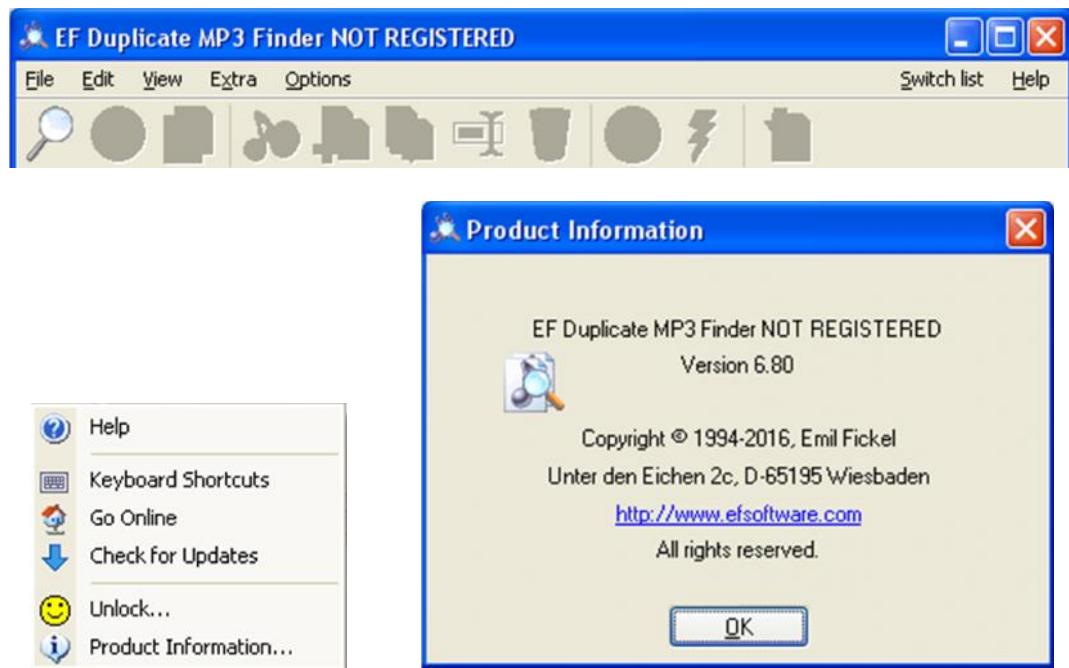




Estudiando la protección de registrado de EF Duplicate MP3 Finder 6.80

Fecha	9 de noviembre de 2016
Victima	EF Duplicate MP3 Finder 6.80
URL de descarga	http://www.efsoftware.com/dw/wzp.cgi?d3
MD5 del instalador	8A1C54C4BD462CADE358D8A5D742F7AC
Protección	Asprotect 2.xx y registrado con archivo de licencia
Herramientas	RDG Packer Detector, OllyDbg 1.10, plugin Ollydbg Conditional Branch Logger, plugin CodeDoctor, IDA Pro con Hexray, C++Builder 6
Objetivo	Estudiar la protección de licencia para crear un keygen
Dificultad	Media
Cracker	Aguml

Veo lo que me tiene preparado:



Y a los pocos segundos aparece esto:



Y al salir veo esto:



Si lo miro con RDG Packer Detector veo esto:



Como ya hemos visto en otros productos de este desarrollador, lo mejor para empezar es usar el truco de poner un BP en CreateFileW para llegar al punto donde se abrirá el archivo de licencia y, en ese momento, miro como se tiene que llamar el archivo de licencia:

0012A1D4	00463200	CALL to <code>CreateFileW</code> from EFD3F.004631FA
0012A1D8	0012DAA8	FileName = "C:\Documents and Settings\BlueDeep\Escritorio\ef_duplicate_mp3_finder_32\EFD3F.LIC"
0012A1DC	80000000	Access = GENERIC_READ
0012A1E0	00000003	ShareMode = FILE_SHARE_READ FILE_SHARE_WRITE
0012A1E4	00000000	pSecurity = NULL
0012A1E8	00000003	Mode = OPEN_EXISTING
0012A1EC	00000080	Attributes = NORMAL
0012A1F0	00000000	hTemplateFile = NULL
0012A1F4	0012C254	

Ya tengo el nombre del archivo de licencia con lo que creo uno cualquiera con cualquier keygen que ya cree o uso cualquier archivo de licencia existente aunque sea de otro producto de EF, lo renombro con ese nombre y lo coloco en el directorio del programa.

Quito el BP, doy Ctrl+F9, pongo un BP en ReadFile, doy a F9 y, cuando pare, voy en el Dump a la dirección apuntada por el parámetro del buffer:

0012C214	0045BDE9	CALL to <code>ReadFile</code> from EFD3F.0045BDE3
0012C218	00000114	hFile = 00000114
0012C21C	00C78ED8	Buffer = 00C78ED8
0012C220	0000033E	BytesToRead = 33E (830.)
0012C224	0012C238	pBytesRead = 0012C238
0012C228	00000000	pOverlapped = NULL

Quito el BP, doy a Ctrl+F9 y ya tengo el buffer lleno con toda la información del archivo de licencia. Busco la parte donde está el ID encriptado y pongo un HBP on Write en el primer carácter de éste.

Voy dando a F9 y voy viendo cómo va haciendo copias pero después de tanto ver productos de este desarrollador he aprendido que es más rápido encargarme del buffer original para llegar más rápido adonde necesito así que sigo dando a F9 hasta que veo esto:

004104D0	\$ 55	push	ebp
004104D1	. 0FB76C24 10	movzx	ebp, word ptr ss:[esp+10]
004104D6	. 56	push	esi
004104D7	. 8B7424 10	mov	esi, ss:[esp+10]
004104DB	. 57	push	edi
004104DC	. 8B7C24 10	mov	edi, ss:[esp+10]
004104E0	. 33C9	xor	ecx, ecx
004104E2	. B8 01000000	mov	eax, 1
004104E7	> 3BC5	cmp	eax, ebp
004104E9	.~ 7C 08	jl	short 004104F3
004104EB	. 8BC1	mov	eax, ecx
004104ED	. 83E0 03	and	eax, 3
004104F0	. 83C0 02	add	eax, 2
004104F3	> 8A1438	mov	dl, ds:[eax+edi]
004104F6	. 881431	mov	ds:[ecx+esi], dl
004104F9	. 41	inc	ecx
004104FA	. 40	inc	eax
004104FB	. 83F9 3A	cmp	ecx, 3A
004104FE	.^ 72 E7	jb	short 004104E7
00410500	. 5F	pop	edi
00410501	. 5E	pop	esi
00410502	. 5D	pop	ebp
00410503	. C3	ret	nop
00410504	CC		

Es la función que sustituye el ID con sucesiones del serial.

Quito el HBP on Access y pongo un HBP on Execution en la línea 410500 ya que ahí ya tendré todo el ID sustituido.

Es todo lo que necesito por ahora. Busco en todas las strings la cadena %0 para conseguir encontrar todas las que sean %0Xlu (donde X es un número) que son las que usará para la función _snprintf y pongo un BP en todas las interesantes que son estas:

Breakpoints

Address	Module	Active	Disassembly	Text String
0040CA08	EFD3F	Always	push 4CB120	ASCII "%04lu"
0040D80C	EFD3F	Always	push 4CB120	ASCII "%04lu"
0040FF49	EFD3F	Always	push 4CB148	ASCII "%09lu"
0041043D	EFD3F	Always	push 4CB120	ASCII "%04lu"
004105AB	EFD3F	Always	push 4CB128	ASCII "%05lu"
00420DD5	EFD3F	Always	push 4CB118	ASCII "%03lu"
0042247C	EFD3F	Always	push 4CB110	ASCII "%02lu"
0042615D	EFD3F	Always	push 4CB15C	ASCII "%06lu-%05lu-%08lu"
004261E4	EFD3F	Always	push 4CB15C	ASCII "%06lu-%05lu-%08lu"

Ya sé que en algún que otro caso va a usar la función similar a strncmp justo después de crear la cadena con _snprintf así que voy mirando qué hay en cada BP, lo quito y lo pongo en el call de debajo y miro si justo después hay un call con 3 push donde el primero de ellos es un valor constante y ese será el call de strncmp. Busco las referencias:

References in EFD3F: to 0043C174

Address	Disassembly	Comment
00401EA8	call 0043C174	EFD3F.0043C174

```

0040FF6C  call    0043C174          (Initial CPU selection)
00410495  call    0043C174          EFD3F.0043C174
0041F683  call    0043C174          EFD3F.0043C174
004260CD  call    0043C174          EFD3F.0043C174
00455220  call    0043C174          EFD3F.0043C174
004609BE  call    0043C174          EFD3F.0043C174
0049D903  call    0043C174          EFD3F.0043C174

```

Voy entrando una por una y poniendo un BP solo a las que se vean interesantes ya que algunas se ven claramente que comparan con cadenas que no nos interesan.

Así queda la lista de BPs:

Breakpoints

Address	Module	Active	Disassembly
0040CA14	EFD3F	Always	call 0043DD1A
0040D81B	EFD3F	Always	call 0043DD1A
0040FF55	EFD3F	Always	call 0043DD1A
0040FF6C	EFD3F	Always	call 0043C174
00410449	EFD3F	Always	call 0043DD1A
00410495	EFD3F	Always	call 0043C174
004105B6	EFD3F	Always	call 0043DD1A
0041F683	EFD3F	Always	call 0043C174
00420DE1	EFD3F	Always	call 0043DD1A
00422488	EFD3F	Always	call 0043DD1A
004260CD	EFD3F	Always	call 0043C174
00426169	EFD3F	Always	call 0043DD1A
004261F0	EFD3F	Always	call 0043DD1A
00455220	EFD3F	Always	call 0043C174
004609BE	EFD3F	Always	call 0043C174
0049D903	EFD3F	Always	call 0043C174

Doy a F9 y para aquí:

0041F67E	. 2BC7	sub eax, edi	
0041F680	. 50	push eax	
0041F681	. 51	push ecx	
0041F682	. 53	push ebx	
0041F683	. E8 ECCA0100	call 0043C174	EFD3F.0043C174
0041F688	. 83C4 0C	add esp, 0C	

0012DA2C	00C78FFB	ASCII LF,"Warning: This product is licensed to	
0012DA30	004CBBAC	ASCII "Name : "	
0012DA34	0000000C		

Es una de las llamadas a strncmp pero no es nada interesante lo que está haciendo. En realidad está recorriendo todo el buffer del archivo buscando que estén las cadenas "Name : ", "Company : ", "Serial No.: " y "Reg.ID. :" y si no están tendríamos problemas jejeje.

