



## Examinando el EF Find 8.50 Portable

Fecha	27 de octubre de 2016
Victima	EF Find 8.50 Portable
URL de descarga	<a href="http://www.efsoftware.com/dw/wzp.cgi?fn">http://www.efsoftware.com/dw/wzp.cgi?fn</a>
MD5 del instalador	021502E5D34440A72276DEF5EAD54BAD
Protección	Asprotect 2.xx, archivo de licencia
Herramientas	RDG Packer Detector, Ollydbg 1.10, IDA, C++Builder
Objetivo	Estudiar su sistema de registro
Dificultad	media
Cracker	Aguml

Lo primero es analizar con que está protegido:



Si ejecuto el programa:



Si doy en “Acerca de”:



Y al cerrar el programa:



Como ya vi antes al EF Checksum Manager, intentaré usar algunos trucos por si usa los mismos algoritmos para saber si estoy registrado. Lo primero es saber cómo se debe llamar el archivo de licencia así que lo cargo en Olly y pongo un BP en la API CreateFileW y voy dando a F9 hasta que llego aquí:

0012A228	0043EE70	CALL to <b>CreateFileW</b> from EFFind.0043EE6A
0012A22C	0012DAFC	FileName = "C:\Documents and Settings\BlueDeep\Escritorio\ef_find_32\EFFin.LIC"
0012A230	80000000	Access = GENERIC_READ
0012A234	00000003	ShareMode = FILE_SHARE_READ FILE_SHARE_WRITE
0012A238	00000000	pSecurity = NULL
0012A23C	00000003	Mode = OPEN_EXISTING
0012A240	00000080	Attributes = NORMAL
0012A244	00000000	hTemplateFile = NULL

Pues ahora creo un archivo usando como plantilla el de licencia del EF Checksum Manager y cambiando donde aparece un nombre de producto por el otro y lo guardo con ese nombre poniéndolo junto al ejecutable.

Reinicio Olly y repito el proceso y cuando pare en CreateFileW para abrir el archivo de licencia doy a Ctrl+F9, quito el BP de CreateFileW y pongo un BP en ReadFile.

Doy a F9 y para aquí:

0012C268	00436F09	CALL to <b>ReadFile</b> from EFFind.00436F03
0012C26C	000000FC	hFile = 000000FC (window)
0012C270	00C0B608	Buffer = 00C0B608
0012C274	0000032D	BytesToRead = 32D (813.)
0012C278	0012C28C	pBytesRead = 0012C28C
0012C27C	00000000	pOverlapped = NULL

Me voy en el Dump a la dirección que indica el buffer y doy a Ctrl+F9, quito el BP de ReadFile y ya tengo lo que ha leído en el buffer así que pongo un HBP on Access en el primer byte del buffer y doy a F9 y para aquí:

0041D5DB	. . 0F84 C0010000	je	0041D7A1	EFFind.0041D7A1
0041D5E1	. . 807D 00 00	cmp	byte ptr ss:[ebp], 0	
0041D5E5	. . 0F84 97010000	je	0041D782	EFFind.0041D782
0041D5EB	. . 56	push	esi	
0041D5EC	. . 57	push	edi	
0041D5ED	. . 8D49 00	lea	ecx, ds:[ecx]	
0041D5F0	> 8D4424 10	lea	eax, ss:[esp+10]	
0041D5F4	. . 50	push	eax	
0041D5F5	. . 45	inc	ebp	
0041D5F6	. . 55	push	ebp	
0041D5F7	. . 68 C88B5A00	push	5A8BC8	
0041D5FC	. . E8 5F24FFFF	call	0040FA60	EFFind.0040FA60
0041D601	. . 036C24 1C	add	ebp, ss:[esp+1C]	
0041D605	. . 0FB7C0	movzx	eax, ax	
0041D608	. . 0FB7C0	movzx	eax, ax	
0041D60B	. . 83C0 DC	add	eax, -24	

Registers (FPU)	
EAX	096C7D19
ECX	097EA7A1
EDX	00000001
EBX	00000001
ESP	0012DAB0
EBP	00C0B608 ASCII "EF Find distribution license
ESI	003B00C2
EDI	00C0B608 ASCII "EF Find distribution license
EIP	0041D5E5 EFFind.0041D5E5

Traceo un poco y cuando llego aquí veo que ha copiado el buffer en otro sitio así que pongo también un HBP al primer byte:

0041D750	>	85ED	test	ebp, ebp	
0041D752	..	74 0C	je	short 0041D760	EFFind.0041D760
0041D754	.	8A45 00	mov	al, ss:[ebp]	
0041D757	.	84C0	test	al, al	
0041D759	..	74 17	je	short 0041D772	EFFind.0041D772
0041D75B	.	45	inc	ebp	
0041D75C	.	3C 0A	cmp	al, OA	
0041D75E	.	75 F0	jnz	short 0041D750	EFFind.0041D750
0041D760	>	8B0D 54465D00	mov	ecx, ds:[5D4654]	Default case of s
0041D766	>	807D 00 00	cmp	byte ptr ss:[ebp], 0	
0041D76A	.	OF85 80FFFF	jnz	0041D5F0	EFFind.0041D5F0
0041D770	..	EB 06	jmp	short 0041D778	EFFind.0041D778
0041D772	>	8B0D 54465D00	mov	ecx, ds:[5D4654]	

Pongo un BP a la salida del bucle y doy a F9 pero no veo nada que me haga sospechar nada así que traceo hasta llegar al retn y salgo. Sigo traceando y mirando dentro de los calls hasta que llego aquí:

0042AC7C	.	56	push	esi	
0042AC7D	.	8BF8	mov	edi, eax	
0042AC7F	.	E8 3CC8FFFF	call	004274C0	EFFind.004274C0
0042AC84	.	57	push	edi	
0042AC85	.	E8 3629FFFF	call	0041D5C0	EFFind.0041D5C0
0042AC8A	.	A1 54465D00	mov	eax, ds:[5D4654]	
0042AC8F	.	8B48 08	mov	ecx, ds:[eax+8]	
0042AC92	.	6A 00	push	0	

En ese call que acaba de pasar ha creado una copia de nuestro buffer pero el ID ha sido sustituido por una concatenación de nuestro serial así que pongo un HBP on Execution en ese call y quito todos los demás y cuando para entro con F7, pongo un HBP on Write en el primer byte del ID encriptado y para aquí:

00412430	\$ 55	push	ebp	
00412431	. 0FB76C24 10	movzx	ebp, word ptr ss:[esp+10]	
00412436	. 56	push	esi	
00412437	. 8B7424 10	mov	esi, ss:[esp+10]	
0041243B	. 57	push	edi	
0041243C	. 8B7C24 10	mov	edi, ss:[esp+10]	
00412440	. 33C9	xor	ecx, ecx	
00412442	. B8 01000000	mov	eax, 1	
00412447	> 3BC5	cmp	eax, ebp	
00412449	.~ 7C 08	jl	short 00412453	EFFind.00412453
0041244B	. 8BC1	mov	eax, ecx	
0041244D	. 83E0 03	and	eax, 3	
00412450	. 83C0 02	add	eax, 2	
00412453	> 8A1438	mov	dl, ds:[eax+edi]	
00412456	. 881431	mov	ds:[ecx+esi], dl	
00412459	. 41	inc	ecx	
0041245A	. 40	inc	eax	
0041245B	. 83F9 39	cmp	ecx, 39	
0041245E	.^ 72 E7	jb	short 00412447	EFFind.00412447
00412460	. 5F	pop	edi	
00412461	. 5E	pop	esi	
00412462	. 5D	pop	ebp	
00412463	. C3	ret	n	

Ya tengo la función que reemplaza el ID original por la sucesión de seriales. Quito el HBP on Write y doy a Ctrl+F9 y salgo del retn con F7, voy a la línea de debajo del call y pongo un BP y cuando para sigo traceando hasta que llego aquí:

0042AD11	. 6548 0F	mov	ds:[eax+r], cl	
0042AD14	. 85FF	test	edi, edi	
0042AD16	.~ 74 11	je	short 0042AD29	EFFind.0042AD29
0042AD18	. 8B15 54465D00	mov	edx, ds:[5D4654]	
0042AD1E	. 89BA 78020000	mov	ds:[edx+278], edi	
0042AD24	. E8 7763FFFF	call	004210A0	EFFind.004210A0
0042AD29	> E8 6254FEFF	call	00410190	EFFind.00410190
0042AD2E	. 6A 02	push	2	
0042AD30	. 68 9F000000	push	9F	
0042AD35	. 68 11010000	push	111	
0042AD3A	. 56	push	esi	

Y si entro veo esto:

004210A0	\$. 56	push	esi	
004210A1	. 8B35 54465D00	mov	esi, ds:[5D4654]	
004210A7	. 81C6 D0020000	add	esi, 2D0	
004210AD	.~ 74 40	je	short 004210EF	EFFind.004210EF
004210AF	. 53	push	ebx	
004210B0	. 33C9	xor	ecx, ecx	
004210B2	. 57	push	edi	
004210B3	. 8D79 39	lea	edi, ds:[ecx+39]	
004210B6	> 8A0431	mov	al, ds:[ecx+esi]	
004210B9	. 3C 40	cmp	al, 40	
004210BB	.~ 73 16	jnb	short 004210D3	EFFind.004210D3
004210BD	. 8BD1	mov	edx, ecx	
004210BF	. 81E2 03000080	and	edx, 80000003	
004210C5	.~ 79 05	jns	short 004210CC	EFFind.004210CC
004210C7	. 4A	dec	edx	
004210C8	. 83CA FC	or	edx, FFFFFFFC	
004210CB	. 42	inc	edx	
004210CC	> 2AC2	sub	al, dl	
004210CE	. 880431	mov	ds:[ecx+esi], al	
004210D1	.~ EB 14	jmp	short 004210E7	EFFind.004210E7
004210D3	> 0FB6C0	movzx	eax, al	
004210D6	. 83E8 41	sub	eax, 41	
004210D9	. 99	cdq		
004210DA	. BB 0A000000	mov	ebx, 0A	
004210DF	. F7FB	idiv	ebx	
004210E1	. 80C2 30	add	dl, 30	

