

# ESTUDIO DE PROTECCIONES BASICO PARA PRINCIPIANTES

(También llamado cracking)

Impartido por Ratón (Nivel principiante)

## Nota

Cada capitulo ira acompañado de su crackme correspondiente. No facilitare páginas de donde bajarse herramientas ni enlaces a páginas de crackers, la intención es que busquéis en Internet todo lo necesario. Seguro que encontráis mas paginas de herramientas, tutoriales y utilidades relacionadas con este tema que las que yo pueda deciros.

Con esto solo quiero fomentar vuestro interés, además se que la búsqueda os proporcionara gratas sorpresas.

A todos un saludo.

## Capitulo 9

### Victima

Crackmes VB de Alfa

### Herramientas

Oilly Debugger.

Smartcheck

Instinto (en mi caso mucho)

### Objetivo

"Tocar" Visual Basic y Smartcheck

#### Nota:

Dejo para el final el lenguaje que menos me gusta y el que menos he tocado, pues suelo huir de instalar en mi PC programas en visual basic al igual que he evitado los crackmes en vb.

No pretendo levantar la típica polémica sobre vb que podemos ver en todos los foros de programación, simplemente le tengo manía por que a la hora de desensamblar me resulta confuso y me pierdo.

Quizás si hubiera empezado desensamblando crackmes en vb ahora serian mis favoritos, pero la verdad el desensamblado de otros lenguajes se me hace mas comprensible.

Quede claro que es cuestión de gusto y quizá también de limitación personal, a lo mejor según vaya aprendiendo este tema del cracking los crackmes en visual basic acaban convirtiéndose en mis favoritos.

### Al ataque

Podría explicar lo de las DLLs de visual basic y que estos programas (vb) dependen de ellas para su ejecución, también podría escribir un listado de todas las API s necesarias, pero seria prácticamente un copy – paste de tutoriales de otros crackers que controlan el tema y que en mi opinión deberíais leer.

Antes que copiar a nadie para rellenar espacio en este capitulo y quedar de put@ madre

demostrando mis falsos conocimientos en el tema prefiero recomendaros que leáis tutoriales de otros crackers como Coco o Shoulck donde se explica el tema del crack en visual basic. Yo seguiré aquí mi línea de explicar lo más claramente posible lo que aprendí por mis propias y pocas experiencias sobre el crack en visual basic. Estos crackmes de Alfa me vinieron al pelo para esta introducción al crack en vb, gracias Alfa.

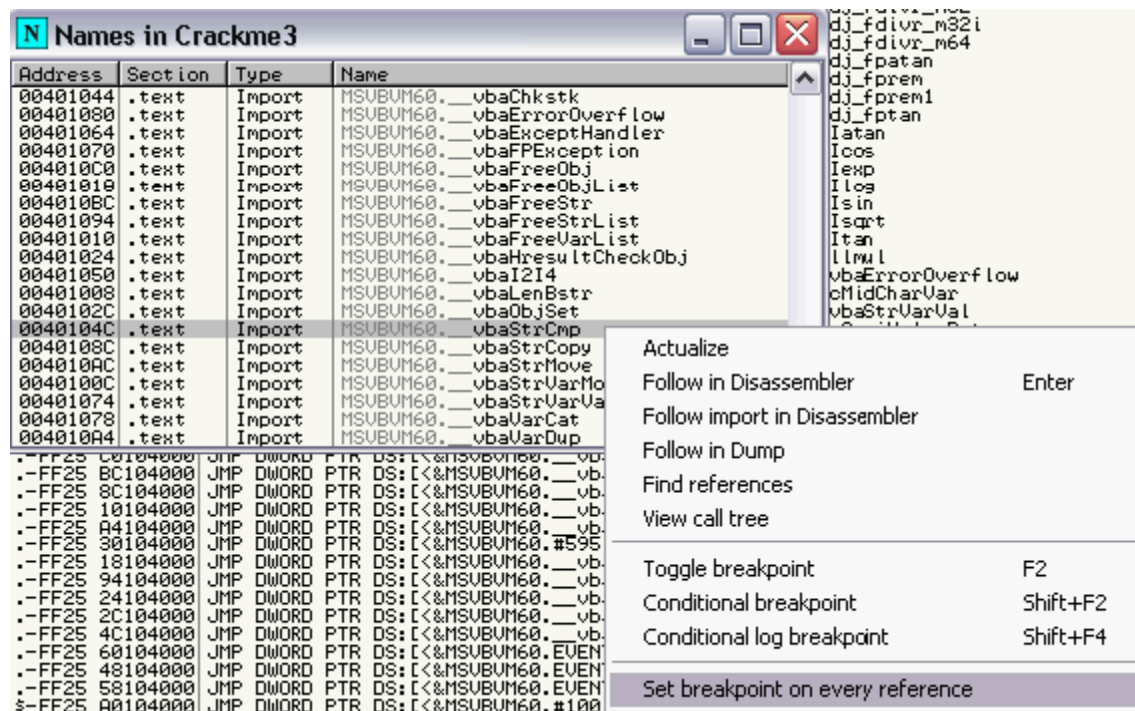
## Crackme 3 de Alfa

El típico crackme con numero y nombre, lo mejor para ir viendo poco a poco este tema.



