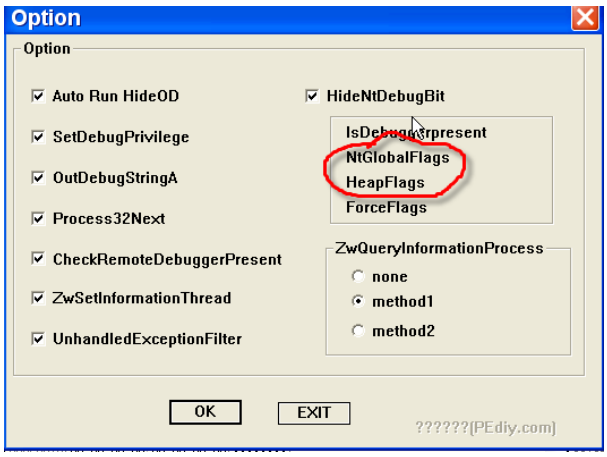
INTRODUCCION AL CRACKING CON OLLYDBG parte 23

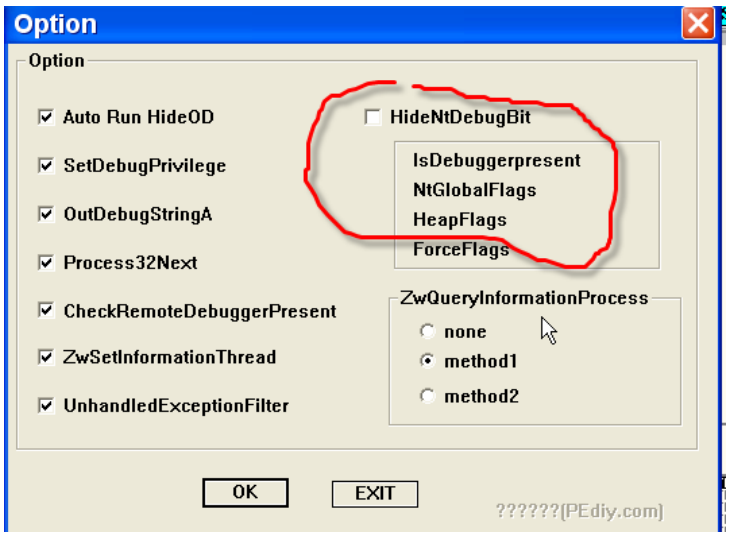
Vamos a ver aquí la ultima parte sobre antidebugging que versara sobre los ProcessHeap Flag y NTGlobalFlag y con eso ya tenemos una idea de los trucos antidebugging mas conocidos, por supuesto no son todos pero creo que son los básicos que hay que saber y con eso correrán la mayoría de los programas 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 execriptor 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.

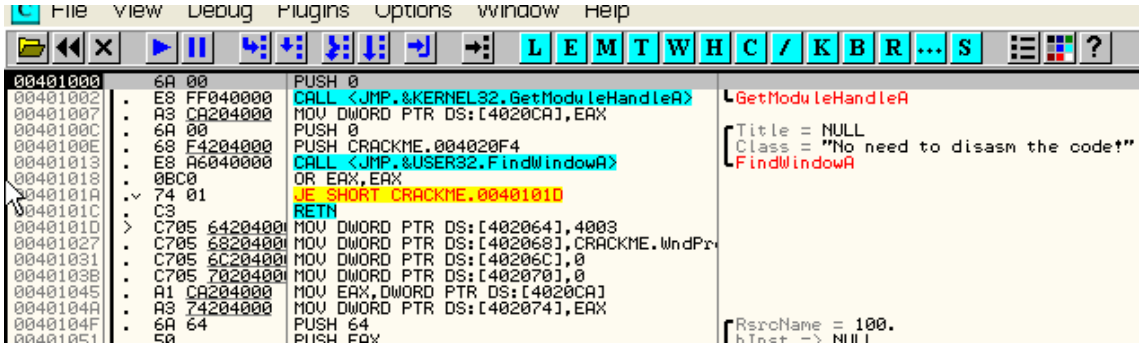
Estos dos últimos flags muestran que el proceso esta siendo debuggeador, son fácilmente localizables, si no recuerdan como hallar la zona del byte IsDebuggerPresent relean la parte 19 que es la que explicábamos como hallar a mano la zona del byte IsDebuggerPresent ya que estos bytes son vecinos de ese.



