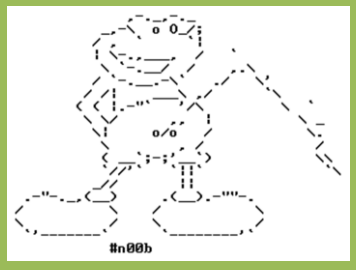
2017

[Cracking with OllyDbg]

Based on OllyDbg tuts of Ricardo Narvaja (CrackLatinos Team)

www.reaonline.net

kienmanowar



19/05/2017

Mục Lục

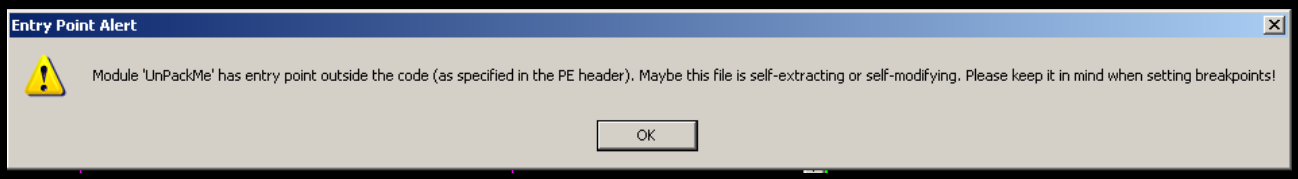
I. Giới thiệu chung

Packer tiếp theo để thực hành trong phần này là ASPack, khác với UPX, packer này tập trung hơn vào phần security, nó áp dụng các kĩ thuật nâng cao như self-modifying code nhằm làm cho việc đặt các breakpoint khó khăn hơn, ... Tuy nhiên, cũng tương tự như với UPX, ta hoàn toàn có thể thực hiện manual unpack bằng cách sử dụng HWBP tại stack address. Target thực hành trong phần này là file **UnPackMe_ASPack2.12.exe** (được gửi kèm ở phần 26). Việc tìm OEP của target này cũng đã được đề cập trong phần 26, các bạn có thể đọc lại.

Trong phần này, thay vì sử dụng các công cụ để dump file như LordPE hay PETools, tôi sẽ sử dụng một Plugin được viết cho OllyDbg là **OllyDump**, xem như là thêm một tùy chọn trong việc thực hiện dump file sau khi đã tới được OEP.

II. Phân tích và xử lý target

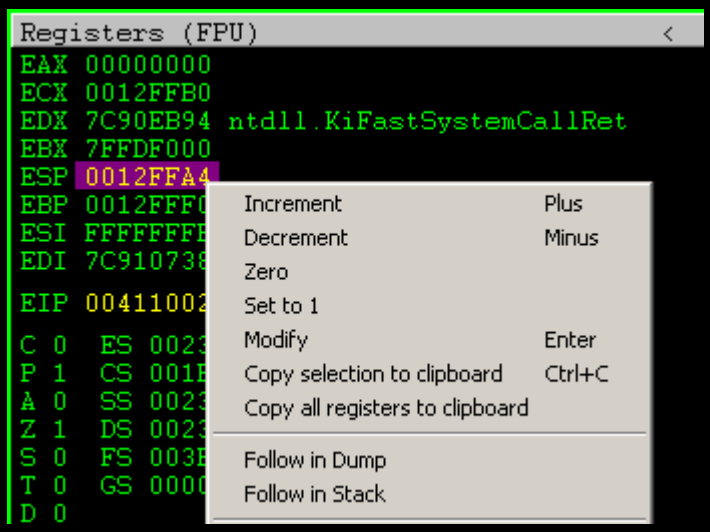
Ok, tiến hành load unpackme vào OllyDbg, nhận được thông báo thường thấy đối với một file bị pack:



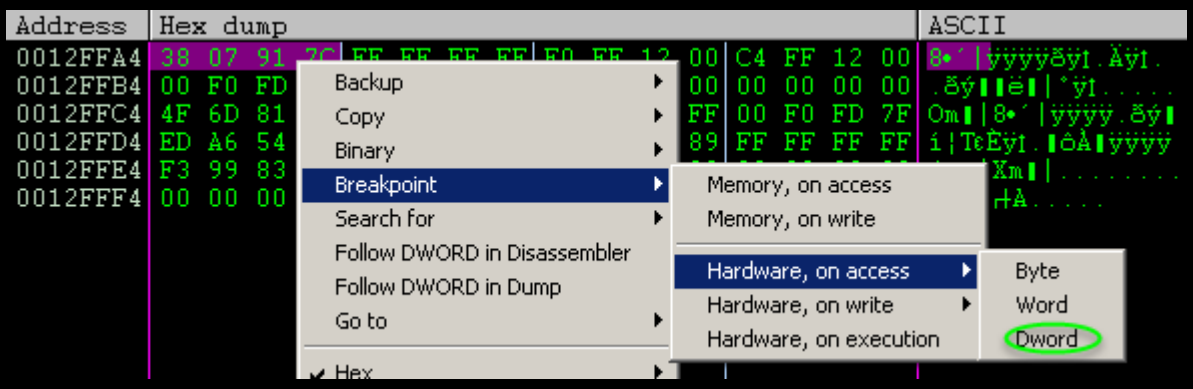
Nhấn OK, sẽ dừng lại tại lệnh `PUSHAD` như hình dưới, với dấu hiệu này ta sẽ áp dụng phương pháp `PUSHAD` (*đã giới thiệu ở phần trước*) để tìm tới OEP của unpackme.

```
00411001    60              pushad
00411002    E8 03000000     call    UnPackMe.0041100A
00411007 -  E9 EB045D45     jmp     459E14F7
0041100C    55              push    ebp
0041100D    C3              retn
0041100E    E8 01000000     call    UnPackMe.00411014
00411013 v  EB 5D           jmp     short UnPackMe.00411072
00411015    BB EDFFFFFF     mov     ebx,-13
0041101A    03DD            add     ebx,ebp
0041101C    81EB 00100100   sub     ebx,11000
00411022    83BD 22040000   cmp     dword ptr [ebp+422],0
00411029    899D 22040000   mov     dword ptr [ebp+422],ebx
0041102F v  0F85 65030000   jnz     UnPackMe.0041139A
00411035    8D85 2E040000   lea     eax,dword ptr [ebp+42E]
0041103B    50              push    eax
0041103C    FF95 4D0F0000   call    near dword ptr [ebp+F4D]
00411042    8985 26040000   mov     dword ptr [ebp+426],eax
00411048    8BF8            mov     edi,eax
0041104A    8D5D 5E         lea     ebx,dword ptr [ebp+5E]
0041104D    53              push    ebx
0041104E    50              push    eax
```

