# INTRODUCTION TO THE CRACKING WITH OLLYDBG

# FROM CRACKLATINOS

**(_kienmanowar_)**

*Một cái đầu lạnh để vững vàng, một trái tim đỏ lửa để yêu và làm việc hết mình!*

## I. Giới thiệu chung

Chào các bạn, các bạn còn nhớ loạt tuts này chứ….Kể từ lúc hoàn thành xong phần 9 tôi khá bận với dự án cho nên chưa tập trung vào để dịch và viết tiếp các phần tiếp theo được. Lý do một phần cũng là bởi tôi ngại đọc và trans từ TBN của English quá, vì khi trans chỉ là word by word cho nên lắm khi phải nhìn vào cái tut bằng tiếng TBN để hiểu xem ý nó là gì và mình viết lại theo cách của mình, thêm nữa tôi cũng lười viết lách ☺. Trong phần 9 các bạn đã được làm quen với các phương pháp cơ bản để tìm một hàm API, cách đặt BP, cách edit các cờ và cuối cùng là patch chương trình bằng cách edit code. Ở phần 10 này sẽ đi sâu hơn một chút về quá trình làm việc với BreakPoint trong OllyDbg, việc hiểu rõ quá trình cũng như cách thức đặt BreakPoints sẽ giúp đỡ chúng ta rất nhiều trong việc dừng sự thực thi của một chương trình tại nơi mà chúng ta mong muốn. Thêm vào đó chúng ta cũng tìm hiểu thêm về một số cách đặt Breakpoint, target trong phần này vẫn là Crackme mà các bạn làm việc từ đầu đến giờ,  điều thú vị đang nằm ở phía trước…. N0w….L3t's G0!!!!!!!!!

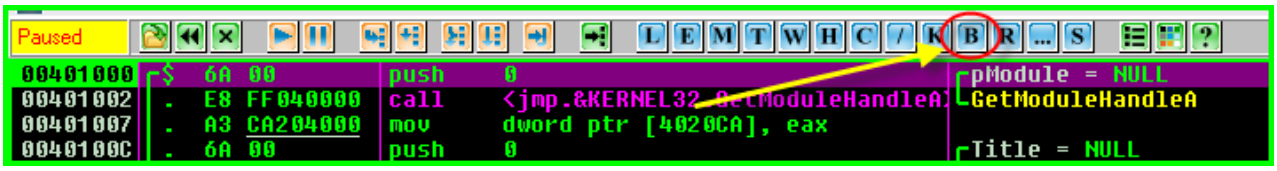
## II. BreakPoints in OllyDbg

***1.Common Breakpoint or BPX :***

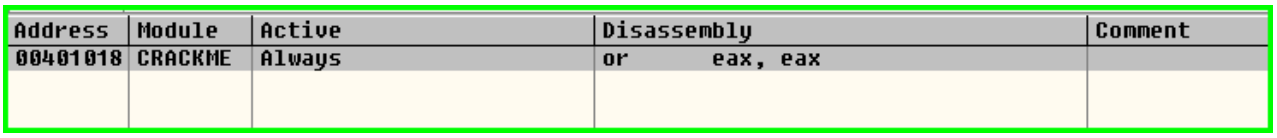
Đây có thể nói là phương pháp đặt breakpoint phổ biến mà chúng ta đã làm việc ngay từ các phần trước của bộ tuts này, nó còn được gọi là BPX bởi vì nó có nhiều điểm tương đồng với lệnh BPX của SoftIce (ai đã từng dùng qua SoftIce thì chắc là biết còn tôi chỉ đụng SoftIce đúng một lần duy nhất ☺), thay vì phải viết dài dòng là ***BreakPoint*** thì để tránh rườm rà trong quá trình làm việc thông qua Command Line, tác giả đã rút gọn lại là **BPX**, tuy nhiên trong Olly chữ **"X"** thường được sử dụng trong việc đặt BP với các hàm API (liên quan tới version của Windows) cho nên tác giả của Olly đã rút gọn lại và thêm một command khác mà ta hay dùng đó là **BP** (vừa có thể đặt BP tại API và address). Thông thường, đối với một câu lệnh nào đó mà chúng ta muốn đặt BPX lên nó thì đơn giản nhất ta chỉ việc chọn dòng lệnh mà ta muốn và nhấn F2, nếu nhấn F2 một lần nữa tức là ta remove BP khỏi câu lệnh.

Giải thích như trên có thể khiến các bạn bị rối, tóm lại nếu muốn đặt BP tại một dòng lệnh bất kì các bạn chỉ cần nhớ là bấm **F2** tại dòng lệnh đó. Giờ ta load Crackme vào Olly, chúng ta sẽ dừng lại tại Entry Point của crackme :

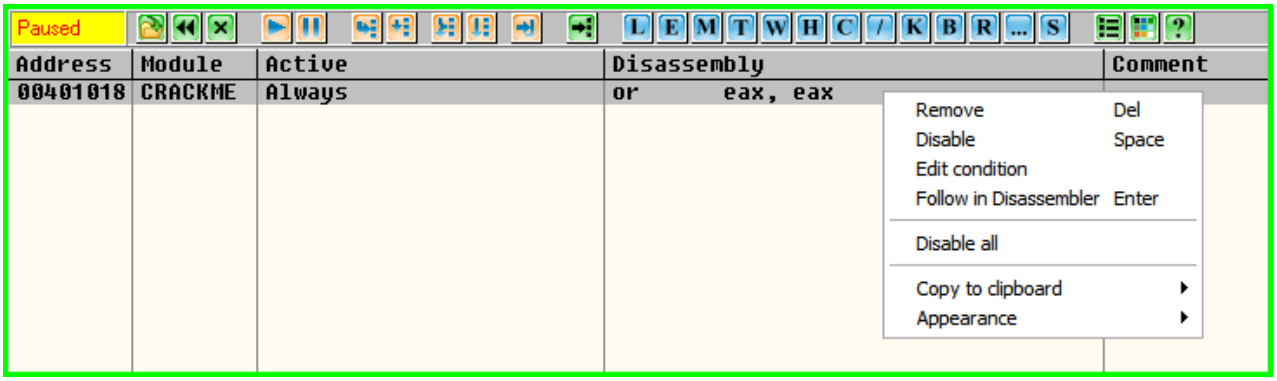

Các bạn có thể thấy trên hình minh họa, tại địa chỉ `00401018` khi tôi bấm **F2** thì ngay lập tức địa chỉ này được tô sáng lên bằng màu vàng (của các bạn có thể là màu đỏ hay bất kì màu nào do bạn cấu hình Olly), điều này có nghĩa là tôi đã đặt BP tại `00401018 |. 0BC0 or eax, eax` . Ngay lúc này, bạn chuyển qua cửa sổ quản lý các Breakpoints bằng cách nhấn chữ **B** (hoặc vào Menu: **Windows -> BreakPoints**) :




Các bạn sẽ thấy BP mà chúng ta vừa đặt đã được liệt kê tại cửa sổ này, ta tìm hiểu qua về cửa sổ quản lý BreakPoints này một chút :

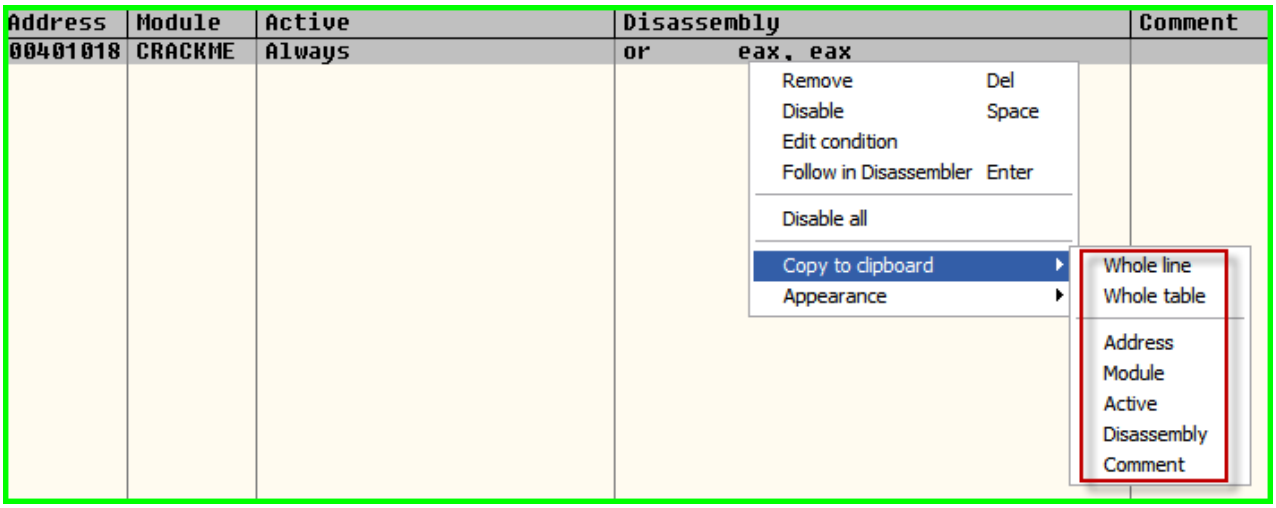



Chúng ta quan sát thấy tại cột Active thì trạng thái của BP ta đặt luôn là Always điều này có nghĩa là BP này sẽ luôn được thực thi hay nói cách khác là nó sẽ được kích hoạt khi ta thực thi chương trình. Khi ta chọn và nhấn chuột phải lên dòng lệnh ta sẽ nhận được một menu ngữ cảnh như sau :




