MEDIO DIFICIL

(LECCION TREINTA Y UNO)

Esta lección se refiere al programa **NBG clean registry 1.7.2** que sirve para limpiar el registro, de entradas no validas, de basura, de restos de entradas de programas desinstalados, etc.

Se baja de **www.tucows.com** y allí esta para bajar esta versión.
Cuando abrimos el programa sale una horrible pantalla roja para registrarse, allí dice los días que te quedan de prueba, y si haces clic en el icono con la llavecita, te trata de conectar a internet vaya a saber para que pero no te podes registrar por allí, (SALVO PAGANDO Y ESO NO LO RECOMIENDO), jua jua, y cuando cerrás el programa vuelve a aparecer la ventan roja horrible esa.

Bueno tratamos de abrir el ejecutable con el WDASM y nada no se desensambla, miro con el Languaje 2000 y me dice que esta comprimido con **ASPACK O ASPROTECT**, así que con el CASPR 1.012 o la ultima que haya salido, lo descomprimo fácil, según vimos en lecciones anteriores, y me entrega el ejecutable sin la protección ASPROTECT o ASPACK, lo que queda es quitar la protección Antidesensamblado con el PROCDUMP cambiando C0000040 por E0000020 y listo queda perfecto para desensamblar con el WDASM para DELPHI. (o sea el que no esta modificado para VISUAL BASIC)
Igual el programa mantiene una protección antisoftice pero con el FROGSICE pasa fácilmente.
Cuando quiero ejecutar el programa descomprimido pongo el ejecutable NBGcleanRE.exe en la carpeta de instalación del programa C:\NBGCleanRE

y lo ejecuto haciendo doble clic y OOOOPS aparece un mensaje diciendo que o un virus o alguien modifico el CRC del programa y no funciona, entre las STRINGS REFERENCES encontramos fácilmente este cartelito que dice exactamente:

INCORRECT CRC CHECKSUM AND SIZE

Si hacemos doble click encima de esa STRING REFERENCE caemos en

4e9f49 75 EF cambiamos el **75** por **EB** y listo saltea ese cartel y arranca el programa perfectamente.

```
:004E9F45 807DF300        cmp byte ptr [ebp-0D], 00
:004E9F49 EB7F            jmp 004E9FCA
```

Con eso ya saltamos ese cartelito que es tipo protección por si alguien le limpia el ASPROTECT o ASPACK.

Bueno ya arranca el programa y ya esta limpio, ahora hay que crackearlo lo cual no es fácil tampoco.

Buscaremos alguna STRING REFERENCE que diga si estas registrado o no.

Me gusta una cuando arrancas el programa y una vez que ingresas, y entras al mismo, haces clic arriba a la izquierda donde dice en azul NBG CLEAN REGISTRY y eso abre una ventana que dice
VERSIÓN 1.7.2 ... SHAREWARE UNREGISTERED...

Vuelvo al WDASM y busco esa STRING y allí esta, si hago clic encima aparece dos veces casualmente una arriba de la otra.

*** Possible StringData Ref from Code Obj ->"Version 1.7.2 Release Skin ENG "**
->"(build 2012) shareware"

```
                        |
:004FAB87 6874AD4F00          push 004FAD74
:004FAB8C FFB308050000         push dword ptr [ebx+00000508]
:004FAB92 68C4AD4F00          push 004FADC4
:004FAB97 FFB30C050000         push dword ptr [ebx+0000050C]
:004FAB9D 8B8308030000         mov eax, dword ptr [ebx+00000308]
:004FABA3 05A0000000          add eax, 000000A0
:004FABA8 BA04000000          mov edx, 00000004
:004FABAD E82E94F0FF          call 00403FE0
:004FABB2 EB15               jmp 004FABC9
:004FABB4 8B8308030000         mov eax, dword ptr [ebx+00000308]
:004FABBA 05A0000000          add eax, 000000A0
```

*** Possible StringData Ref from Code Obj ->"Version 1.7.2 Release Skin ENG "**
->"(build 2012) shareware"

Asi que veamos la de arriba de las dos.

```
:004FAB7E       cmp byte ptr [eax+00000504], 00
:004FAB85       jnz 4fabb4
```

*** Possible StringData Ref from Code Obj ->"Version 1.7.2 Release Skin ENG "**
->"(build 2012) shareware"

Bueno, parece que en el contenido de [eax+504] o sea en 1AE24A8, está el flag de si estas registrado o no, probemos con el softice antes de entrar en esta ventana, ponemos un BPX SHOWWINDOW y una vez que hacemos clic y entra en el softice , volvemos con F12 al ejecutable, y allí borro con bc* y pongo un BPX 4FAB7E, para que pare allí, cierro la ventana la vuelvo a abrir y para allí antes de que aparezca, cuando estoy encima de 4FAB7E hago:

E1AE24A8 y enter con eso puedo cambiar el valor que está comparando, ya que hay un cero, pongo un uno a ver qué pasa, luego hago enter, y hago x

Me aparece la ventana diciendo REGISTRADO, hmmmm, que bueno.

Bueno aparentemente esa posición de memoria es la que siempre testea para ver si estas registrado o no, por lo tanto, podemos poner un
BPR 1AE24A8 1AE24A8 rw
Para que pare allí cada vez que trate de leer el valor de ese flag.
Si quiero poner ese BPR antes de que arranque el programa, no para en ningún lado, alguna protección hay, entonces lo que hacemos es poner
BPX GETVERSION y cuando para allí volvemos al ejecutable (DESPUES DE CÓMO 40 f12) y allí recién pongo el bpr.

Para en estos cuatro lugares:

4EAECA
4EB142
4FF39E
4FAB7E
Donde hay siempre comparaciones de eax+504 con cero o con uno

cmp byte ptr [eax+00000504], 00

también para en otros lugares donde mueve valores allí, pero esos no me interesan solo cuando toma decisiones según el valor que hay allí.

Y espero que ninguno otro más, igual tiene una cierta trampita en algunas de estas comparaciones el valor que hay en 1ae24a8 es un uno y espera un cero y otros es cero y espera un uno, con el softice determinamos qué valor tiene en cada salto, para saber cómo parchear cada uno.

En resumidas cuentas tiene que quedar asi;

La comparación en

```
:004EAECA C6800405000001      mov byte ptr [eax+00000504], 01
:004EAED1 EB04                jmp 004EAED7 (O nop nop )
:004EAED3 90                  nop
:004EAED4 90                   nop
:004EAED5 90                  nop
:004EAED6 90                   nop
:004EAED7 8B45FC              mov eax, dword ptr [ebp-04]
```

La reemplazo por mover a **[EAX+504]** el valor 01 (ya que habia cero)
y despues no salto, ya que vi en el softice que cuando llego alli salta y tengo que hacer lo contrario por eso salto a **4EAED7** que es lo mismo que NOPEAR.

Luego queda la comparación en 4eb142 de esta forma:

```
:004EB142 C6800405000001        mov byte ptr [eax+00000504], 01
:004EB149 E9BC110000            jmp 004EC30A
```

Lo mismo veo que en el contenido de EAX+504 hay un cero por eso hago que la sentencia mueva allí un uno, y como después no salta lo hago saltar (que contrera, jua).

Luego viene 4ff39E queda así

```
:004FF39E C6800405000001        mov byte ptr [eax+00000504], 01
:004FF3A5 90                    nop
:004FF3A6 90                    nop
```

Y la ultima 4FAB7E:

```
:004FAB7E C6800405000001        mov byte ptr [eax+00000504], 01
:004FAB85 90                    nop
:004FAB86 90                    nop
```

Bueno una vez que hacemos todos estos cambios probamos a ver que pasa y la pantalla roja horrible ya no aparece (cuando cerramos el programa tampoco) y el programa arranca directamente y dice que estamos registrados, igual este tipo de programas puede tener por ahí otra Comparación media escondida por lo que habría que probarlo en profundidad pero parece funcionar bien.

Ya veremos si aparece alguna sorpresa lo seguiremos en la lección 32, pero por ahora.

PROGRAMA REGISTRADO

Ricardo Narvaja

SOFTCOPIER 3.4.2.4

(LECCION TREINTA Y DOS)

Este programita es una linda utilidad para sacar fotocopias rápido, scannea y manda directo a la impresora, para fotocopiar, rápido y fácil uno o varios papeles.

Se baja de **www.buzzsoft.com**

Bueno este programa va contando la cantidad de copias que uno hace y cuando llega a 25 se vence, teniendo que actualizarse por Internet para poder copiar 25 mas, ufff.
Bueno para mi gusto, a pesar de que los cracks que hay por allí disponibles, trabajan sobre ese registro, lo que yo use es tratar de que no llegue nunca la cuenta a 25 copias para que nunca venza.

Lo primero es ver con WDASM el ejecutable SC3 que esta en
C:\Archivos de programa\BuzzSoft\Sc

, se descomprime fácilmente y entre las STRINGS REFERENCES lo primero que llama la atención es:

"You have just completed a demonstration of SoftCopier. SoftCopier is a powerful yet"

Ojo que hay que usar el WDASM original o sea sin la modificación para VISUAL BASIC sino esta STRING no aparecerá ya que esta hecho en DELPHI.

Bueno dice que se acabo la demostración y que se va a modificar el programa para que si no lo actualizas no lo puedas usar mas y bla bla bla.

Bueno tenemos que tratar de evitar que por cualquier motivo el programa llegue aquí a esta ZONA de CHICO MALO, yo diría muuuy malo, jua jua.

Si vamos un poco hacia arriba encontramos en 459d25 este salto que esquiva todo este cartelon y todo este sector de chico malo, justo evita lo que no queremos, que por cualquier cosa me venza.

```
:00459D1D A17C1D4600          mov eax, dword ptr [00461D7C]
:00459D22 803800              cmp byte ptr [eax], 00
:00459D25 0F8574060000        jne 0045A39F
```

Obviamente tengo que reemplazar ese salto condicional por un JMP para que nunca entre a esa zona que me puede modificar algo y que no funcione mas, esto hay que hacerlo aunque logremos que el programa no aumente de alguna forma la cuenta de las copias que hizo.

Esto después de modificado quedaría así:

```
:00459D1D A17C1D4600        mov eax, dword ptr [00461D7C]
:00459D22 803800            cmp byte ptr [eax], 00
:00459D25 E975060000        jmp 0045A39F
:00459D2A 90                nop
```

El NOP al final es porque el salto condicional ocupaba 6 valores hexa y el JMP ocupa 5 solamente, para que quede bien y reemplace los seis.

Bueno ahora que ya modificamos esto y estamos un poco mas tranquilos tenemos que encontrar donde guarda este programejo las copias que va haciendo, lo primero que pensé es usar el ART para ver si hace cambios en el REGISTRO al hacer una copia, pero ya que no estoy seguro de que sea en el registro, voy a usar otra herramienta poderosa llamada **TECHFACTS** que entre muchísimas utilidades que tiene permite hacer lo mismo pero en todo el sistema, y cualquier cambio que haga te lo saca en una lista.
Muy bien como nosotros queremos estudiar los cambios que produce el SOFTCOPIER conviene cerrar cualquier otra aplicación que este corriendo ya que puede realizar algún cambio y confundirnos.
Esto lo podemos hacer con CTRL+ALT+DEL y cerrar todo uno por uno hasta que quede solo el explorer y ahí recién arrancar el TECHFACTS.