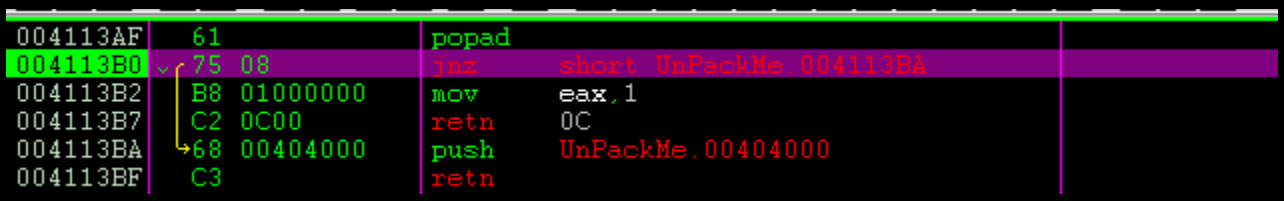
Nhấn **F7/F8** để trace qua lệnh `PUSHAD`, quan sát giá trị của thanh ghi **ESP** tại cửa sổ Registers, ta có được như hình (lưu ý: giá trị này có thể khác ở máy của các bạn):



Chuột phải tại thanh ghi **ESP** và chọn **Follow in Dump**. Tại cửa sổ Dump, thiết lập một Hardware Breakpoint (HWBP) như hình:



Sau khi đặt BP xong, nhấn **F9** để run, break tại đây trong OllyDbg:



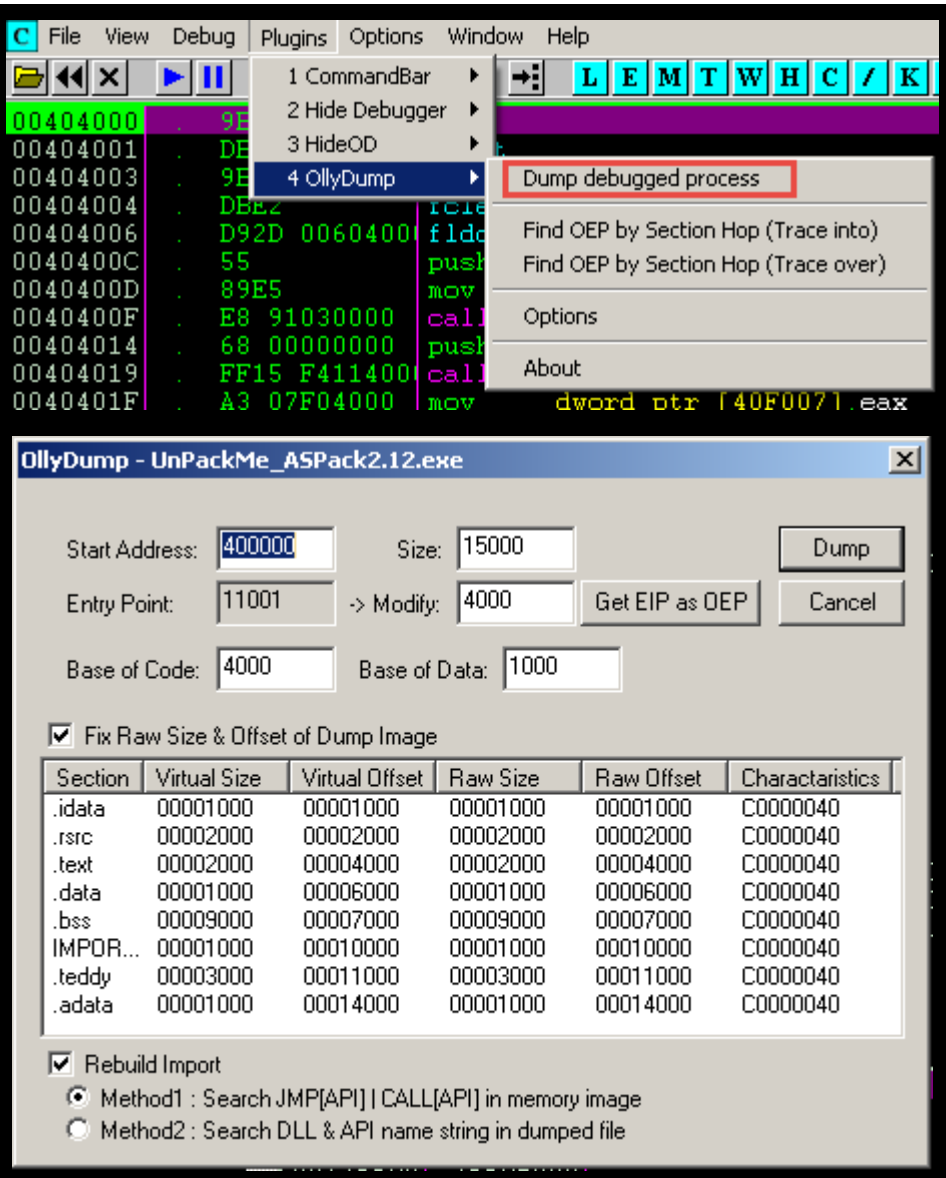
Để ý cặp lệnh `PUSH & RETN` trên hình, cặp lệnh này sẽ tương đương với một lệnh `JMP` tới địa chỉ được Push lên Stack (ở đây là `0x404000`). Trace bằng **F7/F8** qua lệnh `RETN` sẽ tới OEP của unpackme:

```
00404000   .  9B            wait
00404001   ?  DBE3          finit
00404003      9B            db      9B
00404004      DB            db      DB
00404005      E2            db      E2
00404006      D9            db      D9
00404007      2D            db      2D                      CHAR '-'
00404008      00            db      00
00404009      60            db      60                      CHAR '`'
0040400A      40            db      40                      CHAR '@'
0040400B      00            db      00
0040400C      55            db      55                      CHAR 'U'
0040400D      89            db      89
0040400E      E5            db      E5
0040400F      E8            db      E8
00404010      91            db      91
00404011      03            db      03
00404012      00            db      00
00404013      00            db      00
00404014      68            db      68                      CHAR 'h'
00404015      00            db      00
00404016   .  0000          add     byte ptr [eax],al
00404018   ?  00FF          add     bh,bh
0040401A   ?  15 F4114000   adc     eax,UnPackMe.004011F4
0040401F   .  A3 07F04000   mov     dword ptr [40F007],eax
00404024      60            db      60                      CHAR '`'
```

