

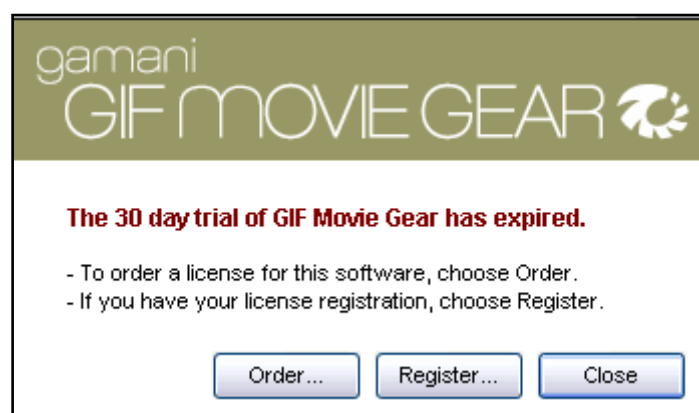


Programa	GIF MOVIE GEAR	
Download	www.gamani.com	
Descripción	Programa para crear gif's.	
Herramientas	OllyDbg 1.10	
Dificultad	Newbie	
Compilador	Microsoft Visual C ++ v 7.0	
Protección	Serial/Name	
Objetivos	Parchar y dejarlo full	
Cracker(?)...Newbie	BioHaZarD	Fecha: 23/07/09
Tutorial nº	1	

Antes que nada PERDÓN!!! Je! Viendo todos los tutos espectaculares de los listeros me da vergüenza escribir esto...pero bue como lei en un post viejo del grupo de google que Ricardo decía que lo que escribamos siempre le sirve a alguien...hací que bueno ahí voy. Espero esto le sirva a algún Newbie como yo...

EMPEZANDO:

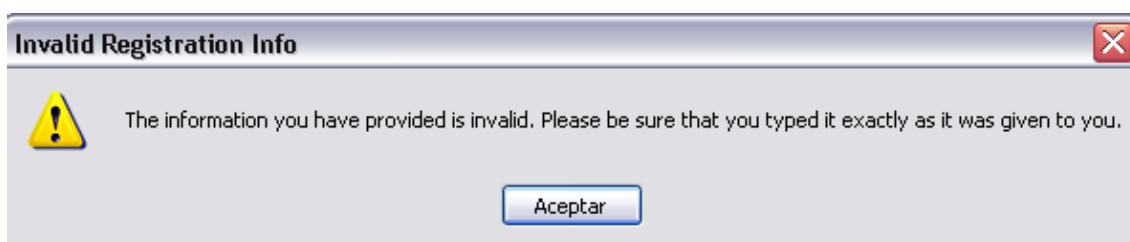
El programa es full por 30 días, tenemos la opción de registrar buscando en "HELP-REGISTER NOW..." en el período trial pero transcurrido este aparece la siguiente nag:



Y luego si pinchamos el botón “Register...” nos aparece una nueva ventana para ingresar un nombre y un serial:



Ingresando un serial cualquiera...



El amigo RDG Packer detector nos da buenas noticia s(por lo menos para mí que soy un Newbie...):



Lo cargamos en Olly y con Ctrl+N (o click con el botón derecho del mouse y...SEARCH FOR/NAME (label) IN CURRENT MODULE) buscamos alguna Api conocida en la lista:

0047F118	.rdata	Import	GUI32.GetTextMetricsH
0047F238	.rdata	Import	KERNEL32.GetTickCount
0047F274	.rdata	Import	KERNEL32.GetVersionExA
0047F4C4	.rdata	Import	USER32.GetWindowLongA
0047F494	.rdata	Import	USER32.GetWindowPlacement
0047F4D0	.rdata	Import	USER32.GetWindowRect
0047F350	.rdata	Import	USER32.GetWindowTextA
0047F344	.rdata	Import	USER32.GetWindowTextLengthA
0047F2E0	.rdata	Import	KERNEL32.GlobalAlloc
0047F2E4	.rdata	Import	KERNEL32.GlobalFree
0047F25C	.rdata	Import	KERNEL32.GlobalLock
0047F2A8	.rdata	Import	KERNEL32.GlobalReAlloc
0047F270	.rdata	Import	KERNEL32.GlobalSize

