


Fuerza bruta mediante inyección de código

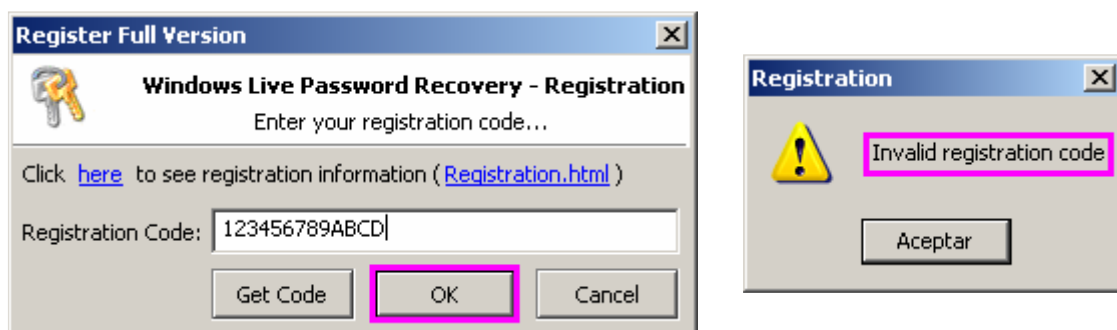
Programa:	 Windows Live Password Recovery Decrypt your Windows Live passwords instantly!
Protección:	ninguna
Descripción:	Programa para recuperar contraseñas del MSN
Dificultad:	Algo mayor de pequeña
Herramientas:	OllyDBG + IIDKing v2.01
Objetivos:	Obtener un serial
Cracker:	Orniaco

Introducción

El programa en cuestión está escrito de Microsoft Visual C++ 7.0 que no presenta ninguna protección salvo la introducción del habitual serial. El programa tiene una rutina que hace ciertas comprobaciones con el serial suministrado para establecer su validez.

Encontrando a chico bueno-chico malo

Arrancamos el programa y procedemos al intento de registro del programa, a través del botón "Register Now". Sólo podemos escribir 13 caracteres y al presionar el botón OK obtenemos:



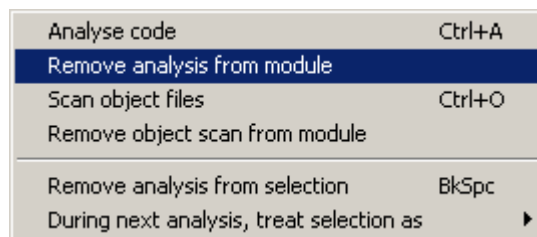
Abrimos el programa con Olly y buscamos la cadena "Invalid registration code", no la encontramos pero si la cadena "Thanks for registering!"

Address	Disassembly	Text string
004051D0	ASCII "P!&8"	
004051D1	ASCII "bu",0	
0040520A	DD Windows_.0041D698	ASCII "Thanks for registering!"
0040520F	DD Windows_.0041D698	ASCII "Thanks for registering!"
00405214	ASCII "f!&8",0	
00405228	DD Windows_.0041D688	ASCII "Registration"

llegamos a la dirección 0040520^a, pero el analizador de Olly se ha hecho un lío y nos muestra lo siguiente:

00405207	6A	DB 6A	CHAR 'j'
00405208	40	DB 40	CHAR '0'
00405209	68	DB 68	CHAR 'h'
0040520A	98D64100	DD Windows_.0041D698	ASCII "Thanks for registering!"
0040520E	68	DB 68	CHAR 'h'
0040520F	98D64100	DD Windows_.0041D698	ASCII "Thanks for registering!"
00405213	8B	DB 8B	
00405214	CE E8 DC EA	ASCII "¡¡¡¡",0	
00405219	00	DB 00	
0040521A	6A	DB 6A	CHAR 'j'

apretando con el botón derecho en código y elegimos “Analysis” y luego “Remove análisis from module”



Se obtiene una visión esclarecedora:

004051E4	8D4C24 04	LEA ECX,DWORD PTR SS:[ESP+4]	
004051E8	51	PUSH ECX	
004051E9	B9 08614200	MOV ECX,Windows_.00426108	
004051EE	E8 ADD5FFFF	CALL Windows_.004027A0	
004051F3	83F8 01	CMP EAX,1	
004051F6	75 2D	JNZ SHORT Windows_.00405225	
004051F8	8D5424 04	LEA EDX,DWORD PTR SS:[ESP+4]	
004051FC	52	PUSH EDX	
004051FD	B9 08614200	MOV ECX,Windows_.00426108	
00405202	E8 C9D4FFFF	CALL Windows_.004026D0	
00405207	6A 40	PUSH 40	
00405209	68 98D64100	PUSH Windows_.0041D698	ASCII "Thanks for registering!"
0040520E	68 98D64100	PUSH Windows_.0041D698	ASCII "Thanks for registering!"
00405213	8BCE	MOV ECX,ESI	
00405215	E8 DCEA0000	CALL Windows_.00413CF6	
0040521A	6A 00	PUSH 0	
0040521C	8BCE	MOV ECX,ESI	
0040521E	E8 02D40000	CALL Windows_.00412625	
00405223	EB 13	JMP SHORT Windows_.00405238	
00405225	6A 30	PUSH 30	
00405227	68 88D64100	PUSH Windows_.0041D688	ASCII "Registration"
0040522C	68 6CD64100	PUSH Windows_.0041D66C	ASCII "Invalid registration code"
00405231	8BCE	MOV ECX,ESI	
00405233	E8 BEEA0000	CALL Windows_.00413CF6	
00405238	8B4424 04	MOV EAX,DWORD PTR SS:[ESP+4]	
0040523C	83C0 F0	ADD EAX,-10	
0040523F	C74424 10 FFFF	MOV DWORD PTR SS:[ESP+10],-1	
00405247	8D48 0C	LEA ECX,DWORD PTR DS:[EAX+C]	
0040524A	83CA FF	OR EDX,FFFFFFFF	
0040524D	F0:0FC111	LOCK XADD DWORD PTR DS:[ECX],EDX	LOCK prefix
00405251	4A	DEC EDX	
00405252	85D2	TEST EDX,EDX	
00405254	5E	POP ESI	
00405255	7F 08	JG SHORT Windows_.0040525F	
00405257	8B08	MOV ECX,DWORD PTR DS:[EAX]	
00405259	8B11	MOV EDX,DWORD PTR DS:[ECX]	
0040525B	50	PUSH EAX	
0040525C	FF52 04	CALL DWORD PTR DS:[EDX+4]	
0040525F	8B4C24 04	MOV ECX,DWORD PTR SS:[ESP+4]	
00405263	64:890D 000000	MOV DWORD PTR FS:[0],ECX	
0040526A	83C4 10	ADD ESP,10	
0040526D	C3	RET	

La rutina 004027A0 chequea el serial introducido, si es correcto devuelve en EAX el valor 1. Le ponemos un BP en la dirección 004051EE.

Al reiniciar el programa e introducir el serial, se para y si analizamos un poco el código y la pila tenemos que la instrucción 004051E4 LEA ECX,DWORD PTR SS:[ESP+4] carga en ECX el valor de la pila que apunta al serial.

La instrucción 004051E8 PUSH ECX empuja ese valor a la pila quedando de esta manera:

0012E688	0012E690	
0012E68C	0041D424	Windows_.0041D424
0012E690	00A463A8	ASCII "123456789ABCD"
0012E694	0012E7D4	Pointer to next SEH record
0012E698	0041B2E8	SE handler
0012E69C	00000000	
0012E6A0	004120D6	RETURN to Windows_.004120D6

Una vez llamada la rutina comprobadora del serial, la pila queda así:

0012E68C	0041D424	Windows_.0041D424
0012E690	00A463A8	ASCII "123456789ABCD"
0012E694	0012E7D4	Pointer to next SEH record
0012E698	0041B2E8	SE handler
0012E69C	00000000	
0012E6A0	004120D6	RETURN to Windows_.004120D6

Método Prueba-Error

La idea es construir seriales al azar en la dirección 00A463A8. Llamar a la rutina comprobadora hasta que sea válido.

