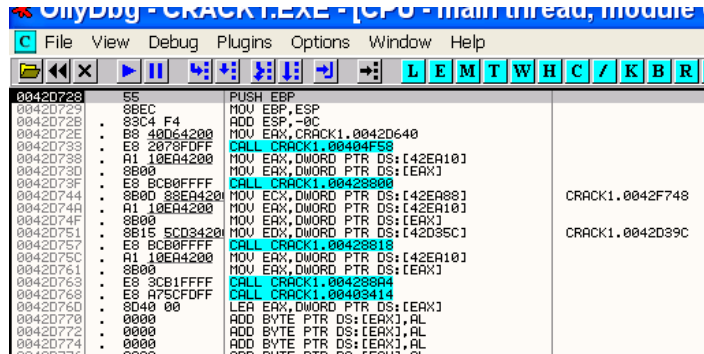


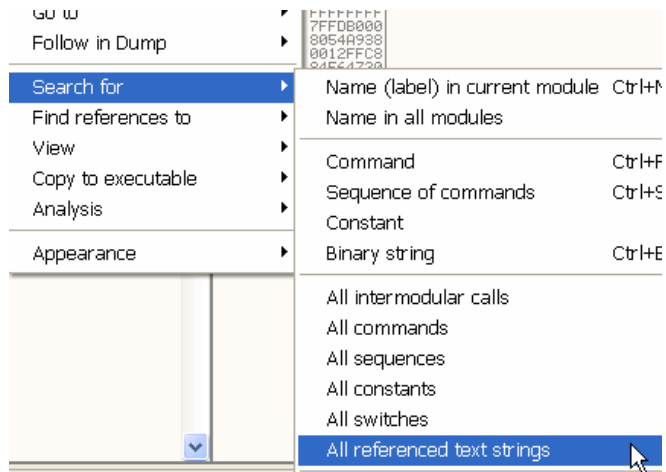
INTRODUCCION AL CRACKING CON OLLYDBG PARTE 17

Bueno antes que nada, vamos a solucionar el crackme MEXCRK1.ZIP pendiente que es muy sencillo.

Abrámoslo en OLLYDBG y por supuesto para en el Entry Point



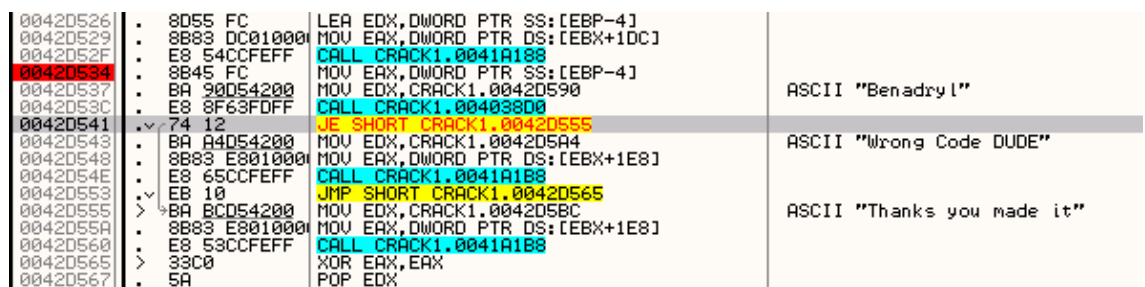
Veamos las strings referentes que utiliza el programa, haciendo click derecho



Se ven entre las strings las siguientes

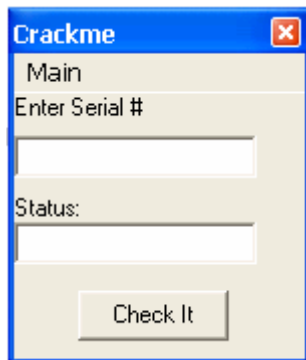


Podemos hacer click en la de correcto o en la de incorrecto, para ver la zona caliente.

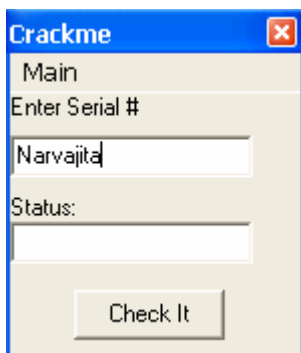


Vemos allí la zona caliente, un Call y dentro se decide ya que a la vuelta del mismo hay un JE que salta a **WRONG CODE DUDE** o a **THANKS YOU MADE IT**, o sea que es el salto decisivo.

Pongo un BPX un poco antes del CALL en 42d534, y doy RUN



Allí tipo mi serial falso, en la parte superior donde dice ENTER SERIAL #



En mi caso tipoo Narvajita.