- **Remove** : loại bỏ breakpoint ra khỏi danh sách các điểm đặt BP mà cửa sổ này quản lý.
- **Disable** : tạm thời disable bp của chúng ta mà không loại nó khỏi danh sách, khi cần ta lại active nó lên.
- **Edit Condition** : khi chọn chức năng này là ta đang muốn chuyển đổi bp của chúng ta sang một dạng khác là conditional bp, kiểu đặt bp này sẽ được bàn tới ở dưới.
- **Follow in Disassembler** : Với một danh sách dài các bp thì chúng ta khó có thể nhớ được nó liên quan tới đoạn code nào, tính năng này cho phép chúng ta tìm tới điểm mà chúng ta đặt bp tại cửa sổ code.
- **Disable all** : tương tự như tính năng Disable tuy nhiên có khác là nó sẽ disable hết toàn bộ các bp có trong cửa sổ Breakpoints.

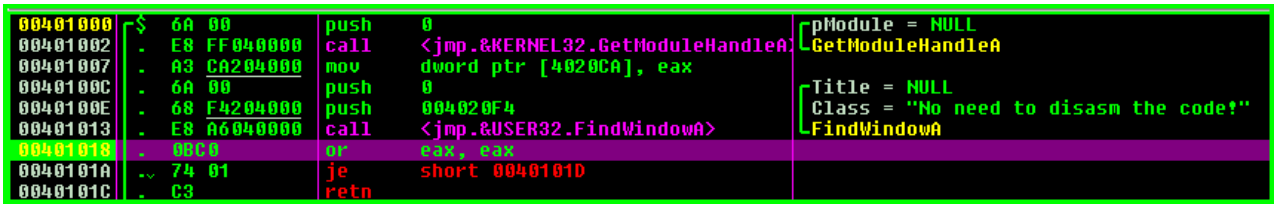
- **Copy to clipboard** : sao chép thông tin về bp vào clipboard. Khi chọn tính năng này sẽ xuất hiện thêm một số chức năng con đi kèm.




Ta thử chọn chức năng **Copy to Clipboard -> Whole line**, tính năng này sẽ copy nguyên dòng mà chúng ta chỉ định. Còn Whole table sẽ copy toàn bộ tất cả những danh sách bp hiện có trong cửa sổ Breakpoints. Đây là kết quả khi ta chọn Whole line :

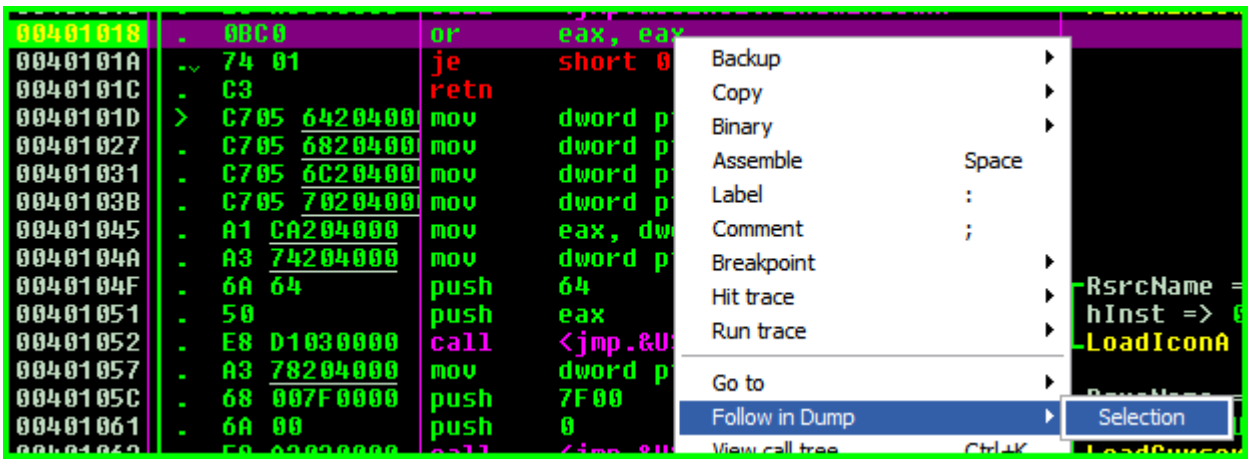
```
Br3akp0ints, item 0
 Address=00401018
 Module=CRACKME
 Active=Always
 Disassembly=or      eax, eax
```

OK, như trên chúng ta đã đặt một bp tại `0x00401018`, bây giờ chúng ta cần kiểm tra xem bp mà chúng ta đặt có hoạt động đúng như chúng ta mong muốn hay không? Nhấn **F9** để thực thi chương trình, ngay lập tức các bạn sẽ thấy ta đã dừng lại tại nơi mà ta đặt bp :



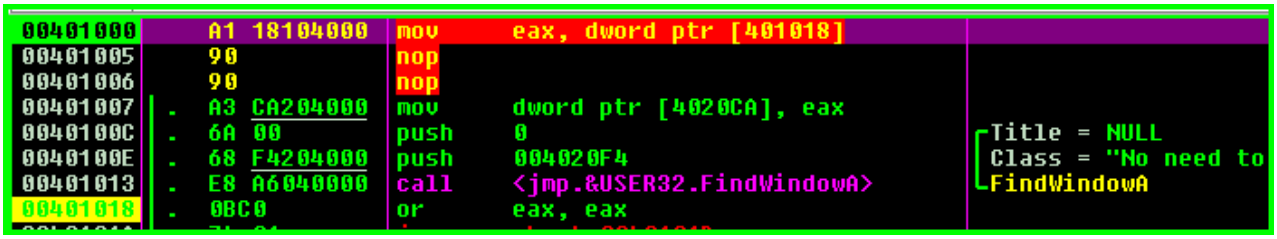

Dòm qua thanh status ta thấy như sau :


Bây giờ chúng ta sẽ tìm hiểu xem điều gì đã thực sự xảy ra? Có sự thay đổi nào không khi ta thiết lập BP tại lệnh mà chúng ta muốn.Ok, chuột phải tại `0x00401018` và chọn :

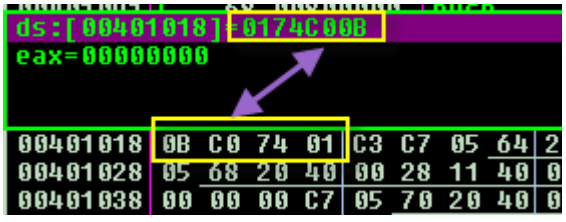



Quan sát nội dung tại cửa sổ Dump ta có như sau :

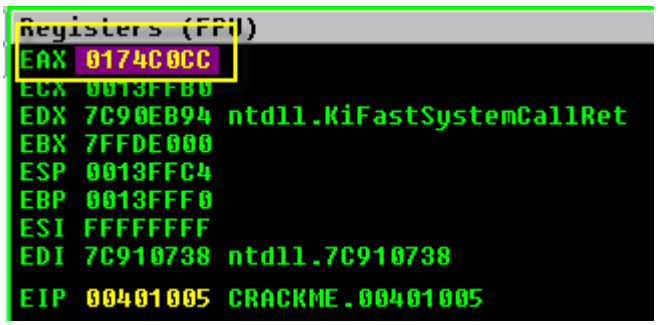

So sánh thông tin ở cửa sổ Dump với thông tin ở cửa sổ CPU ta thấy không có gì thay đổi cả, ở cửa sổ Dump các bạn thấy các bytes `0B C0` tương đương với câu lệnh `or eax, eax` tại cửa sổ CPU. Điều này khiến chúng ta không khỏi thắc mắc, nếu như không có sự thay đổi gì thì tại sao khi tôi run chương trình thì nó lại dừng lại tại điểm mà tôi đặt BP? Để kiểm chứng xem có sự thay đổi hay không, các bạn hãy restart lại Olly (**Ctrl+F2**) và chắc chắn rằng chúng ta vẫn đang đặt BP tại địa chỉ `0x00401018` . Sau khi Restart lại Olly, ta sẽ dừng lại tại EP của chương trình, tiến hành edit một đoạn code như sau tại EP :