Như trên hình, ta thấy code nhìn hơi khó hiểu, tuy nhiên khi tới OEP là code đã được bung hoàn toàn do đó tiến hành Analyse Code trong OllyDbg bằng cách nhấn phím tắt là **Ctrl + A**:

```
00404000   .  9B            wait
00404001   .  DBE3          finit
00404003   .  9B            wait
00404004   .  DBE2          fclex
00404006   .  D92D 0060400  fldcw   word ptr [406000]
0040400C   .  55            push    ebp
0040400D   .  89E5          mov     ebp,esp
0040400F   .  E8 91030000   call    UnPackMe.004043A5
00404014   .  68 00000000   push    0                       pModule = NULL
00404019   .  FF15 F411400  call    near dword ptr [4011F4]  GetModuleHandleA
0040401F   .  A3 07F04000   mov     dword ptr [40F007],eax
00404024   .  60            pushad
00404025   .  8925 0BF0400  mov     dword ptr [40F00B],esp
0040402B   .v E9 30000000   jmp     UnPackMe.00404060
00404030   >  8B25 0BF0400  mov     esp,dword ptr [40F00B]
00404036   .  61            popad
00404037   .  E8 A9080000   call    UnPackMe.004048E5
0040403C   .  E8 FD030000   call    UnPackMe.0040443E
00404041   .  89EC          mov     esp,ebp
00404043   .  5D            pop     ebp
00404044   .  FF35 D4F1400  push    dword ptr [40F1D4]       ExitCode = 0
0040404A   .  FF15 EC11400  call    near dword ptr [4011EC]  ExitProcess
00404050   .  9B            wait
```

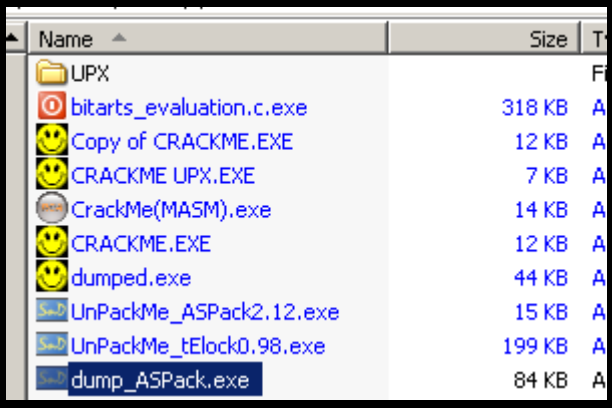
Sau khi analyse xong ta thấy code rõ ràng hơn nhiều. Tại OEP, bước tiếp theo sẽ tiến hành việc dump file. Lựa chọn **OllyDump** plugin như sau:



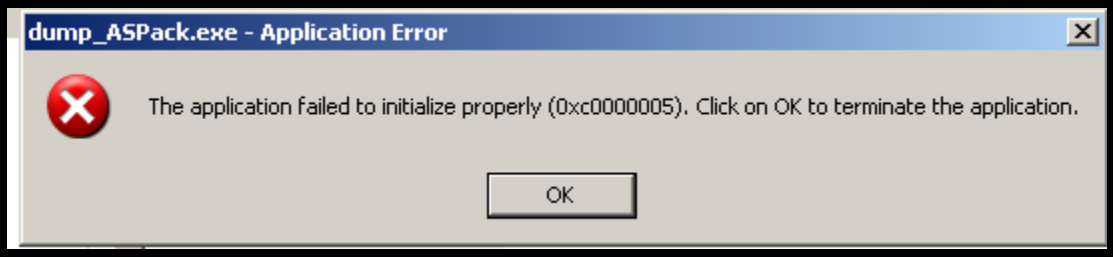
Quan sát tại cửa sổ OllyDump, lúc này ta thấy giá trị EP của file khi bị pack là `0x11001` sẽ được sửa lại thành EP ta đã tìm được ở trên `(0x4000 = 0x404000 – 0x400000 (ImageBase))`.

Bên cạnh các thông tin liên quan đến EP, Base of Code, Sections, ... tại màn hình OllyDump ta thấy có lựa chọn **Rebuild Import** với hai tùy chọn là **Method1** và **Method2**. Khi lựa chọn Rebuild Import, OllyDump sẽ thực hiện công việc tương tự như công cụ ImpREC đã làm để fix lại IAT. Lựa chọn này tùy từng trường hợp và thường chỉ hiệu quả đối với một số packer đơn giản, bạn có thể thử dump file và lựa chọn từng Method để xem kết quả như thế nào. Với những người chuyên thực hiện Unpack, họ rất ít khi sử dụng tùy chọn này mà sẽ nhường lại phần fix IAT cho các công cụ chuyên dụng hơn.

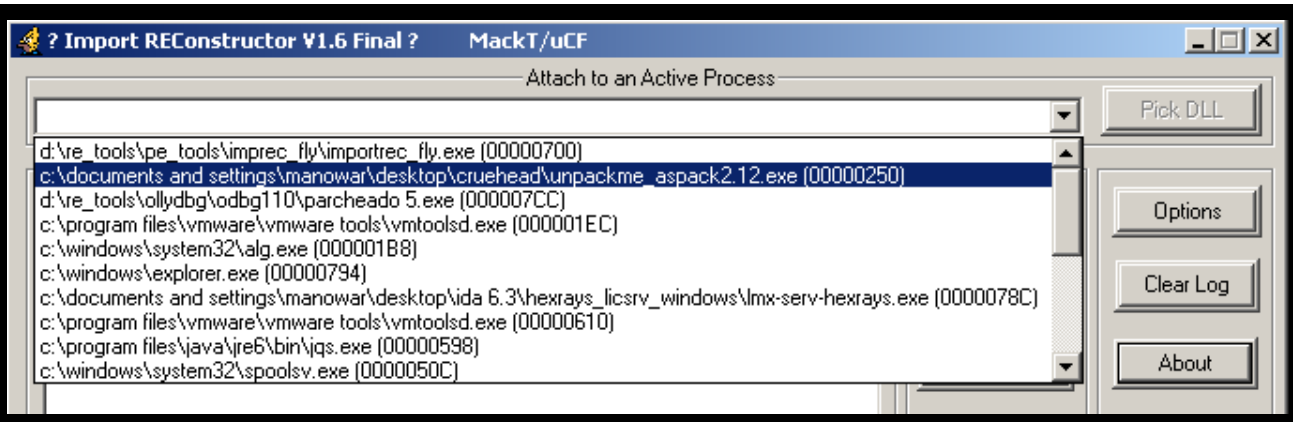
Bỏ tùy chọn Rebuid Import, sau đó nhấn Dump để thực hiện dump ra file. Save file được dump ra với tên mới là **dump_ASPack.exe**:



Thử run file đã dump khi chưa fix IAT xem thế nào, nhận được thông báo lỗi sau:

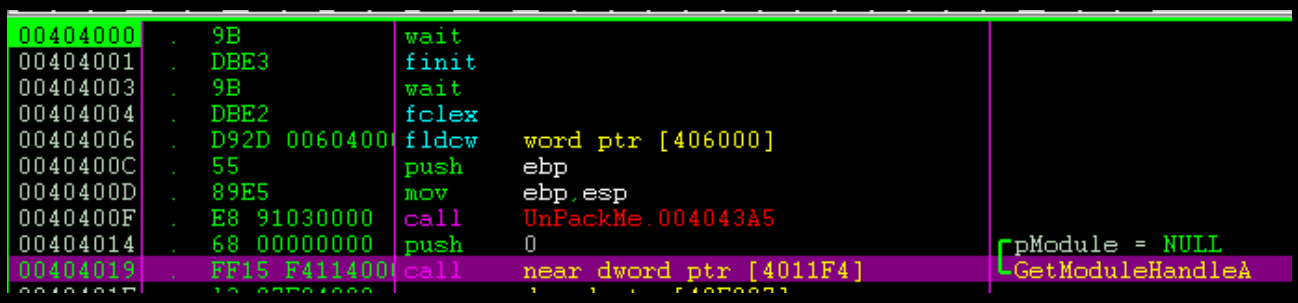


Nhấn OK để thoát chứ biết sao giờ . Giữ nguyên màn OllyDbg đang dừng lại tại OEP, mở ImpREC để tiến hành fix lại IAT. Tại màn hình của ImpREC, lựa chọn active process như hình:

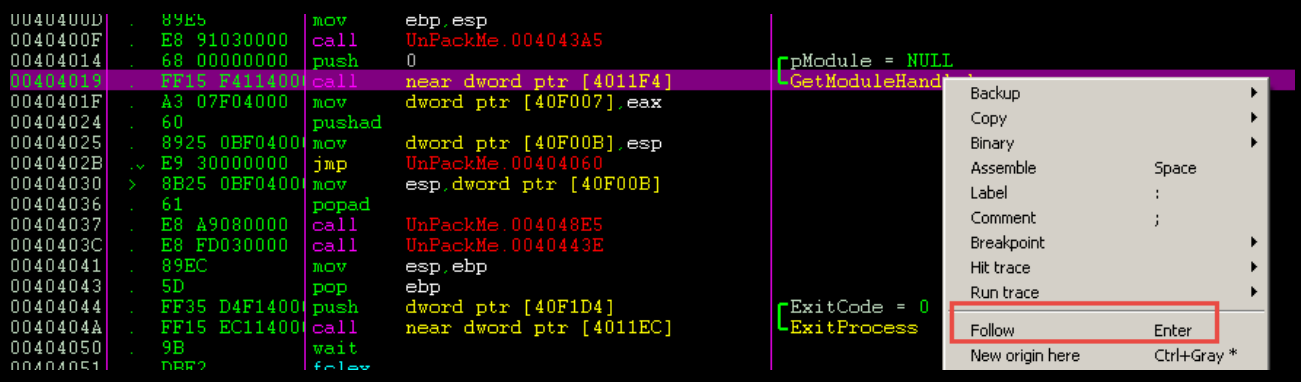


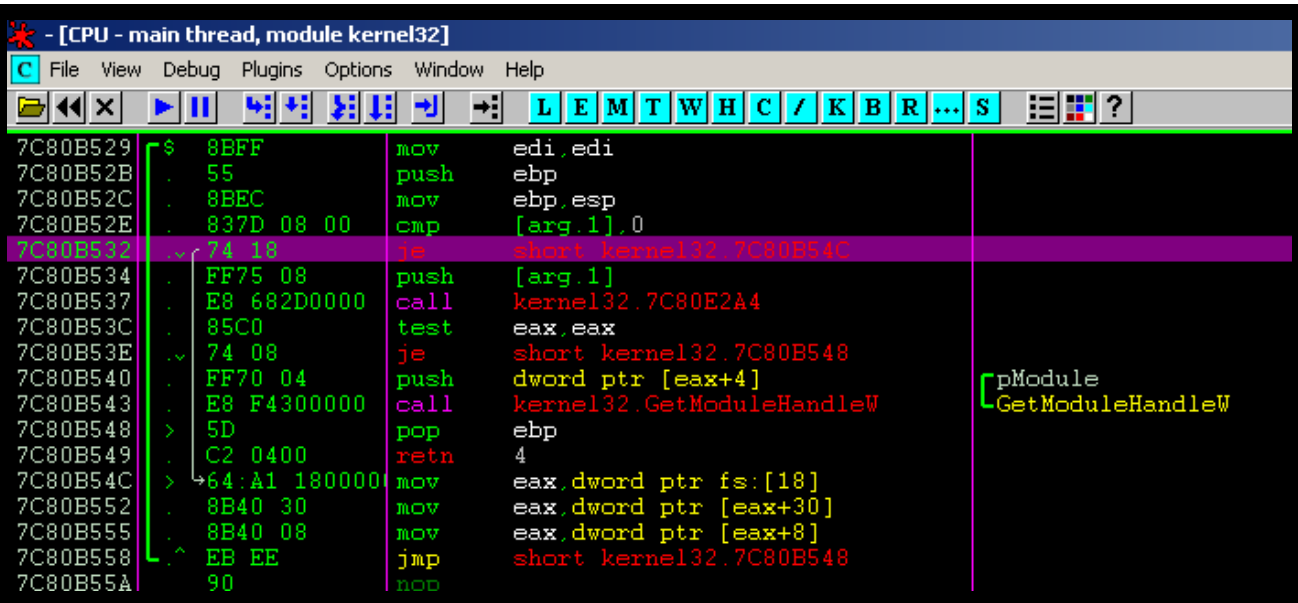
Quay trở lại OllyDbg, tìm cách để xác định lại 3 thông tin là: **IAT Start, IAT Size** và **OEP**. Đây là bước quan trọng, nếu sai sót việc fix IAT sẽ khiến file vẫn bị lỗi, không chạy được, v...v... OEP thì ta đã tìm được trong OllyDbg ở bước trước là `0x404000`, tuy nhiên để sử dụng cho việc fix file bằng ImpREC thì phải trừ đi địa chỉ ImageBase `0x400000`, ta có kết quả là **0x4000**.

