

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 paginas de donde bajarse herramientas ni enlaces a paginas de crackers, la intención es que busquéis en Internet todo lo necesario. Seguro que encontraréis 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.

Con este capitulo adjunto las Cracker notes traducidas al castellano

Capitulo 2

Victima

Fantom Begginer challenge (5 crackmes con distintas protecciones) Nivel principiante.

Herramientas

Olly Debugger

Intuición

Objetivo

Repasar la caza de seriales y tomar contacto con nuevos tipos de protección (NAGs, CD CHECKs).

Dejar claro el parcheo de saltos.

Seguir conociendo a Olly.

Al ataque

Lo primero agradecer al creador de estos crackmes su trabajo.

Seguiremos el orden de los crackmes del 1 al 4 y dejare el 5 para que lo destripéis por vuestra cuenta.

Como los pasos esenciales ya los vimos en el capitulo anterior este ira mas rápido.

Crackme 1 En busca del serial perdido

Tomemos este crackme como un repaso al capitulo 1.

Lo ejecutamos introducimos un password cualquiera y vemos



Te equivocaste...

Ya tenemos algo por donde empezar a buscar.

Cerramos el crackme y abrimos Olly.

Cargamos el crackme en el Olly y con click derecho Search for - All referenced text strings.

Vemos esto

R Text strings referenced in CRACKME1:.text		
Address	Disassembly	Text string
00401000	PUSH 0	(Initial CPU selection)
0040106B	MOV DWORD PTR SS:[EBP-C],CRACKME1.00403009	ASCII "MainMenu"
00401072	MOV DWORD PTR SS:[EBP-8],CRACKME1.00403009	ASCII "DLGCLASS"
004010AA	PUSH CRACKME1.00403009	ASCII "MainDialog"
004011B5	PUSH CRACKME1.00403014	ASCII "AboutDialog"
0040128C	PUSH CRACKME1.00403029	ASCII "m0tNaF-EmKCARc"
0040129D	PUSH CRACKME1.00403038	ASCII "Check Status"
004012A2	PUSH CRACKME1.00403045	ASCII "Wrong Password! Keep trying, you'll get it!"
004012B2	PUSH CRACKME1.00403038	ASCII "Check Status"
004012B7	PUSH CRACKME1.00403071	ASCII "You got it! Your now a cracker! :)"

Vemos todas las cadenas de texto del crackme y entre ellas la que buscamos

La cadena que buscamos **Wrong Password! Keep trying, you'll get it!** Esta en la dirección **004012A2**.

Nos fijamos también en otra cadena que nos dice **You got it! Your now a cracker!** En la dirección **004012B7**

Doble Click izquierdo sobre la cadena **Wrong Password! Keep trying, you'll get it!**

Aparecemos aquí

0040127A	. 68 E8030000	PUSH 3E8	ControlID = 3E8 (1000.)
0040127F	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	hWnd
00401282	. E8 6B000000	CALL <JMP.&USER32.GetDlgItemTextA>	GetDlgItemTextA
00401287	. 68 9C304000	PUSH CRACKME1.0040309C	String2 = ""
0040128C	. 68 29304000	PUSH CRACKME1.00403029	String1 = "m0tNaF-EmKARc"
00401291	. E8 BC000000	CALL <JMP.&KERNEL32.lstrcmpA>	lstrcmpA
00401296	. 83F8 00	CMP EAX,0	
00401299	. 74 15	JE SHORT CRACKME1.004012B0	
0040129B	. 6A 30	PUSH 30	Style = MB_OK!MB_ICONEXCLAMATION!MB_APPLMODAL
0040129D	. 68 38304000	PUSH CRACKME1.00403038	Title = "Check Status"
004012A2	. 68 45304000	PUSH CRACKME1.00403045	Text = "Wrong Password! Keep trying, you'll get it!"
004012A7	. 6A 00	PUSH 0	hOwner = NULL
004012A9	. E8 5C000000	CALL <JMP.&USER32.MessageBoxA>	MessageBoxA
004012AE	. EB 13	JMP SHORT CRACKME1.004012C3	
004012B0	. 6A 40	PUSH 40	Style = MB_OK!MB_ICONASTERISK!MB_APPLMODAL
004012B2	. 68 38304000	PUSH CRACKME1.00403038	Title = "Check Status"
004012B7	. 68 71304000	PUSH CRACKME1.00403071	Text = "You got it! Your now a cracker! :)"
004012BC	. 6A 00	PUSH 0	hOwner = NULL
004012BE	. E8 47000000	CALL <JMP.&USER32.MessageBoxA>	MessageBoxA

Encontrando nuestra cadena de texto. Esta parte del código es muy parecida al crackme del capítulo 1. En rojo las instrucciones que nos interesan

Analizando el código.

CALL 00401291

CMP 00401296

JE 00401299 si es igual salta a 004012B0 que es donde debemos llegar para tener el crackme registrado.

Nuestra String 004012A2 la cual tenemos que evitar.

Tenemos claro que nuestro objetivo es hacer aparecer el cartelito de registrado (You got it! Your now a cracker!)

Podemos cambiar en memoria el valor del salto JE por JNE y probar si cambiándolo salta el cartel bueno.

Vemos que si lo invertimos aparecemos registrados.

Recordamos lo visto en el capítulo 1.

Ponemos un Breakpoint en el CALL 00401291 (F2) y corremos el programa (F9).

Introducimos nuestro serial (15151515) y pulsamos el botón Check.

