las mil y una formas de CRackear un Software

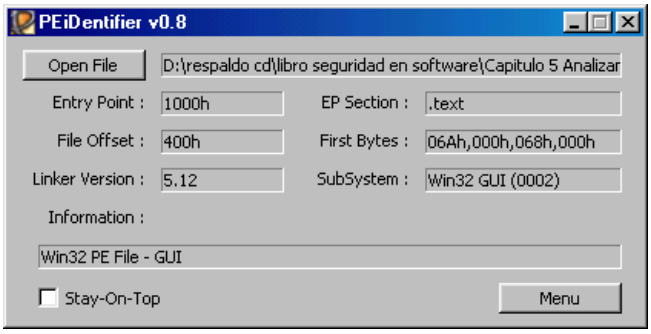
(Método de desensamblado de Ejecutable)

Después de un largo rato de no escribir, bueno pues auqi els mando un tutorial donde explico como se puede crackear un software de diferentes formas.

EJEMPLO NO. 1

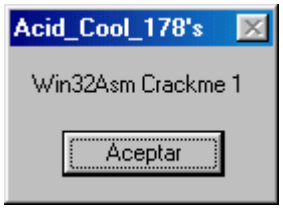
Ahora veremos unos ejemplos de como los crackers atacan las Nag´s Screen´s, para este ejemplo usaremos el Crackme No. 1 de Acid_Cool, comencemos.

Lo primero que tendremos que hacer es analizar el crackme para saber si esta empacado o en su caso en que lenguaje fue compilado, para eso utilizaremos el programa llamado " PEid v 0.8 " y nos muestra las siguiente información:

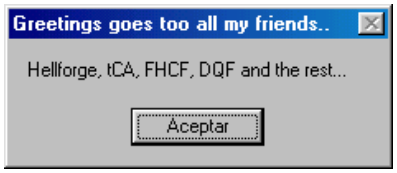


Como podemos ver no dice mucho el PEid v 0.8, solo nos muestra que es un Win32 PE File-Gui, estará hecho en assembler puro?, yo creo que si, OK ahora vamos y lo ejecutamos para ver que es lo que nos muestra:

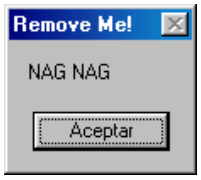
Primera ventana:



Segunda ventana:



Tercera ventana:

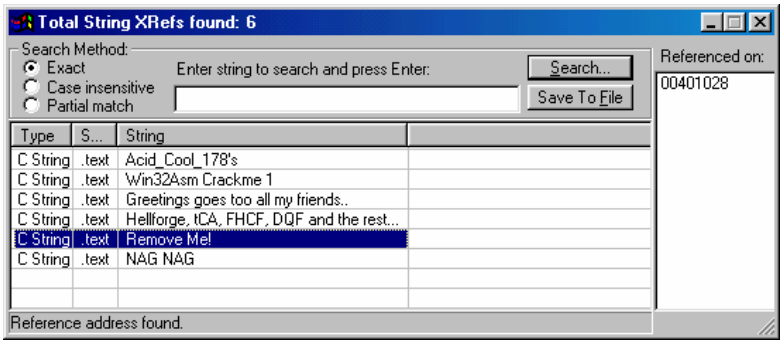


OK, ya tenemos el objetivo bien estudiado, se supone que debemos quitar esta ultima Nag Screen, ahora toca el turno de seleccionar las utilerías que usaremos para poder cumplir nuestro cometido de eliminar esa Nag.