Allí vemos la api GetWindowTextA que utilizada para ingresar texto, le ponemos un Breakpoint (BP) con F2 y con F9 comienza a correr el programa y en la nag inicial pinchamos en “Register...” y en la que sigue ingresamos un nombre y un serial cualquiera:

Olly para aquí,dentro de USER32:

7E3B216B	\$ 6A 0C	PUSH	USER32.7E3B21D0	
7E3B216D	68 D021387E	PUSH	USER32.7E3985C0	
7E3B2172	E8 4964FEFF	CALL	USER32.7E3985C0	
7E3B2177	8B7D 0C	MOV	EDI,DWORD PTR SS:[EBP+C]	
7E3B217A	33DB	XOR	EBX,EBX	
7E3B217C	3BFB	CMP	EDI,EBX	
7E3B217E	0F84 318F0000	JE	USER32.7E3BB0B5	
7E3B2184	395D 10	CMP	DWORD PTR SS:[EBP+10],EBX	
7E3B2187	0F84 288F0000	JE	USER32.7E3BB0B5	
7E3B218D	895D FC	MOV	DWORD PTR SS:[EBP-4],EBX	
7E3B2190	8B1F	MOV	BYTE PTR DS:[EDI],BL	
7E3B2192	8B4D 08	MOV	ECX,DWORD PTR SS:[EBP+8]	
7E3B2195	E8 4663FEFF	CALL	USER32.7E3984E0	
7E3B219A	8BF0	MOV	ESI,EAX	
7E3B219C	8975 E4	MOV	DWORD PTR SS:[EBP-1C],ESI	
7E3B219F	3BF3	CMP	ESI,EBX	
7E3B21A1	0F84 A3F20000	JE	USER32.7E3C144A	
7E3B21A7	56	PUSH	ESI	
7E3B21A8	E8 6A6CFFFF	CALL	USER32.7E3A8E17	
7E3B21AD	6A 01	PUSH	1	
7E3B21AF	57	PUSH	EDI	
7E3B21B0	FF75 10	PUSH	DWORD PTR SS:[EBP+10]	
7E3B21B3	6A 0D	PUSH	0D	
7E3B21B5	56	PUSH	ESI	
7E3B21B6	85C0	TEST	EAX,EAX	
7E3B21B8	0F85 ED8E0000	JNZ	USER32.7E3BB0AB	
7E3B21BE	E8 177BFFFF	CALL	USER32.7E3A9CDA	
7E3B21C3	834D FC FF	OR	DWORD PTR SS:[EBP-4],FFFFFFFF	
7E3B21C7	E8 3464FEFF	CALL	USER32.7E398600	
7E3B21CC	C2 0C00	RETN	0C	
7E3B21CF	90	NOP		
7E3B21D0	FF	DB	FF	
7E3B21D1	FF	DB	FF	
7E3B21D2	FF	DB	FF	

Si miramos el Stack:

0012F224	00433CCE	CALL to GetWindowTextA from movgear.00433CCC
0012F228	000407EA	hWnd = 000407EA (class='Edit',parent=000407EE)
0012F22C	0012F294	Buffer = 0012F294
0012F230	00000064	Count = 64 (100.)

Hacemos click con el botón derecho en el Buffer y FOLLOW IN DUMP que por ahora esta haci:

Address	Hex dump	ASCII
0012F294	0C F5 12 00 01 15 6B 74 48 01 00 00 58 40 16 00	.s+.0\$ktH0..%0..
0012F2A4	25 FF 6C 74 F6 07 03 00 02 00 00 00 0F C1 00 00	% lt+..0...*!..

Con un EXECUTE TILL RETURN(Ctrl+F9) ejecutamos el codigo hasta el “Retn” que nos va a devolver al programa, si miramos el Stack en la misma posición que recién,vemos que ahora ingreso allí el nombre que pusimos en la ventana de registro:

Address	Hex dump	ASCII
0012F294	62 69 6F 68 61 7A 61 72 64 00 00 00 58 40 16 00	biohazard...%0..
0012F2A4	25 FF 6C 74 F6 07 03 00 02 00 00 00 0F C1 00 00	% lt+..0...*!..

