

INTRODUCCION AL CRACKING CON OLLYDBG PARTE 9

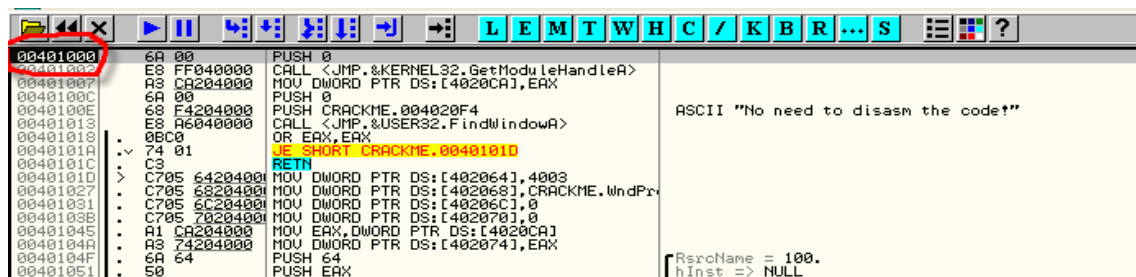
Bueno ya hemos terminado de explicar las instrucciones assembler y ahora se nos presenta una disyuntiva, por donde seguir, hay muchísimo que aprender y practicar, por lo cual trataremos de ir como siempre pasito a pasito, sin apuros usando lo que aprendimos, y agregando mas cosas que faltan aun.

Por supuesto nuestra primera victima será el famoso CRACKME DE CRUEHEAD pero no nos limitaremos solo a ir viendo las distintas formas de crackearlo, si no también que nos iremos ubicando con conceptos que luego nos serán básicos a la hora de profundizar en el arte.

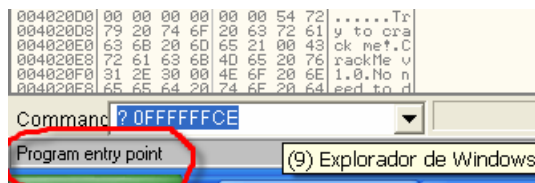
Abramos el famoso crackme en OLLYDBG y orientémonos un poco antes de empezar.

Veremos en el mismo crackme algunas definiciones que son útiles para cualquier programa.

ENTRY POINT: Es la primera línea que se ejecuta del programa normalmente, no confundir con OEP (Original Entry Point) que es otra cosa que definiremos en partes posteriores de esta introducción, o sea si abrimos un programa en OLLYDBG, este para y lo analiza, allí donde termina de analizar y queda detenido es el ENTRY POINT del programa.



En nuestro caso del Crackme de Cruehead, el ENTRY POINT será 401000 y OLLYDBG nos muestra en el margen inferior que normalmente usa para avisarnos porque esta detenido un programa, que en este caso estamos detenidos en el ENTRY POINT.



Casi todos los programas (el 99 %), cuando arrancan en OLLYDBG se detienen en el ENTRY POINT, los que no lo hacen es porque tienen alguna modificación especial realizada para evitar que pare en el mismo, ese tipo de trucos veremos mas adelante, pero la idea es saber esto.

Otro concepto que necesitamos y que usaremos en el de DLLs y sus APIS

Vemos que en ciertos puntos del programa el mismo nos muestra una CALL o JMP que salta en vez de a una dirección como normalmente vimos por ejemplo CALL 401020 o JMP 421367, en la imagen anterior vemos que el call por ejemplo es

CALL LoadIconA

Y a la derecha nos muestra cierta información, pero que es en este caso LoadIconA?

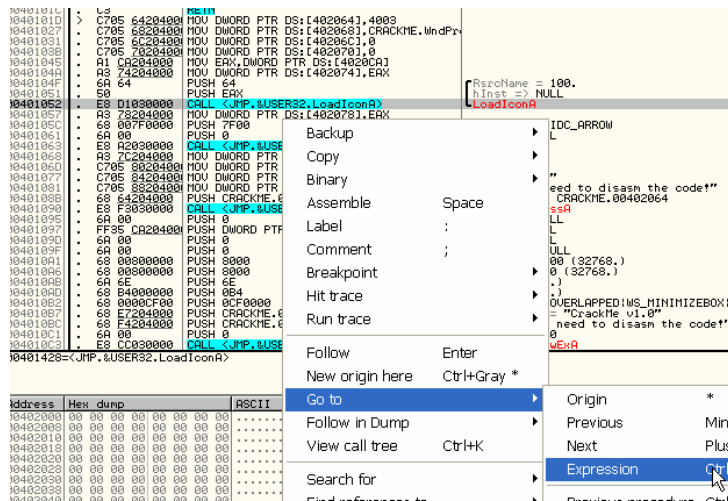
El sistema WINDOWS utiliza para evitar que los programadores repitan las mismas rutinas que en casi todos los programas son usadas, un sistema de archivos de extensión DLL que son archivos ejecutables, pero además tienen la propiedad de tener FUNCIONES DE EXPORTACION O APIS que no son mas que funciones que pueden ser utilizadas por cualquier programa, para tener que evitar repetir lo mismo en todos los programas.

Así en este caso llama a la DLL user32.dll y ella tiene una función denominada LoadIconA, que realizara cierto trabajo para facilitarme la programación.

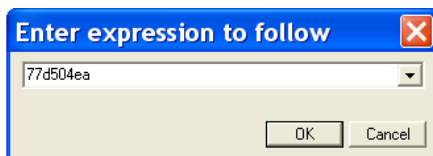
El caso mas sencillo de entender es la api MessageBoxA

Si en la commandbar del OLLYBD tipeo

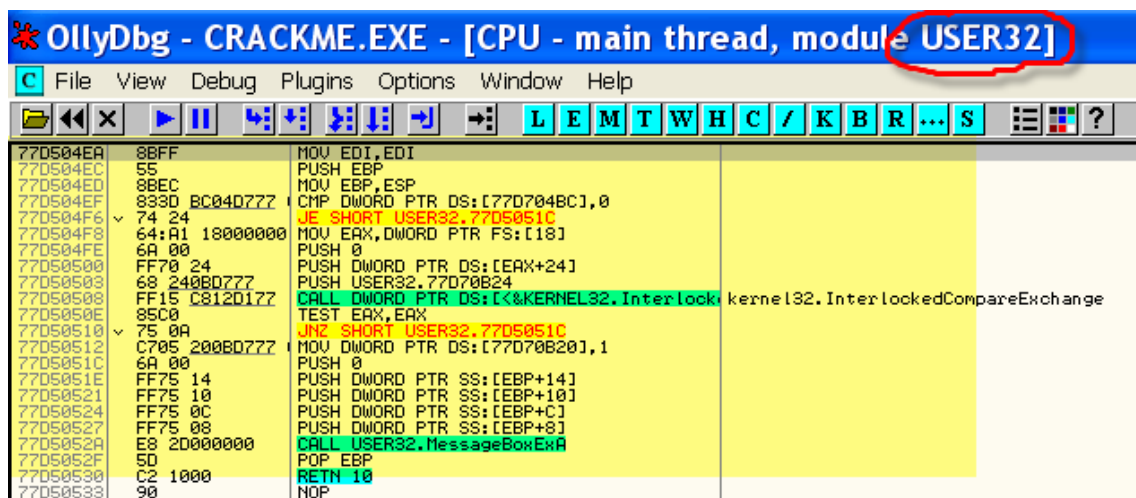
Nos muestra la dirección de dicha api, si voy a mirar a dicha dirección en el listado haciendo CLICK DERECHO y copiando la dirección que salio en sus maquinas ya que puede variar en cada una.



Allí tipo la dirección que en mi maquina apareció

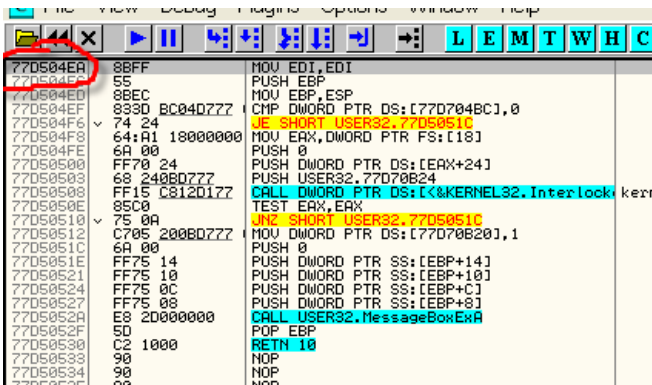
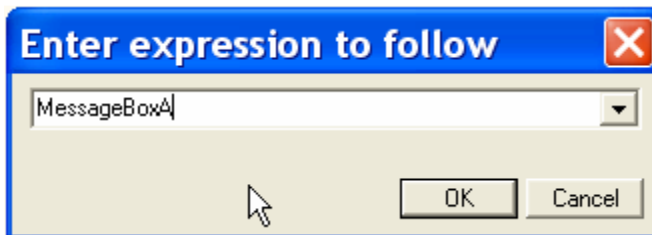


USUARIOS DE WINDOWS 98 no hace esto ya verán porque



Allí vemos que pertenece a la dll llamada USER32.dll y que no es mas que una rutina que termina en un RET, lo único que nos salva esto es de tener que agregar toda esta rutina en nuestro programa, así se hacen programas mas pequeños y se nos facilita la vida a los crackers jeje.

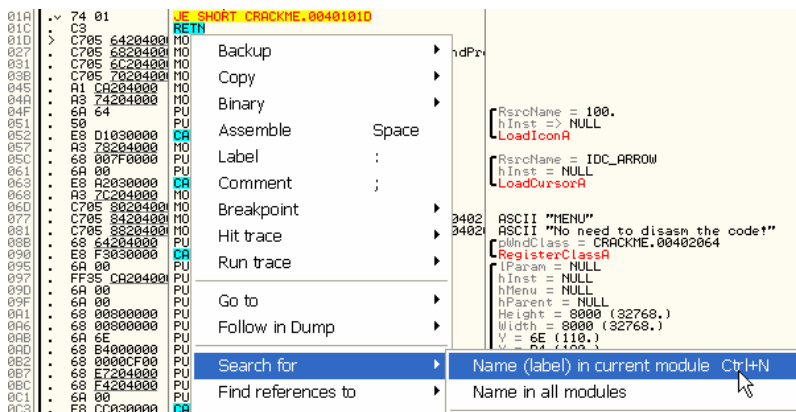
Si apreto la tecla MENOS volveré al punto donde estaba anteriormente y puedo verificar también que si hago GOTO EXPRESIÓN y directamente tipo MessageBoxA en dicha ventana nos lleva al mismo lugar,



Allí vemos la rutina de la api MessageBoxA correspondiente a USER32.dll

Como verán tuve que tipear MessageBoxA y no messageboxa ya que el nombre de la api es reconocido por OLLYDBG si tipeamos en forma correcta sus mayúsculas y minúsculas, y la pregunta siguiente es pensar como se como se escribe una api cuales son mayúsculas y minúsculas en su nombre, pues es sencillo.

Volvamos con MENOS al ENTRY POINT



Haciendo en el listado CLICK DERECHO-SEARCH FOR NAME (label) in CURRENT MODULE nos dará la lista de Apis o NAMES utilizadas por este CRACKME DE CRUEHEAD.