Olly se detiene en el CALL

00401287	. 68 9C304000	PUSH CRACKME1.0040309C	String2 = ""
0040128C	. 68 29304000	PUSH CRACKME1.00403029	String1 = "m0tNaF-EmKARc"
00401291	. E8 BC000000	CALL <JMP.&KERNEL32.lstrcmpA>	lstrcmpA
00401296	. 83F8 00	CMP EAX,0	
00401299	. 74 15	JE SHORT CRACKME1.004012B0	
0040129B	. 6A 30	PUSH 30	Style = MB_OK!MB_ICONEXCLAMATION!MB_APPLMODAL
0040129D	. 68 38304000	PUSH CRACKME1.00403038	Title = "Check Status"
004012A2	. 68 45304000	PUSH CRACKME1.00403045	Text = "Wrong Password! Keep trying, you'll get it!"
004012A7	. 6A 00	PUSH 0	hOwner = NULL
004012A9	. E8 5C000000	CALL <JMP.&USER32.MessageBoxA>	MessageBoxA
004012AE	. EB 13	JMP SHORT CRACKME1.004012C3	
004012B0	. 6A 40	PUSH 40	Style = MB_OK!MB_ICONASTERISK!MB_APPLMODAL
004012B2	. 68 38304000	PUSH CRACKME1.00403038	Title = "Check Status"
004012B7	. 68 71304000	PUSH CRACKME1.00403071	Text = "You got it! Your now a cracker! :)"
004012BC	. 6A 00	PUSH 0	hOwner = NULL
004012BE	. E8 47000000	CALL <JMP.&USER32.MessageBoxA>	MessageBoxA

Olly detenido en el CALL

Pulsamos F7 o Debug - Step into una sola vez para entrar dentro del CALL y ver que pasa ahí dentro



Dentro del CALL vemos nuestra cadena 15151515 y otra más que podría ser el serial correcto

Al pulsar F7 aparecemos en la dirección 00401352.

Pulsamos F8 y bajamos por las direcciones una a una (cada vez que pulsemos F8 avanzamos una línea de código) fijándonos en la ventana de Registros (derecha).

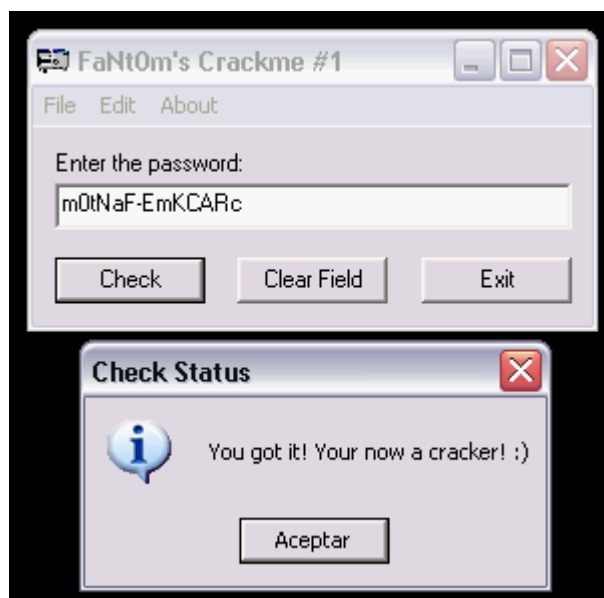
Como en el crackme del capítulo anterior vemos las cadenas falsa (la nuestra 15151515)

y

la buena (el password correcto

m0tNaF-EmKCARc).

Lo apuntamos, cerramos Olly, ejecutamos el crackme e introducimos el serial y...



serial correcto

Crackme registrado con el número de registro correcto.

Bien un pequeño apunte aun a riesgo de ser pesado (que lo soy) pero en estos capítulos tenéis que tener las cosas claras para poder ir más ligeros después:

F8 Step over vamos examinando las instrucciones una a una sin entrar en CALLs.

F7 Step into vamos examinando las instrucciones entrando en los CALLs.

Cuando estemos parados en un Breakpoint y tengamos que "meternos dentro" para examinarlo pulsaremos primero y una sola vez F7 y

Continuaremos con F8 poco a poco y observando la ventana Registers en el Olly.

Esto nos servirá para crackmes que no canten tanto enseñando el serial a la primera de cambio como es el caso de estos dos primeros crackmes que hemos visto.

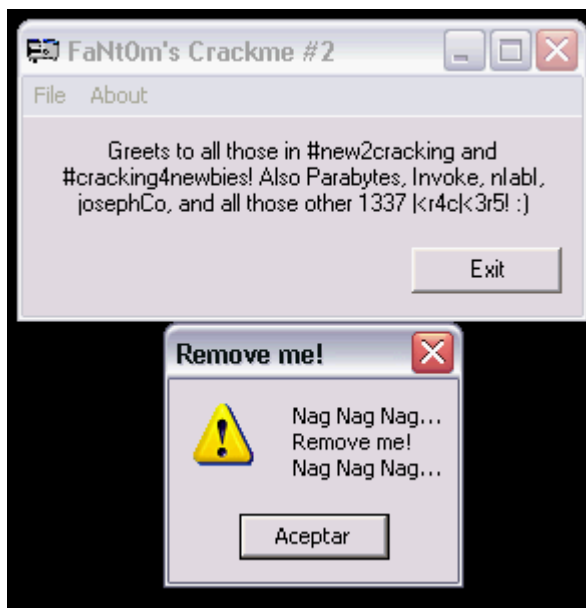
Para ver el serial bueno tendremos que utilizar [F7 para entrar en el CALL y examinarlo con F8 instrucción a instrucción](#) hasta ver en la ventana de Registers la comparación de nuestro serial falso con el verdadero o alguna String que aparentemente pueda ser el serial correcto del crackme.

Esto se conoce como trazar o tracear.

Continuemos aprendiendo juntos.

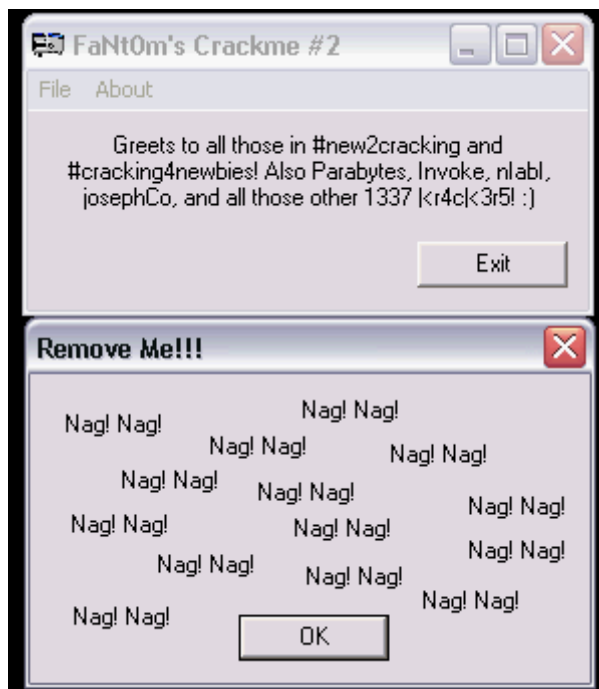
Crackme 2 NAG NAG

Lo ejecutamos y vemos esto



Ummmmmmmmmm...

