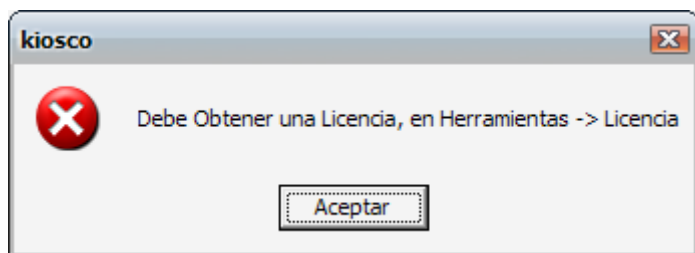




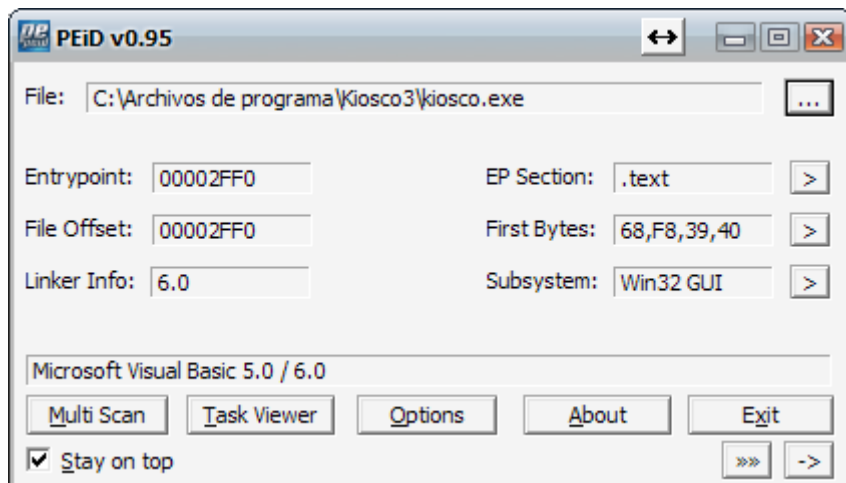
## Estudiando un Visual Basic, Kiosco 3.0

Fecha	30-3-2012
Victima	Kiosco 3.0
URL de descarga	<a href="http://www.softweb.com.ar/kiosco/descargar">http://www.softweb.com.ar/kiosco/descargar</a>
MD5 del instalador	CA3E9FE4248B1EE70891A04FB329D9FF
Protección	Serial
Herramientas	Ollydbg 1.10
Objetivo	Hacer el software funcional
Dificultad	Media
Cracker	Agum1

Lo descargamos de su página web y en ella no vemos nada que nos hable ni de restricciones ni nada pero si lo instalamos e intentamos trastear nos encontramos con este cartelito al intentar acceder a las siguientes opciones del menú: Imprimir, Estadísticas, Cajas, Herramientas→Configurar, Herramientas→Hacer backup, Herramientas→Restaurar backup, y seguro que hay mas pero esas son las restricciones que pude ver yo.



Le pasamos el Peid a ver que nos dice y vemos esto:



Bueno, pues es un VB sin protección de terceros así que no tendremos que preocuparnos de desempacar nada. Lo abrimos en Olly y vemos esto:

00402FF0	\$ 68 F8394000	push	4039F8	
00402FF5	. E8 EFFFFFFF	call	00402FE8	<jmp. \$MSVBVM60.#100>
00402FFA	. 0000	add	ds:[eax], al	
00402FFC	. 0000	add	ds:[eax], al	
00402FFE	. 0000	add	ds:[eax], al	
00403000	. 3000	xor	ds:[eax], al	
00403002	. 0000	add	ds:[eax], al	
00403004	. 40	inc	eax	
00403005	. 0000	add	ds:[eax], al	
00403007	. 0000	add	ds:[eax], al	
00403009	. 0000	add	ds:[eax], al	
0040300B	. 00CD	add	ch, cl	
0040300D	. BC 2A70955B	mov	esp, 5B95702A	
00403012	F0	db	F0	
00403013	4F	db	4F	CHAR 'O'
00403014	B6	db	B6	
00403015	87	db	87	
00403016	65	db	65	CHAR 'e'
00403017	9B	db	9B	
00403018	F8	db	F8	

Y se ve claramente que si es un VB así que manos a la obra.

Pues bien, usaré algo que me suele funcionar a ver que pasa. Lo que hago es hacer que me muestre ese mensaje de chico malo y lo voy a hacer en este caso eligiendo la opción del menú Estadísticas y cuando me salga el mensaje pausaré el Olly y luego continuaré con Ctról + F9 para llegar al return y luego un F7 para ejecutar el return y así siempre hasta que lleguemos a la sección code del ejecutable y con este método llego aquí:

004242B3	. 894D B4	mov	ss:[ebp-4C], ecx	
004242B6	. 894D C4	mov	ss:[ebp-3C], ecx	
004242B9	. 8D55 94	lea	edx, ss:[ebp-6C]	
004242BC	. 8D4D D4	lea	ecx, ss:[ebp-2C]	
004242BF	. 8945 AC	mov	ss:[ebp-54], eax	
004242C2	. 8945 BC	mov	ss:[ebp-44], eax	
004242C5	. 8945 CC	mov	ss:[ebp-34], eax	
004242C8	. C745 9C 5001	mov	dword ptr ss:[ebp-64], 410150	UNICODE "Debe Obtener una Licencia, en Herramientas -> Licencia"
004242CF	. C745 94 0800	mov	dword ptr ss:[ebp-6C], 8	
004242D6	. FF15 78124000	call	ds:[401278]	MSVBVM60.__vbaVarDup
004242DC	. 8D4D A4	lea	ecx, ss:[ebp-5C]	
004242DF	. 8D55 B4	lea	edx, ss:[ebp-4C]	
004242E2	. 51	push	ecx	
004242E3	. 8D45 C4	lea	eax, ss:[ebp-3C]	
004242E6	. 52	push	edx	
004242E7	. 50	push	eax	
004242E8	. 8D4D D4	lea	ecx, ss:[ebp-2C]	
004242EB	. 6A 10	push	10	Arg2 = 00000010
004242ED	. 51	push	ecx	Arg1
004242FE	. FF15 A8104000	call	ds:[4010A8]	rtcMsgBox
004242F4	. 8D55 A4	lea	edx, ss:[ebp-5C]	
004242F7	. 8D45 B4	lea	eax, ss:[ebp-4C]	
004242FA	. 52	push	edx	

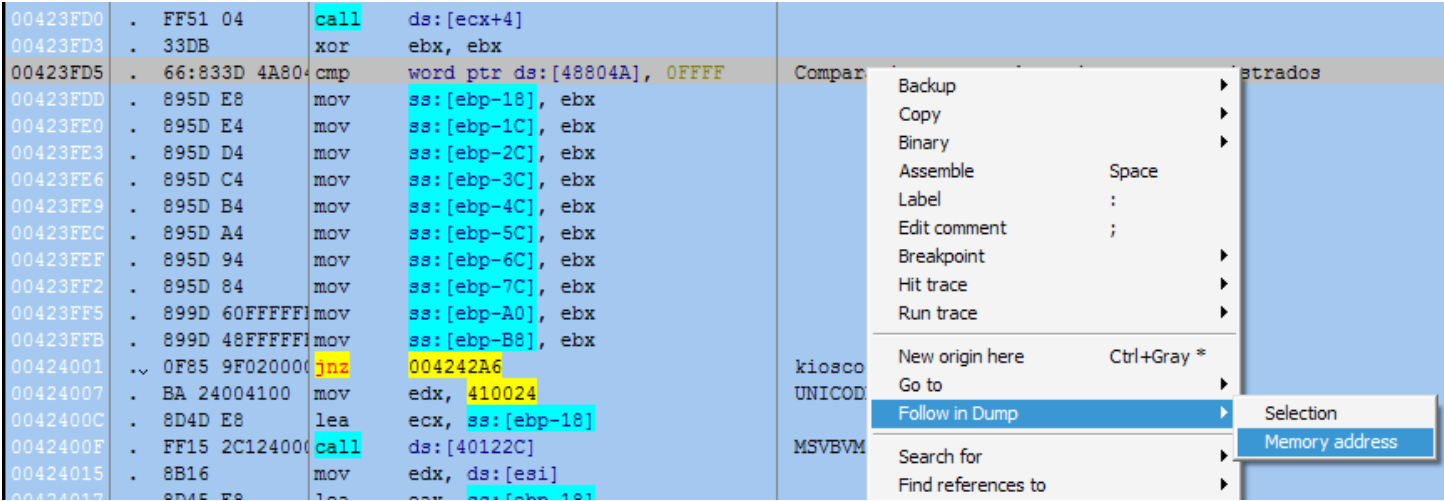
La verdad es que tiene muy buena pinta y si subimos un poco más:

00424287	. 68 A4F74000	push	40F7A4	Arg3 = 0040F7A4
0042428C	. 56	push	esi	Arg2
0042428D	. 50	push	eax	Arg1
0042428E	. FF15 80104000	call	ds:[401080]	__vbaHresultCheckObj
00424294	> 8D4D E4	lea	ecx, ss:[ebp-1C]	
00424297	. FF15 EC124000	call	ds:[4012EC]	MSVBVM60.__vbaFreeObj
0042429D	. E8 BE550500	call	00479860	kiosco.00479860
004242A2	. 33DB	xor	ebx, ebx	
004242A4	~ EB 69	jmp	short 0042430F	kiosco.0042430F
004242A6	> B9 0A000000	mov	ecx, 0A	
004242AB	. B8 04000280	mov	eax, 80020004	
004242B0	. 894D A4	mov	ss:[ebp-5C], ecx	
004242B3	. 894D B4	mov	ss:[ebp-4C], ecx	
004242B6	. 894D C4	mov	ss:[ebp-3C], ecx	
004242B9	. 8D55 94	lea	edx, ss:[ebp-6C]	
004242BC	. 8D4D D4	lea	ecx, ss:[ebp-2C]	
004242BF	. 8945 AC	mov	ss:[ebp-54], eax	
004242C2	. 8945 BC	mov	ss:[ebp-44], eax	
004242C5	. 8945 CC	mov	ss:[ebp-34], eax	
004242C8	. C745 9C 5001	mov	dword ptr ss:[ebp-64], 410150	UNICODE "Debe Obtener una Licencia, en Herramientas -> Licencia"

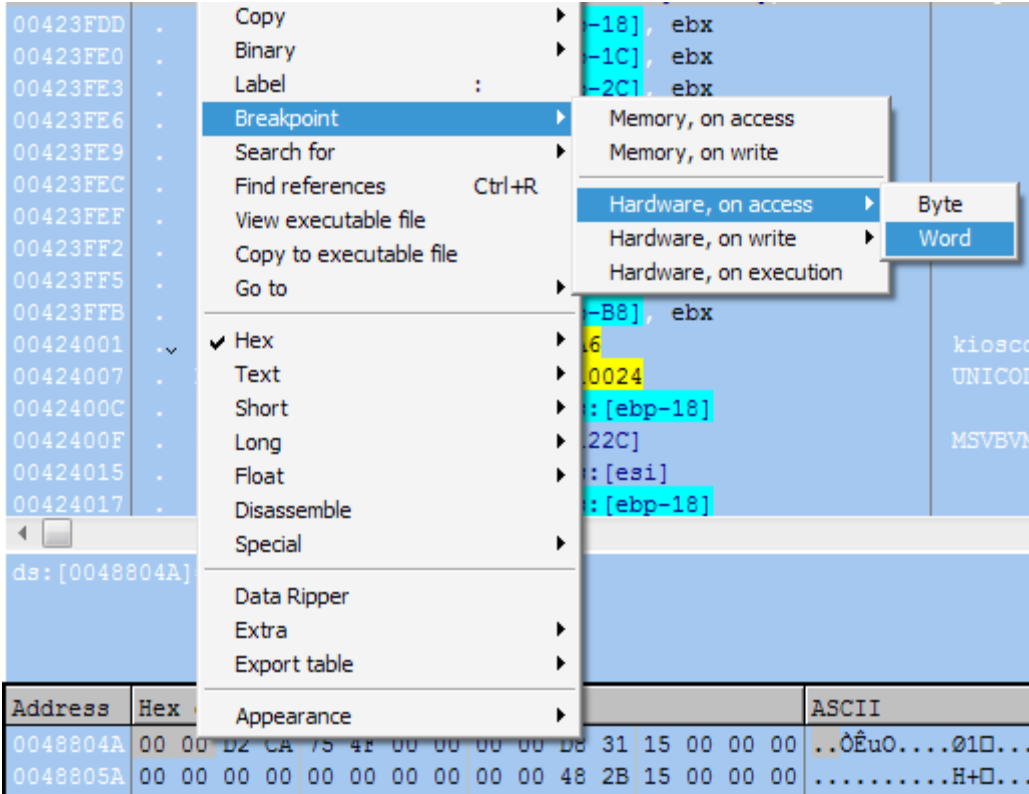
¿Desde donde llegará a esta función?