| Address | Section | Type | Name | Comment |
|----------|---------|--------|------------------------------|---------|
| 004031E4 | .idata | Import | USER32.BeginPaint | |
| 00403230 | .idata | Import | KERNEL32.CloseHandle | |
| 00403250 | .idata | Import | COMCTL32.CreateToolBar | |
| 0040324C | .idata | Import | COMCTL32.CreateToolBarEx | |
| 004031E8 | .idata | Import | USER32.CreateWindowExA | |
| 004031EC | .idata | Import | USER32.DefWindowProcA | |
| 00403278 | .idata | Import | GDI32.DeleteDC | |
| 00403274 | .idata | Import | GDI32.DeleteObject | |
| 004031F0 | .idata | Import | USER32.DialogBoxParamA | |
| 004031F4 | .idata | Import | USER32.DispatchMessageA | |
| 004031F8 | .idata | Import | USER32.DrawMenuBar | |
| 004031FC | .idata | Import | USER32.EndDialog | |
| 00403270 | .idata | Import | GDI32.EndDoc | |
| 0040326C | .idata | Import | GDI32.EndPage | |
| 00403200 | .idata | Import | USER32.EndPaint | |
| 00403240 | .idata | Import | KERNEL32.ExitProcess | |
| 00403204 | .idata | Import | USER32.FindWindowA | |
| 00403208 | .idata | Import | USER32.GetDC | |
| 0040320C | .idata | Import | USER32.GetDlgItem | |
| 00403210 | .idata | Import | USER32.GetDlgItemTextA | |
| 0040321C | .idata | Import | KERNEL32.GetLocalTime | |
| 00403214 | .idata | Import | USER32.GetMessageA | |
| 00403238 | .idata | Import | KERNEL32.GetModuleHandleA | |
| 00403284 | .idata | Import | COMDLG32.GetOpenFileNameA | |
| 00403280 | .idata | Import | COMDLG32.GetSaveFileNameA | |
| 00403268 | .idata | Import | GDI32.GetStockObject | |
| 00403188 | .idata | Import | USER32.GetSystemMetrics | |
| 00403264 | .idata | Import | GDI32.GetTextMetricsA | |
| 00403198 | .idata | Import | USER32.GetWindowRect | |
| 00403228 | .idata | Import | KERNEL32.GlobalAlloc | |
| 00403224 | .idata | Import | KERNEL32.GlobalFree | |
| 00403248 | .idata | Import | COMCTL32.InitCommonControls | |
| 004031B8 | .idata | Import | USER32.InvalidRect | |
| 00403184 | .idata | Import | USER32.KillTimer | |
| 00403190 | .idata | Import | USER32.LoadAcceleratorsA | |
| 004031A4 | .idata | Import | USER32.LoadBitmapA | |
| 0040318C | .idata | Import | USER32.LoadCursorA | |
| 004031A0 | .idata | Import | USER32.LoadIconA | |
| 004031C8 | .idata | Import | USER32.LoadMenuA | |
| 0040319C | .idata | Import | USER32.LoadStringA | |
| 0040322C | .idata | Import | KERNEL32.lstrlen | |
| 00403194 | .idata | Import | USER32.MessageBeep | |
| 004031AC | .idata | Import | USER32.MessageBoxA | |
| 00401000 | CODE | Export | <ModuleEntryPoint> | |
| 004031C0 | .idata | Import | USER32.MoveWindow | |
| 00403220 | .idata | Import | KERNEL32.OpenFile | |
| 004031B0 | .idata | Import | USER32.PostQuitMessage | |
| 00403288 | .idata | Import | COMDLG32.PrintDlgA | |
| 0040323C | .idata | Import | KERNEL32.ReadFile | |
| 004031E0 | .idata | Import | USER32.RegisterClassA | |
| 004031D0 | .idata | Import | USER32.SendMessageA | |
| 004031A8 | .idata | Import | USER32.SetFocus | |
| 004031D4 | .idata | Import | USER32.SetTimer | |
| 004031D8 | .idata | Import | USER32.SetWindowPos | |
| 004031CC | .idata | Import | USER32.ShowWindow | |
| 00403260 | .idata | Import | GDI32.StartDocA | |
| 0040325C | .idata | Import | GDI32.StartPage | |
| 00403258 | .idata | Import | GDI32.TextOutA | |
| 004031BC | .idata | Import | USER32.TranslateAcceleratorA | |
| 004031C4 | .idata | Import | USER32.TranslateMessage | |
| 004031DC | .idata | Import | USER32.UpdateWindow | |
| 004031B4 | .idata | Import | USER32.WinHelpA | |
| 00401128 | CODE | Export | WndProc | |
| 00403234 | .idata | Import | KERNEL32.WriteFile | |

Para encontrar la que quiero no necesito bajar y buscarla a mano solo en esta ventana apretando la M

| | | | | |
|----------|--------|--------|------------------------|--|
| 004031A4 | .idata | Import | USER32.LoadBitmapA | |
| 0040318C | .idata | Import | USER32.LoadCursorA | |
| 004031A0 | .idata | Import | USER32.LoadIconA | |
| 004031C8 | .idata | Import | USER32.LoadMenuA | |
| 0040319C | .idata | Import | USER32.LoadStringA | |
| 0040322C | .idata | Import | KERNEL32.lstrlen | |
| 00403194 | .idata | Import | USER32.MessageBeep | |
| 004031AC | .idata | Import | USER32.MessageBoxA | |
| 00401000 | CODE | Export | <ModuleEntryPoint> | |
| 004031C0 | .idata | Import | USER32.MoveWindow | |
| 00403220 | .idata | Import | KERNEL32.OpenFile | |
| 004031B0 | .idata | Import | USER32.PostQuitMessage | |
| 00403288 | .idata | Import | COMDLG32.PrintDlgA | |
| 0040323C | .idata | Import | KERNEL32.ReadFile | |
| 004031A8 | .idata | Import | USER32.RegisterClassA | |

Vemos que el cursor se acomoda en la primera api que empieza con M si a continuación sigo tipeando letras del nombre de la api,



Arriba nos va marcando las letras que tipeamos y el cursor halla la api

| | | | |
|----------|--------|--------|--------------------|
| 0040319C | .idata | Import | USER32.LoadStringH |
| 0040322C | .idata | Import | KERNEL32.lstrlen |
| 00403194 | .idata | Import | USER32.MessageBeep |
| 004031AC | .idata | Import | USER32.MessageBoxA |
| 00401000 | CODE | Export | <ModuleEntryPoint> |
| 004031C0 | .idata | Import | USER32.MoveWindow |
| 00403220 | .idata | Import | KERNEL32.OpenFile |

Si hago click derecho encima del nombre de la api tengo diferentes opciones

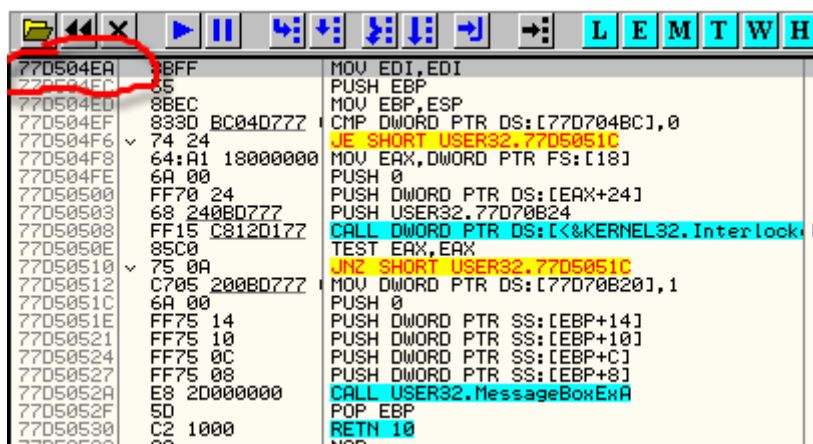
| | | | |
|----------|--------|--------|------------------------|
| 004031H0 | .idata | Import | USER32.LoadIconH |
| 004031C8 | .idata | Import | USER32.LoadMenuA |
| 0040319C | .idata | Import | USER32.LoadStringA |
| 0040322C | .idata | Import | KERNEL32.lstrlen |
| 00403194 | .idata | Import | USER32.MessageBeep |
| 004031AC | .idata | Import | USER32.MessageBoxA |
| 00401000 | CODE | Export | <ModuleEntryPoint> |
| 004031C0 | .idata | Import | USER32.MoveWindow |
| 00403220 | .idata | Import | KERNEL32.OpenFile |
| 004031B0 | .idata | Import | USER32.PostQuitMessage |
| 00403288 | .idata | Import | COMDLG32.PrintDlgA |
| 0040323C | .idata | Import | KERNEL32.ReadFile |
| 004031E4 | .idata | Import | USER32.BeginPaint |

Actualize

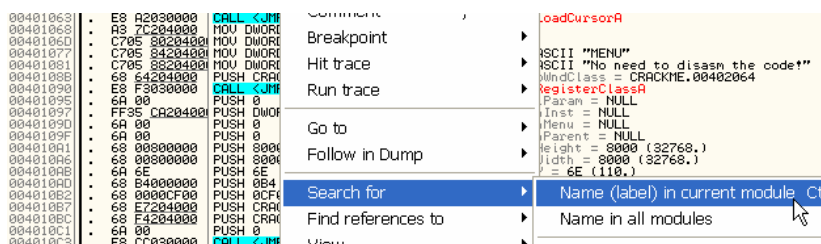
Follow import in Disassembler

Follow in Dump

Si elijo FOLLOW IMPORT IN DISSASSEMBLER nos llevara a la dirección de la api, este es otro método para llegar a la dirección, si no tenemos ganas de tipear en la comandbar.

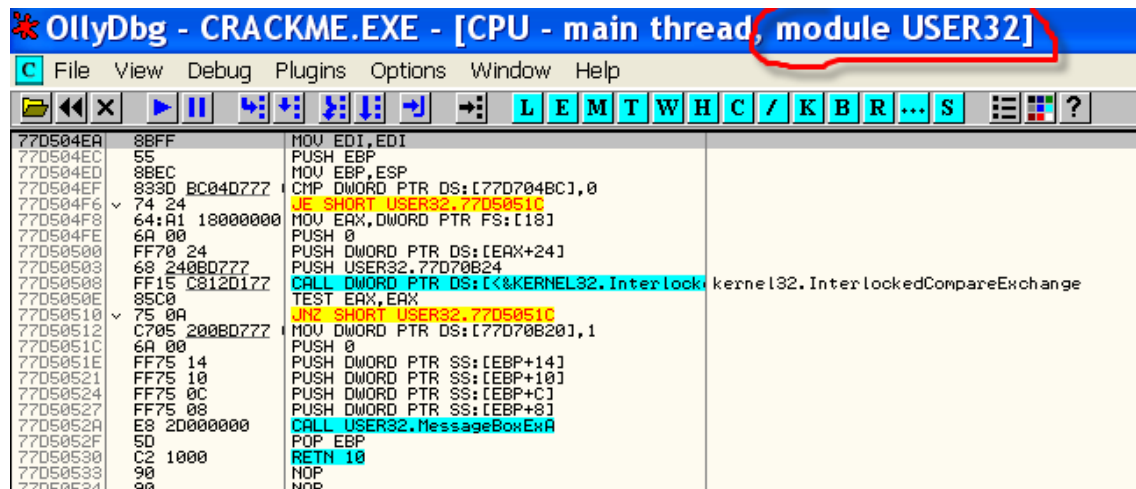


Aquí vemos un común error que cometen muchos que recién se inician, si allí en la api hago SEARCH FOR NAME (LABEL) IN CURRENT MODULE



OLLYDBG buscara en este caso las APIS o NAMES correspondientes a USER32.dll ya que allí lo especifica busca en el MODULO que esta visible en el listado, CURRENT MODULE y el

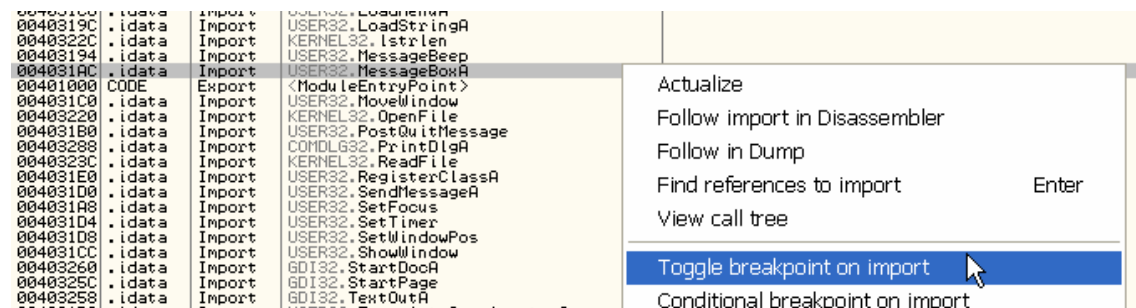
que esta visible cuando estamos en la api es USER32.dll lo podemos ver en el mismo OLLYDBG arriba



Aun cuando nosotros no estemos ejecutando en este momento la api y solo mirando al buscar NAMES saldrán las de USER32.dll que no son las que en este momento nos interesan, por lo tanto si queremos volver a ver las apis del Crackme de Crucehead debemos apretar MENOS hasta volver a ver el listado del CRACKME por ejemplo en el ENTRY POINT y allí si, si hacemos SEARCH FOR NAMES nos mostrara las apis del mismo.

[PARA WINDOWS NT/2000, XP o 2003](#)

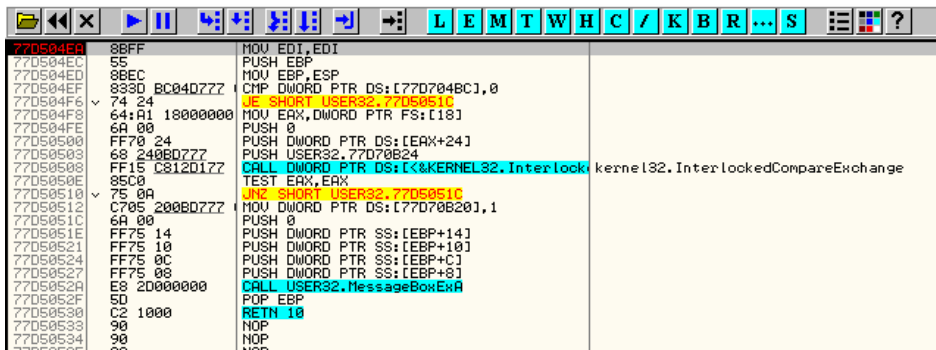
A PARTIR DE AQUÍ EL TUTORIAL CONTINUA PARA SISTEMAS NT/2000 y XP, les recomiendo a los que tienen WINDOWS 95 o 98 pasarse a cualquiera de esos sistemas, que allí es donde OLLYDBG es mas potente, pero si no pueden hacerlo salteen esta parte y vayan a donde dice [APENDICE PARA WINDOWS 98](#)



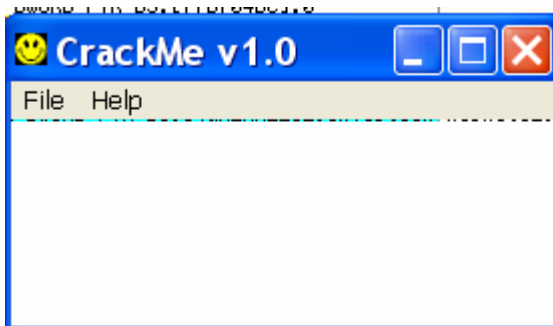
Vemos que cuando estoy en la lista de apis del crackme otra opción es poner un BREAKPOINT en dicha api, así cuando el programa llama a la misma parara, hagámoslo con CLICK DERECHO-TOGGLE BREAKPOINT ON IMPORT.