Aceptamos y pasamos a ver esto otro



Otra NAG

El cometido de este crackme es enseñarnos a librarnos de esas dos ventanas llamadas NAG.

Os recordaran las advertencias o recordatorios de algún que otro programa comercial que tanto estorban cuando lo iniciamos.

Para solucionarlo necesitaremos nuestros pequeños conocimientos sobre saltos (espero que al menos sepáis que las instrucciones de salto están en el capítulo 0 y por supuesto en Google).

Cargamos el crackme en el Olly y buscamos las Strings

CPU - main thread, module CRACKME2		
R Text strings referenced in CRACKME2:.text		
Address	Disassembly	Text string
00401000	PUSH 0	(Initial CPU selection)
00401068	MOV DWORD PTR SS:[EBP-C],CRACKME2.00403009	ASCII "MAINMENU"
00401072	MOV DWORD PTR SS:[EBP-8],CRACKME2.00403009	ASCII "DLGCLASS"
004010AA	PUSH CRACKME2.00403009	ASCII "MAINDIALOG"
004010D2	PUSH CRACKME2.00403033	ASCII "Remove me!"
004010D7	PUSH CRACKME2.0040303E	ASCII "Nag Nag Nag...!Nag Nag Nag..."
004010ED	PUSH CRACKME2.00403029	ASCII "NAGDIALOG"
00401188	PUSH CRACKME2.00403014	ASCII "ABOUTDIALOG"

Vamos a la dirección 004010D2 Remove me!

004010A8	. 6A 00	PUSH 0	hOwner = NULL
004010AA	. 68 09304000	PUSH CRACKME2.00403009	pTemplate = "MAINIALOG"
004010AF	. FF35 6C304000	PUSH DWORD PTR DS:[40306C]	hInst = NULL
004010B5	. E8 B8010000	CALL <JMP.&USER32.CreateDialogParamA>	CreateDialogParamA
004010BA	. 8945 B0	MOV DWORD PTR SS:[EBP-50],EAX	
004010BD	. FF75 14	PUSH DWORD PTR SS:[EBP+14]	ShowState
004010C0	. FF75 B0	PUSH DWORD PTR SS:[EBP-50]	hWnd
004010C3	. E8 F8010000	CALL <JMP.&USER32.ShowWindow>	ShowWindow
004010C8	. FF75 B0	PUSH DWORD PTR SS:[EBP-50]	hWnd
004010CB	. E8 FC010000	CALL <JMP.&USER32.UpdateWindow>	
004010D0	. 6A 30	PUSH 30	Style = MB_OK MB_ICONEXCLAMATION MB_APPLMODAL
004010D2	. 68 33304000	PUSH CRACKME2.00403033	Title = "Remove me!"
004010D7	. 68 3E304000	PUSH CRACKME2.0040303E	Text = "Nag Nag Nag...[?]Remove me![?]Nag Nag Nag..."
004010DC	. 6A 00	PUSH 0	hOwner = NULL
004010DE	. E8 C5010000	CALL <JMP.&USER32.MessageBoxA>	
004010E3	. 6A 00	PUSH 0	lParam = NULL
004010E5	. 68 10124000	PUSH CRACKME2.00401210	DlgProc = CRACKME2.00401210
004010EA	. FF75 B0	PUSH DWORD PTR SS:[EBP-50]	hOwner
004010ED	. 68 29304000	PUSH CRACKME2.00403029	pTemplate = "NAGIALOG"
004010F2	. FF35 6C304000	PUSH DWORD PTR DS:[40306C]	hInst = NULL
004010F8	. E8 87010000	CALL <JMP.&USER32.DialogBoxParamA>	DialogBoxParamA
004010FD	. 6A 00	PUSH 0	MsgFilterMax = 0
004010FF	. 6A 00	PUSH 0	MsgFilterMin = 0
00401101	. 6A 00	PUSH 0	hWnd = NULL
00401103	. 8D45 B4	LEA EAX,DWORD PTR SS:[EBP-4C]	pMsg
00401106	. 50	PUSH EAX	GetMessageA
00401107	. E8 8A010000	CALL <JMP.&USER32.GetMessageA>	
0040110C	. 83F8 00	CMPL EAX,0	

En la imagen superior podemos ver las String de la NAG, un CALL (004010CB marcado en rojo) y mas abajo en la dirección 004010FD el principio de otra sección de código.

[El principio de las instrucciones Olly lo marca con el símbolo ➤](#)

Intuimos por lo que vemos que el CALL 004010CB llama a la creación de las NAGs pero no vemos ningún salto para poder pasar al principio de la siguiente instrucción donde intuimos que habremos pasado la zona de las NAGs.

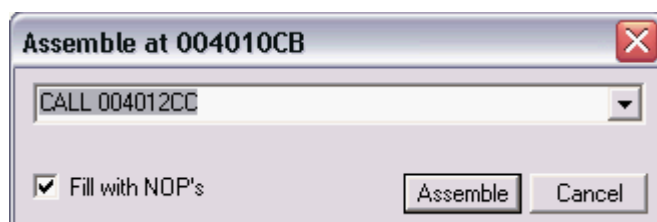
Podemos intentar crear un salto que nos lleve donde queremos ir.

Habíamos visto dos tipos de salto JE saltar si es igual y JNE saltar si no es igual (saltos condicionales), ahora veremos **JMP saltar siempre** (salto incondicional).

Vamos a intentar cambiar ese CALL 004010CB por un salto **JMP (salta siempre)** para llegar al principio de la siguiente instrucción y ver si no aparecen las NAGs como sospechamos.

Probemos, nos colocamos encima del CALL que creemos que llama a las NAG y doble click o click derecho - Assemble.