Mục đích của đoạn code này chỉ đơn giản là đọc ra nội dung tại địa chỉ `0x401018` và lưu vào thanh ghi eax. Để dễ hiểu hơn bạn nhìn vào cửa sổ Tip :



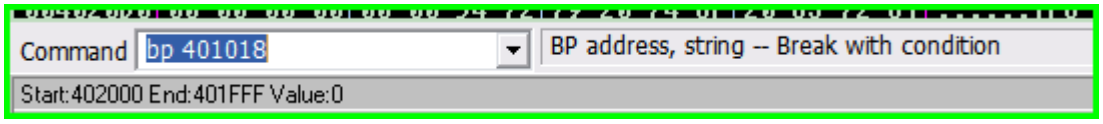

Nhìn vào cửa sổ Dump ta sẽ biết được đây là những bytes gốc ban đầu, chưa bị thay đổi gì.Nhưng tôi lấy làm ngờ vực ☺, tôi thử nhấn **F7** để trace và kết quả là thanh ghi EAX đã có sự thay đổi rất thú vị :



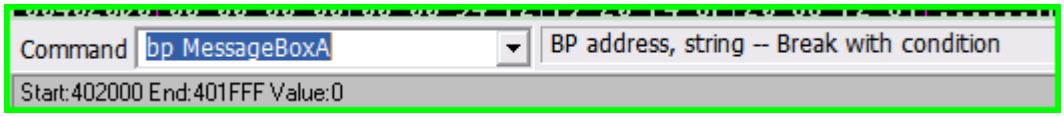

Để ý cửa sổ Dump bạn thấy các bytes vẫn được giữ nguyên như ban đầu, nhưng ở thanh ghi EAX thì ta lại thấy giá trị đã khác là : `0x0174C0CC`. Điều này được giải thích sơ bộ như sau: khi tôi đặt một BP tại địa chỉ `0x401018` thì Olly sẽ tiến hành thay thế byte đầu tiên mà ở đây là giá trị 0B bằng một giá trị khác là 0xCC. Nếu như bạn quy đổi byte CC này sang lệnh asm thì nó là `int3`, đây là một câu lệnh đặc biệt (nó còn được gọi là ***Trap to Debugger***) nó sẽ gây ra một exception khi chúng ta cố gắng thực thi chương trình. Các bạn đọc thêm thông tin sau :

*"So generally it is complex to set a breakpoint in an arbitrary place of the program. The debugger should save the current value of the memory location at the specified address, then write the code 0xCC there. Before exiting the debug interrupt, the debugger should return everything to its former place, and should modify IP saved in the stack so that it points to the beginning of the restored instruction. (Otherwise, it points to its middle.)"*

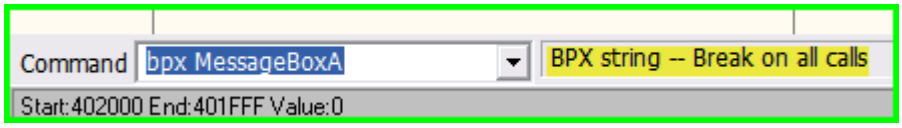
Hơi lằng nhằng một chút nhưng hi vọng các bạn cũng thấm vào đầu được ít nhiều ☺, ngoài việc đặt BP thông qua việc chọn câu lệnh và nhấn **F2** thì ta còn một cách đặt BP khác nữa như sau, tìm địa chỉ cần đặt BP và gõ lệnh tại Command Bar plugin :



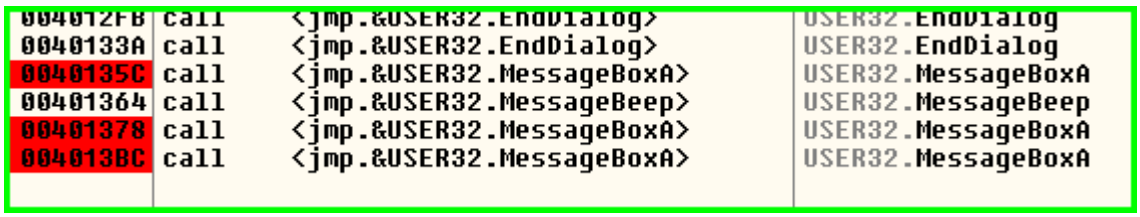

Giờ ta tìm hiểu một chút về cách đặt BP cho hàm API. Như ở trên tôi đã nói lệnh BPX thường được sử dụng cho việc đặt BP tại các hàm API của chương trình, thêm nữa nó lại phụ thuộc vào phiên bản Windows mà bạn đang sử dụng. Đối với những ai sử dụng Windows NT/2000/XP/2003 thì việc đặt BP tại một API function cụ thể rất đơn giản, chỉ việc gõ BP [tên của hàm] trong Command Bar như sau :



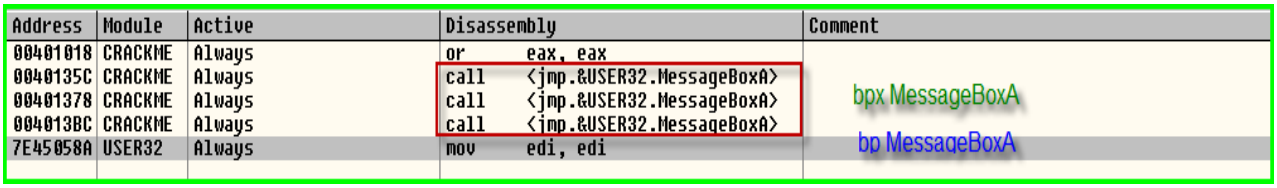

Bạn nhớ gõ đúng chữ hoa chữ thường trong tên hàm nhé!! Trên Windows 98 thì ngược lại, việc đặt BP tại 1 hàm API cụ thể không được hỗ trợ mà thay vào đó là việc đặt BP tới những nơi có lời gọi tới hàm API mà chúng ta cần. Ví dụ :



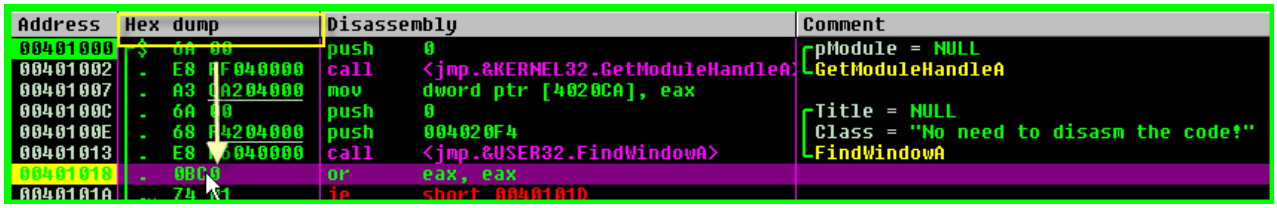

Kết quả ta có được như sau :




Vậy là ta đã thấy được sự khác biệt giữa BPX và BP, BPX không thiết lập một breakpoint tại một địa chỉ cụ thể đã được chỉ định như BP đã làm mà nó chỉ đặt BP tại những câu lệnh tham chiếu tới địa chỉ đó mà thôi. Lý thuyết phải đi kèm thực tế, các bạn đặt liền một lúc cả BP và BPX tại API MessageBoxA và quan sát cửa sổ Breakpoints :




Ngoài việc hỗ trợ đặt BP thông qua phím F2 và thông qua command line, Olly còn hỗ trợ cho chúng ta việc đặt bp thông qua việc nhấn đúp chuột. Để thực hiện việc đặt bp tại một địa chỉ, bạn chỉ việc chọn cột Hex Dump tại cửa sổ CPU và nhấp đúp chuột lên đó.Nếu nhấp đúp chuột một lần nữa thì sẽ loại bỏ BP :



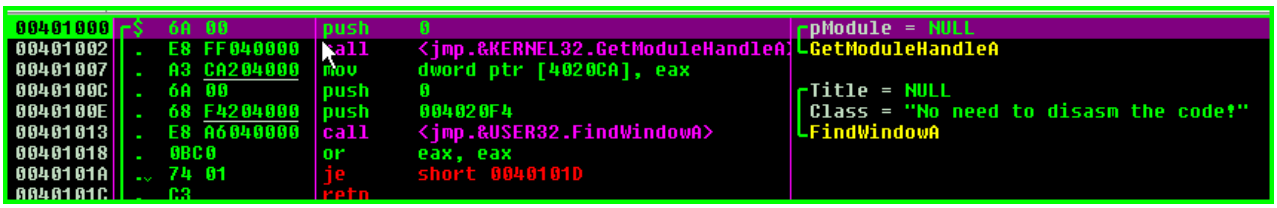

Tiếp theo ta sẽ tìm hiểu về Memory BreakPoint.