Để xác định IAT Start và IAT Size, thực hiện theo cách tôi đã giới thiệu ở phần trước. Bên dưới OEP ta sẽ thấy một lời gọi tới hàm API là `GetModulehandleA`.



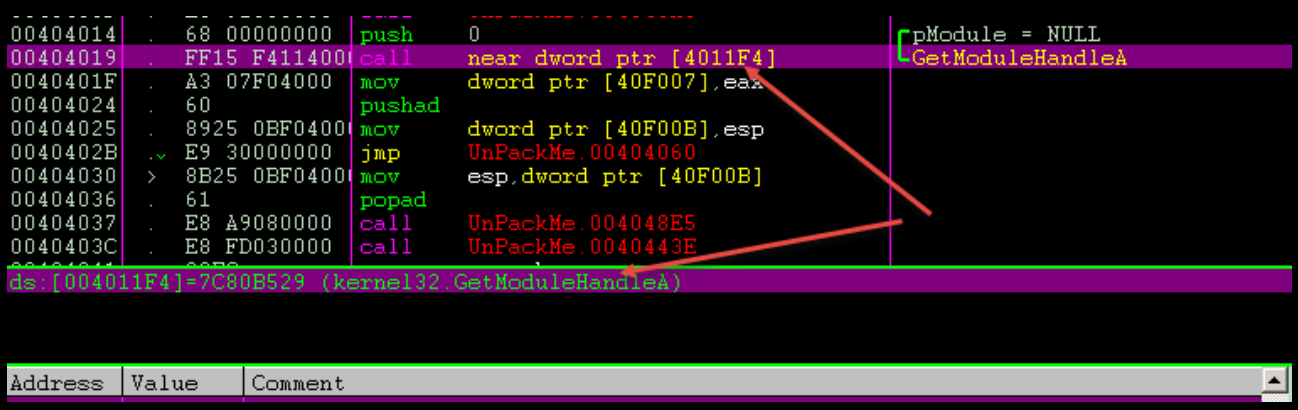
Chọn lệnh call này, nhấn chuột phải và chọn **Follow** (hoặc nhấn phím tắt là **Enter**) sẽ tới nơi thực hiện API trong module `kernel32`:



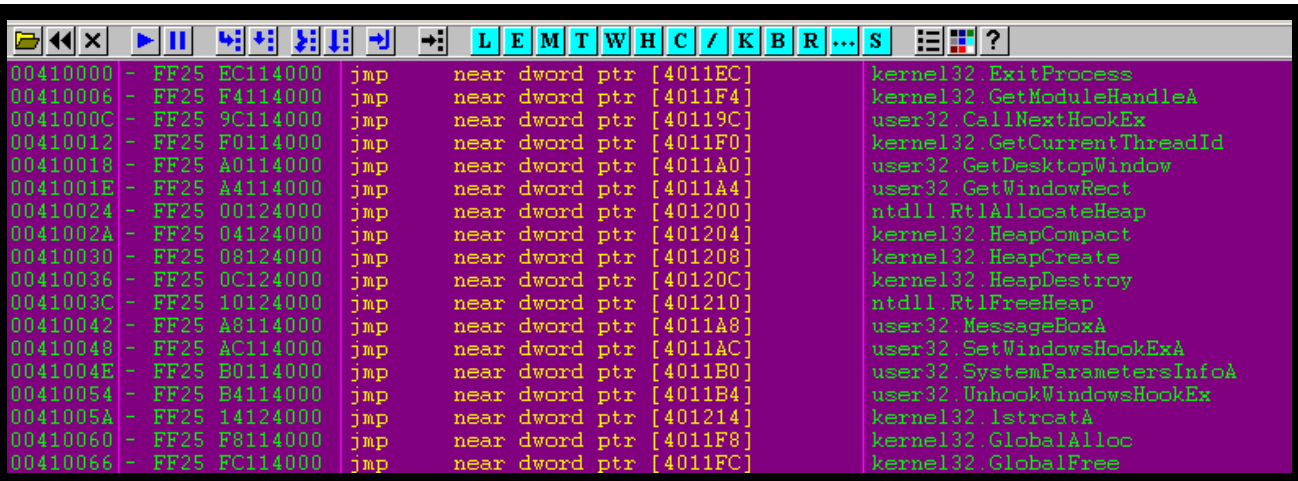


Như các bạn thấy, ta đi thẳng tới hàm API mà không cần phải thông qua một lệnh nhảy gián tiếp (indirect jump (`FF 25`)) như đã thấy ở phần trước. Ở target này đã sử dụng một lệnh call gián tiếp (indirect call (`FF 15`)), để chuyển tới API. Với lệnh call kiểu này, nó sẽ đọc địa chỉ của hàm API được lưu tại vị trí bộ nhớ, sau đó chuyển tới địa chỉ hàm API để thực hiện hàm.

