

INTRODUCCION AL CRACKING CON OLLYDBG DESDE CERO

La idea de esta INTRODUCCION AL CRACKING CON OLLYDBG DESDE CERO es la de dar una base para todos los que recién se inician en el arte del cracking con OLLYDBG, tratando de ser una introducción pero a su vez que proporcione una base fuerte para poder ingresar a la lectura y comprensión de tutoriales mas avanzados como los que se encuentran en el actual NUEVO CURSO de CRACKSLATINOS, el cual por supuesto sigue abierto para seguir agregando novedades, concursos y teorías como hasta ahora.

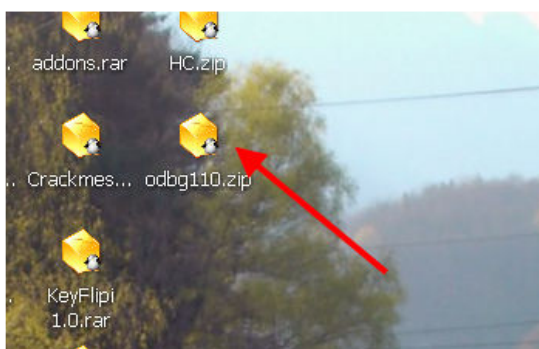
La idea se genero a partir de que los newbies actuales que leen el llamado NUEVO CURSO de CRACKSLATINOS, se encuentran con que este se inicia en un nivel muy alto, y no pueden insertarse gradualmente en el mismo, por lo cual se sienten frustrados y muchas veces abandonan antes de empezar, la idea de esta INTRODUCCION es no repetir los grandes tutes que existen en ese curso que son ya mas de 500 y de un nivel espectacular, si no mas bien sentar la base para que el que termine esta introducción, le sea mas fácil leer cualquier tutorial, obviamente requerirá esfuerzo como todo en el cracking, pero la tarea nuestra es tratar de alivianar ese esfuerzo, sentando aquí las bases del cracking en OLLYDBG para que sea comprensible y se pueda entender fácilmente.

PORQUE OLLYDBG?

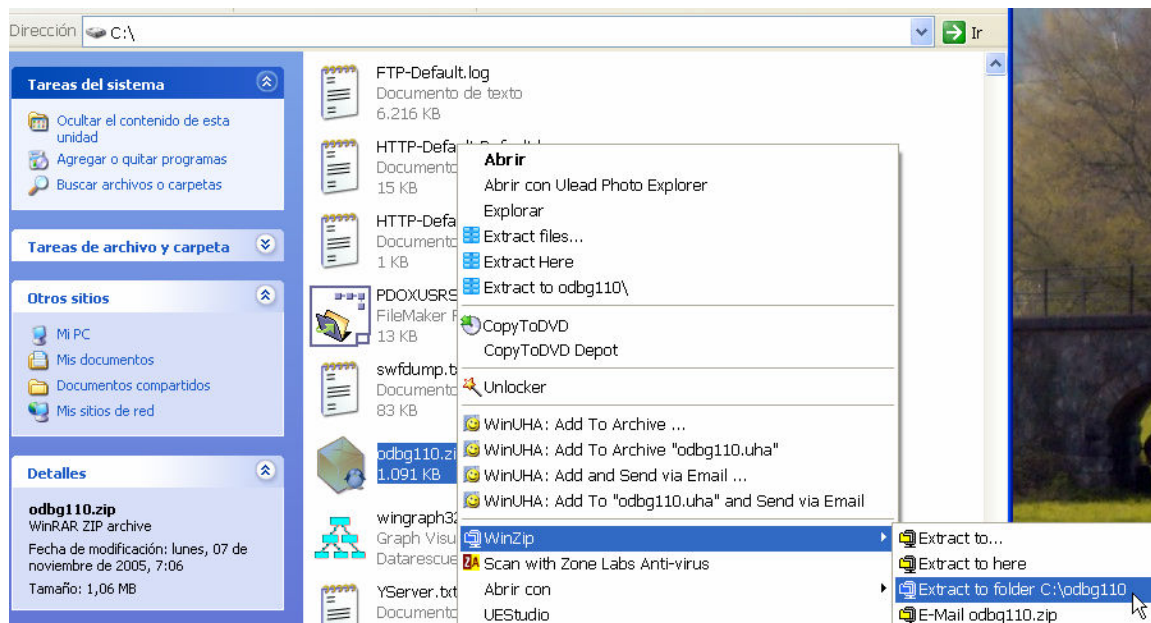
Aquí no entraremos a hacer grandes elucubraciones o reeditar viejas polémicas de SOFTICE vs OLLYDBG de cual es mejor ni nada de eso, creo que hasta los fanáticos de SOFTICE reconocen que es mas sencillo empezar con OLLYDBG, ya que muestra mayor información y es mas cómodo para aprender, la idea es ingresar al mundo del cracking, por la puerta del OLLYDBG, mas adelante cuando uno ya conoce, puede trasladar fácilmente a cualquier debugger lo aprendido pues cambian las formas de usar de los programas, pero no la esencia.

LO PRIMERO ES LO PRIMERO

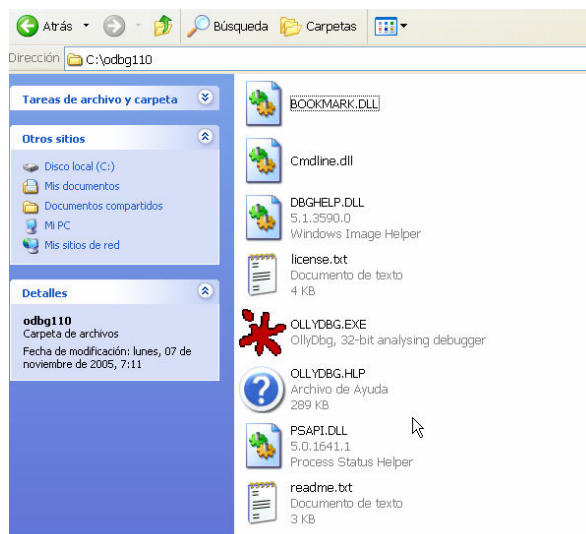
Exactamente lo primero es munirse de la herramienta que vamos a utilizar mayormente, para ello pueden hacer clic <http://www.ollydbg.de/odbg110.zip> para bajarlo.



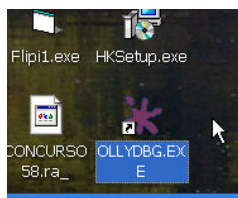
Como aquí estamos empezando desde cero pues, recién nos estamos haciendo del archivo, y ahora ya que es un archivo zipeado, lo unzipearemos con WINZIP preferentemente a una carpeta en nuestro disco rígido que podamos localizar fácilmente, una buena idea seria poner dicha carpeta en C:/ aunque funciona en cualquier lugar, yo la pondré en C:/.