***2.Breakpoints on Memory (Memory Breakpoint) :***

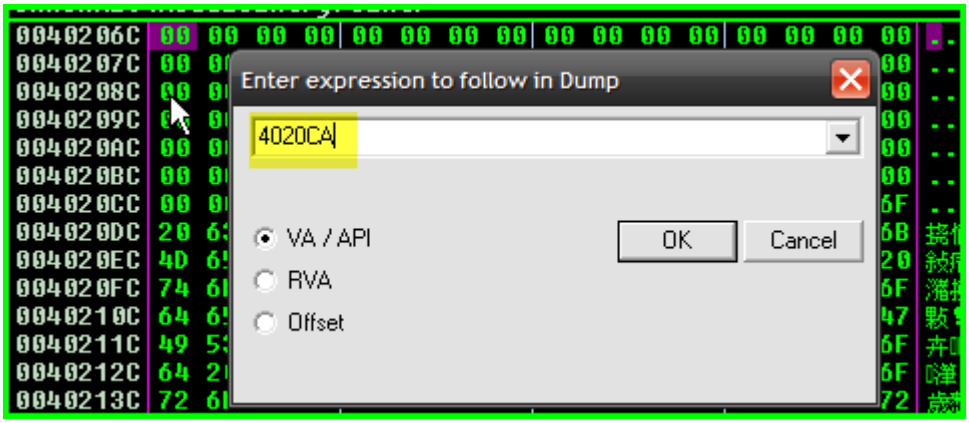
Ở phần trên các bạn đã tìm hiểu về việc đặt breakpoint thông qua BP và BPX, tuy nhiên việc đặt bp này là đối với những câu lệnh hay những opcodes không bị thay đổi trong suốt quá trình thực thi chương trình. Còn ở phần này chúng ta sẽ tìm hiểu về việc đặt bp trên memory(nơi mà dữ liệu thường xuyên thay đổi). Tại một thời điểm Olly chỉ cho phép duy nhất một memory breakpoint. Olly hỗ trợ cho chúng ta hai kiểu đặt bp trên memory là : Breakpoint Memory on access và Breakpoint memory on write.Vậy hai cách đặt BP này có gì khác nhau :

- **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.
- **BreakPoint Memory on write:** Khác một chút với kiểu đặt bp ở trên, kiểu này cho phép chúng ta dừng sự thực thi của chương trình khi có bất kì dữ liệu nào được ghi lên vùng nhớ mà ta đặt bp.

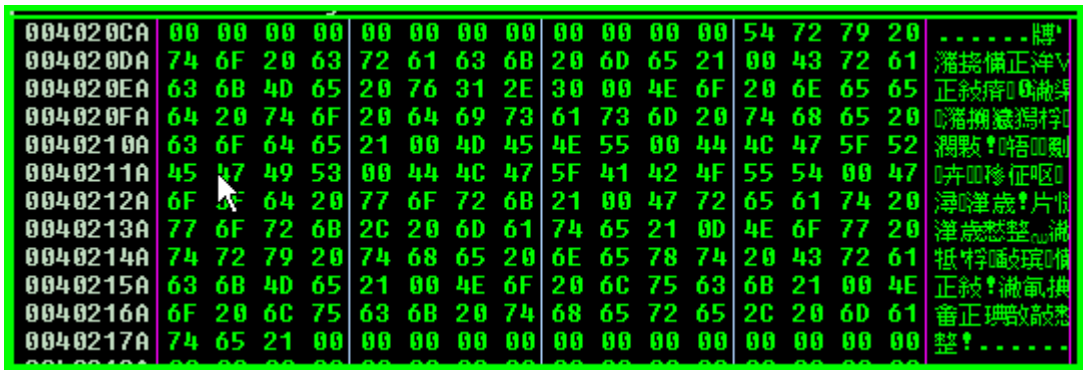
Lý thuyết là như thế ☺, chúng ta sẽ thực hành một chút để hiểu thêm nhé :
Chúng ta đã load crackme vào Olly và đang dừng lại tại EP của crackme.



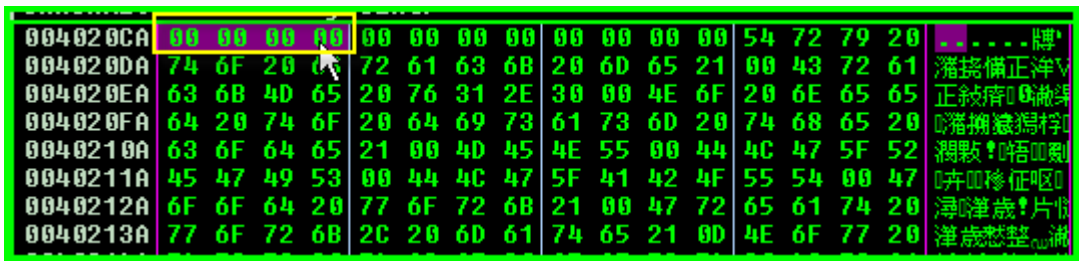

Chuyển xuống cửa sổ Dump và nhấn chuột phải chọn **Goto > Expression** hay nhấn phím tắt là **Ctrl+G** và gõ vào địa chỉ như sau :



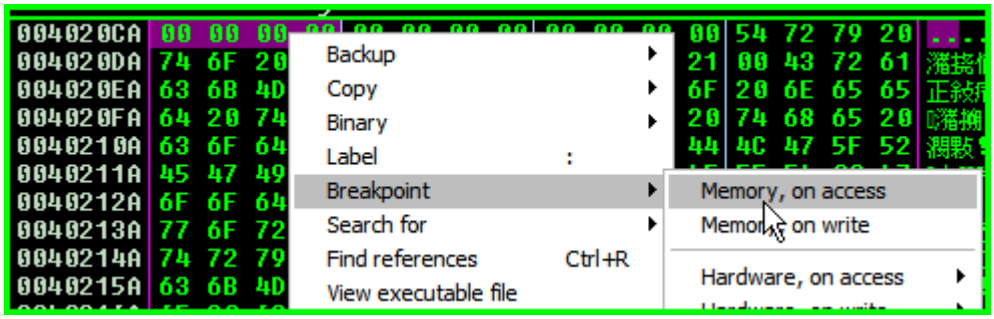

Olly sẽ đưa ta tới đây :



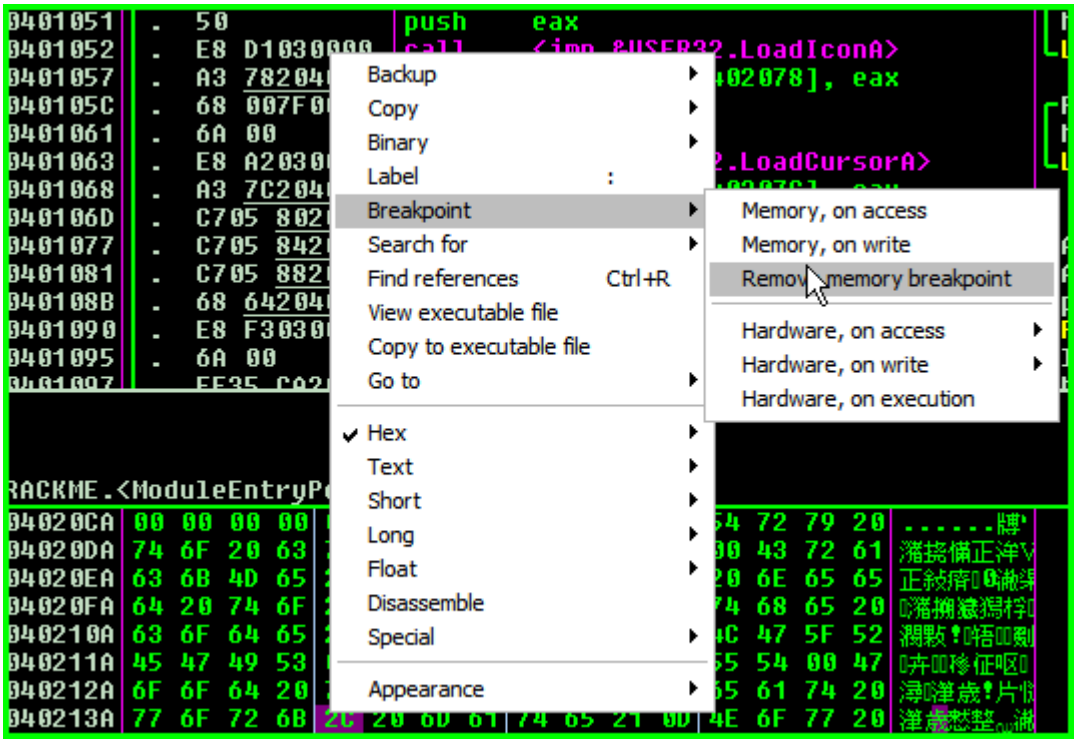

Ok chúng ta đã tới vùng nhớ để thực hành đặt bp, tôi sẽ tiến hành đặt một bp là Memory on access lên 4 bytes tại `4020CA`, sau đó tiến hành thực thi chương trình và hi vọng rằng chương trình sẽ dừng quá trình thực thi lại nếu như có một đoạn code nào đó đọc 4 bytes tại `4020CA`, hoặc ghi vào 4 bytes tại `4020CA` hay thực thi lệnh tại `4020CA`. Để đặt bp tại 4 bytes tôi làm như sau, dùng chuột bôi đen 4 bytes :



