nào, đây chính là một bước quan trọng trong quá trình Reverse, tương tự như ta tiếp xúc với một người, ta phải quan sát thái độ, hành vi của người đó trước để còn liệu đường mà xử lý lại cho phải phép ☺.

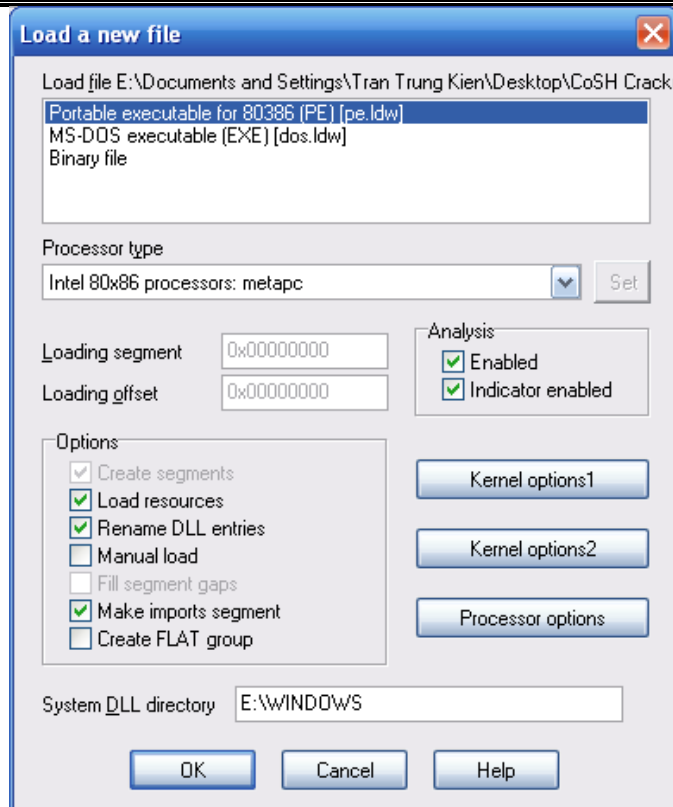


Nhập thử Fake Name và Fake Serial vào xem thế nào, chúng ta nhận được thông báo sau :



Ok như vậy quá trình “trình sát” thế là đủ rồi, chúng ta đã có được thông tin rất quan trọng. Việc tiếp theo bây giờ là mở IDA lên và load Crackme vào trong IDA.

Ngay lập tức tại cửa sổ chương trình IDA các bạn sẽ nhận được một hộp thoại như sau :



Tại hộp thoại thông báo này, chúng ta có thể chỉnh sửa, điều chỉnh các thông số của IDA để thay đổi cách ứng xử của IDA với các file được Disassembly, đồng thời IDA cũng cung cấp thông tin về kiểu Processor được sử dụng. Nhấn OK, IDA sẽ tiến hành quá trình phân tích và disassembly file Crackme của chúng ta, quá trình này phụ thuộc vào kích thước cũng như độ phức tạp của chương trình mà IDA có thể mất ít hoặc nhiều thời gian để hoàn tất công việc của mình. Trong quá trình IDA tiến hành Disassembly, các bạn có thể dạo qua các cửa sổ trong chương trình như Names, Imports, Functions v..v..thậm chí là xem mã nguồn của chương trình đã được Disassembly trong cửa sổ IDA View-A, tuy nhiên tôi khuyên bạn không nên làm những điều này, hãy cứ để cho IDA hoàn thành nhiệm vụ của nó đã, cho đến khi bạn nhận được một thông báo sau từ IDA :

```
Using FLIRT signature: Microsoft VisualC 2-7/net runtime
Using FLIRT signature: MFC 3.1/4.0/4.2/7.1 32bit
Propagating type information...
Function argument information is propagated
The initial autoanalysis is finished.
```

Đó chính là lúc mà IDA đã hoàn toàn hoàn tất nhiệm vụ rất tuyệt vời của nó là phân tích code của chương trình và các References. Nhiệm vụ tiếp theo bây giờ thuộc về chúng ta. Tương tự như trong W32Dasm điều đầu tiên mà tôi và các bạn quan tâm là gì, đó chính là các String References, chúng ta sẽ đi tìm đoạn thông báo ở trên khi mà chúng ta nhập các thông tin sai. Trong W32Dasm thì công việc này quá dễ rồi, nhưng trong IDA thì chúng ta phải tìm ở đâu. Rất may mắn cho chúng ta là IDA cung cấp một cửa sổ là "**Names**", nơi đây chứa các String mà chúng ta quan tâm.

Vậy cửa sổ Names này nằm ở đâu, nó nằm trong Menu **View>Open Subviews>Names (Shift+F4)**, trong cửa sổ này chúng ta sẽ thấy được tất cả các References, không chỉ có các Strings mà còn có các các hàm trong chương trình, đã được IDA nhận diện rất hoàn hảo.