Aparece la ventanita de la imagen inferior con las instrucciones del CALL

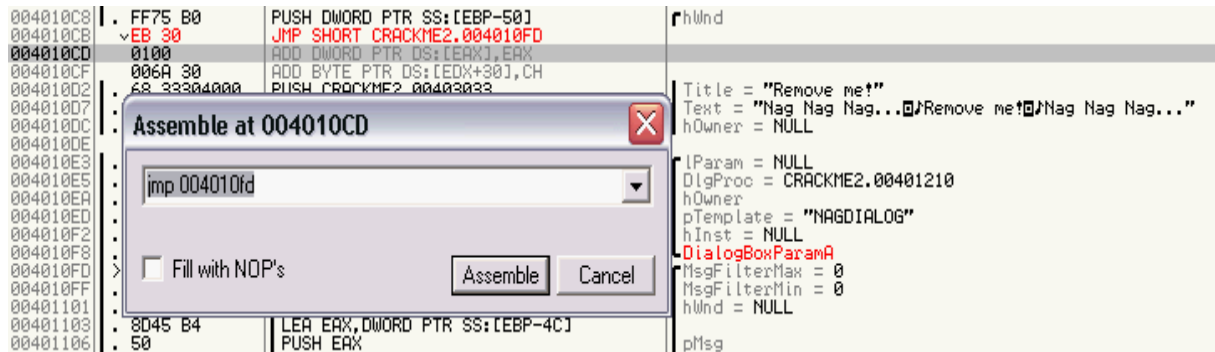


Lo borramos y escribimos en ella **JMP 004010FD** (004010FD es el principio de la instrucción donde queremos saltar) **desmarcamos la casilla Fill with NOP's** y

pulsamos Assemble el CALL cambia a **JMP SHORT CRACKME2.004010FD**

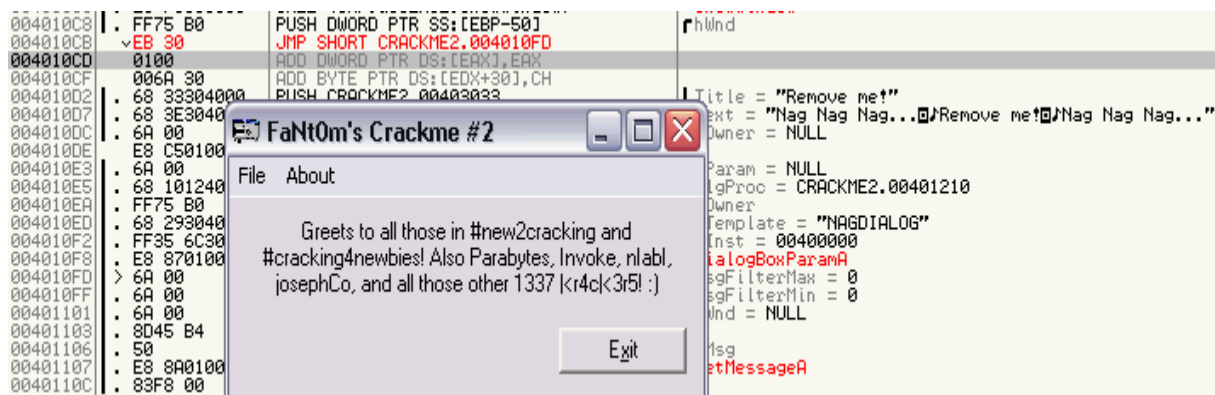


NOP (No Operation) quedamos con lo que significa de momento ya veremos los NOP mas adelante.



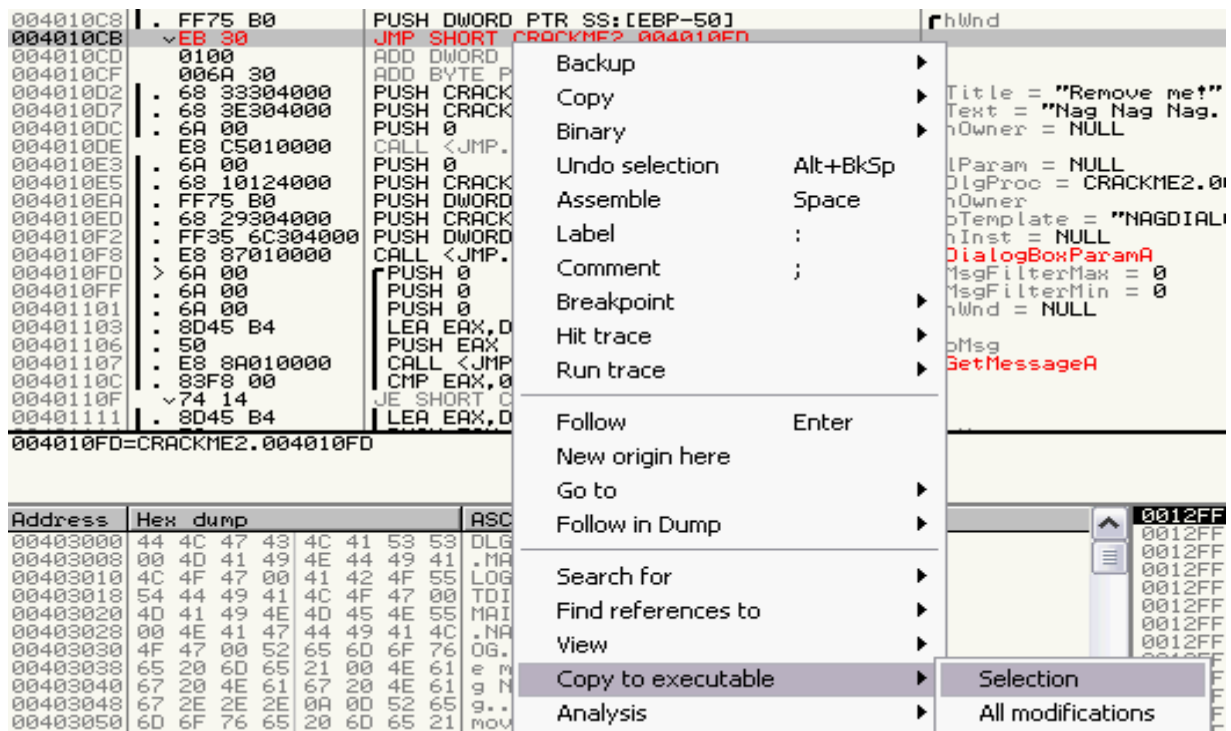
Vemos los cambios en memoria Hemos cambiado un CALL por un JMP

Corremos el programa (F9) y aparece esta ventana felicitándonos por haberlo conseguido.