Como en el archivo hay todo eso, quito ese BP, doy a F9 y para aquí:

004609B4	. 8B4424 10	mov eax, ss:[esp+10]	
004609B8	. 57	push edi	
004609B9	. 50	push eax	
004609BA	. 8D4E 01	lea ecx, ds:[esi+1]	
004609BD	. 51	push ecx	
004609BE	. E8 B1B7FDFF	call 0043C174	EFD3F.0043C174
004609C3	. 83C4 0C	add esp, 0C	
004609C6	. 85C0	test eax, eax	

0012DA1C	00C72B26	ASCII "Player]",CR,LF,"Pos=22,29,300,250,1",CR,	
0012DA20	004CB21C	ASCII "Preset"	
0012DA24	00000006		

Es una de las llamadas a strncmp y está buscando algo en el archivo de configuración así que no me interesa. Quito ese BP y doy a F9 y para aquí:

			jmp	SNPRINT 00435251	EFD3F.00435251
0045521D	. 57		push	edi	
0045521E	. 53		push	ebx	
0045521F	. 56		push	esi	
00455220	. E8 4F6FFF	call	0043C174		EFD3F.0043C174
00455225	. 83C4 0C	add	esp, 0C		
00455228	. 85C0	test	eax, eax		
0045522A	.~ 75 25	jnz	short 00455251		EFD3F.00455251
0045522C	8A043F	mov	al, ds:[esi+4]		

Es otra llamada a strncmp y compara la ruta completa hacia un “archivo preset” con un buffer vacío. Supongo que lo que hace es buscar ese archivo en esa ruta y si no son iguales es porque no existe, como pasa en mi caso así que quito este BP también ya que no es de mi interés y doy a F9 y para aquí:

00426130	. BE B4B44C00	mov	esi, 4CB44D4		
00426141	> 8B46 04	mov	eax, ds:[esi+4]		
00426144	. 8B0E	mov	ecx, ds:[esi]		
00426146	. 8B56 FC	mov	edx, ds:[esi-4]		
00426149	. 05 FC5FC701	add	eax, 1C75FFC		
0042614E	. 50	push	eax		
0042614F	. 81C1 90030000	add	ecx, 390		
00426155	. 51	push	ecx		
00426156	. 81C2 3C0C0000	add	edx, 0C3C		
0042615C	. 52	push	edx		
0042615D	. 68 5CB14C00	push	4CB15C	ASCII "%06lu-%05lu-%08lu"	
00426162	. 8D4424 1C	lea	eax, ss:[esp+1C]		
00426166	. 6A 3F	push	3F		
00426168	. 50	push	eax		
00426169	. E8 AC7B0100	call	0043DD1A		EFD3F.0043DD1A
0042616E	. 83C4 18	add	esp, 18		
00426171	. 8BCB	mov	ecx, ebx		
00426173	. 8D4424 0C	lea	eax, ss:[esp+C]		
00426177	> 8A10	mov	dl, ds:[eax]		

Es la primera de las dos llamadas a _snprintf que realiza en la función de lista negra así que pongo los dos BP Conditional Log en las líneas:

```
00426173 |. 8D4424 0C |lea      eax, ss:[esp+C]
004261FA |. 8D4424 0C lea      eax, ss:[esp+C]
```

Con la siguiente configuración:

```
Expression: STRING[esp+C]
Pause program: Never
Log value of expresión: Alwais
```

Limpio el log y quito los BPs de las líneas:

```

00426169 |. E8 AC7B0100 |call 0043DD1A ; EFD3F.0043DD1A
004261F0 |. E8 257B0100 call 0043DD1A ; EFD3F.0043DD1A

```

Cambio los saltos:

```

0042617B /75 1A jnz short 00426197 ; EFD3F.00426197
00426204 /0F85 B6000000 jnz 004262C0 ; EFD3F.004262C0

```

Por dos saltos incondicionales y pongo un BP en la línea:

```
004262D5 |. 5F pop edi ; 00C78B40
```

Doy a F9 y en el Log ya tengo la lista negra de seriales:

Log data	
Address	Message
00426173	COND: 153554-41091-90750337
00426173	COND: 163243-71020-90751331
00426173	COND: 133173-33063-90851332
00426173	COND: 123086-72014-90750332
00426173	COND: 113590-62045-90850337
00426173	COND: 143066-52012-90750332
00426173	COND: 213596-51195-90760337
00426173	COND: 203506-61196-90760337
00426173	COND: 293516-71197-90760337
00426173	COND: 173736-92019-90750335
00426173	COND: 103002-42096-90751333
00426173	COND: 273536-91199-90760337
00426173	COND: 353556-01291-90770337
00426173	COND: 333576-21293-90770337
004261FA	COND: 103203-41016-90751331
004261FA	COND: 203006-73116-90760332
004261FA	COND: 193815-83017-90750344
004261FA	COND: 183029-72018-90751333
004261FA	COND: 173834-21029-90750334
004261FA	COND: 103708-23026-90750335
004261FA	COND: 193515-52027-90750337
004261FA	COND: 173134-13039-90751332
004261FA	COND: 173233-12069-90851331
004261FA	COND: 103505-62006-90850337
004261FA	COND: 193116-13007-90750331
004261FA	COND: 143165-82002-90750331
004261FA	COND: 183920-63008-90750333
004261FA	COND: 213498-21195-90760338
004261FA	COND: 183824-13008-90750334
004261FA	COND: 103504-36006-90750377
004261FA	COND: 193214-51007-90751331
004261FA	COND: 253255-71101-90761331
004261FA	COND: 103009-35006-90751373
004262D5	Breakpoint at EFD3F.004262D5

Quito el BP y los BPs condicionales y doy a F9 ya que mi serial no está en la lista negra y para aquí:

00410431	. 50	push	eax	
00410432	. FF15 A0014A00	call	ds:[4A01A0]	
00410438	. 8B4C24 58	mov	ecx, ss:[esp+58]	
0041043C	. 51	push	ecx	
0041043D	. 68 20B14C00	push	4CB120	ASCII "%04lu"
00410442	. 8D5424 18	lea	edx, ss:[esp+18]	
00410446	. 6A 3F	push	3F	
00410448	. 52	push	edx	
00410449	. E8 CCD802000	call	0043DD1A	EFD3F.0043DD1A
0041044E	. A1 589D4D00	mov	eax, ds:[4D9D58]	
00410453	. 8B88 86020000	mov	ecx, ds:[eax+286]	
00410459	. 83C4 10	add	esp, 10	
0041045C	. 3B4C24 10	cmp	ecx, ss:[esp+10]	
00410460	.~ 74 53	je	short 004104B5	EFD3F.004104B5

No veo ninguna cadena que me indique de dónde saca el valor que, si miro en ECX veo que es 0x2266. Si paso el call veo que el resultado es la cadena "8806" así que continuo hasta la línea de la comparación de más abajo y veo que lo va a comparar con las posiciones 55°, 56°, 57°, y 58° del ID desencriptado y mi ID solo tiene 57 caracteres pero hay espacio suficiente para sustituir mis caracteres por estos y que quede un carácter de fin de cadena detrás así que modiflico el ID para que contenga la cadena "8806" en esa posición.

Lo más raro es que este ID tiene 58 caracteres por ahora y no 57 como el resto y eso se podía apreciar también en la función que rellena el ID con sucesiones de seriales ya que la comprobación para salir del bucle era mayor por 1 más que en el resto de programas.

Traceo hasta la comparación y la paso sin problemas así que desactivo el BP del call y doy a F9 y ahora para aquí:

0042245A	. 33C0	xor	eax, eax	
0042245C	. 880C24	mov	ss:[esp], cl	
0042245F	. 8D48 01	lea	ecx, ds:[eax+1]	
00422462	. 56	push	esi	
00422463	> 0FB65404 05	movzx	edx, byte ptr ss:[esp+eax+5]	
00422468	. 0FB67404 04	movzx	esi, byte ptr ss:[esp+eax+4]	
0042246D	. 03D6	add	edx, esi	
0042246F	. 83C0 02	add	eax, 2	
00422472	. 83F8 3A	cmp	eax, 3A	
00422475	. 8D4C91 0E	lea	ecx, ds:[ecx+edx*4+E]	
00422479	.^ 7C E8	jl	short 00422463	EFD3F.00422463
0042247B	. 51	push	ecx	
0042247C	. 68 10B14C00	push	4CB110	ASCII "%02lu"
00422481	. 8D4424 48	lea	eax, ss:[esp+48]	
00422485	. 6A 3F	push	3F	
00422487	. 50	push	eax	
00422488	. E8 8DB80100	call	0043DD1A	EFD3F.0043DD1A
0042248D	. A1 589D4D00	mov	eax, ds:[4D9D58]	
00422492	. 8A4C24 50	mov	cl, ss:[esp+50]	
00422496	. 83C4 10	add	esp, 10	
00422499	. 5E	pop	esi	
0042249A	. 3A88 50020000	cmp	cl, ds:[eax+250]	
004224A0	.~ 74 06	je	short 004224A8	EFD3F.004224A8
004224A2	. FE80 BA000000 inc		byte ptr ds:[eax+BA1]	

Tiene toda la pinta de ser donde trabaja con el ID desencriptado con el primer carácter sustituido por un espacio así que doy a F8, sigo hasta la comparación y veo que va a comparar el primer carácter de la cadena obtenida con el primer carácter del ID desencriptado que es justo lo que hace en todos los productos que he visto de este desarrollador cuando obtiene el valor a partir del ID.

Sustituyo el primer carácter del ID para que coincidan, desactivo el BP, doy a F9 y ahora para aquí:

00426090	6 83EC 24	sub esp, 24	
00426093	. A1 D0184D00	mov eax, ds:[4D18D0]	
00426098	. 33C4	xor eax, esp	
0042609A	. 894424 20	mov ss:[esp+20], eax	
0042609E	. 56	push esi	
0042609F	. 8B35 589D4D00	mov esi, ds:[4D9D58]	
004260A5	. 83BE 44020000	cmp dword ptr ds:[esi+244], 0	Comprueba si esta lleno el buffer del archivo de licencia
004260AC	.~ 74 4A	je short 004260F8	EFD3F.004260F8
004260AE	. 8D4424 04	lea eax, ss:[esp+4]	
004260B2	. 50	push eax	
004260B3	. 8D8E BA000000	lea ecx, ds:[esi+BA]	Mueve el puntero al serial a ECX
004260B9	. 51	push ecx	
004260BA	. E8 01AFFEFF	call 00410FC0	Funcion que quita los guiones del serial
004260BF	. 6A 13	push 13	
004260C1	. 8D5424 10	lea edx, ss:[esp+10]	
004260C5	. 52	push edx	
004260C6	. 81C6 70020000	add esi, 270	
004260CC	. 56	push esi	
004260CD	. E8 A2600100	call 0043C174	_strncpy compara el serial sin guiones con parte del ID desencriptado
004260D2	. 83C4 14	add esp, 14	
004260D5	. 85C0	test eax, eax	
004260D7	.~ 74 1F	je short 004260F8	EFD3F.004260F8
004260D9	. 8B0D 589D4D00	mov ecx, ds:[4D9D58]	

Es otra llamada a strncpy y va a comparar la cadena del serial sin guiones con lo que haya en el ID desencriptado desde la posición 33º así que modiflico el ID en el Dump para que coincida con el serial sin guiones, desactivo el BP y doy a F9 y para aquí:

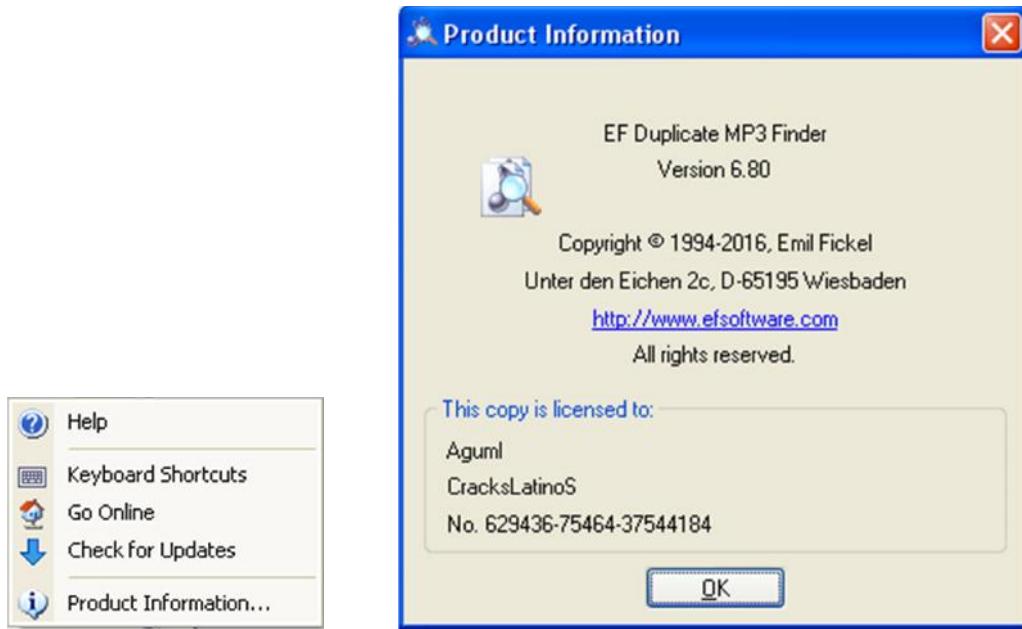
0040C9F5	. 56	push esi	
0040C9F6	> 0FB67404 04	movzx esi, byte ptr ss:[esp+eax+	
0040C9FB	. 03F0	add esi, eax	
0040C9FD	. 40	inc eax	
0040C9FE	. 8D4C31 FF	lea ecx, ds:[ecx+esi-1]	
0040CA02	. 3BC2	cmp eax, edx	
0040CA04	.^ 72 F0	jb short 0040C9F6	EFD3F.0040C9F6
0040CA06	. 5E	pop esi	
0040CA07	> 51	push ecx	
0040CA08	. 68 20B14C00	push 4CB120	ASCII "%04lu"
0040CA0D	. 8D5424 08	lea edx, ss:[esp+8]	
0040CA11	. 6A 3F	push 3F	
0040CA13	. 52	push edx	
0040CA14	. E8 01130300	call 0043DD1A	EFD3F.0043DD1A
0040CA19	. A1 589D4D00	mov eax, ds:[4D9D58]	
0040CA1E	. 8A4C24 10	mov cl, ss:[esp+10]	
0040CA22	. 83C4 10	add esp, 10	
0040CA25	. 3A88 69020000	cmp cl, ds:[eax+269]	
0040CA2B	.~ 75 18	jnz short 0040CA45	EFD3F.0040CA45
0040CA2D	. 8A5424 01	mov dl, ss:[esp+1]	
0040CA31	. 3A90 6A020000	cmp dl, ds:[eax+26A]	
0040CA37	.~ 75 0C	jnz short 0040CA45	EFD3F.0040CA45
0040CA39	. 8A4C24 02	mov cl, ss:[esp+2]	
0040CA3D	. 3A88 6B020000	cmp cl, ds:[eax+26B]	
0040CA43	.~ 74 10	je short 0040CA55	EFD3F.0040CA55
0040CA45	> 80R8 6B020000	cmpl byte ptr ds:[eax+26B], 0	
	0012F120	0012F130	ASCII "629436-75464-37544184"
	0012F124	0000003F	
	0012F128	004CB120	ASCII "%04lu"
	0012F12C	00000503	

Ahora lo que veo es que va a llamar a `_snprintf` y que ha estado trabajando con la cadena del serial con guiones así que ahí va a crear la cadena a partir del valor obtenido con las operaciones con la cadena del serial así que llego traceando con F8 hasta las comparaciones y veo que va a comparar los 3 primeros caracteres de la cadena obtenida con los caracteres 26º, 27º, y 28º del ID desencriptado así que modiflico el ID en el Dump para que coincidan, desactivo el BP, doy a F9 y para aquí:

00410590	. 33C9	xor	ecx, ecx	
00410592	. 8D51 0F	lea	edx, ds:[ecx+F]	
00410595	> 3BC8	cmp	ecx, eax	
00410597	.~ 73 11	jnb	short 004105AA	EFD3F.004105AA
00410599	. 0FB6740D 00	movzx	esi, byte ptr ss:[ebp+ecx]	
0041059E	. 03F2	add	esi, edx	
004105A0	. 8D1471	lea	edx, ds:[ecx+esi*2]	
004105A3	. 8D5402 01	lea	edx, ds:[edx+eax+1]	
004105A7	. 41	inc	ecx	
004105A8	.^ EB EB	jmp	short 00410595	EFD3F.00410595
004105AA	> 52	push	edx	
004105AB	. 68 28B14C00	push	4CB128	ASCII "%05lu"
004105B0	. 8D45 00	lea	eax, ss:[ebp]	
004105B3	. 6A 7F	push	7F	
004105B5	. 50	push	eax	
004105B6	. E8 5FD70200	call	0043DD1A	EFD3F.0043DD1A
004105BB	. A1 589D4D00	mov	eax, ds:[4D9D58]	
004105C0	. 8A4D 01	mov	cl, ss:[ebp+1]	
004105C3	. 83C4 10	add	esp, 10	
004105C6	. 3A88 6C020000	cmp	cl, ds:[eax+26C]	
004105CC	.~ 75 16	jnz	short 004105E4	EFD3F.004105E4
004105CE	. 8A55 02	mov	dl, ss:[ebp+2]	
004105D1	. 3A90 6D020000	cmp	dl, ds:[eax+26D]	
004105D7	.~ 75 0B	jnz	short 004105E4	EFD3F.004105E4
004105D9	. 8A4D 03	mov	cl, ss:[ebp+3]	
004105DC	. 3A88 6E020000	cmp	cl, ds:[eax+26E]	
004105E2	.~ 74 20	je	short 00410604	EFD3F.00410604
004105E4	~ 80B8 6E020000 cmp		byte ptr ds:[eax+26F1]	0
<hr/>				
0012F208	0012F438	ASCII "Agum1"		
0012F20C	0000007F			
0012F210	004CB128	ASCII "%05lu"		
0012F214	00001778			

Veo que ha estado trabajando con el nombre de registro y justo arriba veo las operaciones que ha realizado con él para obtener el valor así que lo paso y sigo hasta las comparaciones y veo que va a comparar los caracteres 2º, 3º, y 4º de la cadena obtenida con los caracteres 29º, 30º y 31º del ID desencriptado así que modiflico el ID en el Dump para que coincida, quito el BP y doy a F9 y no para más pero faltan aún más comprobaciones que he visto que se realizan en el resto de productos de este desarrollador como la del nombre de compañía y la de la concatenación nombre+serial+compañía así que me pongo a trastear en las opciones y botones.

Si doy en "Acerca de" no para en ningún sitio y aparezco como registrado:



Al cerrar esa ventana no para en ningún sitio así que ahí no es.

Doy al botón Buscar y no para en ningún sitio y me sale una ventana nueva para configurar la búsqueda así que configuro una búsqueda, doy a OK y para aquí:

0040FF20	\$ 83EC 44	sub	esp, 44	
0040FF23	. A1 D0184D00	mov	eax, ds:[4D18D0]	
0040FF28	. 33C4	xor	eax, esp	
0040FF2A	. 894424 40	mov	ss:[esp+40], eax	
0040FF2E	. A1 589D4D00	mov	eax, ds:[4D9D58]	
0040FF33	. 83B8 44020000	cmp	dword ptr ds:[eax+244], 0	
0040FF3A	.~ 74 47	je	short 0040FF83	EFD3F.0040FF83
0040FF3C	. 8B80 44020000	mov	eax, ds:[eax+244]	
0040FF42	. 50	push	eax	
0040FF43	. E8 38F2FFFF	call	0040F180	EFD3F.0040F180
0040FF48	. 50	push	eax	
0040FF49	. 68 48B14C00	push	4CB148	ASCII "%09lu"
0040FF4E	. 8D4C24 0C	lea	ecx, ss:[esp+C]	
0040FF52	. 6A 3F	push	3F	
0040FF54	. 51	push	ecx	
0040FF55	. E8 C0DD0200	call	0043DD1A	EFD3F.0043DD1A
0040FF5A	. A1 589D4D00	mov	eax, ds:[4D9D58]	
0040FF5F	. 6A 09	push	9	
0040FF61	. 8D5424 18	lea	edx, ss:[esp+18]	
0040FF65	. 52	push	edx	
0040FF66	. 05 51020000	add	eax, 251	
0040FF6B	. 50	push	eax	
0040FF6C	. E8 03C20200	call	0043C174	EFD3F.0043C174
0040FF71	. 83C4 20	add	esp, 20	
0040FF74	. 85C0	test	eax, eax	
0040FF76	.~ 74 0B	je	short 0040FF83	EFD3F.0040FF83
0040FF78	. A1 589D4D00	mov	eax, ds:[4D9D58]	

Tiene toda la pinta de ser donde crea la cadena a partir de las operaciones con el buffer del archivo de licencia con el ID sustituido por sucesiones del serial sin guiones así que traceo hasta llegar al siguiente call que es una llamada a strncmp y veo que va a comparar los 9 primeros caracteres de la cadena obtenida con lo que hay desde

la posición 2º del ID desencriptado con lo que modifíco el ID en el Dump para pasar la comprobación, pongo un BP al inicio de la función y quito el BP de ambas calls.