00423FCB	. 8975 08	mov	ss:[ebp+8], esi	
00423FCE	. 8B0E	mov	ecx, ds:[esi]	
00423FD0	. FF51 04	call	ds:[ecx+4]	
00423FD3	. 33DB	xor	ebx, ebx	
00423FD5	. 66:833D 4A80	cmp	word ptr ds:[48804A], 0FFFF	Comparacion para saber si estamos registrados
00423FDD	. 895D E8	mov	ss:[ebp-18], ebx	
00423FE0	. 895D E4	mov	ss:[ebp-1C], ebx	
00423FE3	. 895D D4	mov	ss:[ebp-2C], ebx	
00423FE6	. 895D C4	mov	ss:[ebp-3C], ebx	
00423FE9	. 895D B4	mov	ss:[ebp-4C], ebx	
00423FEC	. 895D A4	mov	ss:[ebp-5C], ebx	
00423FEF	. 895D 94	mov	ss:[ebp-6C], ebx	
00423FF2	. 895D 84	mov	ss:[ebp-7C], ebx	
00423FF5	. 899D 60FFFFFF	mov	ss:[ebp-A0], ebx	
00423FFB	. 899D 48FFFFFF	mov	ss:[ebp-B8], ebx	
00424001	~ 0F85 9F020000	jnz	004242A6	kiosco.004242A6
00424007	. BA 24004100	mov	edx, 410024	UNICODE "frm_estadisticas"
0042400C	. 8D4D E8	lea	ecx, ss:[ebp-18]	
0042400F	. FF15 2C124000	call	ds:[40122C]	MSVBVM60.__vbaStrCopy
00424015	. 8B16	mov	edx, ds:[esi]	

Ahí tenemos el salto y si miramos un poquito mas arriba vemos una comparación para decidir si salta o no y vemos que compara una dirección fija con una constante. Probemos a poner un HBP on access en el word en la dirección fija de la comparación. Para ello hacemos clic derecho sobre la línea de la comparación y vamos a la siguiente ruta del menú contextual:



Nos vamos al Dump y seleccionamos el word que compara y hacemos clic derecho sobre el y navegamos en el menú para poner un HBP:



Damos a F9 y volvemos a dar en Estadísticas->Abrir en el programa y vemos que para justo debajo de la comparación con lo cual ya sabemos que el valor a esa zona se lo tiene que dar justo al arrancar el programa y no durante la ejecución del mismo así que reiniciemos con Ctról + F2 y demos a F9 y vemos que la primera vez que para lo hace en la librería de VB así que volvemos a dar a F9 a ver donde para esta vez:




00422944	. 6A 04	push	4	
00422946	. FF15 38104000	call	ds:[401038]	MSVBVM60.__vbaFreeVarList
0042294C	. 83C4 14	add	esp, 14	
0042294F	. FF15 3C104000	call	ds:[40103C]	L __vbaEnd
00422955	. 33DB	xor	ebx, ebx	
00422957	> A1 788B4800	mov	eax, ds:[488B78]	
0042295C	. 66:891D 4A80	mov	ds:[48804A], bx	Aqui cambia el word "magico"
00422963	. 3BC3	cmp	eax, ebx	
00422965	. 75 10	jnz	short 00422977	kiosco.00422977
00422967	. 68 788B4800	push	488B78	Arg2 = 00488B78 ASCII "dã0"
0042296C	. 68 00F44000	push	40F400	Arg1 = 0040F400
00422971	. FF15 18124000	call	ds:[401218]	__vbaNew2

Y vemos que EBX vale 0 con lo cual ahí ya nos ha puesto como no registrado así que probemos a cambiar el word al que apunta en la línea de arriba de donde paró con un FFFF y podemos ver que ya no tenemos ninguna restricción pero tenemos algo que no se quitó del menú y es esto:



Falta algo por ahí ya que nos sigue mostrando para que nos registremos y si damos a Obtener una Licencia nos abre el navegador y vemos esto:

Kiosco 3.0 Descargar Ayuda ▾ Foro Screenshot ▾ Newsletter Contacto

Compartir   

Apellido:

Nombre:

Email:

Telefono:

**Precio: \$100 arg (U\$S 24)**

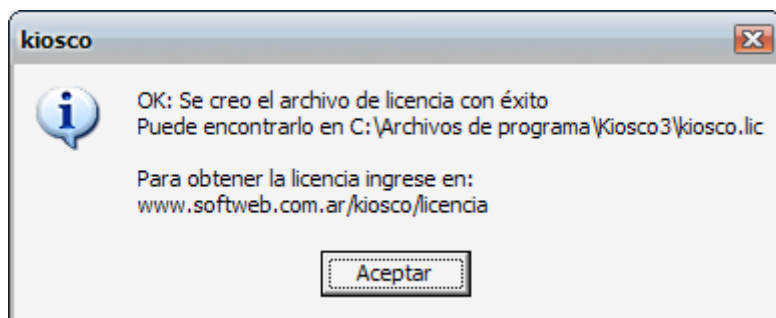
**Forma de pago:**

- ☐ Pago Facil, Rapipago, Bapro
- ☐ DineroMail
- ☐ Transferencia/Deposito Bancario
- ☐ Western Union
- ☐ MoneyGram
- ☐ Otra Forma de Pago

**Archivo licencia.lic generado en Herramientas->Crear Archivo de Licencia**

Nota: Recibira un e-mail con los datos para realizar el pago correspondiente, y luego de confirmar el mismo le enviaremos por e-mail el archivo de licencia .

O sea que primero hay que crear un archivo de licencia como nos indica aquí así que lo hacemos y listo:



Resumamos, tenemos que crear un archivo llamado kiosco.lic el cual lo mandamos al proveedor de licencias y este nos manda una licencia previo pago. ¿Qué habrá dentro de ese archivo de licencia?

Este es su contenido:

140,147,147,95,107,110,98,135,225,149,105,105,110,120,156,156,154,93,106,123,115,152,155,149,104,181,122,105,152,153,147,114,125,106,94,136,168,166,97,113,105,100,211,149,149,96,107,105,97,135,156,222,114,124,138,133,176,184,130,94,89,110,98,139,149,146,97,105,180,98,141,148,146,99,104,105,103,211,179,215,157,165,180,115,164,169,130,133,174,170,155,198,211,138,165,166,97,82,141,153,130,126,168,154,155,195,202,130,133,158,155,154,197,212,206,160,160,177,82,164,176,143,100,113



Sabemos que tiene que ser algo relacionado con información de nuestro PC pero tiene pinta de estar encriptado así que pondremos un BP en la función usada normalmente en VB para crear un archivo que es `__vbaFileOpen` y volvemos a crear el archivo para ver si desde ahí llegamos al lugar donde encripta y así saber que estamos mandando.

Una vez volvemos a intentar crear el archivo de licencia, nos para en el BP que pusimos y damos a `Ctrol + F9` y a `F7` para pasar el return y volvemos al código de la víctima:

00425F9A	. FF15 80104000	call	ds:[401080]	MSVBVM60.__vbaHresultCheckObj
00425FA0	> 8B45 DC	mov	eax, ss:[ebp-24]	
00425FA3	. 50	push	eax	
00425FA4	. 68 20084100	push	410820	UNICODE "\kiosco.lic"
00425FA9	. FFD7	call	edi	
00425FAB	. 8BD0	mov	edx, eax	
00425FAD	. 8D4D D8	lea	ecx, ss:[ebp-28]	
00425FB0	. FFD6	call	esi	
00425FB2	. 50	push	eax	[Arg4 Arg3 = 00000001 Arg2 = FFFFFFFF Arg1 = 00000002 __vbaFileOpen
00425FB3	. 6A 01	push	1	
00425FB5	. 6A FF	push	-1	
00425FB7	. 6A 02	push	2	
00425FB9	. FF15 08124000	call	ds:[401208]	
00425FBF	. 8D4D D8	lea	ecx, ss:[ebp-28]	
00425FC2	. 8D55 DC	lea	edx, ss:[ebp-24]	
00425FC5	. 51	push	ecx	
00425FC6	. 52	push	edx	

Voy subiendo a ver si veo algo interesante pero no veo nada que me llame la atención y si muchos calls así que subo bastante para ir traceando con `F8` pasando los calls a ver que se puede ver y pongo un BP en la siguiente dirección:

00425D77	. C2 0400	retn	4	
00425D7A	> FF15 04124000	call	ds:[401204]	MSVBVM60.__vbaErrorOverflow
00425D80	> 55	push	ebp	
00425D81	. 8BEC	mov	ebp, esp	
00425D83	. 83EC 0C	sub	esp, 0C	
00425D86	. 68 862B4000	push	402B86	SE handler installation
00425D8B	. 64:A1 00000000	mov	eax, fs:[0]	
00425D91	. 50	push	eax	
00425D92	. 64:8925 00000000	mov	fs:[0], esp	
00425D99	. 81EC C8000000	sub	esp, 0C8	
00425D9F	. 53	push	ebx	

Vuelvo a crear el archivo para ver si para en esa parte e ir traceando con `F8` y veo esto:

Registers (FPU)		< < < < < < < < < < < < < <															
EAX	001757D4	UNICODE "5.1.2600 3806FE78,1CAA637 B7A41AD2,1CD0812 03/21/07 ACRSYS - 6040000 06/02/15 Null AMD Turion(tm) 64"															
ECX	0012F7E8																
EDX	00730002																
EBX	00000001																
ESP	0012F714																
EBP	0012F7FC																
ESI	0012F8D8																
EDI	00000000																
EIP	00425E02	kiosco.00425E02															

Al pasar la call de 425E02 y un poco mas abajo vemos una cadena con el texto `Web1982`:

00425DF1	. 89BD 78FFFFFF	mov	ss:[ebp-88], edi	
00425DF7	. 89BD 68FFFFFF	mov	ss:[ebp-98], edi	
00425DFD	. E8 8EAC0500	call	00480A90	Aqui consigue todos los datos de la pc para crear el archivo a enviar
00425E02	. 8B35 B0124000	mov	esi, ds:[4012B0]	MSVBVM60.__vbaStrMove
00425E08	. 8BD0	mov	edx, eax	
00425E0A	. 8D4D D4	lea	ecx, ss:[ebp-2C]	
00425E0D	. FFD6	call	esi	<MSVBVM60.__vbaStrMove>
00425E0F	. BA 6CEF4000	mov	edx, 40EF6C	UNICODE "Web1982"
00425E14	. 8D4D D8	lea	ecx, ss:[ebp-28]	
00425E17	. FF15 2C124000	call	ds:[40122C]	MSVBVM60.__vbaStrCopy

La cadena esa tan larga que está en EAX tiene información de mí maquina y a simple vista puedo reconocer:

- La versión de mi sistema operativo que es la 5.1.2600.
- Información sobre mi procesador.

Con lo cual ya sabemos que si queremos hacer una medicina o parcheamos el programa o nos toca crear un keygen porque no nos servirá un archivo de licencia para 2 maquinas diferentes y es un inconveniente porque si adquieres una licencia y se te fastidia por ejemplo el procesador o cambias de S.O. ya no te serviría la licencia.

Traceamos un poco mas para ver que va haciendo y voy pasando todos los calls de VB hasta que llego a un call que es del programa en 425E30 y entro con F7 y sigo traceando hasta que llego aquí:

0047A017	> 66:8B55 DC	mov	dx, ss:[ebp-24]	Mueve a dx el contador para el largo de nuestra cadena
0047A01B	. 66:0395 7CFF	add	dx, ss:[ebp-84]	incrementa contador
0047A022	~ 0F80 5F020000	jo	0047A287	Salta si se desborda dx
0047A028	. 66:8955 DC	mov	ss:[ebp-24], dx	salva el contador
0047A02C	> 66:8B45 DC	mov	ax, ss:[ebp-24]	pasa el contador a ax
0047A030	. 66:3B85 78FF	cmp	ax, ss:[ebp-88]	compara el contador con el largo de nuestra cadena
0047A037	~ 0F8F F2010000	jg	0047A22F	salta si es mayor
0047A03D	. C745 FC 0400	mov	dword ptr ss:[ebp-4], 4	
0047A044	. C745 C0 0100	mov	dword ptr ss:[ebp-40], 1	
0047A04B	. C745 B8 0200	mov	dword ptr ss:[ebp-48], 2	
0047A052	. 8B4D 08	mov	ecx, ss:[ebp+8]	Mueve un puntero a nuestra cadena a ecx
0047A055	. 894D A0	mov	ss:[ebp-60], ecx	Mueve un puntero a Web1982
0047A058	. C745 98 0840	mov	dword ptr ss:[ebp-68], 4008	
0047A05F	. 8D55 B8	lea	edx, ss:[ebp-48]	
0047A062	. 52	push	edx	Arg4
0047A063	. 0FBF45 DC	movsx	eax, word ptr ss:[ebp-24]	Arg3
0047A067	. 50	push	eax	Arg2
0047A068	. 8D4D 98	lea	ecx, ss:[ebp-68]	Arg1
0047A06B	. 51	push	ecx	rtcMidCharVar
0047A06C	. 8D55 A8	lea	edx, ss:[ebp-58]	
0047A06F	. 52	push	edx	
0047A070	. FF15 F8104000	call	ds:[4010F8]	
0047A076	. 8D45 A8	lea	eax, ss:[ebp-58]	
0047A079	. 50	push	eax	
0047A07A	. FF15 2C104000	call	ds:[40102C]	MSVBVM60.__vbaStrVarMove
0047A080	. 50	push	eax	
0047A081	. 8D4D D8	lea	ecx, ss:[ebp-28]	
0047A084	. 51	push	ecx	
0047A085	. 6A 01	push	1	
0047A087	. FF15 F0114000	call	ds:[4011F0]	MSVBVM60.__vbaLsetFixstrFree
0047A08D	. 8D55 A8	lea	edx, ss:[ebp-58]	
0047A090	. 52	push	edx	
0047A091	. 8D45 B8	lea	eax, ss:[ebp-48]	
0047A094	. 50	push	eax	



