Registrando NeoBook 5.8.7	
Fecha	**25 de marzo de 2017**
Victima	**NeoBook 5.8.7**
URL de descarga	**www.neosoftware.com/software/nbw.exe**
MD5 del instalador	**028068C853BD47988EB75B19DC723BAB**
Protección	**Armadillo 9.64**
Herramientas	**Analizadores, ArmaGeddon 1.9, Ollydbg 1.10 de Neutrino, Import Rec 1.7c Final, Dede, un cerebro y mucha paciencia**
Objetivo	**Registrar la aplicación**
Dificultad	**Media/Alta**
Cracker	**Aguml**

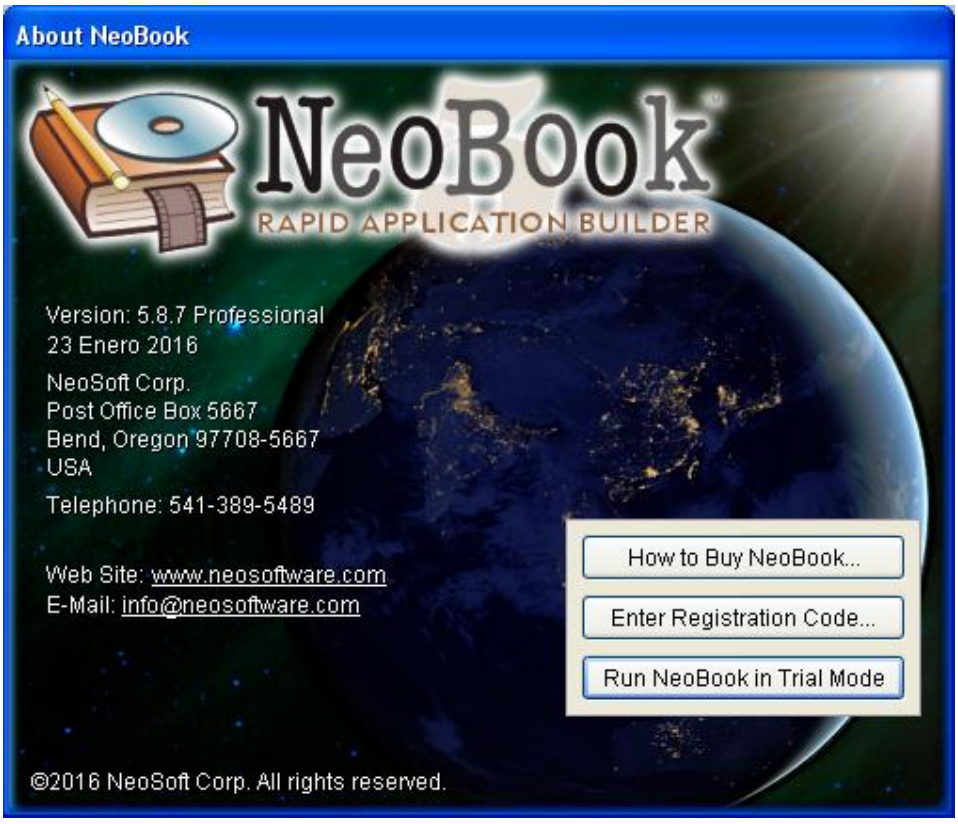
Indice

Introducción

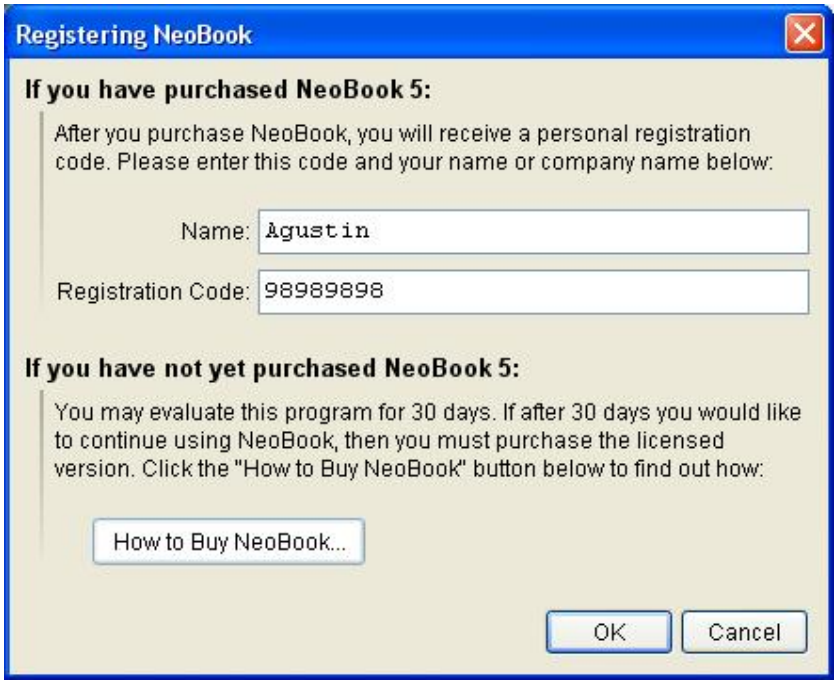
Lo primero decir que este programa me llegó de rebote sin saber ni siquiera para que servía y cuyo único objetivo era instalarlo para poder analizar un plugin de este. Después de conseguir resolver el plugin me dije “¿te atreves con un bicho así? No hay huevos” y en España el “no hay huevos” es algo que se toma como muy en serio así que me puse a ello jajaja.

Analizando a la víctima

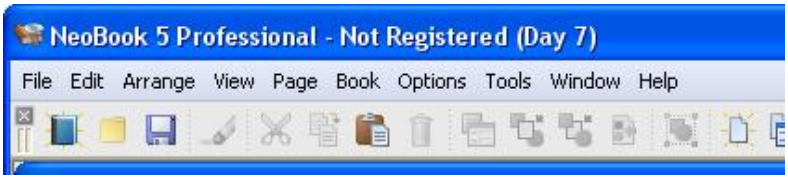
Lo primero será ejecutar el programa y ver que hace:



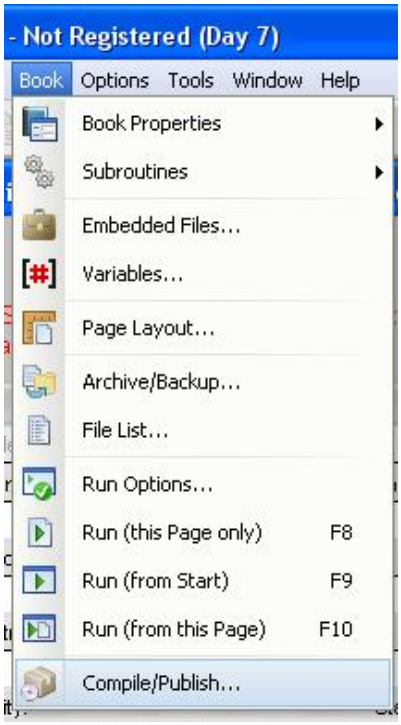
Una bonita ventana que nos indica que podemos registrarnos o ejecutarlo en modo trial. Damos a registrar:



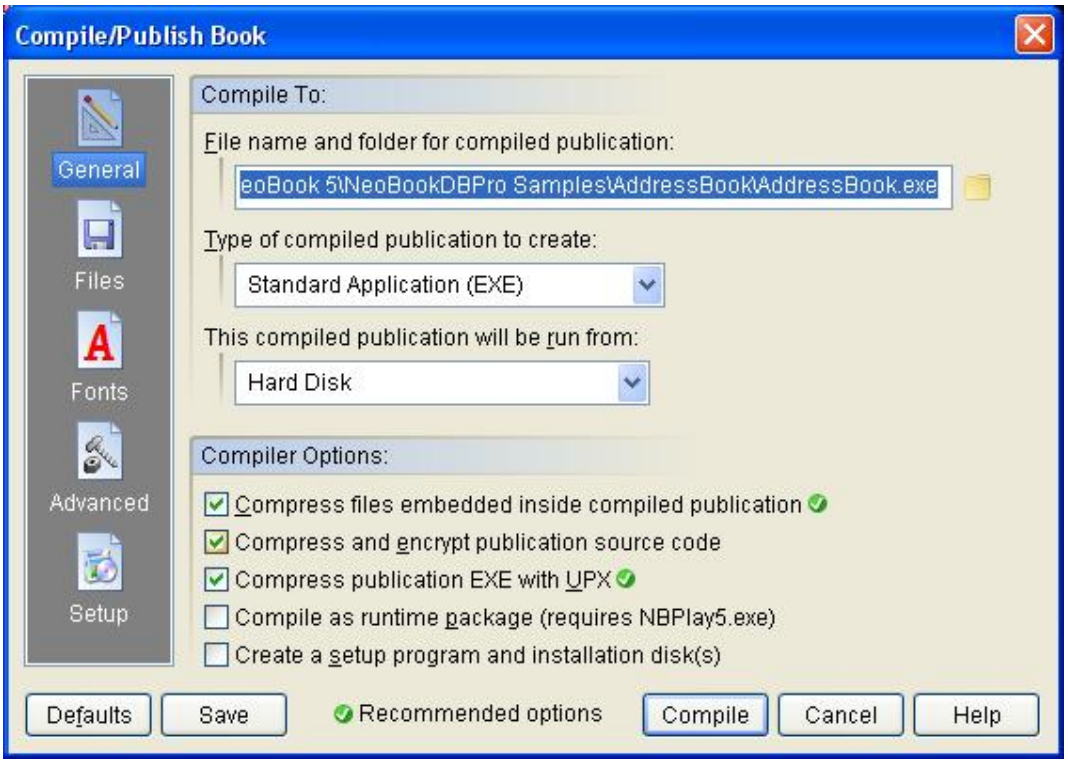
Vale pues ahora entraré en el modo trial a ver que veo:



El About es la misma ventana que para el registro.
Doy a compilar el proyecto:

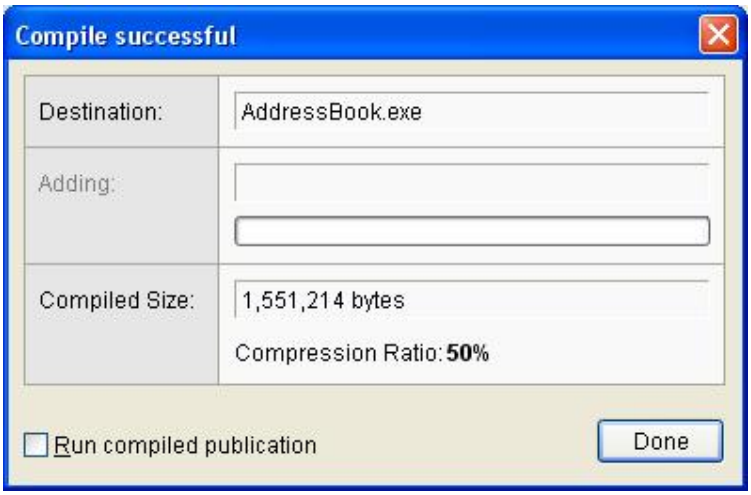


Y veo esto:

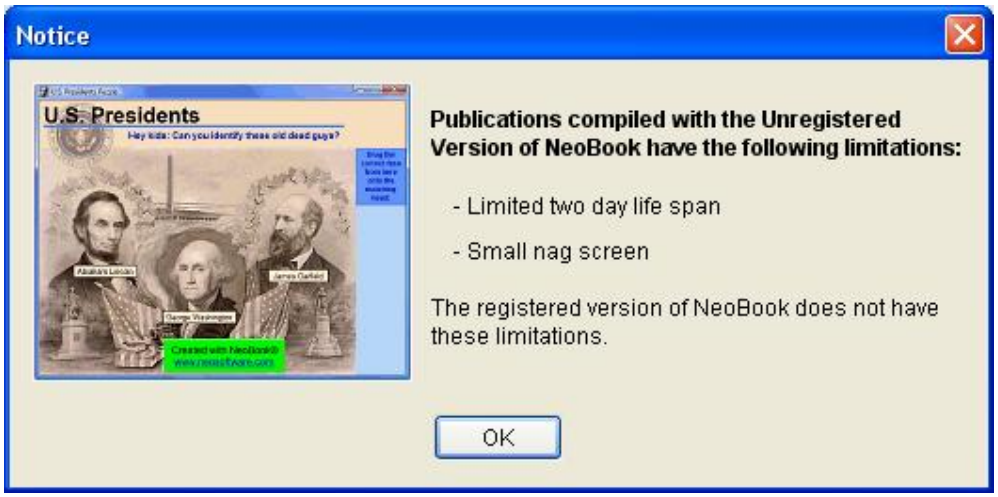


Me pongo a toquetear todas las opciones por si hay algo que se me escape y al destildar la opción “Compress and encrypt publication source code” me sale esto:

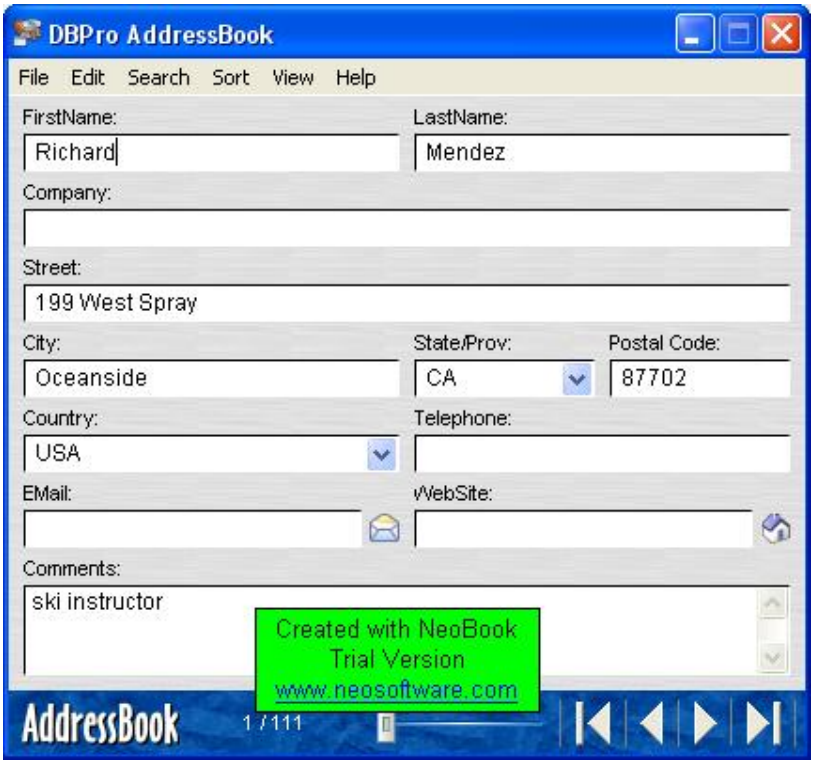
Pues nada, lo dejo tal cual y doy a compilar:



Parece que todo fue bien así que cierro la ventana y me encuentro con otra sorpresita:

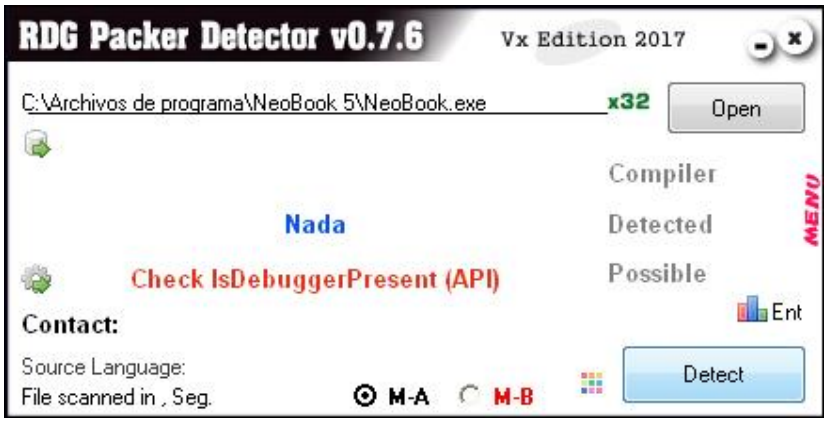


Si ejecuto el compilado me encuentro con esto:

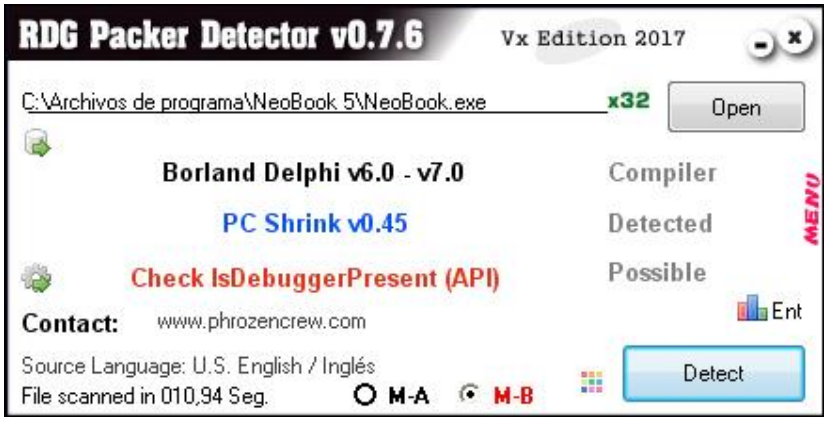


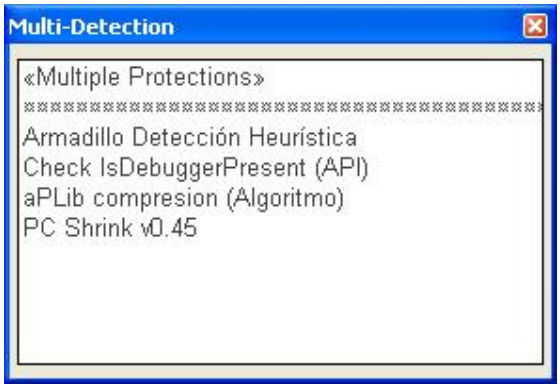
La nag que aparece es una ventana sin bordes ni barra de titulos por lo que no podemos cerrarla pero se cierra al cerrarse la aplicación compilada.

Lo siguiente es ver si tiene alguna protección comercial así que lo miro con algún identificador.
Empiezo con RDG Packer Detector:

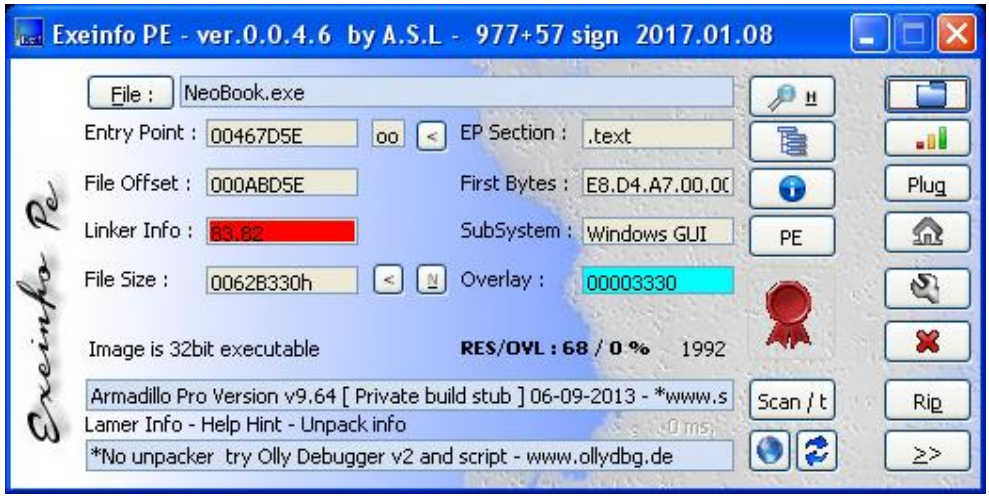


En el modo M-A no sabe ni que es asi que miro en el modo M-B:

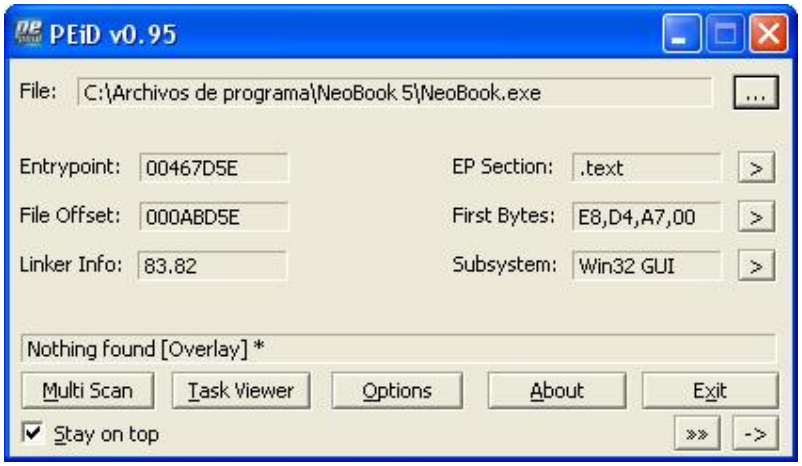




Sigo con Exeinfo PE:



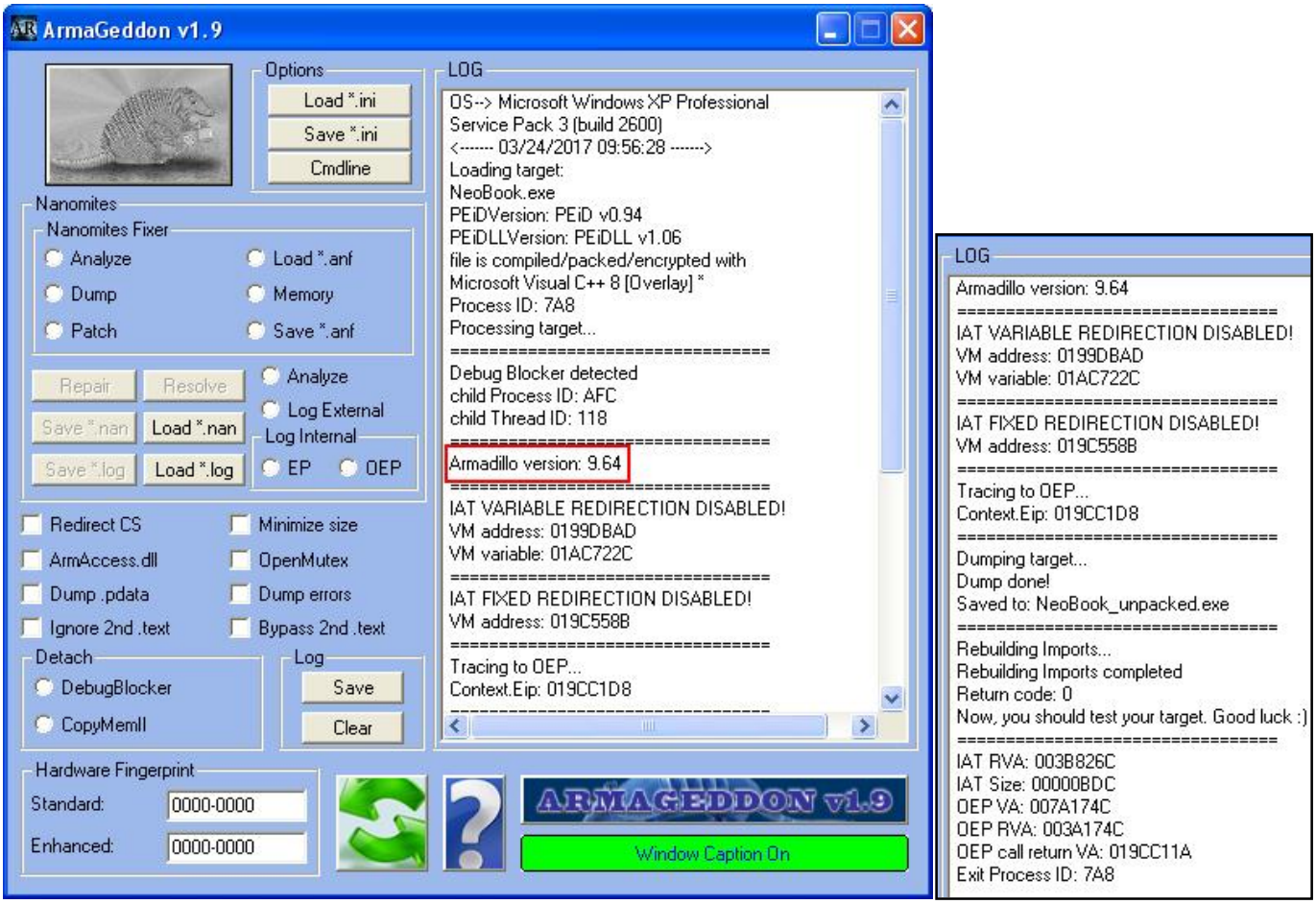
Y Peid:



Peid ya está bastante desactualizado y se ve que esta version de armadillo es muy nueva para el jejeje.

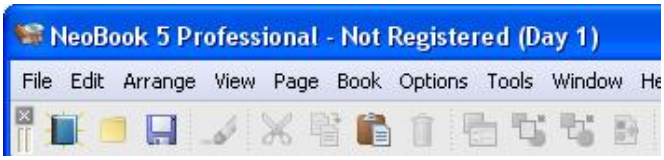
Desempacando a la víctima

Intentaré a ver que pasa con Armageddon 1.9 ya que el 2.2 no hay manera de que me funcione en mi máquina:



Parece ser que sí que es un Armadillo 9.64 y también parece ser que lo ha descomprimido sin problemas.
Ejecuto el desempacado y pruebo a ver qué ha pasado con las protecciones que trae.

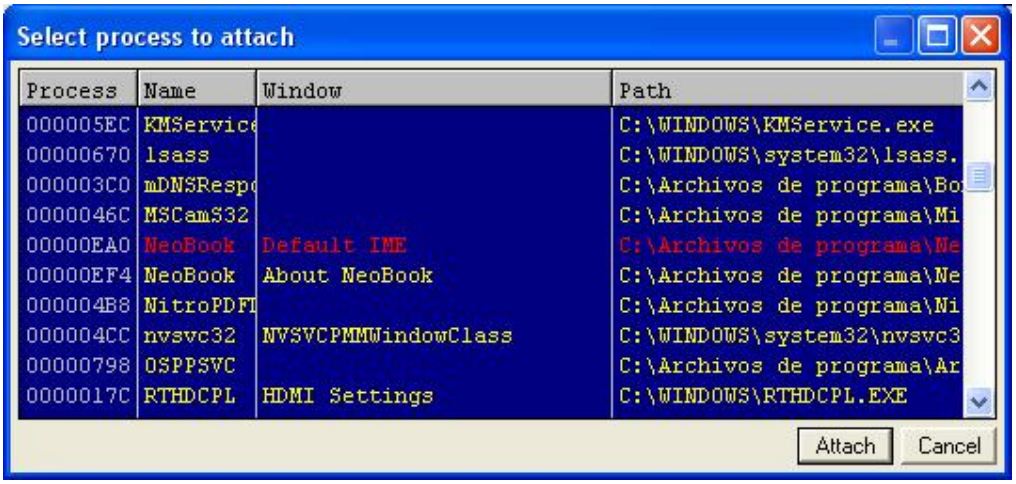
Sigue apareciendo la ventana del About al iniciarse pero si intento registrarme poniendo unos datos y dando a OK ya no aparece ningún mensaje de chico malo y no me deja registrarme. ¿Será que Armadillo era el encargado de esa tarea? Ni idea pero seguro que puedo registrarlo de otra forma.
El número de días transcurridos desde la instalación se ha reiniciado y ya no pasan los días:



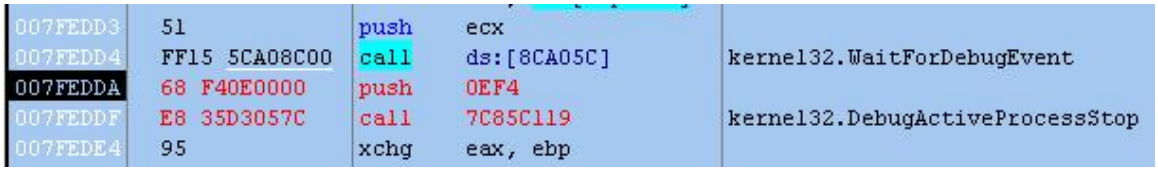
Si doy a compilar ya no aparece la ventana así que tampoco puedo compilar nada.
Parece ser que ArmaGeddon tampoco es que haya hecho un trabajo excelente y hay cosas que no funcionan bien.