0042D529	8B83 001000	MOV EAX,DWORD PTR DS:[EBX+100]	
0042D52F	E8 54CCFEFF	CALL CRACK1.0041A188	
0042D534	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
0042D537	BA 90D54200	MOV EDX,CRACK1.0042D590	ASCII "Benadryl"
0042D53C	E8 8F63FDFF	CALL CRACK1.004038D0	
0042D541	74 12	JE SHORT CRACK1.0042D555	
0042D543	BA A4D54200	MOV EDX,CRACK1.0042D5A4	ASCII "Wrong Code DUDE"
0042D548	8B83 E8010000	MOV EAX,DWORD PTR DS:[EBX+1E8]	
0042D54E	E8 65CCFEFF	CALL CRACK1.0041A188	
0042D553	EB 10	JMP SHORT CRACK1.0042D565	
0042D555	BA BCD54200	MOV EDX,CRACK1.0042D5BC	ASCII "Thanks you made it"
0042D55A	8B83 E8010000	MOV EAX,DWORD PTR DS:[EBX+1E8]	
0042D560	E8 53CCFEFF	CALL CRACK1.0041A188	
0042D565	33C0	XOR EAX,EAX	
0042D567	EB 00	RET	

Si entro traceando dentro del CALL veo que compara mi serial falso Narvajita, con la string Benadryl.

004038F9	8B0E	MOV ECX,DWORD PTR DS:[ESI]	
004038FB	8B1F	MOV EBX,DWORD PTR DS:[EDI]	
004038FD	39D9	CMP ECX,EBX	
004038FF	75 58	JNZ SHORT CRACK1.00403959	
EBP	0012F9B0		
ESI	00925948	ASCII "Narvajita"	
EDI	0042D590	ASCII "Benadryl"	
EIP	004038F9	CRACK1.004038F9	

Bueno ya llegamos a la comparación, así que el serial correcto es la palabra Benadryl si damos RUN nuevamente quitando todos los BPXs



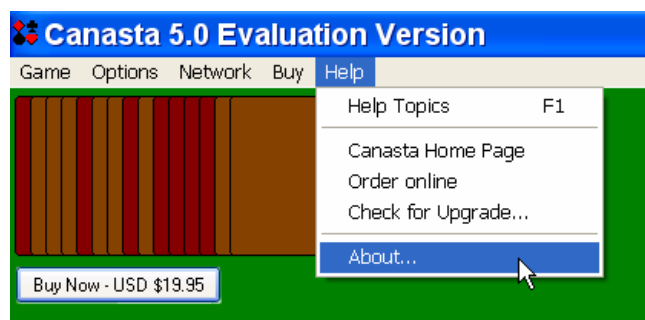
Allí vemos como nos felicita por haber hallado el serial correcto y felicitaciones porque si llegaron aquí también lo han sacado ustedes.

Bueno vamos a hacer el ultimo user, name ya que en la próxima parte comenzaremos con un nuevo tema.

Uno de los lectores de este curso, tuvo problemas con un programita que en realidad es un juego que se llama canasta 5.0 y que ira adjunto al tutorial.

El me comentaba que se le quemaron los papeles porque siempre, en los tutes vemos casos de user y name que se ingresan apretando un botón y en este caso, solo hay que tipear y si lo que tipeas es correcto, el botón OK se habilita pues esta deshabilitado.

Si instalamos el programa y vamos al about



Y si vale 20 dolares amigo, jeje, oferta 19,95.

