

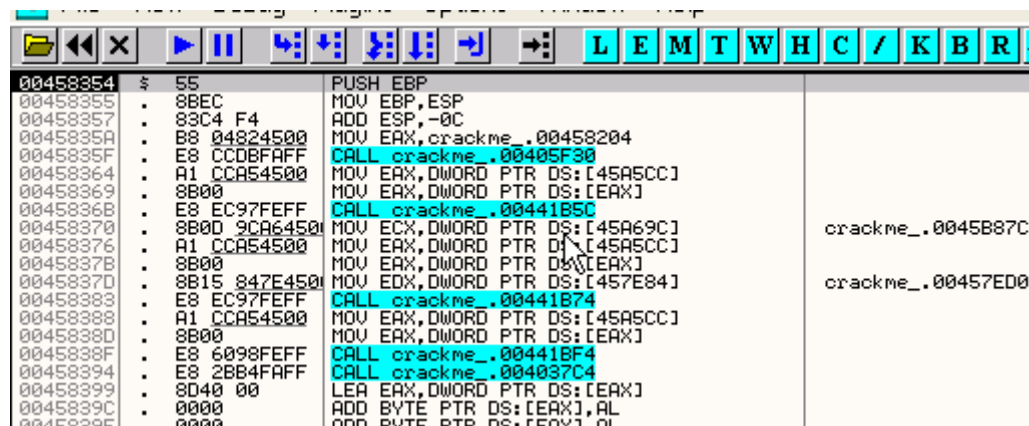
INTRODUCCIÓN AL CRACKING CON OLLYDBG PARTE 18

Bueno, en la parte anterior había comentado que iba aquí a empezar otro tema que sería el tema de la detección del OLLYDBG, por los programas que ejecutamos en él, lo que comúnmente se llama antidebugging, como evitarlo manualmente, y con los plugins disponibles en cada caso y con ejemplos.

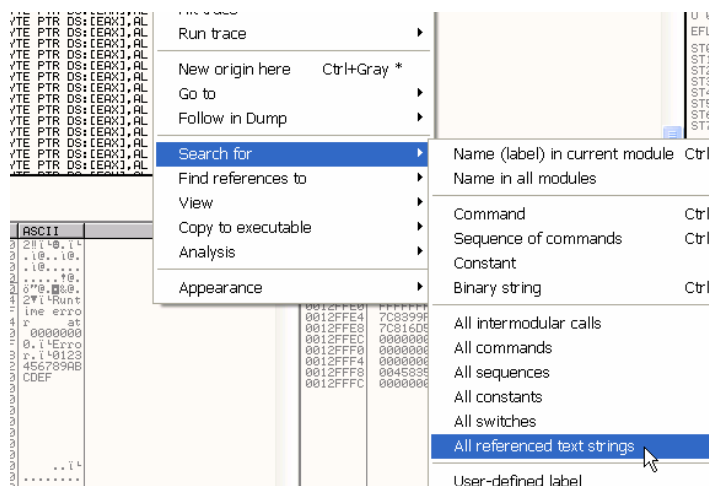
Pero eso con ese tema comenzare en la parte 19, ya que surgió entre varios seguidores del curso que me escribieron al mail, el pedido de que haga un ejemplo mas de crackme sin botones, como el de la parte anterior, y si es posible usando la otra técnica la de WM_KEYUP .

Bueno así que usaremos un crackme que me facilito Stzwei, gracias por el mismo, llamado crackme_4stz.exe y que por supuesto estará adjunto con este tutorial.

Abramos el crackme en OLLYDBG



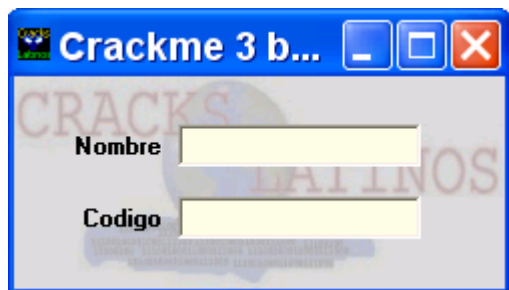
Allí esta parado en el Entry Point, veamos las strings referentes ya que no esta empacado.



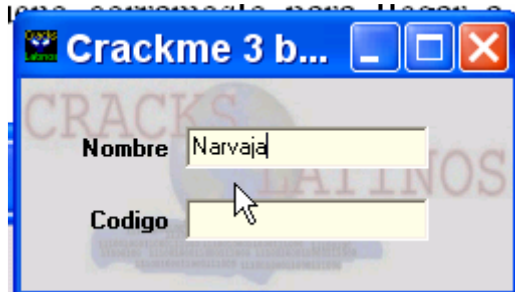
Allí vemos las Strings que usa el programa.