Ya estoy en la función que desencripta el ID así que paso el call con F8 y si miro mi ID lo veo desencriptado:

Address	Hex dump	ASCII
00C08558	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00C08568	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00C08578	31 31 32 35 33 30 31 37 38 33 36 33 35 32 32 38	1125301783635228
00C08588	32 38 37 2B 31 30 38 38 38 30 36 21 21 2F 32 34	287+1088806!!/24
00C08598	2B 22 23 23 26 32 36 35 31 32 30 37 33 38 33 35	+##426512073835
00C085A8	38 36 39 33 37 31 38 36 38 00 00 00 00 00 00	869371868

Entonces se me ocurrió algo, como en el caso del EF Checksum usaba para todo \_snprintf y, como uno de sus parámetros era el formato, busco el '%' en todas las strings y pongo un BP a todas las strings que tengan el formato "%XXlu" donde XX son números y estos son todos los que cumplen el formato que busco:

0040AA7B	. 68 DC725A00	push	5A72DC	; ASCII "%04lu"
0040C3F4	. 68 D4725A00	push	5A72D4	; ASCII "%03lu"
0040E9E9	. 68 04735A00	push	5A7304	; ASCII "%09lu"
0041036D	. 68 DC725A00	push	5A72DC	; ASCII "%04lu"
004103A2	. 68 0C735A00	push	5A730C	; ASCII "%02lu%02lu"
004107D3	. 68 CC725A00	push	5A72CC	; ASCII "%02lu"
004121F6	. 68 E4725A00	push	5A72E4	; ASCII "%05lu"
004140FF	. 68 D4725A00	push	5A72D4	; ASCII "%03lu"
0041DFFD	. 68 18735A00	push	5A7318	; ASCII "%06lu-%05lu-%08lu"
0041E071	. 68 18735A00	push	5A7318	; ASCII "%06lu-%05lu-%08lu"

Así que con todos esos BPs reinicio y se desactivarán todos, pongo un BP en CreateFileW, cuando pare para leer el archivo de licencia quito el Bp, pongo un BP en ReadFile y cuando pare quito el BP y me voy a la ventana de BPs y activo todos los BPs que había puesto y doy a F9 y para aquí:

0040AA79	. 5B	pop	ebx	
0040AA7A	> 56	push	esi	
0040AA7B	. 68 DC725A00	push	5A72DC	ASCII "%04lu"
0040AA80	. 8D5424 10	lea	edx, ss:[esp+10]	
0040AA84	. 6A 3F	push	3F	
0040AA86	. 52	push	edx	
0040AA87	. E8 47E60E00	call	004F90D3	EFFind.004F90D3
0040AA8C	. A1 54465D00	mov	eax, ds:[5D4654]	
0040AA91	. 8A4C24 18	mov	cl, ss:[esp+18]	
0040AA95	. 83C4 10	add	esp, 10	
0040AA98	. 5F	pop	edi	
0040AA99	. 5E	pop	esi	
0040AA9A	. 3A88 DD020000	cmp	cl, ds:[eax+2DD]	
0040AA9O	.~ 75 18	jnz	short 0040AABA	EFFind.0040AABA
0040AAA2	. 8A5424 01	mov	dl, ss:[esp+1]	
0040AAA6	. 3A90 DE020000	cmp	dl, ds:[eax+2DE]	
0040AAC	.~ 75 OC	jnz	short 0040AABA	EFFind.0040AABA
0040AAAE	. 8A4C24 02	mov	cl, ss:[esp+2]	
0040AAB2	. 3A88 DF020000	cmp	cl, ds:[eax+2DF]	
0040AAB8	.~ 74 10	je	short 0040AAC	EFFind.0040AAC
0040AABA	> 80B8 E1020000	cmp	byte ptr ds:[eax+2E1], 0	
0040AAC1	.~ 74 07	je	short 0040AAC	EFFind.0040AAC

Si voy hasta la call veo que retorna la cadena “0000” y compara con una parte llena de ceros con lo que está vacío aun y no va a coincidir así que doy a F9 y ahora caigo aquí:

00410361	. 50	push	eax	
00410362	. FF15 70415600	call	ds:[564170]	
00410368	. 8B4C24 58	mov	ecx, ss:[esp+58]	
0041036C	. 51	push	ecx	
0041036D	. 68 DC725A00	push	5A72DC	ASCII "%04lu"
00410372	. 8D5424 18	lea	edx, ss:[esp+18]	
00410376	. 6A 3F	push	3F	
00410378	. 52	push	edx	
00410379	. E8 558D0E00	call	004F90D3	EFFind.004F90D3
0041037E	. A1 54465D00	mov	eax, ds:[5D4654]	
00410383	. 8B88 E7020000	mov	ecx, ds:[eax+2E7]	
00410389	. 83C4 10	add	esp, 10	
0041038C	. 3B4C24 10	cmp	ecx, ss:[esp+10]	
00410390	.~ 74 53	je	short 004103E5	EFFind.004103E5
00410392	. 0FB75424 02	movzx	edx, word ptr ss:[esp+2]	
00410397	. 0FB70424	movzx	eax, word ptr ss:[esp]	
0041039B	. 52	push	edx	
0041039C	. 2D A8070000	sub	eax, 7A8	
004103A1	. 50	push	eax	
004103A2	. 68 0C735A00	push	5A730C	ASCII "%02lu%02lu"
004103A7	. 8D4C24 1C	lea	ecx, ss:[esp+1C]	
004103AB	. 6A 3F	push	3F	
004103AD	. 51	push	ecx	
004103AE	. E8 208D0E00	call	004F90D3	EFFind.004F90D3

Si paso el call veo que obtengo el “8806” que obtenía en EF Checksum:

0012BDEC	0012BDDC	ASCII "8806"
0012BDCC	0000003F	

Si miro la comparación veo que sigue comparando con un buffer vacío así que voy dando a F9 hasta que pare en uno y el buffer ya esté lleno y es aquí:

0041DFFD	. 52	<b>push</b>	<b>eax</b>	
0041E002	. 68 18735A00	<b>push</b>	5A7318	ASCII "%06lu-%05lu-%08lu"
0041E006	. 8D4424 1C	<b>lea</b>	eax, ss:[esp+1C]	
0041E008	. 6A 3F	<b>push</b>	3F	
0041E009	. 50	<b>push</b>	eax	
0041E009	. E8 C5B00D00	<b>call</b>	004F90D3	EFFind.004F90D3
0041E00E	. 83C4 18	<b>add</b>	esp, 18	
0041E011	. 8BCB	<b>mov</b>	ecx, ebx	
0041E013	. 8D4424 0C	<b>lea</b>	eax, ss:[esp+C]	
0041E017	> 8A10	<b>mov</b>	dl, ds:[eax]	
0041E019	. 3A11	<b>cmp</b>	dl, ds:[ecx]	
0041E01B	.~ 75 1A	<b>jnz</b>	short 0041E037	EFFind.0041E037
0041E01D	. 84D2	<b>test</b>	dl, dl	
0041E01F	.~ 74 12	<b>je</b>	short 0041E033	EFFind.0041E033
0041E021	. 8A50 01	<b>mov</b>	dl, ds:[eax+1]	
0041E024	. 3A51 01	<b>cmp</b>	dl, ds:[ecx+1]	
0041E027	.~ 75 0E	<b>jnz</b>	short 0041E037	EFFind.0041E037
0041E029	. 83C0 02	<b>add</b>	eax, 2	
0041E02C	. 83C1 02	<b>add</b>	ecx, 2	
0041E02F	. 84D2	<b>test</b>	dl, dl	
0041E031	.^ 75 E4	<b>jnz</b>	short 0041E017	EFFind.0041E017

Esa es la función que compara nuestro serial con la lista negra de seriales. Para obtener toda la lista negra de seriales voy a hacer lo mismo que hice con EF Checksum Manager:

```
0041E01B    /75 1A      jnz      short 0041E037 ; EFFind.0041E037
```

Por:

```
0041E01B    /EB 1A      jmp      short 0041E037 ; EFFind.0041E037
```

```
0041E03E    /74 7B      je       short 0041E0BB ; EFFind.0041E0BB
```

Por:

```
0041E03E    90          nop
0041E03F    90          nop
```

```
0041E04F    /0F84 72010000 je      0041E1C7 ; EFFind.0041E1C7
```

Por:

```
0041E04F    90          nop
0041E050    90          nop
0041E051    90          nop
0041E052    90          nop
0041E053    90          nop
0041E054    90          nop
```

```
0041E094    /0F85 BA000000 jnz     0041E154 ; EFFind.0041E154
```

Por:

```
0041E094    /E9 BB000000 jmp     0041E154 ; EFFind.0041E154
0041E099    |90          nop
```