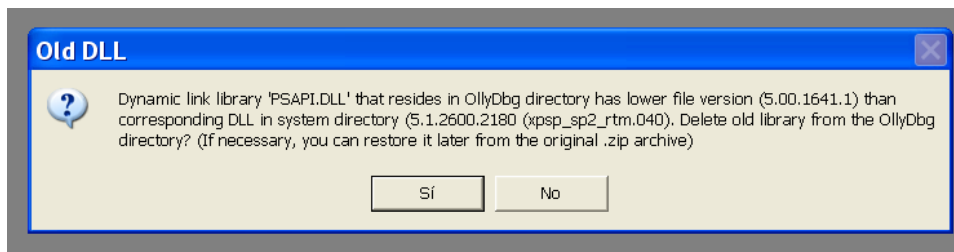
Una vez descomprimido podemos entrar a la carpeta y ver



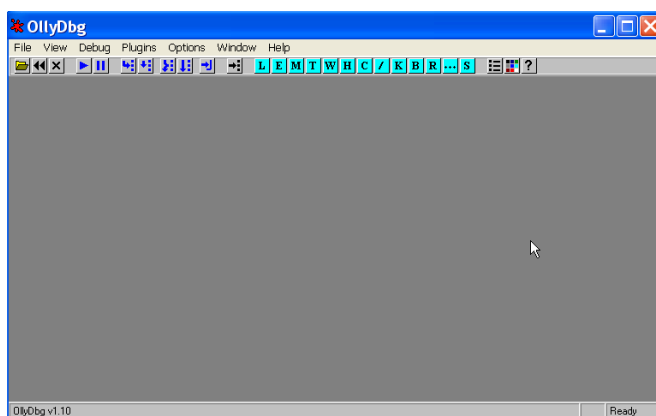
Allí está el archivo ejecutable OLLYDBG.exe el cual ejecutaremos para arrancar el OLLYDBG y al cual para comodidad le haré un acceso directo en mi escritorio.



Bueno ya tenemos bajado y preparado para arrancar a nuestro OLLYDBG.exe, lo ejecutamos.



Nos aparece este cartel avisándonos que la DLL que esta en la carpeta de OLLYDBG es mas antigua que la de sistema, si apretamos SI, borrara la antigua de la carpeta del OLLY y usara la de sistema, yo a pesar de no ver grandes diferencias siempre prefiero elegir que use la propia antes que la de sistema, ya que fue concebido con esa dll, por lo tanto elijo NO.

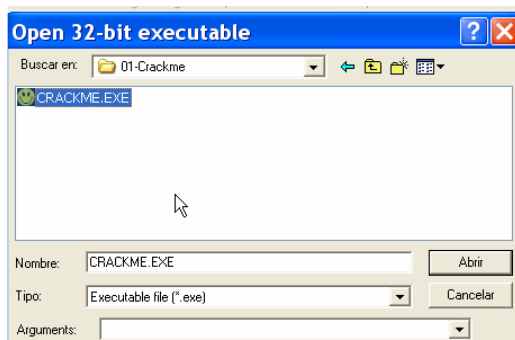


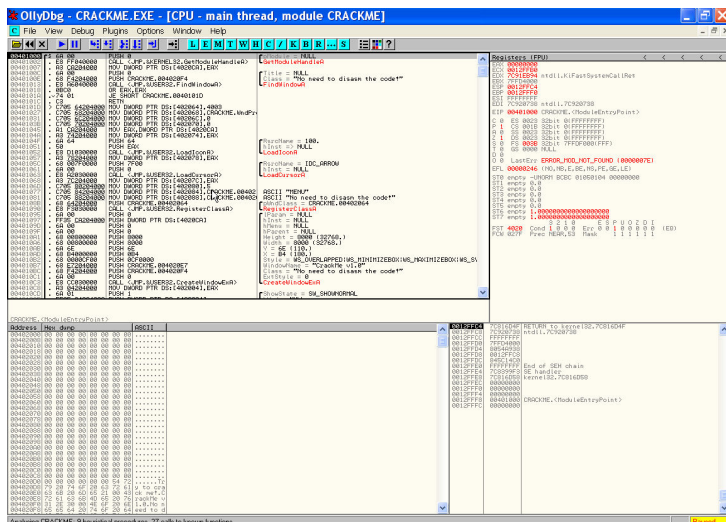
Allí esta el OLLYDBG vacío, y como siempre el primer programa que abriremos mas que nada para mirar las diferentes partes del OLLYDBG y a vuelo de pájaro poder ubicarnos en sus diferente partes, es el famoso CRACKME DE CRUEHEAD que vendrá adjunto en este tutorial.

Para abrir el archivo a debuggear en el OLLYDBG, vamos a FILE OPEN o hacemos clic en el icono

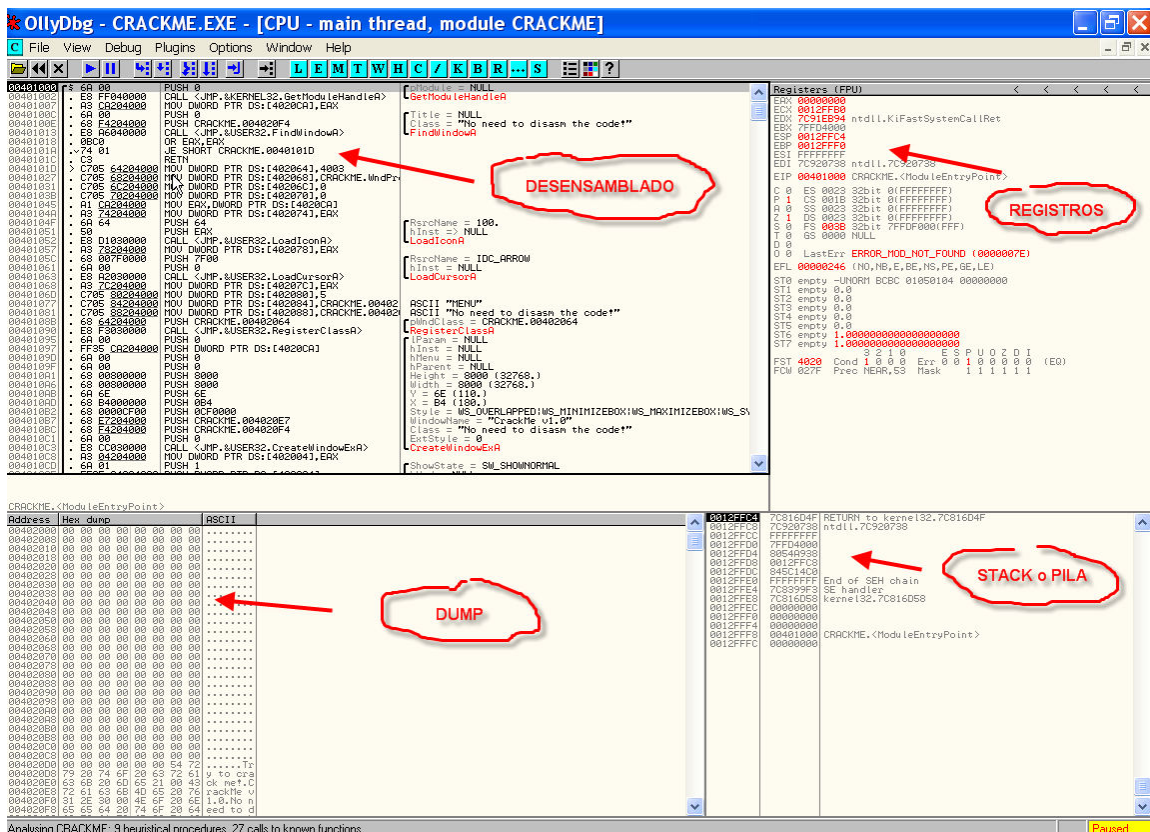


Se abrirá la ventana para que busquemos el archivo a debuggear en este caso es el crackme de CRUEHEAD.