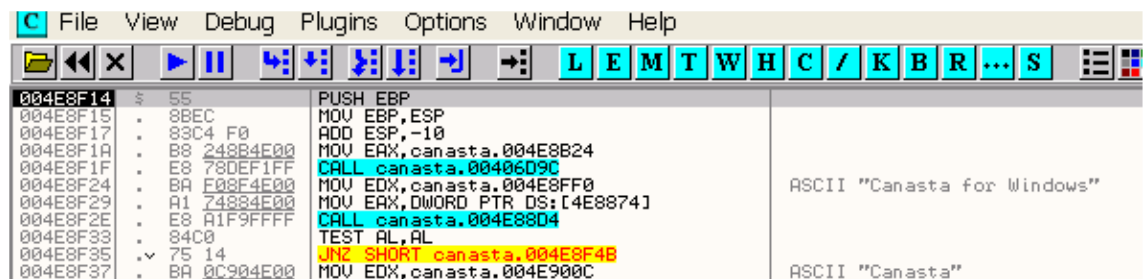


Vemos el botón ENTER LICENSE

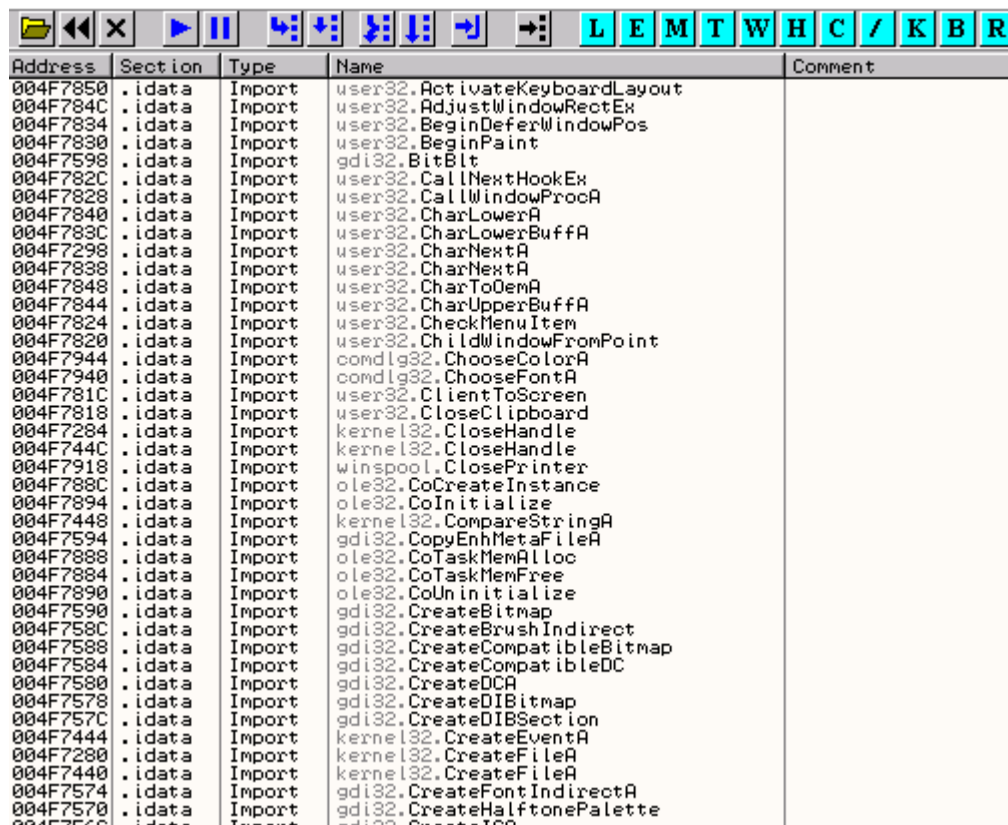


Allí dice que el botón OK se habilitara cuando se coloque la combinación correcta jeje.

Arranquémoslo en OLLYDBG.



Allí vemos el ENTRY POINT



Uff la lista de apis es larguísima y las strings

Address	Disassembly	Text string
004B40E5	ASCII "als/RTD1"	
004B40F1	ASCII "Error"	
004B4130	DD canasta.004B417C	ASCII 08,"TLlicense"
004B4150	DD canasta.004B417C	ASCII 08,"TLlicense"
004B417D	ASCII "TLlicense"	
004B4188	DD canasta.004B41D4	ASCII 0F,"TLlicenseManager"
004B41A8	DD canasta.004B41D4	ASCII 0F,"TLlicenseManager"
004B41D5	ASCII "TLlicenseManager"	
004B4204	MOV ESI,canasta.004B423C	ASCII 11,"Unregistered copy"
004B423D	ASCII "Unregistered cop"	
004B424D	ASCII "y"	
004B42A3	MOV EDX,canasta.004B4370	ASCII "Settings\"
004B42D0	MOV EDX,canasta.004B4370	ASCII "Settings\"
004B4370	ASCII "Settings\",0	
004B43CF	MOV EDX,canasta.004B446C	ASCII "Settings\"
004B4400	MOV EDX,canasta.004B446C	ASCII "Settings\"
004B446C	ASCII "Settings\",0	
004B4484	MOV ESI,canasta.004B44A4	ASCII 11,"Unregistered copy"
004B44A5	ASCII "Unregistered cop"	
004B44B5	ASCII "y"	
004B462D	MOV EDX,canasta.004B4690	ASCII "Unregistered copy"
004B4690	ASCII "Unregistered cop"	
004B46A0	ASCII "y",0	
004B46CF	MOV EDX,canasta.004B46E8	ASCII "Unregistered copy"
004B46E8	ASCII "Unregistered cop"	
004B46F8	ASCII "y",0	
004B47EB	MOV EDX,canasta.004B4800	ASCII "Settings\DateFormat"
004B4804	PUSH canasta.004B489C	ASCII "Registration Key"
004B4813	MOV ECX,canasta.004B48B8	ASCII "User Name"
004B4823	PUSH canasta.004B48CC	ASCII "CodePageEx"
004B4832	MOV ECX,canasta.004B48E0	ASCII "CodePage"
004B4880	ASCII "Settings\DateFor"	
004B4890	ASCII "mat",0	
004B489C	ASCII "Registration Key"	
004B48AC	ASCII 0	
004B48B8	ASCII "User Name",0	
004B48CC	ASCII "CodePageEx",0	
004B48E0	ASCII "CodePage",0	
004B493E	MOV EDX,canasta.004B49C4	ASCII "Settings\DateFormat"
004B495C	PUSH canasta.004B49E0	(Initial CPU selection)
004B496B	MOV ECX,canasta.004B49FC	ASCII "User Name"
004B497B	PUSH canasta.004B4A10	ASCII "CodePageEx"
004B498A	MOV ECX,canasta.004B4A24	ASCII "CodePage"
004B49C4	ASCII "Settings\DateFor"	
004B49D4	ASCII "mat",0	
004B49E0	ASCII "Registration Key"	
004B49F0	ASCII 0	
004B49FC	ASCII "User Name",0	
004B4A10	ASCII "CodePageEx",0	
004B4A24	ASCII "CodePage",0	
004B4AC6	MOV EAX,canasta.004B4B24	ASCII "LICENSE INVALID"
004B4B24	ASCII "LICENSE INVALID",0	
004B4BB8	MOV EDX,canasta.004B4D70	ASCII "http://www.canasta.net/keychecker/"
004B4BE0	MOV EAX,canasta.004B4D9C	ASCII "000"
004B4C0A	MOV EAX,canasta.004B4DA8	ASCII "yyyy/mm/dd"
004B4C2D	MOV EAX,canasta.004B4DA8	ASCII "yyyy/mm/dd"
004B4C49	MOV EAX,canasta.004B4DBC	ASCII "app=%s&name=%s&postfix=%s&key=%s&instal
004B4D70	ASCII "http://www.canas	
004B4D80	ASCII "ta.net/keychecke"	