También podríamos hacerlo directamente en la commandbar tipeando

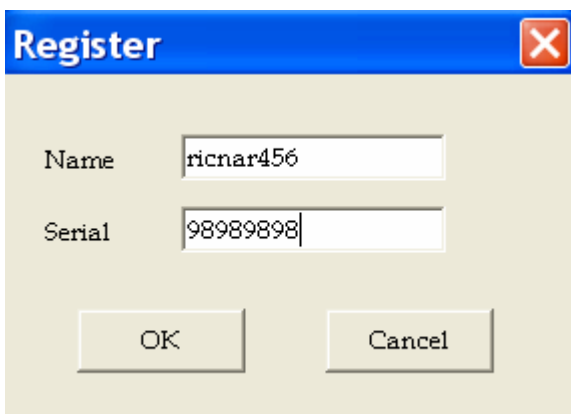
Bp MessageBoxA



Si ahora vemos la direccion de la api apreciamos que se ha puesto un BREAKPOINT en la direcci3n de inicio de la misma, de esta forma si la api es usada parara OLLYDBG en ella, apretemos F9 para correr el CRACKME DE CRUEHEAD y ver si para alli.



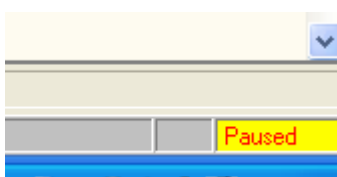
Vemos que aparece la ventanita y aun no paro vayamos a HELP-REGISTER



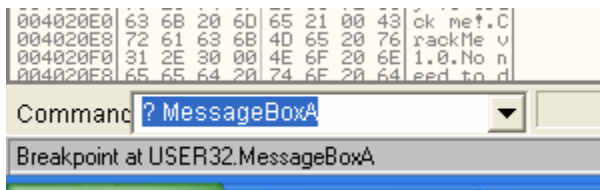
En la ventana tipeemos alg3n nombre y serial falso y apretemos OK

Vemos que el OLLYDBG paro veamos porque

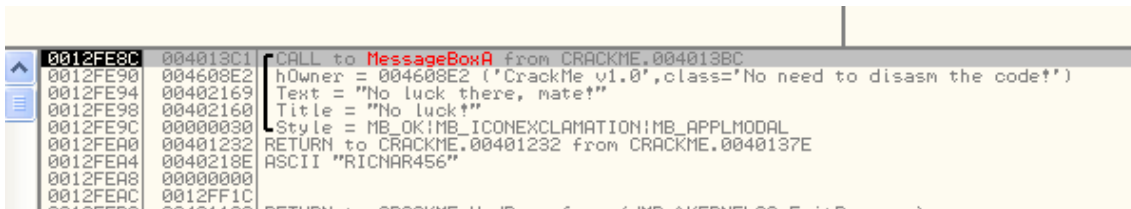
En la esquina inferior derecha vemos que esta PAUSADO



Y en la esquina inferior izquierda siempre nos muestra el motivo porque paro.

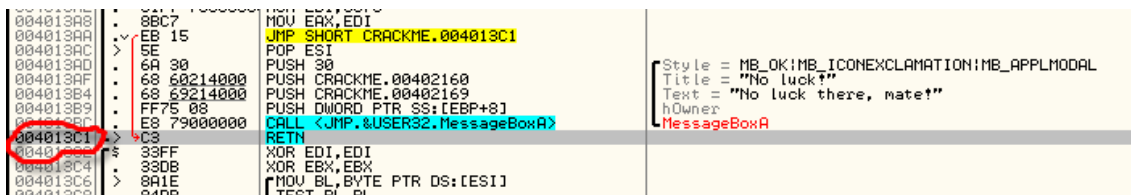


Allí dice BREAKPOINT en USER32.MessageBoxA o sea que paro en nuestro BREAKPOINT en la API.



Vemos que OLLYDBG nos muestra información ya que cada api se llama con determinados parámetros que se pasan al stack antes de llamarla, y en este caso vemos

En la primera línea la dirección de retorno del CALL que nos hizo llegar aquí en este caso 4013C1



Como vimos cuando explicamos CALL Y RET siempre se pasa al stack la dirección de retorno del mismo, y allí esta cuando llegas al RET de la api volverá a 4013c1.

Luego abajo los parámetros de la misma nos muestran entre otras cosas ya que la api MessageBoxA es la encargada de mostrarnos los típicos cartelitos de mensajes de Windows, el título del mensaje, el texto, el estilo etc.

Ya vemos que el TEXTO es NO LUCK THERE, MATE que es el cartel que coloca el Crackme de Cruehead cuando el serial que introdujiste no es correcto.

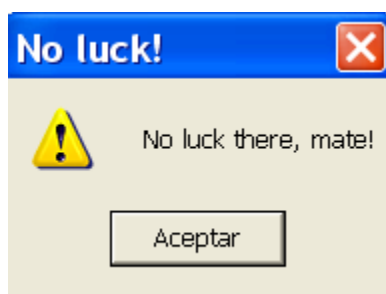
Pues allí estamos y el crackme esta a punto de mostrarnos el fatídico cartelito.

| Address | Disassembly | Comment |
|----------|----------------|---|
| 77D504E9 | 8BFF | MOV EDI,EDI |
| 77D504EC | 55 | PUSH EBP |
| 77D504ED | 8BEC | MOV EBP,ESP |
| 77D504EF | 833D BC04D777 | CMP DWORD PTR DS:[77D704BC],0 |
| 77D504F6 | 74 24 | JE SHORT USER32.77D50510 |
| 77D504F8 | 64:A1 18000000 | MOV EAX,DWORD PTR FS:[18] |
| 77D504FE | 6A 00 | PUSH 0 |
| 77D50500 | FF70 24 | PUSH DWORD PTR DS:[EAX+24] |
| 77D50503 | 68 240BD777 | PUSH USER32.77D70B24 |
| 77D50508 | FF15 C812D177 | CALL DWORD PTR DS:[<&KERNEL32.Interlock kernel32.InterlockedCompareExchange |
| 77D5050E | 85C0 | TEST EAX,EAX |
| 77D50510 | 75 0A | JNZ SHORT USER32.77D50510 |
| 77D50512 | C705 200BD777 | MOV DWORD PTR DS:[77D70B20],1 |
| 77D5051C | 6A 00 | PUSH 0 |
| 77D5051E | FF75 14 | PUSH DWORD PTR SS:[EBP+14] |
| 77D50521 | FF75 10 | PUSH DWORD PTR SS:[EBP+10] |
| 77D50524 | FF75 0C | PUSH DWORD PTR SS:[EBP+C] |
| 77D50527 | FF75 08 | PUSH DWORD PTR SS:[EBP+8] |
| 77D5052A | E8 2D000000 | CALL USER32.MessageBoxExA |
| 77D5052F | 5D | POP EBP |
| 77D50530 | C2 1000 | RETN 10 |
| 77D50533 | 90 | NOP |
| 77D50534 | 90 | NOP |

Para que vean que no le miento y que la api es la encargada de hacerlo, pongamos un BREAKPOINT en el RET 10 que es el final de la api, allí lo vemos debajo, en su maquina puede cambiar pero siempre es el primer RET a partir de la dirección de inicio de la api que vemos al ir bajando.

| Address | Disassembly | Comment |
|----------|----------------|---|
| 77D504E9 | 8BFF | MOV EDI,EDI |
| 77D504EC | 55 | PUSH EBP |
| 77D504ED | 8BEC | MOV EBP,ESP |
| 77D504EF | 833D BC04D777 | CMP DWORD PTR DS:[77D704BC],0 |
| 77D504F6 | 74 24 | JE SHORT USER32.77D50510 |
| 77D504F8 | 64:A1 18000000 | MOV EAX,DWORD PTR FS:[18] |
| 77D504FE | 6A 00 | PUSH 0 |
| 77D50500 | FF70 24 | PUSH DWORD PTR DS:[EAX+24] |
| 77D50503 | 68 240BD777 | PUSH USER32.77D70B24 |
| 77D50508 | FF15 C812D177 | CALL DWORD PTR DS:[<&KERNEL32.Interlock kernel32.Interlocke |
| 77D5050E | 85C0 | TEST EAX,EAX |
| 77D50510 | 75 0A | JNZ SHORT USER32.77D50510 |
| 77D50512 | C705 200BD777 | MOV DWORD PTR DS:[77D70B20],1 |
| 77D5051C | 6A 00 | PUSH 0 |
| 77D5051E | FF75 14 | PUSH DWORD PTR SS:[EBP+14] |
| 77D50521 | FF75 10 | PUSH DWORD PTR SS:[EBP+10] |
| 77D50524 | FF75 0C | PUSH DWORD PTR SS:[EBP+C] |
| 77D50527 | FF75 08 | PUSH DWORD PTR SS:[EBP+8] |
| 77D5052A | E8 2D000000 | CALL USER32.MessageBoxExA |
| 77D5052F | 5D | POP EBP |
| 77D50530 | C2 1000 | RETN 10 |
| 77D50533 | 90 | NOP |
| 77D50534 | 90 | NOP |
| 77D50535 | 90 | NOP |

Allí tenemos la api cercada apretemos F9 o RUN.



Allí vemos el típico cartelito de la MessageBoxA y como nos aviso OLLYDBG el titulo es NO LUCK! Y el texto NO LUCK THERE MATE! o sea que no tuvimos suerte con nuestro serial.

Por supuesto al aceptar para en el RET de la api

| | | | |
|----------|---------------|--|-------------------|
| 77D504E9 | 8BFF | MOV EDI,EDI | |
| 77D504EC | 55 | PUSH EBP | |
| 77D504ED | 8BEC | MOV EBP,ESP | |
| 77D504EF | 833D BC04D777 | CMP DWORD PTR DS:[77D704BC],0 | |
| 77D504F0 | 74 24 | JC SHORT USER32.77D50510 | |
| 77D504F3 | 64A1 18000000 | MOV EAX,DWORD PTR FS:[18] | |
| 77D504FE | 6A 00 | PUSH 0 | |
| 77D50500 | FF70 24 | PUSH DWORD PTR DS:[EAX+24] | |
| 77D50503 | 68 240BD777 | PUSH USER32.77D70B24 | |
| 77D50508 | FF15 C812D177 | CALL DWORD PTR DS:[<KERNEL32.Interlock | kernel32.Interloc |
| 77D5050E | 85C0 | TEST EAX,EAX | |
| 77D50510 | 75 0A | JNZ SHORT USER32.77D5051C | |
| 77D50512 | C705 200BD777 | MOV DWORD PTR DS:[77D70B20],1 | |
| 77D5051C | 6A 00 | PUSH 0 | |
| 77D5051E | FF75 14 | PUSH DWORD PTR SS:[EBP+14] | |
| 77D50521 | FF75 10 | PUSH DWORD PTR SS:[EBP+10] | |
| 77D50524 | FF75 0C | PUSH DWORD PTR SS:[EBP+0C] | |
| 77D50527 | FF75 08 | PUSH DWORD PTR SS:[EBP+08] | |
| 77D5052A | E8 2D000000 | CALL USER32.MessageBoxExA | |
| 77D5052F | 5D | POP EBP | |
| 77D50530 | C2 1000 | RETN 10 | |
| 77D50533 | 90 | NOP | |
| 77D50534 | 90 | NOP | |
| 77D50535 | 90 | NOP | |
| 77D50536 | 90 | NOP | |

Como vimos el proceso de aparición del cartel, ocurrió entre el inicio y final de la api, allí estamos en el RETN10.