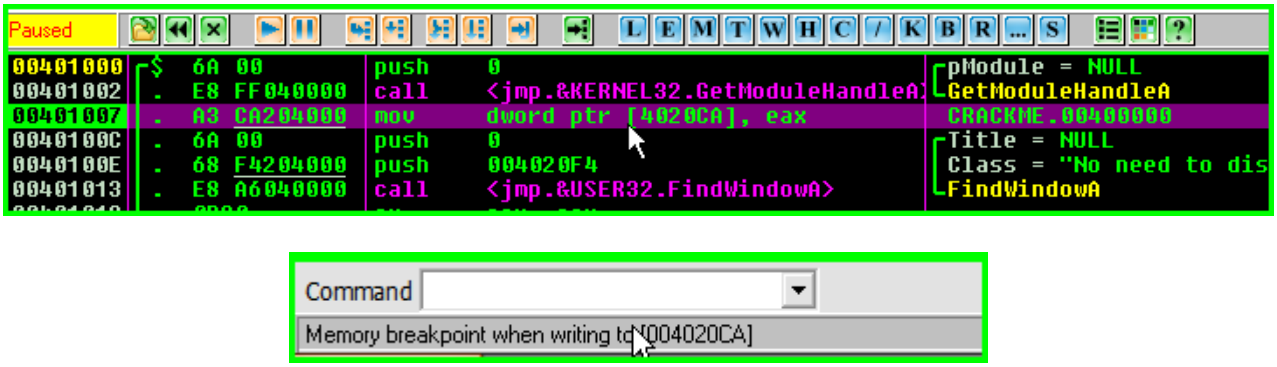

Tiếp theo nhấn chuột phải và chọn như hình :




Một điểm đáng lưu ý là không giống như việc đặt với lệnh BP hay BPX , ***việc đặt bp tại memory sẽ không được lưu giữ thông tin tại cửa sổ Breakpoint*** vì vậy ta phải tự ghi nhớ xem mình đã đặt ở đâu, nếu chúng ta muốn remove breakpoint đã đặt thì nhấn chuột phải và chọn :

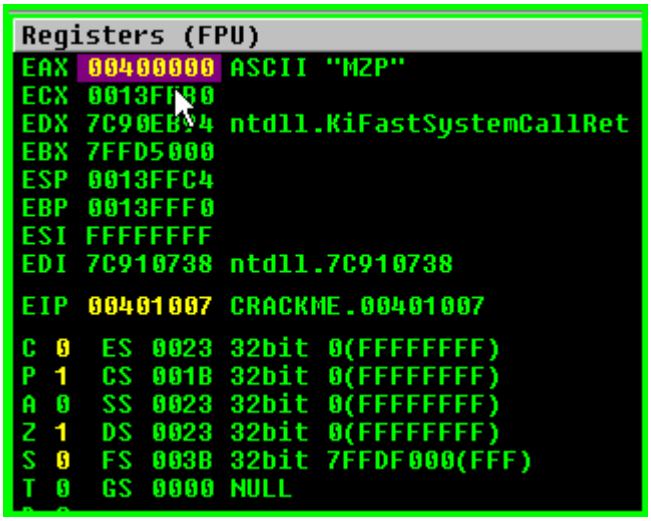



Như đã nói ở trên, Olly chỉ cho phép tại một thời điểm có một memory bp duy nhất, nếu bạn đặt một bp khác thì bp đã đặt trước đó sẽ tự động bị loại bỏ ☺. Ok, sau khi đã đặt bp theo hướng dẫn như trên, các bạn cho thực thi chương trình bằng cách nhấn F9,chương trình ngay sau đó sẽ dừng lại tại đoạn code bên dưới EP chút xíu :



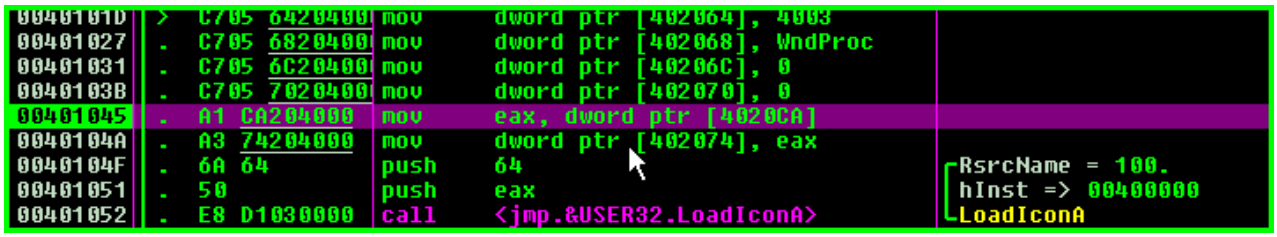

Thông tin ở Status Bar đã cho chúng ta biết được tại sao Olly lại break, đó là bởi vì câu lệnh tại địa chỉ `0x00401007` chuẩn bị ghi giá trị của thanh ghi eax vào nội dung của địa chỉ `4020CA`, việc này

sẽ gây ra một exceptions khiến cho dừng sự thực thi của chương trình.Giờ ta nhìn qua cửa sổ Register để xem giá trị của eax là gì :



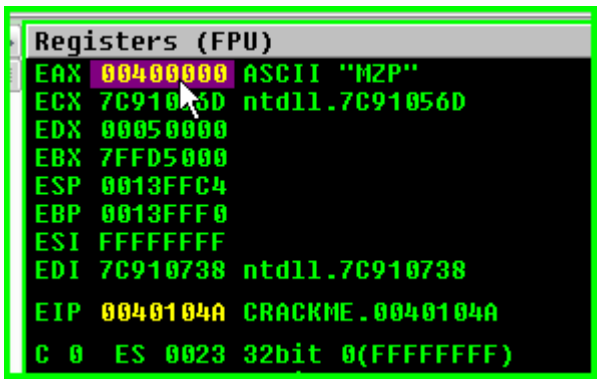

Ta nhấn F7 để trace qua đoạn code `mov dword ptr [4020CA], eax` và quan sát giá trị tại của sổ dump :


Như vậy là giá trị `0x00400000` được lưu giữ tại [`4020CA`], tiếp tục một lần nữa ta nhấn F9 để thực thi chương trình để xem chuyện gì sẽ xảy ra :

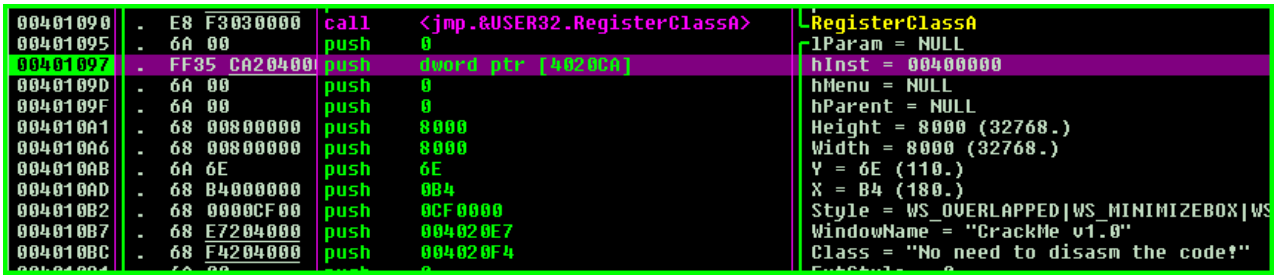


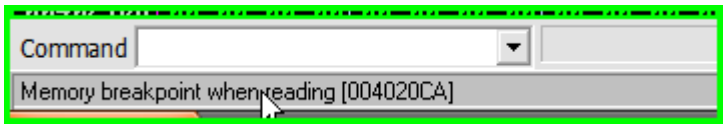


Chà, một lần nữa Olly lại break và lần này là tại `0x401045`, nhìn xuống thanh Status Bar ta thấy được lý do tại sao lại dừng sự thực thi của chương trình. Đó chính là do đoạn code tại `0x401045` đang cố gắng đọc nội dung tại `4020CA` và lưu vào thanh ghi eax. Nội dung tại `4020CA` là gì thì ta đã biết ở trên, giờ ta nhấn **F7** để trace và quan sát thanh ghi eax :

