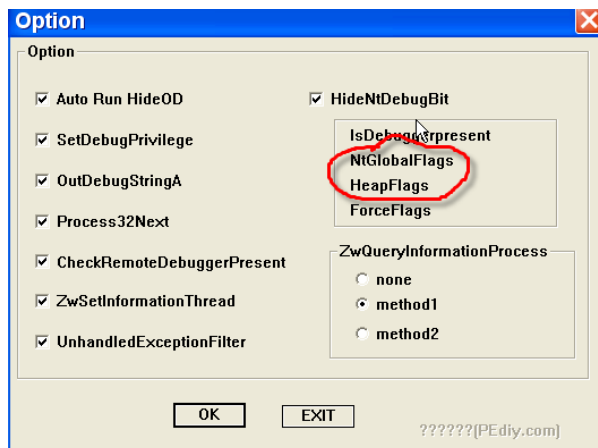


INTRODUCCION AL CRACKING CON OLLYDBG parte 23

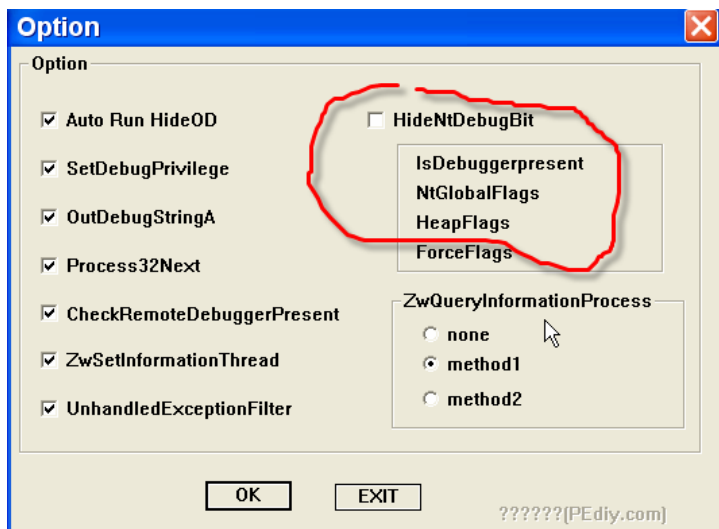
Vamos a ver aquí la ultima parte sobre antidebugging que versara sobre los ProcessHeap Flag y NTGlobalFlag y con eso ya tenemos una idea de los trucos antidebugging mas conocidos, por supuesto no son todos pero creo que son los básicos que hay que saber y con eso correrán la mayoría de los programas en OLLYDBG, por supuesto hay algunos protectores como execryptor que son campeones de la detección de OLLYDBG y además de todos los trucos que enseñamos aquí, añaden 4 o 5 mas de su propia cosecha, pero eso es algo mas especifico para leer tutes sobre dicho packer como los escritos por Juan Jose y además saber que execryptor cada versión que saca nueva, trata de agregarle nuevas detecciones y trucos por lo cual es necesario hacer un estudio sobre los antidebugging de execryptor mas que sobre antidebugging en general.

Ya sabemos que con los plugins que tenemos ambos flags no serán detectados, pero es bueno saber como ubicarlos en nuestra maquina.



Allí están en el PLUGIN HIDEOD, las opciones para ocultar el OLLYDBG de la detección por ambos flags, pero como podemos localizarlos en nuestra maquina a mano, pues eso es saber y el saber no ocupa lugar.

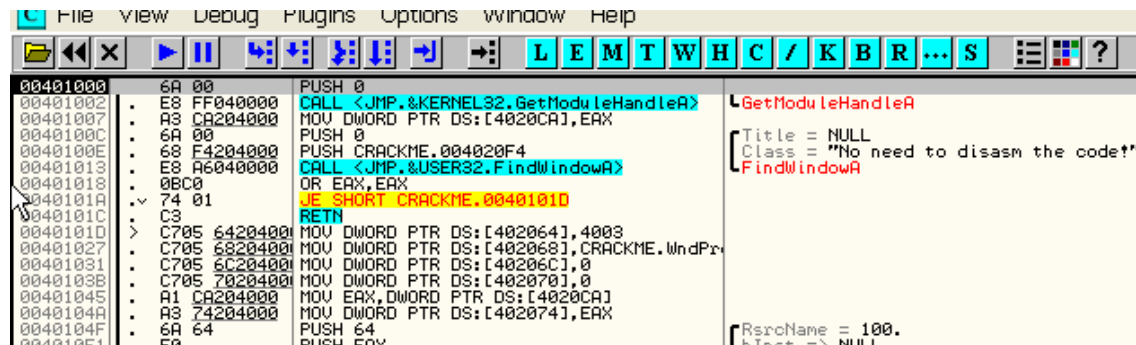
Estos dos últimos flags muestran que el proceso esta siendo debuggeador, son fácilmente localizables, si no recuerdan como hallar la zona del byte IsDebuggerPresent releen la parte 19 que es la que explicábamos como hallar a mano la zona del byte IsDebuggerPresent ya que estos bytes son vecinos de ese.