Hemos creado nuestro propio salto y el resultado ha sido el esperado: NAGs eliminadas

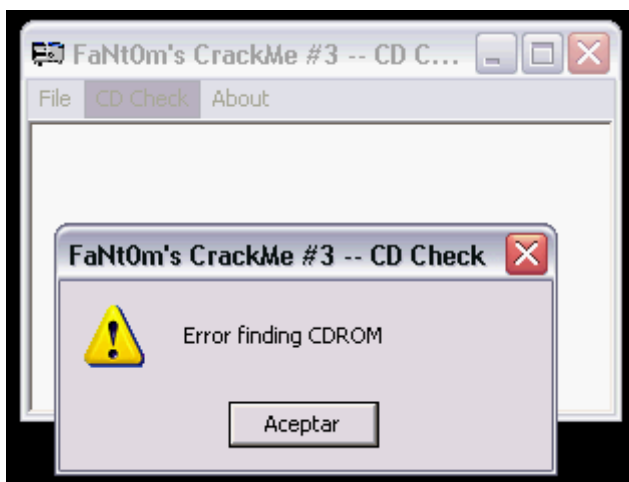
Lo hemos probado en memoria, ahora efectuemos los cambios en el ejecutable



Aceptamos los cambios en el ejecutable como vimos en el capítulo 1 y crackme resuelto.

Crackme 3 CD CHECK

Es la típica protección de algunos juegos, la famosa frase "no se encuentra el CD ROM". Ejecutemos el crackme y pulsemos en CD Check



Vaya por Dios...

Quedémonos con la frase **Error finding CDROM**.

Cerramos el crackme, abrimos Olly y buscamos la String.

```

CPU - main thread, module CRACKME3
0040110C .vEB 10      JMP SHORT CRACKME3.004011EE
0040110E > 6A 00      PUSH 0
004011E0 . 6A 00      PUSH 0
004011E2 . 6A 10      PUSH 10
004011E4 . FF75 08    PUSH DWORD PTR SS:[EBP+8]
004011E7 . E8 90000000 CALL <JMP.&USER32.SendMessageA>
004011EC .vEB 00      JMP SHORT CRACKME3.004011EE
004011EE > B8 01000000 MOV EAX,1
004011F3 . C9        LEAVE
004011F4 . C2 1000    RETN 10
004011F7 $ 55        PUSH EBP
004011F8 . 8BEC      MOV EBP,ESP
004011FA . 6A 00      PUSH 0
004011FC . E8 99000000 CALL <JMP.&KERNEL32.GetDriveTypeA>
00401201 . 83F8 05    CMP EAX,5
00401204 <v74 17    JE SHORT CRACKME3.0040121D
00401206 . 6A 30      PUSH 30
00401208 . 68 14304000 PUSH CRACKME3.00403014
0040120D . 68 40304000 PUSH CRACKME3.00403040
00401212 . 6A 00      PUSH 0
00401214 . E8 51000000 CALL <JMP.&USER32.MessageBoxA>
00401219 . C9        LEAVE
0040121A . C2 0400    RETN 4
0040121D > 6A 00      PUSH 0
0040121F . 68 14304000 PUSH CRACKME3.00403014
00401224 . 68 54304000 PUSH CRACKME3.00403054
00401229 . 6A 00      PUSH 0
0040122B . E8 3A000000 CALL <JMP.&USER32.MessageBoxA>
00401230 . C9        LEAVE

```

[IParam = 0
 wParam = 0
 Message = WM_CLOSE
 hWnd
 SendMessageA

[RootPathName = NULL
 GetDriveTypeA

[Style = MB_OK!MB_ICONEXCLAMATION!MB_APPLMODAL
 Title = "FaHt0m's CrackMe #3 -- CD Check"
 Text = "Error finding CDROM"
 hWnd = NULL
 MessageBoxA

[Style = MB_OK!MB_APPLMODAL
 Title = "FaHt0m's CrackMe #3 -- CD Check"
 Text = "Found a CDROM! Good job!"
 hWnd = NULL
 MessageBoxA

Las tres instrucciones de siempre, vamos bien

La vemos y justo encima un CALL un CMP y un JE

Pues sencillo no ¿?

Invertiremos ese salto para que nos lleve a la zona buena 0040121D que es donde se llama al MessageBox que nos dice que encontramos el CD ROM

El CALL llama a la API GetDriveTypeA.

Mirad información en la red sobre las APIs de Windows. En las Cracker notes que acompañan a este capítulo tenéis las APIs de Windows mas frecuentes

```

004011FC . E8 99000000 CALL <JMP.&KERNEL32.GetDriveTypeA>
00401201 . 83F8 05    CMP EAX,5
00401204 <v74 17    JE SHORT CRACKME3.0040121D
00401206 . 6A 30      PUSH 30
00401208 . 68 14304000 PUSH CRACKME3.00403014

```

Assemble at 00401204

JE SHORT 0040121D

☐ Fill with NOP's

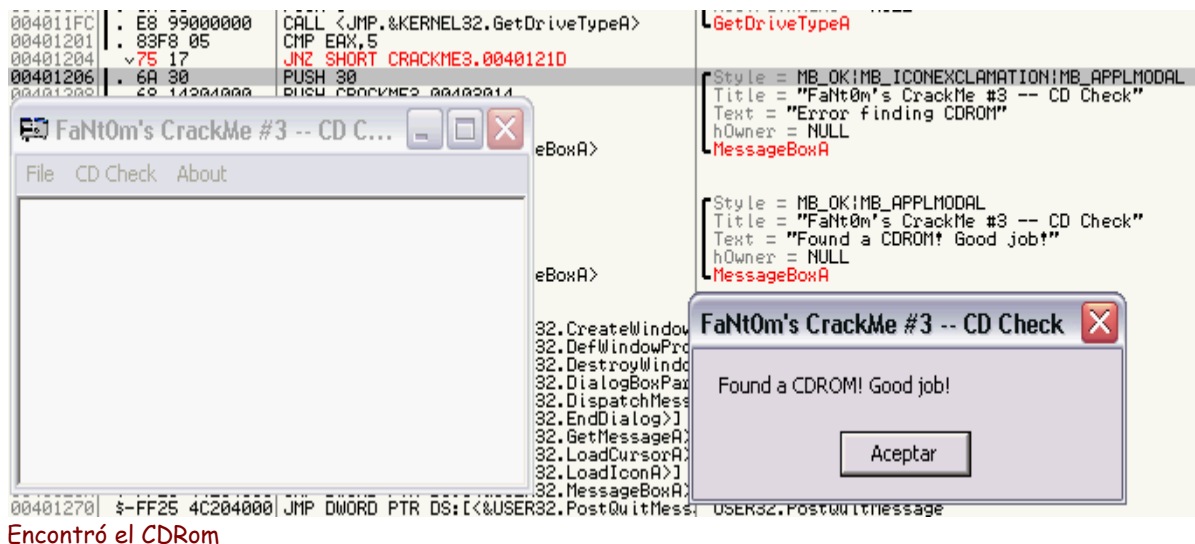
Assemble Cancel

[GetDriveTypeA

[Style = MB_OK!MB_ICONEXCLAMATION!MB_APPLMODAL
 Title = "FaHt0m's CrackMe #3 -- CD Check"
 Text = "Error finding CDROM"
 hWnd = NULL
 MessageBoxA