Ya que no aclaramos la diferencia entre en RETN 10 y el RET común lo haremos aquí en este caso si el RET fuera común, al ejecutarlo, volvería a la dirección de retorno 4013C1

| | | |
|----------|----------|--|
| 0012FE8C | 004013C1 | RETURN to CRACKME.004013C1 from <JMP.&USER32.MessageBoxA> |
| 0012FE90 | 01E6079A | ASCII "No luck there, mate!" |
| 0012FE93 | 00402169 | ASCII "No luck!" |
| 0012FE98 | 00402160 | ASCII "No luck!" |
| 0012FE9C | 00000030 | |
| 0012FEA0 | 00401232 | RETURN to CRACKME.00401232 from CRACKME.0040137E |
| 0012FEA4 | 0040218E | ASCII "RICHAR456" |
| 0012FEA8 | 00000000 | |
| 0012FEAC | 0012FF1C | |
| 0012FEB0 | 00401128 | RETURN to CRACKME.WndProc from <JMP.&KERNEL32.ExitProcess> |
| 0012FEB4 | 0012FEE0 | |
| 0012FEB8 | 77D18734 | RETURN to USER32.77D18734 |
| 0012FEBD | 01E6079A | |
| 0012FEC0 | 00000111 | |
| 0012FEC4 | 00000066 | |
| 0012FEC8 | 00000000 | |
| 0012FECB | 00401128 | RETURN to CRACKME.WndProc from <JMP.&KERNEL32.ExitProcess> |

Y al quitar el valor de la dirección de retorno de arriba del stack, este quedaría en mi caso en 12Fe90, en el caso del RETN10 vuelve a la misma dirección 4013C1, pero a ESP se le suma 10 con lo cual el stack debería quedar en 12fe90 mas 10 seria esp=12fea0 veamos apretemos F7.

[RETORNO DEL APÉNDICE PARA WINDOWS 98 desde aquí sigue para todos los SO.](#)

| | | | |
|----------|-------------|--------------------------------|--|
| 004013A8 | 8BC7 | MOV EAX,EDI | |
| 004013AA | EB 15 | JMP SHORT CRACKME.004013C1 | |
| 004013AC | 5E | POP ESI | |
| 004013AD | 6A 30 | PUSH 30 | |
| 004013AF | 68 20214000 | PUSH CRACKME.00402160 | |
| 004013B4 | 68 20214000 | PUSH CRACKME.00402160 | |
| 004013B9 | FF75 08 | PUSH DWORD PTR SS:[EBP+8] | |
| 004013BC | E8 79000000 | CALL <JMP.&USER32.MessageBoxA> | |
| 004013C1 | C3 | RETN | |
| 004013C2 | 33FF | XOR EDI,EDI | |
| 004013C4 | 33DB | XOR EBX,EBX | |
| 004013C6 | 8A1E | MOV BL,BYTE PTR DS:[ESI] | |
| 004013C8 | 84D0 | TEST BL,BL | |

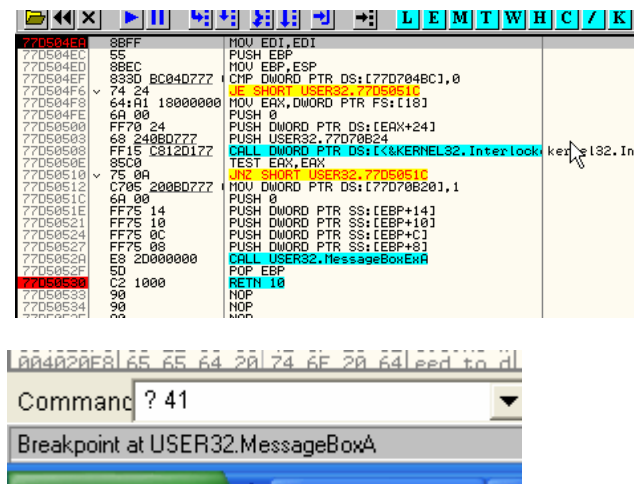
Style = MB_OK|MB_ICONEXCLAMATION|MB_APPLMODAL
 Title = "No luck?"
 Text = "No luck there, mate!"
 hOwner
 MessageBoxA

Allí retornamos de la api al crackme y vemos que en el stack se cumplió lo que mostramos el RETN 10 le suma a ESP 10 mas de lo que valdría si retorna como RET solo.

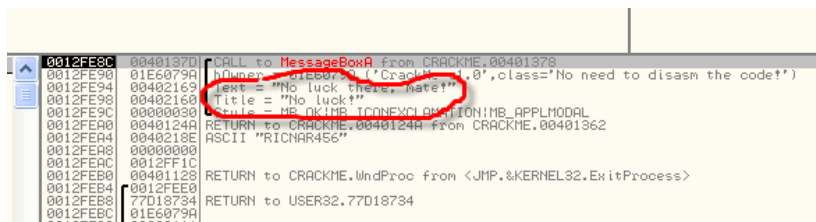
| | | |
|----------|----------|---|
| 0012FE90 | 00401232 | RETURN to CRACKME.00401232 from CRACKME.0040137E |
| 0012FE93 | 0040218E | ASCII "RICHAR456" |
| 0012FE98 | 00000000 | |
| 0012FEAC | 0012FF1C | |
| 0012FEB0 | 00401128 | RETURN to CRACKME.WndProc from <JMP.&KERNEL32.ExitProce |
| 0012FEB4 | 0012FEE0 | |
| 0012FEB8 | 77D18734 | RETURN to USER32.77D18734 |
| 0012FEBD | 01E6079A | |
| 0012FEC0 | 00000111 | |
| 0012FEC4 | 00000066 | |
| 0012FEC8 | 00000000 | |
| 0012FECB | 00401128 | RETURN to CRACKME.WndProc from <JMP.&KERNEL32.ExitProce |

La cuestión es que la decisión ya fue tomada y nosotros estamos en el horno, ya nos dijo mala suerte amigo, el serial que tipeaste no sirve.

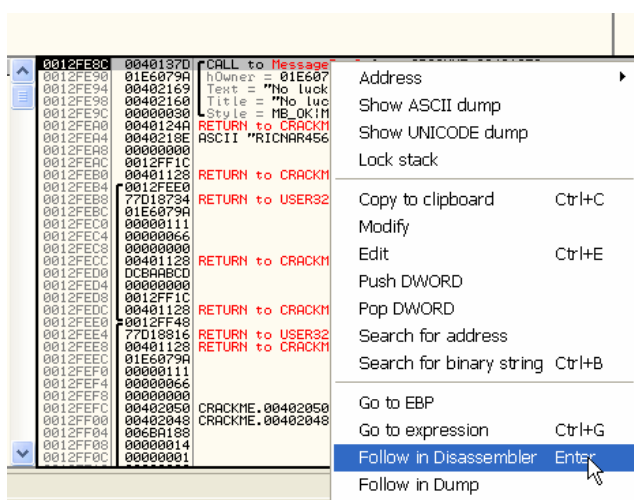
Apretemos F9 nuevamente



Vemos que para nuevamente en la api, y que por si no me di cuenta me va a decir que no tuve suerte jeje



Vemos que en este caso la dirección de retorno es 40137d veamos que hay allí con GOTO EXPRESIÓN 40137D en el listado o bien en la primera línea del stack CLICK DERECHO-FOLLOW IN DISASSEMBLER



Allí vemos que volvería a 40137D y que estamos dentro del call que llama a MessageBoxA que esta justo arriba en 401378.

| | | | |
|----------|-------------|--------------------------------|-------------|
| 0040133B | E8 73010000 | CALL <JMP.&USER32.EndDialog> | EndDialog |
| 0040133C | B8 01000000 | MOV EAX,1 | |
| 00401344 | EB DF | JMP SHORT CRACKME.00401325 | |
| 00401346 | B8 00000000 | MOV EAX,0 | |
| 00401348 | EB D8 | JMP SHORT CRACKME.00401325 | |
| 0040134D | 6A 30 | PUSH 30 | |
| 0040134F | 68 23214000 | PUSH CRACKME.00402129 | |
| 00401354 | 68 34214000 | PUSH CRACKME.00402134 | |
| 00401359 | FF75 08 | PUSH DWORD PTR SS:[EBP+8] | |
| 0040135C | E8 D9000000 | CALL <JMP.&USER32.MessageBoxA> | MessageBoxA |
| 00401361 | C3 | RETN | |
| 00401362 | 6A 00 | PUSH 0 | |
| 00401364 | E8 AD000000 | CALL <JMP.&USER32.MessageBeep> | MessageBeep |
| 00401369 | 6A 30 | PUSH 30 | |
| 0040136B | 68 60214000 | PUSH CRACKME.00402160 | |
| 00401370 | 68 69214000 | PUSH CRACKME.00402169 | |
| 00401375 | FF75 08 | PUSH DWORD PTR SS:[EBP+8] | |
| 00401378 | E8 BD000000 | CALL <JMP.&USER32.MessageBoxA> | MessageBoxA |
| 0040137D | C3 | RETN | |
| 0040137E | 8B7424 04 | MOV ESI,DWORD PTR SS:[ESP+4] | |
| 00401382 | 56 | PUSH ESI | |
| 00401383 | 8A06 | MOV AL,BYTE PTR DS:[ESI] | |
| 00401385 | 84C0 | TEST AL,AL | |

Vemos arriba que hay otro MessageBoxA pero con el mensaje de felicitación de que acertamos GREAT WORK, jeje si pudiéramos llegar allí en vez de al cartel de que no tuvimos suerte seria un primer gran paso.

| | | | |
|----------|-------------|--------------------------------|-------------|
| 0040133B | E8 73010000 | CALL <JMP.&USER32.EndDialog> | EndDialog |
| 0040133C | B8 01000000 | MOV EAX,1 | |
| 00401344 | EB DF | JMP SHORT CRACKME.00401325 | |
| 00401346 | B8 00000000 | MOV EAX,0 | |
| 00401348 | EB D8 | JMP SHORT CRACKME.00401325 | |
| 0040134D | 6A 30 | PUSH 30 | |
| 0040134F | 68 23214000 | PUSH CRACKME.00402129 | |
| 00401354 | 68 34214000 | PUSH CRACKME.00402134 | |
| 00401359 | FF75 08 | PUSH DWORD PTR SS:[EBP+8] | |
| 0040135C | E8 D9000000 | CALL <JMP.&USER32.MessageBoxA> | MessageBoxA |
| 00401361 | C3 | RETN | |
| 00401362 | 6A 00 | PUSH 0 | |
| 00401364 | E8 AD000000 | CALL <JMP.&USER32.MessageBeep> | MessageBeep |
| 00401369 | 6A 30 | PUSH 30 | |
| 0040136B | 68 60214000 | PUSH CRACKME.00402160 | |
| 00401370 | 68 69214000 | PUSH CRACKME.00402169 | |
| 00401375 | FF75 08 | PUSH DWORD PTR SS:[EBP+8] | |
| 00401378 | E8 BD000000 | CALL <JMP.&USER32.MessageBoxA> | MessageBoxA |
| 0040137D | C3 | RETN | |
| 0040137E | 8B7424 04 | MOV ESI,DWORD PTR SS:[ESP+4] | |
| 00401382 | 56 | PUSH ESI | |
| 00401383 | 8A06 | MOV AL,BYTE PTR DS:[ESI] | |
| 00401385 | 84C0 | TEST AL,AL | |

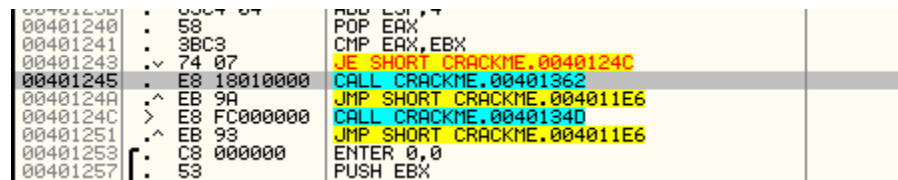
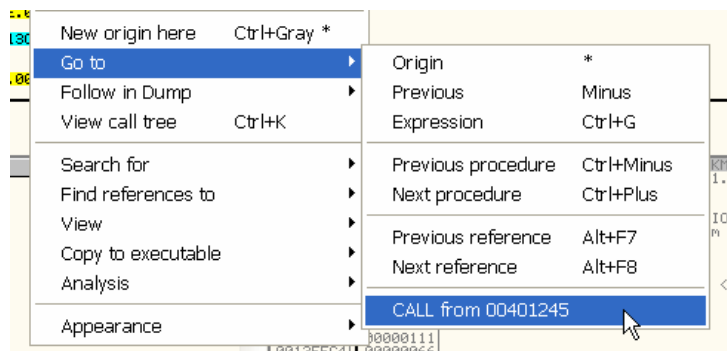
Vemos que OLLYDBG en su análisis inicial nos muestra unos corchetes, que significa que eso es una rutina que empieza y termina allí, vemos que hay dos, una para el cartel de NO LUCK que empieza en 401362 y otra para el GREAT WORK que empieza en 40134D.

Si vamos a 401362 que es el inicio de la rutina donde estamos (aun dentro del MessageBoxA) y hago click allí el OLLYDBG me muestra en las aclaraciones.