Parados en “Retn” con F8 o F7 volvemos a la siguiente instrucción a ejecutarse del módulo “movgear”:

00433CA6	> C2 1000	RETN 10	Case 1 of switch 00433C81
00433CA9	> 8B8C24 2C010	MOV EDI,DWORD PTR SS:[ESP+12C]	USER32.GetDlgItem
00433CB0	> 8B35 CCF4470	MOV ESI,DWORD PTR DS:[<&USER32.GetDlgItem>]	Count = 64 (100.)
00433CB6	> 6A 64	PUSH 64	Buffer
00433CB8	> 8D5424 64	LEA EDI,DWORD PTR SS:[ESP+64]	ControlID = 44F (1103.)
00433CBC	> 52	PUSH EDI	hWnd
00433CBD	> 68 4F040000	PUSH 44F	GetDlgItem
00433CC2	> 57	PUSH EDI	USER32.GetWindowTextA
00433CC3	> FFD6	CALL ESI	hWnd
00433CC5	> 8B1D 50F3470	MOV EBX,DWORD PTR DS:[<&USER32.GetWindowTextA>]	GetWindowTextA
00433CC8	> 50	PUSH EAX	Count = 64 (100.)
00433CCC	> FFD3	CALL EBX	Buffer
00433CCE	> 6A 64	PUSH 64	ControlID = 450 (1104.)
00433CD0	> 8D8424 C8000	LEA EAX,DWORD PTR SS:[ESP+C8]	hWnd
00433CD7	> 50	PUSH EAX	GetDlgItem
00433CD8	> 68 50040000	PUSH 450	USER32.GetWindowTextA
00433CDD	> 57	PUSH EDI	hWnd
00433CDE	> FFD6	CALL ESI	GetWindowTextA
00433CE0	> 50	PUSH EAX	
00433CE1	> FFD3	CALL EBX	

Parados aquí veamos algo: nosotros volvemos en ese PUSH 64 y mas abajo hay dos CALL, la primera llama a GetDlgItem y la segunda es una llamada a la api GetWindowTextA y si miramos arriba de, PUSH 64 vemos que hay otro CALL a dicha api; es en esa en la que paro el Breakpoint que pusimos nosotros,y en ella se ingreso el nombre truco en el Dump, entonces en esta segunda llamada a WindowGetTextA lo que va a hacer es ingresar nuestro serial truco. Si entramos con F7 a “CALL EBX” volvemos a tener un buffer en el Stack y procediendo de igual manera que con el nombre truco:

En el Stack:

0012F224	00433CE3	CALL to GetWindowTextA from movgear.00433CE1
0012F228	000407F4	hWnd = 000407F4 (class='Edit',parent=000507EE)
0012F22C	0012F2F8	Buffer = 0012F2F8
0012F230	00000064	Count = 64 (100.)

En el Dump:

Address	Hex dump	ASCII
0012F2F8	45 43 43 2E 49 43 00 E8 00 00 00 00 01 00 00 00	ECC.IC.b....0...
0012F308	00 00 00 00 F0 21 16 00 EA 07 05 00 69 17 01 E8	...-!..0..i000
0012F318	00 00 01 00 2C F3 12 00 50 6B 3A 77 69 17 01 E8	..0.,%+.Pk:wi000

Luego de hacer un EXECUTE TILL RETURN (Ctrl+F9):

Address	Hex dump	ASCII
0012F2F8	32 39 32 39 32 39 32 39 32 39 00 00 01 00 00 00	2929292929..0...
0012F308	00 00 00 00 F0 21 16 00 EA 07 05 00 69 17 01 E8-f..0+.i#0p
0012F318	00 00 01 00 2C F3 12 00 50 6B 3A 77 69 17 01 E8	..0.,%#.Pk:wi#0p

Con F8 volvemos al modulo "movgear" aqui:

00433CEB	805424 64	LEA EDX, DWORD PTR SS:[ESP+64]	
00433CF0	52	PUSH EDX	
00433CF0	E8 EBF8FFFF	CALL movgear.004338E0	
00433CF5	83C4 08	ADD ESP, 8	
00433CF8	85C0	TEST EAX, EAX	
00433CFA	0F84 B6000000	JE movgear.00433DB6	
00433D00	8D4424 10	LEA EAX, DWORD PTR SS:[ESP+10]	
00433D04	50	PUSH EAX	
00433D05	8D4C24 10	LEA ECX, DWORD PTR SS:[ESP+10]	
00433D09	51	PUSH ECX	
00433D0A	6A 00	PUSH 0	
00433D0C	68 3F00F00	PUSH 0F00F	
00433D11	6A 00	PUSH 0	
00433D13	68 85F64700	PUSH movgear.0047F685	
00433D18	6A 00	PUSH 0	
00433D1A	68 84E44800	PUSH movgear.0048E484	
00433D1F	68 02000000	PUSH 00000002	
00433D24	FF15 0CF04700	CALL DWORD PTR DS:[<&ADVAPI32.RegCreateKeyEx	RegCreateKeyExA
00433D2A	8D4424 60	LEA EAX, DWORD PTR SS:[ESP+60]	
00433D2E	8D50 01	LEA EDX, DWORD PTR DS:[EAX+1]	
00433D31	> 8A08	MOV CL, BYTE PTR DS:[EAX]	
00433D33	40	INC EAX	
00433D34	84C9	TEST CL, CL	
00433D36	75 F9	JNZ SHORT movgear.00433D31	
00433D38	8B35 00F04700	MOV ESI, DWORD PTR DS:[<&ADVAPI32.RegSetValue	ADVAPI32.RegSetValueExA
00433D3E	2BC2	SUB EAX, EDX	
00433D40	40	INC EAX	
00433D41	50	PUSH EAX	
00433D42	8B4424 10	MOV EAX, DWORD PTR SS:[ESP+10]	
00433D46	8D5424 64	LEA EDX, DWORD PTR SS:[ESP+64]	
00433D4A	52	PUSH EDX	
00433D4B	6A 01	PUSH 1	
00433D4D	6A 00	PUSH 0	
00433D4F	68 C8F34800	PUSH movgear.0048F3C8	
00433D54	50	PUSH EAX	
00433D55	FFD6	CALL ESI	RegSetValueExA

Nosotros estamos en LEA EDX, DWORD PTR SS:[ESP+64] y más abajo vemos llamadas a apis que trabajan con el Registro de Windows, Si miramos la que está más abajo en la imagen vemos que crea un valor con el nombre "RegName3" y un poco más abajo tenemos algo similar pero hay un "RegCode". Al principio de la imagen hay JE y más arriba una CALL., dentro de la cual "hace algo" de lo que después va a depender ese salto JE, si no salta guardara nuestros datos en el registro y supongo que los guardara solo si el serial ingresado es correcto. Podríamos probar si este es el salto clave; el JE verificara el estado del flag Z y si este vale 1 saltará y si vale 0 no lo hará.

00433CFA	805424 64	LEA EDX, DWORD PTR SS:[ESP+64]	
00433D00	8D4424 10	LEA EAX, DWORD PTR SS:[ESP+10]	
00433D04	50	PUSH EAX	
00433D05	8D4C24 10	LEA ECX, DWORD PTR SS:[ESP+10]	
00433D09	51	PUSH ECX	
00433D0A	6A 00	PUSH 0	
00433D0C	68 3F00F00	PUSH 0F00F	
00433D11	6A 00	PUSH 0	
00433D13	68 85F64700	PUSH movgear.0047F685	

En la imagen de arriba estamos ya sobre el salto y si miramos bien vemos una línea roja que va hacia abajo; el color de la línea nos indica que el salto se tomará, igual Olly nos da esta información:

```
Jump is taken
00433DB6=movgear.00433DB6
```

El estado del flag Z es el siguiente:

```
C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
D 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDE000(FFF)
```

Con un doble click sobre el 1:

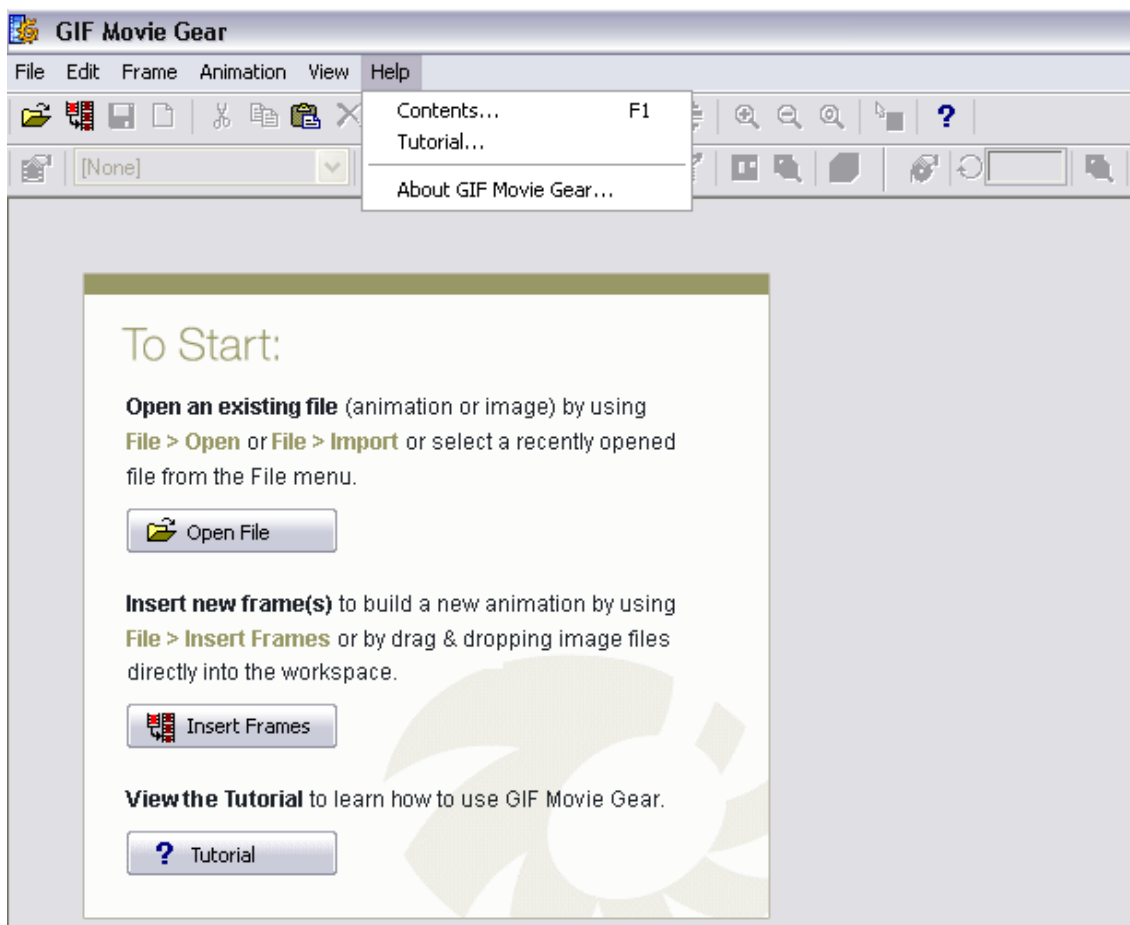
```
C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
D 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDE000(FFF)
```

Ahí vemos que cambia el flag a 0 y ahora nuestro salto se ve haci:

```
0F84 B6000001 LEA EAX,DWORD PTR SS:[ESP+10]
8D4424 10      PUSH EAX
50          LEA ECX,DWORD PTR SS:[ESP+10]
8D4C24 10      PUSH ECX
```

```
Jump is NOT taken
00433DB6=movgear.00433DB6
```

Este cambio no es permanente, se realiza en memoria, es para ver si este salto es el correcto. Pongamos un Breakpoint (F2) ya que si es nuestro salto deberemos volver aquí. Demos Run (F9) y veamos que pasa:



EL programa corre y en HELP ya no aparece la opción REGISTER NOW... Sin duda este es nuestro salto! Carguemos de nuevo el programa (Alt + F2). Bien ahora pensemos como parcharlo. Podriamos cambiar ese JE por un JNE para que cuando el flag Z vale 1 NO salte. Inicialmente pensé que esto funcionaría y de hecho funciona, pero en este caso tiene 2 contras: hay que poner el número de registro (jeje una re vagancia) y al final sale una nag diciendo “THIS TRIAL SOFTWARE EXPIRES IN -1 DAYS”.