Lo cargamos en Olly y vamos a ir directamente a cazarlo por medio de las llamadas a las APIs. Utilizaremos una de las APIs mas comunes para cazar nombres: `vbaStrCmp` (String compare - comparar cadenas de texto)

Con control + N set breakpoint on every reference en la API `__vbaStrCmp`



F9 e introducimos nombre y serial y pulsamos el botón registrarse

Ollly para en uno de los BPs

Vemos la llamada a la API y en registers nuestro numero

```
00401F36 . 68 581B4000 | PUSH Crackme3.00401B58
00401F38 . FF15 4C104000 | CALL DWORD PTR DS:[<&MSUBUM60.__vbaStrC MSUBUM60.__vbaStrCmp
00401F41 . 8B55 E8      | MOV EDX,DWORD PTR SS:[EBP-18]
```

Registers (FPU)

EAX	00000000
ECX	0014F9C4 UNICODE "15151515"
EDX	003C0608

Registers ->

Pulsamos F9 otra vez y un poco más abajo la segunda parada donde vemos nuestro nombre

```
00401F38 . FF15 4C104000 | CALL DWORD PTR DS:[<&MSUBUM60.__vbaStrC MSUBUM60.__vbaStrCmp
00401F41 . 8B55 E8      | MOV EDX,DWORD PTR SS:[EBP-18]
00401F44 . 8BF8        | MOV EDI,EAX
00401F46 . F7DF        | NEG EDI
00401F48 . 1BFF        | SBB EDI,EDI
00401F4A . 52          | PUSH EDI
00401F4B . 47          | INC EDI
00401F4C . 68 581B4000 | PUSH Crackme3.00401B58
00401F51 . F7DF        | NEG EDI
00401F53 . FF15 4C104000 | CALL DWORD PTR DS:[<&MSUBUM60.__vbaStrC MSUBUM60.__vbaStrCmp
00401F59 . F7D8        | NEG EAX
```

Registers (FPU)

EAX	00000001
ECX	00000000
EDX	0014FAA4 UNICODE "raton"
EBX	00000000

Registers ->

F9 por tercera vez y vemos en registers dos cadenas en unicode, parece la típica comparación de seriales verdadero / falso

```
00402206 . 51          | PUSH ECX
00402207 . FF15 4C104000 | CALL DWORD PTR DS:[<&MSUBUM60.__vbaStrC MSUBUM60.__vbaStrCmp
0040220D . 8BF8        | MOV ESI,EAX
```

Registers (FPU)