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.

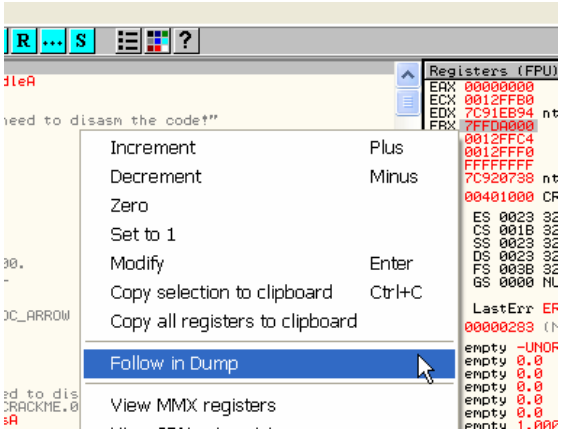


Pues bien, veamos como localizar y cambiar a mano ambos flags.

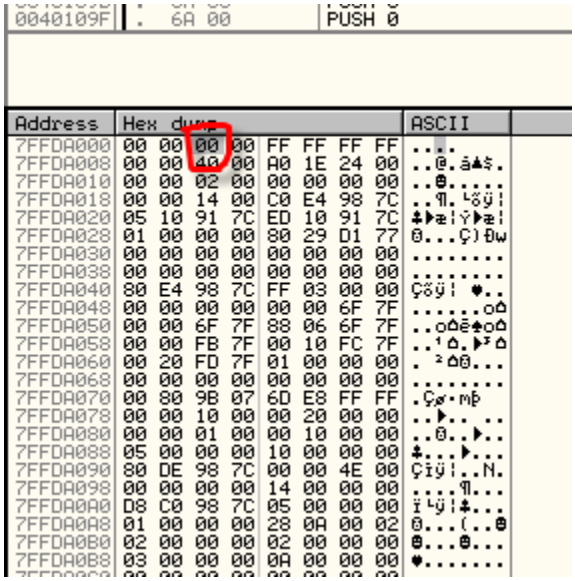
Lo que debemos hallar es la zona del byte IsDebuggerPresent como vimos en el tutorial 19, la forma facil de hallar esa zona es marcar el registro EBX aquí en el ENTRY POINT, y hacer FOLLOW IN DUMP, la forma completa puede leerse en ese tutorial si alguien no la recuerda.

Estoy en el ENTRY POINT

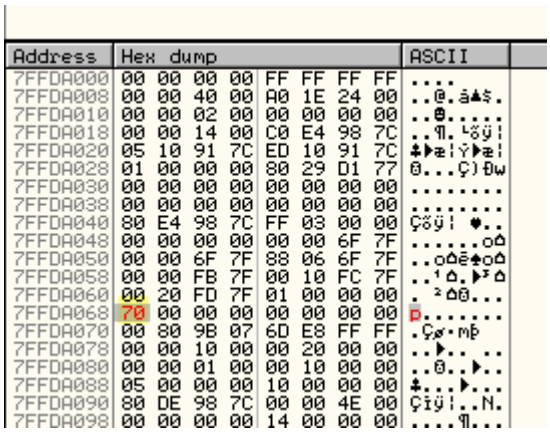
Marco EBX-FOLLOW IN DUMP



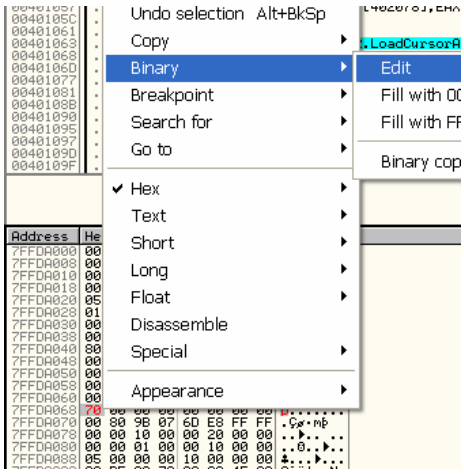
Y en el dump vemos la zona deseada, recuerden que en su maquina puede cambiar la dirección, con respecto a la mía y cada vez que reinicien un programa también variara.