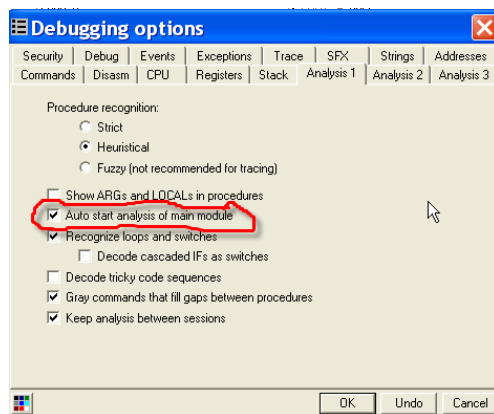
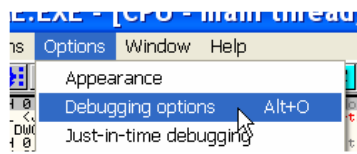
Allí se abre el susodicho crackme y por ahora no importa que no entendamos lo que nos muestra ya mas adelante aprenderemos eso, la idea es ir mostrando las partes del OLLYDBG y ciertas configuraciones del mismo para que cuando en sucesivos tutes, diga, por ejemplo vayan al DUMP, sepan al menos donde esta, así que esto es mas que nada para ubicación, no es un tute profundo sobre OLLY.



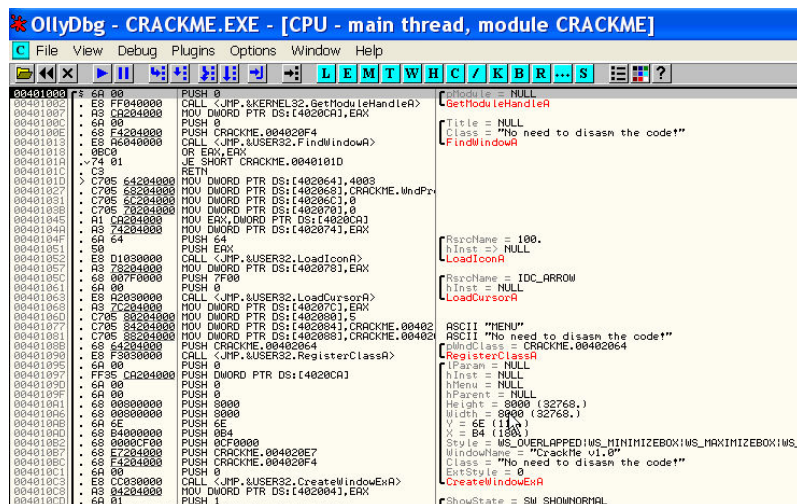
Allí vemos la cuatro partes de la ventana principal del OLLYDBG

1) DESENSAMBLADO :

También llamado listado, aquí el OLLY nos muestra el listado desensamblado del programa que vamos a debuggear, por DEFAULT el OLLY viene configurado para analizar el programa que vamos a debuggear al iniciar, esto se configura en OPTIONS-DEBUGGING OPTIONS.



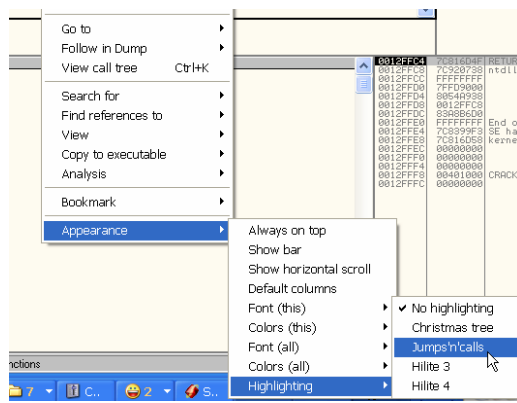
O sea al estar marcada esa tilde en AUTO START ANALYSIS OF MAIN MODULE el OLLYDBG analizara el programa y mostrara información adicional sobre el mismo.



Allí esta el listado inicial del crackme de CRUEHEAD analizado, y si arranca sin analizar debajo podemos ver la diferencia.

0401007	66:2E:6B10 66	INUL DX,WORD PTR CS:[EAX],66	
040100C	CD 72	INT 72	
040100E	1066 4C	ADC BYTE PTR DS:[ESI+4C],AH	
0401011	EA 0F665F5F 0E6	JMP FAR 660E:5F5F660F	Far jump
0401018	DB	???	Unknown command
0401019	0D 0E66A8BD	OR EAX,BDA8660E	
040101E	0C 66	OR AL,66	
0401020	A2 7210663A	MOV BYTE PTR DS:[3A661072],AL	
0401025	F7	???	Unknown command
0401026	0E	PUSH CS	
0401027	66:64:2C 0D	SUB AL,0D	Superfluous prefix
040102B	66:3B9A 0D66C1F	CMPS BX,WORD PTR DS:[EDX+FDC1660D]	
0401032	0E	PUSH CS	
0401033	66:A6	CMPS BYTE PTR DS:[ESI],BYTE PTR ES:[EDI]	
0401035	3E:0E	PUSH CS	Superfluous prefix
0401037	66:05 E3	ADD 0E3	
040103A	0C 66	OR AL,66	
040103C	5A	POP EDI	
040103D	C2 0C66	RETN 660C	
0401040	EC	IN AL,DX	I/O command
0401041	9C	PUSHFD	
0401042	0D 66EEF60E	OR EAX,0EF6EE66	
0401047	66:0D 3F0E	OR AX,0E3F	
040104B	66:699A 0D66F3C1	INUL BX,WORD PTR DS:[EDX+CF3660D],660D	

Otra cosita que hace a la claridad para trabajar y que por lo menos a mi me gusta, aunque cada uno puede variar en estos temas es colorizar los JUMPS Y CALLS eso se hace haciendo clic derecho APPEARENCE – HIGHLIGHTING – JUMPS AND CALLS