Reparando el desempacado

Lo primero que hay que intentar es que muestre de nuevo la ventana de compilación para poder compilar de nuevo.
Abro el original en Olly debidamente protegido para que no me detecte y lo ejecuto. Una vez ejecutado pongo un BP en WaitForDebugEvent y parará ya que en realidad estamos depurando al padre y éste actúa como un debugger. Quito el BP, doy a Ctrl+F9 y a F7 y caigo justo debajo de la llamada a dicha función.
Ahora toca desatachear al padre del hijo y para ello doy a Attach en Olly, ordeno por nombre, busco el proceso y aparecen 2:

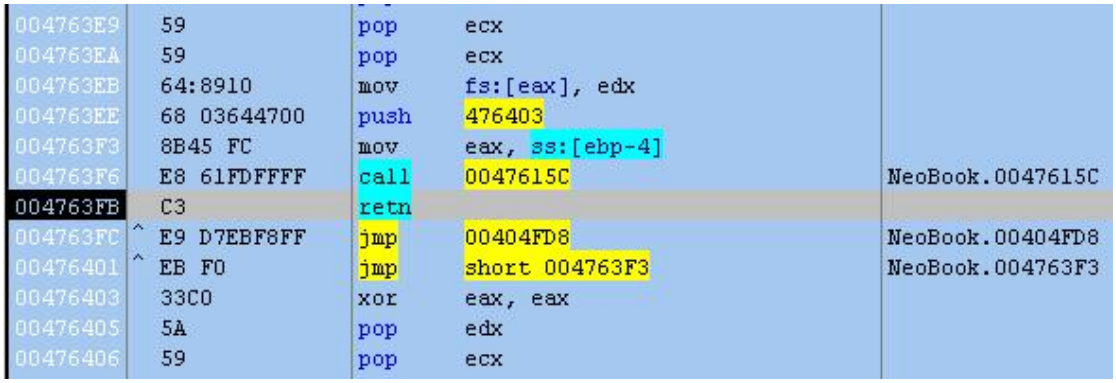


El que nos interesa es el que no está en rojo ya que el que está en rojo es justo el que ya estamos depurando y necesito depurar al hijo que es el de abajo.
Ya tengo el PID del hijo así que ya puedo cancelar en esa ventana y crear el injerto para desatachearlo:

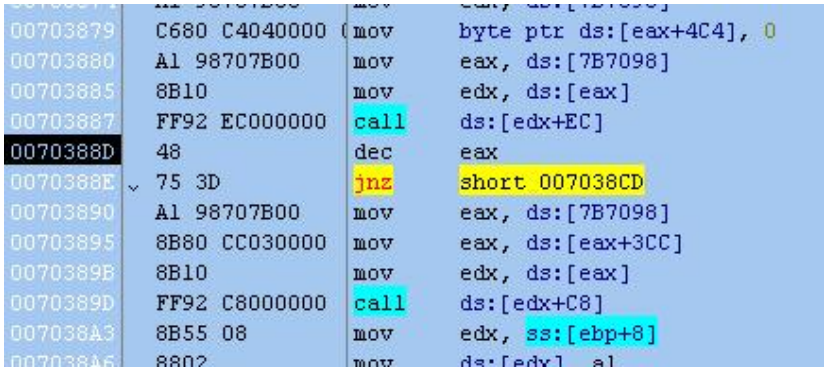


Ejecuto esas dos líneas con F8 y ya puedo atachearme al hijo sin problema y para ello abro otro Olly sin cerrar este y usando ese me atacheo al proceso 0xEF4. Ya solo trabajare con el último Olly que he abierto así que doy en éste a F9 y ya puedo trabajar perfectamente con él.

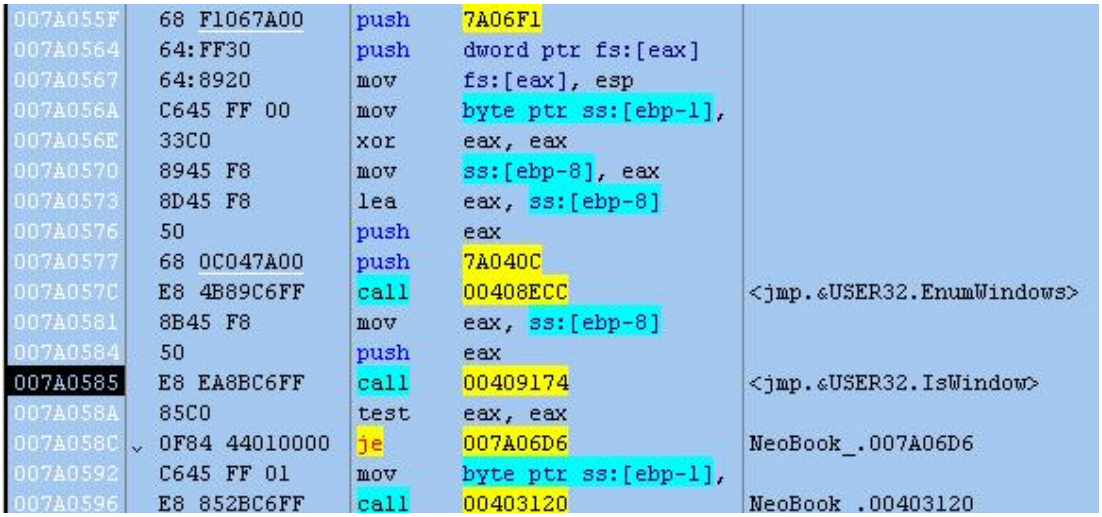
Hago que aparezca el formulario de compilación del programa, pauso Olly y voy saliendo de los RETN con F7 y Ctrl+F9 hasta que recupere el control del formulario nuevamente y entonces doy en el botón Cancel del formulario y caigo aquí:



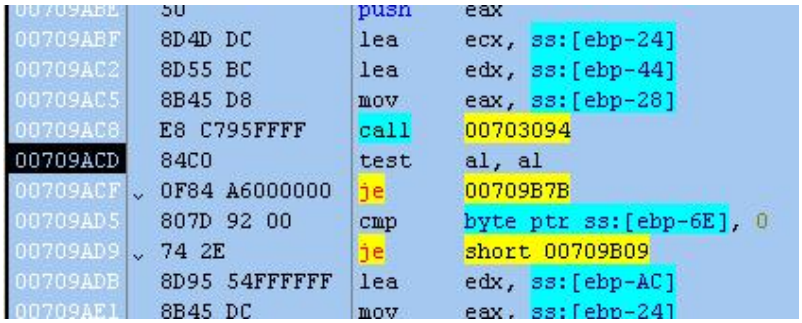
Voy con F7 y Ctrl+F9 hasta que llego aquí:



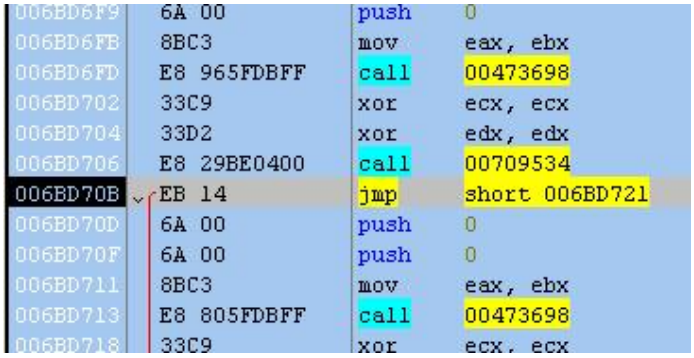
Ahora abro otro Olly y en el abro el desempacado y al ejecutarlo veo que se cierra con lo que seguramente está detectando que ya hay una instancia. Pensé en poner un BP en EnumProcesses y otro en FindWindowA pero la primera no me la reconoce y en la segunda no para.
Reinicio Olly y voy traceando hasta el segundo CALL que hay, entro y veo esto:



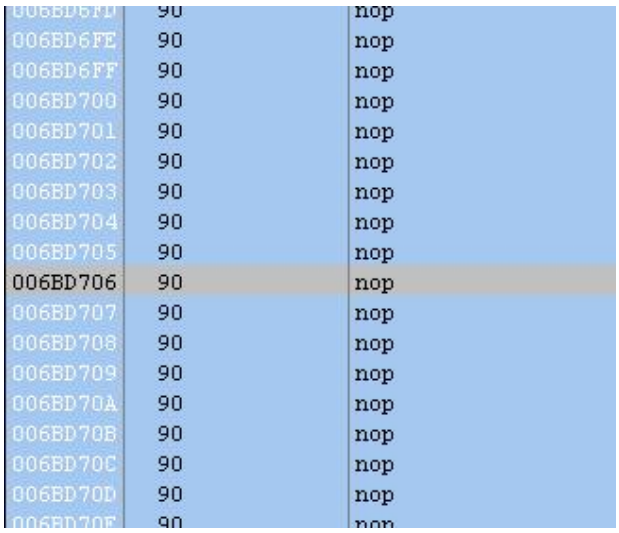
O sea que enumera las ventanas y busca la ventana del programa y justo detrás hay un salto condicional. Si paso el CALL veo que EAX vale 1 así que lo pongo a 0 o hago que el salto sea un JMP y doy a F9 y el programa ya arranca sin más problemas.
En el otro Olly donde tengo corriendo al hijo estoy parado justo debajo de la CALL de la línea 0x703887 así que pongo un BP en esa línea en este Olly e intento mostrar el formulario de compilación y para mi sorpresa no para.
Ahora sigo en el Olly que tiene al hijo con Ctrl+F9 y F7 hasta que llego aquí:



Ahora voy al Olly que tiene al desempacado y pongo un BP en el CALL de la línea 0x709AC8 y vuelvo a intentar mostrar la ventana de compilación pero sigue sin parar así que vuelvo a repetir el proceso en el hijo y ahora llego aquí:



Me dispongo a poner un BP en la línea 0x6BD706 del desempacado y me encuentro con esto:



Puto Armadillo, parece ser que o ha nopeado zonas o a lo mejor lo ha hecho ArmaGeddon, no sé.
Me voy al hijo y miro más arriba y veo esto:

```
006BD63A   64:8920          mov     fs:[eax], esp
006BD63D   90               nop
006BD63E   90               nop
006BD63F   90               nop
006BD640   90               nop
006BD641   90               nop
006BD642   8BC3             mov     eax, ebx
006BD644   E8 4F60DBFF      call    00473698                NeoBook.00473698
006BD649   8B15 9CFE7400    mov     edx, ds:[74FE9C]        NeoBook.0074FEE8
006BD64F   E8 7473D4FF      call    004049C8                NeoBook.004049C8
006BD654   84C0             test    al, al
006BD656   0F84 C5000000    je      006BD721                no salta
006BD65C   6A 00            push    0
006BD65E   6A 00            push    0
006BD660   68 50D76B00      push    6BD750                  ASCII "USERKEY"
006BD665   E8 3AAFD4FF      call    004085A4                NeoBook.004085A4
006BD66A   8BF0             mov     esi, eax
006BD66C   85F6             test    esi, esi
```

Y si voy para abajo veo esto:

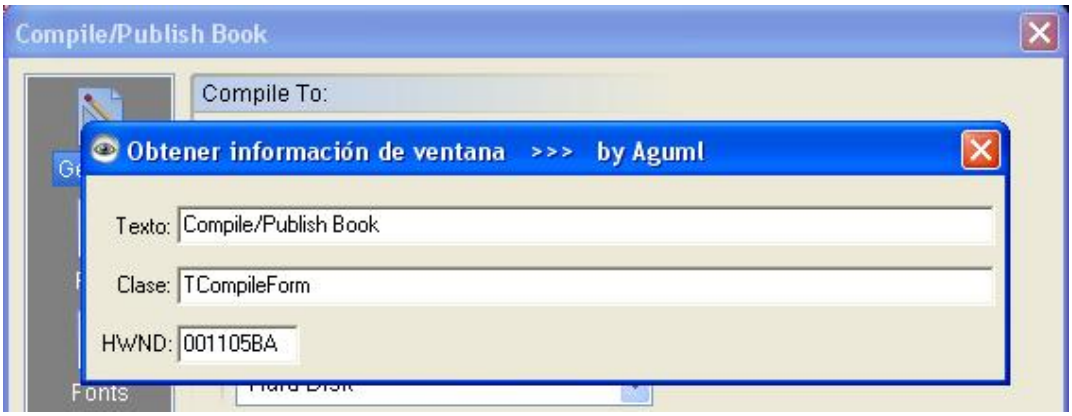
```
006BD704   33D2             xor     edx, edx
006BD706   E8 29BE0400      call    00709534
006BD70B   EB 14            jmp     short 006BD721
006BD70D   6A 00            push    0
006BD70F   6A 00            push    0
006BD711   8BC3             mov     eax, ebx
006BD713   E8 805FDBFF      call    00473698
006BD718   33C9             xor     ecx, ecx
006BD71A   33D2             xor     edx, edx
006BD71C   E8 13BE0400      call    00709534
006BD721   90               nop
006BD722   90               nop
006BD723   90               nop
006BD724   90               nop
006BD725   90               nop
006BD726   33C0             xor     eax, eax
006BD728   5A               pop     edx
006BD729   59               pop     ecx
```

Puedo ver que esa zona de código tiene por arriba una zona de NOP´s y por debajo igual y es justo el tamaño que tengo en el desempacado lleno de NOP´s así que hago un Binary Copy de todo el bloque desde 0x6BD642 hasta 0x6BD71C del hijo y hago un Binary Paste en el desempacado justo en el mismo bloque:

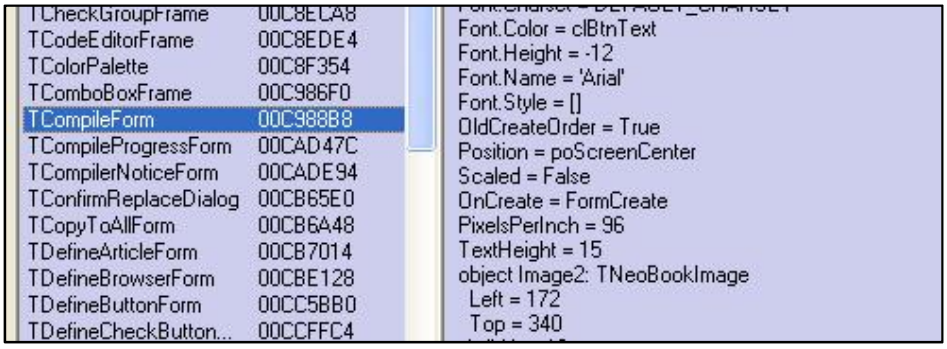
```
006BD6FD   E8 965FDBFF      call    00473698
006BD702   33C9             xor     ecx, ecx
006BD704   33D2             xor     edx, edx
006BD706   E8 29BE0400      call    00709534
006BD70B   EB 14            jmp     short 006BD721
006BD70D   6A 00            push    0
006BD70F   6A 00            push    0
006BD711   8BC3             mov     eax, ebx
006BD713   E8 805FDBFF      call    00473698
006BD718   33C9             xor     ecx, ecx
006BD71A   33D2             xor     edx, edx
006BD71C   E8 13BE0400      call    00709534
006BD721   90               nop
006BD722   90               nop
006BD723   90               nop
006BD724   90               nop
006BD725   90               nop
006BD726   33C0             xor     eax, eax
```

Vuelvo a intentar mostrar la ventana de compilación y ahora para en los dos BP´s que puse antes pero los voy quitando y dando a F9 y por fin muestra la ventana. Intento compilar dando al botón y me muestra esto:

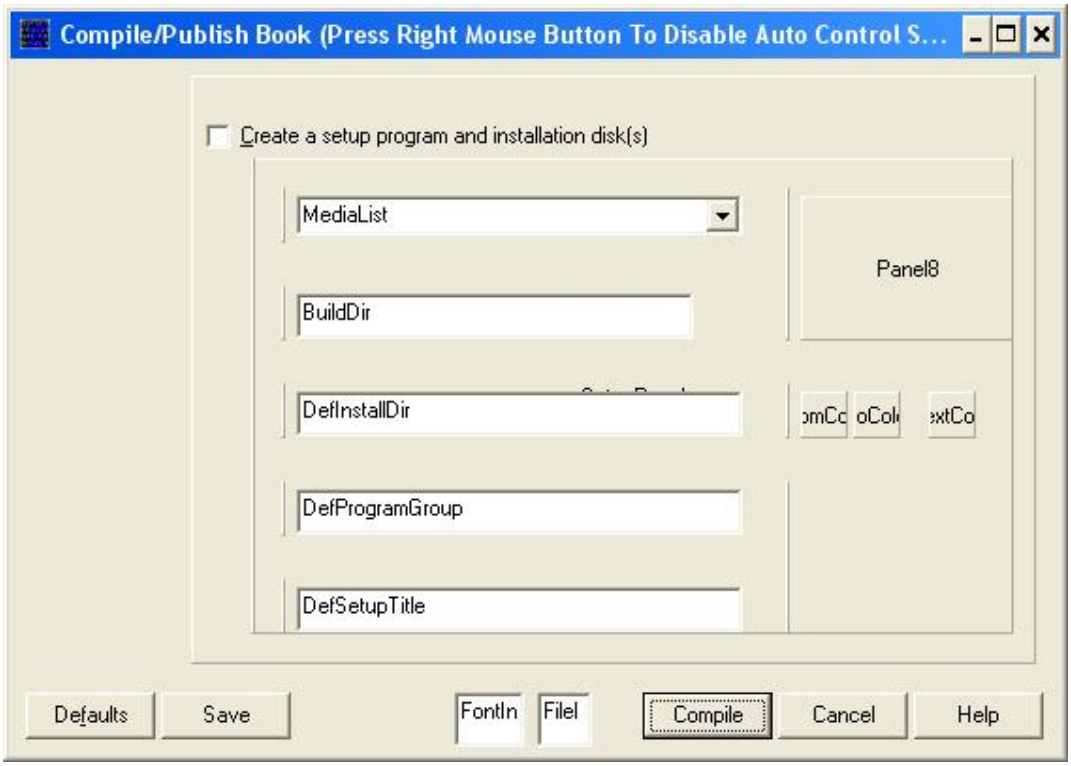
Como RDG nos dice que es un Delphi lo abro con Dede y lo analizo y luego intento obtener el nombre del formulario para encontrarlo más fácil:



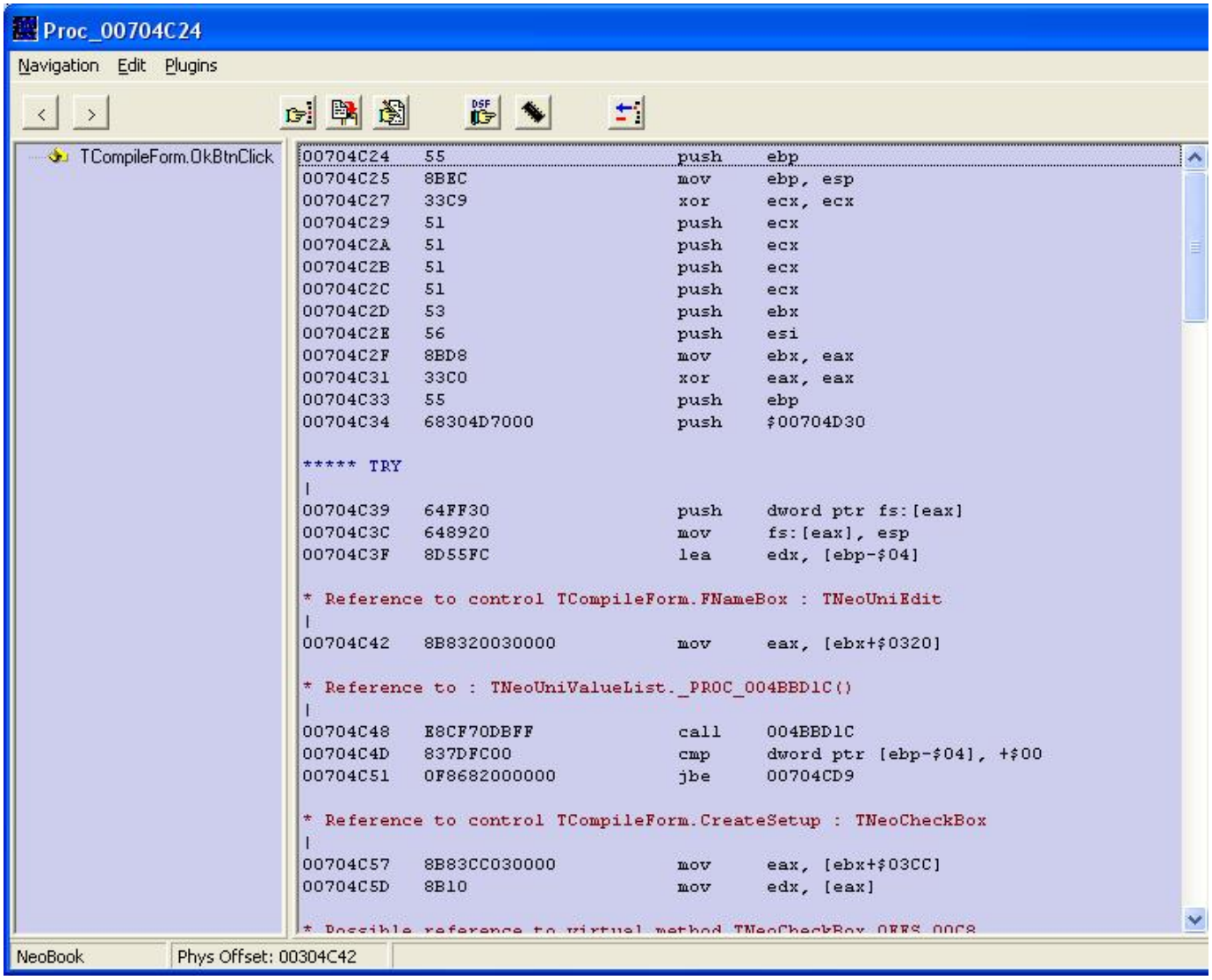
Sé que hay aplicaciones que muestran esa información pero si no uso yo la que creé ¿Quién lo va a hacer? Jajaja. Busco ese formulario en Dede:



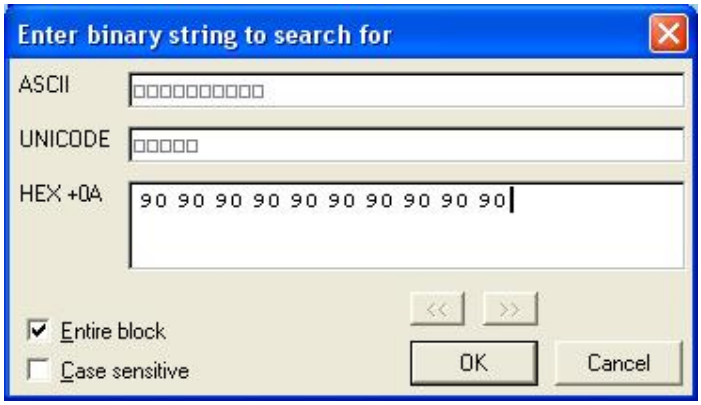
Hago doble clic en él y me lo muestra:



Ahora doy doble clic en el botón “Compile” y me muestra los eventos para ese botón:



Solo está el evento OnClic y veo que la primera línea que ejecuta es en 0x704C24 así que pongo un BP en esa línea y vuelvo a intentar compilar. Una vez para en el BP sigo traceando pero no doy con nada interesante y acabo saliendo de la función. Se me ocurrió que me puede estar jodiendo con algún otro bloque de NOP´s así que me voy al inicio en la ventana CPU del Olly del desempacado y busco una cadena binaria de varios NOP´s:



Me voy al último NOP y pongo un HBP on Execution en el y vuelvo a intentar compilar pero no para así que repito el proceso con el siguiente bloque de NOP´s que encuentra y tampoco para y vuelvo a repetir el proceso pero a la tercera va la vencida porque ahora sí que para:

```
00660E48   90            nop
00660E49   90            nop
00660E4A   90            nop
00660E4B   90            nop
00660E4C   90            nop
00660E4D   90            nop
00660E4E   90            nop
00660E4F   90            nop
00660E50   90            nop
00660E51   8B45 F8       mov     eax, ss:[ebp-8]
00660E54   5F            pop     edi
00660E55   5E            pop     esi
00660E56   5B            pop     ebx
00660E57   8BE5          mov     esp, ebp
00660E59   5D            pop     ebp
00660E5A   C3            retn
00660E5B   00FF          add     bh, bh
```

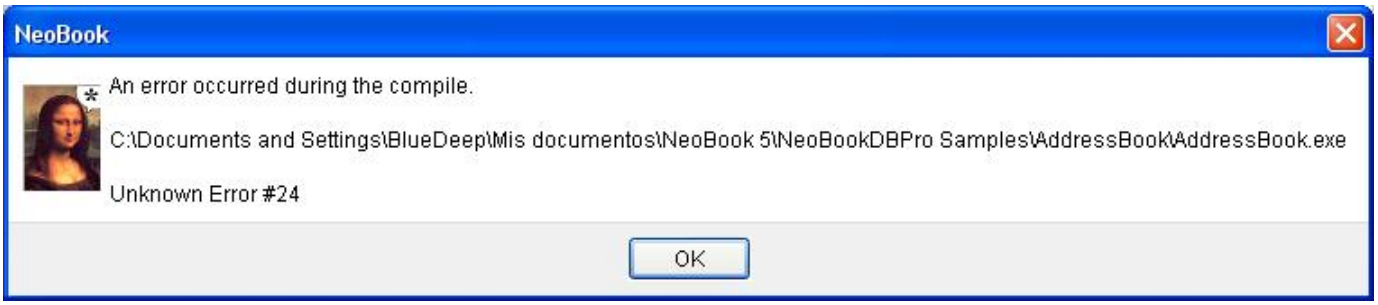
Me voy a esa dirección en el hijo y veo esto:

```
00660E35   8B45 FC       mov     eax, ss:[ebp-4]
00660E38   8B53 04       mov     edx, ds:[ebx+4]
00660E3B   E8 7449DAFF   call    004057B4
00660E40   C745 F8 9A3A00( mov     dword ptr ss:[ebp-8], 3A9A
00660E47   E8 0443DAFF   call    00405150
00660E4C   90            nop
00660E4D   90            nop
00660E4E   90            nop
00660E4F   90            nop
00660E50   90            nop
00660E51   8B45 F8       mov     eax, ss:[ebp-8]
00660E54   5F            pop     edi
00660E55   5E            pop     esi
00660E56   5B            pop     ebx
00660E57   8BE5          mov     esp, ebp
00660E59   5D            pop     ebp
00660E5A   C3            retn
```

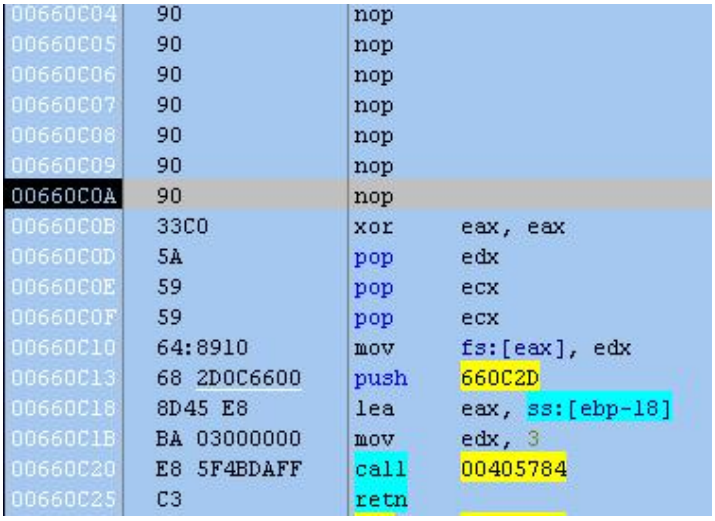
Otro bloque de código que falta en el desempacado así que, como hice en el otro bloque, hago un Binary Copy del bloque del hijo y hago un Binary Paste en el desempacado:

```
00660E1E  ⌄ 74 15          je      short 00660E35          NeoBook_.00660E35
00660E20    8BC3           mov     eax, ebx
00660E22    8B15 2C9E4000  mov     edx, ds:[409E2C]        NeoBook_.00409E78
00660E28    E8 BF3BDAFF    call    004049EC                NeoBook_.004049EC
00660E2D    8B40 0C        mov     eax, ds:[eax+C]
00660E30    8945 F8        mov     ss:[ebp-8], eax
00660E33  ⌄ EB 12          jmp     short 00660E47          NeoBook_.00660E47
00660E35    8B45 FC        mov     eax, ss:[ebp-4]
00660E38    8B53 04        mov     edx, ds:[ebx+4]
00660E3B    E8 7449DAFF    call    004057B4                NeoBook_.004057B4
00660E40    C745 F8 9A3A00( mov     dword ptr ss:[ebp-8],
00660E47    E8 0443DAFF    call    00405150                NeoBook_.00405150
00660E4C    90             nop
00660E4D    90             nop
00660E4E    90             nop
00660E4F    90             nop
00660E50    90             nop
00660E51    8B45 F8        mov     eax, ss:[ebp-8]
```

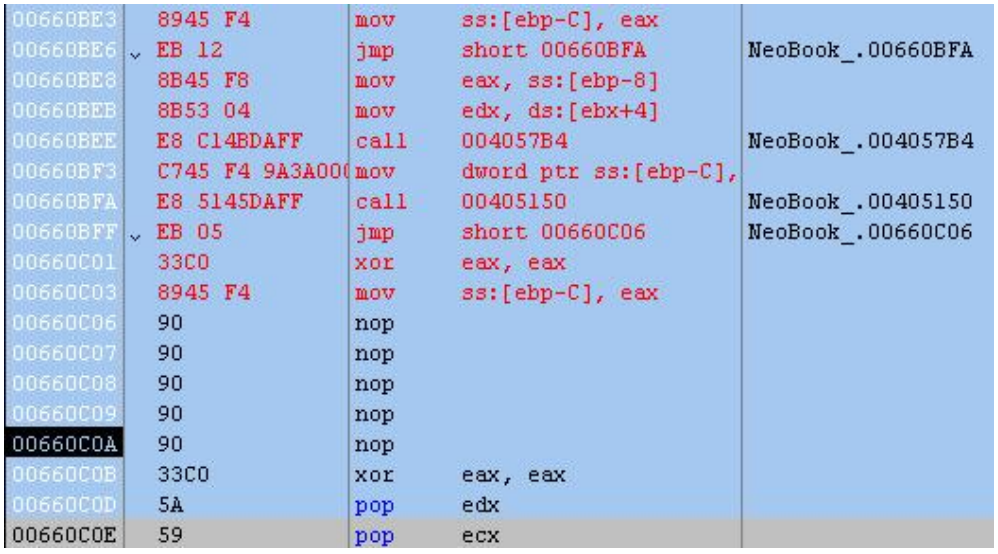
Vuelvo a intentar compilar y ahora me muestra esto:



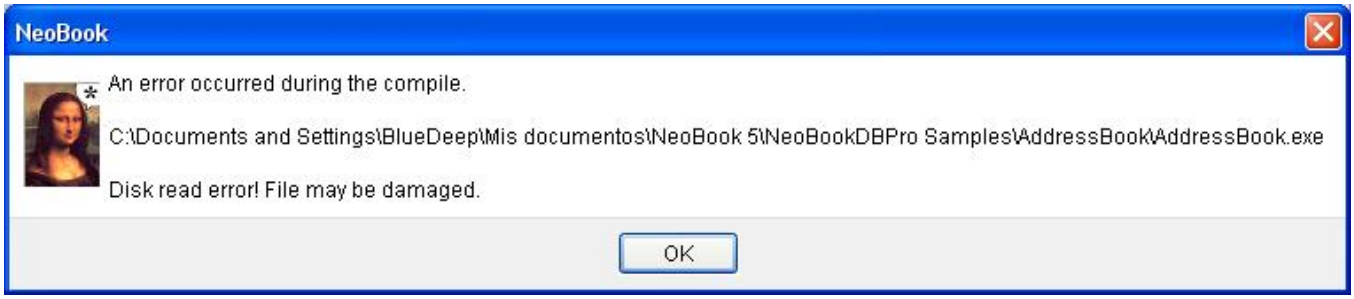
Me voy otra vez al inicio de la ventana CPU del desempacado y vuelvo a repetir el proceso de buscar NOP´s y poner HBP on Execution en el último NOP y para en el segundo bloque que pruebo:



Así que me voy al hijo y repito el proceso para reparar ese bloque en el desempacado:



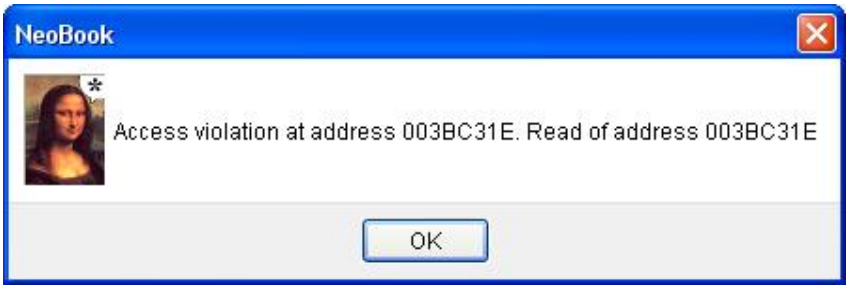
Vuelvo a intentar compilar y muestra otro error diferente:



Vuelvo a repetir el proceso de los bloques de NOP´s y esta vez para en el primero así que lo reparo también:

```
006608ED   33C0           xor     eax, eax
006608EF   5A             pop     edx
006608F0   59             pop     ecx
006608F1   59             pop     ecx
006608F2   64:8910        mov     fs:[eax], edx
006608F5   68 0A096600    push    66090A
006608FA   8B45 F4        mov     eax, ss:[ebp-C]
006608FD   E8 2E3FDAFF    call    00404830          NeoBook_.00404830
00660902   C3             retn
00660903 ^ E9 D046DAFF    jmp     00404FD8          NeoBook_.00404FD8
00660908 ^ EB F0          jmp     short 006608FA    NeoBook_.006608FA
0066090A   90             nop
0066090B   90             nop
0066090C   90             nop
0066090D   90             nop
0066090E   90             nop
0066090F   8B45 F8        mov     eax, ss:[ebp-8]
00660912   5F             pop     edi
```

Vuelvo a intentar compilar y ahora me da una excepción el Olly y si la paso me muestra esto el programa:



Acepto y comienza el proceso de compilación y luego muestra esto:

Vamos a ver que produce el error del primero y para ello vuelvo a intentar compilar con el HBP on Execution en la última zona reparada y cuando pare sigo con F8 hasta que de la excepción y llego aquí:

```
00709E8B   8B45 D8        mov     eax, ss:[ebp-28]
00709E8E   8D90 F4050000  lea     edx, ds:[eax+5F4]
00709E94   33C9           xor     ecx, ecx
00709E96   8B45 DC        mov     eax, ss:[ebp-24]
00709E99   E8 4AB8FFFF    call    007056E8                NeoBook_.007056E8
00709E9E   837D 98 00     cmp     dword ptr ss:[ebp-68]
00709EA2 v 0F85 A1000000  jnz     00709F49                NeoBook_.00709F49
00709EA8   E8 FB61F5FF    call    006600A8                NeoBook_.006600A8
00709EAD   84C0           test    al, al
00709EAF v 0F85 94000000  jnz     00709F49                NeoBook_.00709F49
00709EB5   8B45 D8        mov     eax, ss:[ebp-28]
00709EB8   80B8 FF040000  cmp     byte ptr ds:[eax+4FF]
00709EBF v 0F84 84000000  je      00709F49                NeoBook_.00709F49
00709EC5   8B1D 100C7B00  mov     ebx, ds:[7B0C10]        NeoBook_.007A31D0
00709ECB   8B1B           mov     ebx, ds:[ebx]
```

Pongo un BP en esa línea y vuelvo a intentar compilar y cuando pare entro y sigo traceando con F8 hasta que me vuelva a dar la excepción y ahora es aquí:

```
00705A5E   FF45 EC        inc     dword ptr ss:[ebp-14]
00705A61   FF4D D8        dec     dword ptr ss:[ebp-28]
00705A64 ^ 0F85 20FDFFFF  jnz     0070578A                NeoBook_.0070578A
00705A6A   8B55 F8        mov     edx, ss:[ebp-8]
00705A6D   8B45 F0        mov     eax, ss:[ebp-10]
00705A70   8B08           mov     ecx, ds:[eax]
00705A72   FF51 20        call    ds:[ecx+20]             NeoBook_.006FFB2C
00705A75   33C0           xor     eax, eax
00705A77   5A             pop     edx
00705A78   59             pop     ecx
00705A79   59             pop     ecx
00705A7A   64:8910        mov     fs:[eax], edx
00705A7D   68 925A7000    push    705A92
00705A82   8B45 F0        mov     eax, ss:[ebp-10]
00705A85   E8 A6EDCFFF    call    00404830                NeoBook_.00404830
00705A8A   C3             retn
00705A8B ^ E9 48F5CFFF    jmp     00404FD8                NeoBook_.00404FD8
00705A90 ^ EB F0          jmp     short 00705A82          NeoBook_.00705A82
```

Quito el BP anterior y lo pongo en esta línea y vuelvo a intentar compilar y cuando pare entro con F7 y repito el proceso de tracear con F8 y ahora es aquí:

```
006FFB49   33C0           xor     eax, eax
006FFB4B   55             push    ebp
006FFB4C   68 77FB6F00    push    6FFB77
006FFB51   64:FF30        push    dword ptr fs:[eax]
006FFB54   64:8920        mov     fs:[eax], esp
006FFB57   8B55 FC        mov     edx, ss:[ebp-4]
006FFB5A   8BC6           mov     eax, esi
006FFB5C   8B08           mov     ecx, ds:[eax]
006FFB5E   FF51 18        call    ds:[ecx+18]             NeoBook_.006FFB84
006FFB61   33C0           xor     eax, eax
006FFB63   5A             pop     edx
```

Quito el BP anterior y lo pongo en esa línea y vuelvo a intentarlo entrando con F7 cuando pare y traceando con F8 y ahora es aquí:

```
006FFDC7   8B10          mov     edx, ds:[eax]
006FFDC9   FF12          call    ds:[edx]
006FFDCB   50            push    eax
006FFDCC   8B45 E4       mov     eax, ss:[ebp-1C]
006FFDCF   8B40 04       mov     eax, ds:[eax+4]
006FFDD2   50            push    eax
006FFDD3   E8 B8F3FFFF   call    006FF190                NeoBook_.006FF190
006FFDD8   8945 D8       mov     ss:[ebp-28], eax
006FFDDB   837D D8 00    cmp     dword ptr ss:[ebp-28]
006FFDDF   74 1D         je      short 006FFDFE          NeoBook_.006FFDFE
006FFDE1   33C9          xor     ecx, ecx
```

Repito el proceso anterior y ahora es aquí:

```
006FF187   C3            retn
006FF188   832D 80707B00 sub     dword ptr ds:[7B7080]
006FF18F   C3            retn
006FF190 - FF25 8C8E7B00 jmp     ds:[7B8E8C]
006FF196   8BC0          mov     eax, eax
006FF198   55            push    ebp
006FF199   8BEC          mov     ebp, esp

ds:[007B8E8C]=003BC31E
```

Es justo donde da el error y si pongo un BP en esa dirección en el hijo veo esto:

```
006FF18F   C3            retn
006FF190 - FF25 8C8E7B00 jmp     ds:[7B8E8C]             IMAGEHLP.CheckSumMappedFile
006FF196   8BC0          mov     eax, eax
```

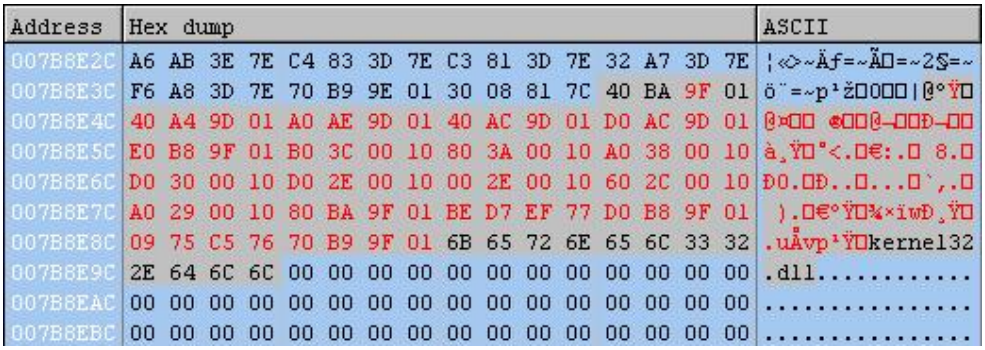
Si miro la dirección que apunta el salto veo esto en el hijo esto:

```
Address   Hex dump                                        ASCII
007B8E2C  A6 AB 3E 7E C4 83 3D 7E C3 81 3D 7E 32 A7 3D 7E  ¦«>~Äƒ=~Ã□=~2§=~
007B8E3C  F6 A8 3D 7E 70 B9 9F 01 30 08 81 7C 40 BA 9F 01  ö¨=~p¹Ÿ□0□□|@°Ÿ□
007B8E4C  40 A4 9D 01 A0 AE 9D 01 40 AC 9D 01 D0 AC 9D 01  @¤□□ ®□□@¬□□Ð¬□□
007B8E5C  E0 B8 9F 01 B0 3C 00 10 80 3A 00 10 A0 38 00 10  à¸Ÿ□°<.□€:.□ 8.□
007B8E6C  D0 30 00 10 D0 2E 00 10 00 2E 00 10 60 2C 00 10  Ð0.□Ð..□...□`,.□
007B8E7C  A0 29 00 10 80 BA 9F 01 BE D7 EF 77 D0 B8 9F 01   ).□€°Ÿ□¾×iwÐ¸Ÿ□
007B8E8C  09 75 C5 76 70 B9 9F 01 6B 65 72 6E 65 6C 33 32  .uÅvp¹Ÿ□kernel32
007B8E9C  2E 64 6C 6C 00 00 00 00 00 00 00 00 00 00 00 00  .dll............
007B8EAC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
007B8EBC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Y en el desempacado esto:

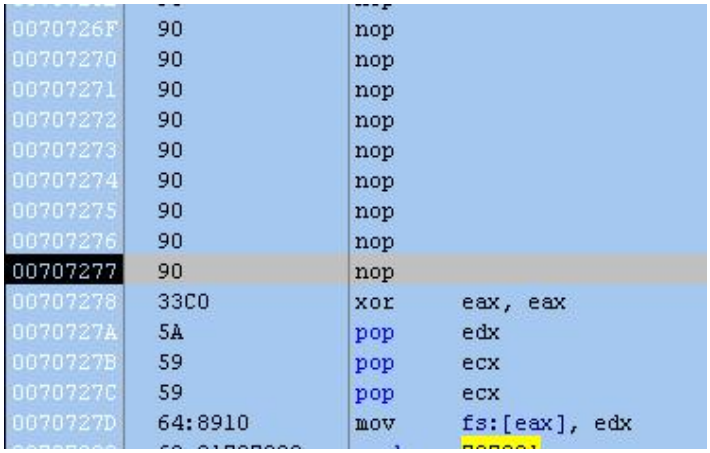
```
Address   Hex dump                                        ASCII
007B8E2C  A6 AB 3E 7E C4 83 3D 7E C3 81 3D 7E 32 A7 3D 7E  ¦«>~Äƒ=~Ã□=~2§=~
007B8E3C  F6 A8 3D 7E 70 B9 9E 01 30 08 81 7C 40 BA 9E 01  ö¨=~p¹ž□0□□|@°ž□
007B8E4C  1E C2 3B 00 32 C2 3B 00 42 C2 3B 00 50 C2 3B 00  □Â;.2Â;.BÂ;.PÂ;.
007B8E5C  00 00 00 00 6A C2 3B 00 7A C2 3B 00 8E C2 3B 00  ....jÂ;.zÂ;.ŽÂ;.
007B8E6C  9E C2 3B 00 B0 C2 3B 00 C4 C2 3B 00 D4 C2 3B 00  žÂ;.°Â;.ÄÂ;.ÔÂ;.
007B8E7C  E6 C2 3B 00 00 00 00 00 00 C3 3B 00 00 00 00 00  æÂ;......Ã;.....
007B8E8C  1E C3 3B 00 00 00 00 00 6B 65 72 6E 65 6C 33 32  □Ã;.....kernel32
007B8E9C  2E 64 6C 6C 00 00 00 00 00 00 00 00 00 00 00 00  .dll............
007B8EAC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Hay varias llamadas a apis en el hijo y en el desempacado hay lo que parecen direcciones propias que usaría Armadillo para recalcular esos valores. Hago un Binary Copy del bloque del hijo y luego hago un Binary Paste en el desempacado para reparar esa zona:



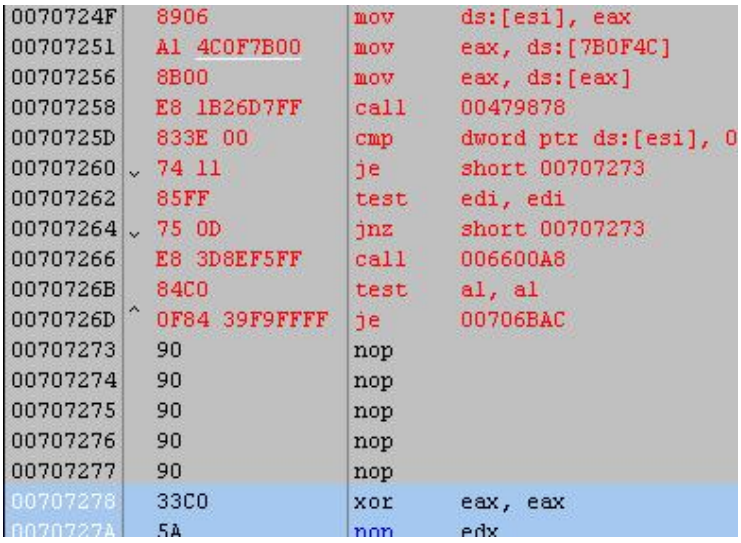
Address	Hex dump	ASCII
007B8E2C	A6 AB 3E 7E C4 83 3D 7E C3 81 3D 7E 32 A7 3D 7E	¦«>~Äƒ=~Ã□=~2§=~
007B8E3C	F6 A8 3D 7E 70 B9 9E 01 30 08 81 7C 40 BA 9F 01	ö¨=~p¹ž□0□□|@°Ÿ□
007B8E4C	40 A4 9D 01 A0 AE 9D 01 40 AC 9D 01 D0 AC 9D 01	@¤□□ ®□□@¬□□Ð¬□□
007B8E5C	E0 B8 9F 01 B0 3C 00 10 80 3A 00 10 A0 38 00 10	à¸Ÿ□°<.□€:.□ 8.□
007B8E6C	D0 30 00 10 D0 2E 00 10 00 2E 00 10 60 2C 00 10	Ð0.□Ð..□...□`,.□
007B8E7C	A0 29 00 10 80 BA 9F 01 BE D7 EF 77 D0 B8 9F 01	).□€°Ÿ□¾×iwÐ¸Ÿ□
007B8E8C	09 75 C5 76 70 B9 9F 01 6B 65 72 6E 65 6C 33 32	.uÅvp¹Ÿ□kernel32
007B8E9C	2E 64 6C 6C 00 00 00 00 00 00 00 00 00 00 00 00	.dll............
007B8EAC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
007B8EBC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....