Address	Section	Type	Name	Comment
3045C590	.idata	Import	user32.DestroyCursor	
3045C5FC	.idata	Import	user32.DestroyIcon	
3045C5F8	.idata	Import	user32.DestroyMenu	
3045C5F4	.idata	Import	user32.DestroyWindow	
3045C5F0	.idata	Import	user32.DispatchMessageA	
3045C5EC	.idata	Import	user32.DrawEdge	
3045C5E8	.idata	Import	user32.DrawFrameControl	
3045C5E4	.idata	Import	user32.DrawIcon	
3045C5E0	.idata	Import	user32.DrawIconEx	
3045C5DC	.idata	Import	user32.DrawMenuBar	
3045C5D8	.idata	Import	user32.DrawTextA	
3045C5D4	.idata	Import	user32.EnableMenuItem	
3045C5D0	.idata	Import	user32.EnableScrollBar	
3045C5CC	.idata	Import	user32.EnableWindow	
3045C5C8	.idata	Import	user32.EndPaint	
3045C5BC	.idata	Import	kernel32.EnterCriticalSection	
3045C5B8	.idata	Import	kernel32.EnumCalendarInfoA	
3045C5B4	.idata	Import	user32.EnumThreadWindows	
3045C5B0	.idata	Import	user32.EnumWindows	
3045C5AC	.idata	Import	user32.EqualRect	
3045C5A8	.idata	Import	gdi32.ExcludeClipRect	
3045C5A4	.idata	Import	kernel32.ExitProcess	
3045C5A0	.idata	Import	user32.FillRect	
3045C59C	.idata	Import	kernel32.FindClose	
3045C598	.idata	Import	kernel32.FindFirstFileA	
3045C594	.idata	Import	kernel32.FindResourceA	
3045C590	.idata	Import	user32.FindWindowA	
3045C58C	.idata	Import	kernel32.FormatMessageA	
3045C588	.idata	Import	user32.FrameRect	
3045C584	.idata	Import	kernel32.FreeLibrary	
3045C580	.idata	Import	kernel32.FreeLibrary	
3045C57C	.idata	Import	kernel32.FreeResource	
3045C578	.idata	Import	gdi32.GdiFlush	
3045C574	.idata	Import	user32.GetActiveWindow	
3045C570	.idata	Import	gdi32.GetBitmapBits	
3045C56C	.idata	Import	gdi32.GetBrushOrgEx	
3045C568	.idata	Import	user32.GetCapture	
3045C564	.idata	Import	user32.GetClassInfoA	
3045C560	.idata	Import	user32.GetClientRect	
3045C55C	.idata	Import	user32.GetClipboardData	
3045C558	.idata	Import	gdi32.GetClipboard	
3045C554	.idata	Import	kernel32.GetCommandLineA	
3045C550	.idata	Import	kernel32.GetCPInfo	
3045C54C	.idata	Import	gdi32.GetCurrentPositionEx	
3045C548	.idata	Import	kernel32.GetCurrentProcessId	
3045C544	.idata	Import	kernel32.GetCurrentThreadId	
3045C540	.idata	Import	kernel32.GetCurrentThreadId	
3045C53C	.idata	Import	user32.GetCursor	
3045C538	.idata	Import	user32.GetCursorPos	
3045C534	.idata	Import	user32.GetDC	
3045C530	.idata	Import	user32.GetDCEx	
3045C52C	.idata	Import	gdi32.GetDCOrgEx	
3045C528	.idata	Import	user32.GetDesktopWindow	
3045C524	.idata	Import	gdi32.GetDeviceCaps	
3045C520	.idata	Import	gdi32.GetDIBColorTable	
3045C51C	.idata	Import	gdi32.GetDIBits	
3045C518	.idata	Import	kernel32.GetDiskFreeSpaceA	
3045C514	.idata	Import	gdi32.GetEnhMetaFileBits	
3045C510	.idata	Import	gdi32.GetEnhMetaFileHeader	
3045C50C	.idata	Import	gdi32.GetEnhMetaFilePaletteEntries	
3045C508	.idata	Import	kernel32.GetFileSize	

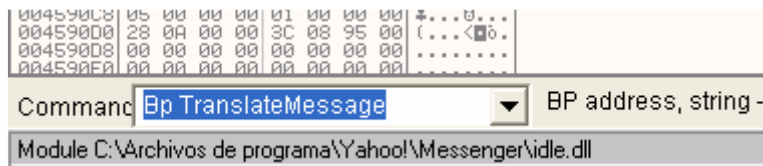
Uff muchísimas para mi gusto, bueno corrámoslo para llegar a la ventana donde se ingresa el serial, apreto F9.