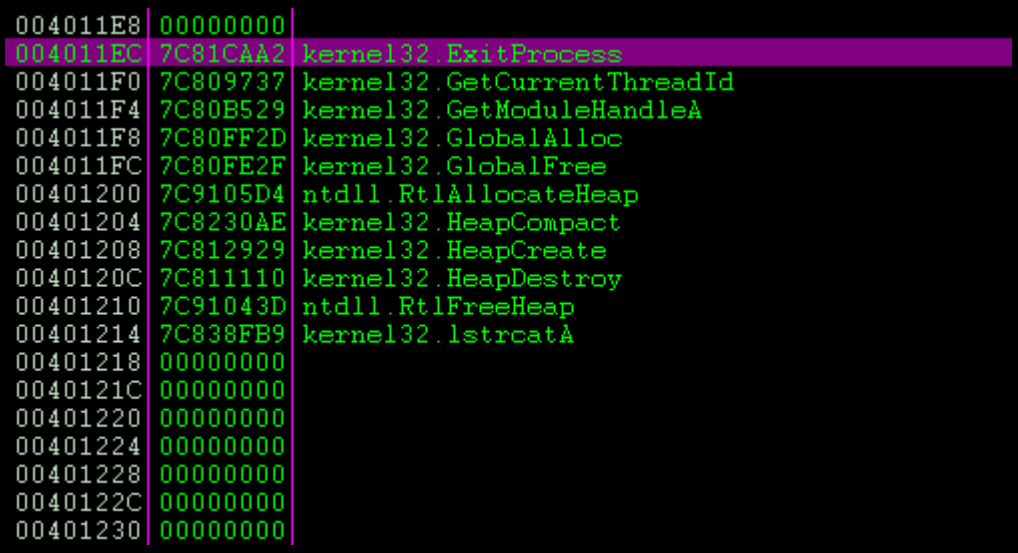
Quan sát thông tin tại của sổ Tip, ta có thể dễ dàng thấy được địa chỉ của API:



Như vậy, ta nhận thấy rằng `4011F4` là một phần của IAT, tại đó lưu trữ địa chỉ của hàm API như đã biết ở trên là `GetModuleHandleA`. Nếu bạn muốn thử tìm kiếm theo các lệnh nhảy giống như cách ở phần trước, thực hiện tìm kiếm theo binary với giá trị cần tìm là `FF 25`, kết quả có được như sau tại vùng nhớ thuộc section **`.teddy`**:



Như quan sát trên hình, lệnh nhảy cũng cho cùng kết quả về IAT. Follow sang cửa sổ Dump và sắp xếp lại theo kiểu **Long > Address**, ta có được thông tin về IAT như sau:



Ta có thể thấy bên dưới địa chỉ `0x401218` sẽ toàn các giá trị `0x0`, như vậy có thể kết luận **IAT End** là `0x401218`. Ta cần phải tìm **IAT Start**.

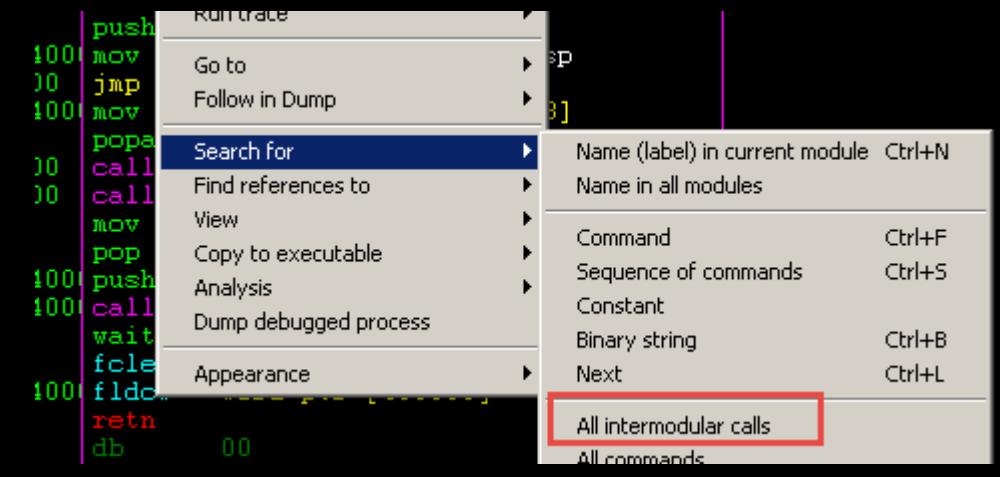
Theo quan sát trên hình, ta mới thấy có các API thuộc các dlls như `kernel32.dll`, `ntdll.dll`, mà theo kinh nghiệm unpack thì thường chắc chắn phải có APIs của dll `user32.dll` nữa. Như phần trước đã đề cập, các địa chỉ APIs giữa các DLLs sẽ phân tách bằng 4 bytes `00`. Từ `004011EC 7C81CAA2 kernel32.ExitProcess,` cuộn chuột lên ta sẽ thấy một vùng nhớ khác nhưng không chứa địa chỉ APIs nào như sau:

```
004011B8 00000000
004011BC 000010C2
004011C0 000010D0
004011C4 000010E6
004011C8 000010FA
004011CC 00001108
004011D0 00001116
004011D4 00001122
004011D8 00001130
004011DC 0000113E
004011E0 0000114C
004011E4 00001158
004011E8 00000000
004011EC 7C81CAA2 kernel32.ExitProcess
```

Tiếp tục cuộn chuột lên để tìm kiếm tiếp thông tin, ta có được:

```
00401198 00000000
0040119C 77D4ED6E user32.CallNextHookEx
004011A0 77D4D7BB user32.GetDesktopWindow
004011A4 77D4B57C user32.GetWindowRect
004011A8 77D8050B user32.MessageBoxA
004011AC 77D702B2 user32.SetWindowsHookExA
004011B0 77D50554 user32.SystemParametersInfoA
004011B4 77D6F29F user32.UnhookWindowsHookEx
004011B8 00000000
004011BC 000010C2
```

Đúng như phán đoán, ta có được thông tin về các APIs thuộc `User32.dll`. Tuy nhiên, nếu các bạn quan sát tại cửa sổ Memory map, các bạn sẽ thấy thông tin của các Dlls khác nữa đã được map vào bộ nhớ như `GDI32.dll`, `ADVAPI32.dll` ..., Để chắc chắn unpackme có gọi hàm APIs nào nữa ngoài kết quả đã có được ở trên, ta thực hiện **Search for –> All intermodular calls**:



Kết quả có được như sau:

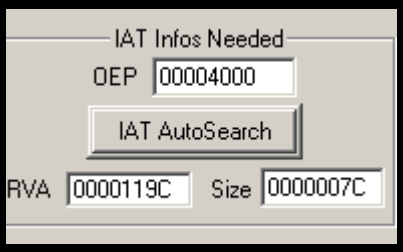
Address	Disassembly	Destination
00404000	wait	(Initial CPU selection)
00404019	call near dword ptr [4011F4]	kernel32.GetModuleHandleA
0040404A	call near dword ptr [4011EC]	kernel32.ExitProcess
004042B6	call near dword ptr [401214]	kernel32.lstrcatA
004043E3	call near dword ptr [401208]	kernel32.HeapCreate
00404514	call near dword ptr [40120C]	kernel32.HeapDestroy
0040457F	call near dword ptr [401200]	ntdll.RtlAllocateHeap
004045B2	call near dword ptr [401204]	kernel32.HeapCompact
004045CE	call near dword ptr [401200]	ntdll.RtlAllocateHeap
0040466D	call near dword ptr [401210]	ntdll.RtlFreeHeap
00404BA7	call near dword ptr [4011B0]	user32.SystemParametersInfoA
00404BF6	call near dword ptr [4011A0]	user32.GetDesktopWindow
00404C0F	call near dword ptr [4011A4]	user32.GetWindowRect
00404E82	call near dword ptr [40119C]	user32.CallNextHookEx
00404F19	call near dword ptr [4011F0]	kernel32.GetCurrentThreadId
00404F34	call near dword ptr [4011AC]	user32.SetWindowsHookExA
00404F59	call near dword ptr [4011A8]	user32.MessageBoxA
00404F6B	call near dword ptr [4011B4]	user32.UnhookWindowsHookEx
004055A4	call near dword ptr [4011FC]	kernel32.GlobalFree
0040575E	call near dword ptr [4011F8]	kernel32.GlobalAlloc

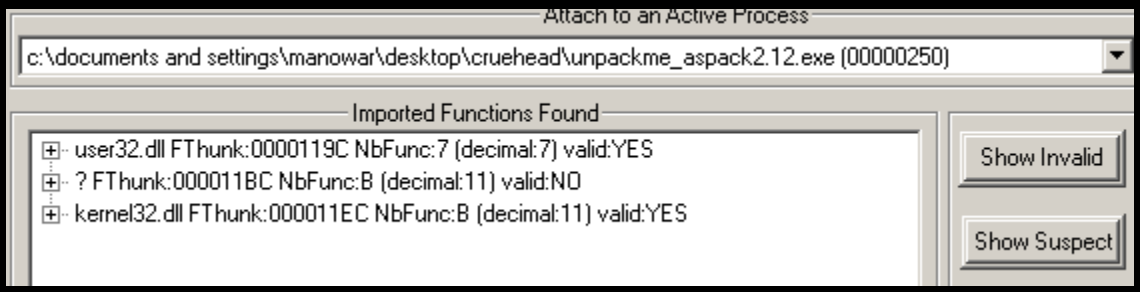
Với kết quả này cho thấy unpackme chỉ sử dụng các API từ 3 dll là `kernel32.dll`, `ntdll.dll` và `user32.dll`. Thông tin này cũng trùng khớp với thông tin IAT mà ta thực hiện tìm kiếm giá trị `FF 25` ở trên. Kết luận, ta có được thông tin của IAT Start là `0x40119C`.

Toàn bộ các thông tin cần đã có, tổng hợp lại ta có kết quả như sau:

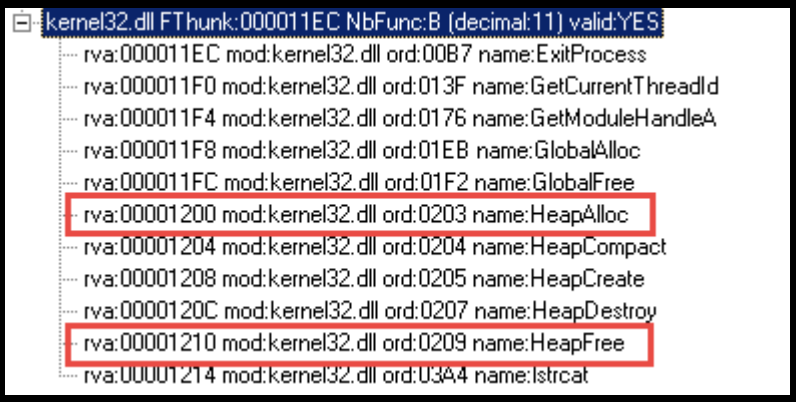
- `OEP=4000`
- `RVA hay IAT Start = 119C (0x40119C – 0x400000)`
- `IAT Size = 0x401218-0x40119c = 0x7C`

Cung cấp toàn bộ các thông tin cho công cụ ImpREC, sau đó nhấn **Get Imports**:

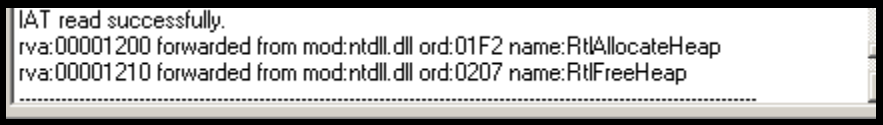




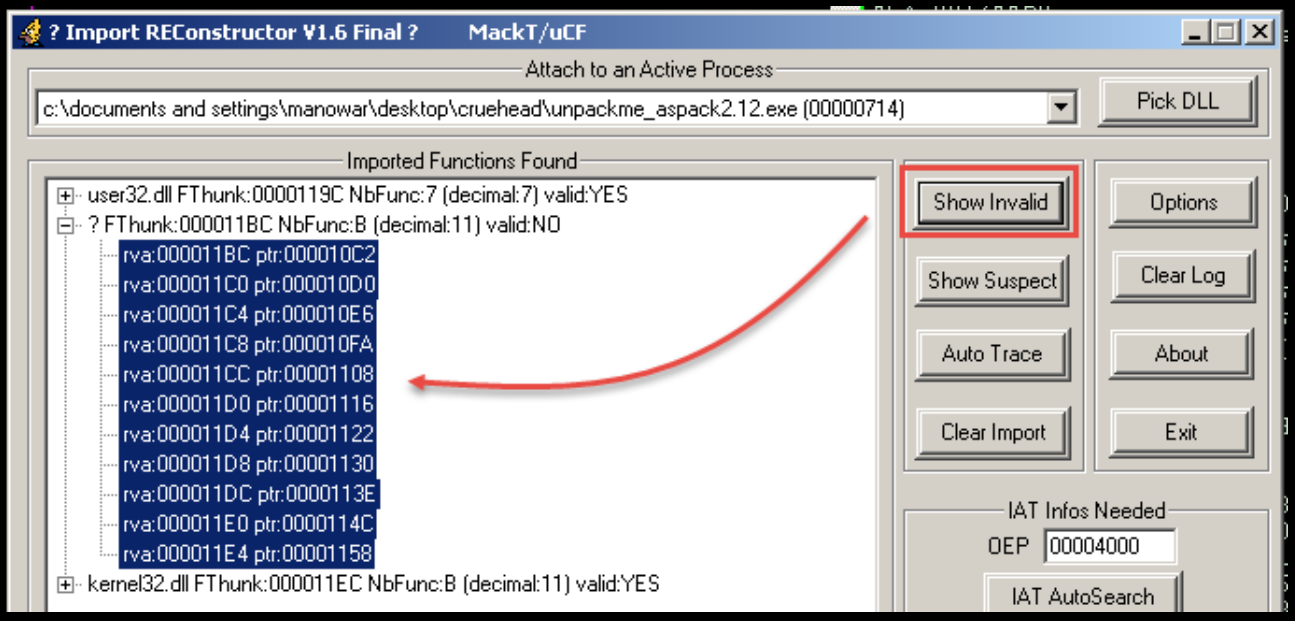
Kết quả ta thấy ImpREC lấy cả các Invalid Thunks, đây chính là vùng junk entries nằm giữa `user32.dll` và `kernel32.dll`. Ngoài ra, quan sát các giá trị tương ứng với các địa chỉ `0x401200` và `0x401210`, ta thấy:



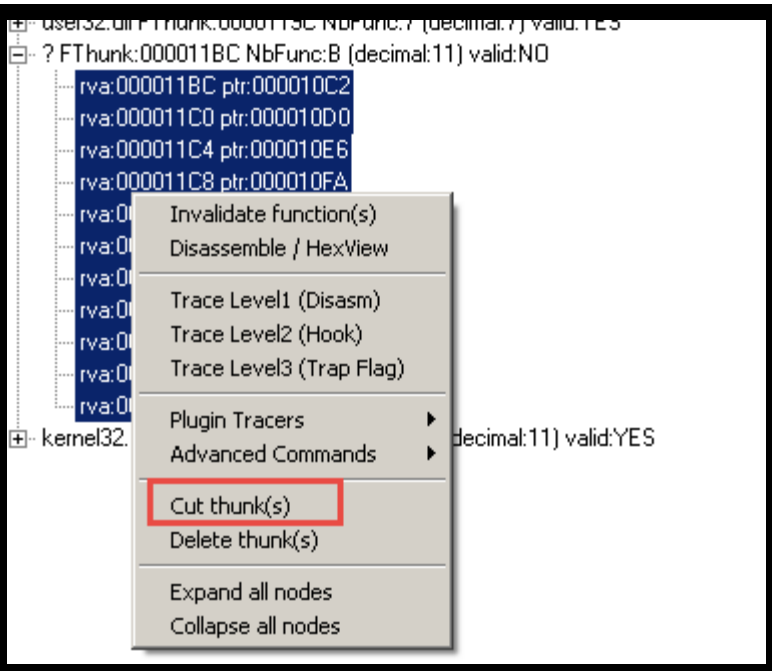
ImpREC đã tự động thay thế bằng các hàm có chức năng tương tự trong `kernel32.dll`, thay vì gọi trực tiếp tới `ntdll.dll`. Xem log của ImpREC:



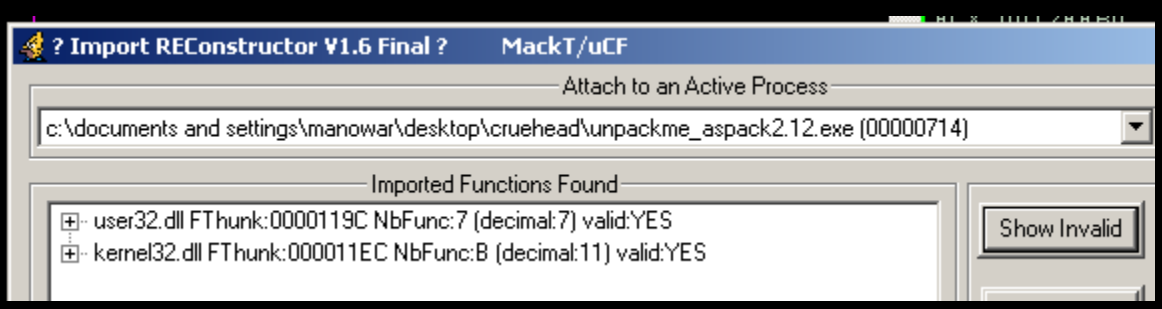
Như vậy, ImpREC đã lấy được đầy đủ thông tin của IAT bao gồm cả các Invalid Thunk. Để loại bỏ các Invalid Thunk này, tại màn hình ImpREC, chọn **Show Invalid**:



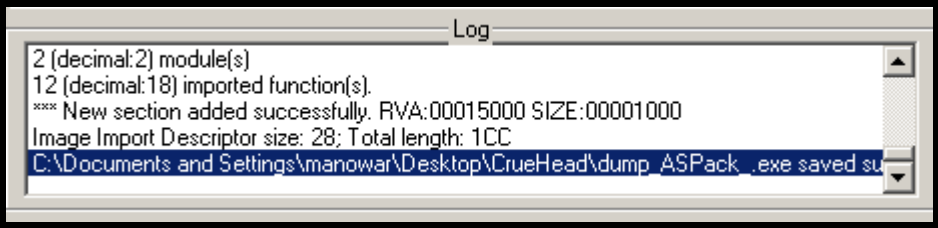
Chuột phải tại vùng Invalid đã được đánh dấu và chọn **Cut thunk(s)**:



Kết quả cuối cùng như sau:



Cuối cùng chọn Fix Dump tại màn hình của ImpREC, lựa chọn file đã dump ở bước trước là **dump_ASPack.exe** để fix:



Sau khi fix xong, ImpREC sẽ tự động lưu lại thành một file mới với tên là **dump_ASPack_.exe**. Chạy thử file đã fix xem có thực thi được không, kết quả file thực thi bình thường:

III. Kết luận

Toàn bộ phần 29 đến đây là kết thúc, cảm ơn các bạn đã dành thời gian để theo dõi. Hẹn gặp lại các bạn ở phần tiếp theo!

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:
kienbigmummy[at]gmail.com