Vuelvo a intentar compilar y ya no muestra el primer mensaje de error pero si el segundo así que vuelvo a intentar con el proceso de poner HBP´s en los bloques de NOP´s y después de varios intentos para en este:



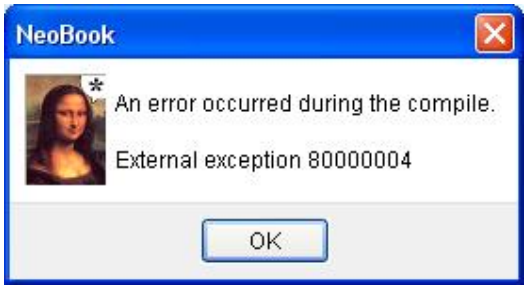
```
0070726F   90              nop
00707270   90              nop
00707271   90              nop
00707272   90              nop
00707273   90              nop
00707274   90              nop
00707275   90              nop
00707276   90              nop
00707277   90              nop
00707278   33C0            xor     eax, eax
0070727A   5A              pop     edx
0070727B   59              pop     ecx
0070727C   59              pop     ecx
0070727D   64:8910         mov     fs:[eax], edx
```

Así que lo reparo también:

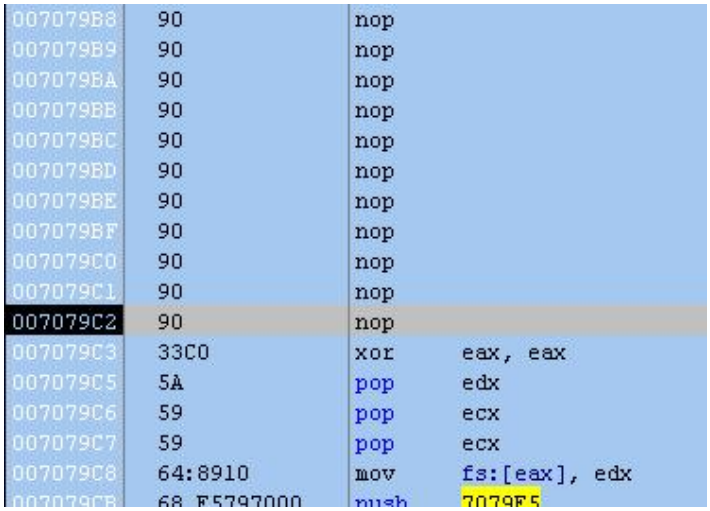


```
0070724F   8906            mov     ds:[esi], eax
00707251   A1 4C0F7B00     mov     eax, ds:[7B0F4C]
00707256   8B00            mov     eax, ds:[eax]
00707258   E8 1B26D7FF     call    00479878
0070725D   833E 00         cmp     dword ptr ds:[esi], 0
00707260 v 74 11           je      short 00707273
00707262   85FF            test    edi, edi
00707264 v 75 0D           jnz     short 00707273
00707266   E8 3D8EF5FF     call    006600A8
0070726B   84C0            test    al, al
0070726D ^ 0F84 39F9FFFF   je      00706BAC
00707273   90              nop
00707274   90              nop
00707275   90              nop
00707276   90              nop
00707277   90              nop
00707278   33C0            xor     eax, eax
0070727A   5A              pop     edx
```

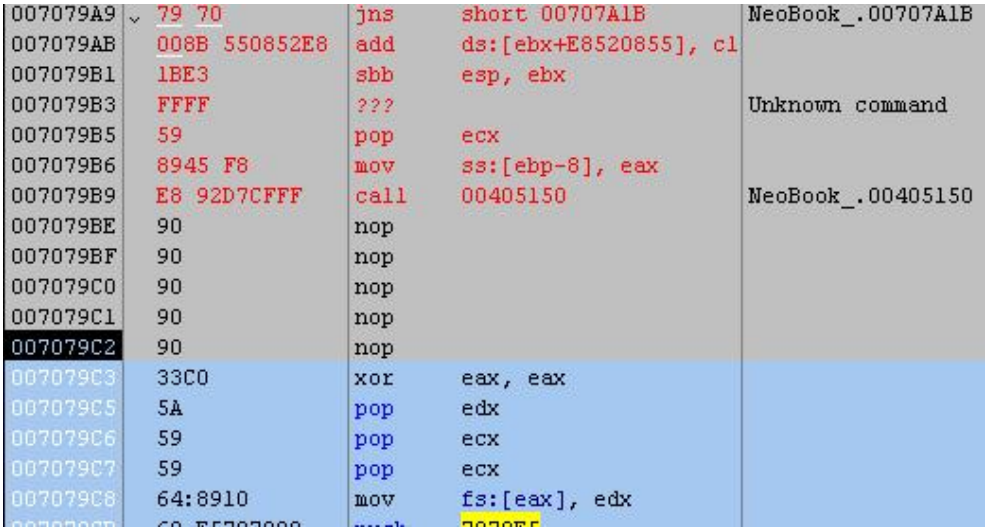
Vuelvo a intentar compilar y ahora me da este error:



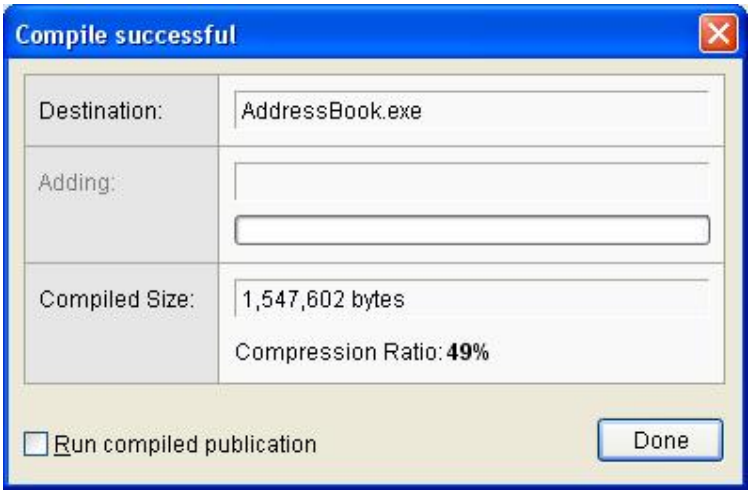
Repito el proceso y ahora para aquí:



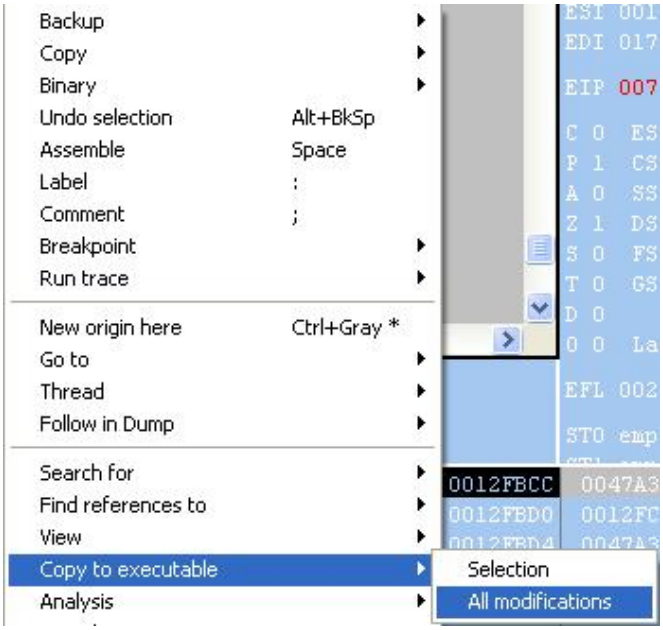
Lo reparo:



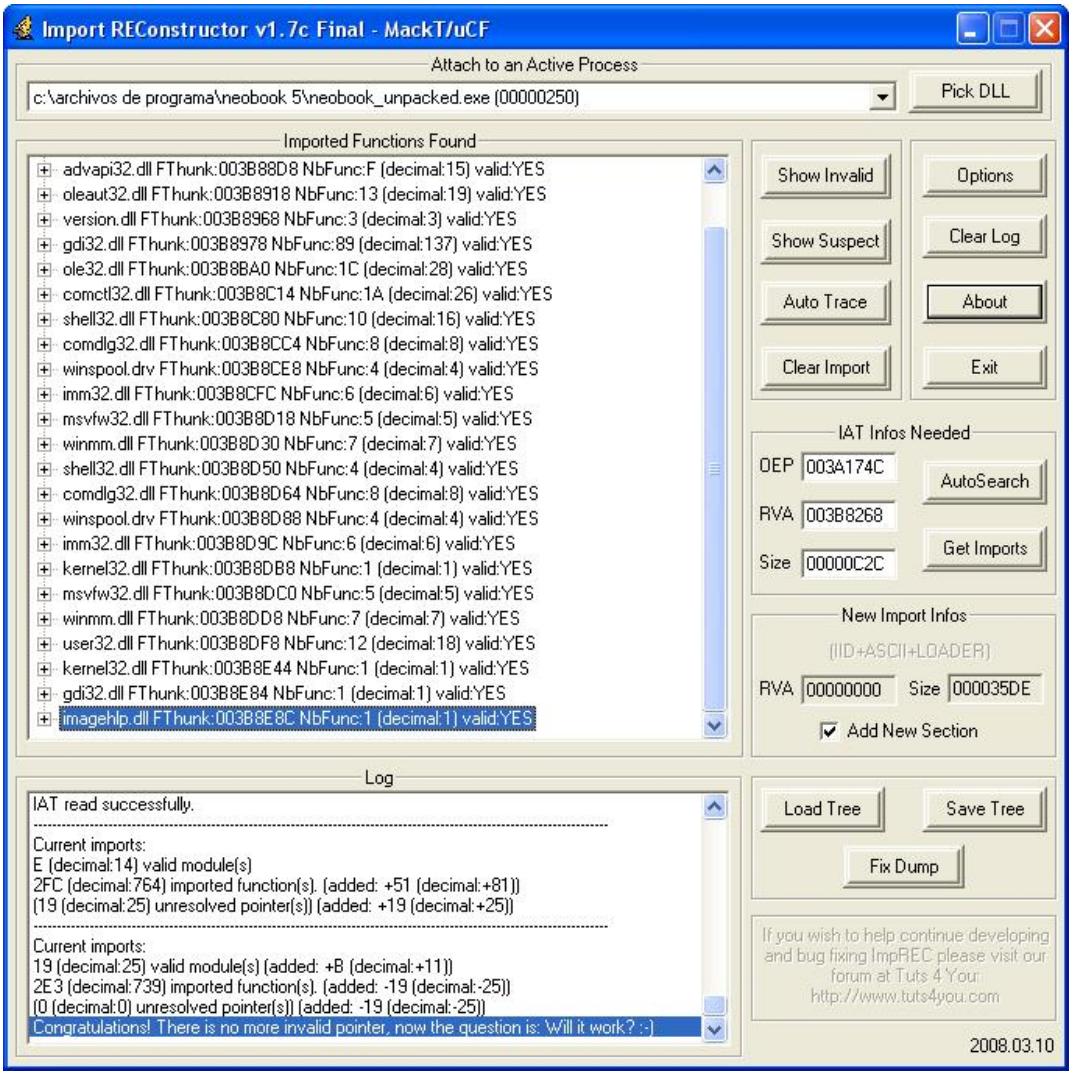
Vuelvo a intentar compilar y:



Ya puesto reparo todos los bloques de NOP´s que quedan y después guardo todos los cambios:



Ahora abro Import Rec, elijo el proceso del desempacado, doy al botón “Auto Search” y cambio el tamaño de la IAT a 0xC2C para que coja también la parte que he reparado de la IAT, doy al botón “Get Imports”, luego al botón “Show Invalids”, hago clic derecho sobre uno de los seleccionados y selecciono “Cut thunk(s)” y ya está listo para reparar la IAT:



Doy a “Fix Dump”, selecciono el nombre del archivo que he guardado con todas las reparaciones y ya tengo el archivo con la IAT reparada.