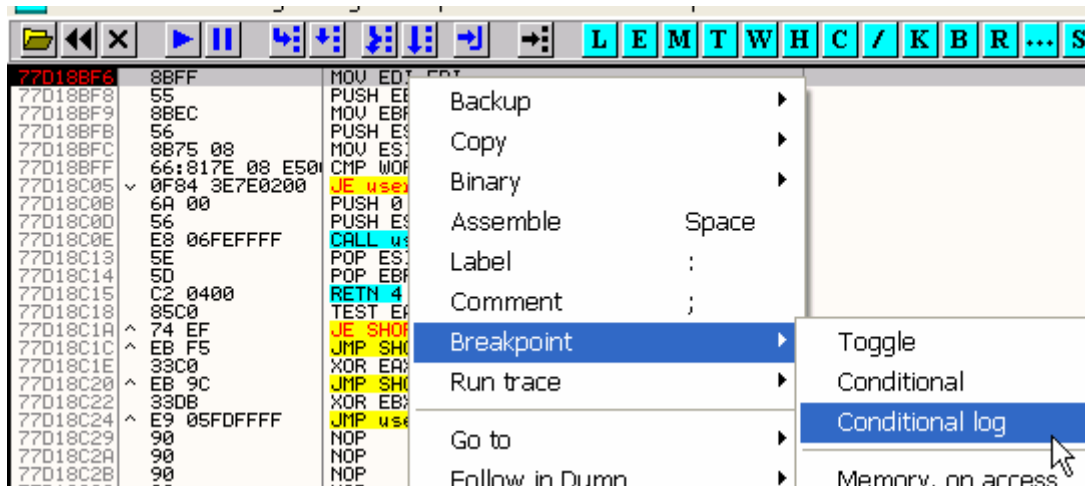
Vemos que no hay botón de registro, así que tipeo un nombre completo



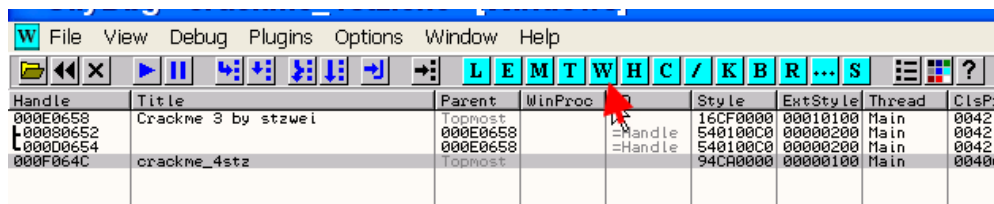
Y pondré un BPX CONDICIONAL LOG en TranslateMessage, primero coloco un BPX común



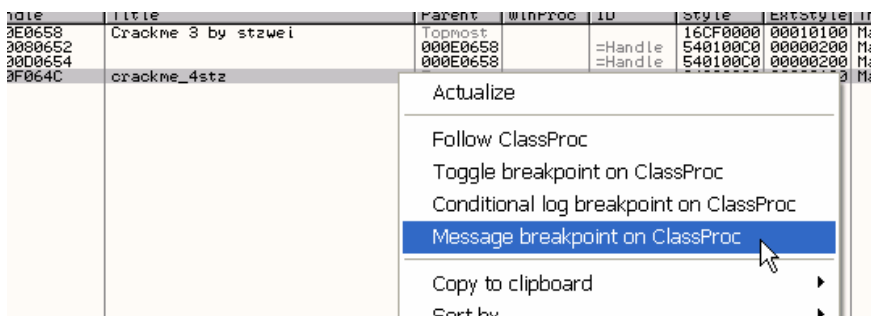
Y cuando voy a acceder al crackme, para solo en el BP allí lo edito.



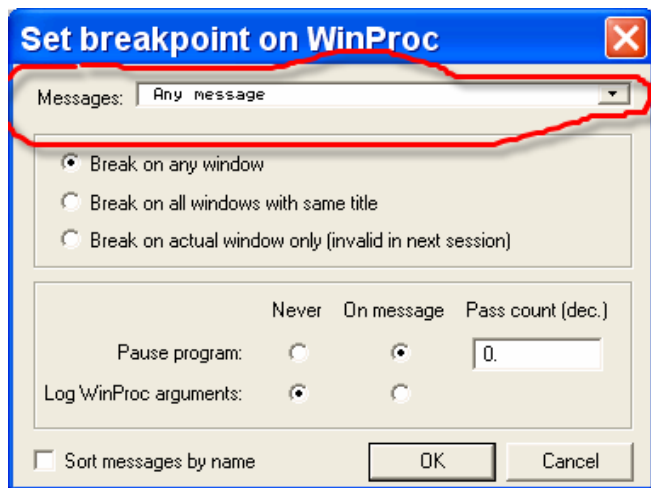
Si alguien no se acuerda el valor correspondiente al WM que vamos a usar, pues vamos a la ventana W, que nos muestra las ventanas del programa.