El resultado es el siguiente

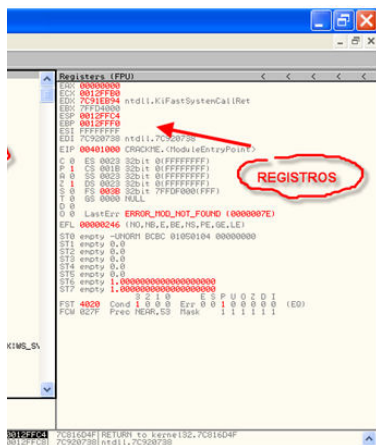
0401000	6A 00	PUSH 0	
0401003	EB FF040000	CALL <JMP.<KERNEL32.GetModuleHandleA>	GetProcAddress
0401007	A3 C8204000	MOV DWORD PTR DS:[4020C8],EAX	
040100C	6A 00	PUSH 0	
040100E	68 F4204000	PUSH CRACKME.004020F4	
0401013	E9 0A040000	CALL <JMP.<USER32.FindWindowA>	FindWindowA
0401018	0BC0	OR EAX,EAX	
040101A	-74 01	JNZ CRACKME.0040101D	
040101C	C3	RET	
040101D	> C705 64204000	MOV DWORD PTR DS:[402064],4000	
0401027	C705 65204000	MOV DWORD PTR DS:[402065],CRACKME.WndPr	
0401031	C705 6C204000	MOV DWORD PTR DS:[40206C],0	
0401036	C705 70204000	MOV DWORD PTR DS:[402070],0	
0401045	A1 05204000	MOV EAX,WORD PTR DS:[402005]	
040104A	A3 74204000	MOV DWORD PTR DS:[402074],EAX	
040104F	6A 64	PUSH 64	
0401051	50	PUSH EAX	
0401052	E8 D1030000	CALL <JMP.<USER32.LoadIconA>	LoadIconA
0401057	A3 70204000	MOV DWORD PTR DS:[402070],EAX	
040105C	68 007F0000	PUSH 7F00	
0401061	6A 00	PUSH 0	
0401063	E8 A2030000	CALL <JMP.<USER32.LoadCursorA>	LoadCursorA
0401065	A3 7C204000	MOV DWORD PTR DS:[40207C],EAX	
040106D	C705 80204000	MOV DWORD PTR DS:[402080],5	
0401077	C705 84204000	MOV DWORD PTR DS:[402084],CRACKME.00402	
0401081	C705 8C204000	MOV DWORD PTR DS:[40208C],CRACKME.00402	
0401086	68 64204000	PUSH CRACKME.00402064	
0401090	E8 F3030000	CALL <JMP.<USER32.RegisterClassA>	RegisterClassA
0401095	6A 00	PUSH 0	
0401097	FF35 C8204000	PUSH DWORD PTR DS:[4020C8]	
040109D	6A 00	PUSH 0	
040109F	6A 00	PUSH 0	
04010A1	EB 0A0A0000	JMP 0A0A	

Allí vemos que en celeste quedan resaltados los CALLS y en amarillo los JUMPS, lo cual es mas claro para la vista.

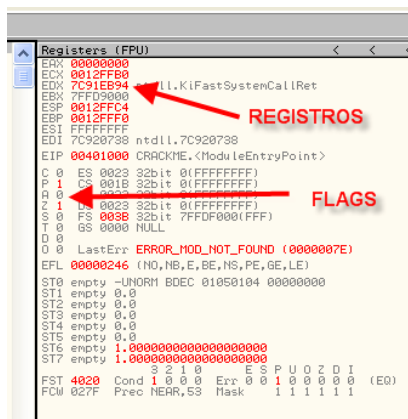
Bueno con eso nuestro listado queda mas fácil de interpretar, aunque aun no tengamos la mas remota idea de que significa, pero bueno hay que preparar antes las herramientas para poder ir de a poco aprendiendo

2)REGISTROS

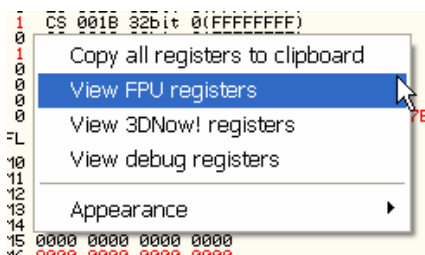
La segunda ventana importante del OLLYDBG es la de los REGISTROS



Recordamos que la ventana de registros se encuentra en la parte superior derecha del OLLYDBG, allí muestra bastante mas información que los registros en si.



Tiene muchísima más información que aun no veremos, pero se puede cambiar el modo de visualización en tres formas. (VIEW FPU REGISTERS, VIEW 3D NOW REGISTERS y VIEW DEBUG REGISTERS) por default viene elegida la primera.

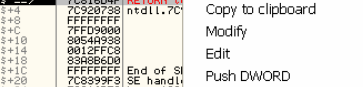


Por ahora no ahondaremos mucho en eso ya que nos preocuparemos más que nada en el tema REGISTROS y FLAGS, lo menciono para que sepan que hay varias vistas en el registro.

3)STACK O PILA:

001F52F0 7C816D4F RETURN to kernel32.7C816D4F
 001F52F2 7C907288 ntdll.7C920788
 001F52F4 FFFFFFFF
 001F52F0 7FD04000
 001F52F2 00540958
 001F52F8 0002FFC8
 001F52FC 84C214C0
 001F5300 FFFFFFFF End of SEH chain
 001F5304 7C399F58 SE handler
 001F53F8 7C816D58 kernel32.7C816D58
 001F53FC 00000000
 001F53F0 00000000
 001F53F4 00000000
 001F53F8 004B1000 CRASHME, {ModuleEntryPoint}
 001F53FC 00000000
 001F53F0 00000000
 001F53F4 00000000
 001F53F8 00000000
 001F53FC 00000000

STACK o PILA



The screenshot shows a debugger window with a list of assembly instructions. A right-click context menu is open over the instruction at address 7C816D58, which is highlighted in blue. The menu options are:

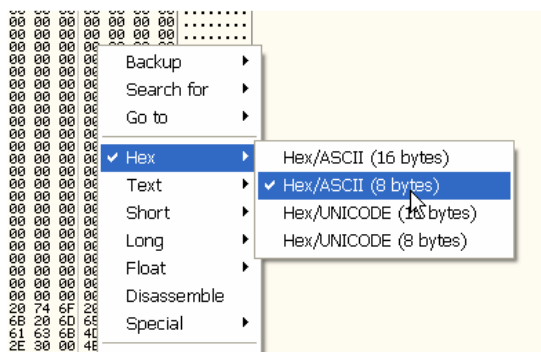
- Copy to clipboard (Ctrl+C)
- Modify
- Edit (Ctrl+E)
- Push DWORD
- Pop DWORD
- Search for address
- Search for binary string (Ctrl+B)
- Go to EBP (highlighted)
- Go to expression (Ctrl+G)
- Follow in Disassembler

The assembly code in the background includes instructions like `ntdll.7C816D4F`, `7C920738`, `FFFFFFF`, `7FFD9000`, `8054A338`, `00127118`, `83A8B6D0`, `FFFFFFFF`, `7C8399F3`, `7C816D58`, `00000000`, `00000000`, `00000000`, `00401000`, and `00000000`. Comments like `SE handl`, `kernel32`, and `CRACKME..` are also visible.

4) DUMP:

[illegible]

La opción por DEFAULT es la que generalmente mas se usa, aunque tenemos opciones para cambiar para mostrar desensamblado (DISASSEMBLE), Texto (TEXT) y diversos formatos (SHORT, LONG, FLOAT)



Y además la opción SPECIAL – PE HEADER que mas adelante en próximos capítulos veremos para que sirve esto que es muy útil.