Name	Address	P.
F CWnd::PreSubclassWindow(void)	004017CC	
F CWnd::GetMessageMap(void)	004017D2	
F CWnd::OnFinalRelease(void)	004017D8	
F CEdit::GetRuntimeClass(void)	004017DE	
F AfxFindResourceHandle(char const *,char const *)	004017E4	
F AfxGetModuleState(void)	004017EA	
F CString::operator=(char const *)	004017F0	
F CString::CString(void)	004017F6	
F CWnd::CWnd(void)	004017FC	
F CDialog::CDialog(uint,CWnd *)	00401802	
F DDV_MaxChars(CDataExchange *,CString const &,int)	00401808	
F DDV_Text(CDataExchange *,int,CString &)	0040180E	
F DDV_Control(CDataExchange *,int,CWnd &)	00401814	
F CDialog::OnInitDialog(void)	0040181A	
F CWnd::Default(void)	00401820	
F CPaintDC::~CPaintDC(void)	00401826	
F CPaintDC::CPaintDC(CWnd *)	0040182C	
F operator+(CString const &,char const *)	00401832	
F operator+(char const *,CString const &)	00401838	

Vậy trong cái mớ bong bóng này làm sao tôi tìm được cái mà tôi cần bây giờ. Thật may mắn IDA đã làm hộ chúng ta, nó sẽ đánh dấu các string mà nó tìm được trong chương trình bằng chữ cái "A", cuộn chuột xuống dưới bạn sẽ tìm thấy được như hình minh họa dưới đây :

LoadIconA	00402200
A szYOUIDIDIT	00403020
A szWelldone	0040302C
A szOneoftheDetailsyouenteredwaswrong	00403038
A szERROR	00403064

Tính tới thời điểm này, tôi đã sử dụng các file *.cfg của anh TQN, cho nên các bạn sẽ thắc mắc là tại sao không thấy có chữ "a" ở đầu các String như trong tut đầu tiên tôi đã viết, đơn giản là vì nó đã được thay bằng "sz". Do đó tại cửa sổ Names các bạn chỉ việc gõ "sz" trên bàn phím, IDA sẽ đưa bạn đến nơi có các String. Khả khả như các bạn thấy, trên hình trên chúng ta có được gì nào, đó có phải là cái mà chúng ta đang đi tìm không nhỉ, xin thưa rằng đúng là thứ mà tôi và các bạn này giờ tìm đồ con mắt đó ☺.

Ngoài cái String mà chúng ta cần tìm bạn sẽ thấy bên cạnh đó sẽ có nhiều String khác cũng đáng quan tâm. Nhấn đúp chuột vào String mà tôi khoanh đỏ ở trên, IDA sẽ đưa chúng ta tới nơi mà String đó đang ở tại.

```

.data:0040301F 00 db 0
.data:00403020 59 4F 55 20 44 49 44 20+ szYOUIDIDIT db 'YOU DID IT',0 ; DATA XREF: .text:0040157F↑o
.data:0040302B 00 align 4
.data:0040302C 57 65 6C 6C 20 64 6F 6E+ szWelldone db 'Well done,',0 ; DATA XREF: .text:00401558↑o
.data:00403037 00 align 4
.data:00403038 4F 6E 65 20 6F 66 20 74+ szOneoftheDetailsyouenteredwaswrong db 'One of the Details you entered was w
.data:00403038 68 65 20 44 65 74 61 69+ ; DATA XREF: .text:0040153D↑o
    
```

Thật không thể tin được quá trực quan và trong sáng. Qua đây chúng ta thấy được String này được đặt trong Section .data của chương trình. Okie như vậy là chúng ta đã biết được vị trí của String rồi, nhưng như thế là chưa đủ, tôi muốn biết String này sẽ được sử dụng ở đâu trong chương trình khi mà tôi nhập thông tin sai nó sẽ hiện thị ra thông qua một MessageBox.


Một lần nữa, thật là may mắn IDA đã chỉ cho chúng ta vị trí của đoạn code sẽ sử dụng đến String này, đó chính là dòng :

'One of the Details you entered was wro
; DATA XREF: .text:0040153D↑

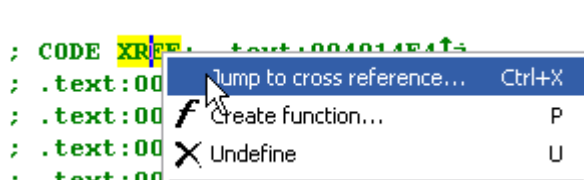
Dòng trên cho chúng ta những ý nghĩa gì, đầu tiên là **XREF** (viết tắt của cross-reference), tiếp theo chúng ta sẽ biết rằng biến String của chúng ta sẽ được sử dụng duy nhất ở một nơi đó là Section `.text` tại địa chỉ `0x0040153D`. Tức là nó nằm ở phía trên vị trí khai báo của String (`0x00403038`). Bên cạnh đó là chữ "o" ám chỉ cho chúng ta biết đây là "Offset". Nếu như bạn thấy chữ "j" thì đó là viết tắt của "Jump" còn chữ "p" sẽ là viết tắt của "Procedure".