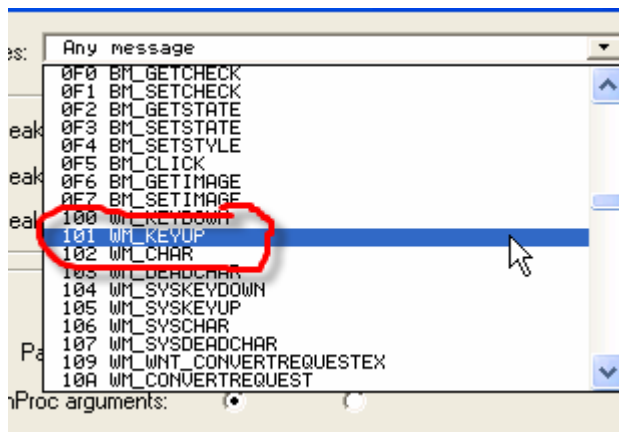
Y nos aparece la lista de ventanas, en cualquiera, total es solo para averiguar el valor numérico, hacemos click derecho



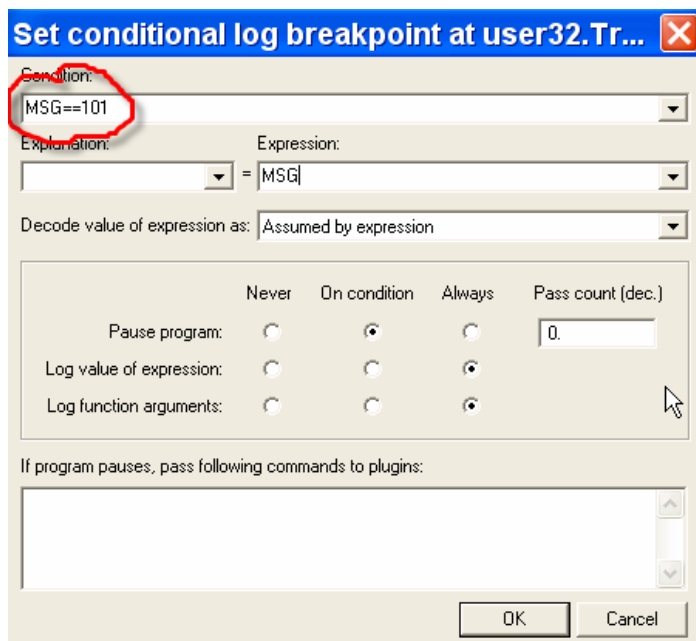
Y nos aparece la ventana de los MESSAGE BREAKPOINTS con el menú desplegable de los WM



Buscamos en la lista WM_KEYUP



Vemos que es 101, así que cancelamos todo esto que fue solo para averiguar ese valor numérico y volvemos al BREAKPOINT CONDICIONAL LOG en TranslateMessage.

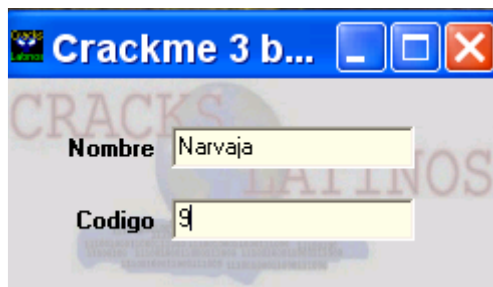


Y allí coloco MSG==101 (recordar el doble signo igual), también podía haber escrito allí si no encuentro el valor, MSG==WM_KEYUP y funcionara, pero a mi me gusta mas usar valores numéricos, cada uno puede hacer como quiera en ese punto.

Bueno allí quedo nuestro BPX transformado en CONDICIONAL LOG (color rosa)

77D18BF6	8BFF	MOV EDI,EDI
77D18BF8	55	PUSH EBP
77D18BF9	8BEC	MOV EBP,ESP
77D18BF8	56	PUSH ESI
77D18BFC	8B75 08	MOV ESI,DWORD PTR SS:[EBP+8]
77D18BFF	66:817E 08 E50	CMP WORD PTR DS:[ESI+8],0E5
77D18C05	0F84 3E7E0200	JE user32.77D40A49
77D18C08	6A 00	PUSH 0
77D18C0D	56	PUSH ESI
77D18C0E	E8 06FEFFFF	CALL user32.TranslateMessageEx
77D18C13	5E	POP ESI
77D18C14	5D	POP EBP
77D18C15	C2 0400	RETN 4
77D18C18	85C0	TEST EAX,EAX
77D18C1A	74 EF	JE SHORT user32.77D18C08
77D18C1C	EB F5	JMP SHORT user32.77D18C13
77D18C1E	33C0	XOR EAX,EAX

Demos RUN y tecleo la primera letra de mi serial falso

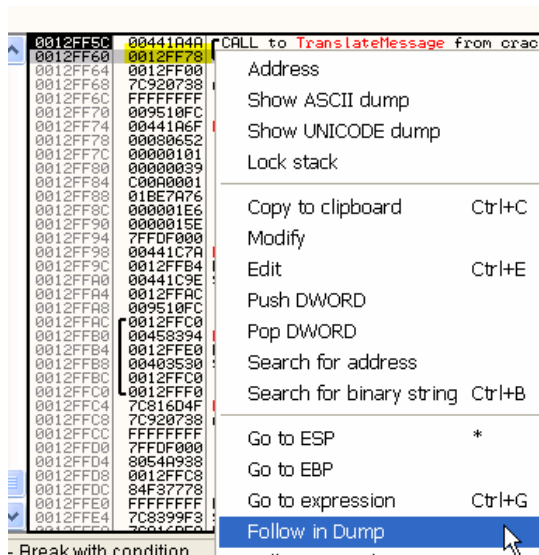


Y allí se detiene el OLLYDBG en el BPX CONDICIONAL.