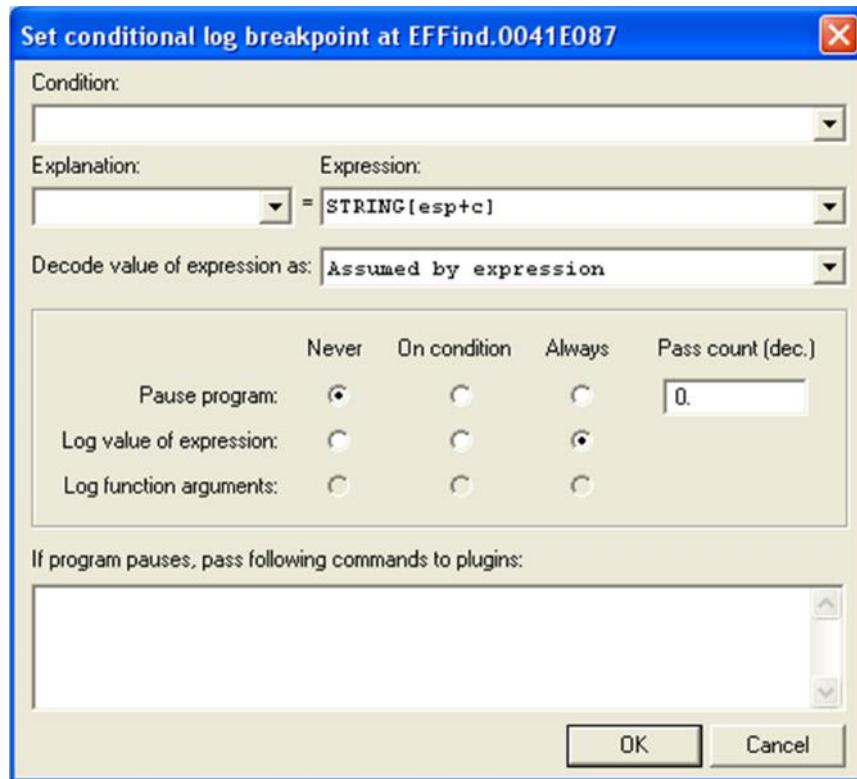
```
0041E15B    /74 24      je       short 0041E181 ; EFFind.0041E181
```

Por:

```
0041E15B    90          nop
0041E15C    90          nop
```

Una vez realizados todos esos cambios pongo dos BPs Conditional Log en las líneas:

```
0041E013 |. 8D4424 0C      |lea      eax, ss:[esp+C]
0041E087 |. 8D4424 0C      lea      eax, ss:[esp+C]
```



Quito el BP de las líneas:

```
0041DFFD |. 68 18735A00 |push      5A7318          ; ASCII "%06lu-%05lu-%08lu"
0041E071 |. 68 18735A00 push      5A7318          ; ASCII "%06lu-%05lu-%08lu"
```

Y pongo un BP a la salida de las comprobaciones de la lista negra que es aquí:

```
0041E169 |. 5F           pop      edi
```

Me voy a la ventana Log del Olly, clic derecho en su interior y le doy a “Clean window”, doy clic derecho y doy en “Log to file” y en la ventana que sale le doy nombre a mi archivo log. Doy a F9 y cuando pare en el BP me voy a la ventana Log, clic derecho y doy en “Close log file” y esto es lo que tendré en el log:

```
0041E013 COND: 323848-53000-90750642
0041E013 COND: 013990-54715-90721932
0041E013 COND: 533278-62213-90770938
0041E013 COND: 983528-11618-90710935
0041E013 COND: 273038-31919-90741931
0041E013 COND: 783721-02428-90790933
0041E013 COND: 443964-62112-90761932
```

```
0041E013 COND: 363140-72000-90850937
0041E013 COND: 323987-11024-90850939
0041E013 COND: 113097-11815-90731931
0041E013 COND: 593913-02237-90770941
0041E013 COND: 583923-12238-90770941
0041E013 COND: 043163-51712-90720939
0041E013 COND: 483924-22118-90761932
0041E013 COND: 623882-81314-90781933
0041E013 COND: 633474-21313-90780936
0041E013 COND: 663445-72310-90780936
0041E013 COND: 713495-12415-90790936
0041E013 COND: 433774-91143-98761932
0041E013 COND: 533870-46223-90771973
0041E013 COND: 833575-83513-90700935
0041E013 COND: 343865-14072-90751941
0041E013 COND: 683022-32318-90781931
0041E013 COND: 283626-02918-90740934
0041E013 COND: 053658-32711-90720934
0041E013 COND: 343061-12062-90750948
0041E013 COND: 583629-51218-90770934
0041E013 COND: 823688-73514-90700934
0041E013 COND: 883829-22518-90701933
0041E013 COND: 593518-12237-90770935
0041E013 COND: 403306-12176-90760935
0041E013 COND: 343662-62052-90751933
0041E013 COND: 453952-63121-90760931
0041E013 COND: 713599-62435-90790935
0041E013 COND: 423885-53114-90761933
0041E013 COND: 333569-01022-90750935
0041E013 COND: 063926-43738-90720931
0041E013 COND: 633271-52343-98780936
0041E013 COND: 563442-12210-90770936
0041E013 COND: 063548-03710-90720935
0041E013 COND: 343766-40032-90750913
0041E013 COND: 493818-25127-90761973
0041E013 COND: 313808-63006-90750642
0041E013 COND: 403900-50126-90761912
0041E087 COND: 483353-01141-90760935
0041E087 COND: 343360-44052-90750935
0041E087 COND: 393715-92067-90750931
0041E087 COND: 323580-83064-90750933
0041E087 COND: 423485-03174-90760944
0041E087 COND: 343863-61072-90751931
0041E087 COND: 393612-82087-90751933
0041E087 COND: 113798-63815-90730933
0041E087 COND: 233078-71913-90741931
0041E087 COND: 753056-12411-90791931
0041E087 COND: 493880-14134-90760942
0041E087 COND: 433374-20133-90760907
0041E087 COND: 573032-23239-90771931
0041E087 COND: 393958-01011-90750931
```

Ya tengo la lista negra controlada así que, como mi serial no está en ella, restauro todas las modificaciones. Para ello voy a la ventana Patches y me coloco en el primero y voy dando a la barra espaciadora hasta que se hayan desactivado todas.

Quito el BP y doy a F9 y vuelve a parar en la zona de la cadena “8806” así que voy con F8 hasta la siguiente línea:

```
00410383 |. 8B88 E7020000 mov      ecx, ds:[eax+2E7]
```

Miro la dirección a la que apunta [eax+2E7] en el panel inferior y hago clic derecho y selecciono “Follow address in Dump” y subo en el dump hasta encontrar el inicio del ID:

Address	Hex dump	ASCII
00C0856F	00 00 00 00 00 00 00 00 00 31 31 32 35 33 30 31	.....1125301
00C0857F	37 38 33 36 33 35 32 32 38 32 38 37 2B 31 30 38	783635228287+108
00C0858F	38 38 30 36 21 21 2F 32 34 2B 22 23 23 26 32 36	8806!!/24+"##&26
00C0859F	35 31 32 30 37 33 38 33 35 38 36 39 33 37 31 38	5120738358693718
00C085AF	36 38 00 00 00 00 00 00 00 00 00 00 82 E9 01	68.....,é

En mi caso lo tengo en el lugar correcto porque ya lo vi antes. Empieza en la posición 24º del ID. En caso de no coincidir se sustituye lo que haya por el “8806” y seguimos hasta la comparación y veremos que la pasamos porque son iguales. Quito el BP anterior y doy a F9 y para aquí:

004107AF	. 8D48 01	lea ecx, ds:[eax+1]	
004107B2	. 56	push esi	
004107B3	> OFB67404 06	movzx esi, byte ptr ss:[esp+eax+6]	
004107B8	. OFB65404 04	movzx edx, byte ptr ss:[esp+eax+4]	
004107BD	. 03D6	add edx, esi	
004107BF	. OFB67404 05	movzx esi, byte ptr ss:[esp+eax+5]	
004107C4	. 03D6	add edx, esi	
004107C6	. 83C0 03	add eax, 3	
004107C9	. 83F8 39	cmp eax, 39	
004107CC	. 8D4C91 15	lea ecx, ds:[ecx+edx*4+15]	
004107D0	.^ 7C E1	jl short 004107B3	EFFind.004107B3
004107D2	. 51	push ecx	
004107D3	. 68 CC725A00	push 5A72CC	ASCII "%02lu"
004107D8	. 8D4424 48	lea eax, ss:[esp+48]	
004107DC	. 6A 3F	push 3F	
004107DE	. 50	push eax	
004107DF	. E8 EF880E00	call 004F90D3	EFFind.004F90D3
004107E4	. A1 54465D00	mov eax, ds:[5D4654]	
004107E9	. 8A4C24 50	mov cl, ss:[esp+50]	
004107ED	. 83C4 10	add esp, 10	
004107F0	. 5E	pop esi	
004107F1	. 3A88 D0020000	cmp cl, ds:[eax+2D0]	
004107F7	.~ 74 06	je short 004107FF	EFFind.004107FF
004107F9	. FE80 0A010000	inc byte ptr ds:[eax+10A]	

Voy con F8 hasta la comparación y miro con que compara:

004107F0	. 5E	pop esi	
004107F1	. 3A88 D0020000	cmp cl, ds:[eax+2D0]	
004107F7	.~ 74 06	je short 004107FF	EFFind.004107FF
004107F9	. FE80 0A010000	inc byte ptr ds:[eax+10A]	
	ds:[00C08578]=31 ('1')		
	cl=31 ('1')		

Address	Hex dump	ASCII	
00C08578	31 31 32 35 33 30 31 37 38 33 36 33 35 32 32 38	1125301783635228	0012B550
00C08588	32 38 37 2B 31 30 38 38 38 30 36 21 21 2F 32 34	287+1088806!!/24	0012B560
00C08598	2B 22 23 23 26 32 36 35 31 32 30 37 33 38 33 35	+"##&26512073835	0012B564
00C085A8	38 36 39 33 37 31 38 36 38 00 00 00 00 00 00	869371868.....	0012B568
			0012B560