Okie vậy là đã rõ, ta đã biết được vị trí nơi mà biến String của chúng ta được sử dụng, để tới được đoạn code đó rất đơn giản bạn chỉ cần nhấp đúp chuột vào dòng **;DATA XREF** IDA sẽ đưa bạn đến nơi.

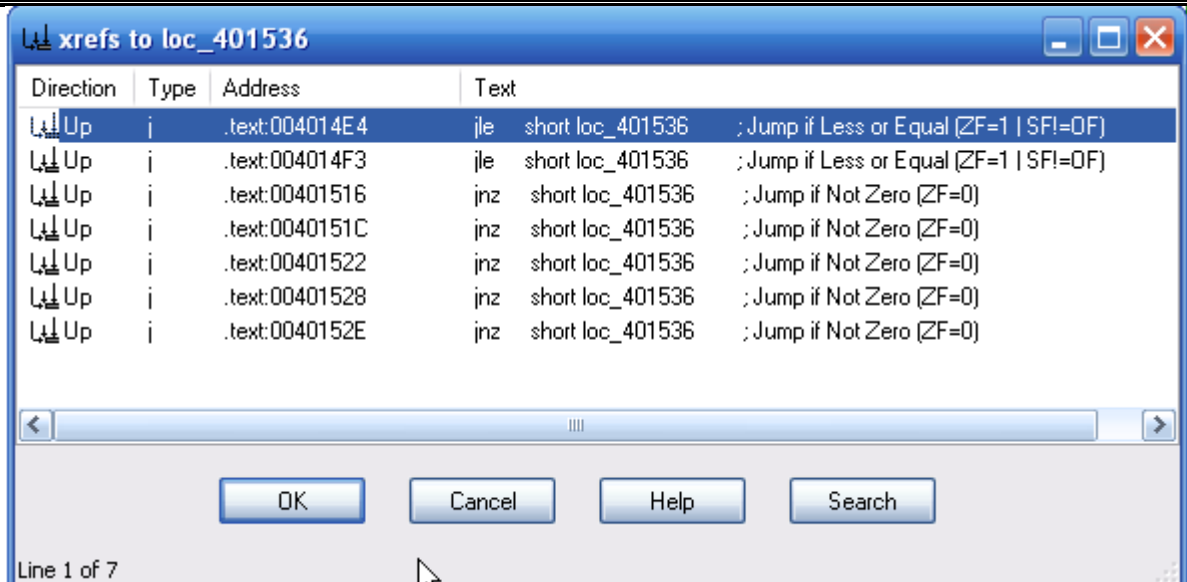
```
.text:00401522
.text:00401524 80 78 03 37      cmp     byte ptr [eax+3], 37h ; Compare Two Operands
.text:00401528 75 0C             jnz     short loc_401536     ; Jump if Not Zero (ZF=0)
.text:00401528
.text:0040152A 80 78 04 2D      cmp     byte ptr [eax+4], 2Dh ; Compare Two Operands
.text:0040152E 75 06             jnz     short loc_401536     ; Jump if Not Zero (ZF=0)
.text:0040152E
.text:00401530 80 78 05 41      cmp     byte ptr [eax+5], 41h ; Compare Two Operands
.text:00401534 74 17             jz      short loc_40154D     ; Jump if Zero (ZF=1)
.text:00401534
.text:00401536
.text:00401536      loc_401536:                ; CODE XREF: .text:004014E4↑j
.text:00401536      ; .text:004014F3↑j
.text:00401536      ; .text:00401516↑j
.text:00401536      ; .text:0040151C↑j
.text:00401536      ; .text:00401522↑j
.text:00401536      ; .text:00401528↑j
.text:00401536      ; .text:0040152E↑j
.text:00401536 6A 00            push    0
.text:00401538 68 64 30 40 00   push    offset szERROR      ; "ERROR"
.text:0040153D 68 38 30 40 00   push    offset szOneoftheDetailsyouenteredwaswrong ; "One of the Det
.text:00401542 8B CE            mov     ecx, esi
.text:00401544 E8 F5 02 00 00   call   CWnd::MessageBoxA(char const *,char const *,uint) ; Call Pro
.text:00401544
```

Hii trong IDA có một tính năng khá thú vị tương tự như trên IE hay trên các trình Web Browser khác, đó chính là tính năng . Cho phép chúng ta quay trở lại vị trí trước đó đơn giản bằng cách nhấn vào đó.