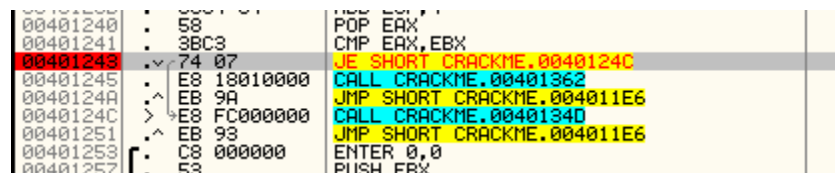
| | | | |
|----------|---------------|--------------------------------|--|
| 0040135C | E8 D9000000 | CALL <JMP.&USER32.MessageBoxA> | |
| 00401361 | C3 | RETN | |
| 00401362 | 6A 00 | PUSH 0 | |
| 00401364 | E8 AD000000 | CALL <JMP.&USER32.MessageBeep> | |
| 00401369 | 6A 30 | PUSH 30 | |
| 0040136B | 68 60214000 | PUSH CRACKME.00402160 | |
| 00401370 | 68 69214000 | PUSH CRACKME.00402169 | |
| 00401375 | FF75 08 | PUSH DWORD PTR SS:[EBP+8] | |
| 00401378 | E8 BD000000 | CALL <JMP.&USER32.MessageBoxA> | |
| 0040137D | C3 | RETN | |
| 0040137E | 8B7424 04 | MOV ESI,DWORD PTR SS:[ESP+4] | |
| 00401382 | 56 | PUSH ESI | |
| 00401383 | 8A06 | MOV AL,BYTE PTR DS:[ESI] | |
| 00401385 | 84C0 | TEST AL,AL | |
| 00401387 | 74 13 | JE SHORT CRACKME.0040139C | |
| 00401389 | 3C 41 | CMP AL,41 | |
| 0040138B | 72 1F | JB SHORT CRACKME.004013AC | |
| 0040138D | 3C 5A | CMP AL,5A | |
| 0040138F | 73 03 | JNB SHORT CRACKME.00401394 | |
| 00401391 | 46 | INC ESI | |
| 00401392 | EB EF | JMP SHORT CRACKME.00401383 | |
| 00401394 | E8 39000000 | CALL CRACKME.004013D2 | |
| 00401399 | 46 | INC ESI | |
| 0040139A | EB E7 | JMP SHORT CRACKME.00401383 | |
| 0040139C | 5E | POP ESI | |
| 0040139D | E8 20000000 | CALL CRACKME.004013C2 | |
| 004013A2 | 31F7 78560000 | XOR EDI,5678 | |
| 004013A8 | 8BC7 | MOV EAX,EDI | |
| 004013AD | EB 15 | JMP SHORT CRACKME.004013C1 | |
| 004013AC | 5E | POP ESI | |

Local call from 00401245

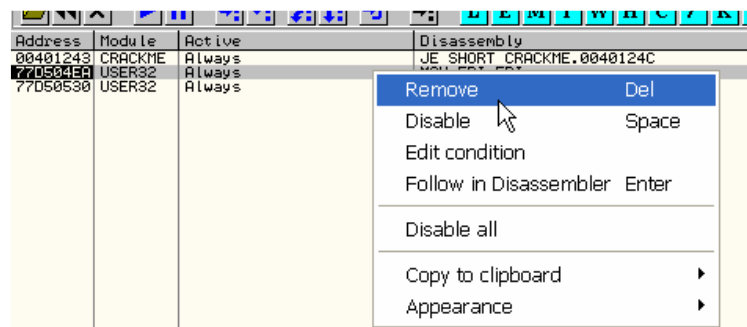
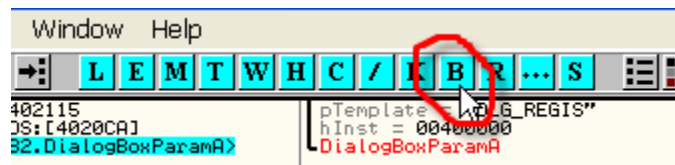
Quiere decir que OLLYDBG sabe que esa rutina ya que es DIRECTA es llamada desde 401245 veamos allí que hay CLICK DERECHO – GOTO CALL FROM 401245.



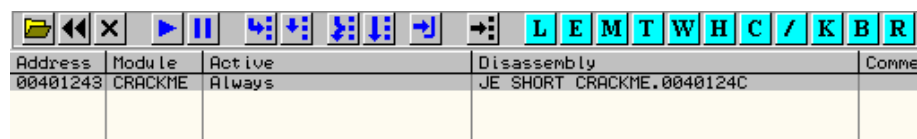
Hmm esta zona parece muy sospechosa hay una comparación y un salto y según el resultado de ese salto va al call de 401362 que es el que muestra NO LUCK y si no va al CALL 40134D que muestra GREAT WORK, esto no podemos perderlo pongamos un BREAKPOINT en dicho salto condicional.



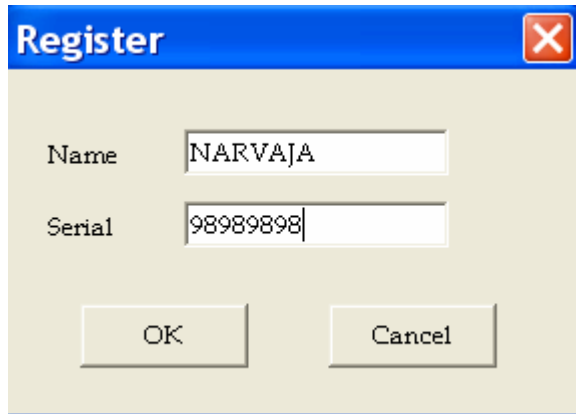
Y quitemos los breakpoints en la api MessageBoxA por ahora, eso puede hacerse en la ventana B de breakpoints



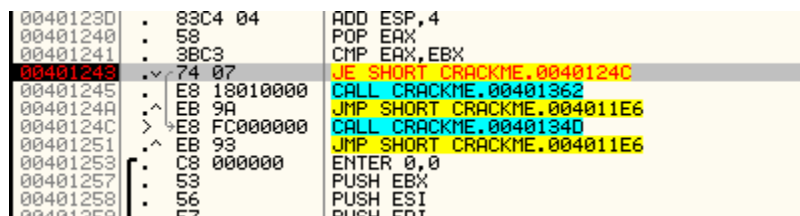
Quito CON CLICK DERECHO-REMOVE los dos BREAKPOINTS y deajo solo el de 401243 que es el salto condicional.



Ahora doy RUN con F9 acepto el NO LUCK que estábamos antes y vuelvo a ingresar a poner el nombre y serial en este caso pondré, usen el mismo que yo, ya verán porque

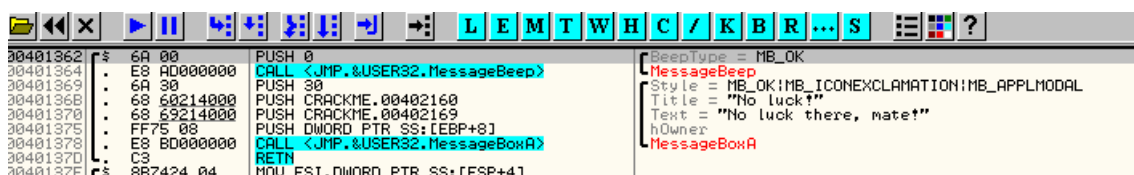


Apreto OK

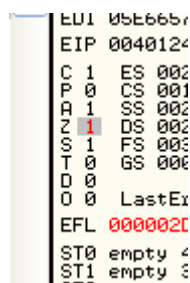


Vemos que de la comparación NO SALTARA al no ser EAX y EBX iguales y seguirá ejecutando

en 401245 que nos llevara a CALL 401362 que sabemos que allí esta el cartel malo, si no recuerdan, hagan click en 401245 y con CLICK DERECHO –FOLLOW pueden ver adonde iria



Allí vemos si ese salto condicional no salta pues, ira al cartel de que el serial es malo, que pasa si cambio el salto condicional, haciendo doble click en el flag Z



Allí cambie el FLAG Z a 1 que seria como si EAX y ECX en la comparación hubieran sido iguales y la resta de la comparación hubiera sido cero y hubiera activado así el flag Z, el salto JE salta si el FLAG Z es uno, así que ahora saltara veamos.

```

00401238 . E8 9B010000 CALL CRACKME.004013D8
0040123D . 83C4 04 ADD ESP,4
00401240 . 58 POP EAX
00401241 . 3BC3 CMP EAX,EBX
00401243 . 74 07 JE SHORT CRACKME.0040124C
00401245 . E8 18010000 CALL CRACKME.00401362
0040124A . EB 9A JMP SHORT CRACKME.004011E6
0040124C . E8 FC000000 CALL CRACKME.0040134D
00401251 . EB 93 JMP SHORT CRACKME.004011E6
00401253 . C8 000000 ENTER 0,0
00401257 . 53 PUSH EBX
00401258 . 56 PUSH ESI
00401259 . 57 PUSH EDI
0040125A . 817D 0C 1001 CMP DWORD PTR SS:[EBP+C1.110]

```

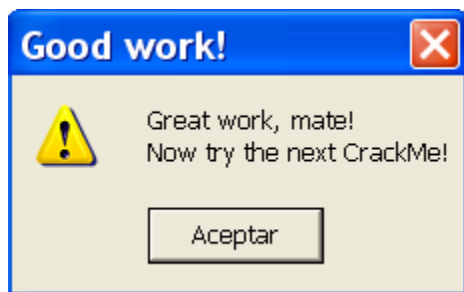
Allí lo cambiamos y si vemos en 40124c con FOLLOW vemos que ira a

```

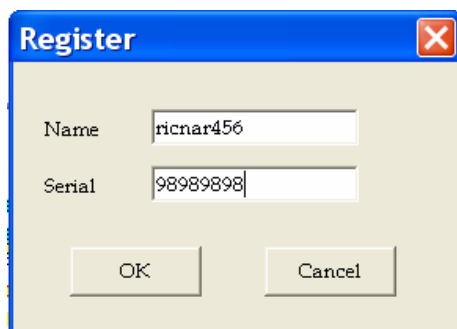
0040134D . 6A 30 PUSH 30
0040134F . 68 29214000 PUSH CRACKME.00402129
00401354 . 68 34214000 PUSH CRACKME.00402134
00401359 . FF75 08 PUSH DWORD PTR SS:[EBP+8]
0040135C . E8 D9000000 CALL <JMP.&USER32.MessageBoxA>
00401361 . C3 RETN

```

Jeje apretemos RUN o F9



O sea que esa comparación y ese salto condicional que invertimos es el punto de inflexión de la registración o validación del serial en este crackme, según si salta o no, pues sale el cartel bueno o malo, pero antes habíamos visto que había 2 carteles malos porque es eso, aquí no salio el primero, y eso es porque el crackme detecta el uso de números en el nombre (antes había puesto como nombre ricnar456) si es así te saca un primer cartel de NO LUCK, prueben nuevamente.



Al aceptar



Y recién al aceptar este primero, llega al salto condicional

| | | |
|----------|---------------|----------------------------|
| 00401238 | . E8 9B010000 | CALL CRACKME.00401308 |
| 0040123D | . 83C4 04 | ADD ESP,4 |
| 00401240 | . 58 | POP EAX |
| 00401241 | . 3BC3 | CMP EAX,EBX |
| 00401243 | . 74 07 | JE SHORT CRACKME.0040124C |
| 00401245 | . E8 18010000 | CALL CRACKME.00401362 |
| 0040124A | . EB 9A | JMP SHORT CRACKME.004011E6 |
| 0040124C | . E8 FC000000 | CALL CRACKME.00401340 |
| 00401251 | . EB 93 | JMP SHORT CRACKME.004011E6 |
| 00401253 | . C8 000000 | ENTER 0,0 |
| 00401257 | . 53 | PUSH EBX |
| 00401258 | . 56 | PUSH ESI |

Que muestra el mensaje definitivo.