Como verán dentro de este programa hay muchísimas utilidades pero lo que nosotros vamos a utilizar esta en la pestaña TOOLS y haciendo click en WATCH SYSTEM, poniendo la tilde en RUN THIS PROGRAM no tocamos mas nada y abrimos el buscador para encontrar el ejecutable de SOFTCOPIER que esta en
C:\Archivos de programa\BuzzSoft\Sc
y se llama SC3 hay otro ejecutable en otro directorio que es una barra de herramientas que despues abre este ejecutable así que nos vamos a concentrar directo en este ya que es el que hace la copias, lo abrimos y abajo hacemos click en GO para que empiece a trabajar y el TECHFACTS hará primero una instantánea del sistema completo, correrá el programa, donde nosotros haremos una copia para que aumente el valor de copias en uno mas, y cerraremos el programa , luego de lo cual el TECHFACTS hará otra instantánea del sistema y me dirá donde realizo cambios el SC3.
El reporte a mi me dice lo siguiente:

TechFacts 98 System Watch Report
04/23/01 08:44:33 pm

The following directories and files were added:(0)

The following files were modified:(4)
c:\WINDOWS\SYSTEM.INI
c:\WINDOWS\TWAIN.LOG
c:\WINDOWS\Twain001.Mtx
c:\WINDOWS\WIN.INI

Changes made to: C:\WINDOWS\WIN.INI...
Keys changed in: C:\WINDOWS\WIN.INI: (1)
[SoftCopier3]Number of Copies=3 to 4

Changes made to: C:\WINDOWS\SYSTEM.INI...
Keys changed in: C:\WINDOWS\SYSTEM.INI: (1)
[NonMSApps]TWAIN_SC3REF=74 to 93

Registry key values changed: (2)
HKEY_USERS\.DEFAULT\Software\Borg
Value "Winmax": from "18973185" to "18813185"
HKEY_USERS\.DEFAULT\TWAIN\BUZZTW32
Value "REFNO": from "845" to "914"

Marque con negrita lo importante, el programa al hacer una copia realizo todos estos cambios.

Vemos que dice que modifico el WIN.ini y el SYSTEM.ini y en el WIN.ini dice el número de copias esto se pone interesante, hay un par de claves de registro que también cambiaron, como se ve.

Bueno yo me tome el trabajo de editar el WINI.ini a mano para ver si cambiando el numero de copias que ahora esta en cuatro, lo ponía a mano en cero, el programa cuando arrancaba me decía que había hecho cero copias, pero no, ese valor no influye, es solamente informativo.
También los valores del registro los probé volver al valor anterior a mano:

HKEY_USERS\.DEFAULT\Software\Borg
Value "Winmax": from "18973185" to "18813185"

fui al regedit y cambié el valor que había en esa clave 18813185 por el que había antes 18973185 y volví a arrancar el programa, y seguía con las cuatro copias, lo mismo ocurrió con la otra clave del registro:

HKEY_USERS\.DEFAULT\TWAIN\BUZZTW32
Value "REFNO": from "845" to "914"

Volví a escribir 845 que era lo que decía allí cuando había hecho solo tres copias y arranque el programa de nuevo, y otra vez cuatro copias.
Lo único que me quedaba era el system.ini

Changes made to: C:\WINDOWS\SYSTEM.INI...
Keys changed in: C:\WINDOWS\SYSTEM.INI: (1)
[NonMSApps]TWAIN_SC3REF=74 to 93

Así que abrí el sysedit y busque el system.ini y cambié el valor de 93 por el de 74, como había antes y arranque el programa de nuevo BIIINGOOO, ahora dice tres copias, me fije que de 74 a 93 hay 19 y trate de ir mas atrás reste a 74 -19=55 lo puse en el SYSTEM.ini y arranque de nuevo el programa y DOS COPIAS, jua jua.
Bueno si a 55 le vuelvo a restar 19 dos veces más puedo llegar hasta cero de nuevo, pero lo dejo así en 55 ya que es un numero fácil de encontrar, sabiendo que en hexadecimal es 37 no? O sea que buscare por allí el 37 con el SOFTICE.

Voy al WDASM y me doy cuenta que esta la STRING REFERENCE "SYSTEM.INI" y cuando hago click encima me lleva a seis posiciones distintas a saber:

45cded-45d673-454261-4542ee-454466-45bad8

Es seguro que cuando utilice esta STRING el programa estará leyendo o grabando el valor del SYSTEM.ini, así que en el softice arranco el programa con el SYMBOL LOADER y cuando arranca pongo BPXs en todos estos puntos.

También puedo poner un BPX GETVERSION y una vez que rompe allí por segunda o tercera vez volver al ejecutable y poner los BPXs es lo mismo.

Para dos veces cuando arranca y una vez después que sacamos una copia y una vez cuando se cierra el programa.

Paro aquí:

```
* Possible StringData Ref from Code Obj ->"System.Ini"
                       |
:0045CDED B9D0CF4500          mov ecx, 0045CFD0
:0045CDF2 B201                mov dl, 01
:0045CDF4 A124EF4300          mov eax, dword ptr [0043EF24]
:0045CDF9 E88221FEFF          call 0043EF80
:0045CDFE A3042B4600          mov dword ptr [00462B04], eax
:0045CE03 6A13                push 00000013

* Possible StringData Ref from Code Obj ->"TWAIN_SC3REF"
                       |
:0045CE05 B9E4CF4500          mov ecx, 0045CFE4

* Possible StringData Ref from Code Obj ->"NonMSApps"
                       |
:0045CE0A BAFCCF4500          mov edx, 0045CFFC
:0045CE0F A1042B4600          mov eax, dword ptr [00462B04]
:0045CE14 E89F22FEFF          call 0043F0B8
:0045CE19 8BF0                mov esi, eax
:0045CE1B 8BC6                mov eax, esi
```

Como ven desde donde está la STRING azul de SYSTEM.ini hasta donde esta lo que hay que parchear (en rojo) no hay mucho, unas cuantas sentencias no más.
Allí al salir del CALL que está en rojo en EAX me aparece el valor 37 que va a pasar a ESI, entonces lo modifico para que no sea así, lo cambio por esto:

:0045CE19 2BF6 sub esi, esi
:0045CE1B 8BC6 mov eax, esi

O sea que SUB es retar ESI-ESI o sea que va a quedar siempre ESI=0, luego ese valor se pasa de nuevo a EAX, que queda valiendo cero, hago los cambios a ver que pasa y cuando arranca el programa me dice CERO COPIAS, jua jua, va mejorando.

El próximo lugar que vi es cuando se cierra el programa allí paró una vez que hice una copia y me aumento el valor de copias a uno, seguro para guardar ese valor en el SYSTEM.ini.

Aquí vemos de nuevo la STRING system.ini

*** Possible StringData Ref from Code Obj ->"System.Ini"**
|
:0045BAD8 B97CBB4500 mov ecx, 0045BB7C
:0045BADD B201 mov dl, 01
:0045BADF A124EF4300 mov eax, dword ptr [0043EF24]
:0045BAE4 E89734FEFF call 0043EF80
:0045BAE9 8906 mov dword ptr [esi], eax
:0045BAEB 6BC313 imul eax, ebx, 00000013
:0045BAEE 83C011 add eax, 00000011
:0045BAF1 8BD8 mov ebx, eax
:0045BAF3 53 push ebx

y tiene que quedar asi

0045BAEB 6BC313 imul eax, ebx, 00000013
0045BAEE 2BC0 sub eax, eax
0045BAF0 2BDB sub ebx, ebx
0045BAF2 90 nop
0045BAF3 53 push ebx

O sea saliendo el valor de eax y ebx siendo cero, esto también se ve claro en el SOFTICE que aquí tiene los valores que va a guardar en el SYSTEM.ini.

La última modificación es para que no aumente la cuenta mientras uno va sacando copias y no cierra el programa ya que a pesar de que siempre empezamos de cero, como sigue aumentando los valores hasta que uno cierra el programa y graba un cero, en el

SYSTEM.ini, puede ocurrir que uno se pase de la 25 copias antes de cerrar el programa y KAPUTTT.

Eso esta unas líneas más arriba justo donde hay una STRING que dice **NUMERO DE COPIAS**

```
:0045BAAD 8B1D802B4600        mov ebx, dword ptr [00462B80]
:0045BAB3 031D842B4600        add ebx, dword ptr [00462B84]
:0045BAB9 53                  push ebx
```

* Possible StringData Ref from Code Obj **->"Number of Copies"**

Allí en 462b80 guarda el número Viejo de copias que uno tenia y en 462b84 el numero Nuevo de copias que hizo, para actualizarlo, los tiene que sumar y es lo que hace en la sentencia

45bab3 que suma a EBX que tiene el valor viejo de copias el numero de copias que hice para actualizarlo, por lo tanto si anulamos esta suma en EBX me quedara el valor viejo de copias que tenia y no se actualizara aunque hayamos hecho varias copias, siempre quedara igual, por lo tanto a

0045BAB3 031D842B4600 add ebx, dword ptr [00462B84]

la reemplazamos por 6 NOPS

quedaria asi

```
0045BAAD 8B1D802B4600        mov ebx, dword ptr [00462B80]
:0045BAB3 90                                          nop
:0045BAB4 90                                          nop
:0045BAB5 90                                          nop
:0045BAB6 90                                          nop
:0045BAB7 90                                          nop
:0045BAB8 90                                          nop
:0045BAB9 53                  push ebx
```

Modifico esto arranco el programa me dice cero copias, sigo haciendo copias y me dice cero copias y cierro el programa y en el system.ini, dice cero copias.

Ustedes dirán porque si logramos que no variara el número de copias pusimos el JMP en el vencimiento del programa ya que no es necesario:

Yo no estoy seguro pero ese valor en el registro que sigue aumentando
HKEY_USERS\.DEFAULT\TWAIN\BUZZTW32
Value "REFNO": from "845" to "914"

cada vez que hacemos una copia sigue aumentando a pesar de todo, por si acaso si hay un límite máximo para este número que el programa vaya al vencimiento por aquí, es conveniente poner el JMP, por si acaso, quizás después de dos meses el programa compruebe este valor y me tire a vencido, al estar el JMP, me salvo
También puedo buscar en el programa cuando lee de aquí y buscar este valor de 914 , para modificar el programa para que no pase nada, pero creo que con lo que se hizo es suficiente.

PROGRAMA CRACKEADO

Ricardo Narvaja

ALGO FÁCIL PARA UNA SEMANA DIFÍCIL (PARA MI)

(LECCION TREINTA Y TRES)

El programa víctima se llama **SMPServer** la versión que yo tengo es la Standard R4 build 20010409) eso dice en el programa si quieren saber para qué sirve y saben inglés aquí esta lo que dice la página del mismo:

http://www.eastbow.com/eng/smp_download.htm

What can you do with SMP ?

You can get the following data from your PCs: (**en red**)
IP address, Computer name, Login name, Department, Etc, CPU, RAM, OS, Disk CD-ROM, Media card, Modem, Video card, Monitor, Printer, Mouse, Keyboard, Network card, SCSI, USB, Current process, Screen settings and installed software list.
With these data, you can make the report
including, *H/WS/W list/statistics*
in Microsoft Excel (tab seperated) and HTML format.
In addition, SMP can catch the changes including,
hardwares,
IP movement.

Bueno la limitación es que es para 5 licencias o PCs solamente y trataremos de que no tenga esta limitación.
Desensamblamos el ejecutable con el WDASM y no hay problema lo hace perfectamente, no esta comprimido.
Allí hay una STRING REFERENCE sospechosa

"You need more licenses to add more PC information."

O sea dice que necesitamos mas licencias para agregar mas información, aquí parece estar la limitación.

```
:0041998B E8E079FEFF        call 00401370
:00419990 85C0              test eax, eax
:00419992 7D19              jge 004199AD
:00419994 6A01              push 00000001
```

*** Possible StringData Ref from Data Obj ->"You need more licenses to add "**
->"more PC information."

Aquí parecería que con parchear este salto, ya estaría pero vamos a ver un poco ese CALL 401370 que hay antes del salto de donde sale EAX tomando un valor que decide si salta o no.

*** Referenced by a CALL at Addresses:**
|:0040DFB5 , :0041998B , :004199E2

:00401370 A1808F4500 mov eax, dword ptr [00458F80]

Bueno vemos que entra a este CALL llamado de varios lugares distintos del programa, o sea que en varios puntos existirá esta limitación pero si parcheamos el CALL para que cuando salga valga EAX=0 eso hace saltar siempre los saltos condicionales que hay a la salida de este CALL en los distintos puntos del programa para que evite la limitación.

Uno de los puntos desde donde llega a ese CALL es

:0040DFB5 E8B633FFFF **call 00401370**
:0040DFBA 8BD8 mov ebx, eax
:0040DFBC 85DB test ebx, ebx
:0040DFBE 0F8D24080000 jnl 0040E7E8

.
.
.
.