EAX	0014C70C UNICODE "15151515"
ECX	0014FCE4 UNICODE "7261746F6E"
EDX	003C0608

Registers ->

Lo probamos



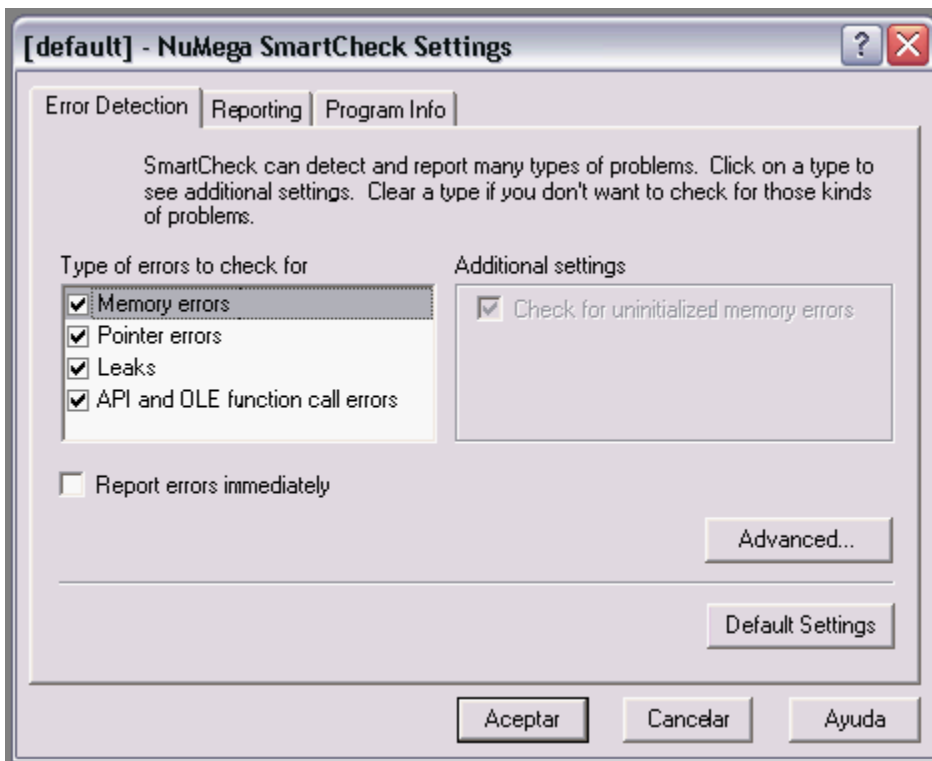
Intentándolo con Smartcheck

## Configuración de Smartcheck

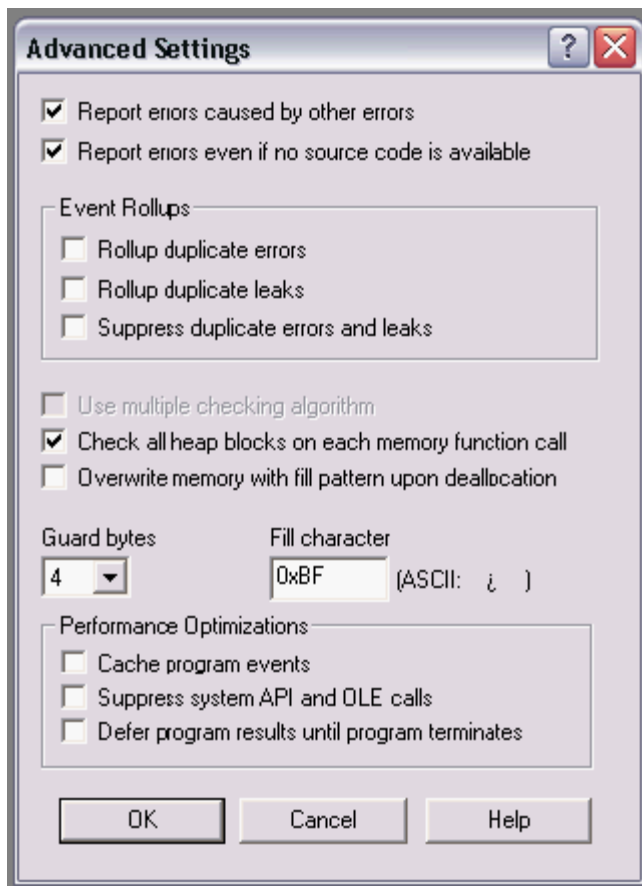
Es un paso necesario tener bien configurado el Smartcheck, sobra decir que la configuración del programa que os voy a mostrar la saque de un tutorial de un gran Cracker: Ricardo Narvaja. Lo adapte haciendo pruebas a mi versión y tal como lo muestro a mi me funciona, no obstante tened en cuenta mi poca experiencia con vb.

Utilizo la versión 6.2

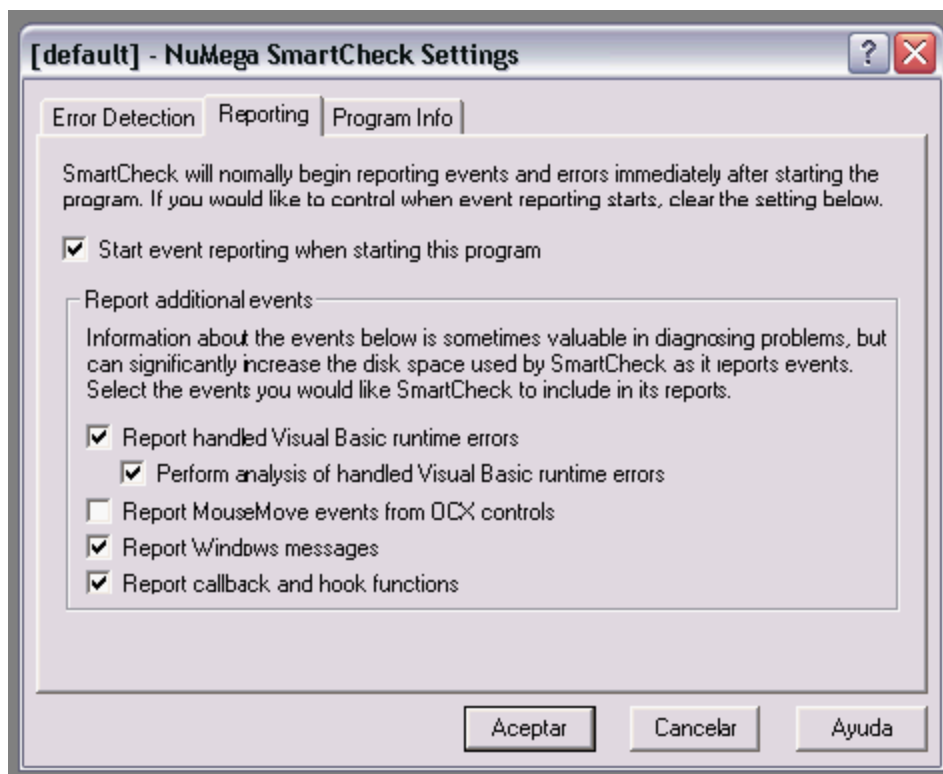
En Error detection marcamos



Pulsamos el botón Advanced... y en Advanced Settings



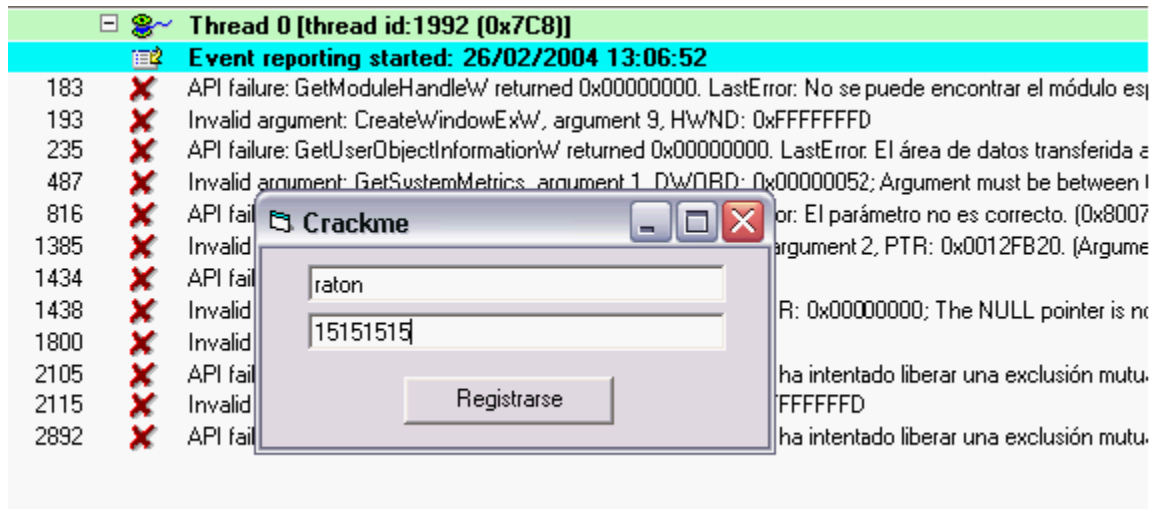
Pestaña reporting todo marcado menos el movimiento del raton (el Raton procura desmarcarse siempre que le resulta posible)