0012FF5C	00441A4A	CALL to TranslateMessage from crackme.00441A45
0012FF60	0012FF78	msg = WM_KEYUP hw = 80652 (class="Tedit") Key = 39 ('9') KeyData = C00A0001
0012FF64	0012FF00	ntdll.7C920738
0012FF68	7C920738	
0012FF6C	FFFFFFFF	
0012FF70	0012FF78	

Vemos allí los parámetros de la api, en 12ff78 hay una estructura que guarda los valores del la tecla que aprete en este caso 39, que corresponde al numero 9.

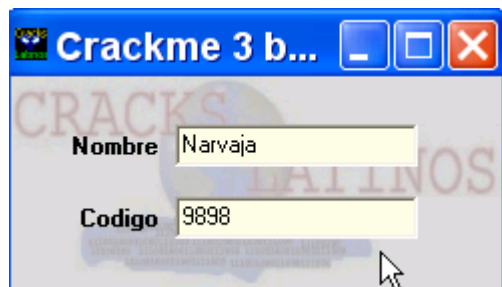
Veamos en el DUMP dicha posición de memoria, haciendo click derecho en dicha dirección y eligiendo FOLLOW IN DUMP.



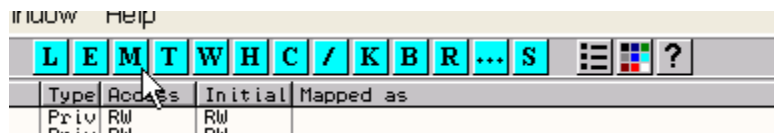
Address	Hex dump	ASCII
0012FF78	52 06 08 00 01 01 00 00	R+0.00..
0012FF80	39 00 00 00 01 00 0A C0	9...0..L
0012FF88	76 7A BE 01 E6 01 00 00	vz#0p0..
0012FF90	5E 01 00 00 00 F0 FD 7F	^0...-20
0012FF98	70 1C 44 00 04 FF 12 00	0. n d 0

Vemos que conseguimos parar y identificar cuando ingresa el byte el problema es que es un programa en DELPHI lo cual veremos mas adelante y si pongo un BPM ON ACCESS en ese byte, dará mil vueltas antes de llegar a la comparación, por lo cual, el método que vimos la vez anterior se aplica mucho mejor aquí y antes de enloquecer pues lo usaremos, de cualquier forma ya saben como parar el programa cuando ingresa por teclado, y como localizar el byte, seguramente en otro crackme que no sea en DELPHI, se podrá seguir mas fácilmente con un BPM ON ACCESS en el mismo, hasta la comparación.

Y doy RUN



Tipeo 9898, iré a la ventana M y buscare en toda la memoria



Hago click derecho- SEARCH

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
00010000	00001000				Priv	RM		
00020000	00001000				Priv	RM		
0012B000	00001000				Priv	RM		
0012C000	00004000			stack of ma	Map	R		
00130000	00003000				Priv	RM		
00140000	00012000				Priv	RM		
00240000	00006000				Priv	RM		
00250000	00003000				Map	R		
00260000	00016000				Map	R		
00290000	00030000				Map	R		
002C0000	00041000				Map	R		
00310000	00006000				Map	R		
00320000	00041000				Map	R		
00370000	00001000				Priv	RM		
00380000	00001000				Priv	RM		

[illegible]

[illegible]

Address	Disassembly	Comment
77D3353D	F3:A5	REP MOVSD WORD PTR ES:[EDI],WORD PTR DS:[ESI]
77D3353F	8BC8	MOV ECX, EAX
77D33541	83E1 03	AND ECX, 3
77D33544	F3:A4	REP MOVSD BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
77D33546	E8 E3FBFFFF	CALL user32.77D3312E
77D33548	5F	POP EDI
77D3354C	5E	POP ESI
77D3354D	8BC3	MOV EAX, EBX
77D3354F	5B	POP EBX

```

EAX 00000007
ECX 00000001
EDX 00140608
EBX 00000007
ESP 0012F090
EBP 0012F0A0
ESI 0014FB88 ASCII "Narvaja"
EDI 009541B4 ASCII "9898"
EIP 77D3353D user32.77D3353D
C 1 ES 0020 32bit 0(FFFFFFFF)
P 0 CS 001B 32bit 0(FFFFFFFF)