0047A097	. FF15 38104000	call	ds:[401038]	MSVBVM60.__vbaFreeVarList
0047A09D	. 83C4 0C	add	esp, 0C	
0047A0A0	. C745 FC 0500	mov	dword ptr ss:[ebp-4], 5	
0047A0A7	. C745 C0 0100	mov	dword ptr ss:[ebp-40], 1	
0047A0AE	. C745 B8 0200	mov	dword ptr ss:[ebp-48], 2	
0047A0B5	. 8B4D 0C	mov	ecx, ss:[ebp+C]	Mueve a ECX un puntero a Web1982
0047A0B8	. 894D A0	mov	ss:[ebp-60], ecx	
0047A0BB	. C745 98 0840	mov	dword ptr ss:[ebp-68], 4008	
0047A0C2	. 8D55 B8	lea	edx, ss:[ebp-48]	
0047A0C5	. 52	push	edx	
0047A0C6	. 66:8B45 DC	mov	ax, ss:[ebp-24]	contador para el bucle de Web1982
0047A0CA	. 66:2D 0100	sub	ax, 1	decrementa el contador de Web1982
0047A0CE	.. 0F80 B3010000	jo	0047A287	salta si se desborda ax
0047A0D4	. 0FBFF0	movsx	esi, ax	
0047A0D7	. 8B4D 0C	mov	ecx, ss:[ebp+C]	Mueve a ECX un puntero a Web1982
0047A0DA	. 8B11	mov	edx, ds:[ecx]	Mueve la direccion a la que apunta el puntero Web1982 a EDX
0047A0DC	. 52	push	edx	mete la direccion de la pila para medir el largo a continuacion
0047A0DD	. FF15 30104000	call	ds:[401030]	MSVBVM60.__vbaLenBstr
0047A0E3	. 8BC8	mov	ecx, eax	Nos da como resultado 7 y lo mueve a ECX
0047A0E5	. 8BC6	mov	eax, esi	Mueve a EAX el contador de Web1982
0047A0E7	. 99	cdq		
0047A0E8	. F7F9	idiv	ecx	Divide EAX que es el contador de Web1982 entre ECX que es el largo de Web1982
0047A0EA	. 83C2 01	add	edx, 1	Añade 1 a EDX
0047A0ED	.. 0F80 94010000	jo	0047A287	Si hay desbordamiento salta
0047A0F3	. 52	push	edx	Arg3
0047A0F4	. 8D55 98	lea	edx, ss:[ebp-68]	
0047A0F7	. 52	push	edx	Arg2
0047A0F8	. 8D45 A8	lea	eax, ss:[ebp-58]	
0047A0FB	. 50	push	eax	Arg1
0047A0FC	. FF15 F8104000	call	ds:[4010F8]	rtcMidCharVar
0047A102	. 8D4D A8	lea	ecx, ss:[ebp-58]	
0047A105	. 51	push	ecx	
0047A106	. FF15 2C104000	call	ds:[40102C]	MSVBVM60.__vbaStrVarMove
0047A10C	. 50	push	eax	

0047A10C	. 50	push	eax	
0047A10D	. 8D55 D0	lea	edx, ss:[ebp-30]	
0047A110	. 52	push	edx	
0047A111	. 6A 01	push	1	
0047A113	. FF15 F0114000	call	ds:[4011F0]	MSVBVM60.__vbaLsetFixstrFree
0047A119	. 8D45 A8	lea	eax, ss:[ebp-58]	
0047A11C	. 50	push	eax	
0047A11D	. 8D4D B8	lea	ecx, ss:[ebp-48]	
0047A120	. 51	push	ecx	
0047A121	. 6A 02	push	2	
0047A123	. FF15 38104000	call	ds:[401038]	MSVBVM60.__vbaFreeVarList
0047A129	. 83C4 0C	add	esp, 0C	
0047A12C	. C745 FC 0600	mov	dword ptr ss:[ebp-4], 6	
0047A133	. 8D55 D8	lea	edx, ss:[ebp-28]	
0047A136	. 52	push	edx	
0047A137	. 6A 01	push	1	
0047A139	. FF15 C8104000	call	ds:[4010C8]	MSVBVM60.__vbaStrFixstr
0047A13F	. 8BD0	mov	edx, eax	
0047A141	. 8D4D CC	lea	ecx, ss:[ebp-34]	
0047A144	. FF15 B0124000	call	ds:[4012B0]	MSVBVM60.__vbaStrMove
0047A14A	. 50	push	eax	Arg1
0047A14B	. FF15 4C104000	call	ds:[40104C]	rtcAnsiValueBstr
0047A151	. 66:8945 84	mov	ss:[ebp-7C], ax	Obtiene el hexa de cada caracter de nuestra cadena y lo guarda
0047A155	. 8B45 CC	mov	eax, ss:[ebp-34]	
0047A158	. 50	push	eax	Arg3
0047A159	. 8D4D D8	lea	ecx, ss:[ebp-28]	
0047A15C	. 51	push	ecx	Arg2
0047A15D	. 6A 01	push	1	Arg1 = 00000001
0047A15F	. FF15 64104000	call	ds:[401064]	__vbaLsetFixstr
0047A165	. 8D55 D0	lea	edx, ss:[ebp-30]	
0047A168	. 52	push	edx	
0047A169	. 6A 01	push	1	
0047A16B	. FF15 C8104000	call	ds:[4010C8]	MSVBVM60.__vbaStrFixstr
0047A171	. 8BD0	mov	edx, eax	

0047A173	. 8D4D C8	lea ecx, ss:[ebp-38]	MSVBVM60.__vbaStrMove
0047A176	. FF15 B0124000	call ds:[4012B0]	[Arg1
0047A17C	. 50	push eax	rtcAnsiValueBstr
0047A17D	. FF15 4C104000	call ds:[40104C]	Obtiene el valor hexa de cada caracter de Web1982 y lo guarda
0047A183	. 66:8945 80	mov ss:[ebp-80], ax	
0047A187	. 8B45 C8	mov eax, ss:[ebp-38]	
0047A18A	. 50	push eax	[Arg3
0047A18B	. 8D4D D0	lea ecx, ss:[ebp-30]	Arg2
0047A18E	. 51	push ecx	Arg1 = 00000001
0047A18F	. 6A 01	push 1	__vbaLsetFixstr
0047A191	. FF15 64104000	call ds:[401064]	
0047A197	. 66:8B55 84	mov dx, ss:[ebp-7C]	
0047A19B	. 66:0355 80	add dx, ss:[ebp-80]	Suma el caracter obtenido de nuestra cadena con el obtenido de la cadena Web1982
0047A19F	. 0F80 E2000000	js 0047A287	Salta si hay desbordamiento
0047A1A5	. 0FBFC2	movsx eax, dx	Mueve el resultado a EAX
0047A1A8	. 50	push eax	[Arg2
0047A1A9	. 8D4D B8	lea ecx, ss:[ebp-48]	
0047A1AC	. 51	push ecx	[Arg1
0047A1AD	. FF15 C4114000	call ds:[4011C4]	rtcVarBstrFromAnsi
0047A1B3	. 8D55 B8	lea edx, ss:[ebp-48]	
0047A1B6	. 52	push edx	
0047A1B7	. FF15 2C104000	call ds:[40102C]	MSVBVM60.__vbaStrVarMove
0047A1BD	. 50	push eax	
0047A1BE	. 8D45 D8	lea eax, ss:[ebp-28]	
0047A1C1	. 50	push eax	
0047A1C2	. 6A 01	push 1	
0047A1C4	. FF15 F0114000	call ds:[4011F0]	MSVBVM60.__vbaLsetFixstrFree
0047A1CA	. 8D4D C8	lea ecx, ss:[ebp-38]	
0047A1CD	. 51	push ecx	
0047A1CE	. 8D55 CC	lea edx, ss:[ebp-34]	
0047A1D1	. 52	push edx	
0047A1D2	. 6A 02	push 2	
0047A1D4	. FF15 38124000	call ds:[401238]	MSVBVM60.__vbaFreeStrList
0047A1DA	. 83C4 0C	add esp, 0C	
0047A1DD	. 8D4D B8	lea ecx, ss:[ebp-48]	
0047A1E0	. FF15 24104000	call ds:[401024]	MSVBVM60.__vbaFreeVar
0047A1E6	. C745 FC 0700	mov dword ptr ss:[ebp-4], 7	
0047A1ED	. 8B45 D4	mov eax, ss:[ebp-2C]	
0047A1F0	. 50	push eax	
0047A1F1	. 8D4D D8	lea ecx, ss:[ebp-28]	
0047A1F4	. 51	push ecx	
0047A1F5	. 6A 01	push 1	
0047A1F7	. FF15 C8104000	call ds:[4010C8]	MSVBVM60.__vbaStrFixstr
0047A1FD	. 8BD0	mov edx, eax	
0047A1FF	. 8D4D CC	lea ecx, ss:[ebp-34]	
0047A202	. FF15 B0124000	call ds:[4012B0]	MSVBVM60.__vbaStrMove
0047A208	. 50	push eax	[Arg1
0047A209	. FF15 60104000	call ds:[401060]	__vbaStrCat
0047A20F	. 8BD0	mov edx, eax	
0047A211	. 8D4D D4	lea ecx, ss:[ebp-2C]	
0047A214	. FF15 B0124000	call ds:[4012B0]	MSVBVM60.__vbaStrMove
0047A21A	. 8D4D CC	lea ecx, ss:[ebp-34]	
0047A21D	. FF15 F0124000	call ds:[4012F0]	MSVBVM60.__vbaFreeStr
0047A223	. C745 FC 0800	mov dword ptr ss:[ebp-4], 8	
0047A22A	. E9 E8FDFFFF	jmp 0047A017	kiosco.0047A017

Bueno, pues no entiendo muy bien lo que pasa con tantas funciones de VB que las usa para casi todo pero algo he entendido. He visto que usa la cadena Web1982 para ir sumando caracteres a la cadena que obtuvo antes de mi pc y, con el idivx que hay por medio, dividiendo el contador entre el largo de la cadena, usa el resto de la división + 1 para posicionarse en el carácter a sumar de la cadena Web1982 con el carácter que toque de la cadena de nuestro PC.

Imaginemos que tenemos en mi caso la cadena:

“5.1.2600|3806FE78,1CAA637|B7A41AD2,1CD0812|03/21/07|ACRSYS - 6040000|06/02/15|Null|AMD Turion(tm) 64”

Pues lo que hace es simplemente esto:

5.1.2600|3806FE78,1CAA637|B7A41AD2,1CD0812|...  
Web1982Web1982Web1982Web1982Web1982Web1982W... +

-----

O sea, que esta es la forma en que encripta la cadena y si vamos al Dump a la parte donde la ha encriptado tenemos esto:

001758DC	52	01	1C	20	1C	20	5F	00	6B	00	6E	00	62	00	21	20	Œ""_knb‡
001758EC	E1	00	22	20	69	00	69	00	6E	00	78	00	53	01	53	01	á•iinxxœœ
001758FC	61	01	5D	00	6A	00	7B	00	73	00	DC	02	3A	20	22	20	šlj{s~>•
0017590C	68	00	B5	00	7A	00	69	00	DC	02	22	21	1C	20	72	00	huzi~™`r
0017591C	7D	00	6A	00	5E	00	C6	02	A8	00	A6	00	61	00	71	00	}j^`" aq
0017592C	69	00	64	00	D3	00	22	20	22	20	60	00	6B	00	69	00	idŒ••`ki
0017593C	61	00	21	20	53	01	DE	00	72	00	7C	00	60	01	26	20	a‡œPr š...
0017594C	B0	00	B8	00	1A	20	5E	00	59	00	6E	00	62	00	39	20	°,^Ynb<
0017595C	22	20	19	20	61	00	69	00	B4	00	62	00	8D	00	1D	20	•'ai`b "
0017596C	19	20	63	00	68	00	69	00	67	00	D3	00	B3	00	D7	00	'chigŒ³x
0017597C	9D	00	A5	00	B4	00	73	00	A4	00	A9	00	1A	20	26	20	Ÿ'su©,...
0017598C	AE	00	AA	00	3A	20	C6	00	D3	00	60	01	A5	00	A6	00	®a>ŒŒšŸ
0017599C	61	00	52	00	8D	00	22	21	1A	20	7E	00	A8	00	61	01	aR™,~"š
001759AC	3A	20	C3	00	CA	00	1A	20	26	20	7E	01	3A	20	61	01	>ĂÊ,...ž>š
001759BC	C5	00	D4	00	CE	00	A0	00	A0	00	B1	00	52	00	A4	00	ĂŒî ±Rœ
001759CC	B0	00	8F	00	64	00	71	00	00	00							° dq.