Bien una vez marcadas estas opciones lo guardamos para que siempre que arranquemos Smartcheck no tengamos que volverlas a marcar

Cargamos el crackme 3 de alfa en Smartcheck y pulsamos play (run) ▶

Aparece el crackme ejecutándose, rellenamos los edit y pulsamos el botón registrarse



Fijaros que una vez que hemos pulsado el botón nos aparece el cartel malo y en el reporte abajo a la izquierda \_Click

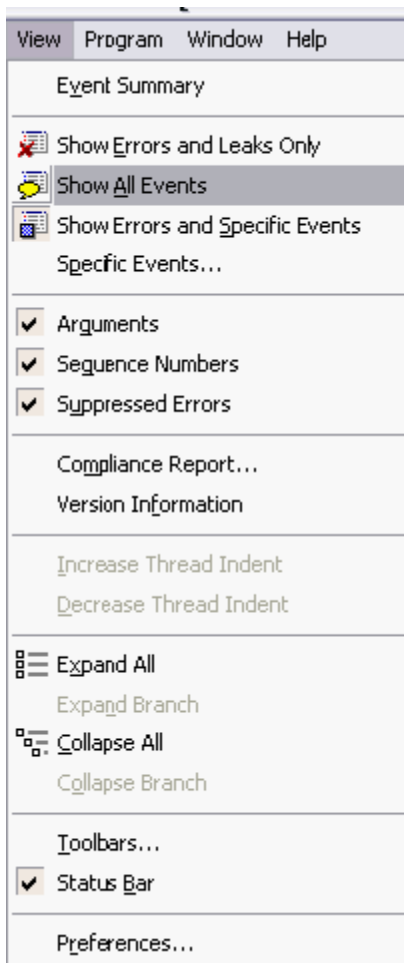
Smartcheck va grabando todo lo que ocurre al ejecutarse el programa y también todos nuestros movimientos



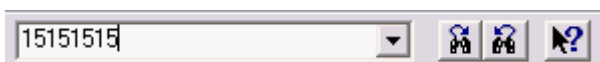
Aceptamos que somos malos crackers – en mi caso sin dudarlo – y pulsamos stop ■

Antes de hacer nada más desplegamos View y marcamos como en la imagen las casillas Arguments, Sequence Numbers y Suppressed Errors

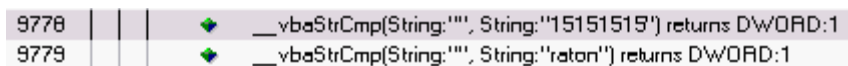
Antes de cerrar este menú haced click en Show All Events



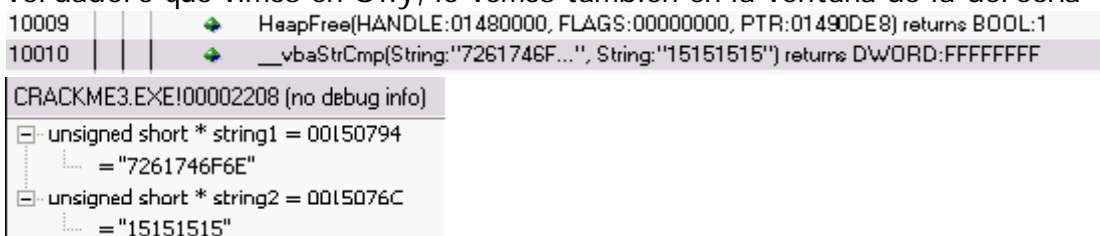
Arriba en el edit escribid el numero malo y picad en la primera imagen de los prismáticos o edit - find



Smart para aquí y vemos el API vbastrcmp con nuestro numero y debajo nuestro nombre



Seguimos buscando con F3 o los prismáticos (y yo que pensaba que esto solo servia para ver a la vecina en bolas) y paramos aquí donde compara nuestro numero con otra String, el serial verdadero que vimos en Olly, lo vemos también en la ventana de la derecha



Fue un crackme fácil de resolver, a mi altura, para no romperme el coco y verlo clarito.