```

Address	Hex dump	ASCII
00954184	00 00 00 00 07 00 00 00
0095418C	4E 61 72 76 68 01 00 00	Narvø0..
00954194	17 00 00 00 00 00 00 00
0095419C	06 00 00 00 34 31 38 35	...4185
009541A4	7C 01 00 00 17 00 00 00	!0.....
009541AC	00 00 00 00 07 00 00 00
009541B4	4E 61 72 76 90 01 00 00	Narvø0..
009541BC	17 00 00 00 00 00 00 00
009541C4	06 00 00 00 34 31 38 35	...4185
009541CC	A4 01 00 00 17 00 00 00	!0.....
009541D4	00 00 00 00 07 00 00 00
009541DC	4E 61 72 76 B8 01 00 00	Narvø0..
009541E4	17 00 00 00 00 00 00 00
009541EC	06 00 00 00 34 31 38 35	...4185
009541F4	CC 01 00 00 17 00 00 00	!0.....
009541FC	00 00 00 00 07 00 00 00
00954204	4E 61 72 76 E0 01 00 00	Narvø0..
0095420C	17 00 00 00 00 00 00 00
00954214	06 00 00 00 34 31 38 35	...4185
0095421C	F4 01 00 00 17 00 00 00	!0.....
00954224	01 00 00 00 07 00 00 00
0095422C	4E 61 72 76 61 6A 61 00	Narvaja..
00954234	16 00 00 00 02 00 00 00
0095423C	06 00 00 00 34 31 38 35	...4185
00954244	30 37 00 40 16 00 00 00	07.0....
0095424C	01 00 00 00 06 00 00 00
00954254	39 38 39 38 39 38 00 00	989898..

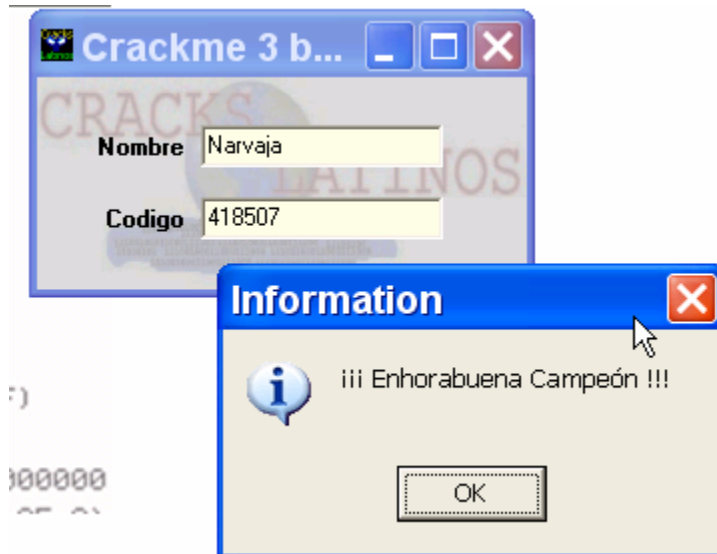
Para justo en la comparación

00403CB3	> 74 26	JE SHORT crackme_.00403CDB
00403CB5	> 8B0E	MOV ECX,DWORD PTR DS:[ESI]
00403CB7	> 8B1F	MOV EBX,DWORD PTR DS:[EDI]
00403CB9	> 39D9	CMP ECX,EBX
00403CBB	> 75 58	JNZ SHORT crackme_.00403D15
00403CBD	> 4A	DEC EDX
00403CBE	> 74 15	JE SHORT crackme_.00403CD5
00403CC0	> 8B4E 04	MOV ECX,DWORD PTR DS:[ESI+4]
00403CC3	> 8B5F 04	MOV EBX,DWORD PTR DS:[EDI+4]
00403CC6	> 39D9	CMP ECX,EBX
00403CC8	> 75 4B	JNZ SHORT crackme_.00403D15
00403CCA	> 83C6 08	ADD ESI,8
00403CCD	> 83C7 08	ADD EDI,8
00403CD0	> 4A	DEC EDX
00403CD1	> 75 E2	JNZ SHORT crackme_.00403CB5
00403CD3	> EB 06	JMP SHORT crackme_.00403CDB
00403CD5	> 83C6 04	ADD ESI,4
00403CD8	> 83C7 04	ADD EDI,4

Allí vemos que ESI y EDI apuntan al serial falso y al verdadero

Registers (FPU)	
EAX	00000000
ECX	38393839
EDX	00000001
EBX	00951804
ESP	0012F3A8
EBP	0012F3E4
ESI	00954254 ASCII "989898"
EDI	00954240 ASCII "418507"
EIP	00403CB7 crackme_.00403CB7
C	1 ES 0023 32bit 0(FFFFFFFF)
P	0 CS 001B 32bit 0(FFFFFFFF)
A	1 SS 0023 32bit 0(FFFFFFFF)
Z	0 DS 0023 32bit 0(FFFFFFFF)
S	0 FS 003B 32bit 7FDF000(FFF)
T	0 GS 0000 NULL
D	0
O	0 LastErr ERROR_SUCCESS (00000000)

Los cuales son comparados de a 4 bytes en ese loop, probemos si el serial hallado es el correcto



Vemos que a pesar de todo es un crackme difícil ya que es un momento justo, que hay que agarrarlo ya que va variando la posición del serial falso y hay que agarrarlo con el BPM justo cuando recién lo escribe en su nueva posición, para poder parar en la comparación.