Es cierto, no hay quien entienda nada jajaja.  
Observemos algo, si sumamos los dos primeros caracteres, o sea, el 5 con la W nos da igual a 8C y esto pasado a decimal nos da 140. Podemos ver que hay algunos valores que no nos cuadran como el primero que en vez de 8C pone 52, el segundo que pone 1C y el tercero que también pone 1C pero sin embargo si cogemos el cuarto que pone 5F y lo pasamos a decimal obtendremos el cuarto valor del archivo kiosco.lic y así con muchos otros. ¿Por qué hay algunos que no cuadran? Pues no se pero si hago la operación a mano veo que lo que está en kiosco.lic cuadra con mis cuentas.  
Una vez salimos de la zona de encriptamiento y, si traceamos un poco mas con F8, llegamos aquí:

00425EDC

• 51

push

ecx

00425EDF

• 52

push

edx

00425EE1

• FF15 E0114000

call

ds:[4011E0]

00425EE7

• 50

push

eax

00425EE8

• FF15 4C104000

call

ds:[40104C]

00425EEE

• 50

push

eax

00425EEF

• FF15 08104000

call

ds:[401008]

00425EF5

• 8BD0

mov

edx, eax

00425EF7

• 8D4D D8

lea

ecx, [ebp-28]

A continuacion convierte el valor obtenido de la suma en su valor hexa

MSVBVM60.\_\_vbaStrVarVal

[Arg1

rtcAnsiValueBstr

A continuacion convierte el resultado a su valor decimal en ascii

MSVBVM60.\_\_vbaStrI2

Aquí lo que hace es que va convirtiendo cada resultado de la suma en su equivalente en decimal y lo convierte en una cadena ASCII.  
Una vez salimos del bucle, ya tenemos toda la cadena de valores concatenada y lista para guardar en el archivo y si seguimos traceando con F8 llegamos aquí:

00425FA3	. 50	push	eax	
00425FA4	. 68 20084100	push	410820	UNICODE "\\kiosco.lic"
00425FA9	. FFD7	call	edi	
00425FAB	. 8BD0	mov	edx, eax	
00425FAD	. 8D4D D8	lea	ecx, ss:[ebp-28]	
00425FB0	. FFD6	call	esi	
00425FB2	. 50	push	eax	[ Arg4 Arg3 = 00000001 Arg2 = FFFFFFFF Arg1 = 00000002 __vbaFileOpen
00425FB3	. 6A 01	push	1	
00425FB5	. 6A FF	push	-1	
00425FB7	. 6A 02	push	2	
00425FB9	. FF15 08124000	call	ds:[401208]	
00425FBF	. 8D4D D8	lea	ecx, ss:[ebp-28]	
00425FC2	. 8D55 DC	lea	edx, ss:[ebp-24]	
00425FC5	. 51	push	ecx	
00425FC6	. 52	push	edx	
00425FC7	. 6A 02	push	2	
00425FC9	. FF15 38124000	call	ds:[401238]	MSVBVM60.__vbaFreeStrList
00425FCF	. 83C4 0C	add	esp, 0C	
00425FD2	. 8D4D B8	lea	ecx, ss:[ebp-48]	
00425FD5	. FF15 EC124000	call	ds:[4012EC]	MSVBVM60.__vbaFreeObj
00425FDB	. 8B45 E4	mov	eax, ss:[ebp-1C]	
00425FDE	. 50	push	eax	
00425FDF	. 6A 01	push	1	
00425FE1	. 68 3C084100	push	41083C	En el siguiente call mete el contenido a kiosco.lic
00425FE6	. FF15 A0114000	call	ds:[4011A0]	MSVBVM60.__vbaPrintFile
00425FEC	. 83C4 0C	add	esp, 0C	

Podemos ver como crea el archivo kiosco.lic y luego guarda la serie de números obtenida en el archivo. Pues bien, ya tenemos todo lo necesario para intentar descryptar el archivo kiosco.lic. Así me quedó el código en C:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    unsigned char buffer[1024];
    char cadenaDesencriptar[] = "Web1982";
    int contador = 0,posicion = 0;
    FILE *origen;
    int valor;

    if((origen = fopen("kiosco.lic","r")) == NULL)
    {
        printf("No se pudo leer el archivo kiosco.lic, si no lo hizo ya, siga estos pasos:\n\n"
            "1- Ejecute kiosco y de en Heramientas->Crear licencia.\n"
            "2- Copie el keygen en el directorio donde se creo el archivo que sera el de\n"
            "   instalacion del programa.\n"
            "3- Vuelva a probar\n\n");
    }
    else
    {
        do
        {
            if(fscanf(origen,"%i",&valor) != EOF)
            {
```

```

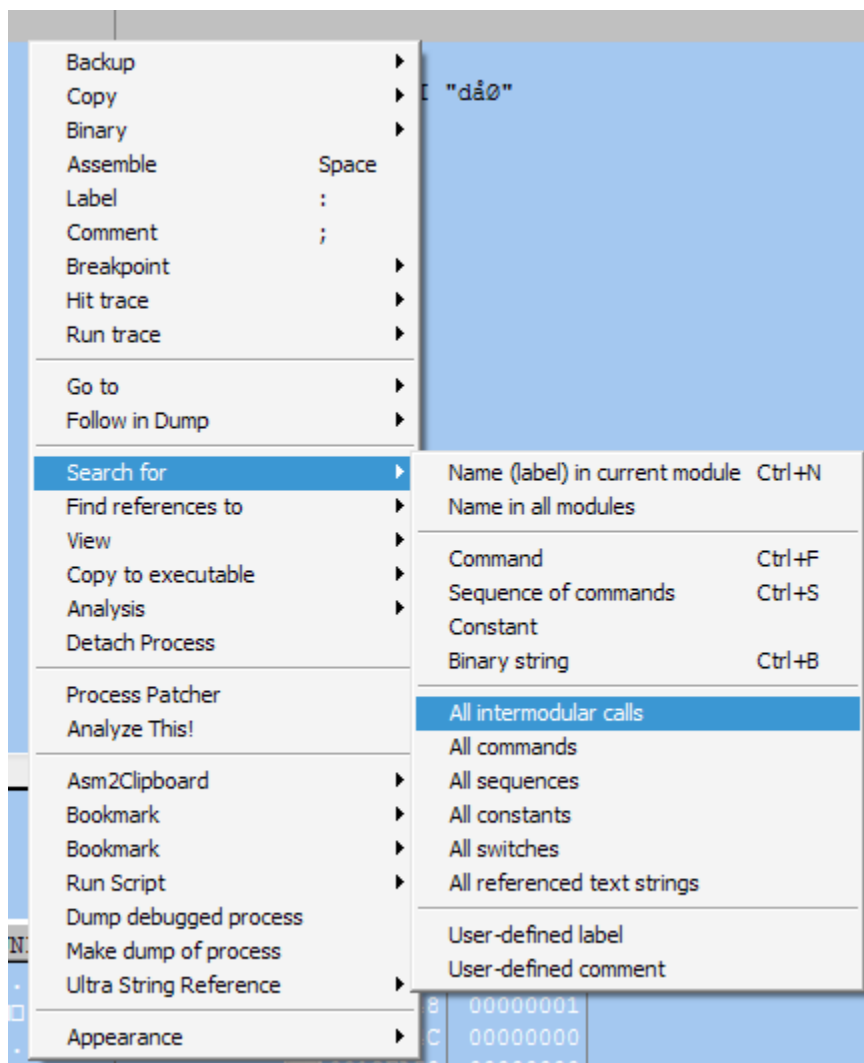
    valor -= (unsigned int) cadenaDesencriptar[contador];
    buffer[posicion] = valor;
    fseek(origen,1,SEEK_CUR);
    if(contador == 6)
        contador = 0;
    else
        contador++;
    posicion++;
}
}while(!feof(origen));
buffer[posicion] = '\0';
printf("Cadena obtenida del archivo kiosco.lic:\n\n%s",buffer);
fclose (origen);
}

```

Con eso ya podemos obtener el contenido desencriptado de kiosco.lic.

Ahora nos falta saber que hace el proveedor con ese archivo y que tipo de archivo nos manda al pagar. Podría ser un .reg que al ejecutarlo metiese entradas en el registro y que el programa al iniciarse compruebe esas entradas o bien podría ser un archivo el cual lea el programa al iniciarse o podría ser algún otro método.

Decido reiniciar y poner BPs a RegOpenKeyA, RegQueryValueExA y a \_\_vbaFileOpen y dar a F9 y me encuentro con que no para en ningún sitio interesante hasta que para en el HBP donde comprobaba si estaba registrado así que algo se me escapa. Entonces se me ocurre buscar todas las calls que llaman a \_\_vbaFileOpen y mirar que hay por ahí cerquita:



Escribo `__vbaFileOpen` y por último doy clic en la columna Destination para ordenarlo de ese modo y que me muestre todas las llamadas a esa función:



Found intermodular calls			
Address	Disassembly		Destination
0047284D	call	ds:[40109C]	MSVBVM60.__vbaExitProc
0047E3A8	call	ds:[40109C]	MSVBVM60.__vbaExitProc
00481DC2	call	ds:[40109C]	MSVBVM60.__vbaExitProc
00422AAE	call	ds:[401104]	MSVBVM60.__vbaFileClose
00425FF1	call	ds:[401104]	MSVBVM60.__vbaFileClose
00428CE3	call	ds:[401104]	MSVBVM60.__vbaFileClose
00481989	call	ds:[401104]	MSVBVM60.__vbaFileClose
00481D4A	call	ds:[401104]	MSVBVM60.__vbaFileClose
00422A82	call	ds:[401208]	MSVBVM60.__vbaFileOpen
00425FB9	call	ds:[401208]	MSVBVM60.__vbaFileOpen
0042883D	call	ds:[401208]	MSVBVM60.__vbaFileOpen
00481882	call	ds:[401208]	MSVBVM60.__vbaFileOpen
00481CE6	call	ds:[401208]	MSVBVM60.__vbaFileOpen
00480B97	call	ds:[401150]	MSVBVM60.__vbaFixstrConstruct
0044489D	call	ds:[401044]	MSVBVM60.__vbaFpCDblR8
00445BAF	call	ds:[40128C]	MSVBVM60.__vbaFpI2
00458F73	call	ds:[40128C]	MSVBVM60.__vbaFpI2

Pues nos tocará mirar una a una así que damos doble clic en la primera entrada y vemos esto:

00422A6E	. FF15 60104000	call	ds:[401060]	__vbaStrCat
00422A74	. 8BD0	mov	edx, eax	
00422A76	. 8D4D E0	lea	ecx, ss:[ebp-20]	
00422A79	. FFD6	call	esi	
00422A7B	. 50	push	eax	Arg4
00422A7C	. 6A 01	push	1	Arg3 = 00000001
00422A7E	. 6A FF	push	-1	Arg2 = FFFFFFFF
00422A80	. 6A 01	push	1	Abre el archivo kiosco_cert.lic
00422A82	. FF15 08124000	call	ds:[401208]	__vbaFileOpen
00422A88	. 8D45 E0	lea	eax, ss:[ebp-20]	
00422A8B	. 8D4D E4	lea	ecx, ss:[ebp-1C]	
00422A8E	. 50	push	eax	
00422A8F	. 51	push	ecx	
00422A90	. 6A 02	push	2	
00422A92	. FFD3	call	ebx	
00422A94	. 83C4 0C	add	esp, 0C	
00422A97	. 8D4D D8	lea	ecx, ss:[ebp-28]	
00422A9A	. FF15 EC124000	call	ds:[4012EC]	MSVBVM60.__vbaFreeObj
00422AA0	. 8D55 E8	lea	edx, ss:[ebp-18]	
00422AA3	. 6A 01	push	1	Arg2 = 00000001
00422AA5	. 52	push	edx	Lee una linea del archivo
00422AA6	. FF15 28104000	call	ds:[401028]	__vbaLineInputStr
00422AAC	. 6A 01	push	1	

Si miramos mas arriba vemos un salto largo que salta esta zona y un poco mas arriba vemos un call que no es de VB:

004229D4	. 50	push	eax	
004229D5	. E8 16730500	call	00479CF0	kiosco.00479CF0
004229DA	. 8B1D 38124000	mov	ebx, ds:[401238]	MSVBVM60.__vbaFreeStrList
004229E0	. 8D4D E0	lea	ecx, ss:[ebp-20]	
004229E3	. 8D55 E4	lea	edx, ss:[ebp-1C]	
004229E6	. 51	push	ecx	
004229E7	. 52	push	edx	
004229E8	. 6A 02	push	2	
004229EA	. 8BF8	mov	edi, eax	
004229EC	. FFD3	call	ebx	<MSVBVM60.__vbaFreeStrList>
004229EE	. 83C4 0C	add	esp, 0C	
004229F1	. 8D4D D8	lea	ecx, ss:[ebp-28]	
004229F4	. FF15 EC124000	call	ds:[4012EC]	MSVBVM60.__vbaFreeObj
004229FA	. 66:85FF	test	di, di	Comprueba que kiosco_cert.lic exista
004229FD	. 0F84 CA010000	je	00422BCD	kiosco.00422BCD
00422A03	. A1 788B4800	mov	eax, ds:[488B78]	
00422A08	. 85C0	test	eax, eax	
00422A0A	. 75 10	jnz	short 00422A1C	kiosco.00422A1C
00422A0C	. 68 788B4800	push	488B78	[Arg2 = 00488B78 ASCII "dã0"

Así que ponemos un BP en ese call, reiniciamos y damos a F9 para ver que pasa y al parar vemos esto:

Registers (FPU)		<	<	<	<	<	<	<
EAX	00151794	UNICODE "C:\Archivos de programa\Kiosco3\kiosco_cert.lic"						
ECX	0012FAF4							
EDX	00151794	UNICODE "C:\Archivos de programa\Kiosco3\kiosco_cert.lic"						
EBX	00000000							
ESP	0012FA3C							
EBP	0012FB14							
ESI	73486A74	MSVBVM60.__vbaStrMove						
EDI	7347A0C0	MSVBVM60.__vbaHresultCheckObj						

Entramos con F7 y seguimos traceando con F8 y llegamos aquí:

00479D50	. C745 FC 0300	mov	[local.1], 3	
00479D57	. 8B45 DC	mov	eax, [local.9]	
00479D58	. 50	push	eax	
00479D5B	. FF15 58124000	call	ds:[401258]	[Usa la siguiente funcion para comprobar si existe el archivo de licencia en vez de usar __vbaopenfile rtcGetFileAttr
00479D61	. 24 EF	and	al, 0EF	
00479D63	. F7D8	neg	eax	
00479D65	. 1BC0	sbb	eax, eax	
00479D67	. F7D8	neg	eax	
00479D69	. F7D8	neg	eax	
00479D6B	. 66:8945 D8	mov	ss:[ebp-28], ax	
00479D6F	. 68 7E9D4700	push	479D7E	
00479D74	. 8D4D DC	lea	ecx, [local.9]	
00479D77	. FF15 F0124000	call	ds:[4012F0]	MSVBVM60.__vbaFreeStr
00479D7E	. C3	retn		

La función rtcGetFileAttr es muy sospechosa y más aun si vemos que lo que ha pusheado es esto:

Registers (FPU)		<	<	<	<	<	<	<	<
EAX	0015E7B4	UNICODE "C:\Archivos de programa\Kiosco3\kiosco_cert.lic"							
ECX	0012FA14								
EDX	0000005E								
EBX	00000000								

Si intentamos continuar con F8 nos da un error de tipo "Inexact floating-point result" y si ponemos un BP en la siguiente línea después de la llamada a rtcGetFileAttr y pasamos la excepción con Shift + F9 podremos ver que nunca para en ese BP y el programa arranca con lo cual me da que pensar que el programa controla esa excepción.

Probaremos a crear un archivo con ese nombre en el directorio de instalación con cualquier contenido, en mi caso pondré “Cracked by Aguml”. Luego reiniciamos y damos a F9 y, cuando pare en el call a rtcGetFileAttr, lo pasamos con F8 y veremos que ya no da excepción y pasamos sin problemas. Seguimos con F8 hasta salir y llegar al salto que nos tiraba fuera de la zona donde abría y leía el archivo y vemos que ya no salta:

```
004229F1 . 8D4D D8      lea     ecx, ss:[ebp-28]
004229F4 . FF15 EC124000 call    ds:[4012EC]
004229FA . 66:85FF      test    di, di
004229FD . 0F84 CA010000 je      00422BCD
00422A03 . A1 788B4800 mov     eax, ds:[488B78]
00422A08 . 85C0        test    eax, eax
```

Registers (FPU)

< < < < < < < <

EAX 00000000  
ECX 00D8E5C4  
EDX 001516B0  
EBX 73486A45 MSVBVM60.\_\_vbaFreeStrList  
ESP 0012FA40  
EBP 0012FB14  
ESI 73486A74 MSVBVM60.\_\_vbaStrMove  
EDI 0000FFFF  
EIP 004229FD kiosco.004229FD

Seguimos traceando con F8 y llegamos al \_\_vbaFileOpen:

```
00422A79 . FFD6        call    esi
00422A7B . 50          push   eax
00422A7C . 6A 01       push   1
00422A7E . 6A FF       push   -1
00422A80 . 6A 01       push   1
00422A82 . FF15 08124000 call    ds:[401208]
00422A88 . 8D45 E0     lea     eax, ss:[ebp-20]
```

Arg4  
Arg3 = 00000001  
Arg2 = FFFFFFFF  
Abre el archivo kiosco\_cert.lic  
\_\_vbaFileOpen

Registers (FPU)

< < < < < < < <

EAX 00151794 UNICODE "C:\Archivos de programa\Kiosco3\kiosco\_cert.lic"  
ECX 0012FAF4  
EDX 00151794 UNICODE "C:\Archivos de programa\Kiosco3\kiosco\_cert.lic"  
EBX 73486A45 MSVBVM60.\_\_vbaFreeStrList  
ESP 0012FA30  
EBP 0012FB14  
ESI 73486A74 MSVBVM60.\_\_vbaStrMove  
EDI 00D950CC

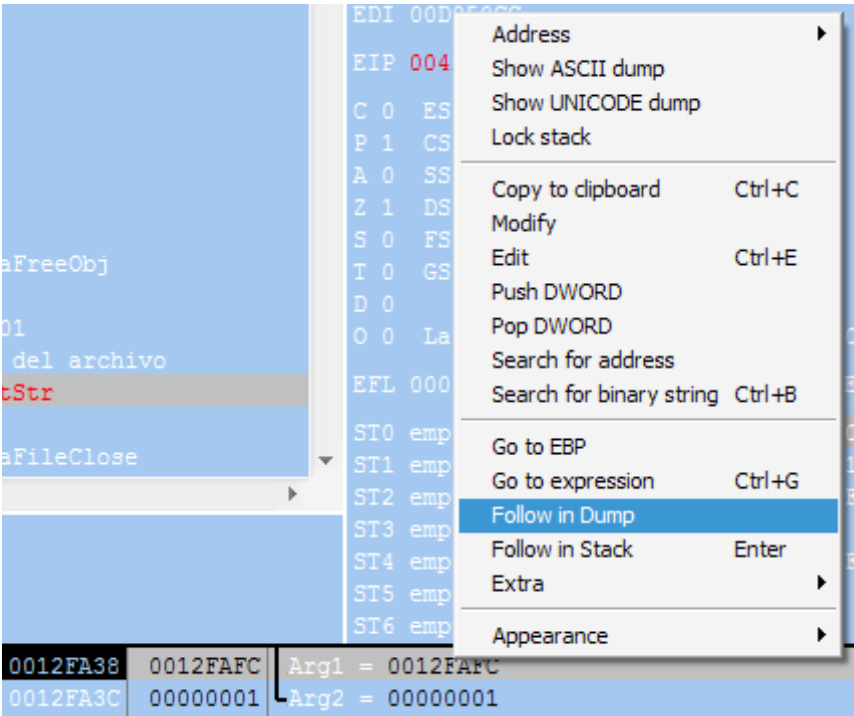
Seguimos traceando con F8 y llegamos aquí:

```
00422AA0 . 8D55 E8     lea     edx, ss:[ebp-18]
00422AA3 . 6A 01       push   1
00422AA5 . 52         push   edx
00422AA6 . FF15 28104000 call    ds:[401028]
00422AAC . 6A 01       push   1
00422AAE . FF15 04114000 call    ds:[401104]
```

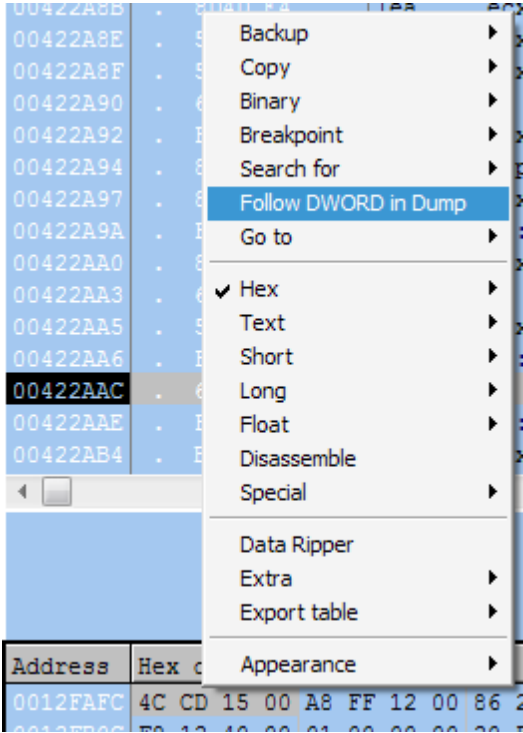
MSVBVM60.\_\_vbaFileClose  
Arg2 = 00000001  
Lee una línea del archivo  
\_\_vbaLineInputStr  
MSVBVM60.\_\_vbaFileClose

Arg1 = 0012FAFC  
Arg2 = 00000001

Hacemos clic derecho sobre el argumento 1 que es donde se guardará el puntero a la línea leída para ver que lee:



Lo pasamos con F8 y seleccionamos el dword en el dump, damos clic derecho encima y seleccionamos “Follow DWORD in Dump”:



Address	Hex dump	UNICODE
0015CD4C	43 00 72 00 61 00 63 00 6B 00 65 00 64 00 20 00	Cracked
0015CD5C	62 00 79 00 20 00 41 00 67 00 75 00 6D 00 6C 00	by Aguml
0015CD6C	00 00 5F 63 65 72 74 2E 6C 69 63 00 06 00 07 00	.0000c000

Ya leyó el contenido de nuestro archivo y si seguimos con F8 vemos esto:

- [CPU - main thread, module kiosco]

File View Debug Plugins Options Window Help UnPackers Identificadores Decompiladores Desensambladores

Paused

Address	Disassembly	Comment
00422AAE	call ds:[401104]	MSVBVM60.__vbaFileClose
00422AB4	mov edx, 40EF80	UNICODE "S1o7f8t2"
00422AB9	lea ecx, ss:[ebp-1C]	
00422ABC	call ds:[40122C]	MSVBVM60.__vbaStrCopy
00422AC2	lea eax, ss:[ebp-1C]	
00422AC5	lea ecx, ss:[ebp-18]	
00422AC8	push eax	
00422AC9	push ecx	
00422ACA	call 0047A290	kiosco.0047A290
00422ACF	mov edx, eax	
00422AD1	lea ecx, ss:[ebp-24]	
00422AD4	call esi	
00422AD6	push eax	
00422AD7	call 00480A90	kiosco.00480A90
00422ADC	mov edx, eax	
00422ADE	lea ecx, ss:[ebp-20]	
00422AE1	call esi	
00422AE3	push eax	
00422AE4	push 0	
00422AE6	call ds:[40127C]	MSVBVM60.__vbaStrComp
00422AEC	mov si, ax	
00422AEF	lea edx, ss:[ebp-24]	
00422AF2	neg si	

0040EF80=kiosco.0040EF80 (UNICODE "S1o7f8t2")  
edx=00000000

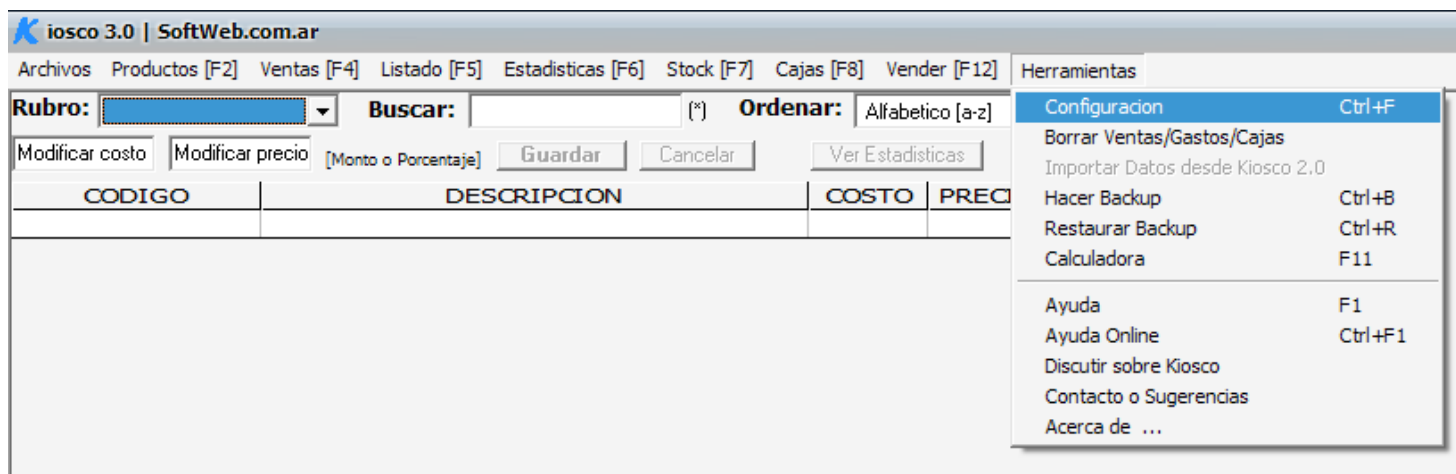
Parece una zona muy interesante porque usa la cadena "S1o7f8t2" para algo, luego entra en dos calls que no son de VB y despues vemos una llamada a \_\_vbaStrComp. Pues bien, sigamos traceando con F8 hasta llegar a la comparacion (podriamos poner un BP y dar a F9 pero yo prefiero ir con F8 para ir viendo un poco por encima lo que hace y cuando intento pasar el call de 422ACA da una excepcion y si damos a F9 seguimos en la linea siguiente donde seguiremos con F8 hasta que llegamos a la comparacion y debajo veremos esto:

00422AE3	. 50	push	eax	
00422AE4	. 6A 00	push	0	
00422AE6	. FF15 7C124000	call	ds:[40127C]	MSVBVM60.__vbaStrComp
00422AEC	. 66:8BF0	mov	si, ax	
00422AEF	. 8D55 DC	lea	edx, ss:[ebp-24]	
00422AF2	. 66:F7DE	neg	si	
00422AF5	. 8D45 E0	lea	eax, ss:[ebp-20]	
00422AF8	. 52	push	edx	
00422AF9	. 1BF6	sbb	esi, esi	
00422AFB	. 8D4D E4	lea	ecx, ss:[ebp-1C]	
00422AFE	. 50	push	eax	
00422AFF	. 46	inc	esi	
00422B00	. 51	push	ecx	
00422B01	. 6A 03	push	3	
00422B03	. F7DE	neg	esi	
00422B05	. FFD3	call	ebx	
00422B07	. 83C4 10	add	esp, 10	
00422B0A	. 66:85F6	test	si, si	
00422B0D	. 0F84 BA000000	je	00422BCD	Es el salto importante que comprueba que el archivo de registro es correcto
00422B13	. 8B5D 08	mov	ebx, ss:[ebp+8]	
00422B16	. 66:C705 4A80	mov	word ptr ds:[48804A], 0FFFF	
00422B1F	. 53	push	ebx	

Aquí hay una zona muy caliente puesto que tenemos la comparación donde una de las cadenas a comparar es en mi caso:

"5.1.2600|3806FE78,1CAA637|B7A41AD2,1CD0812|03/21/07|ACRSYS - 6040000|06/02/15|Null|AMD Turion(tm) 64"

Y si seguimos vemos un salto que evita que ejecute el MOV de la línea 422B16 el cual mete el valor bueno en 48804A para que cuando lleguemos a la comparación aparezcamos como registrados. Pues probaremos ir con F8 hasta llegar al salto y evitar que salte cambiando el flag Z y luego damos a F9 todas las veces que haga falta hasta que arranque el programa y veremos esto al dar en "Herramientas":



Pues con crear un archivo llamado "kiosco\_cert.lic" con cualquier contenido y cambiando este salto ya estaríamos registrados pero intentaremos crear un keygen que siempre es la mejor solución a mi entender. Así que pondremos un BP en el primero de los dos calls que no eran de VB y que estaban encima de la comparación, o sea, 422ACA, para ver que hace y el porque de dicha excepción al pasar por ella. Reiniciamos y llegamos hasta dicho BP usando F9 y entramos con F7.

0047A2E0	. 51	push	ecx	Mide el largo de la cadena obtenida de nuestro archivo
0047A2F1	. FF15 30104000	call	ds:[401030]	MSVBVM60.__vbaLenBstr