:0040DFE9 50 push eax

*** Possible StringData Ref from Data Obj ->"A PC is tried to connect in the "**
->"condition of license full."
|
Aquí hay otro lugar donde hay una limitación vemos que sale del CALL 401370 como en la limitación anterior, y aquí también el valor de EAX, se pasa a EBX y se testea EBX y si es cero salta por encima del cartel molesto que dice que una PC esta tratando de conectarse en la condición de licencia llena, y no te deja.
O sea que aquí también si sale EAX siendo cero vamos a sortear la limitación.

Por lo tanto dentro del CALL 401370 reemplacemos

|:004013A1 83C8FF **or eax, FFFFFFFF**
:004013A4 5E pop esi
:004013A5 C3 ret

Or EAX , FFFFFFFF por XOR EAX , EAX que es una función que hace EAX igual a cero siempre.

Y también aquí

```
:004013A6 8BC6          mov eax, esi
:004013A8 5F            pop edi
:004013A9 5E            pop esi
:004013AA C3            ret
```

Reemplazamos MOV eax,esi por Xor EAX,EAX porque a veces dentro del CALL sale a veces por el primer RET y a veces por este segundo RET ya que hay un salto condicional en

```
:00401392 7412          je 004013A6
```

que hace que salga a veces por un RET y a veces por otro, ahora salga por donde salga siempre va a ser EAX=0 y va a seguir funcionando sin caer en los cartelitos que dicen que no se pueden usar más de cinco licencias.

O sea en 4013a1 **83 c8 ff** lo reemplazo por **33 c0 90** que es **XOR eax,eax** y un **NOP** ya que son tres cifras.
Y en 4013a6 reemplazo **8b c6** por **33 c0** aquí sin NOP ya que son solo dos cifras.

PROGRAMA CRAQUEADO

Ricardo Narvaja

EMPEZANDO A USAR EL REVIRGIN
(Trago amargo y con paciencia)

(LECCION TREINTA Y CUATRO)

El Revirgin es una herramienta que permite reconstruir los ficheros que pudimos dumpear (descomprimir) ya sea con el PROCDUMP o ICEDUMP al disco rígido y no funcionan, generalmente la mayoría de los compresores nuevos destruyen la tabla de funciones importadas, para que no puedan arrancarse y apenas arranquen den errores varios de WINDOWS.
Por lo tanto es un muy buen programa sobre todo en casos como compresiones con ASPROTECT o algunos otros, que no permiten ser parcheados en memoria con RISC PROCESS PATCHER o similares.
También les comento que en el uso del REVIRGIN yo también estoy haciendo mis primeros pininos y aprendiendo a la par de ustedes, así que estamos aprendiendo juntos, en este caso utilice como base un tutorial existente sobre REVIRGIN aunque esta en ingles y además es mas complicado que el que voy a hacer yo pues como es de hace unos meses, y las nuevas versiones de REVIRGIN han avanzado bastante, muchas cosas que eran necesario hacer en esa época a mano, ahora las hace el programa automáticamente, aunque desde ya aclaremos que el programa nos facilita las cosas, pero no hace todo, tendremos aquí que meter mano y buscar a mano ciertos datos necesarios.
Aquí vamos a utilizar otro editor de PE que no es el PROCDUMP, se llama PE Editor y va por la versión 1.7 y lo baje de

http://212.14.34.87/~fkiepski/down/uzytki/peditor.zip

aunque esta en las principales paginas de herramientas de cracks, lo mismo que el ultimo Revirgin esta en SUDDENDISCHARGE, el link directo es

http://www.suddendischarge.com/files/Unprotection%20Tools/revirgin110b9b.zip

Hasta ahora es el ultimo aunque salen versiones con mejoras bastante seguido y es bueno siempre tener la ultima.

La otra herramienta que voy a usar es el ICEDUMP

http://www.suddendischarge.com/files/File%20Dumping/id6023.zip

Ustedes se preguntaran porque no el PROCDUMP, yo también pero el mismo programador del REVIRGIN dice que para mejores resultados usemos el ICEDUMP para dumpear el programa de la memoria al rígido. (el que quiera hacerlo con PROCDUMP creo que funcionara, lo que parece es que es mas incompatible el ejecutable resultante como para llevarlo a otras maquinas)

Bueno lo último que nos falta bajar es la victima (APLAUSSSOSSS) que esta comprimida con ASPROTECT y se llama Cool Mouse 3.4, aclaremos que aquí solo veremos la descompresión del programa y que quede funcionando descomprimido, no haremos el

crack del ejecutable, aunque una vez descomprimido y funcionando ya queda servido en bandeja para hacerlo.
La pagina del programa es
http://www.mycoolmouse.com/

aunque yo lo baje de
http://www.shelltoys.com/files/cmset.exe

Bueno una vez que nos hicimos de todo esto comencemos, instalamos el COOLO MOUSE y luego lo primero es instalar el ICEDUMP, el mismo trae distintos archivos en la carpeta W9x para las distintas versiones de SOFTICE que tengamos, allí vemos desde 3.22 hasta 4.21.53, por lo pronto yo tengo la versión del SOFTICE 4.05 así que pruebo copiando icedump.exe dentro de la carpeta del SOFTICE y lo ejecuto si está todo bien y es el que corresponde debería decir

icedump v6.0.2.3 for winice v4.05 build 334 loader
icedump was not loaded yet
icedump loaded

Lógicamente si da error hay que probar con otro hasta que encontremos el correcto, una vez que lo copiamos a la carpeta del SOFTICE en mi caso
C:\Archivos de programa\NuMega\SoftIce95

Hacemos un acceso directo al archivo icedump.exe para arrancarlo desde el escritorio o algún lugar cómodo ya que cada vez que utilicemos el ICEDUMP en distintas oportunidades lo deberemos ejecutar para que funcione.

Una vez ejecutado y que nos dice ICEDUMP LOADED, podemos comenzar con el DUMPEO.

Podemos ya usar también el PEDITOR para obtener los valores 400000 y 51000 que son La IMAGE BASE y el largo del ejecutable por lo tanto terminara en 400000 +51000 = 451000.
Abriendo el PEEDITOR y yendo a BROWSE y allí buscando en

C:\Archivos de programa\Shelltoys\Cool Mouse

Cargamos el ejecutable Cmouse.exe y allí vemos IMAGE BASE 400000 y SIZE OF IMAGE 51000 o sea donde comienza y el tamaño que tendrá en memoria.

Podemos usar el método del BPR que hemos visto en lecciones anteriores para encontrar el entry point, pero aquí solo ponemos un BPX GETVERSION y la tercera vez que pare volvemos con F12 y estaremos justo un poquito abajo del ENTRY POINT para en 408797 y el entry point está un poquito más arriba en 40876b donde vemos la típica sentencia de comienzo de programa PUSH EBP y luego MOV EBP, ESP

Esta forma de encontrar el ENTRY POINT no siempre funciona pero a veces si, y hace mas rápido el dumpeo que usar el BPR, se basa en que casi todos los programas lo primero que hacen es un BPX GETVERSION entonces si seguimos los BPX GETVERSION y vemos cual cae dentro del ejecutable en la primera sección o en una de las primeras sabremos que mirando un poquito mas arriba encontraremos el PUSH EBP que es el principio del programa o sea el ENTRY POINT.
Bueno volvemos a ejecutar el programa pero ahora después del primer o segundo GETVERSION volvemos al ejecutable y tecleamos

BPM 40876b x

para que pare en el ENTRY POINT es como poner un BPX pero mas disimulado para que ASPROTECT no lo detecte, y borramos el BPX GETVERSION y hacemos X para que siga y pare en 40876b
allí estamos en el ENTRY POINT ahora hay que dumpearlo.
Tecleamos A ENTER y allí hacemos un loop infinito con **JMP EIP**.
(por si acaso copiemos la cadena que había antes de modificarla ya que después tendremos que volver a los valores originales con el ULTRA EDIT)
O sea había **558bEC6AFF68B8D24000** Y REEMPLAZAMOS LOS DOS PRIMEROS POR **EBFE** O SEA QUE QUEDO AHORA

EBFEEC6AFF68B8D24000

La anotamos pues después la utilizaremos y hacemos el DUMP con la sentencia que tecleamos en el SOFTICE directamente

/DUMP 400000 51000 c:/Temp./dump.exe

Si al teclearla les dice Syntax Error es porque no cargaron el ICEDUMP antes.

Con eso tenemos el archivo dumpeado en C:/TEMP sugiero dejarlo allí y trabajar sobre una copia del mismo ya que un error nos obligara a hacer todo el DUMPEADO DE NUEVO, así que copiemos al escritorio el DUMP.EXE y dejemos guardado el DUMP.exe también en c:/TEMP por si acaso.

Lo primero que hay que hacer es arreglar un poco este coso que acabamos de copiar al escritorio, el DUMP.exe, que ni siquiera tiene un icono ya que así como esta es una porquería que ni forma tiene de tan mal que esta.
Bueno abrimos el PE EDITOR y cargamos este DUMP.exe que tenemos en el escritorio, vamos a SECTIONS y como podemos apreciar el tamaño VIRTUAL y el tamaño físico son distintos esto hay que arreglarlo ya que como lo sacamos de la memoria tiene que ser el tamaño en el DISCO igual al tamaño que ocupaba en la memoria, por suerte tenemos un comando que lo hace automáticamente, hacemos click derecho en cualquier sección y ponemos DUMPFIXER y automáticamente se arreglaran todas las secciones y como dice allí RS=VS y RO=VO o sea RAW SIZE IGUAL A VIRTUAL SIZE y RAW OFFSET

IGUAL A VIRTUAL OFFSET que en cristiano quiere decir que hace igual el tamaño de las secciones en el disco como las teníamos en la memoria y punto.

Una vez que hicimos esto vemos que si copiamos el archivo ya casi reparado a otro lugar aparece el icono del COOL MOUSE quiere decir que esta mejorando.
Lo otro que ya que estamos podemos hacer es cambiar las características de C0000040 por E0000020 para que se pueda desensamblar por si necesitamos verlo con el WDASM, hacemos click derecho en la primera sección que queremos editar y elegimos EDIT y allí en la ventana de la derecha en CARACTERÍSTICAS cambiamos C0000040 por E0000020 para quitar la protección Antidesensamblado.
Lo último seria cambiar el ENTRY POINT por el que conseguimos o sea donde dice ENTRY POINT cambio 38001 por 876b que es el offset del nuevo Entry Point,

AHORA REVIRGIN

Como ya dijimos usaremos el revirgin, si hay algún error sepan disculpar pues yo también estoy aprendiendo esto.
Bueno arrancamos el programa comprimido para que este en la memoria y lo dejamos una vez que arranco allí abierto, luego arrancamos el REVIRGIN. (no olvidar copiar lo dos dll que trae en en C:/WINDOWS/SYSTEMS)
En la lista de procesos que están ejecutándose que esta arriba a la izquierda, buscamos el del COOL MOUSE y allí aparece CMOUSE.exe, si no esta pueden hacer click derecho y poner REFRESH para que aparezca, si el programa esta funcionando debe aparecer, luego seleccionamos CMOUSE.exe de esa lista y le hacemos click arriba y nos aparece un cartelito que dice IMAGE IMPORT CORRUPTED try RESOLVER, o sea que deberíamos ir a IAT RESOLVER pero esto no servirá de nada pues falta lo mas importante, llenar los dos datos que hay abajo **IAT START RVA** y **IAT LENGHT** esos son los que necesitamos, hay otros dos IT START y IT LENGHT pero eso los llena solo el REVIRGIN el tema es ahora aprender a encontrar ese IAT STAR RVA y IAT LENGHT para comenzar a trabajar.

Paremos un poco aquí y expliquemos que es eso.