Otra posibilidad es la siguiente veo que con 4 bytes aparece esto

Address	Hex	dump	ASCII
00954198	00 00 00 00 06 00 00 00	4185i0..
009541A0	34 31 38 35 7C 01 00 00	4185i0..	4185i0..
009541A8	17 00 00 00 00 00 00 00	4185i0..
009541B0	07 00 00 00 4E 61 72 76Narv	4185i0..
009541B8	90 01 00 00 17 00 00 00	00...#...	4185i0..
009541C0	00 00 00 00 06 00 00 00	4185i0..
009541C8	34 31 38 35 A4 01 00 00	4185i0..	4185i0..
009541D0	17 00 00 00 00 00 00 00	4185i0..
009541D8	07 00 00 00 4E 61 72 76Narv	4185i0..
009541E0	B8 01 00 00 17 00 00 00	00...#...	4185i0..
009541E8	00 00 00 00 06 00 00 00	4185i0..
009541F0	34 31 38 35 CC 01 00 00	4185i0..	4185i0..
009541F8	17 00 00 00 00 00 00 00	4185i0..
00954200	07 00 00 00 4E 61 72 76Narv	4185i0..
00954208	E0 01 00 00 17 00 00 00	00...#...	4185i0..
00954210	00 00 00 00 06 00 00 00	4185i0..
00954218	34 31 38 35 F4 01 00 00	4185i0..	4185i0..
00954220	17 00 00 00 00 00 00 00	4185i0..
00954228	07 00 00 00 4E 61 72 76Narv	4185i0..
00954230	08 02 00 00 17 00 00 00	00...#...	4185i0..
00954238	01 00 00 00 06 00 00 00	4185i0..
00954240	34 31 38 35 30 37 00 40	418507..	4185i0..
00954248	60 1E 95 00 F4 96 95 00	'..@..	4185i0..
00954250	34 54 00 00 39 38 39 38	4T..9898	4185i0..
00954258	00 10 69 00 74 B4 45 00	..i..t..E..	4185i0..
00954260	F4 96 95 00 20 54 00 00	@..T..	4185i0..
00954268	00 90 3F 00 00 90 3F 00	..E?..E?..	4185i0..
00954270	00 09 1F 40 00 00 00 95	..@..C..	4185i0..

Con 5 Bytes escribe justo abajo

009541E0	00 00 00 00 00 00 00 00	4185i0..
009541F0	34 31 38 35 CC 01 00 00	4185i0..	4185i0..
009541F8	17 00 00 00 00 00 00 00	4185i0..
00954200	07 00 00 00 4E 61 72 76Narv	4185i0..
00954208	E0 01 00 00 17 00 00 00	00...#...	4185i0..
00954210	00 00 00 00 06 00 00 00	4185i0..
00954218	34 31 38 35 F4 01 00 00	4185i0..	4185i0..
00954220	17 00 00 00 00 00 00 00	4185i0..
00954228	07 00 00 00 4E 61 72 76Narv	4185i0..
00954230	08 02 00 00 17 00 00 00	00...#...	4185i0..
00954238	00 00 00 00 06 00 00 00	4185i0..
00954240	34 31 38 35 1C 02 00 00	4185L0..	4185i0..
00954248	17 00 00 00 00 00 00 00	4185i0..
00954250	07 00 00 00 4E 61 72 76Narv	4185i0..
00954258	30 02 00 00 17 00 00 00	00...#...	4185i0..
00954260	01 00 00 00 06 00 00 00	4185i0..
00954268	34 31 38 35 30 37 00 00	418507..	4185i0..
00954270	60 1E 95 00 F4 96 95 00	'..@..	4185i0..
00954278	0C 54 00 00 39 38 39 38	T..9898	4185i0..
00954280	39 00 20 40 74 B4 45 00	9..@t..E..	4185i0..
00954288	F4 96 95 00 F8 53 00 00	@..S..	4185i0..
00954290	00 90 3F 00 00 5C 22 40	..E?..@	4185i0..
00954298	00 FC F4 12 00 80 22 40	..@..C"@"	4185i0..
009542A0	00 87 22 40 00 E4 F5 12	..@..S\$	4185i0..

Pues voy poniendo BPM ON WRITE en la zona justo abajo, donde va a escribir cuando ingrese el próximo carácter

00954208
00402101
00402107
00402108
DS: [0095428C]=000053F
EDX=009596F4

Copy
Binary
Breakpoint
Search for
Follow DWORD in Dump
Go to
Hex
Text
Short
Long
Float
Disassemble
Special
Appearance

ESI
Memory, on access
Memory, on write
Remove memory break
Hardware, on access
Hardware, on write
Hardware, on execution