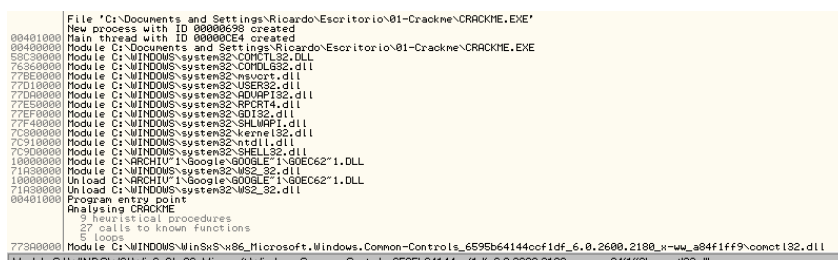


Ya conocemos las partes que se ven en la ventana principal del OLLYDBG, aunque también hay más ventanas que no se ven directamente, se puede acceder a ellas, tanto por el menú, como por los botones de las vistas.

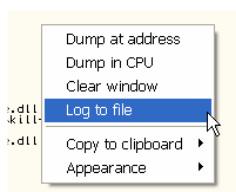


Veremos que es cada uno

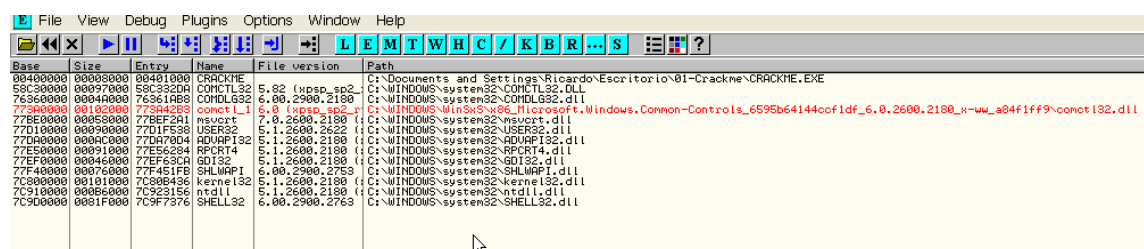
El botón L o VIEW-LOG nos muestra lo que el OLLYDBG escribe en la ventana del LOG lo cual puede ser configurado para mostrar diferentes tipos de información, por default en la ventana del LOG va guardando allí información sobre el arranque, y de la información escrita en el mismo por los diferentes BREAKPOINTS CONDICIONAL LOGS, lo cual se vera mas adelante, por ahora vemos allí ,la información del proceso que arranco, en este caso el crackme de cruehead, las dll que cargo, y ciertos tips sobre el análisis.



Una de las opciones mas importantes de esta ventana es la de logear a una fila, para ciertos casos que deseemos guardar la información en una fila de texto, en ese caso CLICK DERECHO-LOG TO FILE.



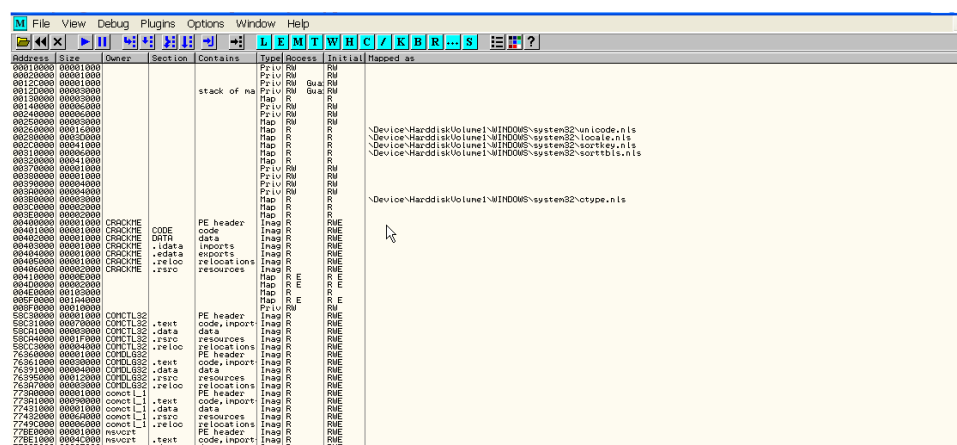
El botón E o VIEW-EXECUTABLES nos muestra la listado de los ejecutables que utiliza el programa, exe, dlls, ocxs, etc



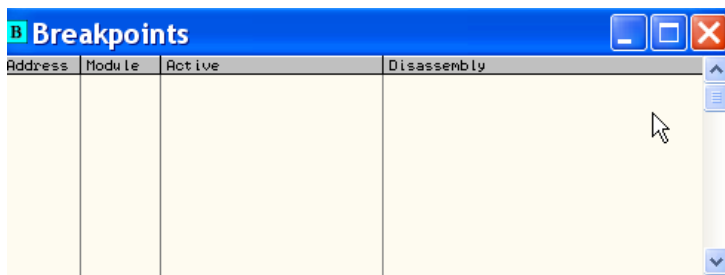
Base	Size	Entry	Name	File version	Path
80400000	00008000	80401000	CRACKME	5.82 (xpsp_sp2_2)	C:\Documents and Settings\Ricardo\Escritorio\01-Crackme\CRACKME.EXE
58C30000	00097000	58C332D8	COMCTL32	6.00,2900,2180	C:\WINDOWS\system32\COMCTL32.DLL
76360000	00044000	76361488	COMDLG32	6.00,2900,2180	C:\WINDOWS\system32\COMDLG32.DLL
77340000	00102000	773442B9	comctl1	6.0 (xpsp_sp2_2)	C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a841ff97-comctl132.dll
77BE0000	00058000	77BEF241	msvcrt	7.0,2600,2180	C:\WINDOWS\system32\msvcrt.dll
77D10000	00098000	77D1F538	USER32	5.1,2600,2652	C:\WINDOWS\system32\USER32.dll
77D30000	0009C000	77D370D4	ADVAPI32	5.1,2600,2180	C:\WINDOWS\system32\ADVAPI32.dll
77E50000	00091000	77E56284	RPCRT4	5.1,2600,2180	C:\WINDOWS\system32\RPCRT4.dll
77EF0000	00046000	77EF65C4	GDI32	5.1,2600,2180	C:\WINDOWS\system32\GDI32.dll
77F40000	00076000	77F451FB	SHLWAPI	6.00,2900,2753	C:\WINDOWS\system32\SHLWAPI.dll
7C900000	00101000	7C90B436	kernel32	5.1,2600,2180	C:\WINDOWS\system32\kernel32.dll
7C910000	00086000	7C923156	ntdll	5.1,2600,2180	C:\WINDOWS\system32\ntdll.dll
7C900000	0001F000	7C9F7376	SHELL32	6.00,2900,2763	C:\WINDOWS\system32\SHELL32.dll

Aquí también el botón derecho tiene muchas opciones que por ahora no veremos ya que estamos mirando en forma general al OLLYDBG.