Doy a F9 y ahora para aquí:

0040D80B	. 57	push edi		
0040D80C	. 68 20B14C00	push 4CB120	ASCII "%04lu"	
0040D811	. 8D5424 18	lea edx, ss:[esp+18]		
0040D815	. 68 FF030000	push 3FF		
0040D81A	. 52	push edx		
0040D81B	. E8 FA040300	call 0043DD1A	EFD3F.0043DD1A	
0040D820	. A1 589D4D00	mov eax, ds:[4D9D58]		
0040D825	. 8B88 5A020000	mov ecx, ds:[eax+25A]		
0040D82B	. 83C4 10	add esp, 10		
0040D82E	. 3B4C24 10	cmp ecx, ss:[esp+10]		
0040D832	.~ 74 0A	je short 0040D83E	EFD3F.0040D83E	
0040D834	. 8A5434 10	mov dl, ss:[esp+esi+10]		
0117F270	0117F290	ASCII "Agum1629436-75464-37544184CracksLatinoS"		
0117F274	000003FF			
0117F278	004CB120	ASCII "%04lu"		
0117F27C	00003B2B			

Y veo que ha estado trabajando con la concatenación de nombre+serial+compañía y estas son las operaciones que realiza:

0040D7AD	.	0FB64424 14	movzx	eax, byte ptr ss:[esp+14]	
0040D7B2	.	33F6	xor	esi, esi	
0040D7B4	.	83FD 02	cmp	ebp, 2	
0040D7B7	.	894424 10	mov	ss:[esp+10], eax	
0040D7BB	.~	7C 33	jl	short 0040D7F0	EFD3F.0040D7F0
0040D7BD	.	0FB64C24 17	movzx	ecx, byte ptr ss:[esp+17]	
0040D7C2	.	8D55 FF	lea	edx, ss:[ebp-1]	
0040D7C5	>	0FB64434 14	movzx	eax, byte ptr ss:[esp+esi+1]	
0040D7CA	.	03C0	add	eax, eax	
0040D7CC	.	03C0	add	eax, eax	
0040D7CE	.	2BC6	sub	eax, esi	
0040D7D0	.	03C1	add	eax, ecx	
0040D7D2	.	03D8	add	ebx, eax	
0040D7D4	.	0FB64434 15	movzx	eax, byte ptr ss:[esp+esi+1]	
0040D7D9	.	03C0	add	eax, eax	
0040D7DB	.	03C0	add	eax, eax	
0040D7DD	.	2BC6	sub	eax, esi	
0040D7DF	.	03C1	add	eax, ecx	
0040D7E1	.	83C6 02	add	esi, 2	
0040D7E4	.	8D7C07 FF	lea	edi, ds:[edi+eax-1]	
0040D7E8	.	3BF2	cmp	esi, edx	
0040D7EA	.^	72 D9	jb	short 0040D7C5	EFD3F.0040D7C5
0040D7EC	.	8B4424 10	mov	eax, ss:[esp+10]	
0040D7F0	>	3BF5	cmp	esi, ebp	
0040D7F2	.	5D	pop	ebp	
0040D7F3	.~	73 12	jnb	short 0040D807	EFD3F.0040D807
0040D7F5	.	0FB64C34 10	movzx	ecx, byte ptr ss:[esp+esi+1]	
0040D7FA	.	0FB65424 13	movzx	edx, byte ptr ss:[esp+13]	
0040D7FF	.	8D0C8A	lea	ecx, ds:[edx+ecx*4]	
0040D802	.	2BCE	sub	ecx, esi	
0040D804	.	03C1	add	eax, ecx	
0040D806	.	46	inc	esi	
0040D807	>	03FB	add	edi, ebx	
0040D809	.	03F8	add	edi, eax	

Con lo que paso el call y llego hasta la línea 40D825 y veo que va a comparar los 4 primeros caracteres de la cadena obtenida con el valor de las operaciones con los caracteres 11º, 12º, 13º y 14º del ID desencriptado con lo que modifíco el ID en el Dump para que pase la comprobación, quito el BP y doy a F9 y no para nunca más. ¿Dónde está la comprobación con el nombre de la compañía? pues no consigo que pare en ella pulsando ningún botón pero la encuentro aquí:

00420DD4	. 56	push	esi		
00420DD5	. 68 18B14C00	push	4CB118	ASCII "%03lu"	
00420DDA	. 8D4424 14	lea	eax, ss:[esp+14]		
00420DDE	. 6A 7F	push	7F		
00420DE0	. 50	push	eax		
00420DE1	. E8 34CF0100	call	0043DD1A	EFD3F.0043DD1A	
00420DE6	. A1 589D4D00	mov	eax, ds:[4D9D58]		
00420DEB	. 8A4C24 1C	mov	cl, ss:[esp+1C]		
00420DEF	. 83C4 10	add	esp, 10		
00420DF2	. 5F	pop	edi		
00420DF3	. 5E	pop	esi		
00420DF4	. 5B	pop	ebx		
00420DF5	. 3A88 66020000	cmp	cl, ds:[eax+266]		
00420DFB	.~ 75 18	jnz	short 00420E15	EFD3F.00420E15	
00420DFD	. 8A5424 01	mov	dl, ss:[esp+1]		
00420E01	. 3A90 67020000	cmp	dl, ds:[eax+267]		
00420E07	.~ 75 0C	jnz	short 00420E15	EFD3F.00420E15	
00420E09	. 8A4C24 02	mov	cl, ss:[esp+21]		
00420E0D	. 3A88 68020000	cmp	cl, ds:[eax+268]		
00420E13	.~ 74 0F	je	short 00420E24	EFD3F.00420E24	
00420E15	~ 8088 68020000	cmp	byte ptr ds:[eax+268]		

Si me coloco en el principio de la función de donde está este código, hago New Origin Here y voy traceando veo que trabaja con el nombre de la compañía así que traceo hasta las comparaciones y veo que compara los 3 primeros caracteres de la cadena obtenida con los caracteres 23º, 24º y 25º del ID desencriptado así que modiflico el ID para pasar la comprobación y copio tanto en forma te texto como en forma binaria el ID desencriptado obtenido.

Para el archivo que he probado me queda el ID desencriptado:

"11172906101505#51,8806489126600732224062332767705097708806"

Binario:

```
31 31 31 37 32 39 30 36 31 30 31 35 30 35 23 35 31 2C 38 38 30 36 34 38 39 31 32 36 36 30
30 37 33 32 32 32 34 30 36 32 33 33 32 37 36 37 37 30 35 30 39 37 37 30 38 38 30 36
```

Con lo que si encripto ese ID y sustituyo el que está en el archivo de licencia por este tendría un archivo de licencia valido así que me voy al principio del evento clic del botón del keygen de cualquiera de los productos que ya he realizado su keygen y pongo esto:

```
id="11172906101505#51,8806489126600732224062332767705097708806";
EditID->Text = Encriptar(id);
return;
```

Y al dar al botón obtengo esto en el Edit:

"BBBHCJAGBABFAF%FB-IIAGEIJBCGGAAHDCCCEAGCDDCHGHHAFAJHHAIAG"

Sustituyo el ID del archivo por este, reinicio Olly, doy a F9 y, cuando pare en el HBP on Execution que tengo en la función que sustituye el ID por la sucesión de seriales, pongo todos los BPs en las llamadas interesantes a _snprintf y a strncmp y veo que paso todas las comparaciones pero hay algo que me está pillando y no doy con lo que es ya que las búsquedas aparecen en blanco y si quito el archivo de registro aparecen bien. Además si miro en la barra de estado veo que dice que la búsqueda se ha cancelado.

Ordeno el listado de las zonas del ID para ver qué zonas aún no se han usado del ID para la verificación:

```

1º -> Operaciones con el ID con espacio al principio.
2º a 10º -> Operaciones con el buffer del archivo de licencia.
11º a 14º -> Operaciones con la concatenacion nombre+serial+compañía.
15º a 22º -> ****?
23º a 25º -> Operaciones con el nombre de compañía.
26º a 28º -> Operaciones con el serial.
29º a 31º -> Operaciones con el nombre.
32º -> ****?
33º a 52º -> serial sin guiones
53º a 54º -> ****?
55º a 58º -> "8806"

```

"**11172906101505#51,8806489126600732224062332767705097708806**"

Intento ver adonde me lleva poner HBPs on Access en los caracteres que no son usados hasta ahora para ver si hace algún uso especial con ellos pero no me lleva a ningún sitio.

Por casualidad observo que, cada vez que doy al botón Buscar, la barra de estado de Olly me indica que se crea un hilo así que pongo un BP en CreateThread, doy al botón y cuando para veo esto:

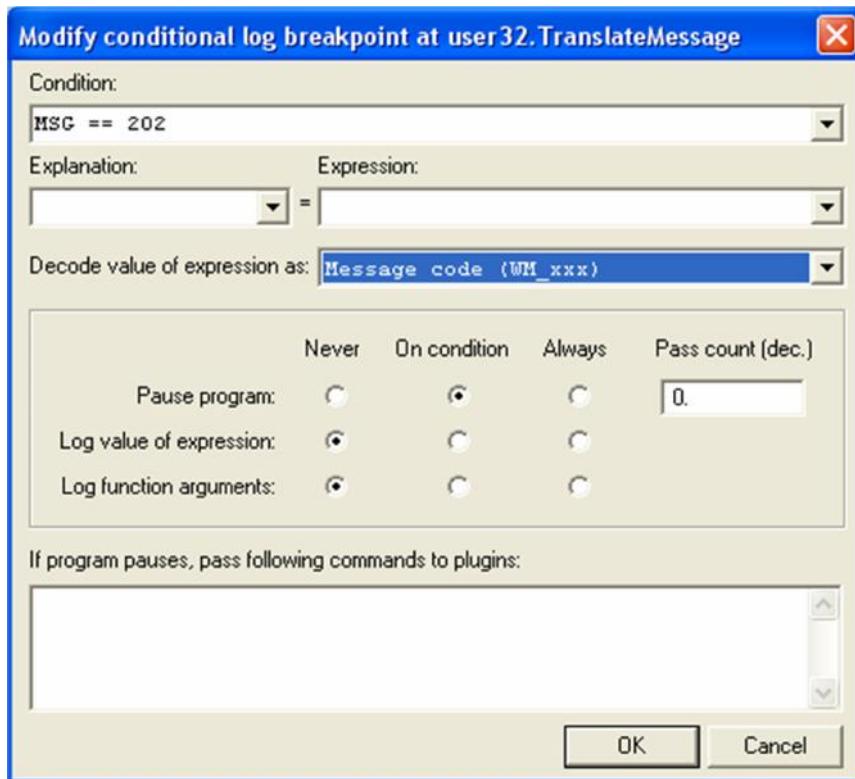
0012F2C4	00456FD3	CALL to CreateThread from EFD3F.00456FCD
0012F2C8	00000000	pSecurity = NULL
0012F2CC	00000000	StackSize = 0
0012F2D0	00435800	ThreadFunction = EFD3F.00435800
0012F2D4	003303C2	pThreadParm = 003303C2
0012F2D8	00000000	CreationFlags = 0
0012F2DC	0012F2E0	pThreadId = 0012F2E0
0012F2E0	00000000	

Voy a la dirección que se indica en el parámetro ThreadFunction y pongo un BP allí, doy a F9 y cuando para veo esto:

00435800	. 55	push	ebp	
00435801	. 8BEC	mov	ebp, esp	
00435803	. 6A FF	push	-1	
00435805	. 68 60F44900	push	49F460	
0043580A	. 64:A1 00000000	mov	eax, fs:[0]	
00435810	. 50	push	eax	
00435811	. 51	push	ecx	
00435812	. 53	push	ebx	
00435813	. 56	push	esi	
00435814	. 57	push	edi	
00435815	. A1 D0184D00	mov	eax, ds:[4D18D0]	
0043581A	. 33C5	xor	eax, ebp	
0043581C	. 50	push	eax	
0043581D	. 8D45 F4	lea	eax, [local.3]	
00435820	. 64:A3 00000000	mov	fs:[0], eax	
00435826	. 8965 F0	mov	[local.4], esp	
00435829	. 8B45 08	mov	eax, [arg.1]	
0043582C	. 50	push	eax	
0043582D	. C745 FC 000000	mov	[local.1], 0	
00435834	. E8 C7FAFFFF	call	00435300	EFD3F.00435300
00435839	. 83C4 04	add	esp, 4	
0043583C	. 33C0	xor	eax, eax	
0043583E	. 8B4D F4	mov	ecx, [local.3]	
00435841	. 64:890D 000000	mov	fs:[0], ecx	
00435848	. 59	pop	ecx	
00435849	. 5F	pop	edi	
0043584A	. 5E	pop	esi	
0043584B	. 5B	pop	ebx	
0043584C	. 8BE5	mov	esp, ebp	
0043584E	. 5D	pop	ebp	
0043584F	. C2 0400	retn	4	

Voy traceando hasta el call y entro y veo como crea un archivo para usarlo como caché y veo como hace las mismas comprobaciones con el serial, con el buffer del archivo de licencia y con el “8806” y no veo donde me pilla pero me pilla ya que se cancela la búsqueda con mi archivo de licencia.