Hay por ahí algunas pero obviamente el botón OK se debe habilitar y si no, no muestra nada así que el tema de las strings lo dejamos jeje.

Enter License Information...

License Name (type exactly as specified)

License Key (6 digits)

OK Cancel

The OK button becomes enabled when you enter a valid name and key combination.

Corramos el OLLYDBG y lleguemos hasta la ventana de registraci3n y veamos cual es uno de los m3todos de ataque este tipo de protecci3n.



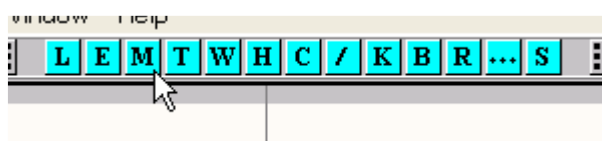
Tipeo un nombre, supongo que aceptara cualquier nombre si no veremos mas adelante, ahora tipearé una LICENSE KEY rara, de 6 letras pongámosle WMYXSZ, apreto la W



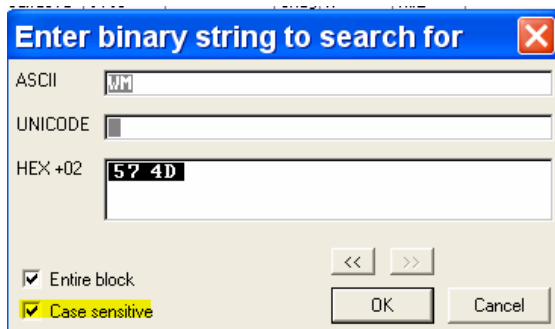
Ahora apreto la M



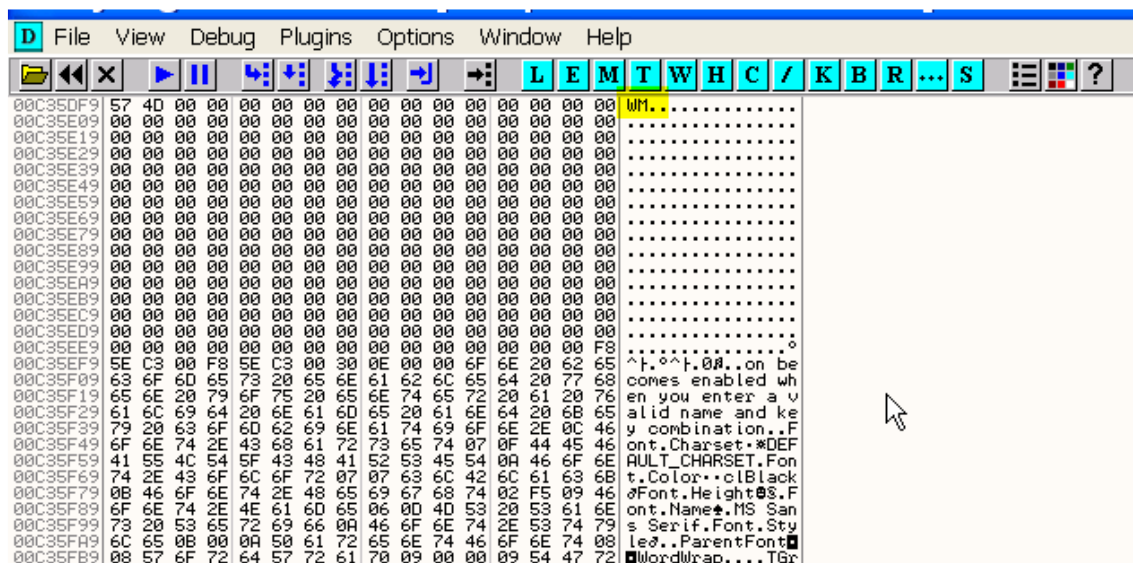
Ya apreté dos letras que pasa si busco en la memoria la string WM