Oki sau khi nhấp đúp chuột chúng ta đã ở chỗ đoạn code của chương trình sử dụng tới String mà chúng ta quan tâm. Nó bắt đầu tại chỗ mà được đặt tên là `loc_401536`, bên cạnh đó ta sẽ thấy có rất nhiều lệnh nhảy nhảy tới vị trí này, thông qua những **XREF** được liệt kê ở bên cạnh. Và để tìm tới vị trí của từng lệnh nhảy này chúng ta thực hiện tương tự như đã nói ở trên. Ngoài ra IDA còn cung cấp cho chúng ta một tính năng nữa cũng tương tự như trong Ollydbg khi bạn muốn xem chi tiết hơn về các lệnh nhảy này, chỉ việc nhấp chuột phải trên dòng **XREF** và chọn :



Ngay lập tức một cửa sổ xuất hiện cho chúng ta biết được các lệnh nhảy sẽ nhảy tới `loc_401536`, kèm thêm đó là các thông tin như Direction (cho chúng ta biết lệnh đó ở trên hay là ở dưới `loc_401536`) v.v...:



Chúng ta sẽ tới vị trí của lệnh nhảy thứ nhất và sẽ phân tích xem tại đó sẽ có lệnh so sánh thế nào mà khiến cho lệnh nhảy này sẽ nhảy tới chỗ bắn ra Nag. Nhấn đúp chuột tại `.text:004014E4` chúng ta sẽ tới đây :

```
.text:004014D4 8B CF          mov     ecx, edi
.text:004014D6 E8 6F 03 00 00 call   CWnd::GetWindowTextLengthA(void) ; Call Procedure
.text:004014D6
.text:004014DB 8B 1D FC 21 40 00 mov     ebx, ds:PostQuitMessage
.text:004014E1 83 F8 05      cmp     eax, 5 ; Compare Two Operands
.text:004014E4 7E 50        jle     short loc_401536 ; Jump if Less or Equal (ZF=1 | SF!=0F)
.text:004014E4
.text:004014E6 8D 6E 60      lea    ebp, [esi+60h] ; Load Effective Address
.text:004014E9 8B CD        mov     ecx, ebp
.text:004014EB E8 5A 03 00 00 call   CWnd::GetWindowTextLengthA(void) ; Call Procedure
.text:004014EB
.text:004014F0 83 F8 05      cmp     eax, 5 ; Compare Two Operands
.text:004014F3 7E 41        jle     short loc_401536 ; Jump if Less or Equal (ZF=1 | SF!=0F)
.text:004014F3
.text:004014F5 8D 86 E0 00 00 00 lea    eax, [esi+0E0h] ; Load Effective Address
.text:004014FB 8B CD        mov     ecx, edi
.text:004014FD 50          push   eax
.text:004014FE E8 41 03 00 00 call   CWnd::GetWindowTextA(CString &) ; Call Procedure
.text:004014FE
.text:00401503 8D BE E4 00 00 00 lea    edi, [esi+0E4h] ; Load Effective Address
.text:00401509 8B CD        mov     ecx, ebp
.text:0040150B 57          push   edi
.text:0040150C E8 33 03 00 00 call   CWnd::GetWindowTextA(CString &) ; Call Procedure
```

Như hình trên, trước lệnh nhảy là một lệnh so sánh `cmp eax, 5`, mà trước lệnh so sánh này tôi thấy có một hàm API là `GetWindowTextLengthA` có nhiệm vụ tính ra chiều dài của chuỗi mà chúng ta nhập vào, sau đó kết quả thu được sẽ lưu vào thanh ghi `eax`. Trong hình trên các bạn để ý thấy hai chỗ có lệnh `cmp eax, 5` tương ứng với việc tính toán chiều dài cho chuỗi Name và Serial mà chúng ta nhập vào, như vậy kết luận là 2 chuỗi chúng ta nhập phải có độ dài lớn hơn 5 (hehe không tin các bạn thử nhập nhỏ hơn hoặc bằng 5 xem).

Khi chúng ta nhập thỏa điều kiện trên chúng ta sẽ vượt qua được 2 lệnh nhảy kiểm tra tính hợp lệ của 2 chuỗi Name và Serial nhập vào, mà cụ thể chiều dài của chúng phải lớn hơn 5. Tiếp đó ta sẽ gặp hai lệnh API là `GetWindowTextA`, có nhiệm vụ copy text của chúng ta nhập vào và lưu vào một vùng nhớ Buffer để dùng cho việc xử lý với đoạn text đó sau này. Đối với hàm đầu tiên nó sẽ copy chuỗi Name của chúng ta vào một Buffer, bạn để ý sau đó sẽ không có quá trình xử lý với chuỗi Name, như vậy kết luận là Name nhập vào lớn hơn 6 và không có quá trình nào tính toán với Name. Lệnh tiếp theo sau đó cũng copy chuỗi chúng ta nhập vào và lưu vào 1 vùng Buffer, nhưng lần này là chuỗi Serial. Và ngay sau đó chúng ta sẽ thấy được quá trình tính toán xử lý với chuỗi Serial như sau :

```
.text:0040150B 57          push     edi
.text:0040150C E8 33 03 00 00 call    CWnd::GetWindowTextA(CString &) ; Call Procedure
.text:0040150E
.text:00401511 8B 07      mov     eax, [edi]
.text:00401513 80 38 36   cmp    byte ptr [eax], 36h ; Compare Two Operands
.text:00401516 75 1E      jnz    short loc_401536 ; Jump if Not Zero (ZF=0)
.text:00401518 80 78 01 32 cmp    byte ptr [eax+1], 32h ; Compare Two Operands
.text:0040151C 75 18      jnz    short loc_401536 ; Jump if Not Zero (ZF=0)
.text:0040151E 80 78 02 38 cmp    byte ptr [eax+2], 38h ; Compare Two Operands
.text:00401522 75 12      jnz    short loc_401536 ; Jump if Not Zero (ZF=0)
.text:00401524 80 78 03 37 cmp    byte ptr [eax+3], 37h ; Compare Two Operands
.text:00401528 75 0C      jnz    short loc_401536 ; Jump if Not Zero (ZF=0)
.text:0040152A 80 78 04 2D cmp    byte ptr [eax+4], 2Dh ; Compare Two Operands
.text:0040152E 75 06      jnz    short loc_401536 ; Jump if Not Zero (ZF=0)
.text:00401530 80 78 05 41 cmp    byte ptr [eax+5], 41h ; Compare Two Operands
.text:00401534 74 17      jz     short loc_40154D ; Jump if Zero (ZF=1)
```