Por otro lado, si pongo un BP condicional en TranslateMessage de la siguiente manera:



Cuando pulso en el botón Buscar para y veo esto:

```
0012FE4C 0043B993 CALL to TranslateMessage from EFD3F.0043B991
0012FE50 0012FE6C pMsg = WM_LBUTTONDOWN hw = 2A0330 (class="ToolbarWindow32") Keys = 0 X = 23. Y = 13.
0012FE54 D425E655
```

Pongo un BP Memory on Access en toda la sección code y voy dando a F9 hasta que llego aquí:

0043AA21	..	E9 10070000	jmp	0043B136	EFD3F.0043B136
0043AA26	>	0FB7F3	movzx	esi, bx	Case 111 (WM_COMMAND) of switch 0043A5CC
0043AA29	.	56	push	esi	
0043AA2A	.	E8 71C5FDFF	call	00416FA0	EFD3F.00416FA0
0043AA2F	.	83C4 04	add	esp, 4	
0043AA32	.	85C0	test	eax, eax	
0043AA34	..	0F85 F5060000	jnz	0043B12F	EFD3F.0043B12F
0043AA3A	.	8BC6	mov	eax, esi	
0043AA3C	.	3D 30750000	cmp	eax, 7530	Switch (cases 14..7530)
0043AA41	..	0F8F E3060000	jg	0043B12A	EFD3F.0043B12A
0043AA47	..	0F84 D4060000	je	0043B121	EFD3F.0043B121
0043AA4D	.	83E8 14	sub	eax, 14	

Voy traceando con F8 hasta que caigo aquí:

				EFD3F.0043B127
0043AE06	> A1 589D4D00	jmp	0043B127	Case 83 of switch 0043AA3C
0043AE0B	. 83B8 E4010000	cmp	dword ptr ds:[eax+1E4], 0	EFD3F.0043B12A
0043AE12	. 0F85 12030000	jnz	0043B12A	Aqui entra solo al dar al boton buscar
0043AE18	. 55	push	ebp	Llamada a la funcion que crea el hilo
0043AE19	. E8 12F7FFFF	call	0043A530	EFD3F.0043B127
0043AE1E	. . 04030000	jmp	0043B127	Case 84 of switch 0043AA3C
0043AE23	> A1 589D4D00	mov	eax, ds:[4D9D58]	
0043AE28	. 83B8 E4010000	cmp	dword ptr ds:[eax+1E4], 0	

Sigo con F8 y caigo aquí:

0043B121	> 55	push	ebp	Case 730 of switch 0043AA3C
0043B122	. E8 D94D0200	call	0045FF00	EFD3F.0045FF00
0043B127	> 83C4 04	add	esp, 4	Llamada a la funcion que obtiene la verificacion con el ID
0043B12A	> E8 F172FEFF	call	00422420	
0043B12F	> 8BB424 880800	mov	esi, ss:[esp+888]	lParam, Default case of switch 0043A703
0043B136	> 57	push	edi	wParam
0043B137	. 53	push	ebx	Message
0043B138	. 56	push	esi	hWnd
0043B139	. 55	push	ebp	DefWindowProcW
0043B13A	. FF15 E8044A00	call	ds:[4A04E8]	
0043B140	> 8B8C24 7C0800	mov	ecx, ss:[esp+87C]	EFD3F.0043BFB5
0043B147	. 5F	pop	edi	
0043B148	. 5E	pop	esi	
0043B149	. 5D	pop	ebp	
0043B14A	. 5B	pop	ebx	
0043B14B	. 33CC	xor	ecx, esp	
0043B14D	. E8 630E0000	call	0043BFB5	
0043B152	. 81C4 70080000	add	esp, 870	
0043B158	. C2 1000	retn	10	

Sigo hasta aquí:

0043B136	> 57	push	edi	lParam; Default case of switch 0043A703
0043B137	. 53	push	ebx	wParam
0043B138	. 56	push	esi	Message
0043B139	. 55	push	ebp	hWnd
0043B13A	. FF15 E8044A00	call	ds:[4A04E8]	DefWindowProcW
0043B140	> 8B8C24 7C0800	mov	ecx, ss:[esp+87C]	
0043B147	. 5F	pop	edi	

Y veo esto:

0012F2F0	003303C2	hWnd = 003303C2 ('EF Duplicate MP3 Finder', class='EFD3F')
0012F2F4	00000111	Message = WM_COMMAND
0012F2F8	00000083	Notify = MENU/BN_CLICKED... ID = 131.
0012F2FC	002A0330	hControl = 002A0330 (class='ToolbarWindow32', parent=003303C2)
0012F2F0	0012F2F4	

Aquí el listado de los comentarios que tengo puestos:

User-defined comments		Comment
Address Disassembly		
004260A5 cmp dword ptr ds:[esi+244], 0		Comprueba si está lleno el buffer del archivo de licencia
archivo de licencia		
004260B3 lea ecx, ds:[esi+BA]		Mueve el puntero al serial a ECX
004260BA call 00410FC0		Función que quita los guiones del serial

```

004260CD  call    0043C174
guiones con parte del ID desencriptado
0042675F  call    004547C0
00435275  cmp     byte ptr ds:[ecx+C0], 2D
primer guion es un guion o fue modificado
0043528C  call    0043D680
archivo de licencia
0043A57C  call    00456FB0
0043A586  jmp    00422420
con el primer carácter de espacio
0043AE18  push    ebp
buscar
0043AE19  call    0043A530
de búsqueda y dentro de esta se usa la función que comprueba el ID con el primer carácter como espacio
0043B12A  call    00422420
verificación con el ID con primer carácter espacio

```

`_strncpy` compara el serial sin
`call` que muestra la lista
Comprobación del serial mirando si el
`_strstr` busca la cadena de eax en el
Aquí crea el hilo de la búsqueda
Va a la función que comprueba el ID
Aquí entra solo al dar al botón
Llamada a la función que crea el hilo
Llamada a la función que obtiene la

Encontré un archivo de licencia por la red pero está en la lista negra así que hago lo siguiente:

- 1- Reinicio Olly y pongo un HBP al inicio de la comprobación de lista negra.
- 2- Guardo en lugar seguro mi registro que he creado yo y coloco en el directorio el que he encontrado por internet.
- 3- Doy a F9 y cuando pare en la función de lista negra cambio los dos saltos que evitan que me detecte y doy a F9.
- 4- Una vez ha arrancado ya el programa pongo un BP en estas dos direcciones:

```

00435834 | . E8 C7FAFFFF  call   00435300 ; EFD3F.00435300
00435839 | . 83C4 04      add    esp, 4

```

Son en el call que tiene las comprobaciones dentro de la función que se ejecuta al dar al botón cuando crea el hilo.

- 5- Doy a F9 y doy al botón Buscar y para en el primer BP.
- 6- Hago uso del plugin de Olly llamado Conditional Branch Logger eligiendo el rango 435300 a 435750 y elijo un archivo donde guardar el resultado del logeo.
- 7- Vuelvo a dar F9 y lo dejo trabajando hasta que pare en el segundo BP.

Reinicio Olly y repito todo el proceso pero ahora con mi archivo de licencia y al comparar veo que hay cambios en los saltos:

Listado de saltos con rango 435300 a 435750	
Saltos con serial descargado	Saltos con mi serial
0x004354d8 Jump Not Taken	0x004353c9 Jump Not Taken
0x004353d7 Jump Taken	0x004353d7 Jump Taken
0x00435435 Jump Not Taken	0x00435435 Jump Not Taken
0x00435444 Jump Not Taken	0x00435444 Jump Not Taken
0x00435458 Jump Taken	0x00435458 Jump Taken
0x004354d4 Jump Not Taken	0x004354d4 Jump Taken
0x004354d8 Jump Not Taken	
0x00435501 Jump Not Taken	0x00435501 Jump Taken
0x00435505 Jump Not Taken	
0x00435511 Jump Taken	
0x0043555e Jump Taken	0x0043555e Jump Not Taken
0x0043556f Jump Taken	0x0043559a Jump Not Taken
0x0043559a Jump Taken	

0x004355dc	Jump Not Taken
0x004355ed	Jump Not Taken
0x004355f1	Jump Not Taken
0x00435614	Jump Not Taken
0x00435643	Jump Not Taken
0x00435651	Jump Taken

Así que fuerzo los saltos para que se cumplan igual que con el archivo que descargué y al terminar y llegar al segundo BP doy a F9 y veo que sigo en las mismas, no me muestra nada y me sigue mostrando que la búsqueda se ha cancelado.

Ahora la idea es ir justo donde cambia el primero de los saltos y repetir la misma técnica para ver por qué cambia.

El primer salto que cambia está en 4354D4 así que hago lo mismo pero ahora con el rango 4351B0 a 4352F9 que es el primer call de los dos que hay justo encima del salto.

Listado de saltos con rango 4351B0 a 4352F9	
Saltos con serial descargado	Saltos con mi serial
0x00435241 Jump Not Taken	0x00435241 Jump Not Taken
0x0043524a Jump Not Taken	0x0043524a Jump Taken
0x00435264 Jump Taken	
0x00435241 Jump Taken	
0x0043527c Jump Not Taken	0x0043527c Jump Not Taken
0x00435296 Jump Not Taken	0x00435296 Jump Not Taken

Al igualar el comportamiento dentro de este call ya veo que mi archivo de licencia funciona perfectamente porque ya me salen los resultados correctamente.

Únicamente tengo que modificar el salto que se ve en la tabla que es diferente y el comportamiento en el resto de saltos es idéntico y al salir del call también se realizan idénticamente todos los saltos.