Quitémosle un minuto la tilde en el plugin para ver los valores que tienen los flags, sin utilizar el plugin para ocultarlos.

Para practicar con estos flags utilizaremos el CRACKME DE CRUEHEAD 1 en el cual localizaremos ambos flags.

Abrámoslo en OLLYDBG, recordando verificar que la tilde en el plugin HideOdbg como muestra la imagen anterior este sin marcar.

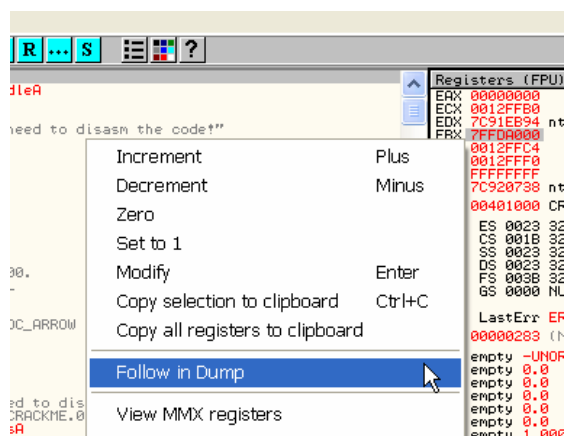


Pues bien, veamos como localizar y cambiar a mano ambos flags.

Lo que debemos hallar es la zona del byte IsDebuggerPresent como vimos en el tutorial 19, la forma facil de hallar esa zona es marcar el registro EBX aquí en el ENTRY POINT, y hacer FOLLOW IN DUMP, la forma completa puede leerse en ese tutorial si alguien no la recuerda.

Estoy en el ENTRY POINT

Marco EBX-FOLLOW IN DUMP



Y en el dump vemos la zona deseada, recuerden que en su maquina puede cambiar la dirección, con respecto a la mía y cada vez que reinicien un programa también variara.

0040109F	6A 00	PUSH 0
----------	-------	--------

Address	Hex dump	ASCII
7FFDA000	00 00 00 00 FF FF FF FF
7FFDA008	00 00 40 00 A0 1E 24 00	..@.â\$. .
7FFDA010	00 00 02 00 00 00 00 00	..@.....
7FFDA018	00 00 14 00 C0 E4 98 7C	..Œ.ŒŸ!
7FFDA020	05 10 91 7C ED 10 91 7C	ŒŸ!ŸŸ!
7FFDA028	01 00 00 00 80 29 D1 77	@...ŸŸw
7FFDA030	00 00 00 00 00 00 00 00
7FFDA038	00 00 00 00 00 00 00 00
7FFDA040	80 E4 98 7C FF 03 00 00	ŸŸ! Ÿ.
7FFDA048	00 00 00 00 00 00 6F 7Foâ
7FFDA050	00 00 6F 7F 88 06 6F 7Foâoâ
7FFDA058	00 00 FB 7F 00 10 FC 7F	..¹â.Ÿâ
7FFDA060	00 20 FD 7F 01 00 00 00	..²â@...
7FFDA068	00 00 00 00 00 00 00 00
7FFDA070	00 80 9B 07 6D E8 FF FF	..Ÿ·mŸ
7FFDA078	00 00 10 00 00 20 00 00	..Ÿ.Ÿ.Ÿ.
7FFDA080	00 00 01 00 00 10 00 00	..@.Ÿ.Ÿ.
7FFDA088	05 00 00 00 10 00 00 00	Ÿ.Ÿ.Ÿ.Ÿ.
7FFDA090	80 DE 98 7C 00 00 4E 00	ŸŸŸ!..N.
7FFDA098	00 00 00 00 14 00 00 00Œ...
7FFDA0A0	08 C0 98 7C 05 00 00 00	ŸŸŸ!Ÿ.Ÿ.
7FFDA0A8	01 00 00 00 28 0A 00 02	@...Ÿ.Ÿ.
7FFDA0B0	02 00 00 00 02 00 00 00	@...@...
7FFDA0B8	03 00 00 00 0A 00 00 00	Ÿ.Ÿ.Ÿ.Ÿ.