Apreto el botón M para ir a VIEW-MEMORY y busco la string WM

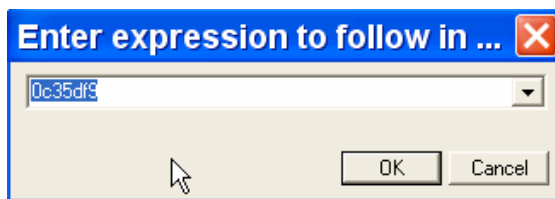


En este caso habilito con la tilde CASE SENSITIVE para que busque exacto WM solo mayúsculas, así hay menos ocurrencias.



Luego de que para dos o tres veces en palabras que comienzan con WM, y que sigo buscando con CTRL+L y si termina una sección salgo y sigo buscando mas abajo en la ventana M, llego a esta ocurrencia que muestra WM en mayúscula y solo, y además abajo esta el texto que muestra de que se habilitara el botón OK cuando apretemos el serial correcto.

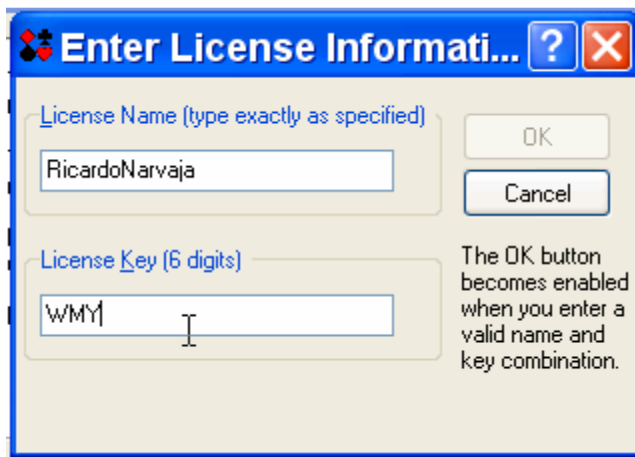
Busquemos esta zona en el dump



Recordemos que a las direcciones que comienzan con letra, como en este caso la C debemos ponerle un cero delante si no el OLLYDBG no las reconocerá.

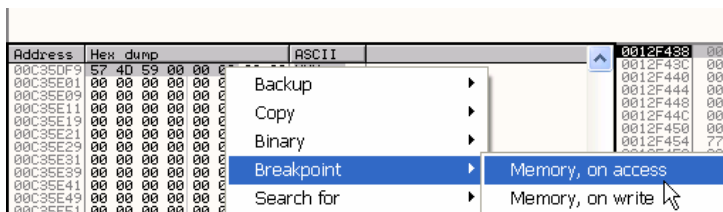
Address	Hex dump	ASCII
00C35DF9	57 4D 00 00 00 00 00 00	WM.....
00C35E01	00 00 00 00 00 00 00 00
00C35E09	00 00 00 00 00 00 00 00
00C35E11	00 00 00 00 00 00 00 00
00C35E19	00 00 00 00 00 00 00 00
00C35E21	00 00 00 00 00 00 00 00
00C35E29	00 00 00 00 00 00 00 00
00C35E31	00 00 00 00 00 00 00 00
00C35E39	00 00 00 00 00 00 00 00

Para estar seguros que esta es la zona donde va guardando el serial, escribamos el siguiente carácter del serial WMYXSZ, o sea tipeo la Y.

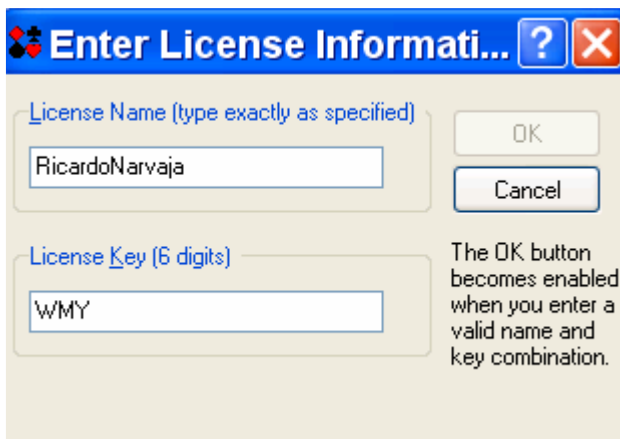


Address	Hex dump	ASCII
00C35DF9	57 40 59 00 00 00 00 00	WMY.....
00C35E01	00 00 00 00 00 00 00 00
00C35E09	00 00 00 00 00 00 00 00
00C35E11	00 00 00 00 00 00 00 00

Al tipearla, veo en el dump que se agrego y que halle la zona donde va guardando el serial, por lo tanto, pongamos allí un BREAKPOINT MEMORY ON ACCESS que abarque 6 cifras ya que sabemos que son 6.