```
Registers (FPU)
EAX 0012FAFC
ECX 0015CD4C UNICODE "Cracked by Agum1"
EDX 00000010
```

Seguimos traceando y veo que esta función es un calco de la que encriptaba nuestra cadena usando “Web1982” para ello pero en este caso lo que hace es usar "S1o7f8t2" y en vez de sumar lo que hace es restar, pero tenemos un problema, al llegar a:

```
0047A4AD . FF15 C4114000 call ds:[4011C4] ; \rtcVarBstrFromAnsi
```

Algo falla y se produce una excepción. Según creo, lo que he podido entender que hace esta función (no he encontrado mucha información al respecto por la web) es convertir de Unicode a Ansi y en mi caso guardé el archivo como Ansi así que podría ser por lo que da excepción así que deshabilito todos los BPs, pongo un BP en esa función, reinicio, abro el archivo con el bloc de notas, lo guardo como Unicode y vuelvo a lanzar con F9 a ver que pasa cuando intente pasarla con F8 y... ya no da excepción así que doy Ctrol + F9 para salir de la función y sigo con F8 hasta que llego a la comparación donde debe comparar la cadena buena con la que acaba de obtener del archivo después de haberle pasado la función de encriptación y, como no se que es lo que usa para comparar esa función, entro con F8 y veo que compara la cadena de mi PC con la obtenida y que está aquí:

```
0015EBC4 AC 00 CD 00 43 00 3B 00 61 00 2B 00 6B 00 33 00 -íC;a+k3
0015EBD4 11 00 20 00 62 00 42 00 20 00 09 00 67 00 43 00 bB .gC
0015EBE4 1A 00 3B 00 00 00 ; .
```

Pues bien, ¿Qué pasaría si cogiésemos la cadena de nuestro PC y la pasáramos por esta función pero si en vez de sumar le restáramos los caracteres? Vamos a probar y para ello me creé este código:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    unsigned char buffer[1024];
    char cadenaDesencriptar[] = "Web1982";
    int contador = 0, posicion = 0;
    FILE *origen;
    int valor;

    char mascara[] = "S1o7f8t2";
    int largo, contador2 = 0;
    FILE *licencia;

    int leído = 0;

    //Parte en la que obtenemos la cadena desde el archivo kiosco.lic y la desencriptamos

    if((origen = fopen("kiosco.lic", "r")) == NULL)
    {
        printf("No se pudo leer el archivo kiosco.lic, si no lo hizo ya, siga estos pasos:\n\n")
    }
```

```

    "1- Ejecute kiosco y de en Heramientas->Crear licencia.\n"
    "2- Copie el keygen en el directorio donde se creo el archivo que sera el de\n"
    "  instalacion del programa.\n"
    "3- Vuelva a probar\n\n");
}
else
{
    do
    {

        if(fscanf(origen,"%i",&valor) != EOF)
        {
            valor -= (unsigned int) cadenaDesencriptar[contador];
            buffer[posicion] = valor;
            fseek(origen,1,SEEK_CUR);
            if(contador == 6)
                contador = 0;
            else
                contador++;
            posicion++;
        }
    }while(!feof(origen));
    buffer[posicion] = '\0';
    printf("Cadena obtenida del archivo kiosco.lic:\n\n%s",buffer);
    fclose (origen);
    leido=1; //Ponemos a 1 esta variable para indicar que se pudo obtener la cadena
}

//Terminamos la desencriptacion de la cadena

//Parte en la que encriptamos la cadena obtenida antes para crear el archivo de licencia final

contador = 0;
largo = posicion;
if(leido == 0) //Comprobamos que se pudo leer el archivo kiosco.lic
{
    printf("No se pudo crear el archivo de licencia porque no se pudo obtener la cadena\n"
        "desde kiosco.lic.\n\n");
}
else if((licencia = fopen("kiosco_cert.lic","w")) == NULL)
{
    printf("No se pudo crear el archivo de licencia.\n\n");
}
else
{
    printf("\n\n\nCadena encriptada:\n\n");

    //Mientras que no lleguemos al final de la cadena obtenida de kiosco.lic
    while(contador < largo)

```

```

{
    if((buffer[contador] - mascara[contador2] < 0) && buffer[contador] + mascara[contador2] > 127)
    {
        //Si el resultado de la resta no es un carácter imprimible estándar guardamos el original
        fputc(buffer[contador],licencia);
        printf("%c",buffer[contador]);
    }
    else
    {
        //Si el resultado de la resta es un carácter estándar guardamos la suma de ambos
        fputc((buffer[contador] + (mascara[contador2])),licencia);
        printf("%c",buffer[contador]+ mascara[contador2]);
    }
    contador++; //Incrementamos el contador que lee del buffer
    contador2++; //Incrementamos el contador de "S1o7f8t2"
    //Si hemos llegado al final de la cadena
    if(contador2 > 7)
        //Ponemos a 0 el contador que usamos para saber que carácter usar de la cadena
        //encriptadora
        contador2=0;
}
printf("\n\nEl archivo de licencia se creo satisfactoriamente, pruebe el programa para ver\n"
       "si ha surtido efecto\n\n");
fclose(licencia);
}
system("PAUSE");
return 0;
}

```

Este código lo que hace es leer el contenido de kiosco.lic y lo desencripta para obtener la cadena y luego lo encripta de la forma que hemos visto que hace la aplicación guardando el resultado en kiosco\_cert.lic y el resultado es este:

**Cadena obtenida del archivo kiosco.lic:**

**5.1.2600|3806FE78,1CAA637|B7A41AD2,1CD0812|03/21/07|ACRSYS - 6040000|06/02/15|Nu  
ll|AMD Turion(tm) 64 Mobile Technology MK-38**

**Cadena encriptada:**

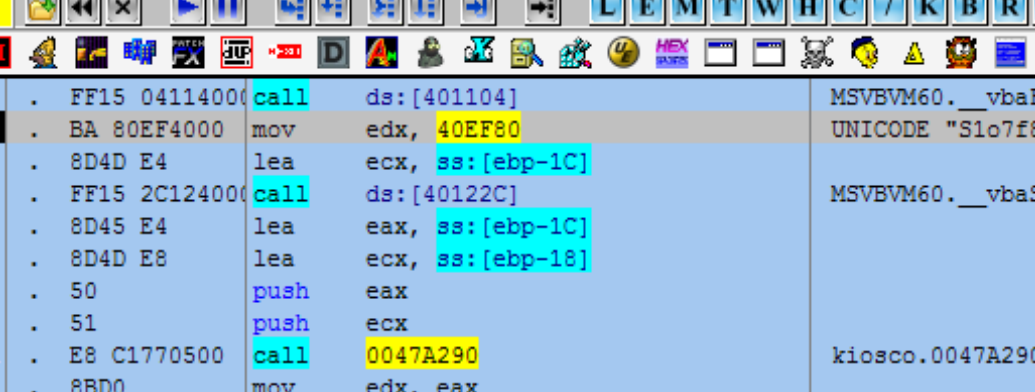
**5\_1e2n0bϣd8g6~Ei8|1zAy6e7;BnAl1sDc,hC|0j1cÛg3g2c/a7 | A{Rà¼ä d n0f0a0gÔh6a0c/h5-| N°  
¬ ØÛxM| à ℒúíªÈ`Pƒ|Q6k àoö⌐ ØeWTØcÛ⊥álª=⌘△K^3o**

**El archivo de licencia se creo satisfactoriamente, pruebe el programa para ver  
si ha surtido efecto**

**Presione una tecla para continuar . . .**

Y si ejecutamos el programa... ¡¡¡Funciona!!!

---



The screenshot shows the OllyDbg interface with the CPU window displaying assembly code. The code is for a function named `0040EF80`, which is a call to `MSVBVM60.__vbaFileClose`. The code is as follows:

```

0040EF80 . FF15 04114000 call ds:[401104]
0040EF85 . BA 80EF4000 mov     edx, 40EF80
0040EF90 . 8D4D E4      lea     ecx, ss:[ebp-1C]
0040EF95 . FF15 2C124000 call    ds:[40122C]
0040EF9A . 8D45 E4      lea     eax, ss:[ebp-1C]
0040EFA0 . 8D4D E8      lea     ecx, ss:[ebp-18]
0040EFA5 . 50          push    eax
0040EFAA . 51          push    ecx
0040EFAD . E8 C1770500 call    0047A290
0040EFAF . 8BD0        mov     edx, eax
0040EFB2 . 8D4D DC      lea     ecx, ss:[ebp-24]
0040EFB7 . FFD6        call    esi
0040EFBC . 50          push    eax
0040EFC0 . E8 B4DF0500 call    00480A90
0040EFC5 . 8BD0        mov     edx, eax
0040EFC8 . 8D4D E0      lea     ecx, ss:[ebp-20]
0040EFD3 . FFD6        call    esi
0040EFD8 . 50          push    eax
0040EFD9 . 6A 00        push    0
0040EFD9 . FF15 7C124000 call    ds:[40127C]
0040EFD9 . 66:8BF0     mov     si, ax
0040EFD9 . 8D55 DC      lea     edx, ss:[ebp-24]
0040EFD9 . 66:F7DE     neg     si

```

The registers window shows the following values:

```

0040EF80=kiosco.0040EF80 (UNICODE "S1o7f8t2")
edx=00000000

```

Así que solo nos queda por ver que pasa en el segundo call que es el que está en 00422AD7 así que deshabilitamos todos los BPs y ponemos uno en esa línea, damos a reiniciar y a F9, entramos en ella con F7 y seguimos con F8 hasta que encontremos un call que no sea de sistema:

00480C10	. C785 D4FEFFFF	mov	dword ptr ss:[ebp-12C], 94	
00480C1A	. FF15 60114000	call	ds:[401160]	__vbaRecUniToAnsi
00480C20	. 50	push	eax	
00480C21	. E8 D2E9F8FF	call	0040F5F8	Aqui obtiene que es Service Pack 3
00480C26	. FF15 78104000	call	ds:[401078]	MSVBVM60.__vbaSetSystemError
00480C2C	. 8D85 C4FAFFFF	lea	eax, ss:[ebp-53C]	

Entraremos en el para ver que hace y como la función es muy pequeña la traceo con F8 y veo esto:

0040F5F8	\$ A1 348B4800	mov	eax, ds:[488B34]	
0040F5FD	. 0BC0	or	eax, eax	
0040F5FF	. 74 02	je	short 0040F603	kiosco.0040F603
0040F601	. FFE0	jmp	eax	
0040F603	> 68 E0F54000	push	40F5E0	
0040F608	. B8 102C4000	mov	eax, 402C10	
0040F60D	. FFD0	call	eax	<jmp.&MSVBVM60.DllFunctionCall>
0040F60F	.- FFE0	jmp	eax	kernel32.GetVersionExA
0040F611	. 00	db	00	

Me empecé a preguntar para que se usa GetVersionEx así que hice uso de Google y en el siguiente enlace se puede ver que se usa para obtener la versión de Windows y creo que por ahí va muy bien:

<http://msdn.microsoft.com/en-us/library/windows/desktop/ms724451%28v=vs.85%29.aspx>

Si miramos en el enlace encontraremos este código de ejemplo que nos viene que ni pintado:

```
void main()
{
    DWORD dwVersion = 0;
    DWORD dwMajorVersion = 0;
    DWORD dwMinorVersion = 0;
    DWORD dwBuild = 0;

    dwVersion = GetVersion();

    // Get the Windows version.

    dwMajorVersion = (DWORD)(LOBYTE(LOWORD(dwVersion)));
    dwMinorVersion = (DWORD)(HIBYTE(LOWORD(dwVersion)));

    // Get the build number.

    if (dwVersion < 0x80000000)
        dwBuild = (DWORD)(HIWORD(dwVersion));

    printf("Version is %d.%d.%d\n", dwMajorVersion, dwMinorVersion,
```

```
        dwBuild);  
    }
```

Y si miramos en el mismo enlace un poco mas abajo veremos este otro enlace a un código de ejemplo para obtener la versión de nuestro S.O. pero de una forma mas detallada:

<http://msdn.microsoft.com/en-us/library/windows/desktop/ms724429%28v=vs.85%29.aspx>