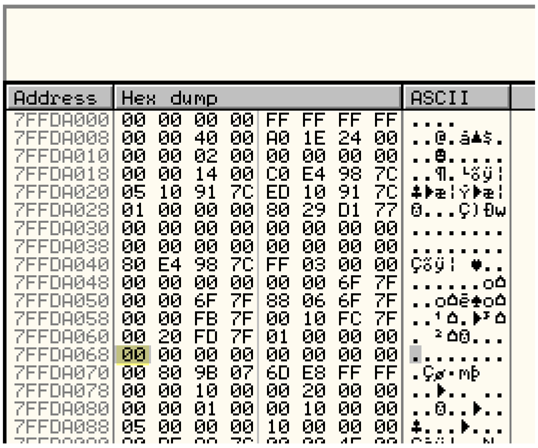
Por supuesto recordamos que ese es el Byte detectado por la api IsDebuggerPresent, pues el NTGlobalFlag es vecino de este lo vemos sumando 68 a la dirección, que nos mostraba EBX, en mi caso EBX era 7ffda000 le sumo 68, será 7ffda068.



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.



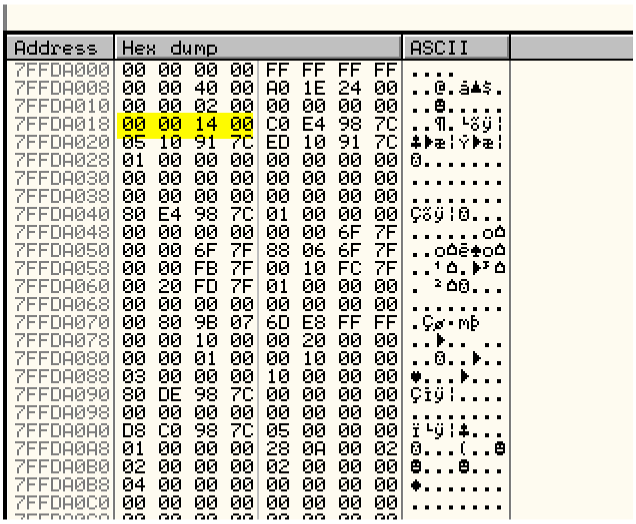
Lo pongo a cero



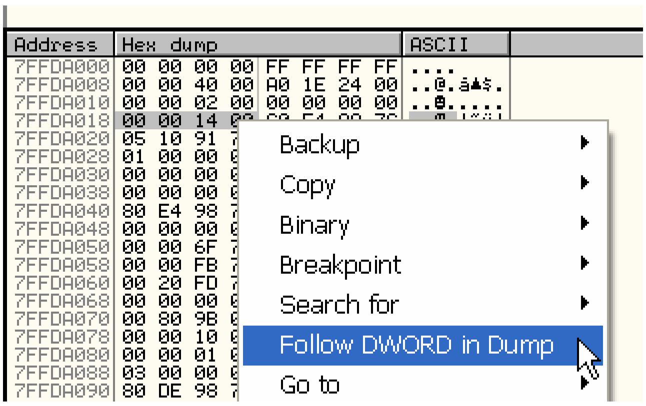
Allí esta localizado y puesto a mano a cero el NtGlobalFlag

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

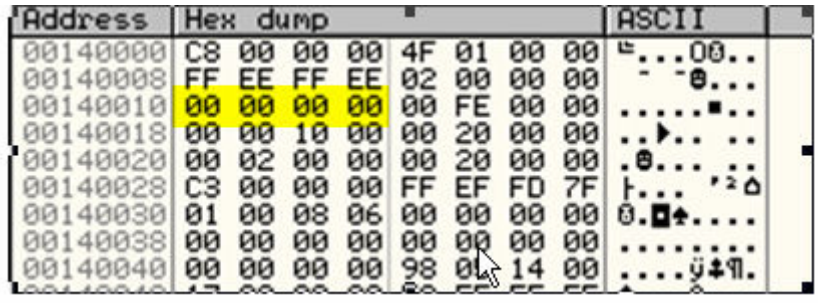
Al valor que me otorgaba EBX en el entry point en este caso le sumo 18, y nos da una dirección que es en mi caso 014000, que es la dirección del heap o sea una zona de memoria creada al arrancar el programa que guarda ciertos datos del mismo, no especificaremos mucho aquí sobre eso.