Ahora doy RUN



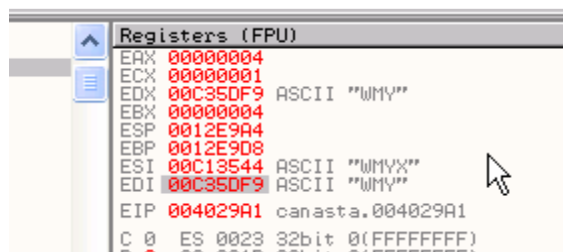
Me aparece la ventana pongo el carácter siguiente que es X.


```

0040297E . 78 11      US SHUNT Canasta.004029B1
004029A0 . FD        STD
004029A1 . F3:A5     REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
004029A3 . 89C1     MOV ECX,EAX
004029A5 . 83E1 03   AND ECX,3
004029A8 . 83C6 03   ADD ESI,3
004029AB . 83C7 03   ADD EDI,3
004029AE . F3:A4     REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
004029B0 . FC        CLD
004029B1 . 5F        POP EDI
004029B2 . 5E        POP ESI
004029B3 . C3        RETN
004029B4 . 53        PUSH EBX
004029B5 . 56        PUSH ESI
004029B6 . 57        PUSH EDI
004029B7 . 55        PUSH EBP

```

Ahora si para cuando lo va a guardar a continuación



Allí copiara de ESI a EDI, al apretar F8

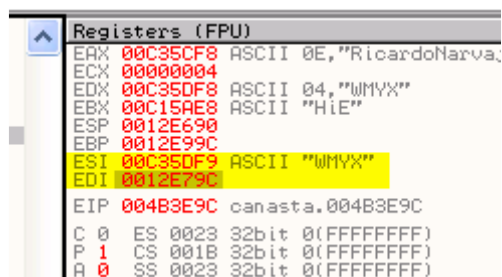
Address	Hex dump	ASCII
00C35DF9	57 4D 59 58 00 00 00 00	WMVX...
00C35E01	00 00 00 00 00 00 00 00
00C35E09	00 00 00 00 00 00 00 00
00C35E11	00 00 00 00 00 00 00 00

```

004B3E97 . 33C9     XOR ECX,ECX
004B3E99 . 8A0E     MOV CL,BYTE PTR DS:[ESI]
004B3E9B . 41       INC ECX
004B3E9C . F3:A4     REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
004B3E9E . 8BF0     MOV ESI,EAX
004B3EA0 . 8DBD FFFEF LEA EDI,DWORD PTR SS:[EBP-101]
004B3EA6 . 33C9     XOR ECX,ECX
004B3EA8 . 8A0E     MOV CL,BYTE PTR DS:[ESI]
004B3EAA . 41       INC ECX

```

Luego para nuevamente cuando copia al stack, los cuatro bytes que entramos hasta ahora.



Lo copiara a 12E79C al apretar F8

Address	Hex dump	ASCII
0012E79C	57 4D 59 58 41 3B 3C 77	WMVXA; <w
0012E7A4	73 B4 D1 77 B4 E7 12 00	s10wHs#.
0012E7AC	0E 00 00 00 2C 00 00 00	#.....
0012E7B4	BE 94 D1 77 98 87 D2 77	#00w0qEw
0012E7BC	46 A8 11 00 00 00 00 00	F#.....

Ya que el BREAKPOINT MEMORY ON ACCESS lo tenemos usando, podemos colocar un HARDWARE BPX ON ACCESS

Address	Hex dump	ASCII	
0012E79C	57 40 59 58 41 3B 3C 77	Backup	
0012E7A4	73 B4 01 77	Copy	
0012E7AC	0E 00 00 00	Binary	
0012E7B4	BE 94 01 77	Breakpoint	
0012E7BC	46 08 11 00	Search for	
0012E7C4	05 00 00 00	Go to	
0012E7CC	EF 08 FF FF	Remove memory breakpoint	
0012E7D4	01 00 00 00	Hardware, on access	
0012E7DC	E3 5A C1 00	Hardware, on write	
0012E7E4	05 E9 01 77	Hardware, on execution	
0012E7EC	46 08 11 00	Byte	
0012E7F4	05 00 00 00	Word	
0012E7FC	01 00 00 00	Dword	
0012E804	24 CA 46 00		
0012E80C	46 08 11 00		
0012E814	05 00 00 00		
0012E81C	5C E9 12 00		
0012E824	E8 5A C1 00		
0012E82C	56 CB 45 00		
0012E834	88 E9 12 00		
0012E83C	C4 E9 12 00		
0012E844	C1 00 00 00		
0012E84C	84 E8 12 00		
0012E854	A4 E9 12 00		
0012E85C	01 00 00 00		