Si miro lo que hay en ese salto veo esto:

```
00435243 | . 83B9 E8010000>| cmp      dword ptr ds:[ecx+1E8], 0
0043524A | . 75 29           | jnz     short 00435275 ; EFD3F.00435275
```

Con lo que busco todas las referencias a la constante 1E8 y pongo un BP en todas las que modifiquen esa dirección, la modiflico para que valga 00 y pongo un BP en esa comparación y vuelvo a dar al botón Buscar. Estas son todas las referencias a esa constante:

References in EFD3F: to constant 1E8	Address	Disassembly	Comment
	004015B1	mov dword ptr ss:[esp+1E8], 3E80	
	00426450	cmp dword ptr ds:[eax+1E8], 0	
	00426560	cmp dword ptr ds:[eax+1E8], 0	
	00428892	mov ds:[ecx+1E8], eax	
	004308E8	cmp dword ptr ds:[eax+1E8], 0	
	004350E5	cmp dword ptr ds:[eax+1E8], 0	
	00435243	cmp dword ptr ds:[ecx+1E8], 0	(Initial CPU selection)
	004354CE	cmp ds:[ecx+1E8], ebx	
	004354FB	cmp ds:[ecx+1E8], ebx	
	00435558	cmp ds:[edx+1E8], ebx	
	004355E3	cmp ds:[eax+1E8], ebx	
	0043560D	cmp dword ptr ds:[eax+1E8], 0	
	0043563C	cmp dword ptr ds:[eax+1E8], 0	
	0043A572	mov dword ptr ds:[eax+1E8], 0	
	0043AE35	inc dword ptr ds:[eax+1E8]	

```
004767CB fst dword ptr ds:[esi+1E8]
```

Y pongo un BP en:

References in EFD3F: to constant 1E8		
Address	Disassembly	Comment
00428892	mov ds:[ecx+1E8], eax	
00435243	cmp dword ptr ds:[ecx+1E8], 0	(Initial CPU selection)
0043A572	mov dword ptr ds:[eax+1E8], 0	
0043AE35	inc dword ptr ds:[eax+1E8]	

Doy a F9 y para aquí:

Address	Disassembly	Comment	
0042887E	.~ 74 2B je short 004288AB	EFD3F.004288AB	
00428880	. OFB681 D80200 movzx eax, byte ptr ds:[ecx+2D8]		
00428887	. 0FBE15 C83041 movsx edx, byte ptr ds:[4D30C8]		
0042888E	. 3BC2 cmp eax, edx		
00428890	.~ 7C OC j1 short 0042889E	EFD3F.0042889E	
00428892	. 8981 E8010000 mov ds:[ecx+1E8], eax		
00428898	. 8B0D 589D4D00 mov ecx, ds:[4D9D58]		
0042889E	> 892D 80BB4C00 mov ds:[4CBB80], ebp		
004288A4	BF 01000000 mov edi, 1		

eax=00000074
ds:[00C74498]=00000000

Pongo un BP en la línea 428880 y vuelvo a dar a F9 y cuando para ahí en ese BP veo esto:

```
ds:[00C74588]=74
eax=00000000
```

Doy a F8 y veo esto en la siguiente línea:

```
ds:[004D30C8]=64 ('d')
edx=01176E01
```

Busco la constante 2D8 y estas son todas las referencias:

References in EFD3F: to 000002D8		
Address	Disassembly	Comment
004064A0	lea edx, ss:[esp+2D8]	
004227B6	cmp byte ptr ds:[edx+2D8], 0	COMPRUEBA SI YA HA MODIFICADO ESE BYTE
004227E0	mov ds:[edx+2D8], al	
00428880	movzx eax, byte ptr ds:[ecx+2D8]	(Initial CPU selection)
00429675	lea edx, ss:[ebp+2D8]	
00436F90	lea esi, ds:[ebx+2D8]	
0045945F	mov ebx, ss:[esp+2D8]	

En principio solo me interesan las que puedan modificar lo que haya en la dirección apuntada por la constante así que solo pongo un BP aquí:

```
004227E0 |. 8882 D8020000 mov ds:[edx+2D8], al
```

Si voy a esa línea veo esto:

004227B0	\$	8B15 589D4D00	mov	edx, ds:[4D9D58]
004227B6	.	80BA D8020000	cmp	byte ptr ds:[edx+2D8], 0
004227BD	.~	75 27	jnz	short 004227E6
004227BF	.	8A0D 2C9B4D00	mov	cl, ds:[4D9B2C]
004227C5	.	84C9	test	cl, cl
004227C7	.~	74 1D	je	short 004227E6
004227C9	.	8B82 001C0000	mov	eax, ds:[edx+1C00]
004227CF	.	8A40 OF	mov	al, ds:[eax+F]
004227D2	.	2AC1	sub	al, cl
004227D4	.	04 3A	add	al, 3A
004227D6	.	3C 39	cmp	al, 39
004227D8	.~	76 02	jbe	short 004227DC
004227DA	.	04 F6	add	al, 0F6
004227DC	>	02C0	add	al, al
004227DE	.	FEC0	inc	al
004227E0	.	8882 D8020000	mov	ds:[edx+2D8], al
004227E6	>	C3	retn	

Pongo un BP en el inicio de la función e intento ver que hace dando a F9 y al botón de Buscar pero no para en dicha línea. Reinicio Olly y, usando el archivo de licencia que me descargué, pongo un HBP on Execution en la función de lista negra y cuando pare habilito todos los BPs y ahora sí que para y puedo ir viendo lo que hace. Lo que me interesa saber es de dónde saca el valor que mete en CL así que busco referencias a la constante y veo esto:

References in EFD3F: to 004D9B2C		
Address	Disassembly	Comment
00404AC9	mov ds:[4D9B2C], al	
004227BF	mov cl, ds:[4D9B2C]	(Initial CPU selection)

Y la que me interesa es la primera así que hago doble clic encima y llego aquí:

00404AA0	\$	A1 589D4D00	mov	eax, ds:[4D9D58]	
00404AA5	.	83B8 001C0000	cmp	dword ptr ds:[eax+1C00], 0	EFD3F.00404AF8
00404AAC	.~	74 4A	je	short 00404AF8	Comprueba si el byte ya se escribió, lo salta
00404AAE	.	8078 0C 00	cmp	byte ptr ds:[eax+C], 0	
00404AB2	.~	74 1B	je	short 00404ACF	
00404AB4	.	8B80 001C0000	mov	eax, ds:[eax+1C00]	Mueve el segundo carácter de la tercera
00404ABA	.	0FB648 0B	movzx	ecx, byte ptr ds:[eax+B]	
00404ABE	.	6A 31	push	31	
00404AC0	.	51	push	ecx	
00404AC1	.	E8 FFA00000	call	0040EBC0	EFD3F.0040EBC0
00404AC6	.	83C4 08	add	esp, 8	
00404AC9	.	A2 2C9B4D00	mov	ds:[4D9B2C], al	
00404ACE	.	C3	retn		
00404ACF	>	8B90 001C0000	mov	edx, ds:[eax+1C00]	Obtiene el puntero al segundo carácter
00404AD5	.	0FB642 01	movzx	eax, byte ptr ds:[edx+1]	Mueve el tercer carácter del serial a e
00404AD9	.	6A 33	push	33	
00404ADB	.	50	push	eax	
00404ADC	.	E8 FFA00000	call	0040EBEO	EFD3F.0040EBEO
00404AE1	.	8B0D 589D4D00	mov	ecx, ds:[4D9D58]	
00404AE7	.	8841 0C	mov	ds:[ecx+C], al	
00404AEA	.	A1 589D4D00	mov	eax, ds:[4D9D58]	
00404AEF	.	83C4 08	add	esp, 8	
00404AF2	.	FF80 001C0000	inc	dword ptr ds:[eax+1C00]	Incremento el puntero a la cadena del s
00404AF8	>	C3	retn		

Pongo un BP en el principio de la función y reinicio y, cuando pare en la función de lista negra, habilito todos los BPs y voy dando a F9 hasta que pare en ese BP y voy traceando.

Veo que el primer salto no se cumple pero el segundo si y lo que va a usar en la call como parámetros son el valor constante 0x33 y el tercer carácter de la primera parte del serial y al entrar en la call veo esto:

0040EBE0	\$ 8A4424 04	mov al, ss:[esp+4]	
0040EBE4	. 2A4424 08	sub al, ss:[esp+8]	
0040EBE8	. 04 3A	add al, 3A	
0040EBEA	. 3C 39	cmp al, 39	
0040EBEC	.~ 76 02	jbe short 0040EBF0	EFD3F.0040EBF0
0040EBEE	. 04 F6	add al, 0F6	
0040EBF0	> C3	ret	

Sigo con F8 hasta salir de la función y al salir veo como guarda el resultado y vuelvo a dar a F9 y vuelve a parar en el mismo BP y al tracear no se cumple ninguno de los dos saltos y se ejecuta el código que hay justo debajo. Va a entrar a otro call pero ahora como parámetros usará el primer carácter de la tercera parte del serial y la constante 31 y al entrar en el call tengo esto:

0040EBCO	\$ 8A4424 04	mov al, ss:[esp+4]	
0040EBC4	. 024424 08	add al, ss:[esp+8]	
0040EBC8	. 2C 30	sub al, 30	
0040EBCA	. 3C 39	cmp al, 39	
0040EBCC	.~ 76 02	jbe short 0040EBD0	EFD3F.0040EBD0
0040EBCE	. 04 F6	add al, 0F6	
0040EBD0	> C3	ret	

Vuelvo a dar a F9 y vuelve a parar de nuevo en el BP y al tracear vuelvo al primer call y con los mismos parámetros y devuelve el mismo resultado así que doy a F9.

Vuelve a parar otra vez y otra vez llego a la misma call con idéntico resultado así que doy a F9.

Vuelve a parar otra vez con idéntico comportamiento y resultado así que otra vez doy a F9.

Luego cae aquí:

004227B0	\$ 8B15 589D4D00	mov edx, ds:[4D9D58]	
004227B6	. 80BA D8020000	cmp byte ptr ds:[edx+2D8], 0	
004227BD	.~ 75 27	jnz short 004227E6	
004227BF	. 8A0D 2C9B4D00	mov cl, ds:[4D9B2C]	
004227C5	. 84C9	test cl, cl	
004227C7	.~ 74 1D	je short 004227E6	
004227C9	. 8B82 001C0000	mov eax, ds:[edx+1C00]	
004227CF	. 8A40 0F	mov al, ds:[eax+F]	
004227D2	. 2AC1	sub al, cl	
004227D4	. 04 3A	add al, 3A	
004227D6	. 3C 39	cmp al, 39	
004227D8	.~ 76 02	jbe short 004227DC	
004227DA	. 04 F6	add al, 0F6	
004227DC	> 02C0	add al, al	
004227DE	. FEC0	inc al	
004227E0	. 8882 D8020000	mov ds:[edx+2D8], al	
004227E6	> C3	ret	

Y si vuelvo a dar a F9 vuelve a caer en la otra función y esta vez caigo en la primera call pero como parámetro usa otro carácter del serial. Después de mucho estudiar este comportamiento acabo comprendiendo lo que hace y

no es otra cosa que coger el 1º carácter de la tercera parte del serial y le incrementa en 1 y si es 0x39 lo convierte en 0x30, luego, usando otra función, hace una serie de operaciones con el resultado del primer carácter y el 5º carácter de la tercera parte del serial y guarda su resultado en la dirección a la que apunta la constante 2D8.

Las operaciones que hace con el 3º carácter de la primera parte del serial no se para que lo hace ya que puse un HBP on Access en donde guarda el resultado y nunca lo usa.

Con esto ya sé que el resultado que tengo que obtener tiene que ser menor que 0x64 que era con lo que se comparaba en la línea 428887.