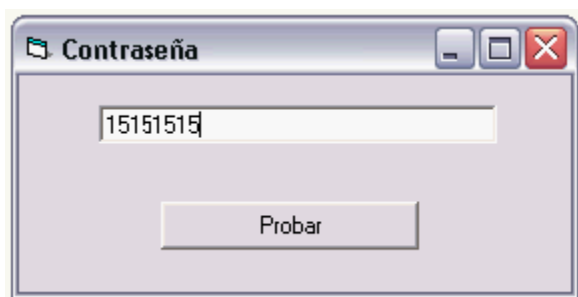
Crackme 5 de Alfa

Otro facilito para ir pillando el rollo a vb

Lo cargamos en Olly y ponemos BP en todas las vbaStrCmp

B Breakpoints			
Address	Module	Active	Disassembly
0040117C	Crackme5	Always	JMP DWORD PTR DS:[<&MSUBUM60.__vbaStrCmp>]
00401D24	Crackme5	Always	CALL DWORD PTR DS:[<&MSUBUM60.__vbaStrCmp>]
00401E7C	Crackme5	Always	MOV EDI,DWORD PTR DS:[<&MSUBUM60.__vbaStrCmp>]

F9 y escribimos nuestro pass



Probamos y Olly para

00401D24	. FF15 40104000	CALL DWORD PTR DS:[<&MSUBUM60.__vbaStrCmp>]	MSUBUM60.__vbaStrCmp
00401D2A	. 8BF0	MOV ESI,EAX	
00401D2C	. 8D4D E8	LEA ECX,DWORD PTR SS:[EBP-18]	
00401D2F	. F7DE	NEG ESI	
00401D31	. 1BF6	SBB ESI,ESI	
00401D33	. 46	INC ESI	

F9 y Olly para de nuevo y vemos en unicode dos nombres: Roman y Fernando (en los anteriores capítulos veíamos las strings que nos interesaban en ascii)

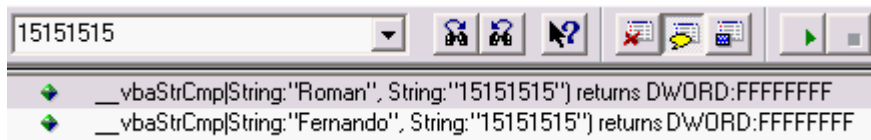
00401E7C	. 8B3D 40104000	MOV EDI,DWORD PTR DS:[<&MSUBUM60.__vbaStrCmp>]	MSUBUM60.__vbaStrCmp
00401E82	. 51	PUSH ECX	
00401E83	. 68 081A4000	PUSH Crackme5.00401AD8	UNICODE "Roman"
00401E88	. FFD7	CALL EDI	<&MSUBUM60.__vbaStrCmp>
00401E8A	. 8B55 E8	MOV EDX,DWORD PTR SS:[EBP-18]	
00401E8D	. 8BF0	MOV ESI,EAX	
00401E8F	. F7DE	NEG ESI	
00401E91	. 1BF6	SBB ESI,ESI	
00401E93	. 52	PUSH EDX	
00401E94	. 46	INC ESI	
00401E95	. 68 C01A4000	PUSH Crackme5.00401AC0	UNICODE "Fernando"
00401E9A	. F7DE	NEG ESI	

Abrámoslo con Smartcheck y busquemos nuestro serial como lo hicimos con el crackme anterior  
Si no podéis encontrar el serial será por que tenáis desmarcadas alguna de la opciones que



marcamos al configurar, marcadas y guardad los cambios.

Llegamos a un punto en que lo compara con Roman y en la línea siguiente con Fernando



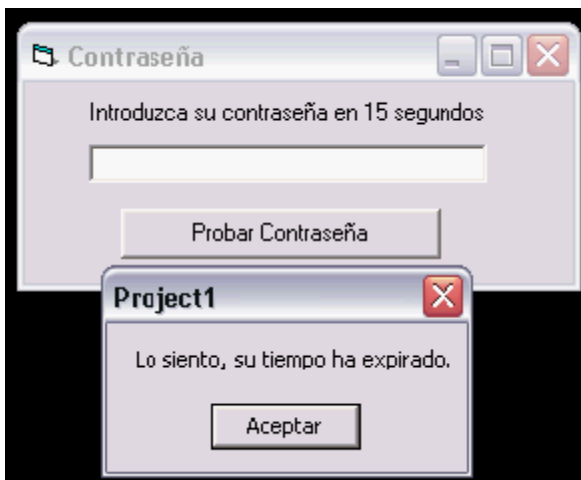
Cerramos Smartcheck y probamos con los dos nombres



Así me gusta que sigamos en mi línea.

Crackme 4 de Alfa

Este nos lo intenta poner un poco más difícil apurándonos con el tiempo, pero vemos que es lo mismo



Con olly

F9 y para, observamos justo la línea encima de donde paro y vemos la palabra secreto