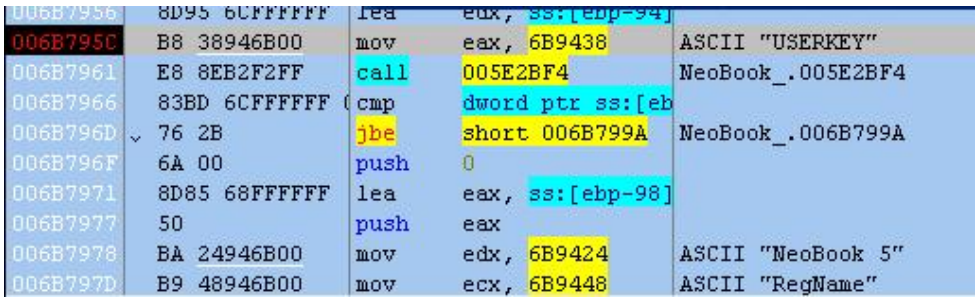
Registrando a la víctima

Cierro todos los Ollys, elimino todos los binarios que he ido creando menos el último que ha creado Import Rec ya que será sobre el que trabaje a partir de ahora.
Abro el nuevo archivo en Olly y busco entre sus strings y veo varias sospechosas:

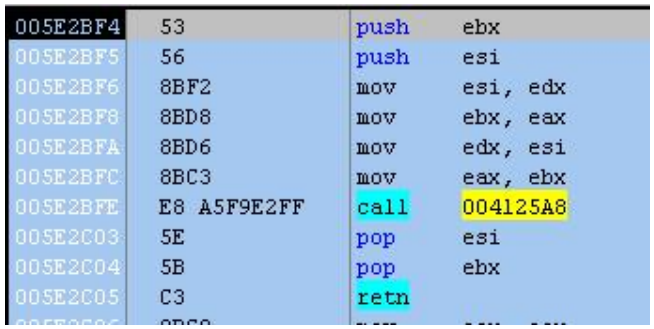
```
006BAC11 mov     edx, 6BAEAC                ASCII "NeoBook 5"
006BAC16 mov     ecx, 6BB720                ASCII "RegValue"
006BAC27 mov     edx, 6BAEAC                ASCII "NeoBook 5"
006BAC2C mov     ecx, 6BB6F0                ASCII "RegName"
006BAC3B mov     edx, 6BAEAC                ASCII "NeoBook 5"
006BAC40 mov     ecx, 6BB700                ASCII "RegNum"
006BAC53 mov     eax, 6BB734                ASCII "TOTALUSES"
```

```
006BAE0D mov     ecx, 6BB7B0                ASCII "Software\NeoSoft\NeoBook 5\ToolPalettes"
006BB8D4 mov     eax, 6BB988                ASCII "USERKEY"
006BB8E7 mov     eax, 6BB998                ASCII "DAYSINSTALLED"
```

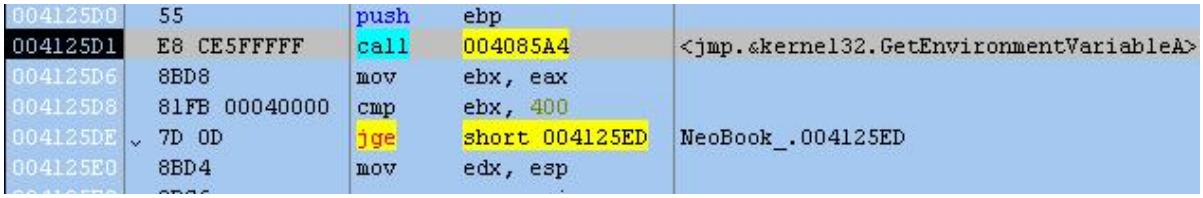
Pongo un BP en todas las ocurrencias de esas cadenas y doy a F9 y para aquí:

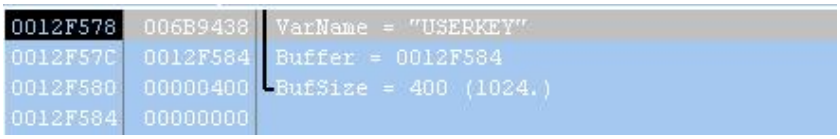


Entro en ese CALL y veo el siguiente código:



Entro en ese CALL y traceo hasta que llego aquí:





Según la MSDN:

"Retrieves the contents of the specified variable from the environment block of the calling process."

"If the function succeeds, the return value is the number of characters stored in the buffer pointed to by lpBuffer, *not including the terminating null character."*

"If the function fails, the return value is zero. If the specified environment variable was not found in the environment block, GetLastError returns ERROR_ENVVAR_NOT_FOUND."

O sea que si no la encuentra va a retornar 0 y si la encuentra retorna el tamaño de la cadena recuperada.
Esto es un truco que usa Armadillo para saber si sigue empacado o no ya que si lo desempacamos no estarán presentes ya que es el propio armadillo quien crea dichas variables de entorno.
La paso con F8 y, como era de esperar, obtengo 0 en EAX con lo que no la encontró y si sigo hasta el salto no saltará. ¿Y qué pasa si modifico esa función para que siempre obtenga un valor diferente de 0?
Reinicio Olly y hago esta modificación:

```
004125D0   55              push    ebp
004125D1   E8 CE5FFFFF     call    004085A4        <jmp.&kernel32.GetEnvironmentVariableA>
004125D6   B8 01000000     mov     eax, 1
004125DB   8BD8            mov     ebx, eax
004125DD   90              nop
004125DE   90              nop
004125DF   90              nop
004125E0   8BD4            mov     edx, esp
004125E2   8BC6            mov     eax, esi
004125E4   8BCB            mov     ecx, ebx
004125E6   E8 6532FFFF     call    00405850        NeoBook_.00405850
004125EB   EB 19           jmp     short 00412606  NeoBook_.00412606
004125ED   8BD3            mov     edx, ebx
```

Y ejecuto quitando todos los BP´s que hagan referencia a la cadena USERKEY y que justo debajo tengan la llamada “CALL 005E2BF4” ya que todas tomarán el mismo retorno y pasaremos todas esas comprobaciones sin problemas.
Queda así la lista de BP´s:

```
006B54E1  NeoBook_  Always    mov     eax, 6B55C0
006BAC53  NeoBook_  Always    mov     eax, 6BB734
006BB8E7  NeoBook_  Always    mov     eax, 6BB998
006BD660  NeoBook_  Always    push    6BD750
006BD685  NeoBook_  Always    push    6BD750
006BD6A2  NeoBook_  Always    mov     eax, 6BD760
```

Todos los MOV hacen referencia a las cadenas de las variables DAYSINSTALLED o TOTALUSES y los dos PUSH hacen referencia a USERKEY y los podemos encontrar aquí:

```
006BD65E   6A 00           push    0
006BD660   68 50D76B00     push    6BD750          ASCII "USERKEY"
006BD665   E8 3AAFD4FF     call    004085A4        <jmp.&kernel32.GetEnvironmentVariableA>
006BD66A   8BF0            mov     esi, eax
006BD66C   85F6            test    esi, esi
006BD66E   7E 21           jle     short 006BD691  NeoBook_.006BD691
006BD670   8BD6            mov     edx, esi
006BD672   4A              dec     edx
006BD673   8D45 FC         lea     eax, ss:[ebp-4]
006BD676   E8 4D87D4FF     call    00405DC8        NeoBook_.00405DC8
006BD67B   56              push    esi
006BD67C   8B45 FC         mov     eax, ss:[ebp-4]
006BD67F   E8 B885D4FF     call    00405C3C        NeoBook_.00405C3C
006BD684   50              push    eax
006BD685   68 50D76B00     push    6BD750          ASCII "USERKEY"
006BD68A   E8 15AFD4FF     call    004085A4        <jmp.&kernel32.GetEnvironmentVariableA>
006BD68F   EB 08           jmp     short 006BD699  NeoBook_.006BD699
006BD691   8D45 FC         lea     eax, ss:[ebp-4]
006BD694   E8 C780D4FF     call    00405760        NeoBook_.00405760
006BD699   837D FC 00      cmp     dword ptr ss:[ebp-4], 0
```

Si intentamos compilar nos parará en el primero y al pasar el CALL tendremos un bonito 0 en EAX con lo que nos estaría pillando y saltaría en el siguiente salto condicional.
Hago la siguiente modificación para pasar esa parte sin problema:

```
006BD656  0F84 C5000000   je      006BD721                NeoBook_.006BD721
006BD65C  6A 00           push    0
006BD65E  6A 00           push    0
006BD660  68 50D76B00     push    6BD750                  ASCII "USERKEY"
006BD665  E8 3AAFD4FF     call    004085A4                <jmp.&kernel32.GetEnvironmentVariableA>
006BD66A  BE 01000000     mov     esi, 1
006BD66F  90              nop
006BD670  8BD6            mov     edx, esi
006BD672  4A              dec     edx
006BD673  8D45 FC         lea     eax, ss:[ebp-4]
```

Sigo traceando y paso el segundo CALL hasta llegar aquí:

```
006BD684  50              push    eax
006BD685  68 50D76B00     push    6BD750                  ASCII "USERKEY"
006BD68A  E8 15AFD4FF     call    004085A4                <jmp.&kernel32.GetEnvironmentVariableA>
006BD68F  EB 08           jmp     short 006BD699          NeoBook_.006BD699
006BD691  8D45 FC         lea     eax, ss:[ebp-4]
006BD694  E8 C780D4FF     call    00405760                NeoBook_.00405760
006BD699  837D FC 00      cmp     dword ptr ss:[ebp-4], 0
006BD69D  75 6E           jnz     short 006BD70D          NeoBook_.006BD70D
006BD69F  8D55 F8         lea     edx, ss:[ebp-8]
006BD6A2  B8 60D76B00     mov     eax, 6BD760             ASCII "DAYSINSTALLED"
006BD6A7  E8 4855F2FF     call    005E2BF4                NeoBook_.005E2BF4
006BD6AC  8B45 F8         mov     eax, ss:[ebp-8]
```

No va a saltar y lo siguiente que veo es que va a buscar la variable de entorno DAYSINSTALLED así que, como sé que es porque no ha encontrado a USERKEY en el CALL último que ejecutó, modifico ese salto para que salte siempre:

```
006BD685  68 50D76B00     push    6BD750                  ASCII "USERKEY"
006BD68A  E8 15AFD4FF     call    004085A4                <jmp.&kernel32.GetEnvironmentVariableA>
006BD68F  EB 08           jmp     short 006BD699          NeoBook_.006BD699
006BD691  8D45 FC         lea     eax, ss:[ebp-4]
006BD694  E8 C780D4FF     call    00405760                NeoBook_.00405760
006BD699  837D FC 00      cmp     dword ptr ss:[ebp-4], 0
006BD69D  EB 6E           jmp     short 006BD70D          NeoBook_.006BD70D
006BD69F  8D55 F8         lea     edx, ss:[ebp-8]
006BD6A2  B8 60D76B00     mov     eax, 6BD760             ASCII "DAYSINSTALLED"
006BD6A7  E8 4855F2FF     call    005E2BF4                NeoBook_.005E2BF4
```

Con esto el programa queda totalmente registrado con lo que podemos guardar los cambios pero lo guardaré con otro nombre diferente, por ejemplo “NeoBook_unpacked _cracked.exe” para no modificar este y poder ver otra forma de pasar todas las comprobaciones de GetEnvironmentVariableA con un simple injerto.
Reinicio Olly sin guardar los cambios y me apunto la dirección del OEP ya que después de mi injerto necesitaré que siga la ejecución justo desde ahí:

```
007A174A  7A 00           jpe     short 007A174C          NeoBook_.<ModuleEntryPoint>
007A174C  55              push    ebp
007A174D  8BEC            mov     ebp, esp
007A174F  83C4 F0         add     esp, -10
007A1752  53              push    ebx
007A1753  B8 2C077A00     mov     eax, 7A072C
007A1758  E8 7B69C6FF     call    004080D8                NeoBook_.004080D8
007A175D  8B1D 4C0F7B00   mov     ebx, ds:[7B0F4C]        NeoBook_.007B2B88
007A1763  E8 E0EDFFFF     call    007A0548                NeoBook_.007A0548
007A1768  84C0            test    al, al
007A176A  75 75           jnz     short 007A17E1          NeoBook_.007A17E1
```

En el injerto necesito llamar a SetEnvironmentVariableA y no está en la IAT por lo que, o bien la añado a la IAT antes de repararla con Import Rec para que al repararla ya esté, o la uso dinámicamente y para eso necesitaré llamar también a GetModuleHandleA y a GetProcAddress y, además, necesitaré las cadenas “kernel32.dll”, “USERKEY”, “SetEnvironmentVariableA”, y una cadena que contendrá el valor de USERKEY.

Como sé que ya usa GetModuleHandleA para kernel32.dll busco todas las referencias a esa api y pongo un BP en todas y doy a F9. La primera vez que para es para obtener el modulo del proceso pero la segunda vez tengo esto:

```
004014D8  - FF25 C8827B00  jmp     ds:[7B82C8]          kernel32.GetModuleFileNameA
004014DE    8BC0           mov     eax, eax
004014E0  - FF25 C4827B00  jmp     ds:[7B82C4]          kernel32.GetModuleHandleA
004014E6    8BC0           mov     eax, eax
004014E8  - FF25 C0827B00  jmp     ds:[7B82C0]          kernel32.GetProcAddress
```

```
0012EA60  00407296  CALL to GetModuleHandleA from NeoBook_.00407
0012EA64  00407428  pModule = "kernel32.dll"
```

Justo lo que necesito así que paso la llamada a la función y llego aquí:

```
0040728C    68 28744000    push    407428               ASCII "kernel32.dll"
00407291    E8 4AA2FFFF    call    004014E0             <jmp.&kernel32.GetModuleHandleA>
00407296    8BF0           mov     esi, eax             kernel32.7C800000
00407298    85F6           test    esi, esi
0040729A  v 74 44          je      short 004072E0       NeoBook_.004072E0
0040729C    68 38744000    push    407438               ASCII "GetLongPathNameA"
004072A1    56             push    esi
004072A2    E8 41A2FFFF    call    004014E8             <jmp.&kernel32.GetProcAddress>
```

Con eso ya tengo gran parte del código necesario para el injerto ya que solo necesitaría modificar el PUSH de la línea 0x40729C para que apunte a la cadena “SetEnvironmentVariableA” y ya me daría la dirección a dicha función.
Busco una zona al final de la sección y dejo mi injerto así:

```
007A1F32    0000           add     ds:[eax], al
007A1F34    0000           add     ds:[eax], al
007A1F36    60             pushad
007A1F37    68 28744000    push    407428               ASCII "kernel32.dll"
007A1F3C    E8 9FF5C5FF    call    004014E0             <jmp.&kernel32.GetModuleHandleA>
007A1F41    68 601F7A00    push    7A1F60               ASCII "SetEnvironmentVariableA"
007A1F46    50             push    eax
007A1F47    E8 9CF5C5FF    call    004014E8             <jmp.&kernel32.GetProcAddress>
007A1F4C    68 7A1F7A00    push    7A1F7A               ASCII "AAAAAA-AAAAAA-AAAAAA-AAAAAA-AAAAAA-AAAAAA"
007A1F51    68 B0556B00    push    6B55B0               ASCII "USERKEY"
007A1F56    FFD0           call    eax
007A1F58    61             popad
007A1F59  ^ E9 EEF7FFFF    jmp     007A174C             NeoBook_.007A174C
007A1F5E    0000           add     ds:[eax], al
```