Con ayuda de IDA y Hexray llego a conseguir pasar esas tres funciones a C++:

```
int func1(char a1, char a2)
{
    int result;

    result = a1 - a2 + 0x3A;
    if ( result > 0x39 )
        result -= 0xA;
    return result;
}

int func2(char a1, char a2)
{
    int result;

    result = a2 + a1 - 0x30;
    if ( result > 0x39 )
        result -= 0xA;
    return result;
}

int func3(AnsiString cadena, int valor)
{
    int v0;

    v0 = cadena[18] - valor + 0x3A;
    if ( v0 > 0x39 )
        v0 -= 0xA;
    v0 = 2 * v0 + 1;
    return v0;
}
```

La primera función es la que decrementa en 1, la segunda función es la que incrementa en 1 y la tercera es la que realiza los cálculos para obtener el resultado que se guarda. Después de darle muchas vueltas me di cuenta que la primera función es la inversa de la segunda con lo que para conseguir el primer carácter solo tengo que usar la primera función indicándole como parámetros el carácter 5º de la tercera parte del serial y el valor 0x31. El programa lo que hace es usar el primer carácter de la tercera parte del serial y el valor 0x31 con la segunda función. También podemos usar el primer carácter de la tercera parte del serial y el valor 0x31 con la segunda función y obtendríamos el valor del carácter 5º de la tercera parte del serial. Si lo hacemos de una de esas dos formas, la tercera función siempre devolverá 61 o 63 con lo que siempre pasamos esa comprobación del programa. Con esto último ya estoy listo para crear mi keygen y, aunque yo he modificado las funciones para adaptarlas a mi código, el funcionamiento es el mismo.

Aquí mi código:

```
//-----
#include <vccl.h>
#include <stdio.h>
```

```

#include <fstream.h>
#pragma hdrstop

#include "Unit1.h"
//-----
#pragma package(smart_init)
#pragma resource "*.*dfm"
TForm1 *Form1;

const char cadenaEncabezado[] = "EF Duplicate MP3 Finder distribution
license\x0D\x0A\x0D\x0A"
                                         "Copyright (c) 1994-2007, Emil Fickel\x0D\x0A"
                                         "All rights reserved.\x0D\x0A\x0D\x0A"
                                         "This copy is licensed to:\x0D\x0A";
const char cadenaNombre[] = "Name : ";
const char cadenaCompania[] = "Company : ";
const char cadenaSerial[] = "Serial No. : ";
const char cadenaID[] = "Reg. ID. : ";
const char cadenaPie[] = "\x0D\x0AWarning: This product is licensed to you pursuant to the
terms of the\x0D\x0A"
                                         "author license agreement included with the original
software.\x0D\x0A"
                                         "It is protected by copyright law and international
treaties.\x0D\x0A"
                                         "Unauthorized reproduction or distribution may result in severe
civil and\x0D\x0A"
                                         "criminal penalties, and will be prosecuted to the maximum extent
possible\x0D\x0A"
                                         "under the law.\x0D\x0A\x0D\x0A"
                                         "One registered copy of this software may be dedicated to a
single\x0D\x0A"
                                         "person who uses the software on one or more computers or to a
single\x0D\x0A"
                                         "workstation used by multiple people.\x0D\x0A";
const char salto[] = "\x0D\x0A";
//-----

_fastcall TForm1::TForm1(TComponent* Owner)
  : TForm(Owner)
{
}
//-----


AnsiString Encriptar(AnsiString cadena)
{
    DWORD valor;
    DWORD contador;
    DWORD aux;

    contador = 0;

    while(contador < cadena.Length()){
        valor = cadena[contador+1];
        if (valor >= 0x30) {
            valor += 0x41 - 0x30;
            cadena[contador+1]=(char)valor;
        } else {
            aux = contador;
            aux &= 0x80000003;
            if(aux < 0){
                aux--;
            }
        }
    }
}

```

```

        aux |= 0xfffffffC;
        aux++;
    }
    valor += aux;
    cadena[contador+1] = (char)valor;
}
contador++;
}
return cadena;
}
//-----
AnsiString Desencriptar(AnsiString cadena)
{
    DWORD valor;
    DWORD contador;
    DWORD aux;

    contador = 0;

    while(contador < cadena.Length()) {
        valor = cadena[contador+1];
        if (valor >= 0x40) {
            valor -= 0x41;
            valor = valor % 0xA;
            valor += 0x30;
            cadena[contador+1] = (char)valor;
        } else {
            aux = contador;
            aux &= 0x80000003;
            if(aux < 0){
                aux--;
                aux |= 0xfffffffC;
                aux++;
            }
            valor -= aux;
            cadena[contador+1] = (char)valor;
        }
        contador++;
    }
    return cadena;
}
//-----
AnsiString CalcularValorSerial(AnsiString serial)
{
    unsigned long retval = 0;
    int contador;

    for(contador = 0; contador < serial.Length(); contador++){
        retval += contador + serial[contador + 1] - 1;
    }

    return AnsiString().sprintf("%04lu", retval);
}
//-----
AnsiString CalcularValorNombre(AnsiString nombre)
{
    unsigned long total = 0xF;

```

```

        for( int contador = 0; contador < nombre.Length(); contador++ )
    {
        total = contador + (nombre[contador + 1] + total) * 2 + nombre.Length() + 1;
    }

    return AnsiString().sprintf("%05lu", total);
}
//-----

AnsiString ObtenerValorCompania(AnsiString compania)
{
    unsigned long retval=0x17;
    int contador=0;

    if (compania.Length() >= 2){
        for(contador = 0; contador < compania.Length() - 1; contador += 2){
            retval += compania.Length() + compania[contador + 1] + compania.Length() +
compania[contador + 2];
        }
    }
    if ( contador < compania.Length() )
        retval += compania[contador + 1] + compania.Length();
    retval %= 0x3E8;
    return AnsiString().sprintf("%03lu", retval);
}
//-----

AnsiString CalcularValorLIC(char *cadena,int sizebuffer)
{
    unsigned long valorl = 0;
    unsigned long retval = sizebuffer;
    int pos;

    for(int contador = 0; contador < sizebuffer; contador++)
    {
        valorl ^= cadena[contador] + contador + 6;
        pos = (valorl % (sizebuffer + 1)) + 1;
        if ( pos > sizebuffer )
            pos = 0x16;
        retval = cadena[pos] + valorl * (retval + 3) + 2 * cadena[contador] + 0xB;
    }

    return AnsiString().sprintf("%09lu", retval);
}
//-----
```

```

AnsiString CalcularValorID(AnsiString id)
{
    unsigned long total = 1;

    id[1] = 0x20;

    for(int contador=0; contador < id.Length(); contador += 2)
    {
        total += 4 * (id[contador + 1] + id[contador + 2]) + 0xE;
    }

    return AnsiString().sprintf("%02lu", total);
}
//-----
```

```

//Limpiar buffer
void LimpiarBuffer(char *buffer,int sizeBuffer)
{
    for(int i = 0; i < sizeBuffer;buffer[i] = '\0', i++);
}
//-----

AnsiString ValorDeConcatenados(AnsiString cadena)
{
    unsigned long retval = cadena[1];
    unsigned long valor1 = 0, valor2 = 0;
    int contador = 0;

    if ( cadena.Length() >= 2 )
    {
        for(contador = 0; contador < cadena.Length() - 1; contador += 2){
            valor1 += cadena[4] + 4 * cadena[contador + 1] - contador;
            valor2 += cadena[4] + 4 * cadena[contador + 2] - contador - 1;
        }
    }
    if ( contador < cadena.Length() )
    {
        retval += cadena[4] + 4 * cadena[contador + 1] - contador;
    }

    retval += valor1 + valor2;
    return AnsiString().sprintf("%04lu", retval);
}
//-----


*****Las siguientes funciones son para la obtención del carácter 14 a partir del
*****carácter 18 del serial.
*****La función CalcularDecDigito es la inversa a CalcularIncDigito.
*****int CalcularDecDigito(int digito, int decremento)
{
    digito -= decremento;
    if ( digito < 0 )
        digito += 10;
    return digito;
}
//-----


int CalcularIncDigito(int digito, int incremento)
{
    digito += incremento;
    if ( digito > 9 )
        digito -= 10;
    return digito;
}
//-----


//Se le pasaría como primer parámetro el dígito de la posición 18 del serial (digito1)
//y como segundo parámetro el dígito de la posición 14 (digito2)
//Si devuelve 1 o 3 pasamos el control
int ComprobarDigito(int digito1, int digito2)
{
    digito1 -= digito2;
    digito1 +=10;
    if ( digito1 > 9 )

```

```

        digitol -= 10;
        digitol *= 2;
        digitol++;
        return digitol;
    }
//-----

unsigned long Exponente(int e)
{
    return (e > 0) ? 10 * Exponente(e - 1) : 1;
}
//-----

ObtenerDigito(unsigned long valor, unsigned indice)
{
    valor %= Exponente(indice + 1);
    valor /= Exponente(indice);
    return valor;
}
//-----

int unsigned long ReemplazarDigito(unsigned long valor, unsigned indice, unsigned nuevoDigito)
{
    unsigned long superior = valor / Exponente(indice + 1);
    unsigned long inferior = valor % Exponente(indice);
    return (superior * 10 + nuevoDigito) * Exponente(indice) + inferior;
}
//-----

void __fastcall TForm1::ButtonCalcularClick(TObject *Sender)
{
    AnsiString id, aux;
    unsigned long serialP1, serialP2, serialP3;
    int digito;

    //Si ya hay memoria asignada al buffer la libero antes de nada
    if(bufferLleno){
        delete bufferArchivo;
        bufferArchivo = NULL;
        bufferLleno = false;
    }

    do{
        //Obtengo las 3 partes del serial con valores aleatorios
        serialP1 = random(999999);
        serialP2 = random(99999);
        serialP3 = random(99999999);

        if(ComprobarDigito(ObtenerDigito(serialP3, 3),ObtenerDigito(serialP3, 7)) > 3){
            //Obtengo el cuarto dígito por la derecha de la tercera parte del serial
            digito = ObtenerDigito(serialP3,3);
            //Ejecuto la función para obtener el valor válido para el 8 dígito por la
derecha
            digito = CalcularDecDigito(digito,1);
            //Reemplazo el valor del octavo dígito por la derecha por el valido
            serialP3 = ReemplazarDigito(serialP3,7,digito);
        }

        //Le coloco un carácter valido en la primera posición de la tercera parte del
serial
    }
}
```

```

EditSerial->Text = AnsiString().sprintf("%06lu-%05lu-%08lu", serialP1, serialP2,
serialP3);
}while(listaNegra->IndexOf(EditSerial->Text) != -1); //Si el serial está en la lista
negra creo otro diferente

//concateno las partes del serial sin guiones
aux = AnsiString().sprintf("%06lu%05lu%08lu", serialP1, serialP2, serialP3);

//Relleno el ID con una sucesión del serial pero empezando siempre por el segundo
//carácter hasta llegar a 58 caracteres
id="";
for(int x=0, pos=1; x < 58; x++, pos++){
    if(pos >= aux.Length())
        pos = (x & 3) + 2;
    id += (char)aux[pos+1];
}

//Obtengo el tamaño que ocupará todo el texto
sizeBuffer = snprintf(NULL, 0, "%s%s%s%s%s%s%s%s%s%s%s",
                      cadenaEncabezado,
                      cadenaNombre, EditNombre->Text.c_str(), salto,
                      cadenaCompania, EditCompania->Text.c_str(), salto,
                      cadenaSerial, EditSerial->Text.c_str(), salto,
                      cadenaID, id.c_str(), salto,
                      cadenaPie);

sizeBuffer++;

//Reservo la memoria necesaria para el buffer
bufferArchivo = new char[sizeBuffer];

//Limpio el buffer
LimpiarBuffer(bufferArchivo, sizeBuffer);

//Concateno todo como lo requiere la operación para obtener el valor sobre el texto
//de la licencia. Viene a ser como el original pero usando el ID con la concatenación
//de seriales sin guiones
snprintf(bufferArchivo, sizeBuffer-1, "%s%s%s%s%s%s%s%s%s%s",
          cadenaEncabezado,
          cadenaNombre, EditNombre->Text.c_str(), salto,
          cadenaCompania, EditCompania->Text.c_str(), salto,
          cadenaSerial, EditSerial->Text.c_str(), salto,
          cadenaID, id.c_str(), salto,
          cadenaPie);

//Creo un ID con caracteres aleatorios desde el carácter 0x21 que es '!' hasta
//el carácter 0x39 que es '9' y como 0x39 - 0x21 = 0x18 uso ese valor para crear
//el valor aleatorio y luego le sumo el valor del primero para obtener un carácter
//dentro de ese rango. Tiene que ser ese rango porque, al desencriptar el ID,
//cualquier ID no se sale de este rango y si ponemos un rango mayor no funcionará
id = "";
for(int i = 0; i < 58; i++){
    id += (char)(random(0x18) + 0x21);
}

//Desencripto el ID
id = Desencriptar(id);

//Concateno las partes del serial sin guiones
aux = AnsiString().sprintf("%06lu%05lu%08lu", serialP1, serialP2, serialP3);

//Coloco el serial sin guiones en el ID desencriptado