[Style = MB_OK!MB_APPLMODAL
 Title = "FaHt0m's CrackMe #3 -- CD Check"
 Text = "Found a CDROM! Good job!"
 hWnd = NULL
 MessageBoxA

Probamos con F9



Encontró el CD Rom

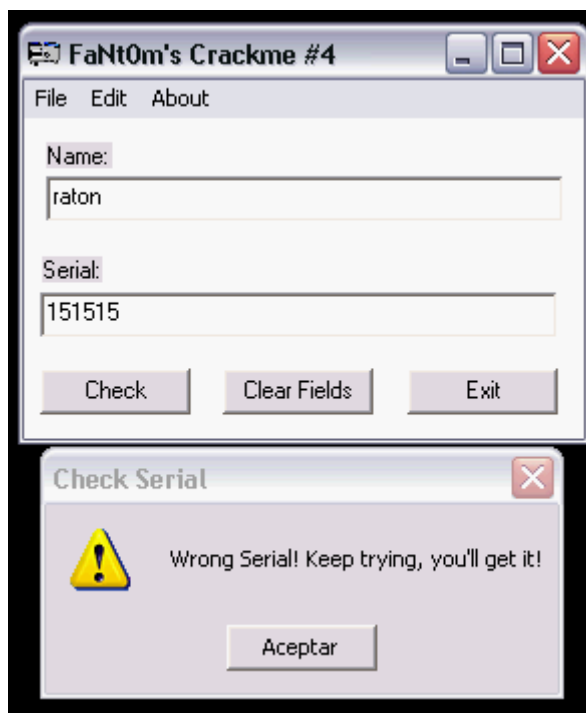
Guardamos los cambios en ejecutable con Copy to executable - Selection y crackme resuelto.

Crackme 4 Nombre y Serial Number

Este crackme varia un poco sobre lo visto, aunque la base para resolverlo es la misma que la del crackme 1.

El crackme crea un serial distinto para cada nombre.

Ejecutando y probando



Parece que tampoco acertamos esta vez

Busquemos la String en el Olly mirando las Strigs references y vayamos a la dirección

donde esta la cadena.

Vemos lo de siempre: LLAMADA - COMPARACION - SALTO (CALL- CMP- JE) justo antes del MessageBox malo.

CPU - main thread, module CRACKME4			
0040122B	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	hWnd
0040122E	. E8 4B010000	CALL <JMP.&USER32.GetDlgItemTextA>	GetDlgItemTextA
00401233	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	
00401236	. E8 BE000000	CALL CRACKME4.004012F9	
0040123B	. 83F8 00	CMP EAX,0	
0040123E	. 74 15	JE SHORT CRACKME4.00401255	
00401240	. 6A 40	PUSH 40	
00401242	. 68 29304000	PUSH CRACKME4.00403029	[Style = MB_OK MB_ICONASTERISK MB_APPLMODAL
00401247	. 68 60304000	PUSH CRACKME4.00403060	Title = "Check Serial"
0040124C	. 6A 00	PUSH 0	Text = "You got it! Congrats! :)"
0040124E	. E8 49010000	CALL <JMP.&USER32.MessageBoxA>	hOwner = NULL
00401253	. 74 13	JMP SHORT CRACKME4.00401268	MessageBoxA
00401255	. 6A 30	PUSH 30	
00401257	. 68 29304000	PUSH CRACKME4.00403029	[Style = MB_OK MB_ICONEXCLAMATION MB_APPLMODAL
0040125C	. 68 60304000	PUSH CRACKME4.00403060	Title = "Check Serial"
00401261	. 6A 00	PUSH 0	Text = "Wrong Serial! Keep trying, you'll get it!"
00401263	. E8 34010000	CALL <JMP.&USER32.MessageBoxA>	hOwner = NULL
00401268	. 68 E8030000	PUSH 3E8	MessageBoxA
0040126D	. 6A 00	PUSH 0	ControlID = 3E8 (1000.)

Las Strings y los MessageBox con sus llamadas y saltos correspondientes

Pero aquí vamos a aprender una cosa que no habíamos visto en anteriores crackmes.

Antes hice un pequeño apunte: [El principio de las instrucciones Olly lo marca con el símbolo >](#)

Ahora veréis porque hice esta reseña.

Nos colocamos encima del comienzo de la instrucción 00401255 > que es la que genera la ventana de no registrados y pulsamos Control+R o click derecho Find references to - Selected command

CPU - main thread, module CRACKME4			
00401253	. 74 13	JMP SHORT CRACKME4.00401268	
00401255	. 6A 30	PUSH 30	
00401257	. 68 29304000	PUSH CRACKME4.00403029	[Style = MB_OK MB_ICONEXCLAMATION MB_APPLMODAL
0040125C	. 68 60304000	PUSH CRACKME4.00403060	Title = "Check Serial"
00401261	. 6A 00	PUSH 0	Text = "Wrong Serial! Keep trying, you'll get it!"
00401263	. E8 34010000	CALL <JMP.&USER32.MessageBoxA>	hOwner = NULL
00401268	. 68 E8030000	PUSH 3E8	MessageBoxA
0040126D	. 6A 00	PUSH 0	ControlID = 3E8 (1000.)
00401270	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	hWnd
00401275	. E8 03010000	CALL <JMP.&USER32.GetDlgItemTextA>	GetDlgItemTextA
0040127A	. 50	PUSH 0	hWnd
0040127C	. E8 3F010000	CALL <JMP.&USER32.SetFocus>	SetFocus
0040127F	. 74 15	JMP SHORT CRACKME4.00401294	
00401280	. FF75 14	PUSH DWORD PTR SS:[EBP+14]	[Param
00401283	. FF75 10	PUSH DWORD PTR SS:[EBP+10]	wParam
00401286	. FF75 0C	PUSH DWORD PTR SS:[EBP+0C]	lParam
00401289	. FF75 08	PUSH DWORD PTR SS:[EBP+08]	hWnd
0040128E	. E8 CC000000	CALL CRACKME4.0040128E	DefWindowProcA
00401291	. C9	RET	
00401294	. C2 1000	RET 1000	
00401297	. 55	PUSH 0	
00401299	. 8BEC	MOV ECX,EBP	
0040129B	. 837D 0C 10	CMP DWORD PTR [EBP+0C10],0	
0040129F	. 75 0C	JNZ CRACKME4.004012A1	
004012A1	. 6A 00	PUSH 0	
004012A3	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	

Jump from 0040123E

Address Hex dump

00403000 44 4C 47 43 4C 41 53

Aparece esta ventana

En rojo vemos la dirección donde estamos o sea al principio de la rutina que llama a la

MessageBox mala.