Por supuesto recordamos que ese es el Byte detectado por la api IsDebuggerPresent, pues el NTGlobalFlag es vecino de este lo vemos sumando 68 a la dirección, que nos mostraba EBX, en mi caso EBX era 7ffda000 le sumo 68, será 7ffda068.

Address	Hex dump	ASCII
7FFDA000	00 00 00 00 FF FF FF FF
7FFDA008	00 00 40 00 A0 1E 24 00	..@.â\$. .
7FFDA010	00 00 02 00 00 00 00 00	..@.....
7FFDA018	00 00 14 00 C0 E4 98 7C	..Œ.ŒŸ!
7FFDA020	05 10 91 7C ED 10 91 7C	ŒŸ!ŸŸ!
7FFDA028	01 00 00 00 80 29 D1 77	@...ŸŸw
7FFDA030	00 00 00 00 00 00 00 00
7FFDA038	00 00 00 00 00 00 00 00
7FFDA040	80 E4 98 7C FF 03 00 00	ŸŸ! Ÿ.
7FFDA048	00 00 00 00 00 00 6F 7Foâ
7FFDA050	00 00 6F 7F 88 06 6F 7Foâoâ
7FFDA058	00 00 FB 7F 00 10 FC 7F	..¹â.Ÿâ
7FFDA060	00 20 FD 7F 01 00 00 00	..²â@...
7FFDA068	00 00 00 00 00 00 00 00
7FFDA070	00 80 9B 07 6D E8 FF FF	..Ÿ·mŸ
7FFDA078	00 00 10 00 00 20 00 00	..Ÿ.Ÿ.Ÿ.
7FFDA080	00 00 01 00 00 10 00 00	..@.Ÿ.Ÿ.
7FFDA088	05 00 00 00 10 00 00 00	Ÿ.Ÿ.Ÿ.Ÿ.
7FFDA090	80 DE 98 7C 00 00 4E 00	ŸŸŸ!..N.
7FFDA098	00 00 00 00 14 00 00 00Œ...

Ese es el famoso NTGlobalFlag que es diferente de cero si hay un debugger y cero si no lo hay, pongamos a mano el flag este a cero, modificando ese valor a mano.

0040105C	Undo selection Alt+BkSp	LoadCursorA
00401061	Copy	Edit
00401063	Binary	Fill with 00
0040106D	Breakpoint	Fill with FF
00401077	Search for	Binary copy
00401081	Go to	
00401088	Hex	
00401090	Text	
00401095	Short	
00401097	Long	
0040109F	Float	
	Disassemble	
	Special	
	Appearance	

Lo pongo a cero

Address	Hex dump	ASCII
7FFDA000	00 00 00 00 FF FF FF FF
7FFDA008	00 00 40 00 A0 1E 24 00	..@.â\$.
7FFDA010	00 00 02 00 00 00 00 00	..@.....
7FFDA018	00 00 14 00 C0 E4 98 7C	..@.L\$ÿ!
7FFDA020	05 10 91 7C ED 10 91 7C	♢!ÿ!ÿ!
7FFDA028	01 00 00 00 80 29 D1 77	@...Ç!ðw
7FFDA030	00 00 00 00 00 00 00 00
7FFDA038	00 00 00 00 00 00 00 00
7FFDA040	80 E4 98 7C FF 03 00 00	Ç\$ÿ! ♢...
7FFDA048	00 00 00 00 00 00 6F 7Foð
7FFDA050	00 00 6F 7F 88 06 6F 7F	..oðë+oð
7FFDA058	00 00 FB 7F 00 10 FC 7F	..!ð.♢!ð
7FFDA060	00 20 FD 7F 01 00 00 00	.. ºð...
7FFDA068	00 00 00 00 00 00 00 00
7FFDA070	00 80 9B 07 6D E8 FF FF	..Ç!m♢
7FFDA078	00 00 10 00 00 20 00 00	..♢...♢
7FFDA080	00 00 01 00 00 10 00 00	..@.♢...
7FFDA088	05 00 00 00 10 00 00 00	♢...♢...