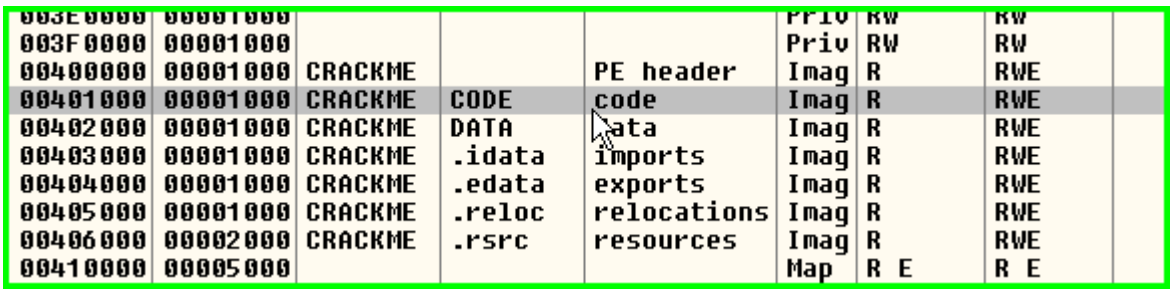



Giá trị `0x400000` lại được đưa trả về cho thanh ghi eax. Lại tiếp tục nhấn **F9** thêm lần nữa :

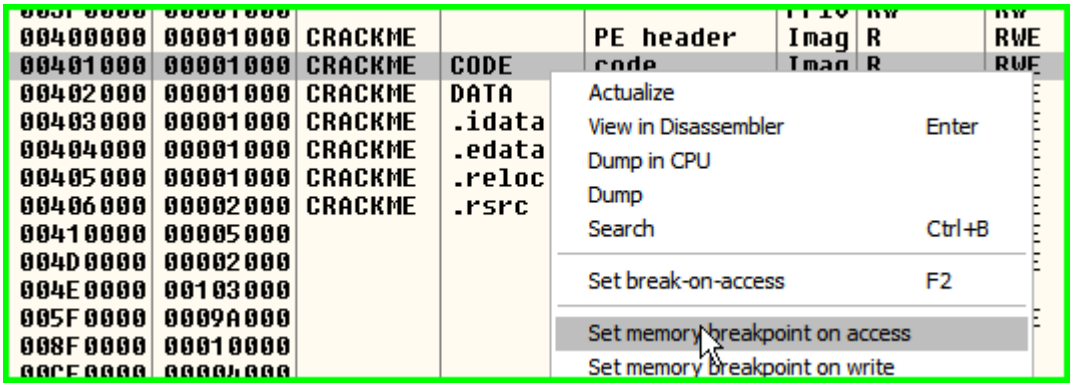


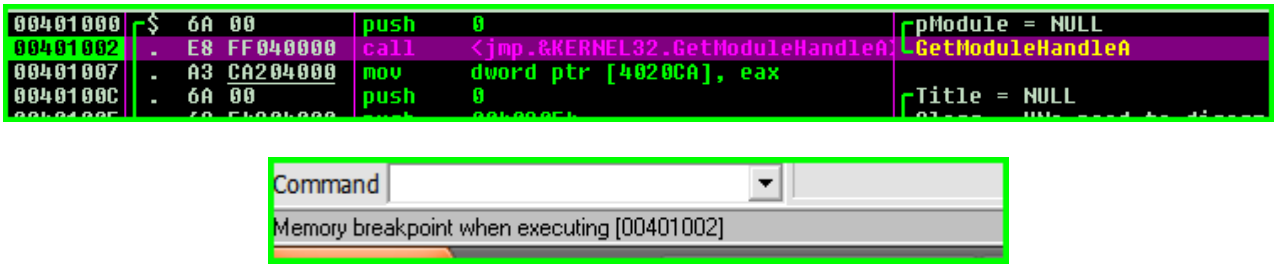

Như vậy là các bạn đã hiểu về Memory on Access và cách thức để có thể đặt một bp Memory on access rồi nhé. Việc đặt bp Memory on Write cũng tương tự như những gì tôi đã làm ở trên, các bạn tự thực hành để rút ra kết luận nhé.
Ngoài việc đặt bp như trên, Olly còn cho phép chúng ta đặt Memory bp lên các section của file. Để thực hiện được điều này các bạn mở cửa sổ **Window -> Memory map** hoặc nhấn phím tắt là **Alt + M** :

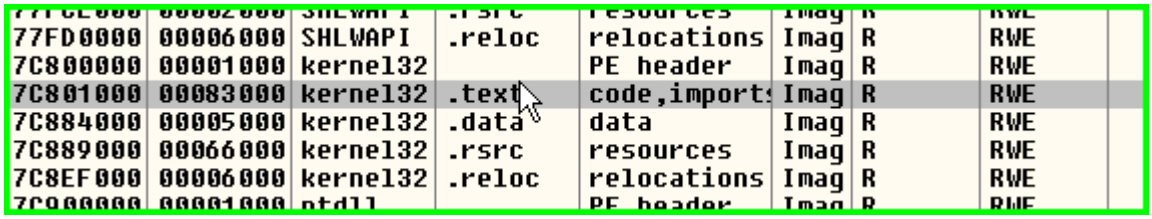

Ta chọn đại lấy một section để làm ví dụ, ở đây để đơn giản tôi chọn section CODE và đặt một Memory bp lên nó. Chỉ việc chọn section, nhấn chuột phải và chọn như hình dưới đây để thiết lập một bp :

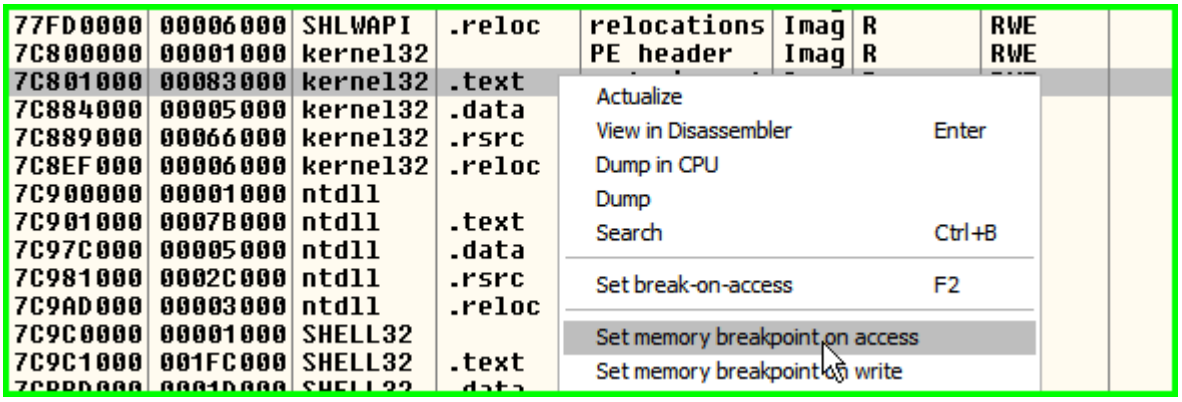



Sau khi đặt xong bp, nhấn F9 để thực thi chương trình :

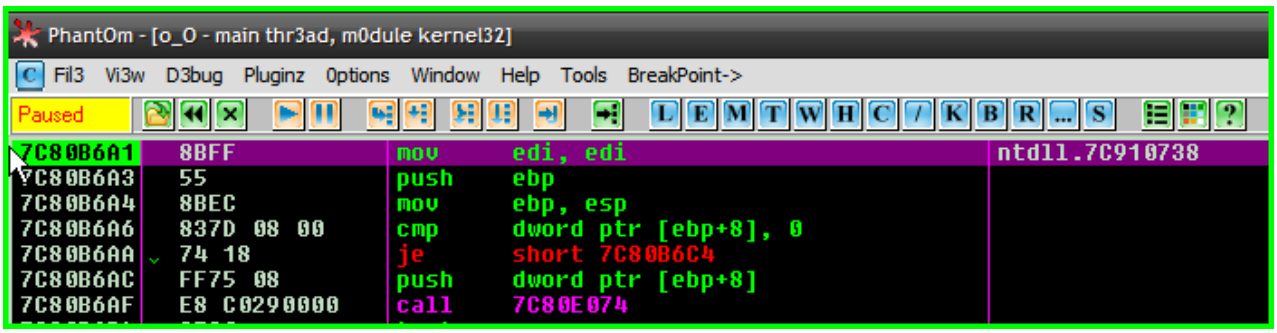


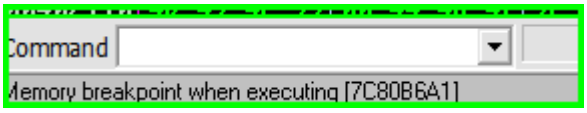

Việc quá trình thực thi bị dừng lại là hoàn toàn chính xác, đó là vì ta đang thực thi lệnh tại [401002]. Giờ ta thử đặt BP lên section của một module khác mà không phải là section của file exe mà ta đang làm việc, ở đây tôi chọn là kernel32 :