Như đã giải thích ở trên hàm `GetWindowTextA`, sẽ copy text vào Buffer. Các bạn sẽ hỏi Buffer này là gì, xin thưa nó cũng là một địa chỉ nhưng khác với các địa chỉ khác nó sẽ chứa chuỗi Serial của chúng ta. Còn thanh ghi `edi` trong lệnh Push trước hàm API này sẽ được dùng để chứa địa chỉ của Buffer này. Để giải thích một cách cụ thể hơn cho các bạn hình dung rõ ràng tôi sẽ lấy Ollydbg ra để demo cho bạn thấy :

```
0040150B . 57          push edi
0040150C . E8 33030000 call <jmp.&MFC42.#3874>
00401511 . 8B07      mov eax,dword ptr ds:[edi]
00401513 . 8038 36   cmp byte ptr ds:[eax],36
00401516 . 75 1E      jnz short CoSH_Cra.00401536
00401518 . 8078 01 32 cmp byte ptr ds:[eax+1],32
0040151C . 75 18      jnz short CoSH_Cra.00401536
0040151E . 8078 02 38 cmp byte ptr ds:[eax+2],38
00401522 . 75 12      jnz short CoSH_Cra.00401536
edi=0012FEF0
```

Vùng tôi khoanh đỏ tương đương những gì bạn thấy trong IDA, nhìn xuống dưới bạn sẽ thấy giá trị của `edi` là (`edi=0x0012FEF0`) , ta sẽ follow in Dump xem giá trị tại `edi` lúc này là bao nhiêu :

```
edi=0012FEF0
Address Hex dump
0012FEF0 58 1C EA 73 65 03 BD 05 B0 FF 12 00
```

Bây giờ tôi sẽ thực hiện lệnh Call và quan sát xem giá trị tại `edi` thay đổi thế nào :

```
0040150B . 57          push edi
0040150C . E8 33030000 call <jmp.&MFC42.#3874>
00401511 . 8B07      mov eax,dword ptr ds:[edi]
00401513 . 8038 36   cmp byte ptr ds:[eax],36
00401516 . 75 1E      jnz short CoSH_Cra.00401536
00401518 . 8078 01 32 cmp byte ptr ds:[eax+1],32
0040151C . 75 18      jnz short CoSH_Cra.00401536
0040151E . 8078 02 38 cmp byte ptr ds:[eax+2],38
00401522 . 75 12      jnz short CoSH_Cra.00401536
Stack ds:[0012FEF0]=00354068 (ASCII "123456")
eax=00000006
Address Hex dump ASCII
0012FEF0 68 40 35 00 65 03 BD 05 B0 FF 12 00 C1 1A 40 00 h05.e040"y0.Á00.
```

Wow quá rõ ràng như ban ngày, khả khả giải thích thêm nữa chỉ là thừa mà thôi. Em Ollydbg đã "show" hàng cho chúng ta thấy hết rồi đó.

Okie chúng ta trở lại với IDA, như trên tôi và bạn đã biết edi sẽ chứa địa chỉ của Buffer, mà Buffer là nơi lưu trữ chuỗi Serial của chúng ta. Câu lệnh tiếp theo mà các bạn thấy đó là : `mov eax, [edi]`, có nghĩa là địa chỉ của Buffer sẽ được đưa vào thanh ghi eax, tức là giá trị tại eax bây giờ sẽ là chuỗi Serial. Và bên dưới là quá trình xử lý, tính toán trên chuỗi Serial này. Nếu quá trình xử lý chuỗi Serial mà không thỏa mãn với sự sắp đặt của Coder, ngay lập tức chúng ta sẽ bị “đó ván” ngay lập tức với dòng thông báo mà các bạn nhận được khi nhập sai.


```

text:00401511 8B 07          mov     eax, [edi]
text:00401513 80 38 36      cmp     byte ptr [eax], 36h ; Compare Two Operands
text:00401516 75 1E          jnz    short loc_401536 ; Jump if Not Zero (ZF=0)
text:00401516
text:00401518 80 78 01 32   cmp     byte ptr [eax+1], 32h ; Compare Two Operands
text:0040151C 75 18          jnz    short loc_401536 ; Jump if Not Zero (ZF=0)
text:0040151C
text:0040151E 80 78 02 38   cmp     byte ptr [eax+2], 38h ; Compare Two Operands
text:00401522 75 12          jnz    short loc_401536 ; Jump if Not Zero (ZF=0)
text:00401522
text:00401524 80 78 03 37   cmp     byte ptr [eax+3], 37h ; Compare Two Operands
text:00401528 75 0C          jnz    short loc_401536 ; Jump if Not Zero (ZF=0)
text:00401528
text:0040152A 80 78 04 2D   cmp     byte ptr [eax+4], 2Dh ; Compare Two Operands
text:0040152E 75 06          jnz    short loc_401536 ; Jump if Not Zero (ZF=0)
text:0040152E
text:00401530 80 78 05 41   cmp     byte ptr [eax+5], 41h ; Compare Two Operands
text:00401534 74 17          jz     short loc_40154D ; Jump if Zero (ZF=1)
    
```