Allí esta localizado y puesto a mano a cero el NtGlobalFlag

Ahora debemos localizar el otro byte el ProcessHeap Flag, ese también se localiza fácil.

Al valor que me otorgaba EBX en el entry point en este caso le sumo 18, y nos da una dirección que es en mi caso 014000, que es la dirección del heap o sea una zona de memoria creada al arrancar el programa que guarda ciertos datos del mismo, no especificaremos mucho aquí sobre eso.

Address	Hex dump	ASCII
7FFDA000	00 00 00 00 FF FF FF FF
7FFDA008	00 00 40 00 A0 1E 24 00	..@.â\$.
7FFDA010	00 00 02 00 00 00 00 00	..@.....
7FFDA018	00 00 14 00 C0 E4 98 7C	..@.L\$ÿ!
7FFDA020	05 10 91 7C ED 10 91 7C	♢!ÿ!ÿ!
7FFDA028	01 00 00 00 00 00 00 00	@...Ç!ðw
7FFDA030	00 00 00 00 00 00 00 00
7FFDA038	00 00 00 00 00 00 00 00
7FFDA040	80 E4 98 7C 01 00 00 00	Ç\$ÿ!@...
7FFDA048	00 00 00 00 00 00 6F 7Foð
7FFDA050	00 00 6F 7F 88 06 6F 7F	..oðë+oð
7FFDA058	00 00 FB 7F 00 10 FC 7F	..!ð.♢!ð
7FFDA060	00 20 FD 7F 01 00 00 00	.. ºð...
7FFDA068	00 00 00 00 00 00 00 00
7FFDA070	00 80 9B 07 6D E8 FF FF	..Ç!m♢
7FFDA078	00 00 10 00 00 20 00 00	..♢...♢
7FFDA080	00 00 01 00 00 10 00 00	..@.♢...
7FFDA088	03 00 00 00 10 00 00 00	♢...♢...
7FFDA090	80 DE 98 7C 00 00 00 00	Ç!ÿ!....
7FFDA098	00 00 00 00 00 00 00 00
7FFDA0A0	08 C0 98 7C 05 00 00 00	!ÿ!+...
7FFDA0A8	01 00 00 00 28 0A 00 02	@...(!.@
7FFDA0B0	02 00 00 00 02 00 00 00	@...@...
7FFDA0B8	04 00 00 00 00 00 00 00	♢...♢...
7FFDA0C0	00 00 00 00 00 00 00 00

Vayamos a ver esa zona del heap

Address	Hex dump	ASCII
7FFDA000	00 00 00 00 FF FF FF FF
7FFDA008	00 00 40 00 A0 1E 24 00	..@.â\$.
7FFDA010	00 00 02 00 00 00 00 00	..@.....
7FFDA018	00 00 14 00 C0 E4 98 7C	..@.L\$ÿ!
7FFDA020	05 10 91 7C ED 10 91 7C	♢!ÿ!ÿ!
7FFDA028	01 00 00 00 00 00 00 00	@...Ç!ðw
7FFDA030	00 00 00 00 00 00 00 00
7FFDA038	00 00 00 00 00 00 00 00
7FFDA040	80 E4 98 7C 01 00 00 00	Ç\$ÿ!@...
7FFDA048	00 00 00 00 00 00 6F 7Foð
7FFDA050	00 00 6F 7F 88 06 6F 7F	..oðë+oð
7FFDA058	00 00 FB 7F 00 10 FC 7F	..!ð.♢!ð
7FFDA060	00 20 FD 7F 01 00 00 00	.. ºð...
7FFDA068	00 00 00 00 00 00 00 00
7FFDA070	00 80 9B 07 6D E8 FF FF	..Ç!m♢
7FFDA078	00 00 10 00 00 20 00 00	..♢...♢
7FFDA080	00 00 01 00 00 10 00 00	..@.♢...
7FFDA088	03 00 00 00 10 00 00 00	♢...♢...
7FFDA090	80 DE 98 7C 00 00 00 00	Ç!ÿ!....

