

Teoría N°	La primera del 2008 ya era hora
Objetivo	Aprender a Eliminar una NAG basándonos en la pila.
Victima	Tune up Programa para mantener el PC brillante
Herramientas	1 Depurador y un poco de Cabeza no vendrá nada mal saber algo sobre ASM.
Dificultad	Hazlo sin leer el tutorial y opina! Si te doy la Solución ☺.
Compilador	Esta claro (Delphi).
Cracker	KmL ReVeRsEr ®



RENUNCIA

Este tutorial tiene una finalidad únicamente educativa, no asumo ninguna Responsabilidad por el mal uso que se le pueda dar a éste material.

Bueno en este nuevo documento vamos a aprender como eliminar una nag con nada mas ver la pila y con un poco de cabeza caer encima del salto mágico.

-No es magia es Ciencia OJO hay que practicar bastante.

Comenzamos

1-Examinamos el Programa.




Ya tenemos la información necesaria así que a atacar se a dicho.

Empezamos el Ataque.

Comenzamos directamente con el depurador así que abrimos nuestro depurador y seleccionamos el Archivo Integrator.exe

Aparecen unas alertas que pasamos de ellas olímpicamente.

Bueno vamos colocando un punto de corte en Show Window así que en nuestra barra de

Comandos ponemos  y ejecutamos el programa con F9 o el botón play.

Bueno vamos a aprovechar la Pila la que nos releva mucha información de lo que se ha hecho ya veréis la importancia de esta ☺

Echamos una mirada a la pila y vemos estoy algo muy sospechoso ☺ pero demasiado.

```
0012FE44 00009A60
0012FE48 00000000
0012FE4C 00000000
0012FE50 2002DD14 rtl100.@Classes@TMemoryStream@
0012FE54 01F332A0
0012FE58 0012FEB4
0012FE5C 008FA71D RETURN to AppIniti.008FA71D
0012FE60 0012FE6C Pointer to next SEH record
0012FE64 008FA735 SE handler
0012FE68 0012FEB4
0012FE6C 0012FEC0 Pointer to next SEH record
0012FE70 008FA772 SE handler
0012FE74 0012FEB4
0012FE78 00000198
0012FE7C 00422264 <JMP.&AppInitialization.@Tuappinit@initialization$qqrv>
0012FE80 000000EE
0012FE84 01EF6728
0012FE88 01EF60B8
0012FE8C 01FE31C8 ASCII "Trial\071213_a\main.oss"
0012FE90 00000000
0012FE94 01FE31F0 ASCII "Trial\071213_a\main.htm"
0012FE98 00000000
0012FE9C 00000000
0012FEA0 00000000
0012FEA4 0000001E
0012FEA8 01F33F08 ASCII "body { margin: 0 0 0; }>:>:>: { color: #393939"
0012FEAC 01FDB268 ASCII "Trial\071213_a\"
0012FEB0 01ECC678 ASCII "<config>>: htmlonly = on activate_over_trial_mode =
0012FEB4 0012FF04
0012FEB8 008FF934 RETURN to AppIniti.008FF934 from AppIniti.@Ufrmagscreen@ShowNa
0012FEBC 00000000
0012FEC0 0012FF0C Pointer to next SEH record
0012FEC4 008FF970 SE handler
```

Nos interesan los punteros

0012FE5C |008FA71D RETURN to AppIniti.008FA71D

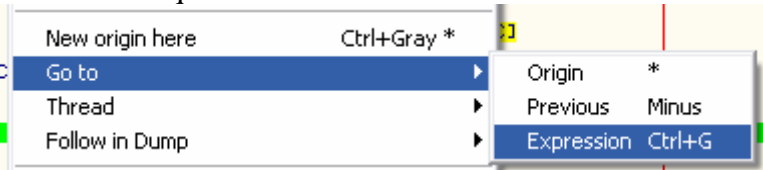
0012FEB8 |008FF934 RETURN to AppIniti.008FF934 fromEtcetera...