Address	Hex	dump
009541F0	34 31 38 35	
009541F8	17 00 00 00	
00954200	07 00 00 00	
00954208	E0 01 00 00	
00954210	00 00 00 00	
00954218	34 31 38 35	
00954220	17 00 00 00	
00954228	07 00 00 00	
00954230	08 02 00 00	
00954238	00 00 00 00	
00954240	34 31 38 35	
00954248	17 00 00 00	
00954250	07 00 00 00	
00954258	30 02 00 00	
00954260	01 00 00 00	
00954268	34 31 38 35	
00954270	16 00 00 00	
00954278	07 00 00 00	
00954280	61 6A 61 00	
00954288	F4 96 95 00	
00954290	00 90 3F 00	
00954298	00 FC F4 12	
009542A0	00 87 22 40	
009542B0	00 78 01 39	
009542B8	00 78 F4 12	

Seguro cuando ingrese el próximo carácter, lo escribirá aquí, así que doy RUN

```

00402811 . F3:A5      REP MOVSB DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
00402813 . 89C1      MOV ECX,EAX
00402815 . 83E1 03   AND ECX,3
00402818 . 83C6 03   ADD ESI,3
0040281B . 83C7 03   ADD EDI,3
0040281E . F3:A4      REP MOVSB BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
00402820 . FC        CLD
00402821 > 5F        POP EDI
00402822 . 5E        POP ESI
00402823 . C3        RETN
00402824 . 83EC 08   SUB ESP,8
00402827 . 8F3C24    FISTP DWORD PTR SS:[FESP]

```

Allí escribo primero el posible serial bueno

```

00954270 10 00 00 00 01 00 00 00 ...0...
00954278 07 00 00 00 4E 61 72 76 ...Narv
00954280 61 6A 61 00 16 00 00 00 aja....
00954288 01 00 00 00 06 00 00 00 0...+...
00954290 34 31 38 35 30 37 00 40 418507.@
00954298 74 B4 45 00 F4 96 95 00 t|E.µ0.
009542A0 E4 53 00 00 00 E4 F5 12 8S...83$
009542A8 00 00 01 00 00 80 3E 00 ..0..C>
009542B0 00 78 01 39 00 F0 56 95 .x09.-U0
009542B8 00 78 F4 12 00 D6 26 40 .x7$.i&0
009542C0 00 8F 46 44 00 00 00 00 .AFD...
009542C8 00 BE 6C 44 00 8C 3F 00 .%ID.i?
009542D0 00 6F 61 44 00 01 00 00 .oαD.0..
009542D8 00 38 1E 95 00 0C 01 00 .8Ao..0.
009542E0 00 38 1E 95 00 58 0F 00 .8Ao.%%.
009542E8 00 94 47 95 00 E4 F5 12 .0G0.83$
009542F0 00 A2 FD 44 00 00 00 00 .o²D....
009542F8 00 00 00 00 00 03 00 00 .....0...

```

Damos RUN

```

77D33538 . 8BC1      MOV EAX,ECX
77D3353A . C1E9 02   SHR ECX,2
77D3353D . F3:A5      REP MOVSB DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
77D3353F . 8BC8      MOV ECX,EAX
77D33541 . 83E1 03   AND ECX,3
77D33544 . F3:A4      REP MOVSB BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
77D33546 . E8 E3FBFFFF CALL user32.77D3312E
77D33548 . 5F        POP EDI
77D3354C . 5E        POP ESI

```

Y luego para cuando copia el serial falso, ahora si que paro apenas se copio, puedo ponerle un BPM on ACCESS al serial falso y al dar RUN para justo en la comparación, nuevamente.

```

00403CB5 > 3B0E      MOV ECX,DWORD PTR DS:[ESI]
00403CB7 . 8B1F      MOV EBX,DWORD PTR DS:[EDI]
00403CB9 . 39D9      CMP ECX,EBX
00403CBB . 75 58     JNZ SHORT crackme_.00403D15
00403CBD . 4A        DEC EDX
00403CBE . 74 15     JE SHORT crackme_.00403CD5
00403CC0 . 8B4E 04   MOV ECX,DWORD PTR DS:[ESI+4]
00403CC3 . 8B5F 04   MOV EBX,DWORD PTR DS:[EDI+4]
00403CC6 . 39D9      CMP ECX,EBX
00403CC8 . 75 48     JNZ SHORT crackme_.00403D15
00403CCA . 83C6 08   ADD ESI,8
00403CCD . 83C7 08   ADD EDI,8
00403CD0 . 4A        DEC EDX
00403CD1 . 75 E2     JNZ SHORT crackme_.00403CB5
00403CD3 . EB 06     JMP SHORT crackme_.00403CDB
00403CD5 > 83C6 04   ADD ESI,4

```

Bueno creo que este fue un caso difícil para cualquier método por lo movetizo de la zona donde guarda el serial.

Bueno les dejare un ejercicio, este será hallar el serial del CRACKME DE CRUEHEAD 2, esto será el fin del primer capítulo de la INTRODUCCIÓN a partir de la parte 19 empezaremos con nuevos temas, y el resolver este sencillo crackme será como la graduación del primer capítulo, recuerden que para abrir el rar de la parte 19 necesitaran del serial verdadero del crackme de cruehead 2 es bien sencillo.

Hasta la parte 19
Ricardo Narvaja
19 de diciembre de 2005