Como podemos recordar la primera vez que paramos en la api MessageBoxA en este tute fue por el primer cartel ese

| | | |
|----------|----------|--|
| 0012FE8C | 004013C1 | RETURN to CRACKME.004013C1 from <JMP.&USER32.MessageBoxA> |
| 0012FE90 | 01E6079A | |
| 0012FE92 | 00402169 | ASCII "No luck there, mate!" |
| 0012FE98 | 00402160 | ASCII "No luck!" |
| 0012FE9C | 00000030 | |
| 0012FEA0 | 00401232 | RETURN to CRACKME.00401232 from CRACKME.0040137E |
| 0012FEA4 | 0040218E | ASCII "RICHAR456" |
| 0012FEA8 | 00000000 | |
| 0012FEAC | 0012FF1C | |
| 0012FEB0 | 00401128 | RETURN to CRACKME.WndProc from <JMP.&KERNEL32.ExitProcess> |
| 0012FEB4 | 0012FEE0 | |
| 0012FEB8 | 77D18734 | RETURN to USER32.77D18734 |
| 0012FEBD | 01E6079A | |
| 0012FEC0 | 00000111 | |
| 0012FEC4 | 00000066 | |
| 0012FEC8 | 00000000 | |
| 0012FECC | 00401128 | RETURN to CRACKME.WndProc from <JMP.&KERNEL32.ExitProcess> |

Y la dirección de retorno era 4013C1

| | | |
|----------|-----------------|--------------------------------|
| 00401370 | . C3 | RETN |
| 0040137D | . 8B7424 04 | MOV ESI,DWORD PTR SS:[ESP+4] |
| 00401382 | . 56 | PUSH ESI |
| 00401383 | . 8B06 | MOV AL,BYTE PTR DS:[ESI] |
| 00401385 | . 84C0 | TEST AL,AL |
| 00401387 | . 74 13 | JE SHORT CRACKME.0040139C |
| 00401389 | . 3C 41 | CMP AL,41 |
| 0040138B | . 72 1F | JB SHORT CRACKME.004013AC |
| 0040138D | . 3C 5A | CMP AL,5A |
| 0040138F | . 73 03 | JNB SHORT CRACKME.00401394 |
| 00401391 | . 46 | INC ESI |
| 00401392 | . EB EF | JMP SHORT CRACKME.00401383 |
| 00401394 | . E8 39000000 | CALL CRACKME.004013D2 |
| 00401399 | . 46 | INC ESI |
| 0040139A | . EB E7 | JMP SHORT CRACKME.00401383 |
| 0040139C | . 56 | POP ESI |
| 0040139D | . E8 20000000 | CALL CRACKME.004013C2 |
| 004013A2 | . 81F7 78560000 | XOR EDI,5678 |
| 004013A8 | . 8BC7 | MOV EAX,EDI |
| 004013AA | . EB 15 | JMP SHORT CRACKME.004013C1 |
| 004013AC | . 5E | POP ESI |
| 004013AD | . 6A 30 | PUSH 30 |
| 004013AF | . 68 60214000 | PUSH CRACKME.00402160 |
| 004013B4 | . 68 60214000 | PUSH CRACKME.00402169 |
| 004013B9 | . FF75 03 | PUSH DWORD PTR SS:[EBP+03] |
| 004013BC | . E8 79000000 | CALL <JMP.&USER32.MessageBoxA> |
| 004013C1 | . C3 | RETN |
| 004013C2 | . 33FF | XOR EDI,EDI |
| 004013C4 | . 33DB | XOR EBX,EBX |
| 004013C6 | . 8A1E | MOV BL,BYTE PTR DS:[ESI] |
| 004013CC | . 33FF | XOR EDI,EDI |

Style = MB_OK|MB_ICONEXCLAMATION|MB_APPLMODAL
 Title = "No luck!"
 Text = "No luck there, mate!"
 hOwner
 MessageBoxA

Allí vemos que el OLLYDBG en su análisis me muestra una rutina que comienza en 40137e y termina en 4013C1 justo donde retorna de la api.

Vemos también otra ayuda de OLLYDBG en 4013AC hay un > que significa que hay un salto que apunta hacia esa dirección si hacemos click allí nos aclarara mas.

```

00401387 74 13 JE SHORT CRACKME.00401390
00401388 3C 11 CMP AL,41
0040138B 72 1F JB SHORT CRACKME.00401390
0040138D 2C 50 CMP AL,50
0040138F 73 03 JNB SHORT CRACKME.00401394
00401391 46 INC ESI
00401392 EB EF JMP SHORT CRACKME.00401383
00401394 EB 39 CALL CRACKME.004013D2
00401399 46 INC ESI
0040139A EB E7 JMP SHORT CRACKME.00401383
0040139C 5E POP ESI
0040139D EB 20 CALL CRACKME.004013C2
004013A2 31F7 78560000 XOR EDI,5678
004013A8 8BC7 MOV EAX,EDI
004013AA EB 15 JMP SHORT CRACKME.004013C1
004013AC 5E POP ESI
004013AD 6A 30 PUSH 30
004013AF 68 68214000 PUSH CRACKME.00402168
004013B4 68 69214000 PUSH CRACKME.00402169
004013B9 FF75 08 PUSH DWORD PTR SS:[EBP+8]
004013BC EB 79 CALL <JMP.&USER32.MessageBoxA>
004013C1 C3 RETN
004013C2 33FF XOR EDI,EDI
004013C4 330B XOR EBX,EBX
004013C6 8A1E MOV BL,BYTE PTR DS:[ESI]
004013C8 84DB TEST BL,BL
004013CA 74 05 JE SHORT CRACKME.004013D1
004013CC 03FB ADD EDI,EBX
004013CE 46 INC ESI
004013CF EB F5 JMP SHORT CRACKME.004013C6
004013D1 C3 RETN
004013D2 2C 20 SUB AL,20
004013D4 8B06 MOV BYTE PTR DS:[ESI],AL
004013D6 C3 RETN
004013D7 C3 RETN
004013D8 3300 XOR EAX,EAX

```

Jump from 0040138B

Allí vemos ora comparación y un salto condicional que nos llevan al cartel maldito pongamos otro BREAKPOINT allí.

```

00401383 5A06 MOV AL,BYTE PTR DS:[ESI]
00401385 84C0 TEST AL,AL
00401387 74 13 JE SHORT CRACKME.00401390
00401389 3C 41 CMP AL,41
0040138B 72 1F JB SHORT CRACKME.00401390
0040138D 2C 5A CMP AL,5A
0040138F 73 03 JNB SHORT CRACKME.00401394
00401391 46 INC ESI
00401392 EB EF JMP SHORT CRACKME.00401383
00401394 EB 39 CALL CRACKME.004013D2
00401399 46 INC ESI
0040139A EB E7 JMP SHORT CRACKME.00401383
0040139C 5E POP ESI
0040139D EB 20 CALL CRACKME.004013C2
004013A2 31F7 78560000 XOR EDI,5678
004013A8 8BC7 MOV EAX,EDI
004013AA EB 15 JMP SHORT CRACKME.004013C1
004013AC 5E POP ESI
004013AD 6A 30 PUSH 30
004013AF 68 68214000 PUSH CRACKME.00402168
004013B4 68 69214000 PUSH CRACKME.00402169
004013B9 FF75 08 PUSH DWORD PTR SS:[EBP+8]
004013BC EB 79 CALL <JMP.&USER32.MessageBoxA>
004013C1 C3 RETN

```

Style = MB_OK!MB_ICONEXCLAMATION!MB_APPLMODAL
Title = "No luck!
Text = "No luck there, mate!"
hOwner
MessageBoxA

Y demos RUN acepto el cartel malo anterior y voy a poner de nuevo

Register

Name

ricnar456

Serial

98989898

OK

Cancel

Al aceptar

```

00401375 . FF75 08      PUSH DWORD PTR SS:[EBP+8]
00401378 . E8 BD000000 CALL <JMP.&USER32.MessageBoxA>
0040137D . C3          RETN
0040137E . 8B7424 04    MOV ESI,DWORD PTR SS:[ESP+4]
00401382 . 56          PUSH ESI
00401383 . 8A06         MOV AL,BYTE PTR DS:[ESI]
00401385 . 84C0         TEST AL,AL
00401387 . 74 13        JE SHORT CRACKME.00401390
00401389 . 3C 41        CMP AL,41
0040138B . 72 1F        JB SHORT CRACKME.004013AC
0040138D . 3C 5A        CMP AL,5A
0040138F . 73 03        JNB SHORT CRACKME.00401394
00401391 . 46          INC ESI
00401392 . EB EF        JMP SHORT CRACKME.00401383
00401394 . E8 39000000 CALL CRACKME.004013D2
00401399 . 46          INC ESI
0040139A . EB E7        JMP SHORT CRACKME.00401383
0040139C . 5E          POP ESI
0040139D . E8 20000000 CALL CRACKME.004013C2
004013A2 . 81F7 78560000 XOR EDI,5678
004013A8 . 8BC7        MOV EAX,EDI
004013AA . EB 15        JMP SHORT CRACKME.004013C1
004013AC . 5E          POP ESI
004013AD . 6A 30        PUSH 30
004013AF . 68 60214000 PUSH CRACKME.00402160
004013B4 . E8 29214000 PUSH CRACKME.00402160

```

Vemos que no saltara la primera vez que para hacia el cartel malo, aunque si compara cada letra que tipee a ver si son números parara una vez por cada letra, F9 nuevamente

A la 7ma vez que apreto recordar que en ricnar456 el 4 es la séptima letra me quiere mostrar el cartel malo saltando

```

00401387 . 74 13        JE SHORT CRACKME.00401390
00401389 . 3C 41        CMP AL,41
0040138B . 72 1F        JB SHORT CRACKME.004013AC
0040138D . 3C 5A        CMP AL,5A
0040138F . 73 03        JNB SHORT CRACKME.00401394
00401391 . 46          INC ESI
00401392 . EB EF        JMP SHORT CRACKME.00401383
00401394 . E8 39000000 CALL CRACKME.004013D2
00401399 . 46          INC ESI
0040139A . EB E7        JMP SHORT CRACKME.00401383
0040139C . 5E          POP ESI
0040139D . E8 20000000 CALL CRACKME.004013C2
004013A2 . 81F7 78560000 XOR EDI,5678
004013A8 . 8BC7        MOV EAX,EDI
004013AA . EB 15        JMP SHORT CRACKME.004013C1
004013AC . 5E          POP ESI
004013AD . 6A 30        PUSH 30
004013AF . 68 60214000 PUSH CRACKME.00402160
004013B4 . 68 60214000 PUSH CRACKME.00402160
004013B9 . FF75 08      PUSH DWORD PTR SS:[EBP+8]
004013BC . E8 79000000 CALL <JMP.&USER32.MessageBoxA>
004013C1 . C3          RETN
004013C2 . 33FF        XOR EDI,EDI
004013C4 . 33DB        XOR EBX,EBX
004013C6 . 8A1F        MOV BL,BYTE PTR DS:[ESI]

```

Style = MB_OK!MB_ICONEXCLAMATION!MB_APPLMODAL
Title = "No luck!"
Text = "No luck there, mate!"
hOwner
MessageBoxA

```

EIP 0040138B CF
C 1 ES 0023 32
R 1 CS 001B 32
A 0 SS 0023 32
Z 0 DS 0023 32
S 1 FS 003B 32
T 0 0000 NL
D 0
O 0 LastErr EF
EFL 00000287 (↑)
ST0 empty 4.137
ST1 empty 3.787
ST2 empty 4.734
ST3 empty 3.491
ST4 empty 6.982

```

Habíamos visto que los JB saltan si es mas bajo y se activa el FLAG C, hago doble click en el FLAG C

| | | |
|----------|---------------|--------------------------------|
| 00401383 | > 8A06 | MOV AL, BYTE PTR DS:[ESI] |
| 00401385 | 84C0 | TEST AL, AL |
| 00401387 | 74 13 | JE SHORT CRACKME.0040139C |
| 00401389 | 3C 41 | CMP AL, 41 |
| 0040138B | 72 1F | JB SHORT CRACKME.004013AC |
| 0040138D | 3C 5A | CMP AL, 5A |
| 0040138F | 73 03 | JNB SHORT CRACKME.00401394 |
| 00401391 | 46 | INC ESI |
| 00401392 | EB EF | JMP SHORT CRACKME.00401383 |
| 00401394 | E8 39000000 | CALL CRACKME.004013D2 |
| 00401396 | 46 | INC ESI |
| 00401398 | EB E7 | JMP SHORT CRACKME.00401383 |
| 0040139A | 5E | POP ESI |
| 0040139C | E8 20000000 | CALL CRACKME.004013C2 |
| 0040139E | 31F7 78560000 | XOR EDI, 5678 |
| 004013A0 | 8BC7 | MOV EAX, EDI |
| 004013A2 | EB 15 | JMP SHORT CRACKME.004013C1 |
| 004013A4 | 5E | POP ESI |
| 004013A6 | 6A 30 | PUSH 30 |
| 004013A8 | 68 60214000 | PUSH CRACKME.00402160 |
| 004013AA | 68 69214000 | PUSH CRACKME.00402169 |
| 004013AC | FF75 08 | PUSH DWORD PTR SS:[EBP+8] |
| 004013AE | E8 79000000 | CALL <JMP.&USER32.MessageBoxA> |

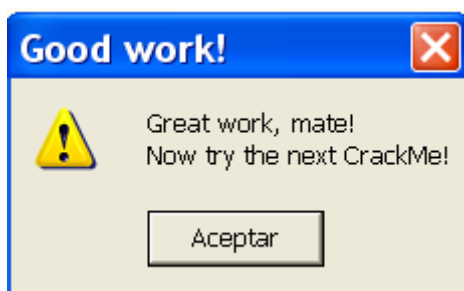
Ahora ya no salta, apreto F9 y repito el mismo procedimiento para los otros dos números que puse en mi nombre fuerza que nunca salte.

| | | |
|----------|--------------|-------------------------------|
| 00401238 | E8 9B010000 | CALL CRACKME.004013D8 |
| 0040123D | 83C4 04 | ADD ESP, 4 |
| 00401240 | 58 | POP EAX |
| 00401241 | 3BC3 | CMP EAX, EBX |
| 00401243 | 74 07 | JE SHORT CRACKME.0040124C |
| 00401245 | E8 18010000 | CALL CRACKME.00401362 |
| 0040124A | EB 9A | JMP SHORT CRACKME.004011E6 |
| 0040124C | E8 FC000000 | CALL CRACKME.00401340 |
| 00401251 | EB 93 | JMP SHORT CRACKME.004011E6 |
| 00401253 | C8 000000 | ENTER 0, 0 |
| 00401257 | 53 | PUSH EBX |
| 00401258 | 56 | PUSH ESI |
| 00401259 | 57 | PUSH EDI |
| 0040125A | 817D 0C 1001 | CMP DWORD PTR SS:[EBP+C], 110 |
| 00401261 | 74 34 | JE SHORT CRACKME.00401297 |
| 00401263 | 817D 0C 1101 | CMP DWORD PTR SS:[EBP+C], 111 |
| 0040126A | 74 35 | JE SHORT CRACKME.004012A1 |