004B3EA6	33C9	XOR ECX,ECX	
004B3EA8	8A0E	MOV CL, BYTE PTR DS:[ESI]	
004B3EAA	41	INC ECX	
004B3EAB	F3:A4	REP MOVS BYTE PTR ES:[EDI], BYTE PTR DS:[ESI]	
004B3EAD	80BD FFFFFFFF	CMP BYTE PTR SS:[EBP-101], 0	
004B3EB4	0F84 F6000000	JE canasta.004B3FB0	
004B3EBA	8D85 FFFFFFFF	LEA EAX, DWORD PTR SS:[EBP-101]	

Si traceo un poco veo que mueve al stack también mi nombre

Registers (FPU)	
EAX	00C35CF8 ASCII 0E,"RicardoNarvaja"
ECX	0000000F
EDX	00C35DF8 ASCII 04,"WMVX"
EBX	00C15AE8 ASCII "HiE"
ESP	0012E690
EBP	0012E99C
ESI	00C35CF8 ASCII 0E,"RicardoNarvaja"
EDI	0012E89B
EIP	004B3EAB canasta.004B3EAB
C 0	ES 0023 32bit 0(FFFFFFFF)
P 1	CS 001B 32bit 0(FFFFFFFF)
A 0	SS 0023 32bit 0(FFFFFFFF)
7	DS 0023 32bit 0(FFFFFFFF)

Address	Hex dump	ASCII
0012E89B	0E 52 69 63 61 72 64 6F	#Ricardo
0012E8A3	4E 61 72 76 61 6A 61 00	Narvaja.
0012E8AB	00 FF FF FF FF 3A EC 12	.\$0\$.10\$
0012E8B3	00 24 EA 12 00 F4 E9 12	.\$0\$.10\$
0012E8BB	0A 0C 0D 01 77 B7 1A 01	.\$0\$.10\$

Antes justo del nombre esta 0E que es el largo del mismo, y lo compara con cero

004B3EAB	F3:A4	REP MOVS BYTE PTR ES:[EDI], BYTE PTR DS
004B3EAD	80BD FFFFFFFF	CMP BYTE PTR SS:[EBP-101], 0
004B3EB4	0F84 F6000000	JE canasta.004B3FB0
004B3EBA	8D85 FFFFFFFF	LEA EAX, DWORD PTR SS:[EBP-101]
004B3EC0	BA 5C404B00	MOV EDX, canasta.004B405C
004B3EC5	33C9	XOR ECX, ECX
004B3EC7	8A08	MOV CL, BYTE PTR DS:[EAX]
004B3EC9	41	INC ECX
004B3ECA	E8 E9F2F4FF	CALL canasta.004031B8
004B3ECF	0F84 DB000000	JE canasta.004B3FB0
004B3ED5	8D85 FFFFFFFF	LEA EAX, DWORD PTR SS:[EBP-101]
004B3EDB	BA 60404B00	MOV EDX, canasta.004B4060
004B3EE0	33C9	XOR ECX, ECX
004B3EE2	8A08	MOV CL, BYTE PTR DS:[EAX]
004B3EE4	41	INC ECX
004B3EE5	E8 CEF2F4FF	CALL canasta.004031B8
004B3EEA	0F84 C0000000	JE canasta.004B3FB0
004B3EF0	8D85 FFFFFFFF	LEA EAX, DWORD PTR SS:[EBP-101]
004B3EF6	BA 60404B00	MOV EDX, canasta.004B4068
004B3EFB	33C9	XOR ECX, ECX
004B3EFD	8A08	MOV CL, BYTE PTR DS:[EAX]
004B3EFF	41	INC ECX
004B3F00	E8 B3F2F4FF	CALL canasta.004031B8
004B3F05	0F84 A5000000	JE canasta.004B3FB0
004B3F0B	8D85 FFFFFFFF	LEA EAX, DWORD PTR SS:[EBP-101]
004B3F11	BA 70404B00	MOV EDX, canasta.004B4070
004B3F16	33C9	XOR ECX, ECX
004B3F18	8A08	MOV CL, BYTE PTR DS:[EAX]
004B3F1A	41	INC ECX
004B3F1B	E8 98F2F4FF	CALL canasta.004031B8