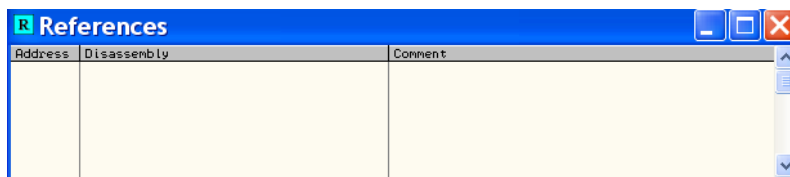
El botón M o VIEW – MEMORY, nos muestra la memoria ocupada por nuestro programa, allí se ven las secciones del ejecutable, dlls que utiliza el proceso, así como el stack y diversas secciones allocadas por el sistema, y muchas veces al correr los programas, los mismos realizan nuevas allocaciones de memoria. En tiempo de ejecución.



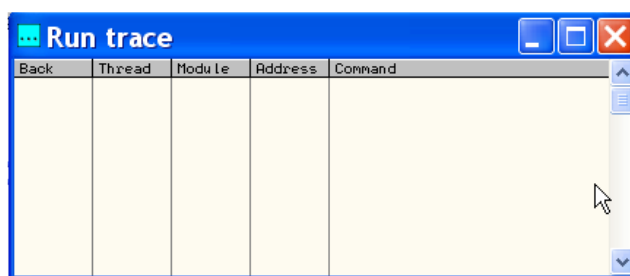
Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
00010000	00001000				Priv	RM	RM	
00020000	00001000				Priv	RM	RM	
00030000	00001000				Priv	RM	RM	
00040000	00001000				Priv	RM	RM	
00050000	00001000				Priv	RM	RM	
00060000	00001000				Priv	RM	RM	
00070000	00001000				Priv	RM	RM	
00080000	00001000				Priv	RM	RM	
00090000	00001000				Priv	RM	RM	
000A0000	00001000				Priv	RM	RM	
000B0000	00001000				Priv	RM	RM	
000C0000	00001000				Priv	RM	RM	
000D0000	00001000				Priv	RM	RM	
000E0000	00001000				Priv	RM	RM	
000F0000	00001000				Priv	RM	RM	
00100000	00001000				Priv	RM	RM	
00110000	00001000				Priv	RM	RM	
00120000	00001000				Priv	RM	RM	
00130000	00001000				Priv	RM	RM	
00140000	00001000				Priv	RM	RM	
00150000	00001000				Priv	RM	RM	
00160000	00001000				Priv	RM	RM	
00170000	00001000				Priv	RM	RM	
00180000	00001000				Priv	RM	RM	
00190000	00001000				Priv	RM	RM	
001A0000	00001000				Priv	RM	RM	
001B0000	00001000				Priv	RM	RM	
001C0000	00001000				Priv	RM	RM	
001D0000	00001000				Priv	RM	RM	
001E0000	00001000				Priv	RM	RM	
001F0000	00001000				Priv	RM	RM	
00200000	00001000				Priv	RM	RM	
00210000	00001000				Priv	RM	RM	
00220000	00001000				Priv	RM	RM	
00230000	00001000				Priv	RM	RM	
00240000	00001000				Priv	RM	RM	
00250000	00001000				Priv	RM	RM	
00260000	00001000				Priv	RM	RM	
00270000	00001000				Priv	RM	RM	
00280000	00001000				Priv	RM	RM	
00290000	00001000				Priv	RM	RM	
002A0000	00001000				Priv	RM	RM	
002B0000	00001000				Priv	RM	RM	
002C0000	00001000				Priv	RM	RM	
002D0000	00001000				Priv	RM	RM	
002E0000	00001000				Priv	RM	RM	
002F0000	00001000				Priv	RM	RM	
00300000	00001000				Priv	RM	RM	
00310000	00001000				Priv	RM	RM	
00320000	00001000				Priv	RM	RM	
00330000	00001000				Priv	RM	RM	
00340000	00001000				Priv	RM	RM	
00350000	00001000				Priv	RM	RM	
00360000	00001000				Priv	RM	RM	
00370000	00001000				Priv	RM	RM	
00380000	00001000				Priv	RM	RM	
00390000	00001000				Priv	RM	RM	
003A0000	00001000				Priv	RM	RM	
003B0000	00001000				Priv	RM	RM	
003C0000	00001000				Priv	RM	RM	
003D0000	00001000				Priv	RM	RM	
003E0000	00001000				Priv	RM	RM	
003F0000	00001000				Priv	RM	RM	
00400000	00001000				Priv	RM	RM	
00410000	00001000				Priv	RM	RM	
00420000	00001000				Priv	RM	RM	
00430000	00001000				Priv	RM	RM	
00440000	00001000				Priv	RM	RM	
00450000	00001000				Priv	RM	RM	
00460000	00001000				Priv	RM	RM	
00470000	00001000				Priv	RM	RM	
00480000	00001000				Priv	RM	RM	
00490000	00001000				Priv	RM	RM	
004A0000	00001000				Priv	RM	RM	
004B0000	00001000				Priv	RM	RM	
004C0000	00001000				Priv	RM	RM	
004D0000	00001000				Priv	RM	RM	
004E0000	00001000				Priv	RM	RM	
004F0000	00001000				Priv	RM	RM	
00500000	00001000				Priv	RM	RM	
00510000	00001000				Priv	RM	RM	
00520000	00001000				Priv	RM	RM	
00530000	00001000				Priv	RM	RM	
00540000	00001000				Priv	RM	RM	
00550000	00001000				Priv	RM	RM	
00560000	00001000				Priv	RM	RM	
00570000	00001000				Priv	RM	RM	
00580000	00001000				Priv	RM	RM	
00590000	00001000				Priv	RM	RM	
005A0000	00001000				Priv	RM	RM	
005B0000	00001000				Priv	RM	RM	
005C0000	00001000				Priv	RM	RM	
005D0000	00001000				Priv	RM	RM	
005E0000	00001000				Priv	RM	RM	
005F0000	00001000				Priv	RM	RM	
00600000	00001000				Priv	RM	RM	
00610000	00001000				Priv	RM	RM	
00620000	00001000				Priv	RM	RM	
00630000	00001000				Priv	RM	RM	
00640000	00001000				Priv	RM	RM	
00650000	00001000				Priv	RM	RM	
00660000	00001000				Priv	RM	RM	
00670000	00001000				Priv	RM	RM	
00680000	00001000				Priv	RM	RM	
00690000	00001000				Priv	RM	RM	
006A0000	00001000				Priv	RM	RM	
006B0000	00001000				Priv	RM	RM	
006C0000	00001000				Priv	RM	RM	
006D0000	00001000				Priv	RM	RM	
006E0000	00001000				Priv	RM	RM	
006F0000	00001000				Priv	RM	RM	
00700000	00001000				Priv	RM	RM	
00710000	00001000				Priv	RM	RM	
00720000	00001000				Priv	RM	RM	
00730000	00001000				Priv	RM	RM	
00740000	00001000				Priv	RM	RM	
00750000	00001000				Priv	RM	RM	
00760000	00001000				Priv	RM	RM	
00770000	00001000				Priv	RM	RM	
00780000	00001000				Priv	RM	RM	
00790000	00001000				Priv	RM	RM	
007A0000	00001000				Priv	RM	RM	
007B0000	00001000				Priv	RM	RM	
007C0000	00001000				Priv	RM	RM	
007D0000	00001000				Priv	RM	RM	
007E0000	00001000				Priv	RM	RM	
007F0000	00001000				Priv	RM	RM	
00800000	00001000				Priv	RM	RM	
00810000	00001000				Priv	RM	RM	
00820000	00001000				Priv	RM	RM	
00830000	00001000				Priv	RM	RM	
00840000	00001000				Priv	RM	RM	
00850000	00001000				Priv	RM	RM	
00860000	00001000				Priv	RM	RM	
00870000	00001000				Priv	RM	RM	
00880000	00001000				Priv	RM	RM	
00890000	00001000				Priv	RM	RM	
008A0000	00001000				Priv	RM	RM	
008B0000	00001000				Priv	RM	RM	