1. **Bdasm v 1.0**
2. **Hex-Editor**
3. **ProcDump v1.6.2**

Ahora empecemos con este reto, se dieron cuenta de que en la tercera ventana **(Message Box)** aparece un String **"Remove me" "NAG NAG",** ahora desensamblemos el Crackme con el **Bdasm® v 1.0**, ya desensamblado damos clic en el botón donde nos muestra las String References: 

El cual nos mostrar la siguiente ventana:



Regresemos al código desensamblado y veremos el siguiente código:

```
**************************************************************************************

:0040101F 6A00                  push 00000000

* Reference To: USER32.MessageBoxA, Ord:01BBh
                                 |
:00401021 E81A000000            Call 00401040
:00401026 6A00                  push 00000000

* Possible StringData Ref from Data Obj ->"Remove Me!"
                                 |
:00401028 6871304000            push 00403071

* Possible StringData Ref from Data Obj ->"NAG NAG"
                                 |
:0040102D 687C304000            push 0040307C
:00401032 6A00                  push 00000000

* Reference To: USER32.MessageBoxA, Ord:01BBh
                                 |
:00401034 E807000000            Call 00401040
:00401039 6A00                  push 00000000

* Reference To: KERNEL32.ExitProcess, Ord:0075h
                                 |
:0040103B E806000000            Call 00401046

********************************** Fin del Código ************************************
```

OK, ahora como podemos ver esta situación podemos resolverla de cinco formas diferentes:

1. El **Push 00000000** en la dirección de memoria **00401026** podríamos hacerlo saltar a la parte del código donde termina el proceso **" ExitProcess"** y así el crackme no mostrará la ventana del Nag.

2. Podemos también eliminar la call que llama a la Nag en la dirección de memoria **00401034**

3. Hacer que la **Call** en la dirección **00401034** salte a la parte del código donde termina el proceso **" ExitProcess"** y así el crackme no mostrará la ventana del Nag.

4. Creación de un Loader para parchar la NAg Screen en Memoria

5. Injertar código en el Crackme para saltar la NAg Screen.

Ya sabemos las formas de poder reversear esta Nag, la pregunta seria:

¿Que método de los 4 arriba descritos seria el más óptimo para reversear el crackme?

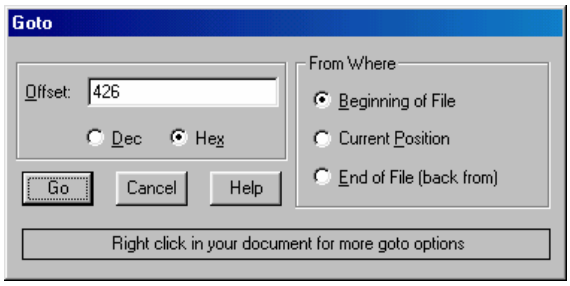
Es difícil de decidir, bueno para no tener que decidir, vamos a reversear este Crackme con los 5 métodos arriba descritos.

Método 1
(Push al ExitProcess)

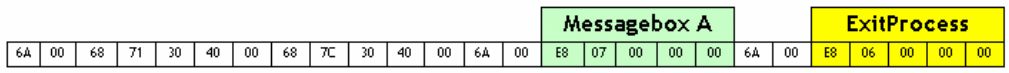
El **Push 00000000** en la dirección de memoria **00401026** vamos a hacer que salte a la parte del código donde termina el proceso **" ExitProcess"** y así el crackme no mostrará la ventana del Nag.

Lo primero que haremos ir al **Bdasm**® y nos posicionamos en la dirección **401026** y le sacamos su offset que es:

Después de esto, vamos y cerramos el desensamblador y abrimos el Crackme con el Editor Hexadecimal, damos control g y nos aparece la ventana:



Insertamos el offset y damos clic en GO
Y vemos esto:

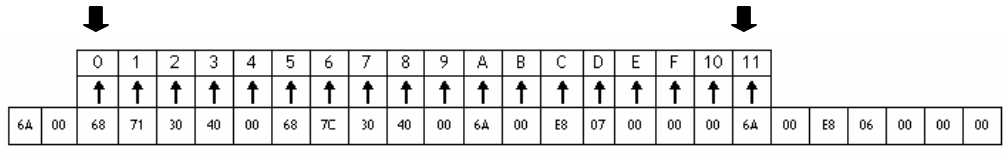


Ahora lo siguiente a hacer es remplazar el primer **"6A 00"** por un salto incondicional "**EB 00**" pero para hacer eso tenemos que saber cuantos bytes tenemos que saltar para llegar al **" ExitProcess"** para esto usamos la siguiente técnica de contar los

bytes donde empieza el **“ Message Box “** al comienzo del **“ ExitProcess”**, no olvides que contaremos en forma hexadecimal.