Nhìn vào đây, chúng ta thấy được rằng chương trình này sẽ lấy chuỗi Serial ta nhập vào đem so sánh với một Constant Serial mặc định của chương trình, chuỗi Constant Serial này tôi có thể đoán được ra ngay lập tức đó là : **6287-A (36h = 6, 32h = 2 v..v..)**. Đến đây có thể nói là chúng ta đã tìm được chìa khóa để mở cửa, tuy nhiên tôi muốn giới thiệu thêm một số tính năng khác nữa trong IDA. Đó chính là tính năng Comment (đặt chú thích cho đoạn code), tính năng này rất quan trọng và không thể thiếu cho những ai nghiên cứu Reversing, nó cũng như khi chúng ta code một chương trình nếu chúng ta không chú thích hàm hay thụ tục này có nhiệm vụ gì, câu lệnh kia có ý nghĩa ra sao thì tôi tin chắc rằng sau này khi bạn đọc lại nó, bạn sẽ lại mất công tìm hiểu lại từ đầu mà như thế thì quá mất thời gian. Cho nên chúng ta thà mất công một chút nhưng lại đem lại hiệu quả lâu dài về sau. Một lời khuyên: **"Đừng làm cái gì quá vội vàng, cũng đừng làm quá cầu thả, hãy từ từ từng chút một, tuy mất thời gian nhưng sẽ giúp ích cho ta rất nhiều"**. Okie để Comment cho đoạn code trong chương trình, rất đơn giản bạn chỉ cần nhấn chuột phải vào đoạn code đó và chọn như sau :

```

-----
.text:00401513 80 38 36      cmp     byte ptr [eax], 36h ; Compare Two Operands
.text:00401516 75 1E          jnz    short loc_401536 ;
.text:00401516
    
```



Một hộp thoại xuất hiện cho phép bạn tiến hành nhập chú thích của bạn, bạn phải chú thích làm sao cho câu văn có đầy đủ ý nghĩa nhất đồng thời làm sao khi người khác xem người đó cũng có thể hiểu được. Đây cũng là một nghệ thuật đó, cuộc sống là phải biết chia sẻ những gì mình biết để giúp người khác. Sau khi chú thích tôi có được như sau :

```

.text:00401511 8B 07          mov     eax, [edi]
.text:00401513 80 38 36      cmp     byte ptr [eax], 36h ; Serial[0] == 6?
.text:00401516 75 1E          jnz    short loc_401536 ; If not go to Bad Message
    
```

Ngoài ra IDA còn cung cấp cho chúng ta khả năng biểu diễn dữ liệu theo những cách khác nhau. Như tôi nói “6” chính là “36h”, tại sao lại thế vì ta biết rằng trong bảng mã ACSII : “36h” chính là “6”. Để IDA thể hiện đúng như chúng ta mong muốn, bạn hãy chọn “36h” nhấn chuột phải và chọn như sau :


```

cmp     byte ptr [eax], 36h ; Serial[0] == 6?
jnz     short loc_401536 ; Bad Message

cmp     byte ptr [eax+1], 54h ; Serial[1] == 'S'
jnz     short loc_401536 ; If not go to Bad Message (ZF=0)

cmp     byte ptr [eax+2], 28h ; Serial[2] == '8'
jnz     short loc_401536 ; If not go to Bad Message (ZF=0)

cmp     byte ptr [eax+3], 110110b ; Serial[3] == '10110'
jnz     short loc_401536 ; If not go to Bad Message (ZF=0)
    
```

Kết quả là chúng ta có được như hình dưới đây :

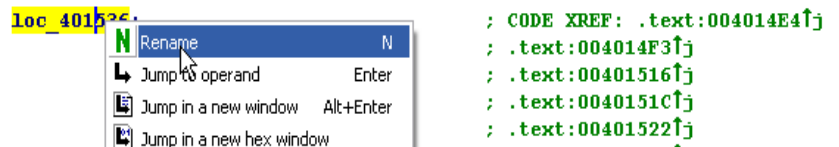
```

.text:00401511 8B 07      mov     eax, [edi]
.text:00401513 80 38 36   cmp     byte ptr [eax], '6' ; Serial[0] == 6?
.text:00401516 75 1E     jnz     short loc_401536 ; If not go to Bad Message
.text:00401516
    
```