004051E4	8D4C24 04	LEA ECX,DWORD PTR SS:[ESP+4]	
004051E8	51	PUSH ECX	
004051E9	B9 08614200	MOV ECX,Windows_.00426108	
004051EE	E8 AD05FFFF	CALL Windows_.004027A0	
004051F3	83F8 01	CMP EAX,1	
004051F6	75 2D	JNZ SHORT Windows_.00405225	
004051F8	8D5424 04	LEA EDX,DWORD PTR SS:[ESP+4]	
004051FC	52	PUSH EDX	
004051FD	B9 08614200	MOV ECX,Windows_.00426108	
00405202	E8 C9D4FFFF	CALL Windows_.004026D0	
00405207	6A 40	PUSH 40	
00405209	68 98D64100	PUSH Windows_.0041D698	ASCII "Thanks for registering!"
0040520E	FF7424 0C	PUSH DWORD PTR SS:[ESP+C]	
00405212	90	NOP	
00405213	8BCE	MOV ECX,ESI	
00405215	E8 DCEA0000	CALL Windows_.00413CF6	
0040521A	6A 00	PUSH 0	
0040521C	8BCE	MOV ECX,ESI	
0040521E	E8 02D40000	CALL Windows_.00412625	
00405223	EB 13	JMP SHORT Windows_.00405238	
00405225	E9 D6650100	JMP Windows_.0041B800	
0040522A	90	NOP	
0040522B	90	NOP	
0040522C	68 6CD64100	PUSH Windows_.0041D66C	ASCII "Invalid registration code"
00405231	8BCE	MOV ECX,ESI	

Si el serial es inválido saltamos a una zona del código que está vacía donde escribiremos nuestro código (0041B800):

0041B800	8B4C24 04	MOV ECX,DWORD PTR SS:[ESP+4]	puntero a serial
0041B804	C781 00010000	MOV DWORD PTR DS:[ECX+100],33323130	construimos
0041B80E	C781 04010000	MOV DWORD PTR DS:[ECX+104],37363534	nuestro
0041B818	C781 08010000	MOV DWORD PTR DS:[ECX+108],42413938	alfabeto
0041B822	C781 0C010000	MOV DWORD PTR DS:[ECX+10C],46454443	"0123456789ABCDEF"
0041B82C	0F31	RDTSC	inicializamos
0041B82E	25 FFFFFFFF	AND EAX,7FFFFFFF	semilla de azar
0041B833	8981 10010000	MOV DWORD PTR DS:[ECX+110],EAX	la guardamos
0041B839	C781 14010000	MOV DWORD PTR DS:[ECX+114],7FFFFFFF	numero maximo
0041B843	C781 18010000	MOV DWORD PTR DS:[ECX+118],41A7	multiplicador
0041B84D	9B	WAIT	
0041B84E	DBE3	FINIT	

Nuestro código va a tener cinco variables, a saber:

[1] El serial, "123456-789ABC", que estará en DS:[ECX], en mi caso la dirección 00A463A8.

[2] El alfabeto, "0123456789ABCDEF", que estará en DS:[ECX+100] en mi caso la dirección 00A464A8.

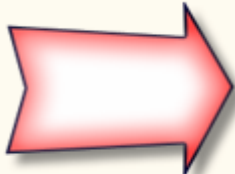
[3] La semilla de los números aleatorios que estará en DS:[ECX+110] en mi caso la dirección 00A464B8.

[4] El número máximo de la rutina de números aleatorios, 7FFFFFFh, que estará en DS:[ECX+114] en mi caso la dirección 00A464BC.

[5] El número multiplicador de la rutina de números aleatorios, 16807, que estará en DS:[ECX+118] en mi caso la dirección 00A464C0.

Address	Hex dump	ASCII
00A46398	80 63 42 00 00 00 00 00 00 00 00 01 00 00 00	0cB.....0...
00A463A8	31 32 33 34 35 36 2D 37 38 39 41 42 43 00 00 00	123456-789ABC...
00A463B8	59 01 05 00 00 10 00 00 78 01 A4 00 78 01 A4 00	80...x00.x00.
00A464A8	30 31 32 33 34 35 36 37 38 39 41 42 43 44 45 46	0123456789ABCDEF
00A464B8	8C 10 AA 10 FF FF FF 7F A7 41 00 00 00 00 00	10A.....
00A464C8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

La rutina generadora de números aleatorios fue descrita en un tutorial anterior **"Números aleatorios en ensamblador usando el coprocesador"**