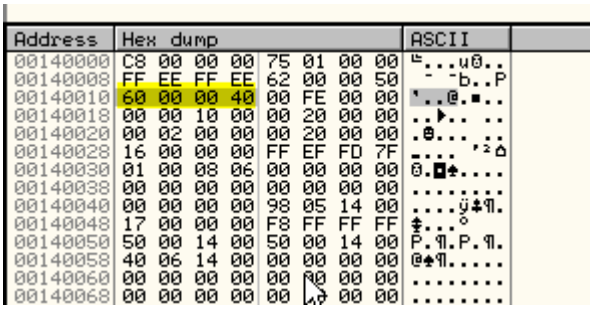
Vayamos a ver esa zona del heap



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

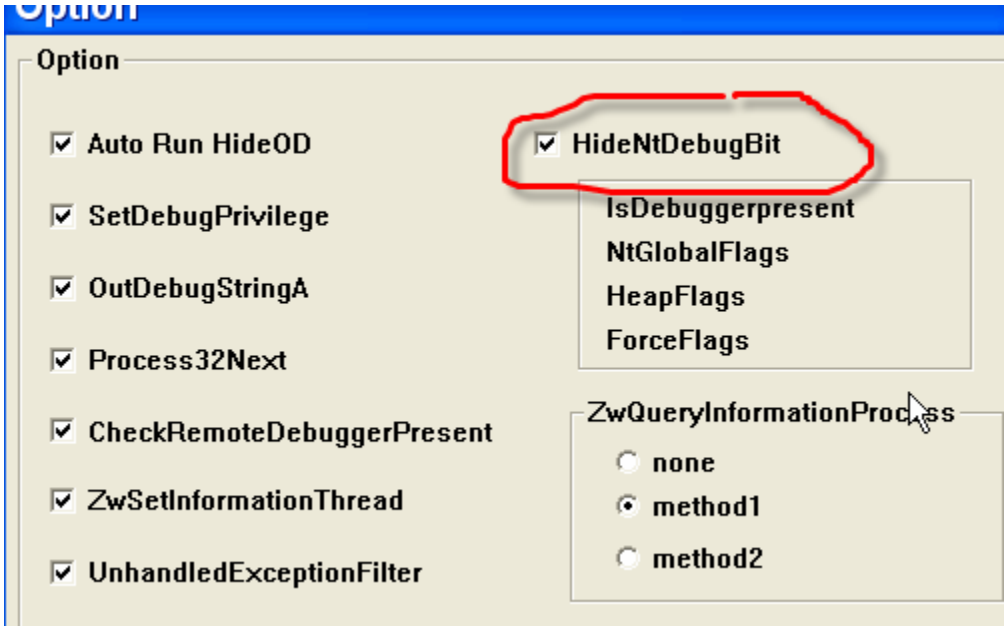


Y sumándole 10 a la dirección base del heap llegamos al dword que esta alli marcado, el cual esta a cero como si no hay debugger, debe ser por tanto plugin que tenemos, si abro el mismo crackme en un ollydbg sin ningún plugin.

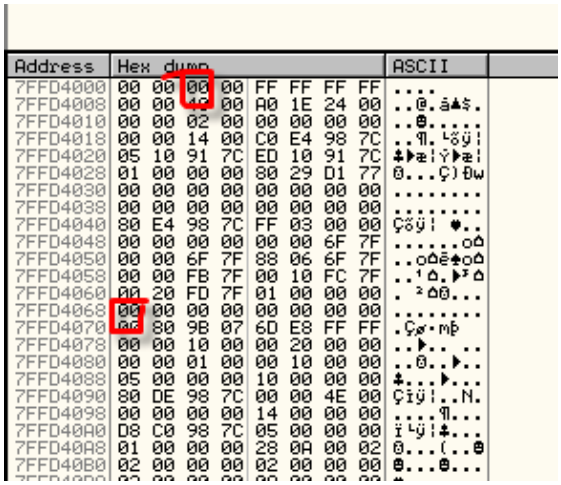


Veo que allí no esta a cero, que es el valor cuando hay debugger quiere decir que entre tanto plugin este byte se pone a cero, por alguno de ellos aun sin estar la tilde marcada.