Backup

Copy

Binary

Breakpoint

Search for

Follow DWORD in Dump

Go to

Marcamos los bytes y elegimos FOLLOW DWORD IN DUMP con lo cual nos llevara a la zona del heap

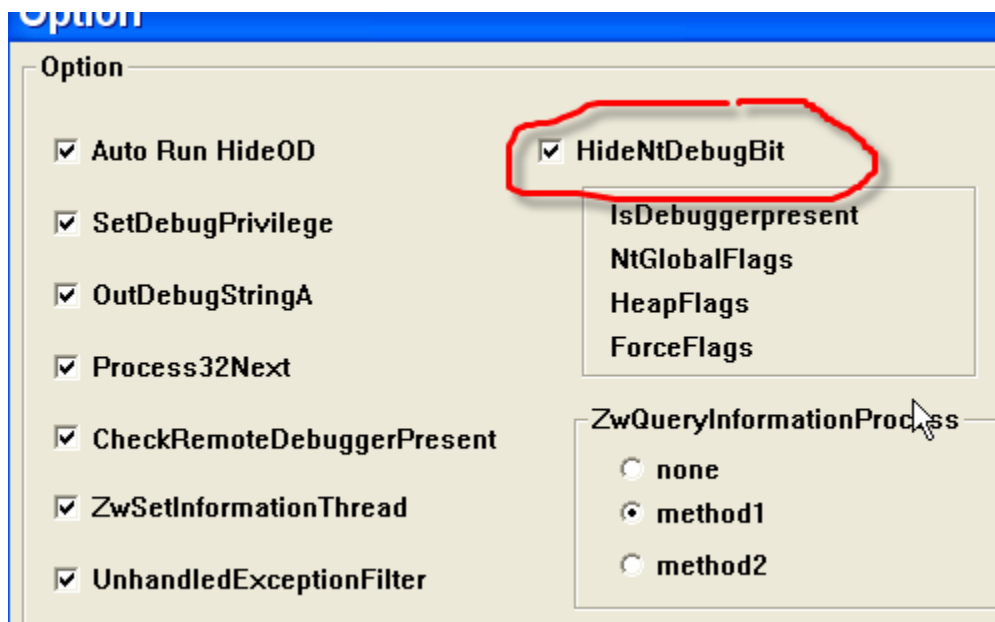
Address	Hex dump	ASCII
00140000	C8 00 00 00 4F 01 00 00	...00..
00140008	FF EE FF EE 02 00 00 00	-0...
00140010	00 00 00 00 00 FE 00 00	...0...
00140018	00 00 10 00 00 20 00 00	...0...
00140020	00 02 00 00 00 20 00 00	...0...
00140028	C3 00 00 00 FF EF FD 7F	...2Δ
00140030	01 00 08 06 00 00 00 00	0.0+...
00140038	00 00 00 00 00 00 00 00	...0...
00140040	00 00 00 00 98 01 14 00	...0+0.

Y sumándole 10 a la dirección base del heap llegamos al dword que esta allí marcado, el cual esta a cero como si no hay debugger, debe ser por tanto plugin que tenemos, si abro el mismo crackme en un ollydbg sin ningún plugin.

Address	Hex dump	ASCII
00140000	C8 00 00 00 75 01 00 00	...u0..
00140008	FF EE FF EE 62 00 00 50	-b..P
00140010	00 00 00 40 00 FE 00 00	...0...
00140018	00 00 10 00 00 20 00 00	...0...
00140020	00 02 00 00 00 20 00 00	...0...
00140028	16 00 00 00 FF EF FD 7F	...2Δ
00140030	01 00 08 06 00 00 00 00	0.0+...
00140038	00 00 00 00 00 00 00 00	...0...
00140040	00 00 00 00 98 05 14 00	...0+0.
00140048	17 00 00 00 F8 FF FF FF	...0
00140050	50 00 14 00 50 00 14 00	P.0.P.0.
00140058	40 06 14 00 00 00 00 00	@+0.....
00140060	00 00 00 00 00 00 00 00	...0...
00140068	00 00 00 00 00 00 00 00	...0...

Veo que allí no esta a cero, que es el valor cuando hay debugger quiere decir que entre tanto plugin este byte se pone a cero, por alguno de ellos aun sin estar la tilde marcada.