Bueno ya tenemos las direcciones mágicas que son

008FA71D

008FF934 (en mi caso) Pueden cambiar en vuestro PC

La mas importante es la primera así que vamos a ver que hacemos bueno vamos a esta dirección así que seleccionamos

 Y pegamos nuestra

dirección en mi caso. 008FA71D

Y caemos aquí

```

008FA715 8B10      MOV EDX,DWORD PTR DS:[EAX]
008FA717 FF92 FC000000 CALL DWORD PTR DS:[EDX+FC]
008FA71D 33C0     XOR EAX,EAX
008FA71F 5A      POP EDX
008FA720 59      POP ECX
008FA721 59      POP ECX
008FA722 64:8910  MOV DWORD PTR FS:[EAX],EDX
  
```

Bueno aquí hay algo interesante

```

008FA6E3 04 01    UL,1
008FA6E5 E8 AA73FFFF CALL <JMP.&vcl100.@Extctrls@TTimer@SetEna
008FA6EA EB 12    JMP SHORT AppIniti.008FA6FE
008FA6EC A1 3C309000 MOV EAX,DWORD PTR DS:[@Ufrmnagscreen@frmn
008FA6F1 8B80 64030000 MOV EAX,DWORD PTR DS:[EAX+364]
008FA6F7 33D2     XOR EDX,EDX
008FA6F9 E8 9673FFFF CALL <JMP.&vcl100.@Extctrls@TTimer@SetEna
008FA6FE 8B15 08599000 MOV EDX,DWORD PTR DS:[&MainControls.@Tur
008FA704 8B12    MOV EDX,DWORD PTR DS:[EDX]
008FA706 A1 3C309000 MOV EAX,DWORD PTR DS:[@Ufrmnagscreen@frmn
008FA708 E8 7C72FFFF CALL <JMP.&vcl100.@Controls@TControl@SetT
008FA710 A1 3C309000 MOV EAX,DWORD PTR DS:[@Ufrmnagscreen@frmn
008FA715 8B10    MOV EDX,DWORD PTR DS:[EAX]
008FA717 FF92 FC000000 CALL DWORD PTR DS:[EDX+FC]
008FA71D 33C0     XOR EAX,EAX
008FA71F 5A      POP EDX
  
```



Justamente donde indica la flecha hay un salto que evita la nag por lo cual tiene que haber otro que evite el salto así que nos colocamos justo debajo del salto

```

008FA6E5 EB 12    CALL <JMP.&vcl100.@Extctrls@TTimer@SetEna
008FA6EA EB 12    JMP SHORT AppIniti.008FA6FE
008FA6EC A1 3C309000 MOV EAX,DWORD PTR DS:[@Ufrmnagscreen@frmn
008FA6F1 8B80 64030000 MOV EAX,DWORD PTR DS:[EAX+364]
008FA6F7 33D2     XOR EDX,EDX
  
```

Y seleccionamos



Y encontramos estos

```

Address Disassembly Comment
008FA6C8 JE SHORT AppIniti.008FA6EC
008FA6D6 JNZ SHORT AppIniti.008FA6EC
008FA6EC MOV EAX,DWORD PTR DS:[@Ufrmnagscreen@fr (Initial CPU selection)
  
```

Bueno seleccionamos el primero y le damos a

```

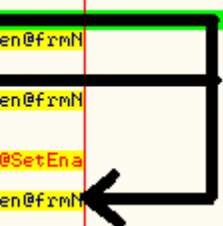
Address Disassembly Comment
008FA6C8 JE SHORT AppIniti.008FA6EC
008FA6D6 JNZ SHORT AppIniti.008FA6EC
008FA6EC MOV EAX,DWORD PTR DS:[@Ufrmnagscreen@fr
  