A ver si recordamos encima del salto había un CALL. Una llamada arriba de un salto clave me dice que en esa CALL se debe generar el serial y en base a ello el JE se produce o no. Situemonos sobre “CALL movgear.004338E0” y entremos con F7 a ver qué pasa. Una vez dentro vemos algo haci:

Address	Disassembly	Comment
004338E0	53	PUSH EBX
004338E1	55	PUSH EBP
004338E2	8B6C24 10	MOV EBP,DWORD PTR SS:[ESP+10]
004338E6	807D 00 6D	CMP BYTE PTR SS:[EBP],6D
004338E8	56	PUSH ESI
004338E8	57	PUSH EDI
004338EC	0F85 A0000000	JNZ movgear.0043399F
004338F2	807D 01 67	CMP BYTE PTR SS:[EBP+1],67
004338F6	0F85 A3000000	JNZ movgear.0043399F
004338FC	807D 02 33	CMP BYTE PTR SS:[EBP+2],33
00433900	0F85 99000000	JNZ movgear.0043399F
00433906	807D 03 37	CMP BYTE PTR SS:[EBP+3],37
0043390A	0F85 8F000000	JNZ movgear.0043399F
00433910	33DB	XOR EBX,EBX
00433912	8B8B F8F34800	MOV EDI,DWORD PTR DS:[EBX+48F3F8]
00433918	8BC7	MOV EAX,EDI
0043391A	8D50 01	LEA EDX,DWORD PTR DS:[EAX+1]
0043391D	8D49 00	LEA ECX,DWORD PTR DS:[ECX]
00433920	8A08	MOV CL,BYTE PTR DS:[EAX]
00433922	40	INC EAX
00433923	84C9	TEST CL,CL
00433925	75 F9	JNZ SHORT movgear.00433920
00433927	2BC2	SUB EAX,EDX
00433929	8BC8	MOV ECX,EAX
0043392B	8BF5	MOV ESI,EBP
0043392D	33C0	XOR EAX,EAX
0043392F	F3A6	REPE CMPS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
00433931	74 65	JE SHORT movgear.00433998
00433933	83C3 04	ADD EBX,4
00433936	81FB 80000000	CMP EBX,80
0043393C	72 D4	JB SHORT movgear.00433912
0043393E	807D 04 73	CMP BYTE PTR SS:[EBP+4],73
00433942	75 01	JNZ SHORT movgear.00433945
00433944	45	INC EBP
00433945	8D4D 07	LEA ECX,DWORD PTR SS:[EBP+7]
00433948	51	PUSH ECX
00433949	E8 66BD0300	CALL movgear.0046F6B4
0043394E	8B5C24 18	MOV EBX,DWORD PTR SS:[ESP+18]
00433952	8A13	MOV DL,BYTE PTR DS:[EBX]
00433954	83C4 04	ADD ESP,4
00433957	33C9	XOR ECX,ECX

PUSH EBX	
PUSH EBP	
MOV EBP,DWORD PTR SS:[ESP+10]	Pone el serial truco en EBP
CMP BYTE PTR SS:[EBP],6D	Compara el primer byte de mi serial truco con 6D (una "n" en ASCII)
PUSH ESI	
PUSH EDI	
JNZ movgear.0043399F	Salta y va a la zona de chico malo pq "2" es destinto de "n"
CMP BYTE PTR SS:[EBP+1],67	Compara el segundo byte con 67, que es "g" en ASCII
JNZ movgear.0043399F	
CMP BYTE PTR SS:[EBP+2],33	Compara el 3 byte con 33 (3 en ASCII)
JNZ movgear.0043399F	
CMP BYTE PTR SS:[EBP+3],37	Compara el 4 byte con 37 (que es 7 en ASCII)
JNZ movgear.0043399F	
XOR EBX,EBX	

Puse algunos comentarios para que vean lo que hace. Todos esos JNZ me llevan a la zona de chico malo (zona donde se genera el MessageBoxA que me dice que es un serial inválido). Si quisiéramos buscar un serial ya tendríamos como dato que los primeros caracteres tendrían que ser **"mg37"**; pero como todavía soy un poco Newbie y ando flojo en buscar seriales lo voy a parchar. Podríamos invertir todos los saltos con JZ pero preferiría "nopear", no sea cosa que a alguien se le ocurra poner m337 como primeros números de su serial truco... jeje... obviamente saltaría a chico malo. Por ahora anótense estos saltos MALOS en algún lado o pongan BP en ellos para no olvidarse.