El botón R o VIEW- REFERENCES nos muestra la ventana de referencias la cual nos da los resultados de cuando hacemos una búsqueda de referencias en el OLLY



El botón ... o VIEW-RUN TRACE, nos muestra el listado si hemos hecho algún RUN TRACE en nuestra maquina, y tiene también la posibilidad de elegir LOG TO FILE, para guardar el resultado del traceo en un archivo txt

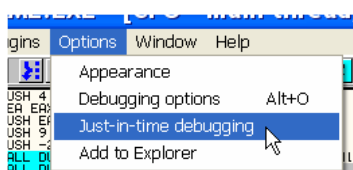


Bueno hasta aquí un paneo a vuelo de pájaro por los botones mas importantes, no detallamos explicación porque aun hay que aprender antes algo de ASM, y practicando el uso del OLLYDBG podremos ir aclarando mas profundamente para que sirve cada botón y cada OPCION, la idea es irse familiarizándose con donde están las cosas que veremos en las próximas entregas.

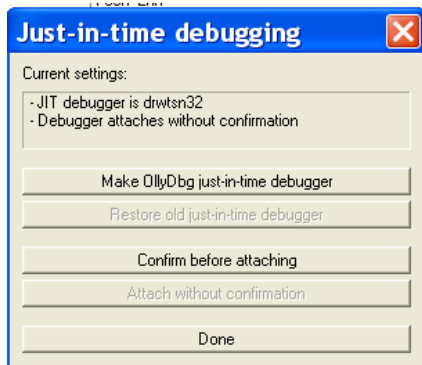
COMO CONFIGURAR EL OLLYDBG COMO JIT (JUST IN TIME DEBUGGER)

Aclaro que no conviene tener configurado el OLLYDBG constantemente COMO JIT, solo conviene hacerlo en ocasiones especiales, ya que al estar como JIT capturara el error de cualquier programa de nuestra maquina y arrancara solo, lo cual puede resultar molesto si no estamos debuggeando o crackeando, por lo tanto les enseño como se configura para casos especiales, pero conviene dejarlo con la opción que trae por default que no esta como JIT.

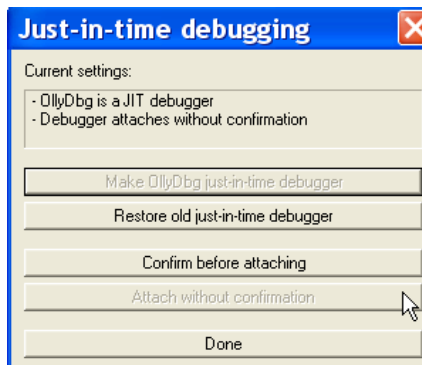
Para colocar el OLLYDBG como JIT vamos a OPTIONS-JUST IN TIME DEBUGGING



Y aprieto el botón MAKE OLLYDBG JUST IN TIME DEBUGGER y DONE



Para quitarlo, en el mismo lugar aprieto RESTORE JUST IN TIME DEBUGGER y DONE



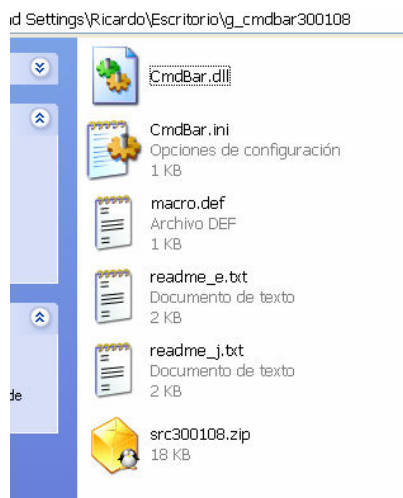
Agregando PLUGINS al OLLYDBG

El OLLYDBG trae la opción para agregar plugins que nos son necesarios para realizar cierta tarea, por ahora solo agregaremos el plugin COMMAND BAR para aprender como se agregan los mismos.

Bajamos el plugin COMMAND BAR el cual puede ser bajado de [AQUÍ](#) y la mayoría de los plugins se hallan [AQUI](#)

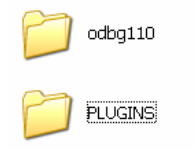


Allí esta bajado en mi escritorio el plugin lo descomprimo con WINZIP entro a la carpeta que descomprimí a ver el contenido



Ahora antes que nada crearemos una carpeta para los PLUGINS en nuestra maquina, yo la creare en C:/ y la llamare PLUGINS nada mas.

Voy a C y creo una NUEVA CARPETA

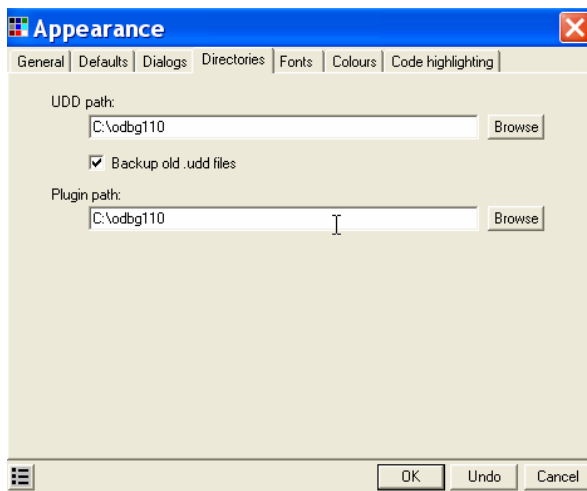


Allí esta, puede estar ubicada en cualquier lugar, pero a mi me gusta tener todo en C por eso la coloque allí, de cualquier forma debemos configurar el OLLYDBG para que reconozca esta carpeta como la que tendrá los plugins.