0 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12 13 14 .....

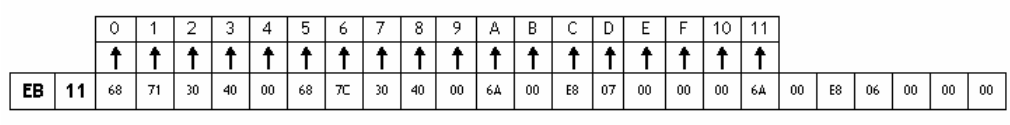
Contamos de aquí hasta aquí que es el principio del **“ ExitProcess”**



% % %

Valor del Salto a remplazar parte de la instrucción a alcanzar

Como podemos ver al contar en hexadecimal obtenemos el valor **11** que cambiaremos por un salto incondicional **EB 11** para así alcanzar el **“ ExitProcess “** entonces quedaría así:



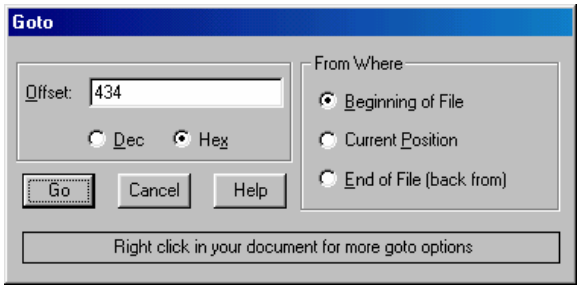
OK, remplacemos en el editor hexadecimal **"6A00"** por **"EB11**" en la Address **"00401026"** salvamos los cambios y ejecutemos de nuevo el Crackme y ya no aparece la Nag Screen, primer método realizado con éxito.

Método 2
(Eliminando la CAll)

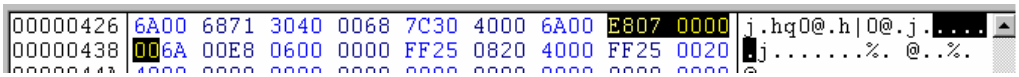
OK, vamos a ver el siguiente método en el cual eliminaremos la call que genera la Nag Screen remplazando con **NOP** los bytes a cambiar.

Como sabemos la NAg es creada en la dirección **" 00401034"** le sacamos su offset que es 434h:

Vamos y abrimos en el editor hexadecimal el Crackme y damos **CONTROL G** y ponemos lo siguiente:



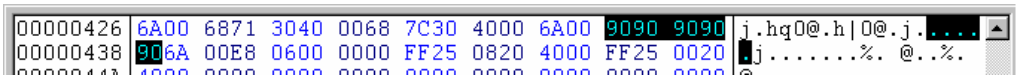
De ahí damos clic en **Go** y nos manda a la parte del código en la cual cambiaremos:



Como podemos ver la **Call** consta de 5 bytes que son:

E8 07 00 00 00 = 5 Bytes

Por lo tanto tenemos que poner 5 bytes de cancelación que en ensamblador se ocupa **NOP** y en formato hexadecimal el **NOP** es **90** tendremos que poner como se muestra en la siguiente figura:



Guardamos los cambios, ejecutamos de nuevo el Crackme y podemos ver que Nag Screen Eliminada.

Método 3
(JMP la Call to ExitProcess)

OK, ahora vamos a seguir el mismo proceso que el método 1, pero ahora haremos que la **CALL** que esta en la dirección **00401034** salte hasta el **" ExitProcess "** y así poder eliminar la Nag Screen.