Como no es cero continua

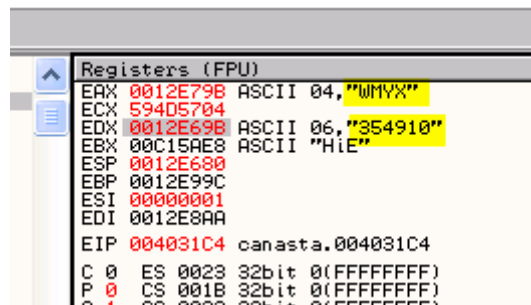
004B3EAB	F3:A4	REP MOVS BYTE PTR ES:[EDI], BYTE PTR DS	
004B3EAD	80BD FFFFFFFF	CMP BYTE PTR SS:[EBP-101], 0	
004B3EB4	0F84 F6000000	JE canasta.004B3FB0	
004B3EBA	8D85 FFFFFFFF	LEA EAX, DWORD PTR SS:[EBP-101]	
004B3EC0	BA 5C404B00	MOV EDX, canasta.004B405C	ASCII 03,"TNO"
004B3EC5	33C9	XOR ECX, ECX	
004B3EC7	8A08	MOV CL, BYTE PTR DS:[EAX]	
004B3EC9	41	INC ECX	
004B3ECA	E8 E9F2F4FF	CALL canasta.004031B8	
004B3ECF	0F84 DB000000	JE canasta.004B3FB0	
004B3ED5	8D85 FFFFFFFF	LEA EAX, DWORD PTR SS:[EBP-101]	ASCII 05,"afdad"
004B3EDB	BA 60404B00	MOV EDX, canasta.004B4060	
004B3EE0	33C9	XOR ECX, ECX	
004B3EE2	8A08	MOV CL, BYTE PTR DS:[EAX]	
004B3EE4	41	INC ECX	
004B3EE5	E8 CEF2F4FF	CALL canasta.004031B8	
004B3EEA	0F84 C0000000	JE canasta.004B3FB0	
004B3EF0	8D85 FFFFFFFF	LEA EAX, DWORD PTR SS:[EBP-101]	ASCII 05,"Gauss"
004B3EF6	BA 60404B00	MOV EDX, canasta.004B4068	
004B3EFB	33C9	XOR ECX, ECX	
004B3EFD	8A08	MOV CL, BYTE PTR DS:[EAX]	
004B3EFF	41	INC ECX	
004B3F00	E8 B3F2F4FF	CALL canasta.004031B8	
004B3F05	0F84 A5000000	JE canasta.004B3FB0	
004B3F0B	8D85 FFFFFFFF	LEA EAX, DWORD PTR SS:[EBP-101]	ASCII 16,"StARDoGG CHaMPiON PC97"
004B3F11	BA 70404B00	MOV EDX, canasta.004B4070	
004B3F16	33C9	XOR ECX, ECX	
004B3F18	8A08	MOV CL, BYTE PTR DS:[EAX]	
004B3F1A	41	INC ECX	
004B3F1B	E8 98F2F4FF	CALL canasta.004031B8	

Luego vienen comparaciones del nombre con personas que están en la lista negra, o sea mejor no ser TNO, afdad etc jeje.

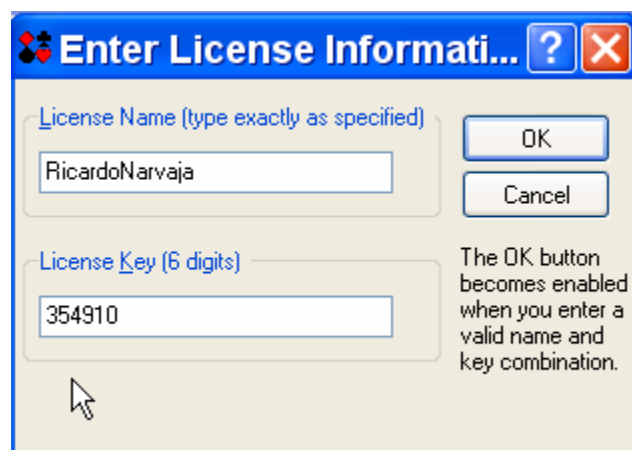
Al dar RUN nuevamente

```
004031C0 | < 74 26 | JE SHORT canasta.004031E8
004031C2 | > 8B08 | MOV ECX,DWORD PTR DS:[EAX]
004031C4 | < 8B1A | MOV EBX,DWORD PTR DS:[EDX]
004031C6 | < 39D9 | CMP ECX,EBX
004031C8 | < 75 45 | JNZ SHORT canasta.0040320F
004031CA | < 4E | DEC ESI
```

Para en una comparación allí, y vemos que compara con el serial



Y podemos probar si es el serial correcto quitando todos los BPM y BPX y HARDWARE BPX



Al colocarlo se habilita el botón y al apretarlo estamos registrados jeje, otra opción que no voy a detenerme a explicar aquí, pero no es muy difícil, es usar WM_KEYUP cosa de que cada vez que apretamos una tecla para OLLYDBG y podamos seguir el valor del byte que tipeamos, aunque lógicamente es más engorroso, este método si nos funciona es mucho más sencillo.

Bueno los dejo descansar un poco, la próxima parte ya veremos un poco más de teoría, acompañada de ejemplos para poder ir avanzando más en conocimientos lo cual derivará en programas más complejos resueltos.

Hasta la parte 18

Ricardo Narvaja

15 de diciembre de 2005