Por lo demás volvamos a poner las tildes en el HideOdb



Y reiniciemos el crackme de cruehead

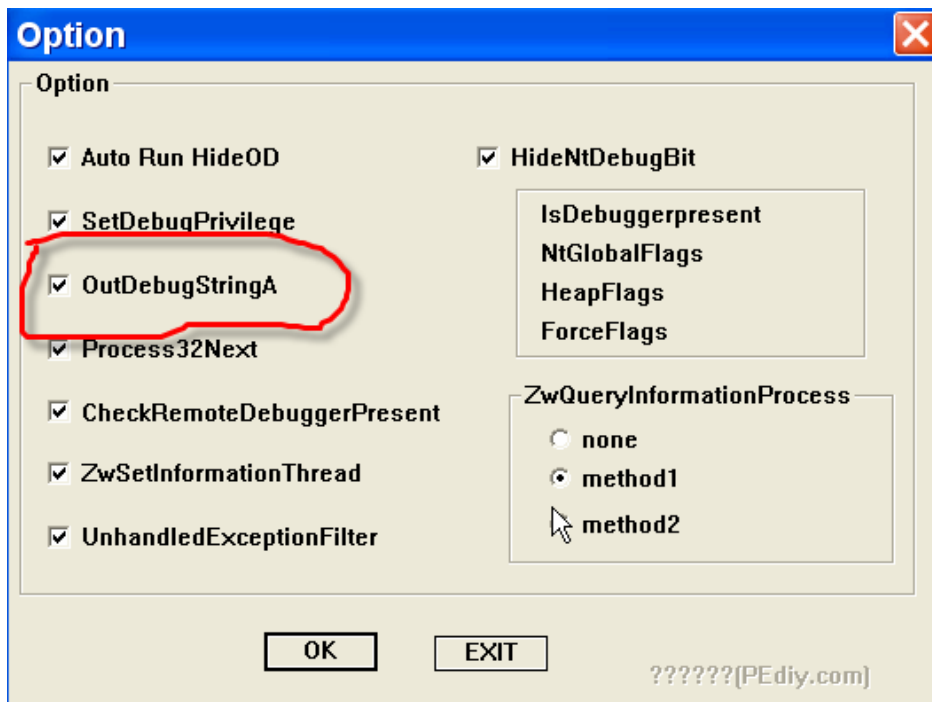
Address	Hex dump	ASCII
7FFD4000	00 00 00 00 FF FF FF FF
7FFD4008	00 00 00 00 A0 1E 24 00	..@.\$.
7FFD4010	00 00 02 00 00 00 00 00	..B....
7FFD4018	00 00 14 00 C0 E4 98 7C	..\$.t\$y!
7FFD4020	05 10 91 7C ED 10 91 7C	..e!y!e!
7FFD4028	01 00 00 00 80 29 D1 77	0...C)0w
7FFD4030	00 00 00 00 00 00 00 00
7FFD4038	00 00 00 00 00 00 00 00
7FFD4040	80 E4 98 7C FF 03 00 00	C\$y! .
7FFD4048	00 00 00 00 00 00 6F 7Fo
7FFD4050	00 00 6F 7F 88 06 6F 7F	..o\$+o
7FFD4058	00 00 FB 7F 00 10 FC 7F	..!o.}o
7FFD4060	00 20 FD 7F 01 00 00 00	..z0...
7FFD4068	00 00 00 00 00 00 00 00
7FFD4070	00 00 9B 07 6D E8 FF FF	.Qxmp
7FFD4078	00 00 10 00 00 20 00 00	.. .
7FFD4080	00 00 01 00 00 10 00 00	..0. .
7FFD4088	05 00 00 00 10 00 00 00	.. .
7FFD4090	80 DE 98 7C 00 00 4E 00	Ciy!..N.
7FFD4098	00 00 00 00 14 00 00 00	..\$. .
7FFD40A0	D8 C0 98 7C 05 00 00 00	iLy!z..
7FFD40A8	01 00 00 00 28 0A 00 02	0...(.0
7FFD40B0	02 00 00 00 02 00 00 00	0...0...