MessageBoxA					Push		Exit Process					MessgeBoxA						ExitProcess					
E8	07	00	00	00	6A	00	E8	06	00	00	00	FF	25	08	20	40	00	FF	25	00	20	40	00

Por lo tanto contaremos desde donde empieza el **PUSH** hasta donde empieza el **ExitProcess**:

MessageBoxA					Push		Exit Process					MessgeBoxA						ExitProcess					
					0	1	2	3	4	5	6	7	8	9	A	B	C	D					
					↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑					
E8	07	00	00	00	6A	00	E8	06	00	00	00	FF	25	08	20	40	00	FF	25	00	20	40	00

Por lo tanto ya tenemos el valor que es 0D, vamos al Editor Hexadecimal y cambiamos:

```
00000426 6A00 6871 3040 0068 7C30 4000 6A00 E807 0000 j.hq0@.h|0@.j.....
00000438 006A 00E8 0600 0000 FF25 0820 4000 FF25 0020 .j.......%. @..%.
```

Por

```
00000426 6A00 6871 3040 0068 7C30 4000 6A00 E80D 0000 j.hq0@.h|0@.j.....
00000438 006A 00E8 0600 0000 FF25 0820 4000 FF25 0020 .j.......%. @..%.
```

Guardemos ahora los cambios y ejecutemos de nuevo el Crackme y ya no sale la Nag Screen, tercer método realizado.

Método 4
(Parchando en Memoria)

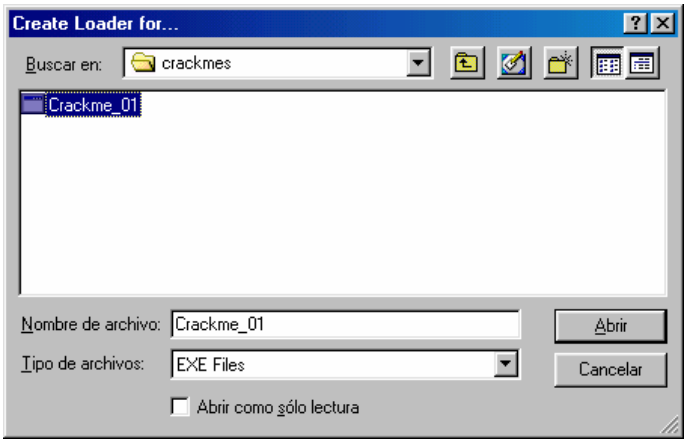
Ahora veremos el 4to método para crackear esa Nag Screen, el cual consiste en la creación de un Loader (pequeño programa que parcha en memoria las instrucciones del programa al momento de estarse ejecutando, par esto usaremos una de las utilerías llamada **Princess Sandy 1.0**, empecemos con este método.

Como recordamos en los métodos anteriores tenemos que saltar de la dirección **00401026** hasta el **ExitProcess** que esta en la dirección **0040103B** para poder eliminar esa Nag Screen

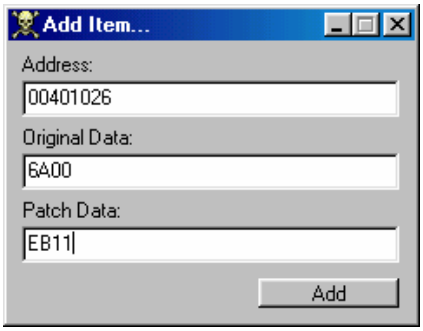
```
* Reference To: USER32.MessageBoxA, Ord:01BBh

:00401021 E81A000000                Call 00401040
:00401026 6A00                      push 00000000

* Possible StringData Ref from Data Obj ->"Remove Me!"

:00401028 6871304000                push 00403071

* Possible StringData Ref from Data Obj ->"NAG NAG"

:0040102D 687C304000                push 0040307C
:00401032 6A00                      push 00000000

* Reference To: USER32.MessageBoxA, Ord:01BBh

:00401034 E807000000                Call 00401040
:00401039 6A00                      push 00000000

* Reference To: KERNEL32.ExitProcess, Ord:0075h

:0040103B E806000000                Call 00401046

* Referenced by a CALL at Addresses:
|:0040100E   , :00401021   , :00401034

* Reference To: USER32.MessageBoxA, Ord:01BBh

:00401040 FF2508204000          Jmp dword ptr [00402008]
```