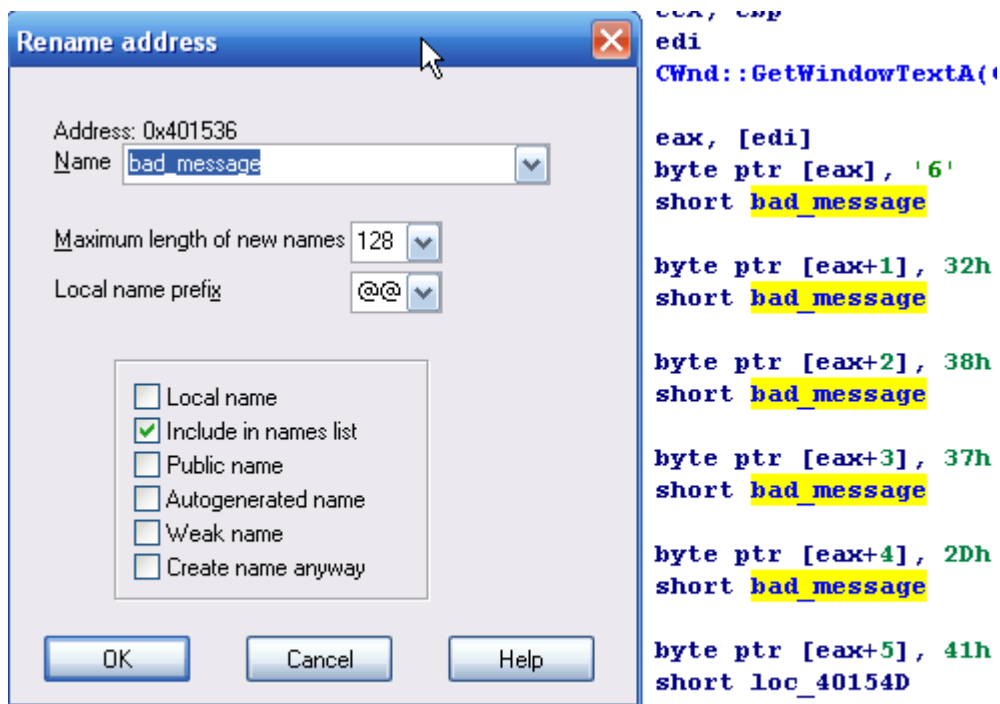
Th4t's gr34t !! © Ngoài ra bạn còn thấy nhiều cách chuyển đổi khác mà IDA cung cấp cho chúng ta. Tiếp theo chúng ta sẽ tìm hiểu một tính năng tuyệt vời khác của IDA đó là : **renaming references**. Như các bạn và tôi đã thấy trên hình minh họa ở trước, khi quá trình xử lý chuỗi Serial vi phạm bất kì điều kiện gì thì chúng ta đều bị đưa tới `loc_401536` thông qua các lệnh nhảy. Nhưng nhìn vào label `loc_401536` chúng ta thấy rằng nó không đem lại cho ta một ý nghĩa gì cả. Mặc dù trong quá trình RCE ta biết rằng đó là nơi mà sẽ bắn ra cái thông báo lỗi. Cho nên để biến cái chuỗi này thành một chuỗi có tính gợi nhớ và đầy đủ ý nghĩa thì IDA cho phép ta đổi tên của nó. Để thực hiện điều này chúng ta làm như hình minh họa dưới đây :

```

.text:00401536
.text:00401536
.text:00401536
.text:00401536
.text:00401536
    
```



Một hộp thoại xuất hiện, bạn chỉ cần đặt lại tên trong textbox Name, kết quả sẽ có được như sau :



bad_message:

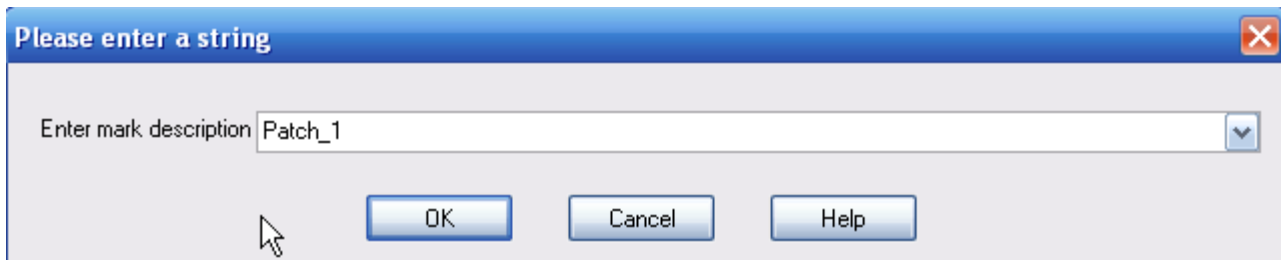
Một tính năng khác nữa trong IDA tương tự tính năng Bookmark trong Olly đó là : **Mark position**. Mục đích của Bookmark là để khi chúng ta đi quá xa đoạn code mà chúng ta đang làm việc thì việc

cuộn chuột để tìm kiếm dẫn đến mất thời gian, chi bằng ta đánh dấu nó lại thì khi đó có đi đâu chẳng nữa bạn vẫn biết được chỗ để mà quay về. Để đánh dấu một Bookmark các bạn làm như sau, chuột phải tại vị trí cần đánh dấu và chọn :