Compara solo el primer carácter de la cadena obtenida con el primero del ID pero no sé de dónde saca la cadena así que cambio el primer byte para que coincida, pongo un BP al inicio de la función y doy a F9 y para aquí:

0040AA75	. 3BCF	cmp	ecx, edi	
0040AA77	.^ 72 E7	jb	short 0040AA60	EFFind.0040AA60
0040AA79	. 5B	pop	ebx	
0040AA7A	> 56	push	esi	
0040AA7B	. 68 DC725A00	push	5A72DC	ASCII "%04lu"
0040AA80	. 8D5424 10	lea	edx, ss:[esp+10]	
0040AA84	. 6A 3F	push	3F	
0040AA86	. 52	push	edx	
0040AA87	. E8 47E60E00	call	004F90D3	EFFind.004F90D3
0040AA8C	. A1 54465D00	mov	eax, ds:[5D4654]	
0040AA91	. 8A4C24 18	mov	cl, ss:[esp+18]	
0040AA95	. 83C4 10	add	esp, 10	
0040AA98	. 5F	pop	edi	
0040AA99	. 5E	pop	esi	
0040AA9A	. 3A88 DD020000	cmp	cl, ds:[eax+2DD]	
0040AAA0	.~ 75 18	jnz	short 0040AABA	EFFind.0040AABA
0040AAA2	. 8A5424 01	mov	dl, ss:[esp+1]	
0040AAA6	. 3A90 DE020000	cmp	dl, ds:[eax+2DE]	
0040AAC	.~ 75 0C	jnz	short 0040AABA	EFFind.0040AABA
0040AAAE	. 8A4C24 02	mov	cl, ss:[esp+2]	
0040AAB2	. 3A88 DF020000	cmp	cl, ds:[eax+2DF]	
0040AAB8	.~ 74 10	je	short 0040AAC	EFFind.0040AAC
0040AABA	> 80B8 E1020000	cmp	byte ptr ds:[eax+2E1], 0	
0040AAC1	.~ 74 07	je	short 0040AAC	EFFind.0040AAC
0040AAC3	. C680 0E010000	mov	byte ptr ds:[eax+10E], 0	

Si voy con F8 hasta el call veo que en el buffer se encuentra mi serial así que esta es la zona que se encarga de hacer cálculos con mi serial y crear una cadena con el resultado para comparar con una parte concreta del ID. Traceo hasta la primera comparación y veo con que parte del ID compara:

0040AA9A	. 5E	pop	esi	
0040AAA0	. 3A88 DD020000	cmp	cl, ds:[eax+2DD]	
0040AAA2	.~ 75 18	jnz	short 0040AABA	EFFind.0040AABA
0040AAA6	. 8A5424 01	mov	dl, ss:[esp+1]	
0040AAC	. 3A90 DE020000	cmp	dl, ds:[eax+2DE]	
0040AAAE	.~ 75 0C	jnz	short 0040AABA	EFFind.0040AABA
0040AAAE	. 8A4C24 02	mov	cl, ss:[esp+2]	
0040AAB2	. 3A88 DF020000	cmp	cl, ds:[eax+2DF]	
0040AAB8	.~ 74 10	je	short 0040AAC	EFFind.0040AAC
0040AABA	> 80B8 E1020000	cmp	byte ptr ds:[eax+2E1], 0	
0040AAC1	.~ 74 07	je	short 0040AAC	EFFind.0040AAC
0040AAC3	C680 0E010000	mov	byte ptr ds:[eax+10E], 0	

ds:[00C08585]=32 ('2')	
cl=32 ('2')	
Address Hex dump ASCII	0012F14
00C08575 00 00 00 31 31 32 35 33 30 31 37 38 33 36 33 35 ...1125301783635 0012F14	
00C08585 32 32 38 32 38 37 2B 31 30 38 38 38 30 36 21 21 228287+1088806!! 0012F15	
00C08595 2F 32 34 2B 22 23 23 26 32 36 35 31 32 30 37 33 /24+"##&26512073 0012F15	
00C085A5 38 33 35 38 36 39 33 37 31 38 36 38 00 00 00 00 835869371868.... 0012F15	

Compara los 3 primeros caracteres de la cadena obtenida con los del ID empezando por el carácter 14º.

Cambio esos caracteres del ID para que coincidan, quito el BP y doy a F9 y ahora para en el BP que puse al inicio de la otra función y si voy traceando veo que hace una copia del ID desencriptado y sustituye el primer carácter por un espacio y en el bucle veo como hace las operaciones con esta cadena con lo que ya se con que trabaja para crear esa cadena de comprobación así que quito ese BP y el que sigue debajo del bucle y doy a F9 y para aquí:

004121E3	> 3BC8	cmp	ecx, eax		
004121E5	.~ 73 0E	jnb	short 004121F5		
004121E7	. 0FB6740D 00	movzx	esi, byte ptr ss:[ebp+ecx]		
004121EC	. 03F1	add	esi, ecx		
004121EE	. 8D5456 FF	lea	edx, ds:[esi+edx*2-1]		
004121F2	. 41	inc	ecx		
004121F3	.^ EB EE	jmp	short 004121E3		EFFind.004121E3
004121F5	> 52	push	edx		
004121F6	. 68 E4725A00	push	5A72E4		ASCII "%05lu"
004121FB	. 8D55 00	lea	edx, ss:[ebp]		
004121FE	. 6A 7F	push	7F		
00412200	. 52	push	edx		
00412201	. E8 CD6E0E000	call	004F90D3		EFFind.004F90D3
00412206	. A1 54465D00	mov	eax, ds:[5D4654]		
0041220B	. 8A4D 01	mov	cl, ss:[ebp+1]		
0041220E	. 83C4 10	add	esp, 10		
00412211	. 3A88 E0020000	cmp	cl, ds:[eax+2E0]		
00412217	.~ 75 16	jnz	short 0041222F		EFFind.0041222F
00412219	. 8A55 02	mov	dl, ss:[ebp+2]		
0041221C	. 3A90 E1020000	cmp	dl, ds:[eax+2E1]		
00412222	.~ 75 0B	jnz	short 0041222F		EFFind.0041222F
00412224	. 8A4D 03	mov	cl, ss:[ebp+3]		
00412227	. 3A88 E2020000	cmp	cl, ds:[eax+2E2]		
0041222D	.~ 74 19	je	short 00412248		EFFind.00412248
0041222F	> 3898 E2020000	cmp	ds:[eax+2E2], bl		

Si voy hasta el call veo que en el buffer está el nombre que tengo en el archivo de licencia con lo que es lo que usó para las operaciones de más arriba. Voy hasta la primera comparación y veo que va a comparar los caracteres 2º, 3º y 4º de la cadena obtenida con los caracteres 17º, 18º y 19º del ID:

0041220E	. 83C4 10	add	esp, 10		
00412211	. 3A88 E0020000	cmp	cl, ds:[eax+2E0]		
00412217	.~ 75 16	jnz	short 0041222F		EFFind.0041222F
00412219	. 8A55 02	mov	dl, ss:[ebp+2]		
0041221C	. 3A90 E1020000	cmp	dl, ds:[eax+2E1]		
00412222	.~ 75 0B	jnz	short 0041222F		EFFind.0041222F
00412224	. 8A4D 03	mov	cl, ss:[ebp+3]		
00412227	. 3A88 E2020000	cmp	cl, ds:[eax+2E2]		
0041222D	.~ 74 19	je	short 00412248		EFFind.00412248
0041222F	> 3898 E2020000	cmp	ds:[eax+2E2], bl		

ds:[00C08588]=32 ('2')  
cl=32 ('2')

Address	Hex dump	ASCII	0012F2
00C08578	31 31 32 35 33 30 31 37 38 33 36 33 35 32 32 38	1125301783635228	0012F2
00C08588	32 38 37 2B 31 30 38 38 38 30 36 21 21 2F 32 34	287+1088806!!/24	0012F2
00C08598	2B 22 23 23 26 32 36 35 31 32 30 37 33 38 33 35	+">#&26512073835	0012F2
00C085A8	38 36 39 33 37 31 38 36 38 00 00 00 00 00 00	869371868.....	0012F2

Realizo los cambios en el ID para que coincidan, quito el BP y doy a F9 y ya no vuelve a parar.