Continuemos y veamos si hay más saltos que nos envíen a la zona de serial inválido:

33DB	XOR EBX,EBX
8BBB F8F3480	MOV EDI,DWORD PTR DS:[EBX+48F3F8]
8BC7	MOV EAX,EDI
8D50 01	LEA EDX,DWORD PTR DS:[EAX+1]
8D49 00	LEA ECX,DWORD PTR DS:[ECX]
8A08	MOV CL,BYTE PTR DS:[EAX]
40	INC EAX
84C9	TEST CL,CL
75 F9	JNZ SHORT movgear.00433920
2BC2	SUB EAX,EDX
8BC8	MOV ECX,EAX
8BF5	MOV ESI,EBP
33C0	XOR EAX,EAX
F3A6	REPE CMPS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
74 65	JE SHORT movgear.00433998
83C3 04	ADD EBX,4
81FB 80000000	CMP EBX,80
72 D4	JB SHORT movgear.00433912
807D 04 73	CMP BYTE PTR SS:[EBP+4],73
75 01	JNZ SHORT movgear.00433945
45	INC EBP
8D4D 07	LEA ECX,DWORD PTR SS:[EBP+7]
51	PUSH ECX
E8 66BD0300	CALL movgear.0046F6B4
8B5C24 18	MOV EBX,DWORD PTR SS:[ESP+18]
8A13	MOV DL,BYTE PTR DS:[EBX]
83C4 04	ADD ESP,4
33C9	XOR ECX,ECX
84D2	TEST DL,DL
8BFB	MOV EDI,EBX
BE DF0B0000	MOV ESI,0BDF
74 26	JE SHORT movgear.0043398A

Aquí hay un bucle....trabaja con el serial en ellos, veo unos saltos pero ninguno me lleva a chico malo. Pongo un BP en el CMP a la salida del bucle y doy RUN.


```

807D 04 73 CMP BYTE PTR SS:[EBP+4], 73
JNZ SHORT movgear.00433945
45 INC EBP
804D 07 LEA ECX, DWORD PTR SS:[EBP+7]
51 PUSH ECX
E8 66BD0300 CALL movgear.0046F6B4
8B5C24 18 MOV EBX, DWORD PTR SS:[ESP+18]
8A13 MOV DL, BYTE PTR DS:[EBX]
83C4 04 ADD ESP, 4
33C9 XOR ECX, ECX
84D2 TEST DL, DL
8BF8 MOV EDI, EBX
BE DF0B0000 MOV ESI, 0BDF
74 26 JE SHORT movgear.0043398A
0FBED2 MOVSX EDX, DL
41 INC ECX
0FAFD1 IMUL EDX, ECX
03F2 ADD ESI, EDX
81FE BE170000 CMP ESI, 17BE
7E 06 JLE SHORT movgear.0043397B
81EE BE170000 SUB ESI, 17BE
83F9 0A CMP ECX, 0A
7E 02 JLE SHORT movgear.00433982
33C9 XOR ECX, ECX
8A57 01 MOV DL, BYTE PTR DS:[EDI+1]
47 INC EDI
84D2 TEST DL, DL
75 DA JNZ SHORT movgear.00433964
3BF0 CMP ESI, EAX
75 15 JNZ SHORT movgear.004339A3
5F POP EDI
5E POP ESI
5D POP EBP
B8 01000000 MOV EAX, 1
5B POP EBX
C3 RETN

```

Ahora vemos un salto y una llamada, las pasamos con F8 y no pasa nada. Pongamos un BP en ese CMP ESI, EAX a la salida de este otro bucle:

```

75 DA JNZ SHORT movgear.00433964
3BF0 CMP ESI, EAX
75 15 JNZ SHORT movgear.004339A3
5F POP EDI
5E POP ESI
5D POP EBP
B8 01000000 MOV EAX, 1
5B POP EBX
C3 RETN
5F POP EDI
5E POP ESI
5D POP EBP
33C0 XOR EAX, EAX
5B POP EBX
C3 RETN
8B5C24 14 MOV EBX, DWORD PTR SS:[ESP+14]
55 PUSH EBP
53 PUSH EBX
E8 16FCFFFF CALL movgear.004335C0