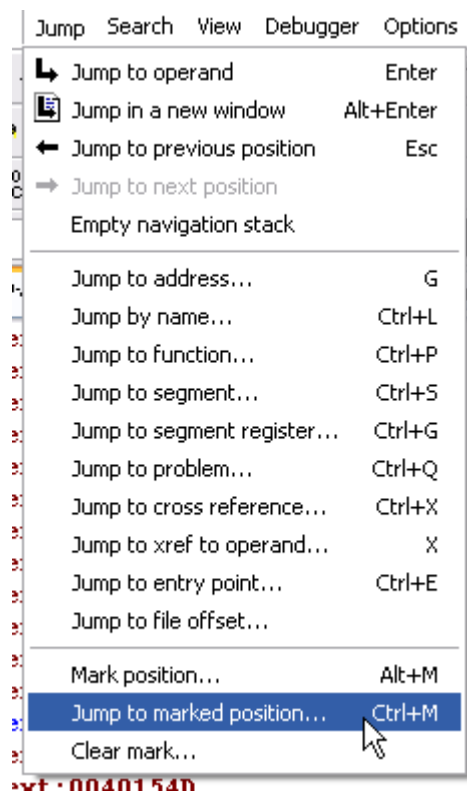
```

.text:00401513 80 38 36          cmp     byte ptr [eax], '6'      ; Serial[0] == 6?
.text:00401514 75 17          jnz     short bad_message      ; If not go to Bad Message
.text:00401515          Mark position... Alt+M
.text:00401516          cmp     byte ptr [eax+1], 32h   ; Compare Two Operands
.text:00401517          jnz     short bad_message      ; Jump if Not Zero (ZF=0)
    
```

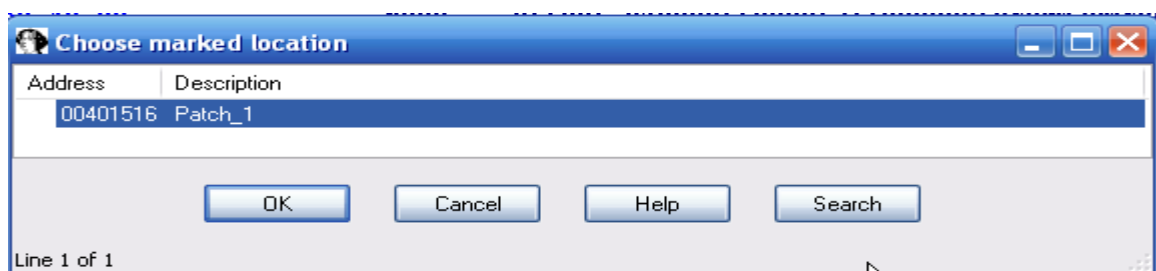
Một hộp thoại xuất hiện, ta đặt tên cho Bookmark giả sử tôi đặt như sau :



Để tìm lại các Bookmark hay nói cách khác để quay về vị trí đánh dấu thì trên menu **Jump** chúng ta chọn như sau :



Ngay lập tức ta sẽ nhận được một danh sách các BookMark, việc còn lại là tùy vào bạn muốn về Bookmark nào :



IV. Lời kết

Bài viết thứ 2 của tôi về IDA đến đây là kết thúc, những tính năng về IDA còn quá nhiều mình tôi không đủ sức kham nổi, hi vọng anh em trong quá trình làm việc với IDA có những thủ thuật, mẹo vặt hay thì chia sẻ cho mọi người cùng biết. Hi vọng bài viết này sẽ cung cấp thêm cho anh em những điều lý thú nữa về IDA. Rất cảm ơn anh em đã dành thời gian đọc nó.

*Nam nhi sống chấp nhận đôi lần bại danh
Ngại ngần chi ngã đôi lần lại đứng lên*

PS : Hi vọng anh TQN, Thug và light.phoenix có thời gian viết vài bài cho anh em mở rộng tầm mắt ☺

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, dqtl, ARTEAM all my friend, and YOU.

--++---==[Special Thanks To]===--+ +--

- coruso_trac, pat, trm_tr. Thug4lif3, vn_blackrain, v..v.. and all brothers in VSEC

--++---==[Thanks To]===--+ +--

iamidiot, WhyNotBar, trickyboy, dzungltn, takada, hurt_heart, haule_nth, hytkl v..v.. các bạn đã đóng góp rất nhiều cho REA. Hi vọng các bạn sẽ tiếp tục phát huy ☺

>>>> If you have any suggestions, comments or corrections email me:

[kienbigmummy\[at\]gmail.com](mailto:kienbigmummy[at]gmail.com)