Si voy al menú ayuda veo que me sigue apareciendo el “Unlock” y que en “Acerca de” sigo sin estar registrado con lo que algo se me escapa así que, como en el EF Checksum Manager vi que colocaba el serial sin guiones en el ID, reinicio el Olly y realizo el proceso de poner el BP en CreateFileW y ReadFile para llegar justo al momento donde leía el archivo de licencia, me voy a ese buffer en el Dump y pongo un HBP on Access en el primer carácter del serial y doy a F9 y para aquí:

```
0041D62F . 8BF5    mov    esi, ebp
0041D631 . F3:A5   rep    movs dword ptr es:[edi], dword
0041D633 . 8B0D 54465D00 mov    ecx, ds:[5D4654]
0041D639 . 33C0   xor    eax, eax
0041D63B . EB 03   jmp    short 0041D640
0041D63D . 8D49 00   lea    ecx, ds:[ecx]
0041D640 > 0FB7D0  movzx edx, ax
0041D643 . 807C0A 68 0D  cmp    byte ptr ds:[edx+ecx+68], 0D
0041D648 . 74 0D   je     short 0041D657
0041D64A . 40      inc    eax
0041D64B . 66:83F8 51  cmp    ax, 51
0041D64F . 72 EF   jb    short 0041D640
0041D651 . 43      inc    ebx
0041D652 . E9 0F010000 jmp    0041D766
0041D657 > 0FB7C0  movzx eax, ax
0041D65A . C64408 68 00  mov    byte ptr ds:[eax+ecx+68], 0
0041D65F . 8B0D 54465D00 mov    ecx, ds:[5D4654]
0041D665 . 43      inc    ebx
0041D666 . E9 FB000000 jmp    0041D766

ecx=00000008 (decimal 8.)
ds:[esi]=[00C0B6BD]=37303231
es:[edi]=[00C08340]=00000000
```

Me voy en el Dump a la dirección apuntada por EDI, doy a F8 y vuelvo a poner un HBP on Access en el primer carácter del serial y doy a F9 y ahora para aquí:

```
0041D67C . 8BF5    mov    esi, ebp
0041D67E . F3:A5   rep    movs dword ptr es:[edi], dword
0041D680 . 8B0D 54465D00 mov    ecx, ds:[5D4654]
0041D686 . 33C0   xor    eax, eax
0041D688 . EB 06   jmp    short 0041D690
0041D68A . 8D9B 00000000 lea    ebx, ds:[ebx]
0041D690 > 0FB7D0  movzx edx, ax
0041D693 . 80BC0A B90000 cmp    byte ptr ds:[edx+ecx+B9], 0D
0041D69B . 74 0D   je     short 0041D6AA
0041D69D . 40      inc    eax
0041D69E . 66:83F8 51  cmp    ax, 51
0041D6A2 . 72 EC   jb    short 0041D690
0041D6A4 . 43      inc    ebx

ecx=0000000D (decimal 13.)
ds:[esi]=[00C0B6BC]=30323135
es:[edi]=[00C0837D]=00000000
```

Vuelvo a ir en el Dump a la dirección apuntada por EDI, paso la instrucción con F8, vuelvo a poner otro HBP en el primer carácter del serial, doy a F9 y ahora para aquí:

0041D6D2	. 8BF5	mov	esi, ebp	
0041D6D4	. F3:A5	rep	movs dword ptr es:[edi], dword	
0041D6D6	. 66:A5	movs	word ptr es:[edi], word ptr ds:	
0041D6D8	. A4	movs	byte ptr es:[edi], byte ptr ds:	
0041D6D9	. 8B0D 54465D00	mov	ecx, ds:[5D4654]	
0041D6DF	. 33C0	xor	eax, eax	
0041D6E1	> 0FB7D0	movzx	edx, ax	
0041D6E4	. 80BC0A 0A0100	cmp	byte ptr ds:[edx+ecx+10A], 0D	
0041D6EC	.~ 74 0A	je	short 0041D6F8	EFFind.0041D6F8
0041D6EE	. 40	inc	eax	
0041D6EF	. 66:83F8 18	cmp	ax, 18	
0041D6F3	.^ 72 EC	jb	short 0041D6E1	EFFind.0041D6E1
0041D6F5	. 43	inc	ebx	
0041D6F6	.~ EB 6E	jmp	short 0041D766	EFFind.0041D766
0041D6F8	> 0FB7C0	movzx	eax, ax	

Vuelvo a ir a la dirección apuntada por EDI, paso la instrucción con F8, pongo otro HBP on Access en el primer carácter del serial y al dar a F9 paro dentro del bucle que hay más abajo y que lo que hace es medir el largo del serial. Vuelvo a dar a F9 y ahora para aquí:

0041D74A	. 8D9B 00000000	lea	ebx, ds:[ebx]	
0041D750	> 85ED	test	ebp, ebp	
0041D752	.~ 74 0C	je	short 0041D760	EFFind.0041D760
0041D754	. 8A45 00	mov	al, ss:[ebp]	
0041D757	. 84C0	test	al, al	
0041D759	.~ 74 17	je	short 0041D772	EFFind.0041D772
0041D75B	. 45	inc	ebp	
0041D75C	. 3C 0A	cmp	al, 0A	
0041D75E	.^ 75 F0	jnz	short 0041D750	EFFind.0041D750
0041D760	> 8B0D 54465D00	mov	ecx, ds:[5D4654]	Default case of :
0041D766	> 807D 00 00	cmp	byte ptr ss:[ebp], 0	
0041D76A	.^ 0F85 80FFFF	jnz	0041D5F0	EFFind.0041D5F0
0041D770	.~ EB 06	jmp	short 0041D778	EFFind.0041D778
0041D772	> 8B0D 54465D00	mov	ecx, ds:[5D4654]	
0041D778	> 5F	pop	edi	

En principio no es importante ya que lo que nos interesa ver es que compare con un guion o que lo copie a otro sitio y no es el caso así que doy a F9 y ahora para aquí:

00401AD7	.	33D2	xor	edx, edx	
00401AD9	.	380E	cmp	ds:[esi], cl	
00401ADB	..	74 22	je	short 00401AFF	EFFind.00401AFF
00401ADD	.	53	push	ebx	
00401ADE	.	57	push	edi	
00401ADF	.	8B7C24 14	mov	edi, ss:[esp+14]	
00401AE3	.	8BC6	mov	eax, esi	
00401AE5	>	8A00	mov	al, ds:[eax]	
00401AE7	.	3C 2D	cmp	al, 2D	
00401AE9	..	74 07	je	short 00401AF2	EFFind.00401AF2
00401AEB	.	0FB7DA	movzx	ebx, dx	
00401AEE	.	88043B	mov	ds:[ebx+edi], al	
00401AF1	.	42	inc	edx	
00401AF2	>	41	inc	ecx	
00401AF3	.	0FB7C1	movzx	eax, cx	
00401AF6	.	03C6	add	eax, esi	
00401AF8	.	8038 00	cmp	byte ptr ds:[eax], 0	
00401AFB	^	75 E8	jnz	short 00401AE5	EFFind.00401AE5

Vuelvo a dar a F9 y ahora cae dentro del bucle que hay justo debajo:

00401AD7	.	53D2	xor	eax, eax	
00401AD9	.	380E	cmp	ds:[esi], cl	
00401ADB	▼	74 22	je	short 00401AFF	EFFind.00401AFF
00401ADD	.	53	push	ebx	
00401ADE	.	57	push	edi	
00401ADF	.	8B7C24 14	mov	edi, ss:[esp+14]	
00401AE3	.	8BC6	mov	eax, esi	
00401AE5	>	8A00	mov	al, ds:[eax]	
00401AE7	.	3C 2D	cmp	al, 2D	
00401AE9	▼	74 07	je	short 00401AF2	EFFind.00401AF2
00401AEB	.	0FB7DA	movzx	ebx, dx	
00401AEE	.	88043B	mov	ds:[ebx+edi], al	
00401AF1	.	42	inc	edx	
00401AF2	>	41	inc	ecx	
00401AF3	.	0FB7C1	movzx	eax, cx	
00401AF6	.	03C6	add	eax, esi	
00401AF8	.	8038 00	cmp	byte ptr ds:[eax], 0	
00401AFB	^	75 E8	jnz	short 00401AE5	EFFind.00401AE5
00401AFD	.	5F	pop	edi	

Este sí que es interesante ya que lo que hace es copiar el serial sin guiones en otro sitio así que me voy traceando hasta la línea:

00401AEE |. 88043B | mov ds:[ebx+edi], al

Voy en el Dump a la dirección donde se guardará el serial sin guiones, pongo un BP al final del bucle, doy a F9 y cuando pare quite el BP y ya tengo el serial sin guiones:

Address	Hex dump	ASCII
0012DABC	36 35 31 32 30 37 33 38 33 35 38 36 39 33 37 31	6512073835869371
0012DACC	38 36 38 00 26 E5 3A 77 53 8E 3A 7E 62 AF 3A 7E	868.ä:wsž:~b-:~
0012DAE0	62 2B B0 5A 3C 42 20 28 B6 C0 C0 44 84 10 20 FF17ž B FFž BFFž	

En principio ya no necesito ninguno de los HBPs puestos así que los quito todos. Voy con F8 hasta salir de la función y a las pocas líneas veo esto:

0041D735	. E8 9643FEFF	call	00401AD0	EFFind.00401AD0
0041D73A	. 6A 13	push	13	
0041D73C	. 8D4424 20	lea	eax, ss:[esp+20]	
0041D740	. 55	push	ebp	
0041D741	. 50	push	eax	
0041D742	. E8 E94CFFFF	call	00412430	EFFind.00412430
0041D747	. 83C4 14	add	esp, 14	
0041D74A	. 8D9B 00000000	lea	ebx, ds:[ebx]	
0041D750	> 85ED	test	ebp, ebp	
0041D752	.~ 74 0C	je	short 0041D760	EFFind.0041D760
0041D754	. 8A45 00	mov	al, ss:[ebp]	
0041D757	. 84C0	test	al, al	
0041D759	.~ 74 17	je	short 0041D772	EFFind.0041D772
0041D75B	. 45	inc	ebp	
0041D75C	. 3C 0A	cmp	al, 0A	
0041D75E	.^ 75 F0	jnz	short 0041D750	EFFind.0041D750
0041D760	> 8B0D 54465D00	mov	ecx, ds:[5D465D41]	Default case of

Entro a la función para ver que veo:

00412440	. 33C9	xor	ecx, ecx	
00412442	. B8 01000000	mov	eax, 1	
00412447	> ^3BC5	cmp	eax, ebp	
00412449	.~ 7C 08	jl	short 00412453	EFFind.00412453
0041244B	. 8BC1	mov	eax, ecx	
0041244D	. 83E0 03	and	eax, 3	
00412450	. 83C0 02	add	eax, 2	
00412453	> 8A1438	mov	dl, ds:[eax+edi]	
00412456	. 881431	mov	ds:[ecx+esi], dl	
00412459	. 41	inc	ecx	
0041245A	. 40	inc	eax	
0041245B	. 83F9 39	cmp	ecx, 39	
0041245E	.^ 72 E7	jb	short 00412447	EFFind.00412447
00412460	. 5F	pop	edi	
00412461	. 5E	pop	esi	
00412462	. 5D	pop	ebp	
00412463	. C3	ret		
00412464	cc			

Voy en el Dump a la dirección apuntada por ESI y veo que apunta justo al primer carácter del ID encriptado así que pongo un BP a la salida del bucle y veo esto en el Dump:

Address	Hex dump	ASCII
00C0B6CE	36 38 0D 0A 52 65 67 2E 49 44 2E 20 20 20 3A 20	68..Reg.ID. :
00C0B6DE	35 31 32 30 37 33 38 33 35 38 36 39 33 37 31 38	5120738358693718
00C0B6EE	36 38 30 37 33 38 33 35 38 36 39 33 37 31 38 36	6807383586937186
00C0B6FE	38 32 30 37 33 38 33 35 38 36 39 33 37 31 38 36	8207383586937186
00C0B70E	38 32 30 37 33 38 33 35 38 0D 0A 0D 0A 57 61 72	820738358....War

O sea que en este bucle está el algoritmo para llenar el ID con sucesiones del serial sin guiones y que será el buffer que se use para los cálculos de la creación de la cadena a partir del buffer del archivo de licencia. No es lo que estaba buscando pero es necesario para crear un keygen.

Salgo de la función y traceo pero no doy con el sitio donde hace la comparación así que se me ocurrió otro modo. Ya sé la función donde desencripta mi ID, también muchas funciones donde se comparan partes de mi ID, así

que reinicio y pongo un HBP on Execution en el inicio de la función que desencripta mi ID y pongo un BP Memory on Access en todo el ID desencriptado y voy pasando todos los lugares que ya conozco hasta que llego aquí:

004F95B7	. 8D7B FC	lea edi, ds:[ebx-4]	
004F95BA	. 85FF	test edi, edi	
004F95BC	.~ 76 6E	jbe short 004F962C	EFFind.004F962C
004F95BE	. 8B4D 0C	mov ecx, [arg.2]	Default case of sw
004F95C1	. 8B45 08	mov eax, [arg.1]	
004F95C4	> 8A10	mov dl, ds:[eax]	
004F95C6	. 83C0 04	add eax, 4	
004F95C9	. 83C1 04	add ecx, 4	
004F95CC	. 84D2	test dl, dl	
004F95CE	.~ 74 52	je short 004F9622	EFFind.004F9622
004F95D0	. 3A51 FC	cmp dl, ds:[ecx-4]	
004F95D3	.~ 75 4D	jnz short 004F9622	EFFind.004F9622
004F95D5	. 8A50 FD	mov dl, ds:[eax-3]	
004F95D8	. 84D2	test dl, dl	
004F95DA	.~ 74 3C	je short 004F9618	EFFind.004F9618
004F95DC	. 3A51 FD	cmp dl, ds:[ecx-3]	
004F95DF	.~ 75 37	jnz short 004F9618	EFFind.004F9618
004F95E1	. 8A50 FE	mov dl, ds:[eax-2]	
004F95E4	. 84D2	test dl, dl	
004F95E6	.~ 74 26	je short 004F960E	EFFind.004F960E
004F95E8	. 3A51 FE	cmp dl, ds:[ecx-2]	
004F95EB	.~ 75 21	jnz short 004F960E	EFFind.004F960E
004F95ED	. 8A50 FF	mov dl, ds:[eax-11]	

Si miro el carácter que va a mover es el primero de la selección y compara todos desde ese hasta el final:

Address	Hex dump	ASCII
00C0856E	00 00 00 00 00 00 00 00 00 00 31 31 32 35 33 30	.....112530
00C0857E	31 37 38 33 36 33 35 32 32 38 32 38 37 2B 31 30	1783635228287+10
00C0858E	38 38 38 30 36 21 21 2F 32 34 2B 22 23 23 26 32	88806!!/24+"##&2
00C0859E	36 35 31 32 30 37 33 38 33 35 38 36 39 33 37 31	6512073835869371
00C085AE	38 36 38 00 00 00 00 00 00 00 00 00 00 00 82 E9	868.....,é

Al salir de la función veo esto:

0040F51F	. 6A 13	push 13	
0040F521	. 8D5424 10	lea edx, ss:[esp+10]	
0040F525	. 52	push edx	
0040F526	. 81C6 F602000	add esi, 2F6	
0040F52C	. 56	push esi	
0040F52D	. E8 66A00E00	call 004F9598	EFFind.004F9598
0040F532	. 83C4 14	add esp, 14	
0040F535	. 85C0	test eax, eax	
0040F537	.~ 74 1F	je short 0040F558	EFFind.0040F558
0040F539	. 8B0D 54465D0	mov ecx, ds:[5D4654]	
0040F53F	. 33C0	xor eax, eax	
0040F541	. 81C1 0A010000	add ecx, 10A	

Uno de los parámetros es 0x13 que es justo el tamaño del serial sin guiones, o sea que es algo como el strncmp de C. Pongo un HBP on Execution en el call de arriba, reinicio Olly, doy a F9 y cuando pare en la función

desencriptadora la paso y pongo el BP Memory on Access en todo el ID desencriptado y voy pasando todas las comprobaciones hasta que llego al HBP que acabo de poner y veo lo que tengo en el Stack al parar ahí:

0012C684	00C0859E	ASCII "6512073835869371868"
0012C688	0012C69C	ASCII "6512073835869371868"
0012C68C	00000013	

Y si miro en el Dump adonde apunta el primer parametro:

Address	Hex dump	ASCII
00C0856E	00 00 00 00 00 00 00 00 00 00 00 00 31 31 32 35 33 30	.....112530
00C0857E	31 37 38 33 36 33 35 32 32 38 32 38 37 2B 31 30	1783635228287+10
00C0858E	38 38 38 30 36 21 21 2F 32 34 2B 22 23 23 26 32	88806!!/24+"##&2
00C0859E	36 35 31 32 30 37 33 38 33 35 38 36 39 33 37 31	6512073835869371
00C085AE	38 36 38 00 00 00 00 00 00 00 00 00 00 00 82 E9 868.....,é	

O sea que esa funcion es algo asi como el strncmp de C y lo que hace es comparar la cadena del serial sin guiones con lo que hay en el ID desencriptado desde la posicion 39º.

Si reinicio y, una vez parado en el HBP on Execution de la función donde se desencripta el ID, pongo todos los BPs para parar en todas las comprobaciones que ya he visto y voy sustituyendo todas las partes del ID desencriptado por los caracteres correctos cuando vaya parando en esas zonas, incluida esta última, doy a F9 y hago clic en Ayuda en el EF Find:



Ya no aparece el “Unlock” y si doy en “Acerca de”:



Doy a OK y me pongo a toquetear los botones por si hay alguna sorpresa y al dar al botón para aquí:

0040E9E2	.	50	push	eax	
0040E9E3	.	E8 9892FFFF	call	00407C80	EFFind.00407C80
0040E9E8	.	50	push	eax	
0040E9E9	.	68 04735A00	push	5A7304	ASCII "%09lu"
0040E9EE	.	8D4C24 0C	lea	ecx, ss:[esp+C]	
0040E9F2	.	6A 3F	push	3F	
0040E9F4	.	51	push	ecx	
0040E9F5	.	E8 D9A60E00	call	004F90D3	EFFind.004F90D3
0040E9FA	.	A1 54465D00	mov	eax, ds:[5D4654]	
0040E9FF	.	6A 09	push	9	
0040EA01	.	8D5424 18	lea	edx, ss:[esp+18]	
0040EA05	.	52	push	edx	
0040EA06	.	05 D4020000	add	eax, 2D4	
0040EA0B	.	50	push	eax	
0040EA0C	.	E8 87AB0E00	call	004F9598	EFFind.004F9598
0040EA11	.	83C4 20	add	esp, 20	
0040EA14	.	85C0	test	eax, eax	
0040EA16	..	74 0B	je	short 0040EA23	EFFind.0040EA23
0040EA18	.	A1 54465D00	mov	eax, ds:[5D4654]	

Voy traceando hasta que llego aquí:

0040E9FF	. 6A 09	push 9	
0040EA01	. 8D5424 18	lea edx, ss:[esp+18]	
0040EA05	. 52	push edx	
0040EA06	. 05 D4020000	add eax, 2D4	
0040EA0B	. 50	push eax	
0040EA0C	. E8 87AB0E00	call 004F9598	EFFind.004F9598
0040EA11	. 83C4 20	add esp, 20	
0040EA14	. 85C0	test eax, eax	
0040EA16	.~ 74 0B	je short 0040EA23	EFFind.0040EA23
0040EA18	. A1 54465D00	mov eax, ds:[5D4654]	
0040EA1D	. FE80 15010000	inc byte ptr ds:[eax+115]	
0040EA23	> 8B4C24 40	mov ecx, ss:[esp+40]	

Y si miro en el Stack:

011CB254	00C0857C	ASCII "301783635228287+1088806!!/24+"##&265120
011CB258	011CB274	ASCII "301783635"
011CB25C	00000009	
011CB260	011CB274	ASCII "301783635"
011CB264	0000003F	
011CB268	005A7304	ASCII "%09lu"
011CB26C	11FCDA53	

Ese call es como el que vimos que usaba para comparar con el serial sin guiones y que es como el strncmp de C con lo que va a comparar los primeros nueve caracteres de ambas cadenas. En mi caso coinciden porque ya tengo un archivo de licencia bueno jejeje. Si doy clic derecho en el primero del Stack y elijo “Follow in Dump” veo esto:

Address	Hex dump	ASCII
00C0856C	00 00 00 00 00 00 00 00 00 00 00 00 31 31 32 35	.....1125
00C0857C	33 30 31 37 38 33 36 33 35 32 32 38 32 38 37 2B	301783635228287+
00C0858C	31 30 38 38 38 30 36 21 21 2F 32 34 2B 22 23 23	1088806!!/24+"##
00C0859C	26 32 36 35 31 32 30 37 33 38 33 35 38 36 39 33	&265120738358693
00C085AC	37 31 38 36 38 00 00 00 00 00 00 00 00 00 00 00	71868.....