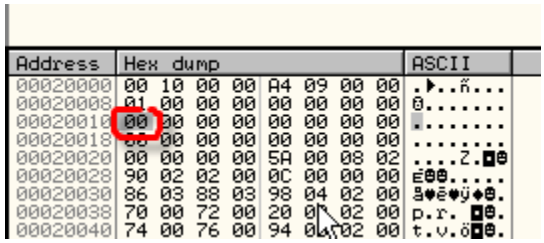
Por lo demás volvamos a poner las tildes en el HideOdb



Y reiniciemos el crackme de cruehead

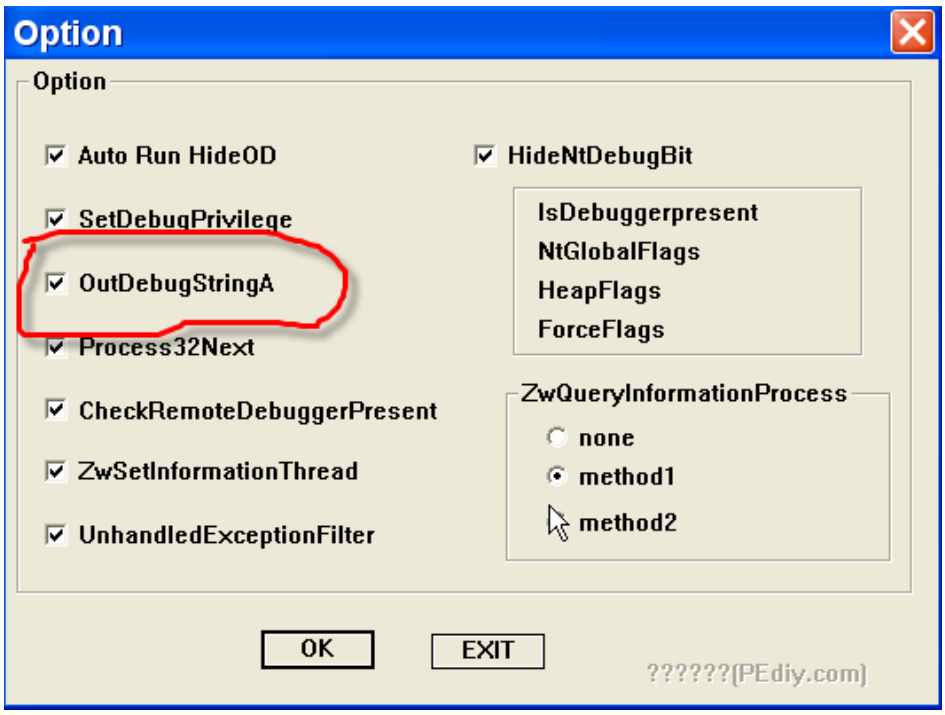


Vemos que parados en el entrypoint tanto el flag de la api IsDebuggerPresent como el NtGlobalFlag están a cero, y si busco el ProcessHeapFlag

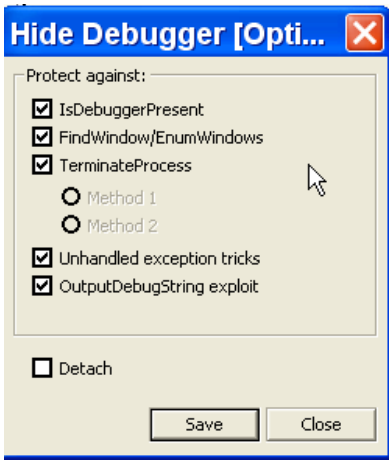


También esta a cero, quiere decir que verifique que el plugin funciona correctamente y que el OLLYDBG no será descubierto por alguno de estos flags, y además aprendimos a hallarlos y cambiarlos a mano.

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

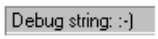


Y



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 execryptos que había 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:



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

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