Y la parte donde tengo las cadenas que necesito está justo debajo:

```
Address   Hex dump                                          ASCII
007A1F60  53 65 74 45 6E 76 69 72 6F 6E 6D 65 6E 74 56 61  SetEnvironmentVa
007A1F70  72 69 61 62 6C 65 41 00 00 20 41 41 41 41 41 41  riableA.. AAAAAA
007A1F80  2D 41 41 41 41 41 41 2D 41 41 41 41 41 41 2D 41  -AAAAAA-AAAAAA-A
007A1F90  41 41 41 41 41 2D 41 41 41 41 41 41 2D 41 41 41  AAAAA-AAAAAA-AAA
007A1FA0  41 41 41 00 00 00 00 00 00 00 00 00 00 00 00 00  AAA.............
007A1FB0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Ahora voy a cambiar el OEP:

Address	Hex dump	Data	Comment
0040013A	00	DB 00	MajorLinkerVersion = 0
0040013B	00	DB 00	MinorLinkerVersion = 0
0040013C	00103A00	DD 003A1000	SizeOfCode = 3A1000 (3805184.)
00400140	00D05500	DD 0055D000	SizeOfInitializedData = 55D000
00400144	00000000	DD 00000000	SizeOfUninitializedData = 0
00400148	4C173A00	DD 003A174C	AddressOfEntryPoint = 3A174C
0040014C	00100000	DD 00001000	BaseOfCode = 1000
00400150	00203A00	DD 003A2000	BaseOfData = 3A2000
00400154	00004000	DD 00400000	ImageBase = 400000
00400158	00100000	DD 00001000	SectionAlignment = 1000

Para que apunte a la primera línea de mi injerto:

Address	Hex dump	Data	Comment
0040013A	00	DB 00	MajorLinkerVersion = 0
0040013B	00	DB 00	MinorLinkerVersion = 0
0040013C	00103A00	DD 003A1000	SizeOfCode = 3A1000 (3805184.)
00400140	00D05500	DD 0055D000	SizeOfInitializedData = 55D000
00400144	00000000	DD 00000000	SizeOfUninitializedData = 0
00400148	361F3A00	DD 003A1F36	AddressOfEntryPoint = 3A1F36
0040014C	00100000	DD 00001000	BaseOfCode = 1000
00400150	00203A00	DD 003A2000	BaseOfData = 3A2000
00400154	00004000	DD 00400000	ImageBase = 400000
00400158	00100000	DD 00001000	SectionAlignment = 1000

Guardo todos los cambios y reinicio Olly y veo esto:

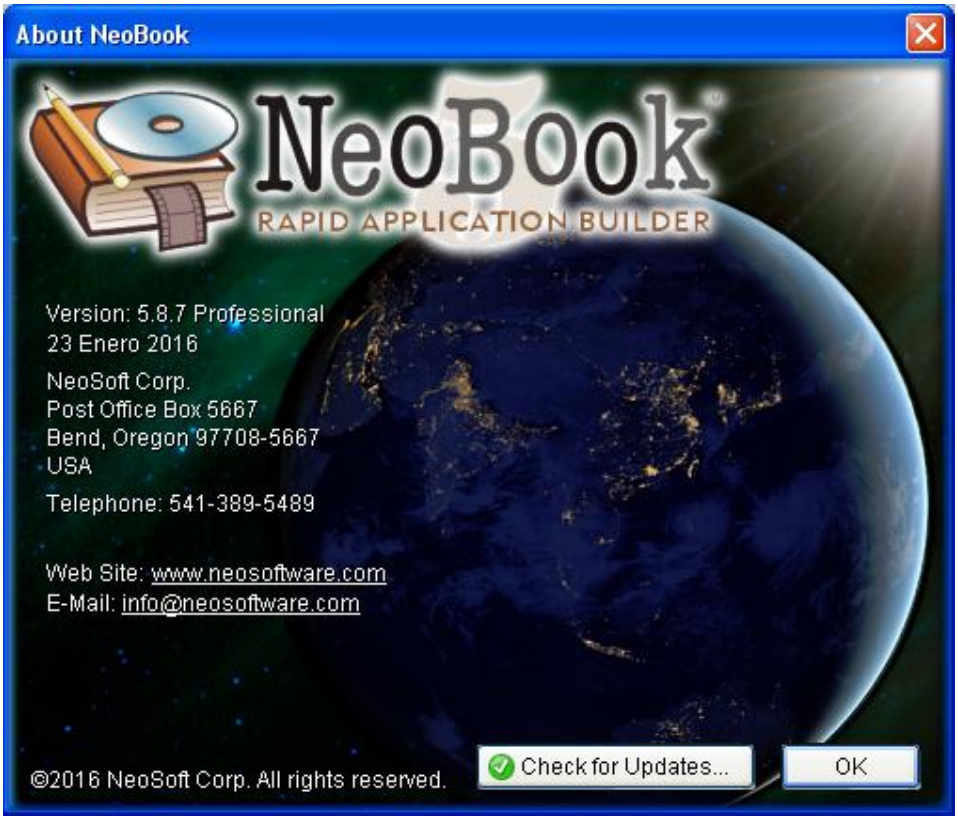
```
007A1F32   0000          add     ds:[eax], al
007A1F34   0000          add     ds:[eax], al
007A1F36   60            pushad
007A1F37   68 28744000   push    407428          ASCII "kernel32.dll"
007A1F3C   E8 9FF5C5FF   call    004014E0        <jmp.&kernel32.GetModuleHandleA>
007A1F41   68 601F7A00   push    7A1F60          ASCII "SetEnvironmentVariableA"
007A1F46   50            push    eax
007A1F47   E8 9CF5C5FF   call    004014E8        <jmp.&kernel32.GetProcAddress>
007A1F4C   68 7A1F7A00   push    7A1F7A          ASCII "AAAAAA-AAAAAA-AAAAAA-AAAAAA-AAAAAA-AAAAAA"
007A1F51   68 B0556B00   push    6B55B0          ASCII "USERKEY"
007A1F56   FFD0          call    eax
007A1F58   61            popad
007A1F59 ^ E9 EEF7FFFF   jmp     007A174C        NeoBook_.007A174C
007A1F5E   0000          add     ds:[eax], al
```

Si Ponemos un BP en todas las llamadas a GetEnvironmentVariableA veo cómo recupera un valor diferente de 0, concretamente 0x29 que es el largo de la cadena que puse y pasamos todas las comprobaciones satisfactoriamente teniendo el programa totalmente funcional pero hay un detalle, ¿recuerdan estas cadenas?

```
006BAC11 mov     edx, 6BAEAC        ASCII "NeoBook 5"
006BAC16 mov     ecx, 6BB720        ASCII "RegValue"
006BAC27 mov     edx, 6BAEAC        ASCII "NeoBook 5"
006BAC2C mov     ecx, 6BB6F0        ASCII "RegName"
006BAC3B mov     edx, 6BAEAC        ASCII "NeoBook 5"
006BAC40 mov     ecx, 6BB700        ASCII "RegNum"
006BAC53 mov     eax, 6BB734        ASCII "TOTALUSES"
```

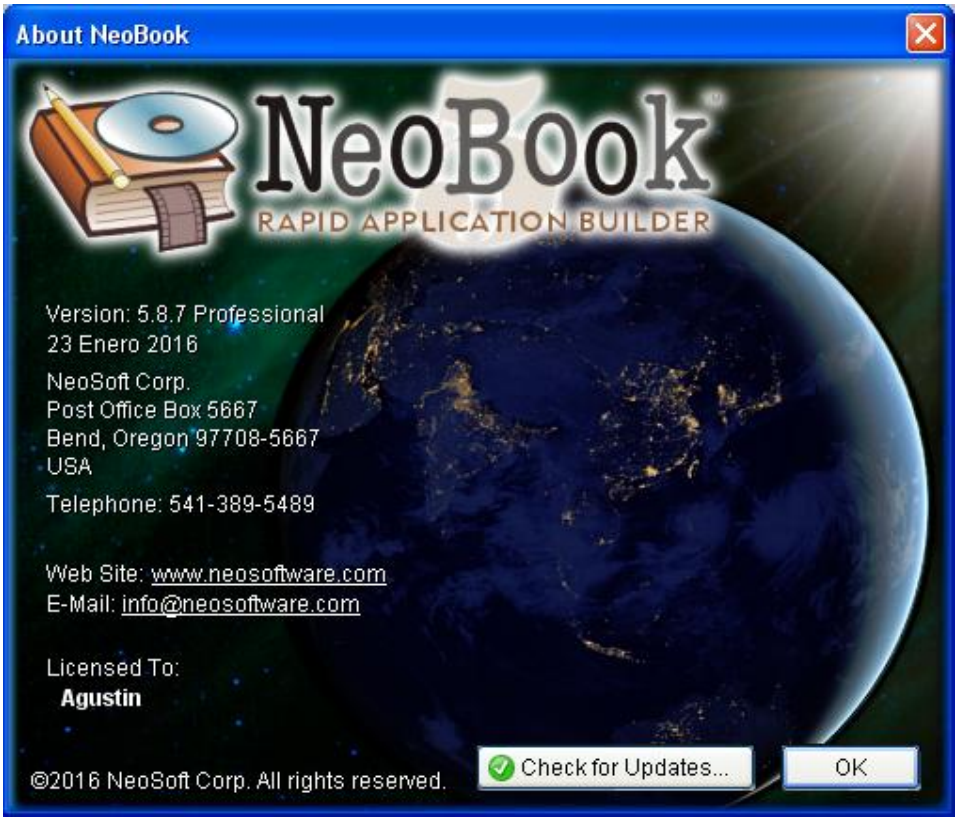
Desde el principio me parecieron sospechosas así que miro regedit y en la ruta “HKEY_CURRENT_USER\Software\NeoSoft\NeoBook 5” veo esto:

Si miro el About del programa tal como está parcheado ahora veo esto:

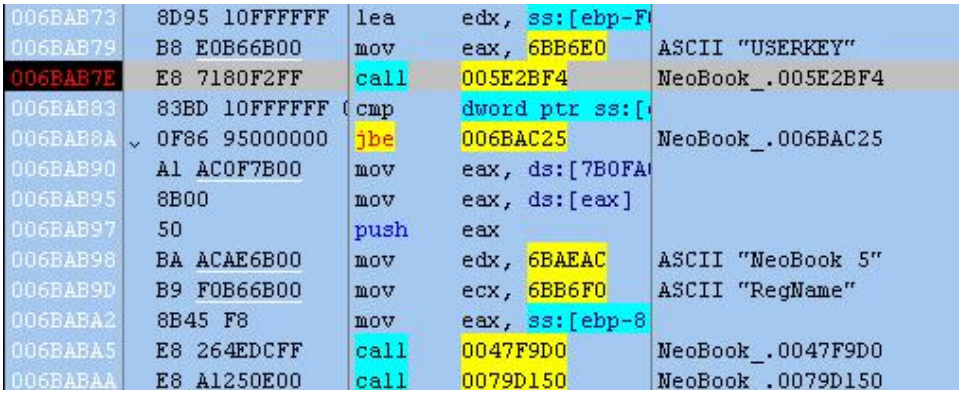


Cambio esos datos:

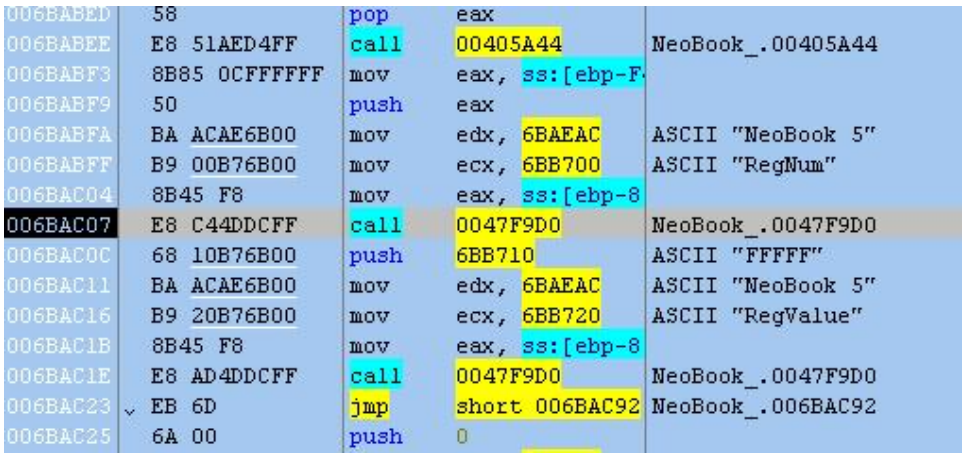
Doy a F9 para que se ejecute el programa y voy al About y veo esto:



Cierro el programa y para aquí en Olly:

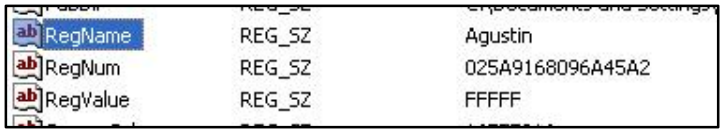


Sigo traceando y paso sin problemas esa comprobación y si sigo traceando puedo ver como en los dos siguientes CALL´s va a guardar en el registro RegName el nombre que puse y luego llego aquí:

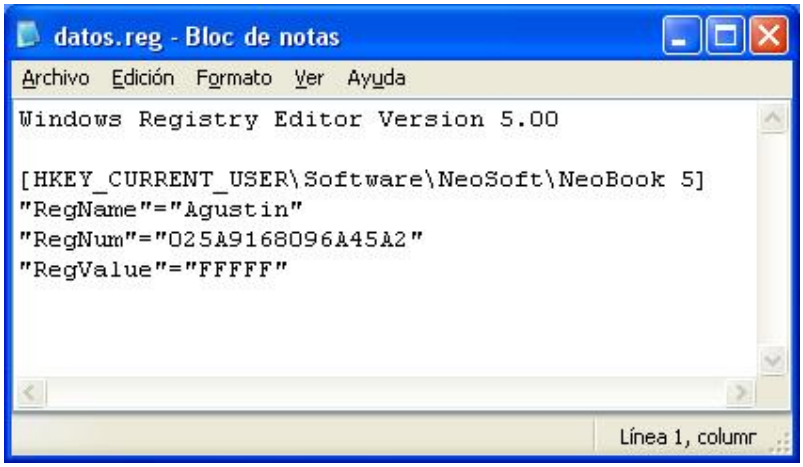


```
0012F0CC  01971E64  ASCII "025A9168096A45A2"
0012F0D0  0012F0DC  Pointer to next SEH record
```

Curioso, va a meter en el valor de RegValue la cadena que vemos en la pila y que supongo que será el serial encriptado pero la verdad es que no tengo ni idea de cómo lo hace pero sí que lo hace más arriba en los CALL´s. Luego hace lo propio con RegValue pero esta vez la cadena es “FFFFF” y está puesta a fuego en el ejecutable por lo que no se calcula en ningún momento. Paso esa zona y ya no hay nada más de interés así que doy a F9 y para en otro CALL que comprobará la variable de entorno USERKEY, la pasamos sin problema y al dar a F9 se termina. En ningún momento parará a buscar las variables de entorno TOTALUSES y DAYSINSTALLED ya que para que eso pase tendría que no pasar la comprobación de USERKEY por lo que puedo decir que estoy registrado jejeje.
Si miro lo que obtuve en el registro veo esto:



Pruebo a cambiar la cadena que usé como valor para la variable de entorno USERKEY en mi injerto y no cambia el valor de RegNum.
Pruebo a cambiar el nombre que tengo puesto en RegName y tampoco hay cambio alguno en el valor de RegNum. La verdad es que no sé de dónde saca esa cadena ni tengo ganas de complicarme más porque el programa queda registrado tal cual pero sí que la curiosidad me puede y me gustaría que si alguien sabe que es y que utilidad puede tener soy todo oídos jejeje. Lo único que queda es exportar los valores de registro para poder colocar más fácil el nombre deseado:



Y listo.