Encima la dirección de un salto condicional JE (si hacemos doble click sobre el nos llevara hasta el salto).

CPU - main thread, module CRACKME4		
References in CRACKME4:.text to 00401255		
Address	Disassembly	Comment
0040123E	JE SHORT CRACKME4.00401255	
00401255	PUSH 30	(Initial CPU selection)

Bien, en estos crackmes sencillos y con pocas líneas de código se ve bien y se pueden seguir las instrucciones fácilmente y al lado de la cadena mala vemos el salto que nos manda hasta ella o que hace que la evitemos (el propósito de los crackmes es ese, para irnos familiarizándonos con el código), pero en crackmes mas avanzados o en programas comerciales a veces el salto que nos lleva a la zona de registrado o no registrado esta lejos y aunque localicemos la cadena mala no veremos el salto que nos lleva hasta ella.

Para eso tenemos Control+R o Find references to Selected command.

Traducción ratonil de Find references to Selected command = Olly buscame desde donde se llama a esta instrucción.

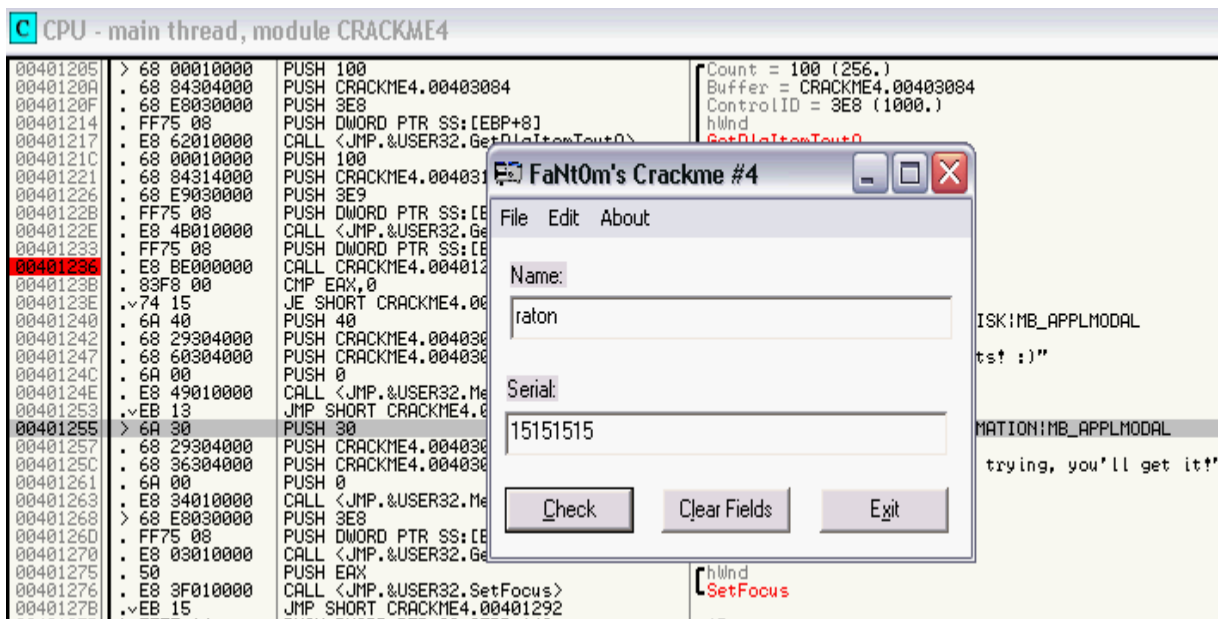
En este caso buscamos desde donde se llama a la instrucción 00401255 que es donde se empieza a generar la ventana mala de no registrado y el Olly con Control+R nos dice que se la llama desde 0040123E.

Si nos desplazamos a JE 0040123E encontramos encima el CALL 00401236 Y el CMP 0040123B y vemos que ese es el salto que debemos cambiar (si queremos invertir el salto) o vemos el CALL donde debemos poner el Breakpoint para cazar el serial bueno.

00401236	. E8 BE000000	CALL CRACKME4.004012F9
0040123B	. 83F8 00	CMP EAX,0
0040123E	. 74 15	JE SHORT CRACKME4.00401255

Ponemos Breakpoint en el CALL para cazar el serial y ejecutamos el programa F9

Introducimos un nombre y un serial cualquiera y pulsamos check



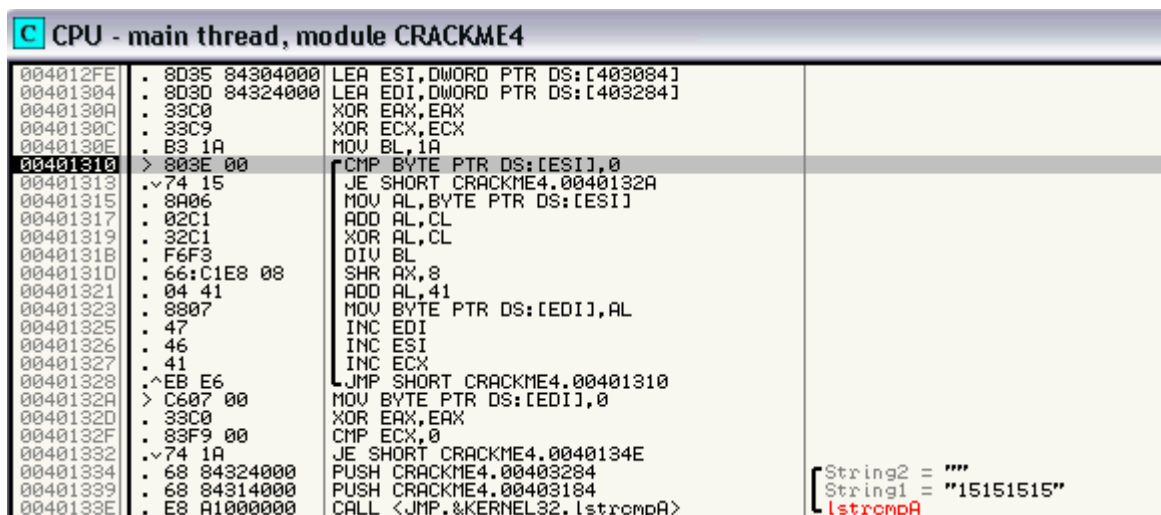
Olly para en el CALL y entramos en el pulsando F7