<pre> 0041B850 33DB XOR EBX,EBX 0041B852 DB81 14010000 FILD DWORD PTR DS:[ECX+114] 0041B854 DB81 10010000 FILD DWORD PTR DS:[ECX+110] 0041B856 DA89 18010000 FINUL DWORD PTR DS:[ECX+118] 0041B858 D9F8 FPREM 0041B85A DDC1 FFREE ST(1) 0041B85C DB99 10010000 FISTP DWORD PTR DS:[ECX+110] 0041B85E BF 10000000 MOV EDI,10 0041B860 8B81 10010000 MOV EAX,DWORD PTR DS:[ECX+110] 0041B862 99 CQ 0041B864 F7F7 DIV EDI 0041B866 8A8411 00010000 MOV AL,BYTE PTR DS:[ECX+EDX+100] 0041B868 8B419 MOV BYTE PTR DS:[ECX+EBX],AL 0041B86A 43 INC EBX 0041B86C 83FB 0D CMP EBX,0D 0041B86E JNZ SHORT Windows_.0041B852 0041B870 C641 06 2D MOV BYTE PTR DS:[ECX+6],2D 0041B872 C641 0D 00 MOV BYTE PTR DS:[ECX+D],0 0041B874 C641 F4 0D MOV BYTE PTR DS:[ECX-C],0D 0041B876 E9 4799FEFF JMP Windows_.004051E4 </pre>	<p>empezamos con la primera letra</p>  <p>generamos en edx un numero al azar entre 0 y 15</p> <p>AL contiene una letra al azar del alfabeto la escribimos en serial en la posicion EBX siguiente letra hasta la letra decimotercera</p> <p>la septima letra del serial es un guion finalizador de toda de cadena ASCII incorporamos la longitud del serial</p>
---	---

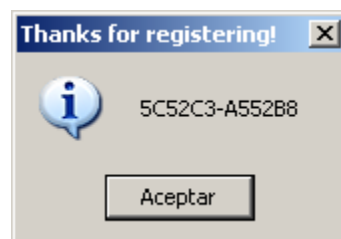
Al analizar el código de la rutina comprobadora del serial se deduce que éste debe contener 13 caracteres dígitos hexadecimales y con un guión en la posición séptima:

004028EB	EB 04	JMP SHORT Windows_.004028F1
004028ED	3C 2D	CMP AL,2D
004028EF	75 16	JNZ SHORT Windows_.00402907
004028F1	8B4424 24	MOV EAX,DWORD PTR SS:[ESP+24]

Por último si modificamos la instrucción 40520E por la que aparece subrayada:

00405202	E8 C9D4FFFF	CALL Windows_.004026D0	ASCII "Thanks for registering!"
00405207	6A 40	PUSH 40	
00405209	68 98D64100	PUSH Windows_.0041D698	
0040520E	FF7424 0C	PUSH DWORD PTR SS:[ESP+C]	
00405212	90	NOP	
00405213	8BCE	MOV ECX,ESI	

Obtendremos el serial en el MessageBox:



Método Fuerza Bruta

La idea es construir, uno a uno, todos los seriales en la dirección 00A463A8. Llamar a la rutina comprobadora y anotar los que sean válidos. Si el serial es inválido saltamos a una zona del código que está vacía donde escribiremos nuestro código (0041B800):

```
0041B800 8B7424 04 MOV ESI,DWORD PTR SS:[ESP+4]
0041B804 B9 0D000000 MOV ECX,0D
0041B809 49 DEC ECX
0041B80A 83F9 FF CMP ECX,-1
0041B80D 75 06 JNZ SHORT Windows_.0041B815
0041B80F FF15 40C14100 CALL DWORD PTR DS:[<&KERNEL32.ExitProcess>] kernel32.ExitProcess
0041B815 83F9 06 CMP ECX,6
0041B818 75 01 JNZ SHORT Windows_.0041B81B
0041B81A 49 DEC ECX
0041B81B 8A0431 MOV AL,BYTE PTR DS:[ECX+ESI]
0041B81E 3C 39 CMP AL,39
0041B820 75 04 JNZ SHORT Windows_.0041B826
0041B822 B0 41 MOV AL,41
0041B824 EB 0D JMP SHORT Windows_.0041B833
0041B826 3C 46 CMP AL,46
0041B828 75 07 JNZ SHORT Windows_.0041B831
0041B82A B0 31 MOV AL,31
0041B82C 8A0431 MOV BYTE PTR DS:[ECX+ESI],AL
0041B82F EB 08 JMP SHORT Windows_.0041B809
0041B831 FEC0 INC AL
0041B833 8A0431 MOV BYTE PTR DS:[ECX+ESI],AL
0041B836 C646 06 2D MOV BYTE PTR DS:[ESI+6],2D
0041B83A C646 0D 00 MOV BYTE PTR DS:[ESI+D],0
0041B83E C646 F4 0D MOV BYTE PTR DS:[ESI-C1],0D
0041B842 E9 9D99FEFF JMP Windows_.004051E4
```

La primera línea carga en ESI un puntero al serial (00A463A8). A continuación, se carga en ECX la longitud del serial.