O sea que empieza a partir del 5º carácter del ID así que si no son iguales hay que asegurarse de que lo sean modificándolo en el Dump. Pongo un BP al inicio de la función para ver de dónde saca la cadena con la que compara, doy a F9 y ahora para aquí:

0040C3F1	. 5F	pop edi		
0040C3F2	. 5E	pop esi		
0040C3F3	> 51	push ecx		
0040C3F4	. 68 D4725A00	push 5A72D4	ASCII "%03lu"	
0040C3F9	. 8D4424 08	lea eax, ss:[esp+8]		
0040C3FD	. 68 FF030000	push 3FF		
0040C402	. 50	push eax		
0040C403	. E8 CBCC0E00	call 004F90D3	EFFind.004F90D3	
0040C408	. A1 54465D00	mov eax, ds:[5D4654]		
0040C40D	. 8A4C24 10	mov cl, ss:[esp+10]		
0040C411	. 83C4 10	add esp, 10		
0040C414	. 3A88 E4020000	cmp cl, ds:[eax+2E4]		
0040C41A	.~ 75 18	jnz short 0040C434	EFFind.0040C434	
0040C41C	. 8A5424 01	mov dl, ss:[esp+1]		
0040C420	. 3A90 E5020000	cmp dl, ds:[eax+2E5]		
0040C426	.~ 75 0C	jnz short 0040C434	EFFind.0040C434	
0040C428	. 8A4C24 02	mov cl, ss:[esp+2]		
0040C42C	. 3A88 E6020000	cmp cl, ds:[eax+2E6]		
0040C432	.~ 74 07	je short 0040C43B	EFFind.0040C43B	
0040C434	> C680_0C01000	mov byte ptr ds:[eax+10C1], 0		

Voy hasta el call y al mirar en el Stack veo esto:

011CAEA0	011CAEB0	ASCII "Agum1651207-38358-69371868CracksLatinos"
011CAEA4	0000003FF	
011CAEA8	005A72D4	ASCII "%03lu"
011CAEAC	00002A63	

Parece ser que usa una concatenación “Nombre+Serial+Compañía” para realizar las operaciones y obtener la cadena con la que comparar. Si llego a la primera comparación veo que compara los 3 primeros caracteres de la cadena obtenida con los caracteres 21º, 22º y 23º del ID:

Address	Hex dump	ASCII
00C0856C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 31 31 32 35	.....1125
00C0857C	33 30 31 37 38 33 36 33 35 32 32 38 32 38 37 2B	301783635228287+
00C0858C	31 30 38 38 38 30 36 21 21 2F 32 34 2B 22 23 23	1088806!!/24+##
00C0859C	26 32 36 35 31 32 30 37 33 38 33 35 38 36 39 33	&265120738358693
00C085AC	37 31 38 36 38 00 00 00 00 00 00 00 00 00 00 00 00	71868.....

Si no coinciden los cambiamos en el Dump por los correctos para que pasemos correctamente las comparaciones, quitamos el BP y damos a F9 y ahora para aquí:

004140F7	. 83C3 20	add ebx, 20	le suma 0x20
004140FA	> 03F7	add esi, edi	suma esi y edi
004140FC	. 03F3	add esi, ebx	le suma ebx a es
004140FE	. 56	push esi	el resultado
004140FF	. 68 D4725A00	push 5A72D4	ASCII "%03lu"
00414104	. 8D5424 14	lea edx, ss:[esp+14]	
00414108	. 6A 7F	push 7F	
0041410A	. 52	push edx	
0041410B	. E8 C34F0E00	call 004F90D3	EFFind.004F90D3
00414110	. A1 54465D00	mov eax, ds:[5D4654]	
00414115	. 8A4C24 1C	mov cl, ss:[esp+1C]	
00414119	. 83C4 10	add esp, 10	
0041411C	. 5F	pop edi	
0041411D	. SE	pop esi	
0041411E	. 5B	pop ebx	
0041411F	. 3A88 D1020000	cmp cl, ds:[eax+2D1]	
00414125	..~ 75 18	jnz short 0041413F	EFFind.0041413F
00414127	. 8A5424 01	mov dl, ss:[esp+1]	
0041412B	. 3A90 D2020000	cmp dl, ds:[eax+2D2]	
00414131	..~ 75 0C	jnz short 0041413F	EFFind.0041413F
00414133	. 8A4C24 02	mov cl, ss:[esp+2]	
00414137	. 3A88 D3020000	cmp cl, ds:[eax+2D3]	
0041413D	..~ 74 06	je short 00414145	EFFind.00414145
0041413F	> FE80 19010000	inc byte ptr ds:[eax+119]	
00414145	> 8B8C24 800000	mov ecx, ss:[esp+80]	

Si traceo hasta el call veo esto en el Stack:

011C9D44	011C9D60	ASCII "CracksLatinoS"
011C9D48	0000007F	
011C9D4C	005A72D4	ASCII "%03lu"
011C9D50	0000004E3	

O sea que va a usar el nombre de compañía para las operaciones de obtención de la cadena así que voy hasta la primera comparación y miro adonde apunta en el Dump y veo que comparará los 3 primeros caracteres de la cadena obtenida con los caracteres 2º, 3º y 4º del ID:

0041411E	. 5B	pop	ebx	
0041411F	. 3A88 D102000	cmp	cl, ds:[eax+2D1]	
00414125	.~ 75 18	jnz	short 0041413F	EFFind.0041413F
00414127	. 8A5424 01	mov	dl, ss:[esp+1]	
0041412B	. 3A90 D202000	cmp	dl, ds:[eax+2D2]	
00414131	.~ 75 0C	jnz	short 0041413F	EFFind.0041413F
00414133	. 8A4C24 02	mov	cl, ss:[esp+2]	
00414137	. 3A88 D302000	cmp	cl, ds:[eax+2D3]	
0041413D	.~ 74 06	je	short 00414145	EFFind.00414145
0041413E	> 33C9 0000000000000000			
ds:[00C08579]=31 ('1') cl=31 ('1')				
Address	Hex dump	ASCII		
00C08569	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 31	.....1		011C9D
00C08579	31 32 35 33 30 31 37 38 33 36 33 35 32 32 38 32	1253017836352282		011C9D
00C08589	38 37 2B 31 30 38 38 38 30 36 21 21 2F 32 34 2B	87+1088806!!/24+		011C9D
00C08599	22 23 23 26 32 36 35 31 32 30 37 33 38 33 35 38	"##&265120738358		011C9D
00C085A9	36 39 33 37 31 38 36 38 00 00 00 00 00 00 00 00	69371868.....		011C9D

Hago los cambios en el Dump para que coincidan, quito el BP y doy a F9 y ya no para más.

Vuelvo a dar al botón y para en el BP que puse al inicio de la función y voy traceando hasta el primer call y al entrar y tracear un poco veo esto:

00407C80	\$ 51	push	ecx	
00407C81	. 53	push	ebx	
00407C82	. 8B5C24 0C	mov	ebx, ss:[esp+C]	
00407C86	. 85DB	test	ebx, ebx	
00407C88	.~ 74 5F	je	short 00407CE9	EFFind.00407CE9
00407C8A	. 8A0B	mov	cl, ds:[ebx]	
00407C8C	. 33C0	xor	eax, eax	
00407C8E	. 84C9	test	cl, cl	
00407C90	.~ 74 0E	je	short 00407CA0	EFFind.00407CA0
00407C92	> 80F9 1A	cmp	cl, 1A	
00407C95	.~ 74 09	je	short 00407CA0	EFFind.00407CA0
00407C97	. 8A4C18 01	mov	cl, ds:[eax+ebx+1]	
00407C9B	. 40	inc	eax	
00407C9C	. 84C9	test	cl, cl	
00407C9E	.~ 75 F2	jnz	short 00407C92	EFFind.00407C92
00407CA0	> 56	push	esi	user32.7E39882A
00407CA1	. 33F6	xor	esi, esi	
00407CA3	. 33C9	xor	ecx, ecx	

Lo que ha hecho es recorrer todo el buffer donde se guardó la información del archivo de licencia y ha salido con el largo de este. Hace una comparación con 0x1A pero esta nunca se va a cumplir puesto que en mi archivo de licencia sólo hay caracteres imprimibles y ese no es uno de ellos.