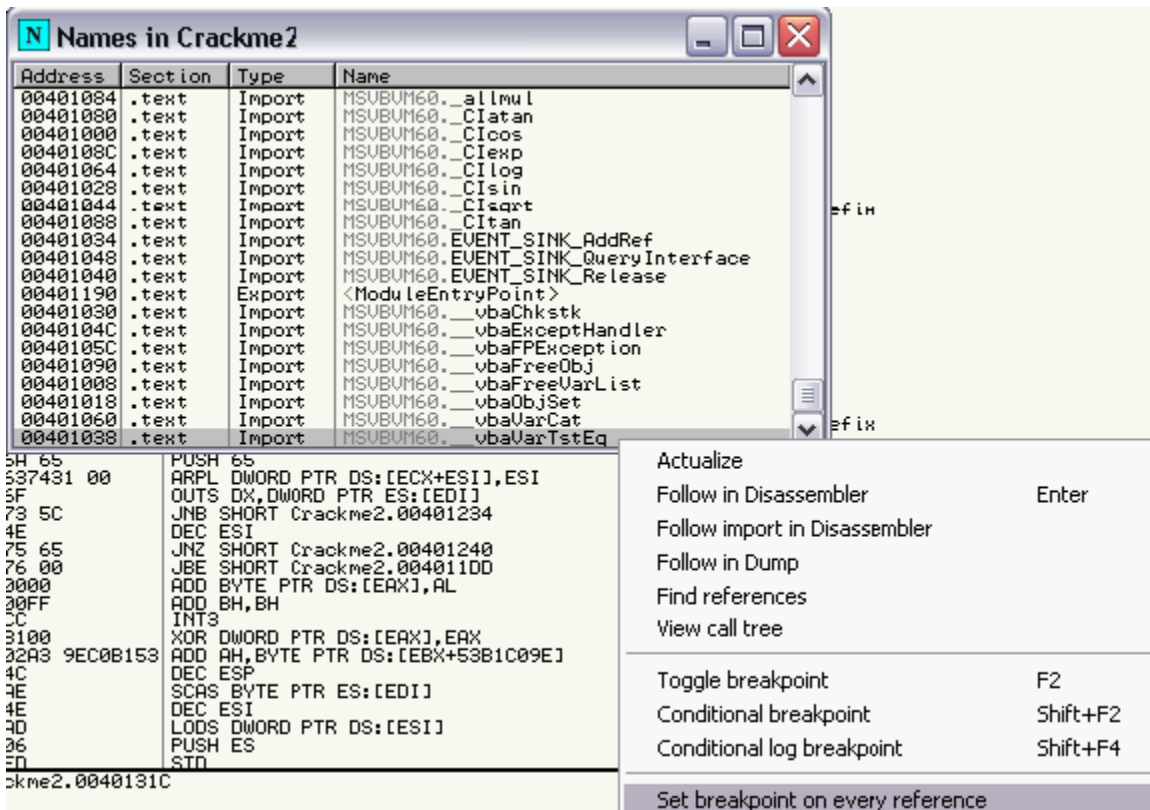
Con smartcheck find 15151515

El único problema para resolver este crackme esta en vuestra rapidez para escribir la palabra secreto

## Crackme 1 de Alfa

Address	Section	Type	Name
00401084	.text	Import	MSUBUM60._allmul
00401086	.text	Import	MSUBUM60._CIsan
00401088	.text	Import	MSUBUM60._Cicos
0040108C	.text	Import	MSUBUM60._CIexp
00401064	.text	Import	MSUBUM60._CIlog
00401028	.text	Import	MSUBUM60._CIsin
00401044	.text	Import	MSUBUM60._CIsqrt
00401088	.text	Import	MSUBUM60._Citan
00401034	.text	Import	MSUBUM60.EVENT_SINK_AddRef
00401048	.text	Import	MSUBUM60.EVENT_SINK_QueryInterface
00401040	.text	Import	MSUBUM60.EVENT_SINK_Release
00401190	.text	Export	<ModuleEntryPoint>
00401030	.text	Import	MSUBUM60._vbaChkst
0040104C	.text	Import	MSUBUM60._vbaExceptionHandler
0040105C	.text	Import	MSUBUM60._vbaFPException
00401090	.text	Import	MSUBUM60._vbaFreeObj
00401088	.text	Import	MSUBUM60._vbaFreeVarList
00401018	.text	Import	MSUBUM60._vbaObjSet
00401060	.text	Import	MSUBUM60._vbaVarCat
00401038	.text	Import	MSUBUM60._vbaVarTestEq

No lo encontramos, pero tenemos otra forma de cazar el serial utilizaremos `vbaVarTstEqBP` como anteriormente



F9, escribimos nuestro numero y para aquí, como no vemos nada alrededor entramos en el Call 00401DDF con F7

00401DDE	. 50	PUSH EAX	
00401DDF	. FF15 40104000	CALL DWORD PTR DS:[&MSUBUM60._vbaVarT	MSUBUM60._vbaVarTstEq
00401DE1	. 804D E4	LEA ECX,DWORD PTR SS:[EBP-1C]	

Aparecemos aquí y empezamos a bajar con F7 y solo con F7 despacio

734AABE6	FF7424 08	PUSH DWORD PTR SS:[ESP+8]	
734AABEA	FF7424 08	PUSH DWORD PTR SS:[ESP+8]	
734AABEE	6A 00	PUSH 0	
734AABF0	E8 2254FFFF	CALL MSUBUM60.734A0017	
734AABF5	000405 44FC3C73	MOV EAX,DWORD PTR DS:[EAX*4+733CFC44]	
734AABFC	C2 0800	RETN 8	

Llegamos aquí y vemos en el Stack el numerito 1633 (os adelanto que vosotros veréis otro distinto, no os preocupéis, al final entenderéis el porque)

734A0018	8BEC	MOV EBP,ESP	
734A001A	83EC 38	SUB ESP,38	
734A001D	8B55 10	MOV EDX,DWORD PTR SS:[EBP+10]	
734A0020	8B4D 0C	MOV ECX,DWORD PTR SS:[EBP+C]	
734A0023	53	PUSH EBX	

Stack -> 0012F0C0 7348A644 RETURN to MSUBUM60  
0012F0C4 0014FB94 UNICODE "1633"