Pruebo a modificar el código del segundo enlace para que me funcione en C y así quedó:

```
#include <stdio.h>  
#include <stdlib.h>  
#include <tchar.h>  
#include <windows.h>  
  
#define BUFSIZE 256  
  
typedef void (WINAPI *PGNSI)(LPSYSTEM_INFO);  
typedef BOOL (WINAPI *PGPI)(DWORD, DWORD, DWORD, DWORD, PDWORD);  
  
BOOL GetOSDisplayString( LPTSTR pszOS);  
  
BOOL GetOSDisplayString( LPTSTR pszOS)  
{  
    OSVERSIONINFOEX osv;  
    SYSTEM_INFO si;  
    PGNSI pGNSI;  
    PGPI pGPI;  
    BOOL bOsVersionInfoEx;  
    DWORD dwType;  
  
    ZeroMemory(&si, sizeof(SYSTEM_INFO));  
    ZeroMemory(&osv, sizeof(OSVERSIONINFOEX));  
  
    osv.dwOSVersionInfoSize = sizeof(OSVERSIONINFOEX);  
    bOsVersionInfoEx = GetVersionEx((OSVERSIONINFO*) &osv);  
  
    if(bOsVersionInfoEx != 1 ) return 1;  
  
    // Call GetNativeSystemInfo if supported or GetSystemInfo otherwise.  
  
    pGNSI = (PGNSI) GetProcAddress(  
        GetModuleHandle(TEXT("kernel32.dll")), "GetNativeSystemInfo");  
    if(NULL != pGNSI)  
        pGNSI(&si);  
    else  
        GetSystemInfo(&si);  
  
    if( VER_PLATFORM_WIN32_NT==osv.dwPlatformId &&  
        osv.dwMajorVersion > 4 )
```



```

{
    strcpy(pszOS,"Microsoft ");

    // Test for the specific product.

    if ( osvi.dwMajorVersion == 6 )
    {
        if( osvi.dwMinorVersion == 0 )
        {
            if( osvi.wProductType == VER_NT_WORKSTATION )
                strcat(pszOS, "Windows Vista ");
            else
                strcat(pszOS, "Windows Server 2008 ");
        }

        if( osvi.dwMinorVersion == 1 )
        {
            if( osvi.wProductType == VER_NT_WORKSTATION )
                strcat(pszOS,"Windows 7 ");
            else strcat(pszOS,"Windows Server 2008 R2 " );
        }

        pGPI = (PGPI) GetProcAddress(GetModuleHandle(TEXT("kernel32.dll")), "GetProductInfo");

        pGPI( osvi.dwMajorVersion, osvi.dwMinorVersion, 0, 0, &dwType);

        switch( dwType )
        {
            case 0x00000001: //PRODUCT_ULTIMATE:
                strcat(pszOS,"Ultimate Edition");
                break;
            case 0x00000030: //PRODUCT_PROFESSIONAL:
                strcat(pszOS,"Professional");
                break;
            case 0x00000003: //PRODUCT_HOME_PREMIUM:
                strcat(pszOS,"Home Premium Edition");
                break;
            case 0x00000002: //PRODUCT_HOME_BASIC:
                strcat(pszOS,"Home Basic Edition");
                break;
            case 0x00000004: //PRODUCT_ENTERPRISE:
                strcat(pszOS,"Enterprise Edition");
                break;
            case 0x00000006: //PRODUCT_BUSINESS:
                strcat(pszOS,"Business Edition");
                break;
            case 0x0000000B: //PRODUCT_STARTER:
                strcat(pszOS,"Starter Edition");
                break;
            case 0x00000012: //PRODUCT_CLUSTER_SERVER:

```

```

        strcat(pszOS,"Cluster Server Edition");
        break;
    case 0x00000008: //PRODUCT_DATACENTER_SERVER:
        strcat(pszOS,"Datacenter Edition");
        break;
    case 0x0000000C: //PRODUCT_DATACENTER_SERVER_CORE:
        strcat(pszOS,"Datacenter Edition (core installation)");
        break;
    case 0x0000000A: //PRODUCT_ENTERPRISE_SERVER:
        strcat(pszOS,"Enterprise Edition");
        break;
    case 0x0000000E: //PRODUCT_ENTERPRISE_SERVER_CORE:
        strcat(pszOS,"Enterprise Edition (core installation)");
        break;
    case 0x0000000F: //PRODUCT_ENTERPRISE_SERVER_IA64:
        strcat(pszOS,"Enterprise Edition for Itanium-based Systems");
        break;
    case 0x00000009: //PRODUCT_SMALLBUSINESS_SERVER:
        strcat(pszOS,"Small Business Server");
        break;
    // El siguiente case lo comento porque al compilar no me reconocía esa constante
    // PRODUCT_SMALLBUSINESS_SERVER_PREMIUM y no se a que podría equivaler
    //case PRODUCT_SMALLBUSINESS_SERVER_PREMIUM:
    //    StringCchCat(pszOS, BUFSIZE, TEXT("Small Business Server Premium Edition" ));
    //    break;
    case 0x00000007: //PRODUCT_STANDARD_SERVER:
        strcat(pszOS,"Standard Edition");
        break;
    case 0x0000000D: //PRODUCT_STANDARD_SERVER_CORE:
        strcat(pszOS,"Standard Edition (core installation)");
        break;
    case 0x00000011: //PRODUCT_WEB_SERVER:
        strcat(pszOS,"Web Server Edition");
        break;
    }
}

if ( osvi.dwMajorVersion == 5 && osvi.dwMinorVersion == 2 )
{
    if( GetSystemMetrics(SM_SERVERR2) )
        strcat(pszOS, "Windows Server 2003 R2, ");
    else if ( osvi.wSuiteMask & VER_SUITE_STORAGE_SERVER )
        strcat(pszOS,"Windows Storage Server 2003");
    // La siguiente condicional la comento porque no me reconoce la constante
    // VER_SUITE_WH_SERVER y no se a que valor podría equivaler
    //else if ( osvi.wSuiteMask & VER_SUITE_WH_SERVER )
    //    strcat(pszOS,"Windows Home Server");
    else if( osvi.wProductType == VER_NT_WORKSTATION &&
        si.wProcessorArchitecture==PROCESSOR_ARCHITECTURE_AMD64)
    {

```

```

    strcat(pszOS,"Windows XP Professional x64 Edition");
}
else
    strcat(pszOS,"Windows Server 2003, ");

// Test for the server type.
if ( osvi.wProductType != VER_NT_WORKSTATION )
{
    if ( si.wProcessorArchitecture==PROCESSOR_ARCHITECTURE_IA64 )
    {
        if( osvi.wSuiteMask & VER_SUITE_DATACENTER )
            strcat(pszOS,"Datacenter Edition for Itanium-based Systems");
        else if( osvi.wSuiteMask & VER_SUITE_ENTERPRISE )
            strcat(pszOS,"Enterprise Edition for Itanium-based Systems");
    }

    else if ( si.wProcessorArchitecture==PROCESSOR_ARCHITECTURE_AMD64 )
    {
        if( osvi.wSuiteMask & VER_SUITE_DATACENTER )
            strcat(pszOS,"Datacenter x64 Edition");
        else if( osvi.wSuiteMask & VER_SUITE_ENTERPRISE )
            strcat(pszOS,"Enterprise x64 Edition");
        else strcat(pszOS,"Standard x64 Edition");
    }

    else
    {
        if ( osvi.wSuiteMask & VER_SUITE_COMPUTE_SERVER )
            strcat(pszOS,"Compute Cluster Edition");
        else if( osvi.wSuiteMask & VER_SUITE_DATACENTER )
            strcat(pszOS,"Datacenter Edition");
        else if( osvi.wSuiteMask & VER_SUITE_ENTERPRISE )
            strcat(pszOS,"Enterprise Edition");
        else if ( osvi.wSuiteMask & VER_SUITE_BLADE )
            strcat(pszOS,"Web Edition");
        else strcat(pszOS,"Standard Edition");
    }
}
}

if ( osvi.dwMajorVersion == 5 && osvi.dwMinorVersion == 1 )
{
    strcat(pszOS,"Windows XP ");

    if( osvi.wSuiteMask & VER_SUITE_PERSONAL )
        strcat(pszOS,"Home Edition");
    else
        strcat(pszOS,"Professional");
}

```

```

if ( osvi.dwMajorVersion == 5 && osvi.dwMinorVersion == 0 )
{
    strcat(pszOS,"Windows 2000 ");

    if ( osvi.wProductType == VER_NT_WORKSTATION )
    {
        strcat(pszOS,"Professional");
    }
    else
    {
        if( osvi.wSuiteMask & VER_SUITE_DATACENTER )
            strcat(pszOS,"Datacenter Server");
        else if( osvi.wSuiteMask & VER_SUITE_ENTERPRISE )
            strcat(pszOS,"Advanced Server");
        else
            strcat(pszOS,"Server");
    }
}

// Include service pack (if any) and build number.

if( _tcslen(osvi.szCSDVersion) > 0 )
{
    strcat(pszOS," ");
    strcat(pszOS,osvi.szCSDVersion);
}

sprintf(&pszOS[0]+strlen(pszOS)," (build %d)", (int)osvi.dwBuildNumber);

if( osvi.dwMajorVersion >= 6 )
{
    if ( si.wProcessorArchitecture==PROCESSOR_ARCHITECTURE_AMD64 )
        strcat(pszOS,", 64-bit");
    else if (si.wProcessorArchitecture==PROCESSOR_ARCHITECTURE_INTEL )
        strcat(pszOS,", 32-bit");
}
return TRUE;
}
else
{
    printf( "This sample does not support this version of Windows.\n");
    return FALSE;
}
}

int main()
{
    TCHAR szOS[BUFSIZE];

    ZeroMemory(szOS,BUFSIZE);

```

```

if( GetOSDisplayString( szOS ) )
    printf("\n%s\n", szOS );
system("PAUSE");
}

```

Con este código nos mostraría la versión de Windows que tenemos pero a nosotros para este caso nos basta con el primer código que en mi caso muestra como resultado “5.1.2600” que es justo lo que necesito.

Si seguimos traceando hasta llegar a otro call que no es de sistema y entramos en el con F7 veremos esto:

0040F644	\$ A1 408B4800	mov	eax, ds:[488B40]	
0040F649	. 0BC0	or	eax, eax	
0040F64B	74 02	je	short 0040F64F	kiosco.0040F64F
0040F64D	. FFE0	jmp	eax	
0040F64F	> 68 2CF64000	push	40F62C	
0040F654	. B8 102C4000	mov	eax, 402C10	
0040F659	. FFD0	call	eax	
0040F65B	.- FFE0	jmp	eax	kernel32.GetSystemDirectoryA

Y sale mostrando esto:

Stack ss:[0012F670]=0015F7B4, (ASCII "C:\WINDOWS\system32")

Seguimos traceando otra vez hasta que lleguemos a otro call que tampoco sea de sistema:

00480DAD	. 52	push	edx	
00480DAE	. FF15 84124000	call	ds:[401284]	MSVBVM60.__vbaStrToAnsi
00480DB4	. 50	push	eax	
00480DB5	. E8 D2E8F8FF	call	0040F68C	kiosco.0040F68C
00480DBA	. FF15 78104000	call	ds:[401078]	MSVBVM60.__vbaSetSystemError
00480DC0	8D85 84F8FF	lea	eax, ss:[ebp-67C]	

Entramos en ese call y volvemos a ver un código familiar pero esta vez llama a otra función:

0040F68C	\$ A1 4C8B4800	mov	eax, ds:[488B4C]	
0040F691	. 0BC0	or	eax, eax	
0040F693	74 02	je	short 0040F697	kiosco.0040F697
0040F695	. FFE0	jmp	eax	
0040F697	> 68 74F64000	push	40F674	
0040F69C	. B8 102C4000	mov	eax, 402C10	
0040F6A1	. FFD0	call	eax	
0040F6A3	.- FFE0	jmp	eax	kernel32.FindFirstFileA
0040F6A5	00	db	00	

Y si seguimos traceando podremos ver como usa varias veces rtcHexVarFromVar.

Para el siguiente parámetro seguimos traceando hasta llegar al próximo call que no es del sistema:

00480FE5	. 50	push	eax	
00480FE6	. E8 A1E6F8FF	call	0040F68C	kiosco.0040F68C
00480FEB	. FF15 78104000	call	ds:[401078]	MSVBVM60.__vbaSetSystemError

Registers (FPU)	<	<	<	<	<	<	<
EAX 0015CD4C ASCII "C:\Archivos de programa\Kiosco3"							
ECX 0000001F							

Entramos y volvemos a ver algo familiar al llegar al salto:

0040F68C	\$ A1 4C8B4800	mov	eax, ds:[488B4C]	
0040F691	. 0BC0	or	eax, eax	
0040F693	.. 74 02	je	short 0040F697	kiosco.0040F697
0040F695	.- FFE0	jmp	eax	kernel32.FindFirstFileA
0040F697	> 68 74F64000	push	40F674	
0040F69C	. B8 102C4000	mov	eax, 402C10	
0040F6A1	. FFD0	call	eax	
0040F6A3	. FFE0	jmp	eax	
0040F6A5	00	db	00	

O sea que busca el primer archivo del directorio de instalación y en la estructura de salida obtiene esto:

```
0012F278 10 00 00 00 D2 1A A4 B7 12 08 CD 01 38 3E CB A5 ...Ò¸·Í8>Ë¥
0012F288 53 0F CD 01 38 3E CB A5 53 0F CD 01 00 00 00 00 SÍ8>Ë¥SÍ...
```

Que curioso, lo que hace es usar FindFirstFileA para obtener el segundo y el tercer parámetro de la cadena así que fuí a buscar a Google y encontré la suficiente información para crear el código generador de la segunda y tercera parte de la cadena y de paso descubrir que la parte que usa de la estructura que llena la función FindFirstFileA son los valores de:

FindFileData.ftCreationTime.dwLowDateTime → Para la primera parte del parámetro en hexadecimal  
FindFileData.ftCreationTime.dwHighDateTime → Para la segunda parte del parámetro en hexadecimal

Separados por una coma.