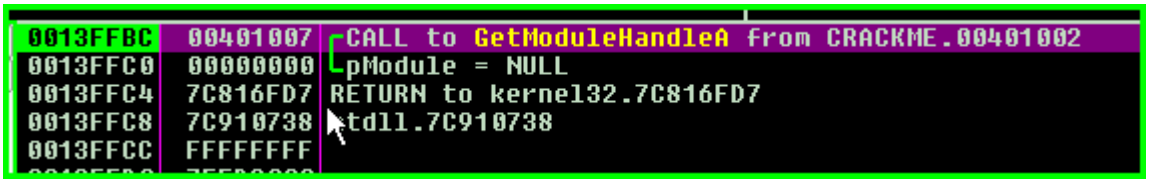




Nhấn **F9** để thực thi chương trình, ta không cần remove memory bp đã đặt ở trước đơn giản là vì khi ta đặt một memory bp khác thì bp trước sẽ bị remove luôn :

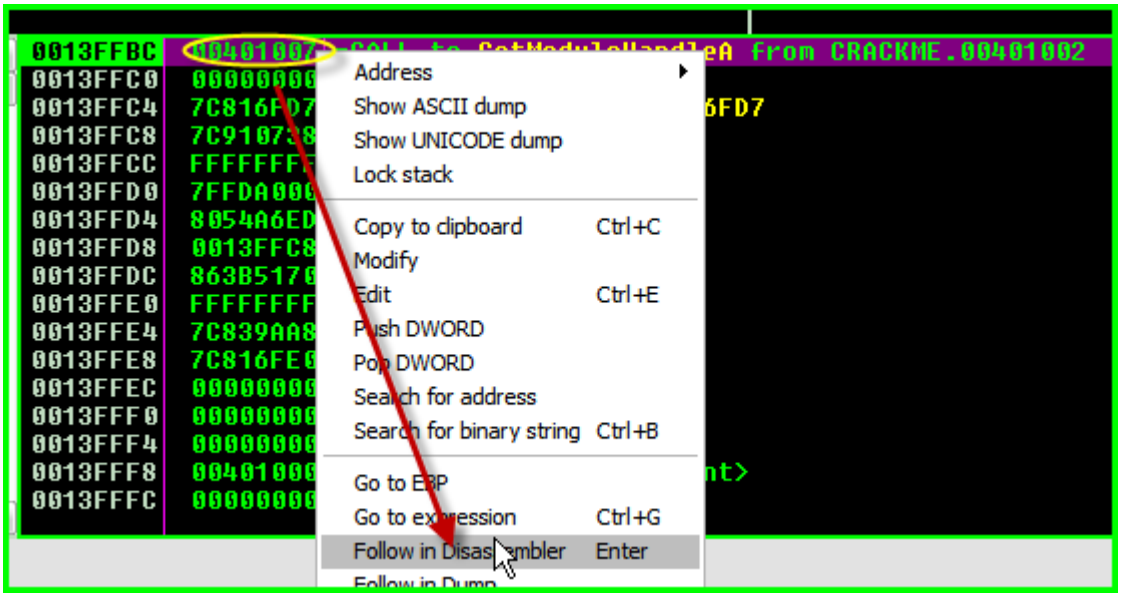


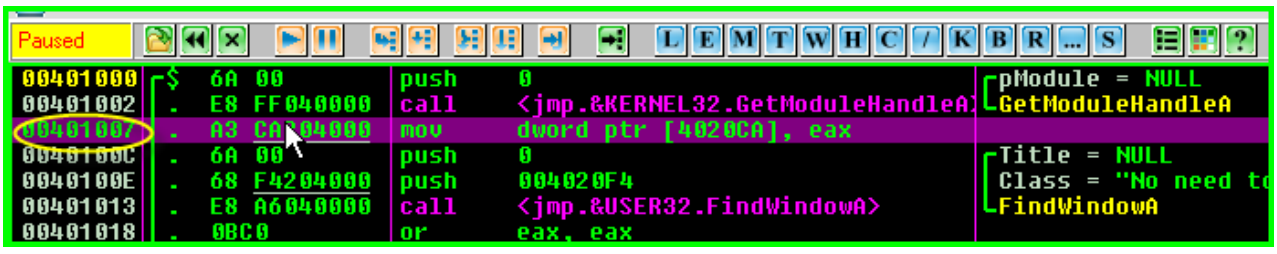




Nhìn sang cửa sổ Stack ta sẽ biết là ta đang dừng tại hàm API nào :



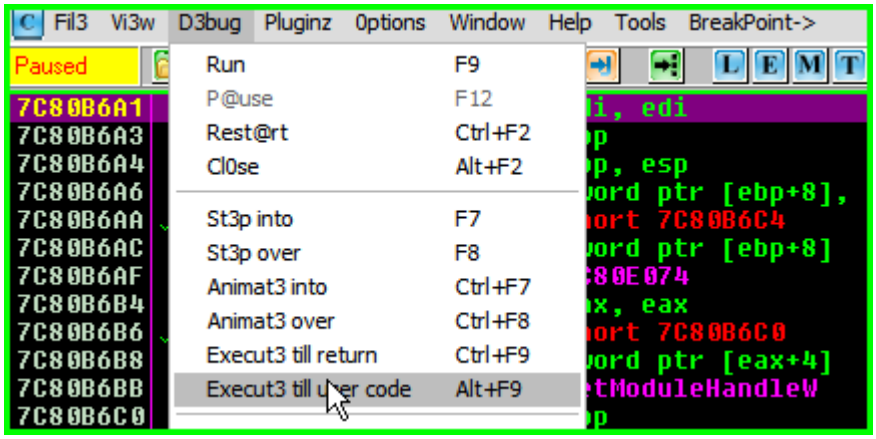

Như vậy là chương trình đã dừng lại khi thực thi lệnh của kernel32.dll, nhìn ở cửa sổ Stack ở hình trên và kết hợp với lý thuyết về Stack mà tôi đã giới thiệu qua ở phần trước thì có thể nhận thấy rằng địa chỉ 0x401007 là địa chỉ trở về của lệnh bên dưới lời gọi hàm **Call GetModuleHandleA** tại địa chỉ 0x401002. Thử nhấn chuột phải và chọn Follow in Disassembler để kiểm chứng lại :

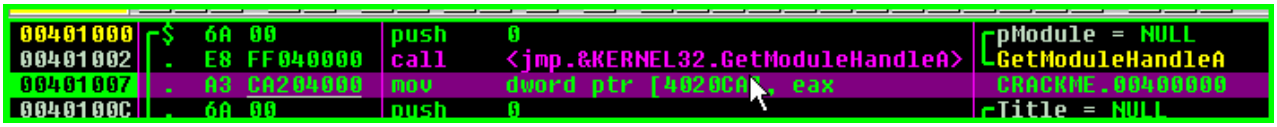



Đây là cách nhanh nhất để quay về địa chỉ của lệnh bên dưới, còn một cách khác cũng hay được áp dụng nữa là khi ta đang dừng tại hàm API như hình minh họa ở trên, ta nhấn chọn Menu **Debug** >

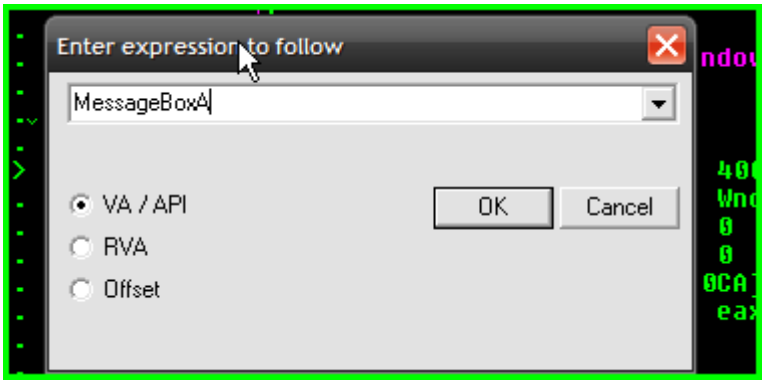
**Execute till User code** hoặc nhấn phím tắt là **Alt+F9**. Olly sẽ đưa chúng ta về đúng nơi như những gì ta vừa mới thực hiện ở trên (tuy nhiên chú ý là phải bỏ Bp trên memory trước rồi mới thực hiện nhấn Alt+F9) :