Cuando estemos parados en un Breakpoint y tengamos que "meternos dentro" para examinarlo pulsaremos **primero y una sola vez F7** y

Continuaremos con F8 poco a poco y observando la ventana Registers en el Olly.

Si ya se que soy pesado.

Seguimos examinando con F8 y llegamos a esta parte del código (00401310) donde el programa realiza unas operaciones para comprobar si el serial generado para nuestro nombre coincide con el que introducimos en la ventana de texto (15151515)

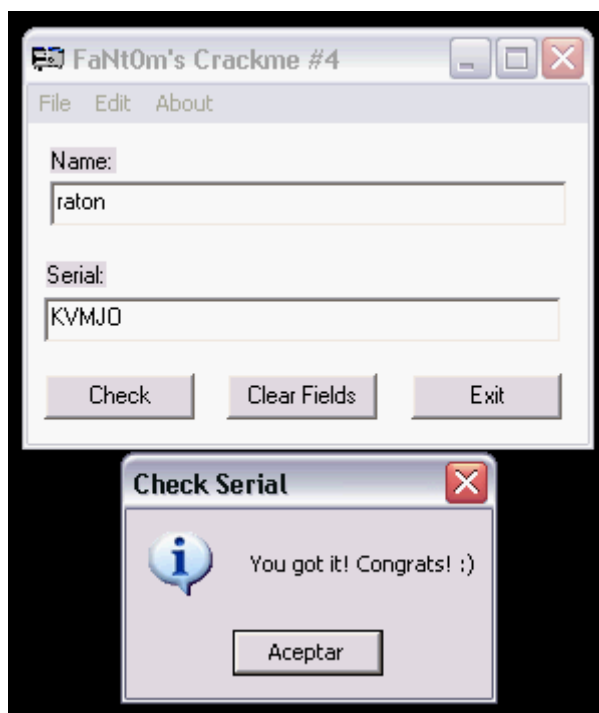


Seguimos con F8 poco a poco y salimos del corchete donde se opera para generar el serial correcto para nuestro nombre y vemos que en string2 que se ha generado una cadena KVMJO (ojo esta cadena se genero para mi nombre: ratón para vosotros será otra distinta según el nombre que introdujerais en la caja de texto)

CPU - main thread, module CRACKME4		
004012FE	. 8D35 84304000	LEA ESI,DWORD PTR DS:[403084]
00401304	. 8D3D 84324000	LEA EDI,DWORD PTR DS:[403284]
0040130A	. 33C0	XOR EAX,EAX
0040130C	. 33C9	XOR ECX,ECX
0040130E	. B3 1A	MOV BL,1A
00401310	> 803E 00	CMP BYTE PTR DS:[ESI],0
00401313	~ 74 15	JE SHORT CRACKME4.0040132A
00401315	. 8A06	MOV AL,BYTE PTR DS:[ESI]
00401317	. 02C1	ADD AL,CL
00401319	. 32C1	XOR AL,CL
0040131B	. F6F3	DIV BL
0040131D	. 66:C1E8 08	SHR AX,8
00401321	. 04 41	ADD AL,41
00401323	. 8B07	MOV BYTE PTR DS:[EDI],AL
00401325	. 47	INC EDI
00401326	. 46	INC ESI
00401327	. 41	INC ECX
00401328	^ EB E6	JMP SHORT CRACKME4.00401310
0040132A	> C607 00	MOV BYTE PTR DS:[EDI],0
0040132D	. 33C0	XOR EAX,EAX
0040132F	. 83F9 00	CMP ECX,0
00401332	~ 74 1A	JE SHORT CRACKME4.0040134E
00401334	. 68 84324000	PUSH CRACKME4.00403284
00401339	. 68 84314000	PUSH CRACKME4.00403184
0040133E	. E8 A1000000	CALL <JMP.&KERNEL32.lstrcmpA>
00401343	. 83F8 00	CMP EAX,0
00401346	~ 74 04	JE SHORT CRACKME4.0040134C
00401348	. 33C0	XOR EAX,EAX
0040134A	^ EB 02	JMP SHORT CRACKME4.0040134E

String2 = "KUMJO"
String1 = "15151515"
lstrcmpA

Apuntamos la cadena que se genero y cerramos el Olly
Abrimos el crackme y probamos



Si probáis a introducir otro nombre y ese mismo serial veréis que os sale el cartel malo.
Un serial para cada nombre, algo distinto a lo visto en crackmes anteriores.

Bueno llegamos al final de este segundo capitulo.

Repasemos lo que hemos visto:

Buscar seriales correctos poniendo Breakpoints en los CALLs correspondientes basándonos en el texto que nos aparece en la ventana de no registrado.

Eliminar NAGs creando un salto a medida.

Saltarnos la protección CD Check invirtiendo saltos.

Buscar un serial específico para nuestro nombre poniendo Breakpoint en el CALL correspondiente basándonos en el texto que nos aparece en la ventana de no registrado.

Saber que pasa cuando tecleamos F7 y F8.

Encontramos la instrucción NOP.

Nos hemos fijado en la llamada de un CALL a una API de Windows.

En vuestra mano esta buscar información en la red sobre lo que no he explicado.

Intentad resolver los crackmes sin el tutorial.

Practicad con más crackmes.

Para el siguiente capitulo necesitaremos DeDe, Peid y Olly.

De regalo tenéis el crackme 5 de Fantom que engloba todas la protecciones vistas en los 4 anteriores, con lo visto aquí debéis ser capaces de resolverlo.

Gracias

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%).

Ratón Enero 2004