ECX será la posición de la letra del serial que vamos a variar (incrementar), que será 0 para el primer carácter, 1 para el segundo, ..., y C para el último.

Si ECX = -1 hemos acabado el proceso de fuerza bruta y salimos del programa.

```
0041B80A 83F9 FF CMP ECX,-1
0041B80D 75 06 JNZ SHORT Windows_.0041B815
0041B80F FF15 40C14100 CALL DWORD PTR DS:[<&KERNEL32.ExitProcess>] kernel32.ExitProcess
```

Si ECX = 6, estamos en el séptimo carácter, que siempre debe ser un guión, no lo alteramos y actuamos con el siguiente carácter que es el sexto.

```
0041B815 83F9 06 CMP ECX,6
0041B818 75 01 JNZ SHORT Windows_.0041B81B
0041B81A 49 DEC ECX
```

Cargamos un carácter

```
0041B81B 8A0431 MOV AL,BYTE PTR DS:[ECX+ESI]
```

Nuestro alfabeto será “123456789ABCDEF” porque hemos aprendido de la parte d Prueba-Error que el cero no genera ningún serial válido. De tal manera que para avanzar un carácter simplemente incrementamos su valor una unidad; excepto el carácter ‘9’ (39) que debe pasar al ‘A’ (41)

```
0041B81E 3C 39 CMP AL,39
0041B820 75 04 JNZ SHORT Windows_.0041B826
0041B822 B0 41 MOV AL,41
0041B824 EB 0D JMP SHORT Windows_.0041B833
```

y el carácter ‘F’ que debe ser sustituido por ‘1’ e incrementar el carácter anterior

0041B826	3C 46	CMP AL,46	
0041B828	75 07	JNZ SHORT Windows_.0041B831	
0041B82A	B0 31	MOV AL,31	
0041B82C	880431	MOV BYTE PTR DS:[ECX+ESI],AL	
0041B82F	EB D8	JMP SHORT Windows_.0041B809	

Con el resto de caracteres basta con:

0041B831	FEC0	INC AL	
0041B833	880431	MOV BYTE PTR DS:[ECX+ESI],AL	

Por ejemplo:

El serial siguiente a “111111-111111” es “111111-111112”.
 El serial siguiente a “111111-111112” es “111111-111113”.
 El serial siguiente a “111111-111119” es “111111-11111A”.
 El serial siguiente a “111111-11111F” es “111111-111120”.
 El último serial es “FFFFFF-FFFFFF”.

Antes de saltar a la rutina comprobadora nos aseguramos que el sexto carácter sea un guión (2D), la cadena ASCII tenga el carácter del final (0) y anotemos la longitud del serial donde el programa lo lee:

0041B836	C646 06 2D	MOV BYTE PTR DS:[ESI+6],2D	
0041B83A	C646 0D 00	MOV BYTE PTR DS:[ESI+D],0	
0041B83E	C646 F4 0D	MOV BYTE PTR DS:[ESI-C],0D	
0041B842	E9 9D99FEFF	JMP Windows_.004051E4	

Ahora necesitamos escribir los seriales que vayamos cazando. Para ello vamos a saltar a una nueva dirección 0041B850 cuando el resultado de la comprobación sea positivo:

004051E4	8D4C24 04	LEA ECX,DWORD PTR SS:[ESP+4]	
004051E8	51	PUSH ECX	
004051E9	B3 08614200	MOV ECX,Windows_.00426108	
004051EB	E9 ADD5FFFF	CALL Windows_.004027A0	
004051F3	83F8 01	CMP EAX,1	
004051F6	75 2D	JNZ SHORT Windows_.00405225	
004051F8	E9 53660100	JMP Windows_.0041B850	
004051FD	B9 08614200	MOV ECX,Windows_.00426108	
00405202	E8 C9D4FFFF	CALL Windows_.004026D0	
00405207	6A 40	PUSH 40	
00405209	68 98D64100	PUSH Windows_.0041D698	ASCII "Thanks for registering!"