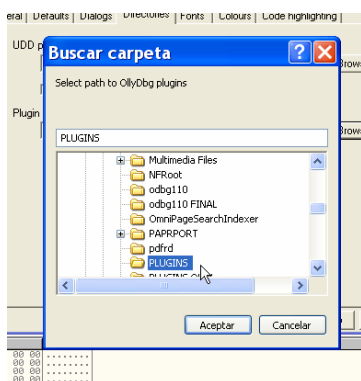
Para ello en el OLLYDBG vamos a OPTIONS-APPEARANCE



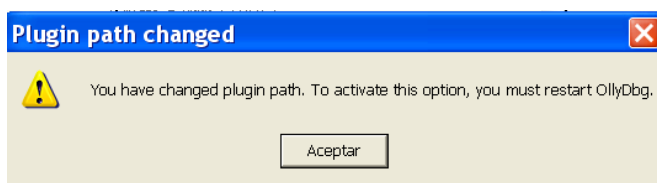
Y en la ventana que se abre vamos a la pestaña DIRECTORIES



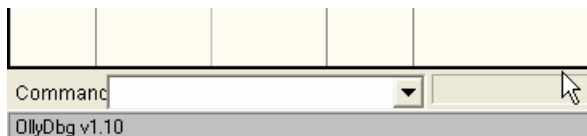
Vemos que en donde apunta al path de los plugins (PLUGIN PATH), en realidad nos esta apuntando a la carpeta donde esta el OLLYDBG.exe y podría dejarlo allí, pero a mi me gusta tener los plugins separados por lo tanto en donde dice PLUGIN PATH- BROWSE busco la carpeta que cree para mis plugins.



Allí elegí la carpeta PLUGINS que cree y sale este aviso



O sea que debo reiniciar el OLLY para que me reconozca la nueva carpeta de plugins, pero antes copio el contenido que baje del comand bar a mi carpeta de plugins.



Es una barra para tipear comandos que nos facilitara mucho las cosas, mas adelante veremos su uso, por ahora lo importante es saber agregar plugins.

Para quitar cualquier PLUGIN con solo quitar la dll correspondiente de nuestra carpeta PLUGINS y reiniciar el OLLYDBG, desaparecerá, les aconsejo que dejen siempre activa la COMMAND BAR.

Arranco nuevamente el crackme de CRUEHEAD EN OLLYDBG

Las teclas mas usadas en el OLLYDBG son:

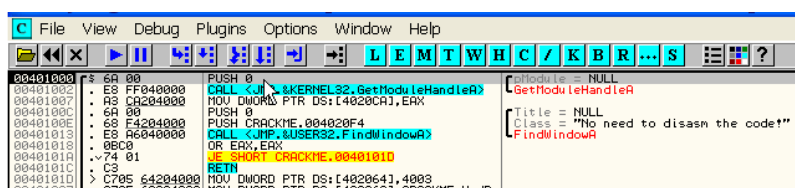
F7: Ejecuta una sola línea de código (si estas en un CALL entra al mismo a ejecutarlo por dentro)

F8: Ejecuta una sola línea de código (si estas en un CALL no entra al mismo lo ejecuta completo sin entrar y sigue en la siguiente línea luego del CALL)

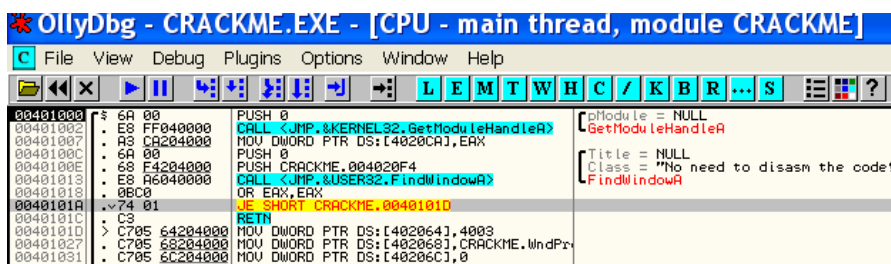
Esos dos formas de tracear manualmente son verdaderamente diferentes y según cada caso usaremos F7 o F8 lo cual veremos más adelante.

F2: Coloca un Breakpoint COMUN en la línea que marcas con el Mouse o esta grisada en el listado, para quitar el BP apretas nuevamente F2.

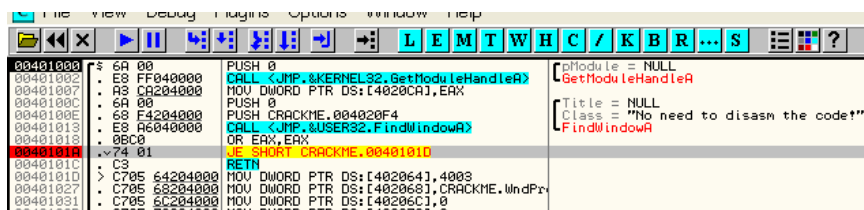
Por ejemplo:



Quiero poner un BP en 40101A pues marco con el Mouse esa linea

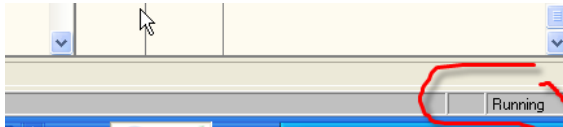


Al hacer clic una sola vez se marca y queda grisada como vemos en la imagen, luego apreto F2.

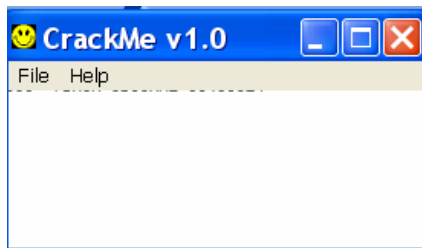


Vemos que se pinta de rojo la zona de la dirección, eso significa que hay activo un BP o Breakpoint allí, si apreto F2 nuevamente se quita.

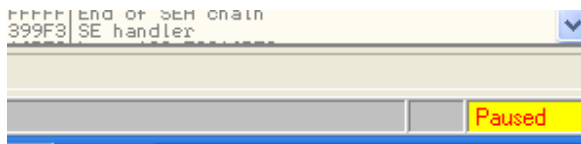
F9: Para Correr el programa es similar a RUN, con esto el programa correrá, hasta que encuentre algún BREAKPOINT, o alguna EXCEPCION que lo detenga o FINALICE por algún motivo, al apretar RUN veremos en la esquina inferior del OLLYDBG la palabra RUNNING o sea que esta CORRIENDO.



Allí arranco el CRACKME DE CRUEHEAD, lo podemos ver correr

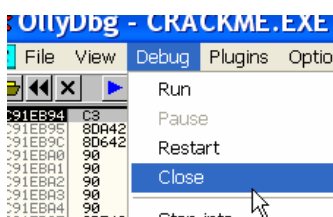


Si PAUSO la ejecución en OLLYDBG apretando F12 o DEBUG -PAUSE



Vemos que el OLLYDBG cambia a mostrar PAUSED o sea que esta PAUSADO, podemos volver a hacerlo correr con F9 o DEBUG-RUN.

Para cerrar el programa que esta siendo DEBUGGEADO apreto DEBUG-CLOSE



Bueno esto a sido una mirada a vuelo de pájaro del OLLYDBG la cual profundizaremos mas adelante pues tiene muchísimas opciones y configuraciones las cuales seguiremos estudiando en las próximas entregas, es muy útil que bajen el programa lo configuren y miren donde están las cosas que muestra este tute, así como le agreguen el plugin para practicar, y hagan correr y pausar el CRACKME DE CRUEHEAD, prueben ponerle un Breakpoint y practiquen esas cosas para que en la segunda entrega estén mas familiarizados con el mismo y podamos avanzar lento pero seguro, y sin dudas.

Un abrazo a todos los CRACKSLATINOS

Hasta la parte 2

Ricardo Narvaja

07 de noviembre de 2005