Si uno toma un ejecutable normal y que funciona, tomaremos como ejemplo el archivo NOTEPAD.exe que viene dentro de la carpeta del REVIRGIN (y que no esta comprimido) si lo arrancamos y entramos en el SOFTICE y ponemos BPX DIALOGBOXPARAMA y vamos a la AYUDA que es un SIGNO DE INTERROGACIÓN (?) y allí elegimos A PROPÓSITO DEL BLOC DE NOTAS (esta en francés) cliqueamos allí y entra en el softice, hicimos esto para entrar fácil, se puede usar cualquier otro BPX que nos deje en el ejecutable principal, ahora volvemos con F12 aceptamos en el cartel y F12 de nuevo y ya estamos en el ejecutable ya que en la línea verde dice NOTEPAD, bueno ahora sin preguntar mucho ya que lo voy a explicar después tecleemos.

S 400000 l ffffffff ff,25

que es la secuencia para buscar cadenas de bits en el SOFTICE así que lo que hago es buscar la cadena FF 25 dentro del ejecutable por eso empiezo en 400000 ya que esa es la dirección de inicio del mismo.

Allí en la primera vez que busque me aparecen una serie de números de la siguiente forma en 404e72 que es la dirección donde halla FF 25.

FF 25 14 65 40
FF 25 **10 65 40** **(la direccion de memoria esta al reves es 406510)**
FF 25 **0C 65 40**

Esta es una tabla de saltos que podemos ver si hacemos u 404e72 allí podemos ver en el SOFTICE una tabla de saltos uno abajo del otro que en el SOFTICE figura con el nombre de la función a la que va a saltar.

En realidad no aparece pero estos saltos son

JMP [406514] **ya que FF25 es la funcion JMP [xxxxxx]**
JMP [406510]
JMP [40650c]

y asi todos, como el contenido de por ejemplo 406514 es la dirección de una función directamente si hacemos d 406514 vemos que las cuatro primeras cifras que es el lugar donde salta son (están al revez**) 7fe1589a** que es la dirección de la función **ComDlgExtendedError** por eso el SOFTICE no nos muestra el salto como

JMP [406514] sino como JMP [CommDlgExtendedError] porque ya sabe que va a saltar allí.

Bueno todo muy lindo pero como encontramos la IAT y IAT lenght aquí y como podemos encontrarlo en el CMOUSE.

Bueno ya vimos que FF 25 corresponde a la sentencia JMP [xxxxxx] o sea saltar al contenido de y que a donde van a saltar definitivamente todos esos saltos esta determinado por el contenido de (en el caso anterior), valores guardados en 406514 , 406510, 40450c, etc, alli guarda a donde va a saltar, pues bien ese lugar donde se guarda a donde va a saltar es la famosa IAT .
Y como obtenemos donde empieza y el largo que tiene?
Bueno, en este caso son pocos saltos pero conviene mirar la lista de JMP [xxxxxx] y anotar el valor máximo y minimo de todos esos para que cuando elijamos no nos quede ninguno afuera.

O sea que si encuentro con el SEARCH todos los FF 25 que hay uno abajo del otro, tendré que fijarme en los numeritos que hay al lado para ver cual es el mayor y el menor para tener una idea de que todos están dentro.

En el caso del NOTEPAD

FF 25 **10 65 40** JMP [406510]

FF 25 **0C 65 40** JMP [40650c]

y así miro, la verdad es un poco incomodo tener que abrir y mirar en el SOFTICE esto pero les quería dar una idea de que hay un salto JMP al contenido de una dirección de memoria y por lo tanto esa dirección de memoria es la que determina a donde va a saltar.
Si abrimos el NOTEPAD.exe con el ULTRAEDIT y en Buscar ponemos FF25 nos aparece

alli los mismos FF 25 que vimos con el SOFTICE seguidos de los números que nos interesan

FF 25 **10 65 40**
FF 25 **0C 65 40**

O sea que desde el ULTRAEDIT en un programa descomprimido se pueden ver los saltos y en que posición de memoria están guardados a donde va a saltar, en este caso 406510, 40650c

Bueno busquemos esa posición de memoria aquí en el ULTRAEDIT 406510 seria 6510 en offset, busquémosla en la columna de la izquierda donde están los offsets, hasta que lleguemos allí. Ahora la cuestión es

1) Saber distinguir el Comienzo de la IAT y verificar que todos los JMP [xxxxxx] o sea las posiciones xxxxxx de memoria están dentro de la zona que vamos a elegir. Es decir si subimos desde 6510 un poco más arriba vemos en 62E0 como comienza la IAT la zona cambia justo allí, y antes hay varios ceros, la finalización de la zona debería ser donde hay una seguidilla de nueve ceros o más. así que si bajamos buscando una cadena de ceros, vemos que en 6De0 termina la IAT, por lo tanto:

   IAT ADRESS: 4062e0
   Iat lenght o largo seria la resta de donde termina menos donde comienza o sea: **406de0- 4062e0 = b00**

   O sea que los datos que deberíamos colocar en el revirgin para el NOTEPAD serian Iat address: 62e0 (ya que es en offset) y Iat lenght = b00

2) La segunda parte era ver si entre 4062e0 y 406de0 están todas las direcciones de los JMP [xxxxxx]

   JMP [406514]
   JMP [406510]
   JMP [40650c]

Aquí vemos que los tres calores que copie 406514,406510 y 40650c están los tres dentro de esa región IAT.

Bueno esto seria como obtener los datos DE IAT ADRESS y IAT lenght en el NOTEPAD, veamos como hacerlo ahora en el COOL MOUSE.

Aclaremos ahora antes que nada que los programas ASPROTECT y comprimidos que se dicen destruyen la IMPORT TABLE lo que hacen es cambiar esta zona IAT, para que en vez de guardar en esas posiciones de memoria que apuntan directamente a las funciones (por ejemplo posiciones de memoria en la zona 7fe15891 , apunten a una parte del programa que se carga al descomprimirse el ASPROTECT generalmente por la zona de C00000 , y al DUMPEAR esa zona no se dumpea, ya que esta muy lejos y quedaría un ejecutable larguísimo, por lo que cuando el programa va a

JMP [406514] en ese caso vamos a la IAT en 406514 y el contenido es una dirección cerca de C00000 y de allí salta a la función es como un paso intermedio que al ejecutar el archivo DUMPEADO cuando este busca una función va a JMP [406514] y alli en esa posición de memoria esta por ejemplo C00000 y como no esta en la memoria cargada esa parte nos tira error.

Por lo tanto en revirgin lo que trata de hacer es volver a restaurar los saltos en la IAT directo a las funciones sin pasar por ese error.
Volviendo a poner en 406514 el contenido que haga saltar directo a la función (7fe15891) por ej.

Bueno basta de teoría y volvamos al Cool Mouse y a la acción:

Teniendo cargado el Cool Mouse en la memoria y abriendo el REVIRGIN hacemos click en el proceso CMOUSE.exe.

Cuando hay que encontrar la IAT no podemos cargar el que esta protegido con ASPROTECT con el ULTRAEDIT ya que esta comprimido, entonces cargamos con el ULTRA EDIT el EJECUTABLE DUMPEADO llamado DUMP.exe.
Y ponemos **FIND FF 25** para encontrar los saltos a la IAT y vemos que aquí hay mas que en el NOTEPAD, la primera vez que para es en 8094 y allí esta el salto a FF25 y a continuación están los números que indican la posición de memoria de la IAT AC D2 40 y la dirección de memoria es 40d2ac (esta al revés).

Luego hay varios FF 25 mas con distintas posiciones de memoria al lado, podemos ver la mínima y la máxima de todas esas posiciones y luego subir hasta donde empieza la IAT o con una sola la buscamos en el ULTRAEDIT y subimos para ver donde empieza.
Tomemos la primera que hallamos 40d2ac el offset es **D2Ac** lo buscamos en el ULTRAEDIT y si subimos un poquitos vemos que se ve claramente que la IAT comienza en D000 ya que antes hay una zona de muchos ceros, y busquemos el final de la zona IAT generalmente donde hay 9 ceros seguidos o mas, puedo poner la raya amarilla cliqueando en el inicio o sea en D000 e ir a FIND y poner que busque la cadena de nueve ceros o sea

000000000000000000, y como la raya amarilla esta en D000 busca a partir de allí hacia abajo, y la encuentra en D2AE que sería el ultimo número antes de que empiecen los ceros, por lo tanto ya tenemos el comienzo de la IAT y el final de la IAT, si chequeamos vamos a ver que todos los ff 25 que aparecen en la tabla de saltos caen todos aquí dentro.
Restamos para saber el largo de la IAT
D2AE -D000 = **2AE**

O sea que debemos colocar en el REVIRGIN como datos IAT address **D000** y IAT lenght **2AE.**
Con esos datos podemos ya trabajar con el revirgin.

Volvemos al revirgin y por supuesto tiene que estar activo el programa COOL MOUSE cuando hacemos click en el nos dice que la tabla esta corrupta, ponemos 0000d000 y 000002ae y hacemos click en IAT RESOLVER, luego nos pedirá que carguemos el ejecutable dumpeado DUMP.exe que tenemos en el escritorio, y con eso nos dará un primer resultado sobre la IAT a la derecha con los nombres de las funciones y las direcciones de memoria, miremos un poco eso.
Hay algunos casilleros que ya están resueltos y completos, y hay algunos que dicen redirected y hay uno (el ultimo) que esta vacio, los que dicen redirected ya están detectados, para que termine de llenar bien las casillas correspondientes le damos a RESOLVE AGAIN.
Bueno ya tenemos la tabla casi completa nos quedan dos lugares sin llenar, el que corresponde a B0c218 y la ultima casilla que corresponde B2246c.

Como vemos en estos dos saltos que quedan el programa protegido con ASPROTECT salta a B0c218 y
B2246c , mientras que las otras casillas ya han sido reparadas y saltaran directamente al valor de la funcion o sea **BFc03e8e** por ejemplo, que es un valor mayor, si nosotros no reparáramos estas dos entradas que quedan y dejáramos la tabla así cada vez que el programa llegara a una llamada a (por ejemplo en la ultima casilla)
JMP [40d2ac] ya que el contenido de 40d2ac es b2236c y como el programa dumpeado no llega hasta allí, nos daría error, hay que buscar reemplazarla por la función directa correspondiente y el salto a la zona de las funciones.
Bueno tomemos la ultima y marquémosla para que resalte, hagamos click derecho encima y elijamos TRACE al tracear ya nos cambia el valor de b2236c por 7fe16112 que es la zona de las funciones y nos aparece traced, ahora hagamos RESOLVE AGAIN y vemos que ya se completa esa casilla con el numero de la función GET OPEN FILE NAME.
Tratamos de hacer lo mismo con el otro casillero vacío que tenemos por el medio y como dice redirected no acepta que traceemos, aquí ya el programa no nos ayuda mas y tendremos que ir a fijarnos con el SOFTICE que función es la que usa el programa cuando llega a B0C218.

Como el programa esta funcionando pongamos un BPX DIALOGBOXPARAMA o cualquier otro que nos haga entrar al ejecutable y hagamos click derecho en el iconito del mouse que aparece en la barra de tareas, y elijamos SETTINGS, con eso parara en el SOFTICE, luego hagamos F12 y cuando aparezca la ventana pongamos OK y caeremos de nuevo en el SOFTICE, justito dentro del ejecutable CMOUSE.

Ahora hagamos U b0c218 para ver que función llama el programa allí, y vemos que unas líneas más abajo aparece KERNEL 32 - GETPROCADDRESS.

Volvamos al REVIRGIN y hagamos click derecho encima de la entrada que no esta terminada y pongamos EDIT, y escribamos el nombre de la función GetProcAddress, ponemos un BPX en B0c218, y arrancamos el programa de nuevo y cuando pare allí tracemos hasta ver la dirección de la función GETPROCADDRESS y es Bff76da8, reemplazamos en el REVIRGIN entonces escribimos en lugar de B0c218, el nuevo valor y KERNEL32.dll y GetProcADDRESS, hacemos RESOLVE AGAIN. (hay veces que cuando pones el valor de la función solo y haces RESOLVE AGAIN te completa solas las otras dos casillas sin necesidad de buscarlas a mano)

A esta altura antes de seguir conviene ir a SAVE RESOLVED y guardar un archivo con la tabla como esta hasta ahora, por cualquier error la podemos cargar con LOAD RESOLVED de nuevo.