Con este código tendríamos los datos para el segundo y el tercer parámetro de la cadena:

```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
```

```
TCHAR* envVarStrings[] =
{
    TEXT("OS      = %OS%"),
    TEXT("PATH     = %PATH%"),
    TEXT("HOMEPATH  = %HOMEPATH%"),
    TEXT("TEMP      = %TEMP%")
};
```

```
#define ENV_VAR_STRING_COUNT (sizeof(envVarStrings)/sizeof(TCHAR*))
#define INFO_BUFFER_SIZE 32767
```

```
void printError( TCHAR* msg );
```

```
int main()
{
```



```

TCHAR infoBuf[INFO_BUFFER_SIZE];

WIN32_FIND_DATA FindFileData;
HANDLE hFind;

// Get and display the system directory.
if( !GetSystemDirectory( infoBuf, INFO_BUFFER_SIZE ) )
    printError( TEXT("GetSystemDirectory") );
printf("\nSystem Directory:  %s\n", infoBuf );

hFind = FindFirstFile(infoBuf , &FindFileData);
if (hFind == INVALID_HANDLE_VALUE)
{
    printf ("FindFirstFile failed (%d)\n", (int)GetLastError());
    return GetLastError();
}
else
{
    printf ("The first file found is %s\n", FindFileData.cFileName);
    printf ("The first file found is %IX\n", FindFileData.ftCreationTime.dwLowDateTime);
    printf ("The first file found is %IX\n", FindFileData.ftCreationTime.dwHighDateTime);
    FindClose(hFind);
}

strcpy(infoBuf,"C:\\Archivos de programa\\kiosco3");
hFind = FindFirstFile(infoBuf , &FindFileData);
if (hFind == INVALID_HANDLE_VALUE)
{
    printf ("FindFirstFile failed (%d)\n", (int)GetLastError());
    return GetLastError();
}
else
{
    printf ("The first file found is %s\n", FindFileData.cFileName);
    printf ("The first file found is %IX\n", FindFileData.ftCreationTime.dwLowDateTime);
    printf ("The first file found is %IX\n", FindFileData.ftCreationTime.dwHighDateTime);
    FindClose(hFind);
}
system("PAUSE");
return 0;
}

void printError( TCHAR* msg )
{
    DWORD eNum;
    TCHAR sysMsg[256];
    TCHAR* p;

    eNum = GetLastError( );
    FormatMessage( FORMAT_MESSAGE_FROM_SYSTEM |

```

```

    FORMAT_MESSAGE_IGNORE_INSERTS,
    NULL, eNum,
    MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
    sysMsg, 256, NULL );

// Trim the end of the line and terminate it with a null
p = sysMsg;
while( ( *p > 31 ) || ( *p == 9 ) )
    ++p;
do { *p-- = 0; } while( ( p >= sysMsg ) &&
    ( ( *p == '.' ) || ( *p < 33 ) ) );

// Display the message
printf("\n\t%s failed with error %d (%s)", msg, (int)eNum, sysMsg );
}

```

Y aquí tenemos el resultado:

**System Directory: C:\WINDOWS\system32**

**The first file found is system32**

**The first file found is 3806FE78 → Primera parte del segundo parametro y concatenamos una coma detras**

**The first file found is 1CAA637 → Segunda parte del segundo parametro**

**The first file found is Kiosco3**

**The first file found is B7A41AD2 → Primera parte del tercer parámetro y concatenamos una coma detras**

**The first file found is 1CD0812 → Segunda parte del tercer parámetro**

**Presione una tecla para continuar . . .**

Si seguimos traceando llegaremos aquí:

00481142	. 50	push	eax	
00481143	. 51	push	ecx	
00481144	. 6A 06	push	6	
00481146	. FF15 38104000	call	ds:[401038]	MSVBVM60.__vbaFreeVarList
0048114C	. 8B8D A0FEFFFF	mov	ecx, ss:[ebp-160]	
00481152	. 8B35 2C124000	mov	esi, ds:[40122C]	MSVBVM60.__vbaStrCopy
00481158	. 83C4 1C	add	esp, 1C	
0048115B	. BA 286D4100	mov	edx, 416D28	UNICODE "HARDWARE\DESCRIPTION\System"
00481160	. FFD6	call	esi	MSVBVM60.__vbaStrCopy; <MSVBVM60.__vbaStrCopy>
00481162	. 8B8D C0FEFFFF	mov	ecx, ss:[ebp-140]	
00481168	. BA 40424100	mov	edx, 414240	UNICODE "SystemBiosDate"
0048116D	. FFD6	call	esi	
0048116F	. 8B85 A0FEFFFF	mov	eax, ss:[ebp-160]	
00481175	. BA 286D4100	mov	edx, 416D28	UNICODE "HARDWARE\DESCRIPTION\System"
0048117A	. 8D48 04	lea	ecx, ds:[eax+4]	
0048117D	. FFD6	call	esi	
0048117F	. 8B8D C0FEFFFF	mov	ecx, ss:[ebp-140]	
00481185	. BA 542D4100	mov	edx, 412D54	UNICODE "SystemBiosVersion"
0048118A	. 83C1 04	add	ecx, 4	
0048118D	. FFD6	call	esi	
0048118F	. 8B85 A0FEFFFF	mov	eax, ss:[ebp-160]	
00481195	. BA 286D4100	mov	edx, 416D28	UNICODE "HARDWARE\DESCRIPTION\System"

00481195	. BA 286D4100	mov	edx, 416D28	UNICODE "HARDWARE\DESCRIPTION\System"
0048119A	. 8D48 08	lea	ecx, ds:[eax+8]	
0048119D	. FFD6	call	esi	
0048119F	. 8B8D C0FEFFFF	mov	ecx, ss:[ebp-140]	
004811A5	. BA 646D4100	mov	edx, 416D64	UNICODE "VideoBiosDate"
004811AA	. 83C1 08	add	ecx, 8	
004811AD	. FFD6	call	esi	
004811AF	. 8B85 A0FEFFFF	mov	eax, ss:[ebp-160]	
004811B5	. BA 286D4100	mov	edx, 416D28	UNICODE "HARDWARE\DESCRIPTION\System"
004811BA	. 8D48 0C	lea	ecx, ds:[eax+C]	
004811BD	. FFD6	call	esi	
004811BF	. 8B8D C0FEFFFF	mov	ecx, ss:[ebp-140]	
004811C5	. BA FC244100	mov	edx, 4124FC	UNICODE "VideoBiosVersion"
004811CA	. 83C1 0C	add	ecx, 0C	
004811CD	. FFD6	call	esi	
004811CF	. 8B85 A0FEFFFF	mov	eax, ss:[ebp-160]	
004811D5	. BA 846D4100	mov	edx, 416D84	UNICODE "HARDWARE\DESCRIPTION\System\CentralProcessor\0"
004811DA	. 8D48 10	lea	ecx, ds:[eax+10]	
004811DD	. FFD6	call	esi	
004811DF	. 8B8D C0FEFFFF	mov	ecx, ss:[ebp-140]	
004811E5	. BA E86D4100	mov	edx, 416DE8	UNICODE "ProcessorNameString"
004811EA	. 83C1 10	add	ecx, 10	
004811ED	. FFD6	call	esi	

En esa parte lo que hace es copiar esas cadenas que son muy sospechosas y si seguimos traceando llegamos a este bucle y traceamos hasta el siguiente call que no sea de sistema:

004811F4	> B8 05000000	mov	eax, 5	
004811F9	. 66:3BF0	cmp	si, ax	
004811FC	.. 0F8F 95000000	jg	00481297	kiosco.00481297
00481202	. 0FBFDE	movsx	ebx, si	
00481205	. 4B	dec	ebx	
00481206	. 83FB 06	cmp	ebx, 6	
00481209	.. 72 11	jb	short 0048121C	kiosco.0048121C
0048120B	. FF15 10114000	call	ds:[401110]	MSVBVM60.__vbaGenerateBoundsError
00481211	. 83FB 06	cmp	ebx, 6	
00481214	.. 72 06	jb	short 0048121C	kiosco.0048121C
00481216	. FF15 10114000	call	ds:[401110]	MSVBVM60.__vbaGenerateBoundsError
0048121C	> 8B95 C0FEFFFF	mov	edx, ss:[ebp-140]	
00481222	. 8B8D A0FEFFFF	mov	ecx, ss:[ebp-160]	
00481228	. 8B049A	mov	eax, ds:[edx+ebx*4]	
0048122B	. 8B1499	mov	edx, ds:[ecx+ebx*4]	
0048122E	. 50	push	eax	
0048122F	. 52	push	edx	
00481230	. 68 02000080	push	80000002	
00481235	. E8 26F5FFFF	call	00480760	kiosco.00480760
0048123A	. 8BD0	mov	edx, eax	
0048123C	. 8D8D ACFEFFFF	lea	ecx, ss:[ebp-154]	
00481242	. FFD7	call	edi	
00481244	. 8B85 D0FEFFFF	mov	eax, ss:[ebp-130]	

Y entramos con F7 y volvemos a encontrarnos con un código :

0040F77C	\$ A1 708B4800	mov	eax, ds:[488B70]	
0040F781	. 0BC0	or	eax, eax	
0040F783	.. 74 02	je	short 0040F787	kiosco.0040F787
0040F785	. FFE0	jmp	eax	
0040F787	> 68 64F74000	push	40F764	
0040F78C	. B8 102C4000	mov	eax, 402C10	
0040F791	. FFD0	call	eax	
0040F793	.. FFE0	jmp	eax	ADVAPI32.RegQueryValueExA
0040F795	. 00	db	00	

Lo que hace es, con las cadenas que vimos antes, obtener los valores del registro, o sea, le da la ruta del registro del que quiere obtener su valor y el nombre de este y así obtiene los demás parámetros de la cadena. El código que hice en C para obtener dichos valores es este:

```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>

//VARIABLES GLOBALES
unsigned char dwKeyEn[1024];

//DECLARACION DE FUNCIONES
HKEY Abrir_Registro(char ruta[]);
int Obtener_Valor(HKEY hKey,char nombre[],DWORD dwType);

//FUNCION PARA OBTENER EL HKEY DE UNA ENTRADA DE REGISTRO
HKEY Abrir_Registro(char ruta[])
{
    HKEY hKey;
    RegOpenKeyEx(HKEY_LOCAL_MACHINE,ruta,0,KEY_READ,&hKey);
    return hKey;
}

//FUNCION PARA OBTENER EL VALOR DE LA ENTRADA DEL REGISTRO QUE LE INDIQUEMOS
CON EL HKEY
int Obtener_Valor(HKEY hKey,char nombre[],DWORD dwType)
{
    DWORD dwLen = 0x0ff;
    return RegQueryValueEx(hKey,nombre,0,&dwType,(LPBYTE)&dwKeyEn,&dwLen);
}

int main(int argc, char *argv[])
{
    HKEY hKey;
    DWORD dwType;
    char
    nombre[][20]={"SystemBiosDate","SystemBiosVersion","VideoBiosDate","VideoBiosVersion","Processor
    NameString"};
    int i;

    for(i=0;i<5;i++)
    {
        if(i%2 == 0)
            dwType = REG_SZ;
        else
            dwType = REG_MULTI_SZ;
        if(i==4)
        {
            hKey=Abrir_Registro("HARDWARE\\DESCRIPTION\\System\\CentralProcessor\\0");
```

```

}
else
{
    hKey=Abrir_Registro("HARDWARE\\DESCRIPTION\\System");
}
if(hKey == ERROR_SUCCESS)
    printf("Error al abrir el registro");
else
{
    if((Obtener_Valor(hKey,nombre[i],dwType)) != ERROR_SUCCESS)
    {
        dwKeyEn[0]= 'N';
        dwKeyEn[1]= 'u';
        dwKeyEn[2]= 'l';
        dwKeyEn[3]= 'l';
        dwKeyEn[4]= '\0';
    }
    printf("%s: %s\n",nombre[i],dwKeyEn);
    RegCloseKey(hKey);
}

    dwKeyEn[0]='\0';
}
system("PAUSE");
return 0;
}

```

En mi caso muestra esto:

```

SystemBiosDate: 03/21/07
SystemBiosVersion: ACRSYS - 6040000
VideoBiosDate: 06/02/15
VideoBiosVersion: Null
ProcessorNameString: AMD Turion(tm) 64 Mobile Technology MK-38
Presione una tecla para continuar . . .

```

Con esto ya tendríamos todos los valores de la cadena para poder crear un keygen sin necesidad de tener que usar el archivo kiosco.lic y ya he cumplido con mi objetivo que era crear un keygen y saber como creaba la cadena. Espero haber explicado todo bien, que os haya gustado y que no os haya parecido aburrido.

No soy ningún portento de la programación así que no voy a entrar a discutir si los códigos están mejor o peor estructurados y si se podría haber conseguido de una forma mas eficiente y ordenada, es más, si alguien sabe como optimizarlos más pues bienvenidos sean esos códigos. Lo que si que puedo asegurar es que funcionan.

Por ultimo quería dar mi especial agradecimiento a Daniel la calabera, Nox, Guan de dio, Edy .:=InDuLgEo=., Vortice, Renato, y Juan el Viejo por toda su ayuda para poder llevar a cabo este estudio.