```

```

id = id.SubString(1,32) + aux + id.SubString(52, id.Length()- 51);

//Obtengo la cadena con el valor del nombre introducido
aux = CalcularValorNombre(EditNombre->Text);

//Coloco los caracteres 2, 3, y 4 de la cadena obtenida en el ID desencriptado
id = id.SubString(1, 28) + aux.SubString(2, 3) + id.SubString(32, id.Length() - 31);

//Obtengo la parte del "8806"
aux = AnsiString().sprintf("%04lu", 0x2266);

//La coloco en el ID
id = id.SubString(1, 54) + aux.SubString(1, 4);

//Obtengo el valor a partir del serial
aux = CalcularValorSerial(EditSerial->Text);

//Coloco los 3 primeros caracteres del valor obtenido en el ID
id = id.SubString(1, 25) + aux.SubString(1, 3) + id.SubString(29, id.Length() - 28);

//Obtengo el valor a partir de la compañía
aux = ObtenerValorCompania(EditCompania->Text);

//La coloco en su posición
id = id.SubString(1, 22) + aux.SubString(1, 3) + id.SubString(26, id.Length() - 25);;

//Obtengo la cadena con el valor que se obtiene a partir del buffer de licencia
aux= CalcularValorLIC(bufferArchivo, sizeBuffer - 1);

//Coloco los 9 primeros caracteres de la cadena en el ID
id = id.SubString(1, 1) + aux.SubString(1, 9) + id.SubString(11, id.Length() - 10);

//Obtengo el valor del nombre, serial y compañía concatenados
aux = ValorDeConcatenados(EditNombre->Text + EditSerial->Text + EditCompania->Text);
id = id.SubString(1,10) + aux.SubString(1,4) + id.SubString(15, id.Length() - 14);

//Obtengo la cadena que se genera a partir del ID
aux = CalcularValorID(id);

//Coloco el que necesito del resultado en su posicion en el ID
id = aux.SubString(1,1) + id.SubString(2, id.Length() - 1);

//Encripto el ID y lo muestro
id = Encriptar(id);

//Como al desencriptar, si es mayor o igual a 0x40, va a dividir su valor entre 0xA
//que es 10 y quedarse con el resto, cualquier valor que sea mayor o igual a 0x40
//podemos multiplicarlo por 10, 20, o 30 sin temor de salirnos de los valores de la
//tabla ASCII y así tener un ID que contenga más caracteres ya que si no solo saldrían
//los menores de 0x40 y el rango de la A a la J
for(int i=1; i<=58; i++)
    if(id[i] >= 0x40)
        id[i] = id[i] + (random(3) * 10);

//Limpio el buffer
LimpiarBuffer(bufferArchivo, sizeBuffer);

//Monto el buffer con la cadena de licencia final
snprintf(bufferArchivo, sizeBuffer - 1, "%s%s%s%s%s%s%s%s%s%s%s",
         cadenaEncabezado,
         cadenaNombre, EditNombre->Text.c_str(), salto,

```

```

        cadenaCompania, EditCompania->Text.c_str(), salto,
        cadenaSerial, EditSerial->Text.c_str(), salto,
        cadenaID, id.c_str(), salto,
        cadenaPie);
bufferLleno = true;

EditID->Text = id;
ButtonCrearLIC->Enabled = true;
}

//-----
AnsiString GetLastErrorAsString()
{
    //Get the error message, if any.
    DWORD errorMessageID = GetLastError();
    if(errorMessageID == 0)
        return NULL; //No error message has been recorded

    LPSTR messageBuffer = NULL;
    FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM |
FORMAT_MESSAGE_IGNORE_INSERTS,
                NULL,
                errorMessageID,
                MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
                (LPSTR)&messageBuffer,
                0,
                NULL);

    AnsiString message = messageBuffer;

    //Free the buffer.
    LocalFree(messageBuffer);

    return message;
}
//-----

void __fastcall TForm1::ButtonCrearLICClick(TObject *Sender)
{
    ofstream archivoLIC;
    AnsiString error;

    //Abro el archivo de licencia
    archivoLIC.open("EFD3F.LIC", ios::binary | ios::out);

    if(archivoLIC.fail()){
        Application->MessageBoxA(AnsiString("No se pudo crear el archivo de licencia.\n") +
GetLastErrorAsString().c_str(),
                            "Error",
                            MB_ICONERROR);
        return;
    }
    //Guardo la informacion en el
    archivoLIC.write(bufferArchivo, sizeBuffer - 1);

    //Cierro el archivo de licencia
    archivoLIC.close();
    Application->MessageBoxA("El archivo de licencia se ha creado satisfactoriamente.\nNo
olvide colocarla en el directorio del programa.",
                            "Enhorabuena",
                            MB_ICONINFORMATION);
}

```

```

}

//-----
void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Action)
{
    if(bufferLleno)
        delete bufferArchivo;
    bufferArchivo != NULL;
    bufferLleno = false;
    delete listaNegra;
}
//-----

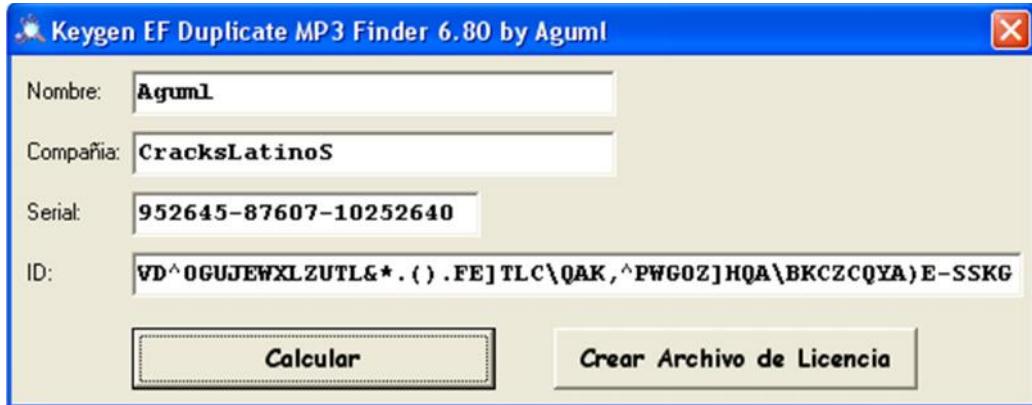
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    randomize();
    bufferLleno = false;
    bufferArchivo = NULL;
    listaNegra = new TStringList;

    listaNegra->Add("703901-21056-90750939");
    listaNegra->Add("823187-21184-90760937");
    listaNegra->Add("733870-22003-90851931");
    listaNegra->Add("743160-52002-90850937");
    listaNegra->Add("703605-21076-90751933");
    listaNegra->Add("703700-38066-90750971");
    listaNegra->Add("763143-24060-90750947");
    listaNegra->Add("713990-83075-90750939");
    listaNegra->Add("333179-21233-90771931");
    listaNegra->Add("463648-42330-90780935");
    listaNegra->Add("823585-01164-90760933");
    listaNegra->Add("483628-22338-90780935");
    listaNegra->Add("493618-12337-90780935");
    listaNegra->Add("913196-61255-90770937");
    listaNegra->Add("383628-32238-90770935");
    listaNegra->Add("703306-82076-90750935");
    listaNegra->Add("733672-32053-90751933");
    listaNegra->Add("593619-92437-90790935");
    listaNegra->Add("403608-02336-90780935");
    listaNegra->Add("003201-42346-98780936");
    listaNegra->Add("363648-52230-90770935");
    listaNegra->Add("323688-92234-90770935");
    listaNegra->Add("343668-72232-90770935");
    listaNegra->Add("793815-24077-90751941");
    listaNegra->Add("733678-52073-90751933");
    listaNegra->Add("703406-64086-90750954");
    listaNegra->Add("713398-33085-90750935");
    listaNegra->Add("793311-14087-90750935");
    listaNegra->Add("713292-92085-90750946");
    listaNegra->Add("713095-33085-90750948");
    listaNegra->Add("733674-72093-90751933");
    listaNegra->Add("763641-23030-90851933");
    listaNegra->Add("393319-55237-90770958");
    listaNegra->Add("343162-63232-90771931");
    listaNegra->Add("843323-01148-90760935");
    listaNegra->Add("763352-01041-90750935");
    listaNegra->Add("703900-87046-90750979");
    listaNegra->Add("713590-53065-90750933");
    listaNegra->Add("713599-45065-90750973");
    listaNegra->Add("723389-42064-90750935");
    listaNegra->Add("733873-41063-90751931");
}

```

```
listaNegra->Add( "733075-43053-90750948" );
listaNegra->Add( "773034-13059-90750938" );
listaNegra->Add( "713993-71075-90750939" );
listaNegra->Add( "833475-53173-90760944" );
listaNegra->Add( "733873-31073-90751931" );
}
//-----
```

Y el resultado:



Y esto es todo.

Espero que hayan disfrutado con la lectura.

Por último agradecer a todos los CracksLatinoS por estar siempre que los necesito para ayudarme.