Una vez que ya completamos todas las casillas vamos a IAT generator y si hay algo incompleto aun, nos dirá que hay algo todavía sin resolver, sino no nos dirá nada, hay que tener cuidado también , la entrada que escribimos a mano que no tenga errores ortográficos ya que a mí me paso que escribí GetProcadres con una sola d y una sola s, y me aparecía que estaba incompleto siempre.

Bueno vamos a IAT GENERATOR aquí no me dice nada de que hay algo mal por lo tanto sigo, me pide que le ponga un nombre para archivar la tabla y le pongo por ejemplo ITdump.exe, pero a la vez que lo guarda en un archivo, también lo agrega al archivo dumpeado, si ven los otros dos valores IT e IT LENGHT nos dice donde lo va a agregar o sea IT 51000 o sea al final del archivo que terminaba en 51000 va a agregar el largo 108 en mi caso probablemente en una nueva sección.

Bueno supuestamente con eso ya debería funcionar el archivo DUMP.exe , lo colocamos en el directorio donde esta el original Cmouse.exe, lo hacemos para probar
Nos habíamos olvidado de cambiar los bytes que modificamos para dumpear en el inicio cuando hicimos JMP EIP y debemos cargar el Dump.exe con el ULTRA EDIT para volverlos a su valor original.
En Find pongo la cadena **EBFEEC6AFF68B8D2** Y CAMBIO **EBFE** POR **558B** que eran los que se encontraban originalmente en el programa, una vez que guardamos los cambios, corremos el Dump.exe y chan chan ERROR DE WINDOWS que pasa aquí, veamos.

Pongamos SET FAULTS OFF y veamos donde fue en error, según sale en el cartelito del error es en 40821c, el tema si ponemos un BPX 40821c , y para allí vemos que no hay ningún llamado a una función, por lo tanto no es un error de que este mal hecha la IAT, lo más probable es que allí se esté verificando si esta presente el ASPROTECT en memoria como una protección por si alguien lo dumpea y lo arregla, por lo tanto, NOPEAREMOS esta instrucción a ver qué pasa vamos al ULTRAEDIT y allí en 40821c está la cadena 8a014184c07440f7c1 reemplazamos 8a 01 por 90 90 y guardamos los cambios ejecutamos y VOILA arranca y funciona perfectamente sin ningún error.

De cualquier manera esta es recién una introducción al tema REVIRGIN en el cual aprenderemos juntos, hay muchísimos compresores diferentes y aunque el método es siempre parecido lo iremos perfeccionando, y buscando como aplicarlo en otros casos más difíciles, ahora que el ejecutable esta descomprimido y se puede parchear directamente, les dejo la tarea de Registrarlo, para ustedes, yo me despido hasta la LECCIÓN 35 y vamos a tratar de seguir con otro programa que haya que reparar con el revirgin, así le vamos tomando la mano a usar esta buenísima pero difícil herramienta.

Ricardo Narvaja

IAT ADDRESS e IAT LENGTH

(LECCION TREINTA Y CINCO)

Esta no es una lección completa sino un pequeño apéndice de la LECCIÓN 34 donde aprendimos a utilizar el REVIRGIN, y vimos también que quizás lo más difícil era localizar la dirección donde comienza la IAT y el largo.

Bueno, ya no es necesario buscarlo a mano, pues hay un programa que lo hace automáticamente, y te dice la IAT ADDRESS Y IAT LENGHT, se llama

Import REConstructor v1.2

Yo lo baje de
\
http://www.suddendischarge.com/files/Unprotection%20Tools/imprec12.zip

o también esta en

http://www.suddendischarge.com/

y lo buscamos en el SEARCH por su nombre y allí esta.

Este programa se supone que hace lo mismo que el REVIRGIN , aunque a mi no me dio tan buen resultado para terminar de reconstruir bien la tabla de importaciones, por lo menos automáticamente nos encuentra la IAT address y IAT lenght necesarias para trabajar en el REVIRGIN.
El que quiera probarlo para reconstruir completamente la tabla de importaciones , hágalo, quizás funcione bien, pero la costumbre de utilizar el REVIRGIN además de que este Import REConstructor v1.2 suele colgarse bastante, hace que lo use solo para que me diga los valores que necesito y llevarlos al Revirgin para continuar allí con el trabajo.
Cuando lo ejecutamos al IMPORT , vamos a ATTACH TO AN ACTIVE PROCESS ya que como en el revirgin el proceso a reconstruir debe estar funcionando en la memoria, y allí elijo el que quiero estudiar, arranquemos el COOL MOUSE original que no esta desprotegido para ver que nos dice sobre la IAT LENGHT y ADDRESS.
Busco entre los procesos el CMOUSE.exe y lo cargo, una vez que se carga coloco el ENTRY POINT que halle al dumpear en la casilla OEP en este caso es 40876b, o sea debemos llenar la casilla con el OFFSET por lo tanto será **876b, lo que coloquemos en la** casilla OEP y luego iremos a IAT AUTOSEARCH y cuando apretemos ese botón nos aparecerá en RVA =d000 y Size= 2b4 que son los valores donde comienza la IAT y su largo como vimos en la LECCIÓN 34 averiguándolos a mano.

IAT ADDRESS=D000 y IAT LENGHT =2b4

En la LECCIÓN anterior habíamos visto

"O sea que debemos colocar en el REVIRGIN como datos **IAT address = D000** y **IAT lenght = 2AE."**

Aquí vemos que el lenght lo calcula un poco mas largo justo entre los ceros donde termina la tabla, pero funciona igual, y si ponemos estos valores en el REVIRGIN vemos que la tabla que calcula es exactamente la misma, siguiendo el método que vimos en la LECCIÓN ANTERIOR.
La idea era simplificar un poco la forma de encontrar la IAT ADDRESS e IAT LENGHT ya que como vimos en la LECCIÓN anterior explicar como obtener esos datos me llevo casi 2 paginas bastante difíciles, aquí en el programa con solo ponerle el dato del ENTRY POINT que ya lo tenemos nos calcula directamente los valores necesarios sin rompernos la cabeza ni buscar como locos a mano.

Bueno, espero que esto le facilite las cosas al que quiera reconstruir una tabla de importaciones, no lo puse en la LECCIÓN ANTERIOR sencillamente porque no lo sabia en ese entonces y me enteré recién ahora, igual no les va a venir mal saber hallar a mano esos valores.

Ricardo Narvaja

EN LA CARA DEL NUEVO ASPROTECT 1.2

(LECCION TREINTA Y SEIS)

Seguimos por varias lecciones mas con el tema Revirgin, y ahora haremos la reconstrucción de el ejecutable de un programa protegido con el ultimo y nuevo e imbatible (hasta ahora) ASPROTECT 1.2 y que mejor para probar si podemos con el que mojarle la oreja al propio ASPROTECT , reconstruyendo el ejecutable del mismo ASPROTECT 1.2 que esta protegido con ASPROTECT 1.2, jua jua.
Se baja de
http://www.aspack.com/files/asprotect12.zip

El primer paso es arrancar el ICEDUMP, una vez que esta cargado, tenemos que hallar el ENTRY POINT y aquí en esta versión esta un poco mas complicado, pero se puede hacer:

Aquí puedo usar un BPX GETVERSION o BOX GETPROCADDRESS, cualquiera e los dos a la tercera o cuarta vez que para y volvemos al ejecutable con F12, nos damos cuenta que estamos en el momento que se esta descomprimiendo ya que en la línea verde que siempre aparece el nombre del programa que esta ejecutándose aquí no aparece nada, o sea sin nombre, y si vemos con el PEEDITOR las secciones del ejecutable vamos a ver que algunas no tienen nombre, es obvio que esta ejecutándose alguna de estas secciones.
Usamos el famoso BPR INICIO FINAL r if (eip>=INICIO) && (eip<=FINAL)

Dado que aquí esta en alguna de esas secciones sin nombre, podemos pensar que en ENTRY POINT estará en la primera sección y casi siempre es así, usaremos pues como INICIO = 400000 y FINAL el final de la primera SECCION o sea 459000, lo que podemos ver con el PEEDITOR, probaremos.

BPR 400000 459000 r if (eip>=400000) && (eip<=459000)

Bueno largamos y vemos que la primera vez que para es en **401014 RET**, pero es obvio que este no es el ENTRY POINT ya que no hay programa que comience en un RET (creo, jua jua), así que le doy a X y ENTER de Nuevo para que siga ejecutándose y para la segunda vez en
458f04 PUSH EBP
este tiene un buen aspecto pues es un PUSH EBP, pero ASPROTECT tiene sus tretas así que ejecutemos cinco o seis sentencias para ver que pasa con F10, llega a un RET y vuelve al compresor sin nombre, jua, jua, es un ENTRY POINT falso, puesto para despistar.
Vuelvo a hacer X y ENTER y parara en
458f14
458ef4
Todos **PUSH EBP** y todos falsos ya que enseguida llegan a un RET y vuelven al compresor lo que nos damos cuenta ejecutando cinco o seis líneas.
Al fin LLEGA a
458fD8 PUSH EBP

y este es el verdadero ENTRY POINT ya que no vemos ningún RET y si ejecutamos vemos que no vuelve nunca mas al COMPRESOR SIN NOMBRE.
Bueno, una vez que para aquí, tenemos que DUMPEAR

Ya comprobé que no es necesario con el ICE DUMP hacer el LOOP infinito JMP EIP, así que no necesitamos cambiar nada, una vez que este en 458fd8 escribimos.

/DUMP 400000 49e000 c:/Temp/asp.exe

El 400000 y 49e000 lo sacamos del PEEDITOR son la IMAGE BASE y el largo total SIZE OF IMAGE 9e000.
Por lo tanto si dumpeamos la memoria entre 400000 y 49e000 volcaremos todo lo que tenemos en la memoria.
Anotemos este valor de ENTRY POINT = 458fd8 para colocarlo en el IMPORT RECONSTRUCTOR para que nos calcule el IAT ADDRESS y IAT LENGHT o sea la dirección donde comienza la IAT y el largo.
Usando el Import Reconstructor como vimos la LECCIÓN pasada , buscamos donde dice ATTACH AN ACTIVE PROCESS el proceso del Asprotect.exe y una vez que lo abrimos ponemos en OEP el ENTRY POINT hallado, en offset 58fd8 y al cliquear en IAT AUTOSEARCH obtenemos los valores que necesitamos
RVA = 66114 y SIZE = 5CC estos son los valores que necesitamos en el REVIRGIN para usarlo

IAT ADDRESS= 66114 y LENGTH O SIZE= 5CC

Cierro el IIMPORT RECONSTRUCTOR y abro el REVIRGIN, y busco allí el proceso Asprotect.exe que debe estar abierto .
Coloco esos datos en IAT ADDRESS = 66114 Y LENGHT = 5CC y le doy a IAT RESOLVER, me pide el nombre del ejecutable dumpeado que ya arregle con el PEEDITOR como vimos en las lecciones anteriores (O sea abro las secciones y pongo DUMPFIXER para que repare el archivo DUMPEADO), y luego le doy nuevamente a RESOLVE AGAIN.
La verdad la tabla queda bastante completa, se arregla bastante bien , solo una entrada queda sin solucionar, es la de **C1c424** que ni traceando te la arregla.
Bueno arrancando de nuevo el programa original y haciendo que pare en el ENTRY POINT, allí pongo bpx C1c424 para ver que pasa allí, y entonces veo que unas líneas más abajo esta
GetProcAddress y al entrar en ese CALL la dirección de la función es **BFF76DA8**

Bueno llevo esos datos al REVIRGIN pongo la tilde en EDIT y lleno los casilleros vacíos, el primero con
KERNEL32.dll luego **GetProcAddress** y donde dice **C1c424** pongo **Bff76da8**
Hago Resolver y lo toma, uso la opción AUTOFIX SECTIONS + IT PASTE tildada como vimos las lechones anteriores, voy a IAT GENERATOR y me dice donde quiero guardar la tabla, la guardo en cualquier lado, pero a su vez me la agrega al archivo dumpeado que ya había hecho con el ICEDUMP y reparado con el PEEDITOR, además por si acaso me hace una copia con extensión bak del archivo como era original por si algo sale mal.