Luego de eso si llego a la comparación final y la cual debe saltar hago doble click en Z

| | | |
|----------|--------------|-------------------------------|
| 0040123D | 83C4 04 | ADD ESP, 4 |
| 00401240 | 58 | POP EAX |
| 00401241 | 3BC3 | CMP EAX, EBX |
| 00401243 | 74 07 | JE SHORT CRACKME.0040124C |
| 00401245 | E8 18010000 | CALL CRACKME.00401362 |
| 0040124A | EB 9A | JMP SHORT CRACKME.004011E6 |
| 0040124C | E8 FC000000 | CALL CRACKME.00401340 |
| 00401251 | EB 93 | JMP SHORT CRACKME.004011E6 |
| 00401253 | C8 000000 | ENTER 0, 0 |
| 00401257 | 53 | PUSH EBX |
| 00401258 | 56 | PUSH ESI |
| 00401259 | 57 | PUSH EDI |
| 0040125A | 817D 0C 1001 | CMP DWORD PTR SS:[EBP+C], 110 |

Y al apretar F9

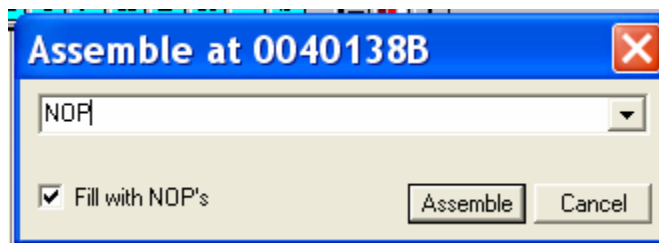


Ahora por supuesto todo esto lo hemos hecho en memoria cambiando FLAGS, como hacemos para guardar definitivos estos cambios y que el crackme acepte cualquier user y serial sin OLLYDBG.

Vayamos al primer salto

| | | | |
|----------|---|---------------|----------------------------|
| 00401382 | > | 56 | PUSH ESI |
| 00401383 | > | 8A06 | MOV AL,BYTE PTR DS:[ESI] |
| 00401385 | > | 84C0 | TEST AL,AL |
| 00401387 | > | 74 13 | JE SHORT CRACKME.0040139C |
| 00401389 | > | 3C 41 | CMP AL,41 |
| 0040138B | > | 72 1F | JB SHORT CRACKME.004013AC |
| 0040138D | > | 3C 5A | CMP AL,5A |
| 0040138F | > | 73 03 | JNB SHORT CRACKME.00401394 |
| 00401391 | > | 46 | INC ESI |
| 00401392 | > | EB EF | JMP SHORT CRACKME.00401383 |
| 00401394 | > | E8 39000000 | CALL CRACKME.004013D2 |
| 00401399 | > | 46 | INC ESI |
| 0040139A | > | EB E7 | JMP SHORT CRACKME.00401383 |
| 0040139C | > | 5E | POP ESI |
| 0040139D | > | E8 20000000 | CALL CRACKME.004013C2 |
| 004013A2 | > | 81F7 78560000 | XOR EDI,5678 |
| 004013A8 | > | 8BC7 | MOV EAX,EDI |
| 004013AA | > | EB 15 | JMP SHORT CRACKME.004013C1 |
| 004013AC | > | 5E | POP ESI |
| 004013AD | > | 6A 30 | PUSH 30 |
| 004013AF | > | 68 60214000 | PUSH CRACKME.00402160 |
| 004013B4 | > | 68 69214000 | PUSH CRACKME.00402169 |

Lo que nosotros hemos hecho en este salto es forzarlo mediante los flags a que no salte nunca a pesar de los valores de la comparación, eso es similar a NOPEAR el salto condicional, si hago click allí, apreto la barra espaciadora y escribo NOP.



| | | | |
|----------|---|-------------|----------------------------|
| 0040138B | > | 90 | NOP |
| 0040138C | > | 90 | NOP |
| 0040138D | > | 3C 5A | CMP AL,5A |
| 0040138F | > | 73 03 | JNB SHORT CRACKME.00401394 |
| 00401391 | > | 46 | INC ESI |
| 00401392 | > | EB EF | JMP SHORT CRACKME.00401383 |
| 00401394 | > | E8 39000000 | CALL CRACKME.004013D2 |

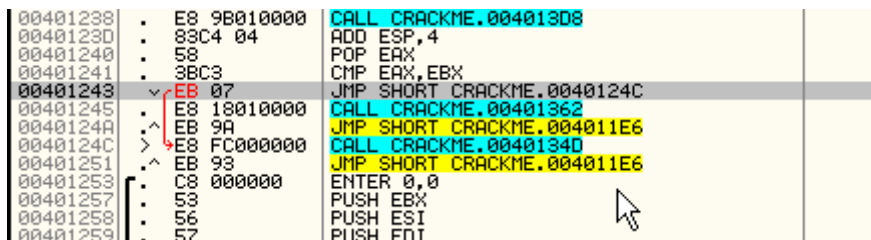
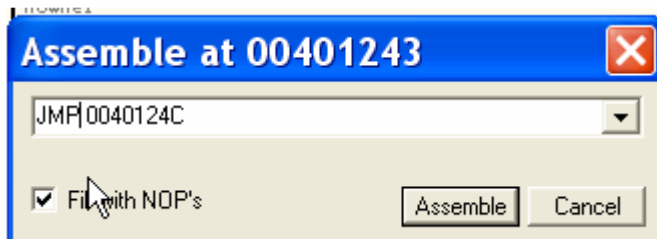
Vemos que nunca saltara quito ese BREAKPOINT apretando F2

| | | | |
|----------|---|-------------|----------------------------|
| 00401383 | > | 8A06 | MOV AL,BYTE PTR DS:[ESI] |
| 00401385 | > | 84C0 | TEST AL,AL |
| 00401387 | > | 74 13 | JE SHORT CRACKME.0040139C |
| 00401389 | > | 3C 41 | CMP AL,41 |
| 0040138B | > | 90 | NOP |
| 0040138C | > | 90 | NOP |
| 0040138D | > | 3C 5A | CMP AL,5A |
| 0040138F | > | 73 03 | JNB SHORT CRACKME.00401394 |
| 00401391 | > | 46 | INC ESI |
| 00401392 | > | EB EF | JMP SHORT CRACKME.00401383 |
| 00401394 | > | E8 39000000 | CALL CRACKME.004013D2 |

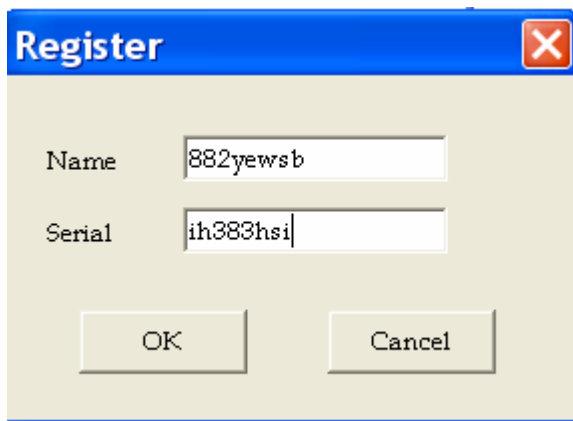
Ahora veamos el otro salto

| | | | |
|----------|---|-------------|----------------------------|
| 00401240 | > | 58 | POP EAX |
| 00401241 | > | 3BC3 | CMP EAX,EBX |
| 00401243 | > | 74 07 | JE SHORT CRACKME.0040124C |
| 00401245 | > | E8 18010000 | CALL CRACKME.00401362 |
| 0040124A | > | EB 9A | JMP SHORT CRACKME.004011E6 |
| 0040124C | > | E8 FC000000 | CALL CRACKME.00401340 |
| 00401251 | > | EB 93 | JMP SHORT CRACKME.004011E6 |
| 00401253 | > | C8 000000 | ENTER 0,0 |
| 00401257 | > | 53 | PUSH EBX |
| 00401258 | > | 5A | PUSH ESI |

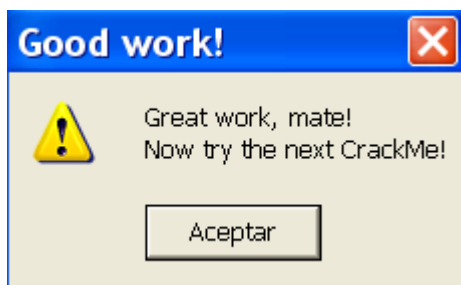
En esta caso fue al revés lo forzamos a saltar siempre mediante la manipulación del FLAG Z, lo cual seria equivalente a cambiar el salto condicional por un JMP.



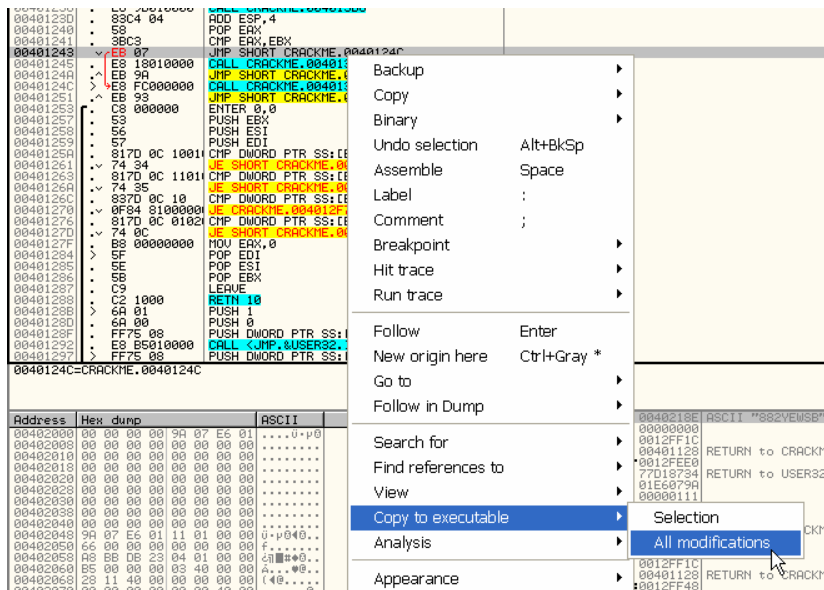
Allí siempre saltara, quitamos también el breakpoint y sin salir de OLLYDBG probamos si quedo bien, apreto F9



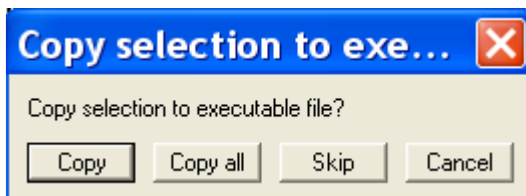
Al apretar OK



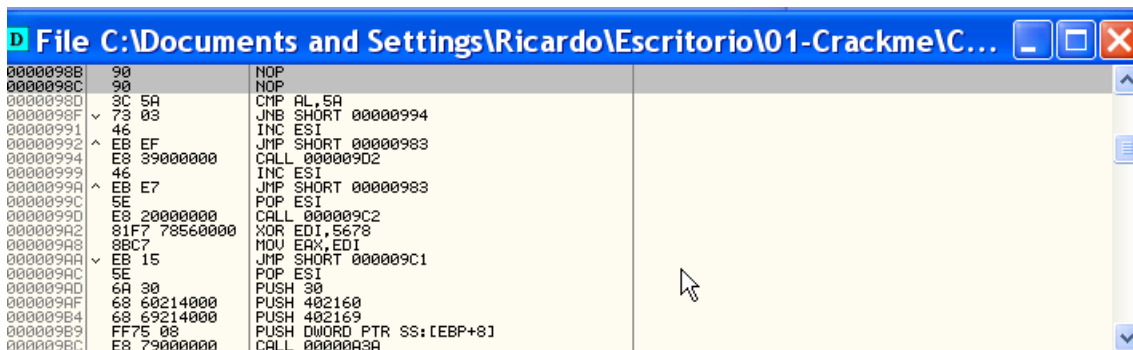
Igual recordamos que cuando escribimos en OLLY con la barra espaciadora o ASSEMBLE los cambios desaparecían al reiniciar, tenemos que hallar la forma de que los guarde de la memoria al archivo definitivo, eso se hace de la siguiente manera



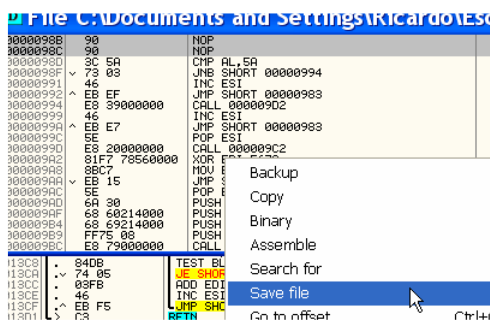
CLICK DERECHO en cualquier parte del listado COPY TO EXECUTABLE-ALL MODIFICATIONS allí se nos abre esto