```

Ese JNZ es el último y si lo pasamos unas instrucciones más abajo viene un RETN, que nos devolverá un poquito más arriba del JE que parchamos al principio y no funcionó. Una vez en el cambiemos el estado del flag Z y veamos si pasando este salto ya estaría todo bien. Recuerdan como cambiar el flag, no? Esta más arriba.

Vemos que ahora el salto no se tomará:

```

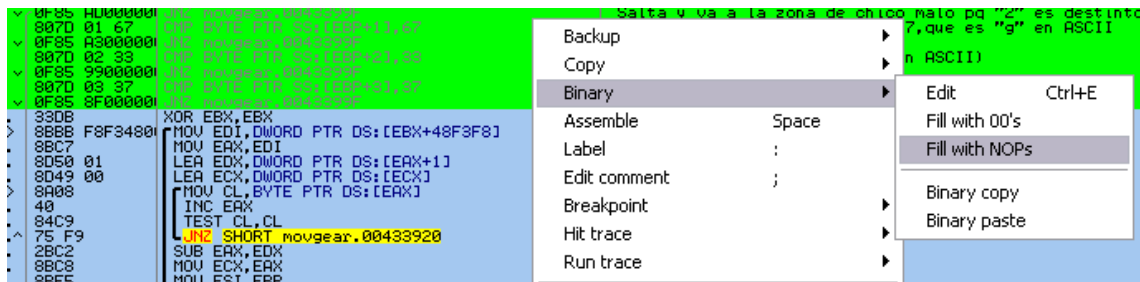
Jump is NOT taken
004339A3=movgear.004339A3

```

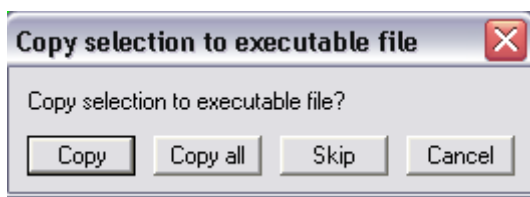
Anoten este salto por las dudas y apretemos F9 a ver si ya esta!!!

El programa corre y en el HELP no aparece más el “REGISTER NOW...”.

Jeje funcionó. Ahora modifiquemos y guardemos los cambios al ejecutable. Primero busquemos todos esos JNZ que estaban juntos y los seleccionamos y hacemos click con el botón derecho y en el menú contextual que se abre elegimos: BINARY/FILL WITH NOPS. :

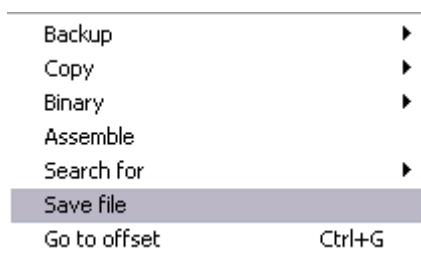


Ahora vamos a el último salto que nos llevaba a chico malo y también lo nopeamos de la misma manera. Una vez hecho esto clickeamos con el botón derecho y en el menú contextual buscamos lo siguiente:



Acá le damos a “Copy all”

Se nos abre una nueva ventana y volvemos a hacer click con el botón derecho y ahora elegimos SAVE FILE.



Guardamos el archivo con otro nombre en el mismo directorio o en otro lado. Con este no hay problema en guardarlo en otro lado pero otros deben estar si o si en el directorio de instalación del programa. Bueno buscando y ejecutando ven que ya esta registrado y la molesta nag del final ya no aparece. DERROTADO.

Bueno este tute está orientado a muy Newbies y espero que les sirva.

De a poquito y mientras más vaya aprendiendo voy a tratar de devolver a la lista todo lo que me va dando haciendo algunos tutes. Saludos a Ricardo, no te conozco pero te estoy agradecido por el curso de “Cracking con Olly DBG desde 0”.

∴ BhZd ∴