Bueno con eso y luego de arreglar el entry point del archivo y colocarlo en 458fd8 o sea 58fd8 en offset y copiar el archivo arreglado a la carpeta donde esta el ejecutable original, lo arranco y salen errores en varios lugares.
Lo miro con el WDASM luego de cambiar las características de la sección a E0000020 con el PEEDITOR y veo que las Funciones Importadas aparecen perfectas e impecables.
Bueno desde el SYMBOL LOADER me dispongo a tracear para reparar lo que es seguramente alguna protección de ASPROTECT para que de error si alguien lo arregla y veo que en los lugares que da error es un CALL [464fac] o CALL [464fb0] y que si nopeo estos calls arranca perfectamente el programa y funciona bien.

Puedo buscar con el ULTRAEDIT la secuencia de estos calls ya que siempre son iguales para saber donde estan y nopearlos
Ya que esos CALL tiene los siguientes valores HEXA, en el ULTRAEDIT pogo FIND y busco en que lugares aparecen:

FF15ac4f4600
FF15b04f4600

Y veo que hay varios lugares donde esta, entre otros:

45495f
454b3a
453437
45777d
458cc6
4579c4
453480

Nopeando en estas direcciones de memoria las llamadas de estos CALL que te direccionan a Cxxxxx el ejecutable funciona perfectamente, es obvio que estos CALL están puestos para que si alguien lo descomprime le de error, por que nopeandolos funciona perfecto el ejecutable, y arranca con todas las funciones activas.
Bueno con esto ya esta limpito, limpito de protección Asprotect, les queda a ustedes la tarea de crackearlo .

Costo pero valió la pena, jua
Gracias a METAMORFER que me enseño como hacerlo.

Ricardo Narvaja

SEGUIMOS APRENDIENDO A USAR REVIRGIN

(LECCION TREINTA Y SIETE)

Esta vez es el turno de otro programa protegido con ASPROTECT, ya que mas adelante veremos otras protecciones, este se baja de

http://www.helexis.com/ic/icatc303.zip

Se llama ICON CATCHER y la versión 3.03 viene protegida con ASPROTECT, aquí en este caso, aprovecharemos dado que la reconstrucción de la tabla no es difícil, sino la verdadera dificultad esta en hallar el ENTRY POINT.
Preparo el ICEDUMP para que este listo y poniendo un BPX GETPROCADDRESS , una vez que arranca el programa y para dentro del SOFTICE a la tercera vez que para ya nos aparece en la esquina inferior derecha IC que significa que paro dado el proceso ICON CATCHER.
Si antes miramos en el PEEEDITOR donde comienza y donde finaliza la primera sección para trasladar estos datos , al SOFTICE escribiendo el famoso BPR

BPR INICIO FINAL r if (eip>=INICIO) && (eip<=FINAL)

En este caso INICIO = **400000** FINAL = **49d000**

por lo tanto borramos el BP que había con BC* y luego escribimos

BPR 400000 49d000 r if (eip>=400000) && (eip<=49d000)

y le damos X para ver donde para

Anoto estos lugares donde fue parando

401000 RET no es un entry point
401005 RET lo mismo
401014 RET ídem
48e38c PUSH EBP

este parece ser un ENTRY POINT pero si mantenemos apretado F10 vemos que después de un tiempo vuelve al compresor por lo que no es el ENTRY POINT

vuelvo a ejecutar con X y para en
40687c PUSH EBP este parece mas, ya que si mantengo apretado F10 no vuelve al compresor y cuando sale del ejecutable es para encontrar alguna función.
Yo Dumpee aquí habiendo buscado el **INICIO = 400000** y **SIZE OF IMAGE =E2000** con el PEEDITOR
/DUMP 400000 e2000 c:/Temp./icon.exe

Con el Import Reconstructor y el programa ICON CATCHER funcionando, le puse el ENTRY POINT hallado en offset 687c y me dio la **IAT ADDRESS = a1164** y **SIZE=738** que es correcta ya que si vemos con el ULTRA EDIT y buscamos con el método de FF 25 para verificar, vemos que los saltos caen en esa posición de memoria (4a1164)
Abro el REVIRGIN y busco el proceso de ICON CATCHER y le pongo IAT ADDRESS =**a1164** y el tamaño IAT LENGHT= **738**.

Me sale una primera tabla, me pide el ejecutable dumpeado lo busco, y luego le doy a RESOLVE AGAIN para que termine, salen solo dos entradas sin resolver, que ni traceando, se resuelven.
Las dos apuntan a C6c424 así que la función es la misma en ambas.
Tengo que fijarme corriendo el programa original adonde va cuando llega a C6c424 así que lo ejecuto y pongo por ejemplo algún BPX que pare o sino
Bpr 40687c 40687d r
que en unas dos o tres veces parara en 40687c el ENTRY POINT.
Una vez allí me fijo con **u c6c424** y veo que la función es GetProcAddress lo único que me quedaría es poner un BPX alli para ver la dirección exacta cuando entra, así que pongo un BPX **c6c424** y corro el programa cuando para allí traceo con F10 y cuando entra a la función copio la dirección de memoria, en mi caso es **Bff76da8** aunque puede variar según la maquina.
Vuelvo al Revirgin y pongo la tilde en EDIT y completo a mano los cuadros vacíos sin errores sino no funcionara en el primero pongo **Kernel32.dll** el numero no lo se, así que pongo 0001, el programa lo corregirá solo, pongo el nombre de la función **GetProcAddress** y la dirección reemplazo **C6c424** por **Bff76da9** o lo que obtuvieron ustedes y le doy a RESOLVE AGAIN y lo toma, es de notar que si a veces dejo 0000 en el numero, no toma la función y vuelve a poner redirected, igual cambió solo el 1 por el valor correspondiente que es 1A3 en mi caso.
En la otra entrada copiamos los mismos datos ya que va al mismo lugar.
Luego de RESOLVER AGAIN y que esta todo lleno, vamos a IAT GENERATOR y ya esta listo, cambiamos el ENTRY POINT con el PEEDITOR ponemos **687c** , supuestamente debería estar listo, lo copio a la carpeta del programa y lo arranco y no funciona por que?
No es ninguna protección de ASPROTECT ni nada de eso, la tabla esta correcta y no hay errores allí, porque no funciona?
El ENTRY POINT no es el correcto, como lo supe?
Después de mucho bregar me di cuenta que apretando F10 el ejecutable se seguía descomprimiendo aunque sin volver al compresor ASPROTECT, como si estuviera comprimido dos veces, primero con Asprotect y ahora sin salir del ejecutable, lo que hace inservible el método del BPR para seguir a partir de aquí.
Vencidos, casi, pero alguien (GRANDE VIPER) me soplo una ayuda: el DEDE te dice el ENTRY POINT si el programa es en DELPHI, tendremos suerte?
Abro el DEDE voy a TOOLS - DUMP ACTIVE PROCESS y allí vemos que la mayoría de los programas que están funcionando cuando los cliqueamos no se activa el botón **GET RVA ENTRY POINT ,** si hacemos click encima del Icon Catcher si se activa , ya que solo se activa con los programas escritos en DELPHI y este lo es, apreto el botón y me dice el ENTRY POINT que es **49d22c.**
Tengo que realizar el dump nuevamente así que repito el proceso, con
BPX GETPROCESSADDRESS y allí puedo hacer que pare en 49d22c con

BPR 49d22c 49d22d r , así después de tres o cuatro veces parará en el ENTRY POINT 49d22c , allí dumpeo igual que antes, arreglo con el PE EDITOR y la opción DUMPFIXER, cambio el ENTRY POINT al nuevo, realizo el mismo trabajo para con el Revirgin agregarle la IMPORT TABLE exactamente igual que antes , y una vez que voy a IAT Generator , y guardo la tabla nueva, el ejecutable sin ninguna reparación posterior copiándolo a la carpeta del programa funcionará perfectamente sin problemas y como siempre en estos últimos ejemplos les dejare el tema de registrarlo para que practiquen, ya que no es muy difícil, una vez desprotegido de ASPROTECT.

Ricardo Narvaja

EL ASPROTECT MÁS DIFICIL QUE VI

(LECCION TREINTA Y OCHO)

CONTROL FREAK para usar el Revirgin brrr.

Asprotect 1.2 tiene varias versiones parece y dado que ya hemos desprotegido algunos programas compactados con el, veremos este que parece ser una versión especial, mas difícil que la común, quizás al pagar mas haya versiones mas difíciles, no lo se.
La cuestión de que arreglar la tabla de importaciones de este me costo bastante mas que cualquiera de los otros ASPROTECTS y fue muy difícil, ni siquiera estoy seguro de que sea ASPROTECT ya que no es reconocido por ningún identificador al día de hoy, pero es casi seguro, igual lo descomprimiremos.
La pagina del programa es:

http://www.hace.us-inc.com/

Aunque ya existe allí la versión 1.01 que no debe diferir mucho de esta, pero será distinta así que subiré la 1.0 al Freedrive para el que quiera practicar.

Bueno con el PEDITOR busco la IMAGE BASE = 400000 y la IMAGE SIZE = 8b000

Arranco el ICE DUMP, y con BPX GETVERSION o BPX GETPROCADDRESS caigo dentro del SOFTICE, y allí después de dos o tres veces al volver con F12, quedare en el descompresor ASPROTECT aunque aquí tampoco aparece ningún nombre en la línea verde donde dice el nombre de lo que se esta ejecutando.
Bueno la primera sección empieza en 400000 y termina en 466000, podemos ver con el PEEDITOR así que hacemos con el famoso BPR.

BPR 400000 466000 r if (eip>=400000) && (eip<= 466000)

Parara en **465654** que es el verdadero ENTRY POINT, allí dumpeo.

/DUMP 400000 8b000 c:/Temp./cfreak.exe

Reparo con el PEEDITOR y DUMPFIXER al archivo DUMPEADO, y cambio el ENTRY POINT a 65654 offset así ya quedara listo cuando le arregle la tabla de importaciones.

Voy al IMPORT RECONSTRUCTOR elijo el proceso del CONTROL FREAK y pongo el ENTRY POINT = 65654 y me sale IAT ADRESS = 69150 y IAT LENGHT = 673.

Traslado estos datos al Revirgin y abro el proceso del CONTROL FREAK, y hago RESOLVER, cargo el dumpeado, y luego RESOLVE AGAIN.
Los resultados no son muy alentadores la tabla tiene diez posiciones sin resolver, el máximo de lo que he visto hasta hoy con ASPROTECT.

Con el método de ir a fijarse al programa original uno por uno, cargando el programa y poniendo primero BPR 465654 465655 R para que pare allí, en el ENTRY POINT y una vez que pare, pongo BPX en las direcciones que nos dice el REVIRGIN que va el programa.

000691A4 00C0C468 0000 KERNEL32.dll GetProcAddress

Aquí debemos buscar en COC468 y cuando entra en la función, en mi caso la dirección de memoria es bff76da8, reemplazo C0c468 por esa dirección que encontré y cambio el numero 0000 por 0001 y le doy a RESOLVE AGAIN, y ya esta solucionada esta entrada, así soluciono:

691a4 GetProcAddress bff76da8
691a8 GetModulehandleA bff77716
69250 GetModulehandleA bff77716
69314 GetProcAddress bff76da8
69318 GetModulehandleA bff77716

La ultima columna puede variar según cada caso, hay que hallarlo uno mismo.
Ahora quedan cinco que si pongo un BPX y voy a ver no hay ninguna función

691b8 c0686c
692c8 c0c874
69300 c0c834
69334 c0c864
6933C c0c87c

Estas son las entradas que no se pueden arreglar.
No importa una vez que arreglo las cinco primeras voy a IAT GENERATOR y aunque me dice que no está terminada de solucionar acepto igual.
Entrando de nuevo en el programa y poniendo BPX en las direcciones de la segunda columna para ver que pasa, copiaremos las sentencias que hallamos.