Seguimos con F7 hasta que en esta dirección (fijaros que no pertenece al programa sino al modulo OLEAUT32 es lo que os comente al principio que visual basic se apoya en las dlls y librerías de Windows) vemos en registers lo que parece la comparación de siempre entre nuestro serial y el bueno

CPU - main thread, module OLEAUT32		
7716B882	50	PUSH EAX
7716B883	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]
7716B886	8B51 08	MOV EDX,DWORD PTR DS:[ECX+8]
7716B889	52	PUSH EDX
7716B88A	E8 300A0000	CALL OLEAUT32.VarBstrCmp
7716B88F	8945 BC	MOV DWORD PTR SS:[EBP-44],EAX

Registers ->

Registers (FPU)	
EAX	0014FB94 UNICODE "1639"
ECX	0012F48C
EDX	0014C5C4 UNICODE "16161616"
EBX	00000000

Lo probamos pero no funciona

Me habré equivocado, vuelvo a repetir la operación y al llegar al mismo punto veo otro serial distinto

7716B886	8B51 08	MOV EDX,DWORD PTR DS:[ECX+8]
7716B889	52	PUSH EDX
7716B88A	E8 300A0000	CALL OLEAUT32.VarBstrCmp

Registers ->

Registers (FPU)	
EAX	0014FB94 UNICODE "1639"
ECX	0012F48C
EDX	0014C5C4 UNICODE "16161616"
EBX	00000000

Ahora seguro que si además justo debajo de donde veo eso hay algo que parece una comparación de strings en 7716B889 VarBstrCmp

Introduzco 1639 (vosotros veréis otro distinto) y tampoco.

Empiezo a mosquearme y pienso que alfa habrá introducido una rutina para cambiar el serial, alguna de esas operaciones matemáticas que odio por que soy torpe y me vuelvo loco intentando averiguar que c#ñ# hacen.

Pienso que es su primer crackme y que no pudo tener tan mala idea, empiezo a utilizar la intuición que es lo único que me queda en estos casos y después de unas cuantas pruebas salta la chispa.

Mirad la solución



Toma como serial la hora / minutos del sistema

Si pasamos ese minuto y probamos el serial no funciona, hay que introducir la hora y los minutos exactos, menos mal que en el fondo fue un buen tipo y no utilizo lo segundos también.

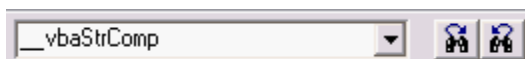


Con Smartcheck

Busco mi serial pero ahora no lo pilla el crackme este de los huevos !!!

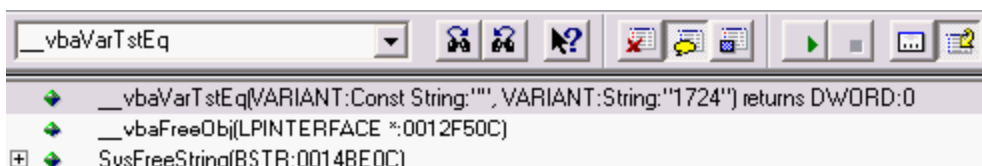


Estoy hundido en la más profunda miseria pues no conozco el funcionamiento de Smartcheck  
Intento una a la desesperada pero como siempre confiando en la intuición



Nada, claro es lógico, probare con `__vbaVarTstEq` que es lo que utilice para sacar el serial con Olly

Ahora si veo 1724, las 5 y 24 minutos se me volvió a pasar la hora del te.



Bueno muchos "crackers de verdad" como yo digo se habrán echado unas risas a costa de este capitulo, lo importante es que veáis como sin conocimientos de un lenguaje de programación y

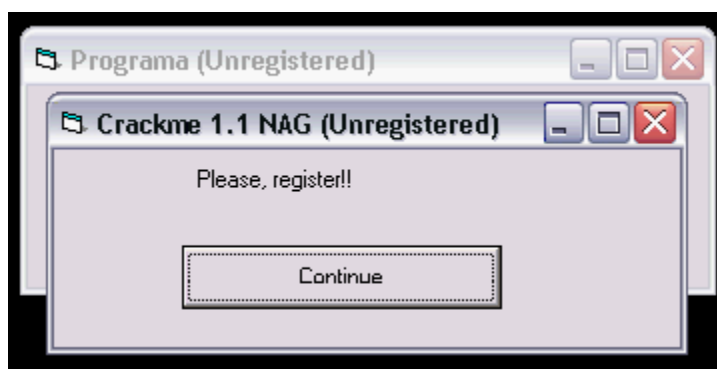
apenas conocimientos de crackeo, con un par de API s sacadas de algún tuto de algún maestro (y sino utilizo la intuición) con un poco de lógica y siguiendo vuestro instinto se puede conseguir el serial.

Nags

Lo poco que se sobre eliminar Nags en vb lo aprendí de los tutos de Coco - si estáis interesados en vb empezad por leerle a el – aquí os dejo un pequeño apunte sobre eliminar Nags.