Vamos a incluir el siguiente código:

0041B84F	90	NOP	
0041B850	> 6A 00	PUSH 0	hTemplateFile = NULL
0041B852	. 68 80000000	PUSH 80	Attributes = NORMAL
0041B857	. 6A 04	PUSH 4	Mode = OPEN_ALWAYS
0041B859	. 6A 00	PUSH 0	pSecurity = NULL
0041B85B	. 6A 01	PUSH 1	ShareMode = FILE_SHARE_READ
0041B85D	. 68 00000040	PUSH 40000000	Access = GENERIC_WRITE
0041B862	. 68 BC841000	PUSH Windows_.0041B8BC	FileName = "RegCodes.txt"
0041B867	. FF15 F4F04300	CALL DWORD PTR DS:[<&kernel32.CreateFileA>]	CreateFileA
0041B86D	. 8946 20	MOV DWORD PTR DS:[ESI+20],EAX	
0041B870	. 6A 02	PUSH 2	Origin = FILE_END
0041B872	. 6A 00	PUSH 0	pOffsetHi = NULL
0041B874	. 6A 00	PUSH 0	OffsetLo = 0
0041B876	. FF76 20	PUSH DWORD PTR DS:[ESI+20]	hFile
0041B879	. FF15 F8F04300	CALL DWORD PTR DS:[<&kernel32.SetFilePointer>]	SetFilePointer
0041B87F	. 6A 00	PUSH 0	pOverlapped = NULL
0041B881	. 83C6 24	ADD ESI,24	pBytesWritten
0041B884	. 56	PUSH ESI	nBytesToWrite = 0 (13.)
0041B885	. 83EE 24	SUB ESI,24	Buffer
0041B888	. 6A 00	PUSH 0	hFile
0041B88A	. 56	PUSH ESI	WriteFile
0041B88B	. FF76 20	PUSH DWORD PTR DS:[ESI+20]	pOverlapped = NULL
0041B88E	. FF15 FCF04300	CALL DWORD PTR DS:[<&kernel32.WriteFile>]	WriteFile
0041B894	. 6A 00	PUSH 0	pOverlapped = NULL
0041B896	. 83C6 24	ADD ESI,24	pBytesWritten
0041B899	. 56	PUSH ESI	nBytesToWrite = 2
0041B89A	. 83EE 24	SUB ESI,24	Buffer = Windows_.0041C620
0041B89D	. 6A 02	PUSH 2	hFile
0041B89F	. 68 20C64100	PUSH Windows_.0041C620	WriteFile
0041B8A4	. FF76 20	PUSH DWORD PTR DS:[ESI+20]	hObject
0041B8A7	. FF15 FCF04300	CALL DWORD PTR DS:[<&kernel32.WriteFile>]	CloseHandle
0041B8AD	. FF76 20	PUSH DWORD PTR DS:[ESI+20]	
0041B8B0	. FF15 F0F04300	CALL DWORD PTR DS:[<&kernel32.CloseHandle>]	
0041B8B6	. E9 45FFFFFF	JMP Windows_.0041B800	
0041B8BB	. 90	NOP	
0041B8BC	. 52 65 67 43	ASCII "RegCodes.txt",0	
0041B8C9	. 00	DB 00	

Nuestro código va a tener tres variables, a saber:

[1] El serial, “123456-789ABC”, que estará en DS:[ESI], en mi caso la dirección 00A463A8.

[2] El manejador del fichero, que estará en DS:[ECX+20] en mi caso la dirección 00A464C8.

[3] La variable que contendrá el número de bytes escritos por la api WriteFile que estará en DS:[ECX+24] en mi caso la dirección 00A464CC.

Vamos a necesitar incluir las apis CloseHandle, CreateFileA, SetFilePointer y WriteFile que no son usadas en el programa. Para ello utilizo el programa IIDKing. Todas ellas residen en la librería Kernel.dll:



El propio programa nos indica como hacer las llamadas a las apis:

Below are the calls you can make to access your added functions...

Format style is: DLL Name::API Name->Call to API

```
kernel32.dll::CloseHandle->call dword ptr [43f0f0]
kernel32.dll::CreateFileA->call dword ptr [43f0f4]
kernel32.dll::SetFilePointer->call dword ptr [43f0f8]
kernel32.dll::WriteFile->call dword ptr [43f0fc]
```

El código lo que hace es abrir un fichero llamado RegCoded.txt (que puede existir o no), posicionarse al final del fichero, escribir el serial, escribir un salto de línea (CR + LF), cerrar el fichero y volver a generar el siguiente serial.

Una última cosa, hay que tener la precaución de introducir inicialmente el serial del cual queremos partir y que tiene que estar formado por letras de nuestro alfabeto, por ejemplo:

Agradecimientos



A todos los CracksLatinos, en especial a Guan de dio y a +NCR por sus comentarios.