C0C86C A1 78 36 C1 00 MOV EAX , [c13678]
C3 RET

C0C834 A1 D8 35 C1 00 MOV EAX, [C135D8]
C3 RET

C0C864 A1 E0 35 C1 00 MOV EAX, [C135E0]
C3 RET

C0C874 55 PUSH EBP
8B EC MOV EBP, ESP

	5D	POP EBP
	C20400	RET 0004

C0C87C	55	PUSH EBP
	8B EC	MOV EBP, ESP
	5D	POP EBP
	C20400	RET 0004

Estas son las cinco entradas que no se pueden RESOLVER, y el programa las necesita ya que si el DUMPEADO busca en el primer caso por ejemplo ir a C0c86c cae a error ya que esa parte no está en el DUMP, y tampoco si ponemos un RET el programa arranca tenemos que tratar de que ejecute las mismas sentencias que el programa original.

Nos queda otra cosita, veamos el primer caso

C0C86C	A1 78 36 C1 00	MOV EAX , [c13678]
	C3	RET

Tenemos que ver cual es el contenido de c13678, para tomar ese valor constante y hacer que EAX, tome ese valor directamente sin buscar en el contenido de c13678 lo cual también daría error.

Vamos de nuevo allí con el SOFTICE y vemos que (en mi caso ya que este número varía en cada computadora) EAX toma el valor 81819974.
O sea que si yo en donde se encuentra el salto en el DUMP reemplazo el salto a C0c86c por directamente MOV EAX, 81819974 y RET, el programa funcionaria de la misma forma que el original.

Busco en el ULTRA EDIT el FF 25 del primer caso que es FF 25 b8 91 46 (las tres ultimas cifras son 4691b8 al revés que es el dato del Revirgin 691b8 de esa primera entrada)
Casualmente

FF 25 b8 91 46 00 que es jmp [4691b8]
tiene la misma cantidad de bytes por lo tanto lo reemplazo
b8 74 99 81 81 c3 que es MOV EAX, 81819974 y RET

Con lo que el programa funcionaria igual

En la segunda entrada
FF 25 00 93 46 00 que es JMP [469300]
lo reemplazo por
b8 04 0a 00 c0 c3 que es MOV EAX, c0000a04 y RET

C0000a04 es el numero que busque con el SOFTICE que mueve a EAX.

en la tercera entrada
FF 25 34 93 46 lo reemplazo por B8 0b cb 06 ff c3

En mi caso C3 ff 06 cb es el numero que pasa a EAX.

En los dos últimos casos ya que aquí no hay transferencia de ninguna posición de memoria sino solo de registros entre si, busco la cadena 55 8B EC 5D C2 04 00 en el dump y aparece exactamente igual en 416040 o sea offset 16040, si redirigimos estas dos entradas allí funcionara bien.

Ya que los dos últimos saltos son

JMP [4692c8] y JMP [46933c]

Cambiando los contenidos de 4692c8 y de 46933c por 416040 saltara dentro del dump y ejecutara los mismos comandos.
Voy al ULTRA EDIT y busco el offset 692c8 allí encuentro 74 c8 c0 que era donde saltaba originalmente (c0c874), lo cambio por 406041 y en el segundo caso igual y habremos logrado terminar.

Con eso en la propia computadora de uno funcionara, pero se me ocurrió instalar el programa en otra computadora y llevar el ejecutable reparado a la otra maquina y no funciono, compare los números que lleva a EAX en las tres entradas que arreglamos y son diferentes, si los cambio por los que genera en esa máquina arranca allí también, probé además que los dos números últimos pueden ser cualquiera, el tema estaba en la primera entrada.

MOV EAX, 81819974 y RET

Aquí era donde se determinaba si funcionaba en otra máquina o no, ya que mas adelante tomaba el contenido de 81819974 y daba error ya que era una zona invalida, probé cambiar el numero 81819974 por 00465654 que es el ENTRY POINT y siempre su contenido tendrá un numero valido, y arranco en las dos maquinas, jua jua, ese era el truco, la verdad que no vi hasta ahora ningún caso de reconstrucción de tabla tan difícil como este, si alguien consigue un método mas fácil, please, avíseme, jua jua.

Ricardo Narvaja

CRACKEANDO EN 16 BITS

(LECCION TREINTA Y NUEVE)

Bueno ya que no pude terminar lo que había prometido para esta semana, lo dejamos para más adelante y seguimos con otro tema pendiente, crackear en 16 bits.

Ojo esto no es solo un tema pendiente en las lecciones porque si, yo odio crackear en 16 bits, le esquivo como a la viruela, y tengo mas dudas que certezas en este tema, y no creo ser el único, muchos a los que consulte sobre el tema me ignoraron olímpicamente, muy probablemente por que tampoco les es cómodo.

Ahora ¿**por qué es difícil?**
Es verdaderamente difícil? o es la falta de costumbre ante casi el 95 por ciento de programas que son en 32 bits y el 5 % restante solo es en 16 bits, como no estamos acostumbrados nos cuesta?
No se , a mi me cuesta horrores y tengo que usar mas trucos y cosas raras que nunca, quizás por desconocimiento, y si alguien sabe de alguna forma mejor de hacer esto que me lo diga así aprendo.
Este es en las casi 40 lecciones el primer programa en 16 bits que crackeamos y tampoco vamos a crackear uno todos los días, pero por ser el primero busque uno muy fácil (para mi) ya que solamente consiste en eliminar un cartel.
Se baja de

http://www.boennrep.de/dl0206/BRep450.exe

y es un programa sobre medicina y síntomas que funciona 100 % solo que cada tres segundos aparece un cartelito molesto, que no te permite usarlo con tranquilidad, lo tenés que estar cancelando y a los dos o tres segundos aparece de nuevo, el tema es eliminar ese cartel molesto.
Hay dos tipos de programas de 16 bits o modo DOS que yo conozco, los que funcionan dentro de WINDOWS y los que lo hacen si o si fuera de WINDOWS.
Este y todos los que veremos estarán en el primer caso o sea dentro de WINDOWS, los del segundo caso, tendrán que buscar otro profesor, jua jua.
Si ya para un programa de 16 bits que funciona en WINDOWS no funcionan la mayoría de las herramientas (PROCDUMP, ni PEEDITOR, ni PUPE, ni DEDE, ni REGMON, ni FILEMON, ni muchísimas herramientas) para los que funcionan fuera no funciona ninguna de las conocidas solo hay una versión de SOFTICE vieja para DOS que es mas mala que la peste y si el programa te la detecta no hay FROGSICE que te salve, hay que hacer todo a mano.
Por lo tanto reagrupemos las fuerzas y veamos con que contamos: con el SOFTICE, con el WDASM, y con el ULTRAEDIT creo que otra cosa no funciona para 16 bits, nos tendremos que arreglar.
COMO DIFERENCIAMOS UN PROGRAMA DE 32 bits de uno de 16 bits (valor ya no quedan muchos y cada vez son menos) de esta forma:

Esto es sacado del WDASM de un programa de 16 bits

16 bits

0010.13A7	668B460E	mov eax, [bp+0E]
:0010.13AB	050800	add ax, 0008
:0010.13AE	6650	push eax

32 bits

:**00403F3A**	57	push edi
:00403F3B	50	push eax
:00403F3C	8D4505	lea eax, dword ptr [ebp+05]

Aquí se ve la diferencia, en el de 32 bits la dirección de memoria es un numero de 8 dígitos por ejemplo **00403f3a** y en el de 16 bits es un numero de cuatro dígitos precedido de un numero de bloque, por ejemplo **0010:13a7.**
Esto que parece tan simple tiene sus cuitas, ya que en 32 bits el numero te indica ya directamente una posición, si en el WDASM ves 403f3a sabes que en el SOFTICE va a ser 403f3a y la encontrás fácil, en cambio en 16 bits la cosa cambia.

El WDASM empieza a contar los bloques del programa desde 0001 o sea desde el primer bloque y el SOFTICE no, en la computadora ese programa no esta en el primer bloque sino en el 5375 (POR EJEMPLO) por lo tanto, la cosa se pone peliaguda.
Si el Wdasm dice 0001:1345 solo sé que será 1345 pero vaya a saber de que bloque, (SI ALGUIEN SABE CALCULAR EXACTAMENTE COMO ENCONTRAR FÁCILMENTE UN BLOQUE DEL WDASM EN EL SOFTICE SIN ANDAR TANTEANDO Y SIN USAR EL SEARCH AYUDE A ESTE POBRE CRACKER EN DESGRACIA)

En resumidas cuentas si yo se que es 1345 en el WDASM y lo tengo que encontrar en el SOFTICE puedo hacer la siguiente trampa:
Correr el programa poner algún BPX que se que use el mismo y una vez adentro, buscar la secuencia de números hexadecimales (cadena) que en el WDASM corresponden a esa sentencia y en el SOFTICE buscarla
S 0 l ffffffff 44,55,66,77, etc

O sea 44 55 66 77 etc es la cadena de valores hexadecimales de la sentencia que encontré en el WDASM, y si es al revés pasar del SOFTICE al WDASM o busco si no son muchos bloques uno por uno, por ejemplo si era 5324:1234
busco en 0001:1234 veo si están las mismas sentencias, y así sigo con 0002:1234 hasta que las encuentro.
Esto quizás parecerá un poco precario, pero hasta hoy nadie me dio ni explico una forma mejor de hacerlo (ESCUCHO SUGERENCIAS, EH?)

Bueno empecemos con el programa lo desensamblo con el WDASM al ejecutable, y encima que tarda como media hora, una vez desensamblado no está la STRING REFERENCE que buscamos, gr…
Encima dentro de la carpeta del programa hay varios dll que puede ser que en alguno de ellos este, desensamblo uno por uno, buscando la STRING pero nada de nada no está. Vamos al SOFTICE, como parece ser una MESSAGEBOX (SIN la A al final ya que A es 32 bits y sin la A es 16 bits) pongo un BPX MESSAGEBOX y corro el programa espero unos segundos y faaaa, cuando quiere aparecer el cartelito, salta el softice, apreto F12 ACEPTO en el cartelito y vuelvo al SOFTICE. Allí caigo (en mi caso) en

1927:1b45 mov [ebp-02],ax

Que es la sentencia siguiente al CALL Messagebox

Y veo que no esta en el ejecutable principal sino en una dll llamada SUWIN30.dll que si la busco esta en la carpeta del programa, la copio a otro lado para trabajar tranquilo y la abro con el WDASM, para no buscar como tarados uno por uno todos los bloques abrimos las Funciones importadas buscamos USER.MESSAGEBOX y hacemos click encima de la funcion hasta que atrás aparezca la parte buscada, tenemos suerte que al hacer click sobre messagebox solo aparecen dos lugares, uno de ellos es 1b40.
Es 0013:1b40

Aquí esta en el WDASM la misma parte que encontramos en el SOFTICE

```
:0013.1B40    9AFFFF0000      call USER.MESSAGEBOX
:0013.1B45    8946FE          mov [bp-02], ax
:0013.1B48    57              push di
```

Era el 0013 :1b40

esta el call maldito y mirando arriba no hay ningun salto condicional que lo pueda evitar para forzarlo a saltarlo, busco de donde proviene esto en la REFERENCE que hay un poco mas arriba pero sigo y sigo para atrás sin encontrar ninguna forma de evitar este MESSAGEBOX y si lo NOPEO me da Error de Windows, grrr..
Como puedo hacer que no aparezca este cartel maldito?