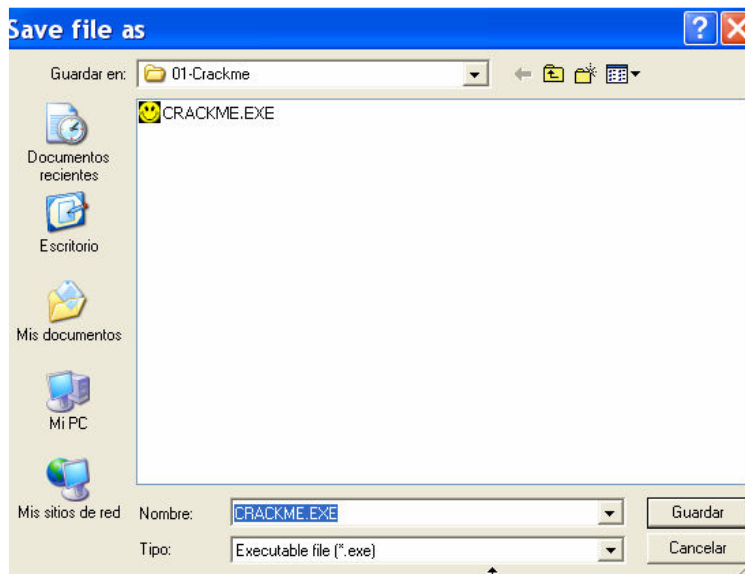


Elegimos COPY ALL para que copie los dos cambios que hicimos

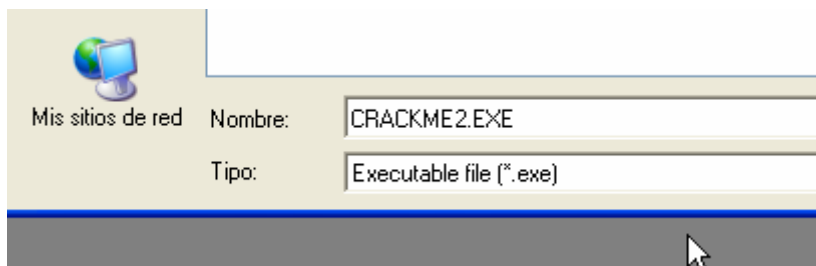


Se nos abre otra ventana, allí hacemos nuevamente CLICK DERECHO-SAVE FILE





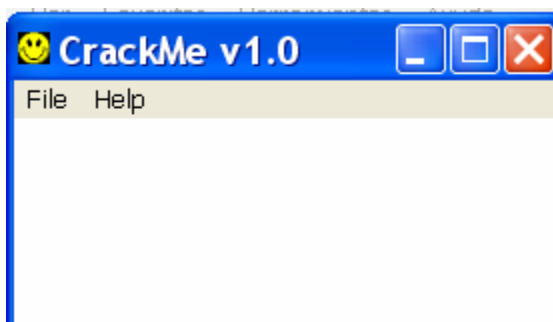
Lo guardamos con OTRO NOMBRE para tener el original para seguir practicando le pondré CRACKME 2



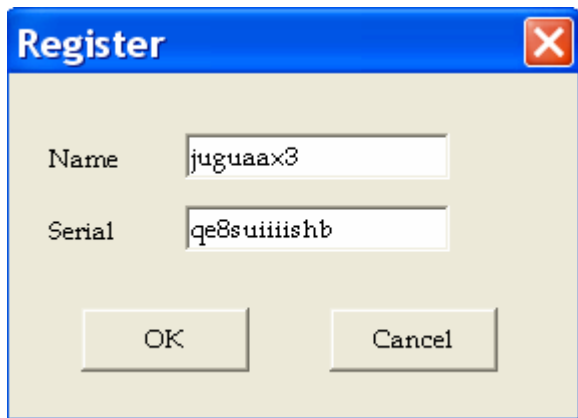
Cierro el OLLYDBG y veo que al lado de donde tenia el crackme esta el crackme2



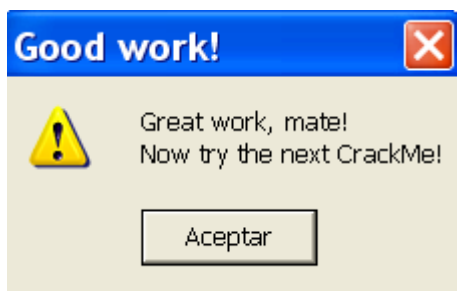
Hago doble click en el crackme2 a ver si quedo bien modificado lo corro sin OLLYDBG.



Voy REGISTER



Apreto OK



Jeje ahí tengo el crackme parcheado pero aun no me conformare con esto, mas adelante veremos como trabaja con los serials y como compara si en el nombre hay números, exhaustivamente y hallaremos serials correctos para nuestro nombre que los acepte sin parchear pero para llegar a eso aun faltan algunos conocimientos y practicas anteriores.

APÉNDICE PARA WINDOWS 98

Bueno usar OLLYDBG en w98 como ya les dije es bastante limitado, el mismo sistema no nos deja poner BREAKPOINTS en las apis directamente como en w98, por lo tanto el método será similar con la única limitación que ustedes no podrán poner BREAKPOINT en la api, si hacen CLICK DERECHO SEARCH FOR NAME (labels) in this module como dice allí la primera parte de la explicación para XP

| | | | | |
|----------|--------|--------|------------------------|--|
| 0040318C | .idata | Import | USER32.LoadCursorA | |
| 004031A0 | .idata | Import | USER32.LoadIconA | |
| 004031C8 | .idata | Import | USER32.LoadMenuA | |
| 0040319C | .idata | Import | USER32.LoadStringA | |
| 0040322C | .idata | Import | KERNEL32.lstrlen | |
| 00403194 | .idata | Import | USER32.MessageBeep | |
| 004031AC | .idata | Import | USER32.MessageBoxA | |
| 00401000 | CODE | Export | <ModuleEntryPoint> | |
| 004031C0 | .idata | Import | USER32.MoveWindow | |
| 00403220 | .idata | Import | KERNEL32.OpenFile | |
| 004031B0 | .idata | Import | USER32.PostQuitMessage | |
| 00403288 | .idata | Import | COMDLG32.PrintDlgA | |

Les saldrá la misma lista y de la misma forma apretando el nombre de la api llegaran hasta el nombre de la misma, pero en 98 no pueden hacer CLICK DERECHO-TOGGLE BREAKPOINT ON IMPORT ya que no permite poner BREAKPOINTS en las apis, si no que deberán hacer.

| | | | | |
|----------|--------|--------|------------------------|--|
| 00403198 | .idata | Import | USER32.LoadMenu | |
| 0040319C | .idata | Import | USER32.LoadStringA | |
| 0040322C | .idata | Import | KERNEL32.lstrlen | |
| 00403194 | .idata | Import | USER32.MessageBeep | |
| 004031AC | .idata | Import | USER32.MessageBoxA | |
| 00401000 | CODE | Export | <ModuleEntryPoint> | |
| 004031C0 | .idata | Import | USER32.MoveWindow | |
| 00403220 | .idata | Import | KERNEL32.OpenFile | |
| 004031B0 | .idata | Import | USER32.PostQuitMessage | |
| 00403288 | .idata | Import | COMDLG32.PrintDlgA | |
| 0040323C | .idata | Import | KERNEL32.ReadFile | |
| 004031E0 | .idata | Import | USER32.RegisterClassA | |
| 004031D0 | .idata | Import | USER32.SendMessageA | |
| 004031A8 | .idata | Import | USER32.SetFocus | |
| 004031D4 | .idata | Import | USER32.SetTimer | |
| 004031D8 | .idata | Import | USER32.SetWindowPos | |
| 004031CC | .idata | Import | USER32.ShowWindow | |
| 00403260 | .idata | Import | GDI32.StartDocA | |
| 0040325C | .idata | Import | GDI32.StartPage | |

Actualize

Follow import in Disassembler

Follow in Dump

Find references to import

View call tree

Toggle breakpoint on import

Creo que el menú es similar si no siempre tendrá la opción para BUSCAR LAS REFERENCIAS o sea los llamados a la api en el crackme.

| Address | Disassembly | Comment |
|----------|--|--------------------|
| 0040135C | CALL <JMP.&USER32.MessageBoxA> | |
| 00401378 | CALL <JMP.&USER32.MessageBoxA> | |
| 004013BC | CALL <JMP.&USER32.MessageBoxA> | |
| 0040143A | JMP DWORD PTR DS:[&USER32.MessageBoxA] | USER32.MessageBoxA |

Esto es limitadísimo comparado con la posibilidad de poner un BREAKPOINT en la API, porque si la API es llamada, si hay un BP en la API parara, en cambio en 98, la API puede ser llamada en alguna forma que engañe el análisis del OLLYDBG (y hay muchas formas créanlo) y no saldrá dicha llamada entre las referencias y no tendrán forma de saber de donde fue llamada la api ni parara.

Igual pueden seguir este tute de la misma forma sabiendo que hay tres llamadas a la api MessageBoxA y en vez de poner UN BREAKPOINT en la api, pongo un BP en cada referencia a la api, allí en el cuadro de referencias apreto F2 en cada una,

| Address | Disassembly | Comment |
|----------|--|--------------------|
| 0040135C | CALL <JMP.&USER32.MessageBoxA> | |
| 00401378 | CALL <JMP.&USER32.MessageBoxA> | |
| 004013BC | CALL <JMP.&USER32.MessageBoxA> | |
| 0040143A | JMP DWORD PTR DS:[&USER32.MessageBoxA] | USER32.MessageBoxA |

Al dar Run voy a HELP-REGISTER tipo estos datos

Register

Name

ricnar456

Serial

98989898

OK

Cancel

Y al apretar OK parara

| | | | |
|----------|-----------------|--------------------------------|---|
| 0040139D | . E8 20000000 | CALL CRACKME.004013C2 | |
| 004013A2 | . 81F7 78560000 | XOR EDI,5678 | |
| 004013A8 | . 8BC7 | MOV EAX,EDI | |
| 004013AA | > EB 15 | JMP SHORT CRACKME.004013C1 | |
| 004013AC | > 5E | POP ESI | |
| 004013AD | > 6A 30 | PUSH 30 | |
| 004013AF | . 68 60214000 | PUSH CRACKME.00402160 | Style = MB_OK!MB_ICONEXCLAMATION!MB_APPLMODAL |
| 004013B4 | . 68 69214000 | PUSH CRACKME.00402169 | Title = "No luck!" |
| 004013B9 | . FF75 08 | PUSH DWORD PTR SS:[EBP+8] | Text = "No luck there, mate!" |
| 004013BC | . E8 79000000 | CALL <JMP.&USER32.MessageBoxA> | hOwner |
| 004013C1 | > C3 | RETN | MessageBoxA |
| 004013C2 | > 33FF | XOR EDI,EDI | |
| 004013C4 | > 33DB | XOR EBX,EBX | |
| 004013C6 | > 8A1E | MOV BL,BYTE PTR DS:[ESI] | |
| 004013C8 | > 84DB | TEST BL,BL | |
| 004013CA | > 74 05 | JE SHORT CRACKME.004013D1 | |

Luego en XP ponemos un BP en el RET de la api o donde termina la misma, eso equivale en 98 a poner un BP en el RET que esta justo debajo ya que no podemos entrar a la api.

| | | | |
|----------|-----------------|--------------------------------|---|
| 0040139D | . 81F7 78560000 | XOR EDI,5678 | |
| 004013A2 | . 8BC7 | MOV EAX,EDI | |
| 004013AA | > EB 15 | JMP SHORT CRACKME.004013C1 | |
| 004013AC | > 5E | POP ESI | |
| 004013AD | > 6A 30 | PUSH 30 | |
| 004013AF | . 68 60214000 | PUSH CRACKME.00402160 | Style = MB_OK!MB_ICONEXCLAMATION!MB_APPLMODAL |
| 004013B4 | . 68 69214000 | PUSH CRACKME.00402169 | Title = "No luck!" |
| 004013B9 | . FF75 08 | PUSH DWORD PTR SS:[EBP+8] | Text = "No luck there, mate!" |
| 004013BC | . E8 79000000 | CALL <JMP.&USER32.MessageBoxA> | hOwner |
| 004013C1 | > C3 | RETN | MessageBoxA |
| 004013C2 | > 33FF | XOR EDI,EDI | |
| 004013C4 | > 33DB | XOR EBX,EBX | |
| 004013C6 | > 8A1E | MOV BL,BYTE PTR DS:[ESI] | |

Pues ya saben cuando el el tute dice que esta en el inicio de la api y ve a donde retorna ustedes saben que retorna en este caso a 4013C1 al ret que esta justo abajo y si es en otra referencia sera el RET que esta debajo correspondiente.

Sabiendo esto ya pueden retornar a

[RETORNO DEL APÉNDICE PARA WINDOWS 98 desde aquí sigue para todos los SO.](#)

Y continuar el tute desde allí siempre sabiendo la diferencia principal que es que en W98 no podemos poner BPX o BREAKPOINTS en las apis si no en las referencias.

Hasta la parte 10

Ricardo Narvaja

22 de noviembre de 2005