Reference To: KERNEL32.ExitProcess, Ord:0075h

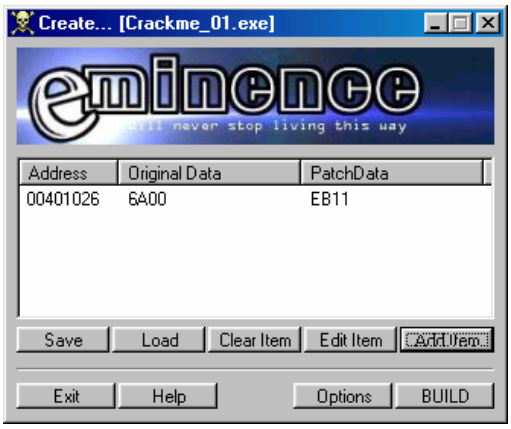
Para eso abrimos el Prices Sandy y nos parece la siguiente ventana:



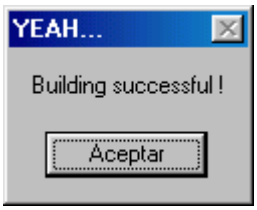
En la cual seleccionaremos el Crackme en cuestión y le damos clic en Abrir y estaremos ya en la ventana de edición del **Princes Sandy v 1.0**, de ahí damos clic en **AddItem** y aparece al ventana en la cual pondremos la dirección de memoria a parchar, la instrucción original y la instrucción como quedará parchada como se muestra en la imagen:



Damos clic en Add y nos muestra en la ventana principal lo siguiente:



Aquí podemos ver que ya esta listo para ser compilado el Loader, Damos clic en BULID y nos sale la ventana donde seleccionaremos el lugar donde se creara el Loader, por default lo crearemos en el fólder donde se encuentra el Crackme en Cuestión y nos aparece la siguiente ventana:



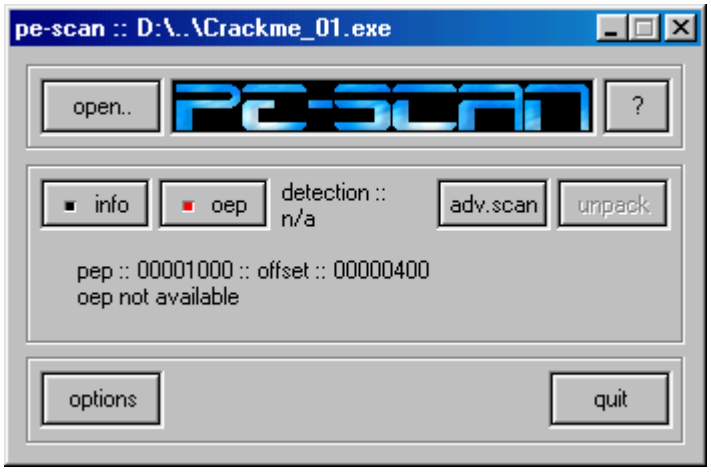
Ahora si creo que ya esta listo, vamos al fólder donde se guardo el Loader y lo ejecutamos, recuerden que para que funcione este debe de ejecutarse dentro del fólder donde esta instalado el Crackme, lo ejecutamos y funciona a la perfección, podemos decir cuarto método listo.

Método 5
(Injertando código)

Bueno ahora creo que viene el proceso mas complicado, que es el de injertar código en el Crackme para que desaparezca esa Nag Screen.