Voy a ver si puedo insertar un JUMP un poco mas arriba del cartel para saltearlo, veo esta parte:

```
:0013.1B11 51              push cx
:0013.1B12 56              push si
:0013.1B13 51              push cx
:0013.1B14 56              push si
:0013.1B15 8976D0          mov [bp-30], si
:0013.1B18 894ED2          mov [bp-2E], cx
:0013.1B1B 9A7E160000      call KEYBOARD.OEMTOANSI
:0013.1B20 FF7606          push word ptr [bp+06]
```

:0013.1B23 9AFFFF0000 call USER.MESSAGEBEEP
:0013.1B28 57 push di
:0013.1B29 8D46D4 lea ax, [bp-2C]
:0013.1B2C 16 push ss
:0013.1B2D 50 push ax
:0013.1B2E 6A00 push 0000
:0013.1B30 0E push cs
:0013.1B31 E85200 call 1B86
:0013.1B34 57 push di
:0013.1B35 66FF76D0 push word ptr [bp-30]
:0013.1B39 66FF760C push word ptr [bp+0C]
:0013.1B3D FF7606 push word ptr [bp+06]
:0013.1B40 9AFFFF0000 call USER.MESSAGEBOX
:0013.1B45 8946FE mov [bp-02], ax

Voy a tratar de insertar un JUMP en 1b11 que salte el MESSAGEBEEP Y EL MESSAGEBOX, pero si lo hago saltar a 1b45 que es la sentencia justo posterior al cartelito MESSAGEBOX me sale error de WINDOWS, veo que mas abajo hay un salto condicional

:0013.1B58 7412 je 1B6C

probare hacer un JUMP a 1b6c directo desde 1b11 puedo probarlo con el SOFTICE una vez que para en el MESSAGEBOX, pongo un BPX 1b11 y una vez que para allí hago

A [ENTER]

y escribo JUMP 1b6c
para que en la memoria quede esa sentencia y borro con BC* y hago X para volver al programa HMMM desapareció el cartelito y no sale mas, jajaja, sin error de WINDOWS ni nada, había que saltar a ese punto, fue un buen intento.
Ahora abro el SUWIN30.dll con el ULTRA EDIT y la cadena que copie antes de cambiar con el SOFTICE era 515651568976d0 y reemplazo el 5156 por EB59 que es el salto a JUMP 1b6c

Era originalmente así
:0013.1B11 **51** push cx
:0013.1B12 **56** push si
:0013.1B13 51 push cx
:0013.1B14 56 push si

y quedo así luego de modificado:

:0013.1B11 **EB59** jmp 1B6C
:0013.1B13 51 push cx
:0013.1B14 56 push si

Por suerte para mi programa crackeado, uff… espero no torturarlos con programas de 16 bits por un tiempo.

Ricardo Narvaja

UNA DE VISUAL BASIC

(LECCION CUARENTA)

El programa víctima se llama MECAMATIC 1.1 se baja de

http://leo.worldonline.es/antgarco/MecaMatic%20v.1.1.zip

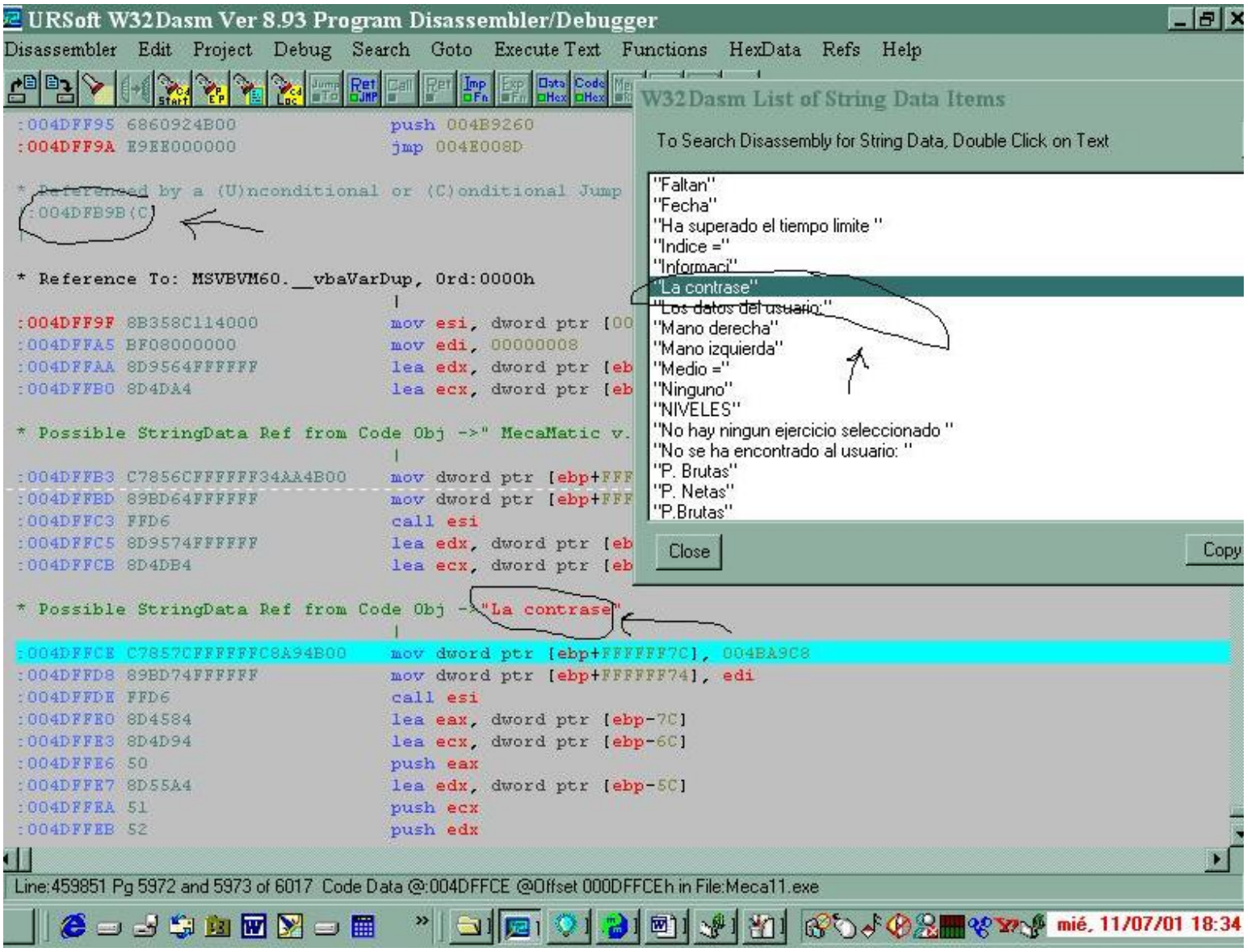
y es un programa para aprender a escribir a máquina y está en castellano y en VISUAL BASIC, te das cuenta pues al instalarlo, ya te dice que esta copiando las librerías de VISUAL BASIC 6, el Wdsam lo abre perfectamente , no está comprimido ni nada, hay que usar el WDASM modificado para VISUAL BASIC, no el de DELPHI.

Una vez que se abre el programa nos pide un nombre de usuario, para iniciar, y luego yendo al menú de ayuda, allí está la ventana para registrarse nos pide nombre y número de serie

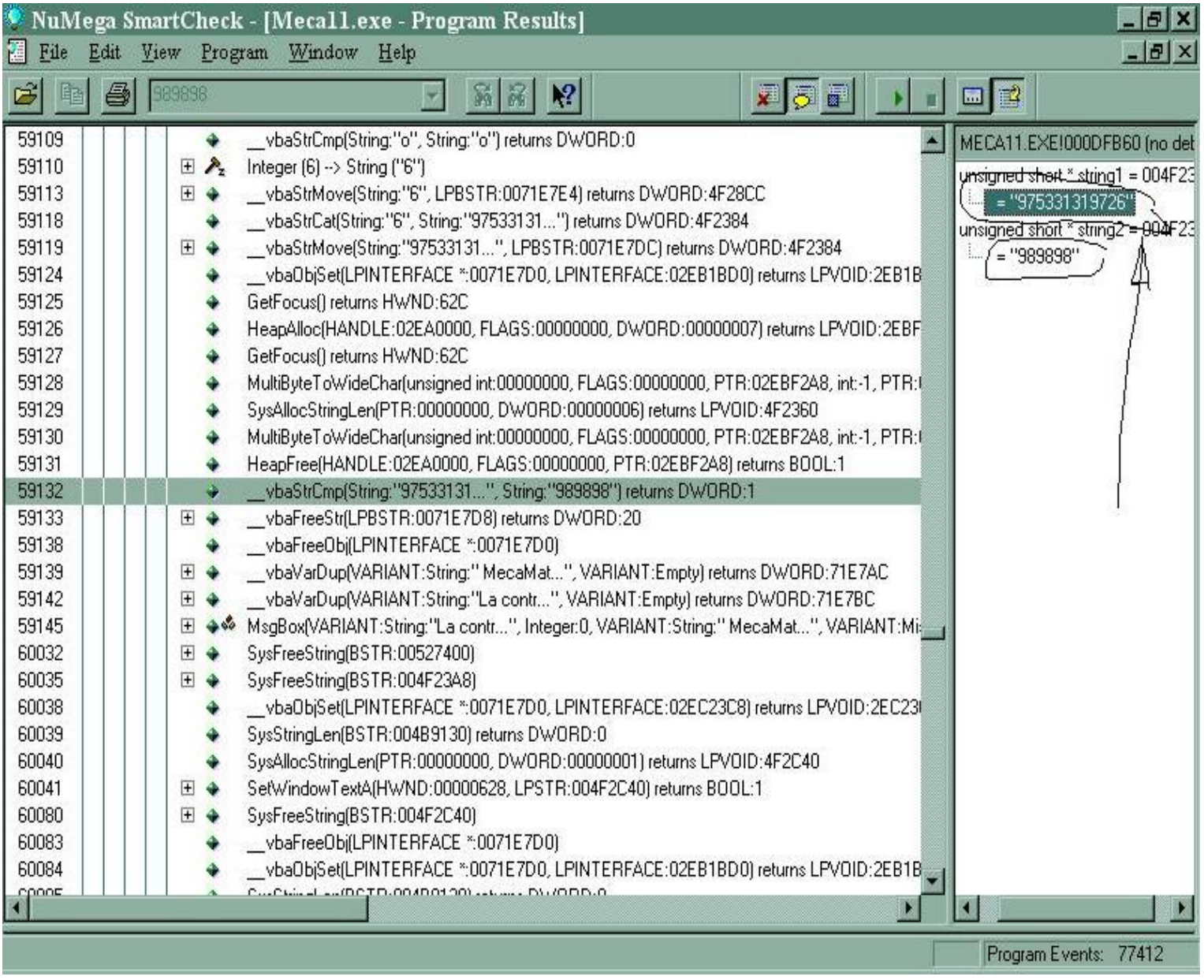
Pongo nombre: **Ricardo**

Numero de serie: **989898**

Y al aceptar nos sale el cartel de que la contraseña es incorrecta, buscando entre las STRINGS REFERENCES en el WDASM aparece la misma en **4dffcb** y proviene de un salto condicional en **4dfb9b**



Y el salto de donde proviene la REFERENCE by a condicional JUMP de **4dfb9b.**
Aquí se pueden tomar varios caminos, analizarlo con el Softice hasta encontrar la comparación o utilizar el Smart Check para ver si compara las claves directamente.
Probaremos el SMART CHECK lo abrimos cargamos el ejecutable MECA11.exe y ejecuta el programa, hay que tener bien configurado el Smart check como dice en la LECCIÓN sobre el mismo y si no se ve la ventana en la cual aparece la clave a la derecha, es porque esta cerrada sobre el margen derecho, poniendo el cursor allí y arrastrando hacia la izquierda se abrirá la ventana donde aparecerán las claves.
Una vez que arranca el programa vamos a la ventana de ayuda y ponemos de nuevo name **Ricardo** serial: **989898** y le damos a ACEPTAR y nos dice que la contraseña es incorrecta, vamos al Smart Check y detenemos la ejecución.
Vamos a FIND y colocamos **989898** y ponemos la marca en la primera línea ya que busca desde allí hacia abajo y en la segunda vez que para ya me aparece el serial correcto.



En la imagen anterior esta el momento de la comparación arriba esta el número de serie verdadero y debajo el falso que puse yo, copio la clave **975331319726** que corresponde a Ricardo y abro de nuevo el programa voy a ayuda y introduzco mi nombre: Ricardo y numero de serie: **975331319726** y al aceptar me dice que la clave es correcta que estoy registrado, eso si puede ser que sirva solo para mi maquina ese número de serie, quizás en otra máquina sea diferente pero la forma de hallarlo es la misma. Por si no se entienden las imágenes las mando adjuntas también.

Ricardo Narvaja