Sigo traceando y veo esto:

00407CA9	. 894424 10	mov ss:[esp+10], eax	
00407CAD	. 85C0	test eax, eax	
00407CAF	.~ 76 34	jbe short 00407CE5	EFFind.00407CE5
00407CB1	. 55	push ebp	
00407CB2	. 57	push edi	
00407CB3	. 8D68 01	lea ebp, ds:[eax+1]	
00407CB6	> 0FB63C1E	movzx edi, byte ptr ds:[esi+ebx]	
00407CBA	. 8D0437	lea eax, ds:[edi+esi]	
00407CBD	. 33C8	xor ecx, eax	
00407CBF	. 33D2	xor edx, edx	
00407CC1	. 8BC1	mov eax, ecx	
00407CC3	. F7F5	div ebp	
00407CC5	. 03F9	add edi, ecx	
00407CC7	. 46	inc esi	
00407CC8	. 0FB6541A 01	movzx edx, byte ptr ds:[edx+ebx+1]	
00407CCD	. 03D1	add edx, ecx	
00407CCF	. 8D047A	lea eax, ds:[edx+edi*2]	
00407CD2	. 034424 18	add eax, ss:[esp+18]	
00407CD6	. 0FAFC1	imul eax, ecx	
00407CD9	. 894424 18	mov ss:[esp+18], eax	
00407CDD	. 3B7424 10	cmp esi, ss:[esp+10]	
00407CE1	.^ 72 D3	jb short 00407CB6	EFFind.00407CB6
00407CE3	. 5F	pop edi	

Ahí va a hacer operaciones con los caracteres del buffer del archivo de licencia y al salir del call retorna el valor obtenido en EAX. Si paso ese valor obtenido a decimal me dará justo la cadena con la que compara con lo que ya sé también de dónde saca esa cadena.

En resumen tenemos esto:

Parte obtenida desde los cálculos de la cadena del ID desencriptada y cuyo primer valor es un espacio.

Parte obtenida desde los cálculos con la cadena de la compañía.

Parte obtenida desde los cálculos con el buffer del archivo de licencia.

Parte obtenida desde los cálculos con la cadena del serial.

Parte obtenida desde los cálculos con la cadena del nombre.

Parte obtenida desde los cálculos con la cadena concatenada nombre+serial+compañía.

Parte del 8806.

Serial sin guiones.

```
1125301783635228287+1088806!!/24+"##&26512073835869371868
```

Si cojo el ID desencriptado de arriba y lo encripto usando la función que cree para el keygen del EF Checksum Manager queda así:

```
BBCFDABHIDGDFCCICIH.BAIIIAG$!0CE+##&&CGFBCAHIDFIGJDHBIGI
```

Con lo que los datos de registro quedan:

```
Name      : Aguml
Company   : CracksLatinos
Serial No.: 651207-38358-69371868
Reg.ID.   : BBCFDABHIDGDFCCICIH.BAIIIAG$!0CE+##&&CGFBCAHIDFIGJDHBIGI
```

Ahora solo queda estudiar cada algoritmo para saber cómo calcular cada cadena y para crear mi keygen uso el proyecto del keygen del EF Checksum Manager ya que es casi lo mismo solo que cambiando algunos de los algoritmos y añadiendo el de las cadenas concatenadas.

Así me quedó el código del keygen:

```
//-----
#include <vcl.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 Find 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 |= 0xfffffffffc;
                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 |= 0xfffffffffc;
                aux++;
            }
            valor -= aux;
            cadena[contador+1] = (char)valor;
        }
        contador++;
    }
    return cadena;
}
//-----
AnsiString CalcularValorSerial(AnsiString serial)
{
    unsigned long total = 0,hi;
    int contador = 0;
    unsigned __int64 aux;

    while(contador < serial.Length()){
        aux = (unsigned __int64)contador * 0xAAAAAAAB;

```

```

        hi = aux >> 32;
        hi = hi >> 1;
        total = total + hi;
        total = total + (serial[contador+1] * 2) + 1;
        contador++;
    }

    return AnsiString().sprintf("%04lu", total);
}
//-----

AnsiString CalcularValorNombre(AnsiString nombre)
{
    unsigned long total = 0x7;
    int contador = 0;

    while(contador < nombre.Length()){
        total = nombre[contador + 1] + contador + (total * 2) - 1;
        contador++;
    }
    return AnsiString().sprintf("%05lu", total);
}
//-----

AnsiString ObtenerValorCompania(AnsiString compania)
{
    unsigned long valor1=0, valor2=0, retval=0x20;
    int pos=0;

    if(compania.Length() >= 2){
        while((compania.Length() - pos) > 1){
            valor1 += compania[pos + 1] - pos;
            pos++;
            valor2 += compania[pos + 1] - (pos - 1) - 1;
            pos++;
        }
    }

    if(compania.Length() - pos == 1)
        retval += (compania[pos + 1] - pos);

    retval += (valor1 + valor2);
    return AnsiString().sprintf("%03lu", retval);
}
//-----

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

    for(int contador = 0; contador < sizebuffer; ++contador)
    {
        resto = resto ^ cadena[contador] + contador;
        retval = (cadena[(resto % (sizebuffer + 1)) + 1] + resto + (cadena[contador] +
resto) * 2 + retval) * resto;
    }
    return AnsiString().sprintf("%09lu", retval);
}
//-----
```

```

AnsiString CalcularValorID(AnsiString id)
{
    unsigned long valor1, valor2, valor3, total = 1;
    int pos = 1;

    id[1] = 0x20;

    while(pos < 57){
        valor1 = id[pos];
        pos++;
        valor2 = id[pos];
        pos++;
        valor3 = id[pos];
        pos++;
        total = total + ((valor1 + valor2 + valor3) * 4) + 0x15;
    }
    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];

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

    return AnsiString().sprintf("%03lu", retval);
}
//-----


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

    //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(9999999);

        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
//caracter hasta llegar a 57 caracteres
id="";
for(int x=1, i=2; x <= 57; x++, i++){
    if(i > 19)
        i = (x & 3) + 2;
    id += (char)aux[i];
}

//Obtengo el tamaño que ocupará todo el texto
sizeBuffer = snprintf(NULL, 0, "%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 < 57; 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(1,38) + aux;

//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, 16) + aux.SubString(2, 3) + id.SubString(20, id.Length() - 19);

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

//La coloco en el ID
id = id.SubString(1, 23) + aux.SubString(1, 4) + id.SubString(28, id.Length() - 27);

//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, 13) + aux.SubString(1, 3) + id.SubString(17, id.Length() - 16);

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

//La coloco en su lugar
id = id.SubString(1, 1) + aux.SubString(1, 3) + id.SubString(5, id.Length() - 4);

//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, 4) + aux.SubString(1, 9) + id.SubString(14, id.Length() - 13);

//Obtengo el valor del nombre, serial y compañía concatenados
aux = ValorDeConcatenados(EditNombre->Text + EditSerial->Text + EditCompania->Text);
id = id.SubString(1,20) + aux.SubString(1,3) + id.SubString(24, id.Length() - 23);

//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<=57; 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("EFFin.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)
{
    bufferLleno = false;
    bufferArchivo = NULL;
    listaNegra = new TStringList;

    listaNegra->Add("323848-53000-90750642");
    listaNegra->Add("013990-54715-90721932");
    listaNegra->Add("533278-62213-90770938");
    listaNegra->Add("983528-11618-90710935");
    listaNegra->Add("273038-31919-90741931");
    listaNegra->Add("783721-02428-90790933");
    listaNegra->Add("443964-62112-90761932");
    listaNegra->Add("363140-72000-90850937");
    listaNegra->Add("323987-11024-90850939");
    listaNegra->Add("113097-11815-90731931");
    listaNegra->Add("593913-02237-90770941");
    listaNegra->Add("583923-12238-90770941");
    listaNegra->Add("043163-51712-90720939");
    listaNegra->Add("483924-22118-90761932");
    listaNegra->Add("623882-81314-90781933");
    listaNegra->Add("633474-21313-90780936");
    listaNegra->Add("663445-72310-90780936");
    listaNegra->Add("713495-12415-90790936");
    listaNegra->Add("433774-91143-98761932");
    listaNegra->Add("533870-46223-90771973");
    listaNegra->Add("833575-83513-90700935");
    listaNegra->Add("343865-14072-90751941");
    listaNegra->Add("683022-32318-90781931");
    listaNegra->Add("283626-02918-90740934");
    listaNegra->Add("053658-32711-90720934");
    listaNegra->Add("343061-12062-90750948");
    listaNegra->Add("583629-51218-90770934");
    listaNegra->Add("823688-73514-90700934");
    listaNegra->Add("883829-22518-90701933");
    listaNegra->Add("593518-12237-90770935");
    listaNegra->Add("403306-12176-90760935");
    listaNegra->Add("343662-62052-90751933");
    listaNegra->Add("453952-63121-90760931");
    listaNegra->Add("713599-62435-90790935");
    listaNegra->Add("423885-53114-90761933");
    listaNegra->Add("333569-01022-90750935");
    listaNegra->Add("063926-43738-90720931");
    listaNegra->Add("633271-52343-98780936");
    listaNegra->Add("563442-12210-90770936");
    listaNegra->Add("063548-03710-90720935");
    listaNegra->Add("343766-40032-90750913");
    listaNegra->Add("493818-25127-90761973");
    listaNegra->Add("313808-63006-90750642");
    listaNegra->Add("403900-50126-90761912");
    listaNegra->Add("483353-01141-90760935");
    listaNegra->Add("343360-44052-90750935");
}
```

```
listaNegra->Add( "393715-92067-90750931" );
listaNegra->Add( "323580-83064-90750933" );
listaNegra->Add( "423485-03174-90760944" );
listaNegra->Add( "343863-61072-90751931" );
listaNegra->Add( "393612-82087-90751933" );
listaNegra->Add( "113798-63815-90730933" );
listaNegra->Add( "233078-71913-90741931" );
listaNegra->Add( "753056-12411-90791931" );
listaNegra->Add( "493880-14134-90760942" );
listaNegra->Add( "433374-20133-90760907" );
listaNegra->Add( "573032-23239-90771931" );
listaNegra->Add( "393958-01011-90750931" );

randomize();
}

//-----
```