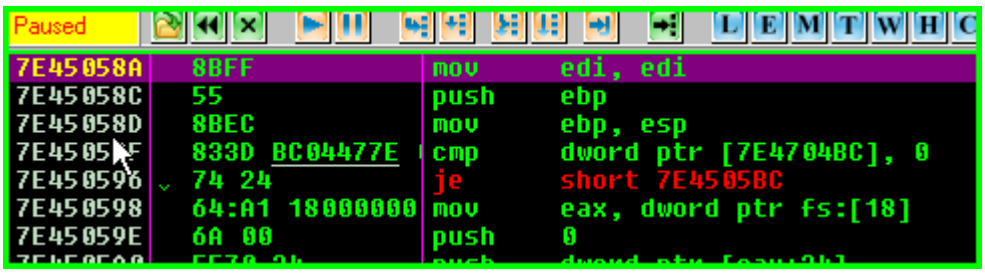


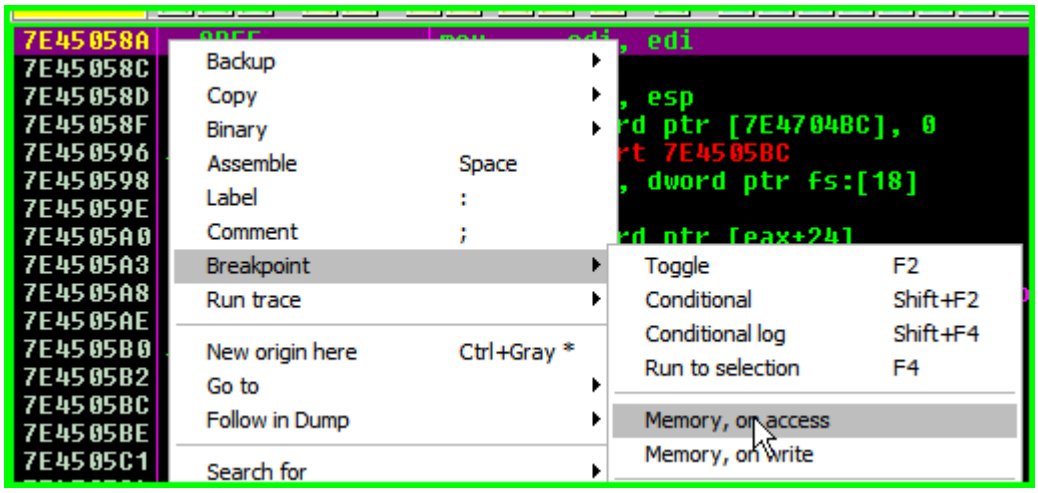

Mở rộng vấn đề ra một chút, ở phần trước tôi đã thực hiện việc phân tích sơ bộ chương trình và các bạn cũng đã biết rằng khi chúng ta nhập sai thông tin về UserName và Serial thì ta sẽ nhận được một thông báo lỗi. Bây giờ tôi muốn đặt BP tại hàm MessageBoxA (vd: Bp MessageBoxA), nhưng vì một lý do nào đó chương trình sử dụng cơ chế anti-bp và phát hiện ra rằng có CC (tức int3) được sử dụng ngay lập tức nó sẽ terminate Olly của ta. Vậy không lẽ ta phải chịu vậy sao và không thể đặt bp tại API MessageBoxA. Xin thưa với các bạn, đến lúc này việc đặt Memory Bp mới có tác dụng và hiệu quả. Để tìm địa chỉ của hàm MessageBoxA tôi làm như sau trong Olly :


Vậy là địa chỉ hàm MessageBoxA của tôi là ở 0x7E45058A (trên máy các bạn có thể khác), lúc này tôi chỉ việc qua cửa sổ CPU và nhấn Ctrl+G, gõ địa chỉ này vào hoặc thay vào đó tôi gõ trực tiếp tên hàm vào và nhấn Ok, Olly sẽ đưa tôi đến đúng nơi cần tìm :

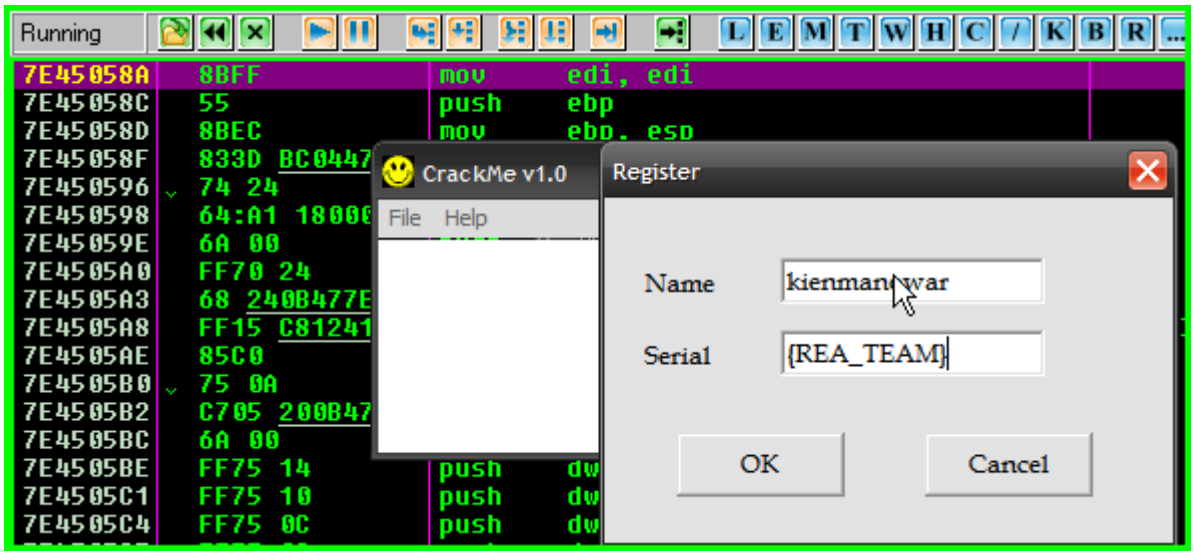




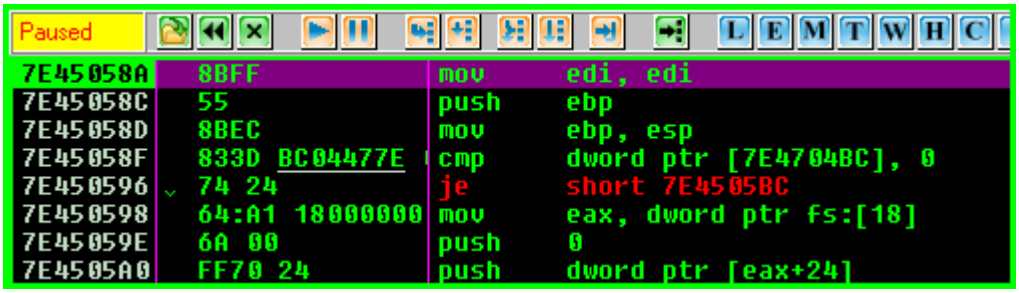

Bây giờ việc đặt bp hết sức đơn giản, thực hiện như hình minh họa dưới đây :



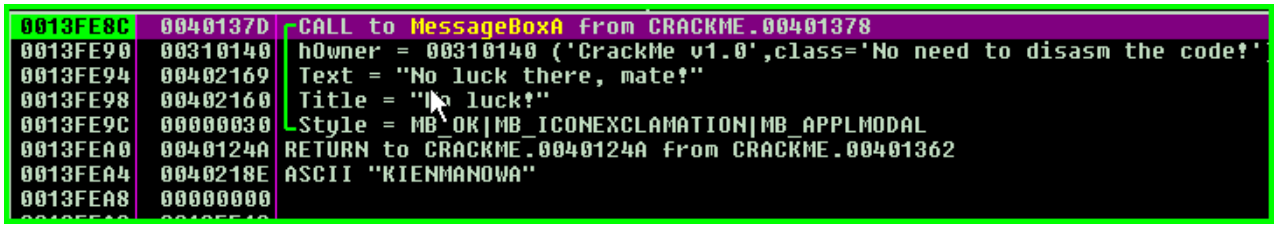

Giờ ta test thử xem việc đặt bp có hoạt động đúng như ta mong đợi không nhé. Nhấn F9 để run chương trình, sau đó nhập fake info gì là tùy bạn :




Nhẹ nhàng bấm OK xem thế nào ☺ :



Khà khà Olly đã break đúng chỗ rồi, quan sát cửa sổ Stack :




Giờ tôi muốn quay trở về code chính của chương trình thì thế nào nhỉ? Quá đơn giản, bạn remove memory bp đi và nhấn Alt+F9, chương trình sẽ thực thi và show nag. Ta nhấn Ok để chấp nhận, sau đó Olly sẽ đưa chúng ta tới đây :


Ok vậy là phần 10 của loạt tuts về Ollydbg đến đây là kết thúc, qua bài viết này tôi đã giới thiệu sơ qua về cách thiết lập BP, các thao tác thông qua command bar để đặt các bp với các lệnh BP và BPX, cung cấp thông tin về việc đặt memory bp và cách xử lý để có thể đặt bp nếu như chương trình sử dụng cơ chế anti-bp. Trong vài viết tiếp theo về loạt tuts này hi vọng tôi sẽ cùng các bạn khám phá thêm về Hardware BP, Message breakpoint và Conditional breakpoint …. chắc sẽ có nhiều điều thú vị lắm.Hẹn gặp lại các bạn trong các phần tiếp theo, By3 By3!! ☺

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 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).Thanx to Orthodox, kanxue, TiGa 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**