Vemos que parados en el entryptpoint tanto el flag de la api IsDebuggerPresent como el NtGlobalFlag están a cero, y si busco el ProcessHeapFlag

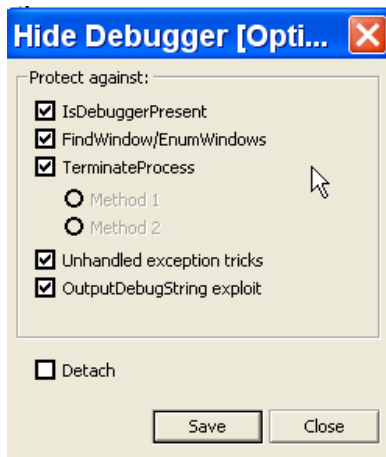
Address	Hex dump	ASCII
00020000	00 10 00 00 A4 09 00 00	.. .K...
00020008	01 00 00 00 00 00 00 00	0.....
00020010	00 00 00 00 00 00 00 00
00020018	00 00 00 00 00 00 00 00
00020020	00 00 00 00 5A 00 08 02	...Z.00
00020028	90 02 02 00 0C 00 00 00	e00....
00020030	86 03 88 03 98 04 02 00	S#e\$U+0.
00020038	70 00 72 00 20 00 02 00	p.r. 00.
00020040	74 00 76 00 94 0A 02 00	t.v.000.

También esta a cero, quiere decir que verifique que el plugin funciona correctamente y que el OLLYDBG no será descubierto por alguno de estos flags, y además aprendimos a hallarlos y cambiarlos a mano.

El ultimo truco antiolly que solo lo mocionare pues al parchear con el repair ya dimos cuenta de el y ambos plugins lo solucionan es el



Y



Se refiere a un bug del OLLYDBG que cuando el programa envía una determinada string muy larga a esa api el OLLYDBG no la puede procesar y se cuelga, teniendo los plugins por supuesto esto no afecta en nada solo lo pongo aquí como dato ilustrativo aquí extractado del tutorial de Juan Jose de execryptor le copiamos la parte que explica esto.

Bueno, me dejo de rollos, de esta manera me di cuenta que el problema venia con el truco antiOlly de **OutputDebugString**, que justamente utiliza este programa y no los demás exectryptos que habia probado. Este truco consiste en mandarle un string preparado al Olly, que no lo sabe aceptar, y te da un error que cierra el Olly (creo ¿?) porque en realidad nunca me he encontrado con él, y si algún programa lo ha tenido el **plugin HideDebugger** lo neutraliza, y te manda una linda sonrisa:

Debug string: :-)

La forma de realizarlo es muy simple tienes que llamar a `OutputDebugStringA` con una cadena de caracteres `%s` bien larga, yo he contado 100d:

```
0013FDE0 004F1757 CALL to OutputDebugStringA  
0013FDE4 0013FDE8 String = "%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s"  
0013FDE8 73257325  
0013FDEC 73257325
```

Bueno como el tute esta genial, para que abundar, allí esta clara la explicación y los plugins dan cuenta de este bug del OLLYDBG perfectamente.

Para practicar les dejo el crackme adjunto antisocial tiene todos los trucos que vimos en la parte de antidebugging , mas un par de trucos caseros jeje, el tema es hacerlo correr en OLLYDBG, si alguien hace un olly sin plugins ni nada en otra carpeta y lo hace correr a mano aplicando todo lo que vimos, pues se recibe de genio antidebugger a mano, por supuesto deberá poner un poco de imaginación pues tiene un truco que le impide correr mas que ser un truco antidebugger hay que reparar una línea nopeandola, para que pase ese primer error, luego de eso tiene los antidebuggers que hemos visto, mas uno que no hemos visto y es el siguiente si al quitarle todas las tildes de DEBUGGING OPTION-EXCEPTIONS les para en un INT68 es un comando que provocara un error, el cual hay que nopear para que continué corriendo el programa, lamentablemente si ponemos todas las tildes al tratar de pasar esa excepcion nos dara error el OLLY y se terminara todo.

O sea sepan que si les para en cualquier excepción si no están puestas las tildes, se debe apretar SHIFT mas F9 para pasarlas, eso lo veremos en la parte siguiente de excepciones, pero si la excepción se genero en un INT68, solo se debe nopear ese comando y dar RUN con eso pasara bien.

Bueno les dejo una dura diversión, si quieren pueden tratar de repararlo y hacerlo correr primero con los plugins con lo cual podrán ver el error y ver donde esta el INT68 y ver que el programa corre, y como segundo paso pueden abrirlo en un OLLYDBG sin plugins y tratar de evitar todo a mano, a ver si les sale, eso si será duro, jeje

Hasta la parte 24

Ricardo Narvaja

04 de enero de 2006