Crackme Nag de Alfa creado a propósito para el curso (gracias again)

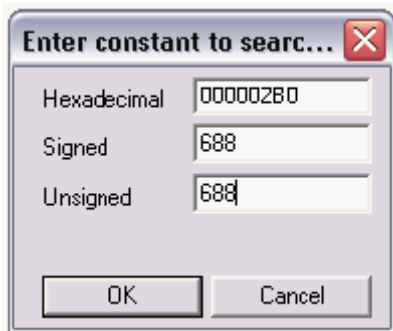
Al ejecutar el crackme aparece una Nag y cuando lo cerramos otra



Sigo los pasos que aprendí de Coco botón derecho [Search for – All constants](#)

Name (label) in current module	Ctrl+N
Name in all modules	
Command	Ctrl+F
Sequence of commands	Ctrl+S
Constant	
Binary string	Ctrl+B
Next	Ctrl+L
All intermodular calls	
All commands	
All sequences	
All constants	
All switches	
All referenced text strings	
User-defined label	
User-defined comment	

Escribo [2B0](#) y OK para buscar



Hay un push y un call, me centro en el call

```
004020DC | . 8951 0C      MOV DWORD PTR DS:[ECX+C],EDX
004020DE | FF97 B0020000 CALL DWORD PTR DS:[EDI+2B0]
004020E5 | . 85C0         TEST EAX,EAX
```

Mas abajo veo el final de este tramo de código, un retn en la dirección [402120](#)

```
004020DC | . 8951 0C      MOV DWORD PTR DS:[ECX+C],EDX
004020DE | FF97 B0020000 CALL DWORD PTR DS:[EDI+2B0]
004020E5 | . 85C0         TEST EAX,EAX
004020E7 | . DBE2        FCLEX
004020E9 | . 7D 12       JGE SHORT crackmen.004020FD
004020EB | . 68 B0020000 PUSH 2B0
004020F0 | . 68 A41C4000 PUSH crackmen.00401CA4
004020F5 | . 56          PUSH ESI
004020F6 | . 50          PUSH EAX
004020F7 | FF15 18104000 CALL DWORD PTR DS:[<&MSUBUM60
004020FD | > C745 FC 00000 MOV DWORD PTR SS:[EBP-4],0
00402104 | . 8B45 08     MOV EAX,DWORD PTR SS:[EBP+8]
00402107 | . 50          PUSH EAX
00402108 | . 8B08       MOV ECX,DWORD PTR DS:[EAX]
0040210A | FF51 08     CALL DWORD PTR DS:[ECX+8]
0040210D | . 8B45 FC     MOV EAX,DWORD PTR SS:[EBP-4]
00402110 | . 8B4D EC     MOV ECX,DWORD PTR SS:[EBP-14]
00402113 | . 5F          POP EDI
00402114 | . 5E          POP ESI
00402115 | 64:890D 00000 MOV DWORD PTR FS:[0],ECX
0040211C | . 5B          POP EBX
0040211D | . 8BE5       MOV ESP,EBP
0040211F | . 5D          POP EBP
00402120 | . C2 0400    RETN 4
00402123 | 90          NOP
```

Y mas arriba el principio de este tramo del código en [402040](#) aquí hago control + R para ver desde donde es llamado este tramo del código donde se genera la Nag

```
0040203F | . 90          NOP
00402040 | > 55          PUSH EBP
00402041 | . 8BEC       MOV EBP,ESP
```

Y Olly me envía a [401AAA](#)

```
00401AA2 | . 916C24 04 330 SUB DWORD PTR SS:[ESP+4],33
00401AAA | ✓ E9 91050000 JMP crackmen.00402040
00401AAF | . 00         DB 00
```

Ahora cambiare la dirección del salto para que vaya directamente al retn [402120](#) y evite la aparición del Nag

```
00401AA2 | . 916C24 04 330 SUB DWORD PTR SS:
00401AAA | -E9 71E6C3FF JMP 00040120
00401AAF | . 00         DB 00
```

Guardad cambios y probad, crackme resuelto solo con buscar esa constante y cambiando el valor de un salto

Como no me gusta solo copiar de otros investigue un poco y probé a hacerlo de otra forma: nopear el call 4020DF

Lo probé y me funciono sin tener que buscar el JMP y cambiarlo.

Esta forma es "experimental" y no se si funcionara siempre.

004020D9	:	8B55 E8	MOV EDX,DWORD
004020DC	:	8951 0C	MOV DWORD PTR
004020DF	:	90	NOP
004020E0	:	90	NOP
004020E1	:	90	NOP
004020E2	:	90	NOP
004020E3	:	90	NOP
004020E4	:	90	NOP
004020E5	:	85C0	TEST EAX,EAX
004020E7	:	DBE2	FCLEX

Fin del capitulo, no fue tan desagradable como pensaba.

Gracias en especial a [Coco](#) por sus tutoriales de visual basic de donde he aprendido lo poco que se de cracking en vb y a [Shoulck](#) por enseñarme alguna cosa de vb.

## Gracias

A todas las personas que colaboran desde el foro de HackxCrack para llevar adelante el curso, tanto los que colaboran aportando sus conocimientos como complemento al curso como a los que postean sus dudas para que aprendamos todos y por supuesto a los moderadores del mismo

A todos los crackers y programadores de los cuales he aprendido y sigo aprendiendo.

A los creadores de crackmes

En especial y sin menospreciar a nadie a [Ricardo Narvaja](#) por su aportación y su trabajo sobre el estudio de las protecciones y sus tutoriales en castellano y a [Makkakko](#) por sus tutoriales con Olly Debugger (Recomendados 100%) y por supuesto a [Shoulck](#) por la ayuda desinteresada que me esta prestando a costa de algo tan preciado como su tiempo.

A ti que me estas leyendo.