Lo primero a hacer es que el **Entry Point** salte hacia el código que nosotros escribimos y entonces parchara el programa, quitara la Nag Screen y después retornará al **Entry Point** Original.

Para eso tendremos que usar la utilería llamada **Pescan v 3.31** y checar el **Entry Point** del Crackme:



Como podemos ver el Entry Point es: **00001000**

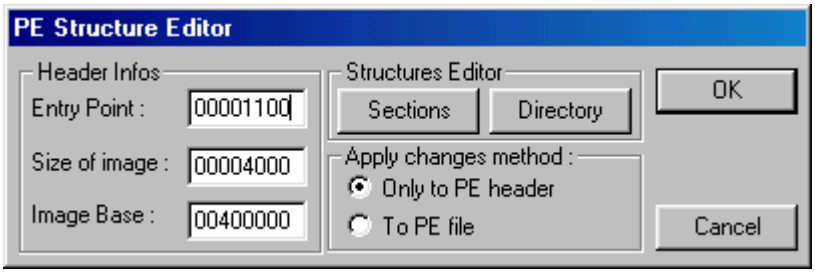
Ahora abramos en el editor Hexadecimal el Crackme y busquemos en la sección de código un poco de espacio para poder injertar nuestras instrucciones y encontramos un poco de espacio en el **Offset 500** y se preguntaran como selecciono ese espacio o como lo encuentro, bueno pues mi razonamiento me dice que el lugar donde existen:

00000000000000000000000000000000000000000
00000000000000000000000000000000000000000
00000000000000000000000000000000000000000
00000000000000000000000000000000000000000

Ahí podríamos poner las instrucciones nuevas, OK ahora vamos a usar el **ProcDump** para cambiar el **EntryPoint** del Crackme al **Offset 500.**

Para eso abrimos el **ProcDump** y damos clic en la opción llamada **Pe Editor**:

Y nos aparecerá una ventana donde seleccionaremos el Crackme, nos parece la siguiente ventana:

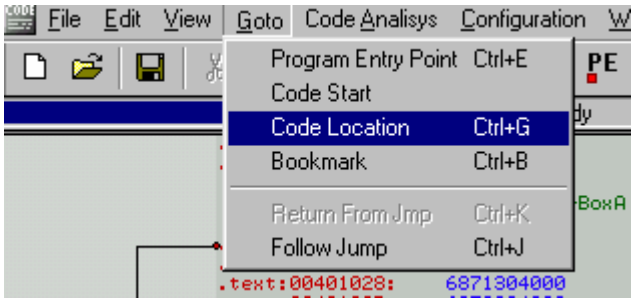


En la cual cambiaremos el **Entry Point** a **00001100**, damos clic en **OK** y nos manda el siguiente error:

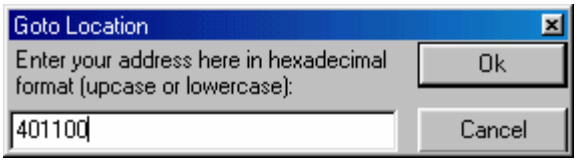
Esto quiere decir que el Crackme tiene las propiedades de solo lectura, vamos y cambiamos sus propiedades a *lectura y escritura*, regresamos a hacer el mismo proceso de cambio de **Entry Point** y ahora si no lo cambia.

Se preguntarán por que cambiamos **"00001000"** por **"00001100"** por que **00001000** es la Virtual Addres para **00000400** y nosotros queremos poner nuestro código en **00000500** por lo tanto tenemos que poner la **Virtual Addres 00001100**, ahora salvemos los cambios en el **ProcDump** y toca el turno de codificar nuestras instrucciones:

Recordemos que debemos remplazar el **"E807"** con "**E8OD"** en el dirección **"00401034"** por lo tanto yo la codificare en el offset **"00401100",** desensamblemos ahora el crackme con el **Bdasm® v 1.0** y después vamos al menú ***GOTO > Code Location***:



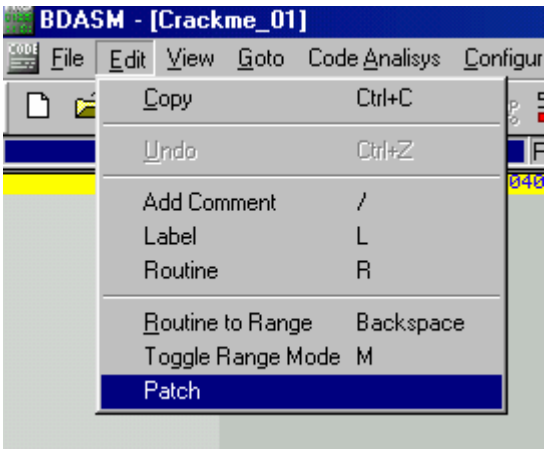
En la cual pondremos:



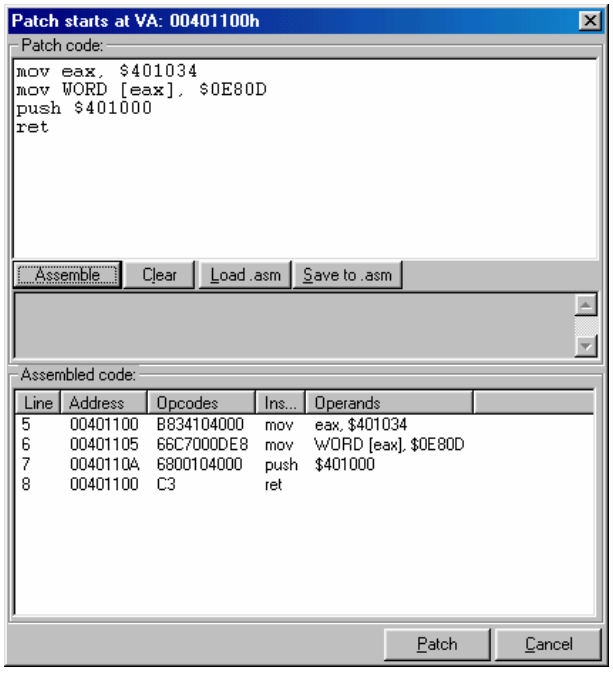
Damos clic en OK y nos mandara a la dirección donde ensamblaremos nuestras instrucciones:

mov eax, $401034 -à **mueve al acumulador Eax lo que hay en la dirección 401034**
mov WORD [eax], $0E80D à **mueve el valor E80D a EAX**
push $401000 à **metemos en la pila el valor 1000**
ret -à **retorna la actividad**

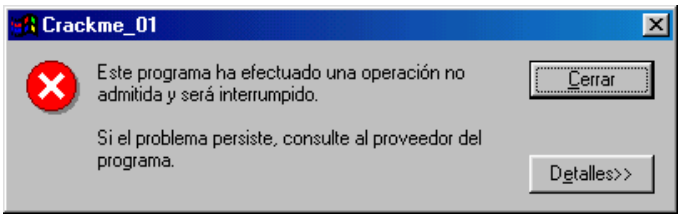
Para ensamblar vamos al menú:



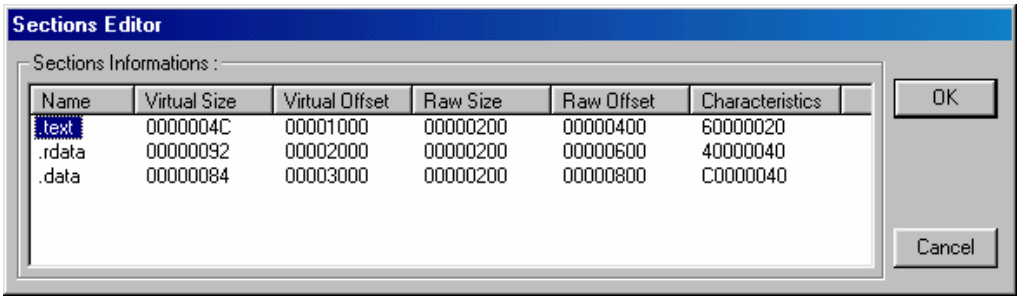
Y nos aparece la siguiente ventana en la cual escribiremos las instrucciones arriba señaladas:



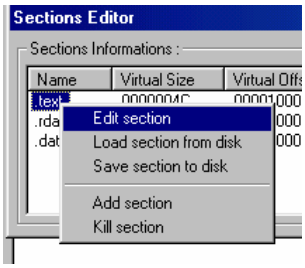
Después de escribir las líneas de código a ensamblar, damos clic en el botón **Assemble** y en la parte inferior de la ventana pondrá las instrucciones ensambladas en sus direcciones de memoria, de ahí damos clic en **Patch** y nos parchara automáticamente el Crackme, cerramos el **Bdasm** y vamos a ejecutar el Crackme y nos manda un error:



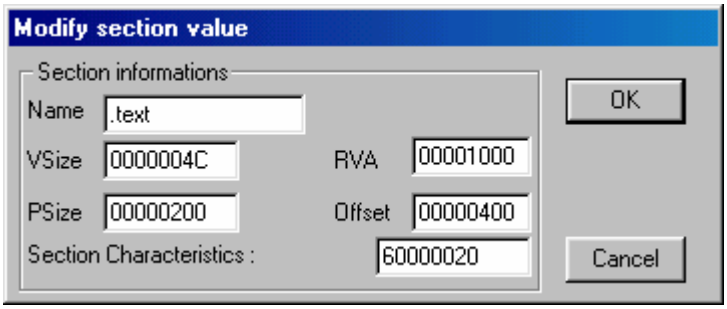
¿ Que será ese error ?, ah creo que ese error es por que el tamaño de la sección no es muy grande, por lo tanto abramos otra ves el Crackme con el **procDump**, clic en **PEditor**, después seleccionan el Crackme y aparece la ventana de **Pe Estructure Editor**, ahí damos clic en **Sections** y nos aparece la siguiente ventana:



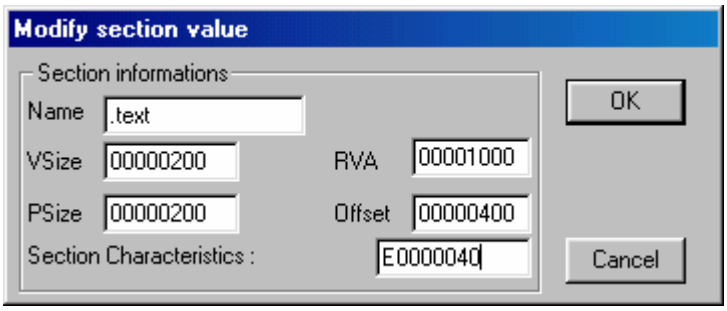
En la cual seleccionamos la sección **.TExt** y damos clic con el Botón derecho sobre esa sección y nos aparece un menú emergente en el cual seleccionaremos



Edit Section y aparece otra ventana:



En la cual modificaremos el **VSize** que es **0000004C** lo cambiamos por **00000200** y también cambiamos el **Section Characteristics** a **E0000040**, quedando así como la imagen lo muestra:



Damos clic en OK salimos del **ProcDump** y ahora si ejecutemos de nuevo el Crackme y se ejecuta muy bien y como podemos ver la Nag la eliminamos, creo que el Método 4 dio resultado.

Nota: Cabe mencionar que para el **método no. 5** la dirección de memoria a parchar puede cambiar en cada Computadora.

2da. Nota: un saludo a todos los amigos de cracklatinos y también espero que les guste este tutorial que ya tenia por aquí guardado y espero les sirva de algo.

Pr@fEs0r X 2008 " Another One Bite The Dust "
Octubre 2008