```

Follow in Disassembler Enter

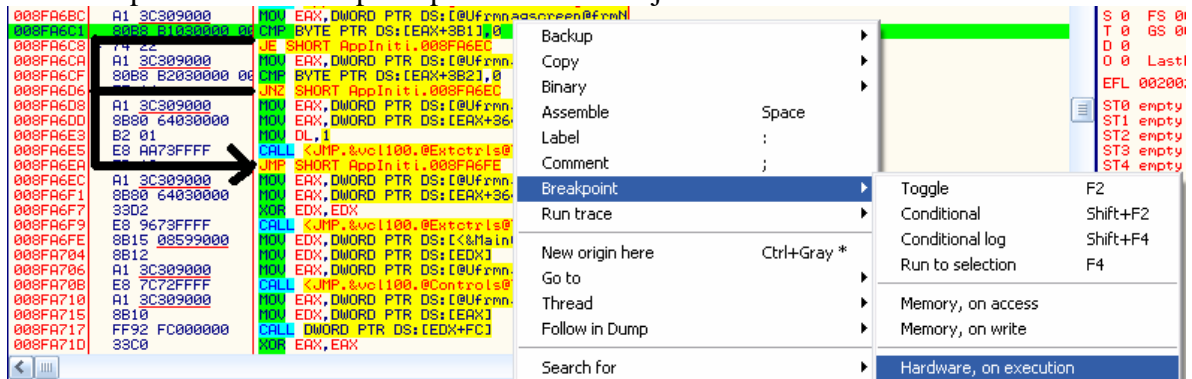
Bueno hay vemos algo interesante esto Exactamente ☺

```

008FA6C1 8088 B1030000 00 CMP BYTE PTR DS:[EAX+3B1],0
008FA6C3 74 22    JE SHORT AppIniti.008FA6EC
008FA6CA A1 3C309000 MOV EAX,DWORD PTR DS:[@Ufrmnagscreen@frmn
008FA6CF 8088 B2030000 00 CMP BYTE PTR DS:[EAX+3B2],0
008FA6D6 75 14    JNZ SHORT AppIniti.008FA6EC
008FA6D8 A1 3C309000 MOV EAX,DWORD PTR DS:[@Ufrmnagscreen@frmn
008FA6DD 8B80 64030000 MOV EAX,DWORD PTR DS:[EAX+364]
008FA6E3 B2 01    MOV DL,1
008FA6E5 E8 AA73FFFF CALL <JMP.&vcl100.@Extctrls@TTimer@SetEna
008FA6EA EB 12    JMP SHORT AppIniti.008FA6FE
008FA6EC A1 3C309000 MOV EAX,DWORD PTR DS:[@Ufrmnagscreen@frmn
008FA6F1 8B80 64030000 MOV EAX,DWORD PTR DS:[EAX+364]
  
```



Bueno le ponemos un Brakpoint por hardware al ejecutarse a la instrucción



Bueno lo del BP por hardware es porque para mi es mas eficaz y cómodo (Todo esto relativo) en este caso nos Da mejores ventajas este.

Después de ponerlo reiniciamos el Depurador. iniciamos la ejecución normal de la victima hasta que el depurador

Corte la ejecución aquí.

```

008FA6C1 808B B1030000 06 CMP     BYTE PTR DS:[EAX+3B1],0
008FA6C8 74 22      JE     SHORT AppIniti.008FA6EC
008FA6CA A1 3C309000 MOV    EAX,DWORD PTR DS:[@Ufrmnagscreen@frmN
008FA6CF 808B B2030000 06 CMP     BYTE PTR DS:[EAX+3B2],0
008FA6D6 75 14      JNZ    SHORT AppIniti.008FA6EC
008FA6D8 A1 3C309000 MOV    EAX,DWORD PTR DS:[@Ufrmnagscreen@frmN
008FA6DD 8B80 64030000 MOV    EAX,DWORD PTR DS:[EAX+364]
008FA6E3 B2 01      MOV    DL,1
008FA6E5 E8 AA73FFFF CALL   <JMP.&vc1100.@ExtCtrls@TTimer@SetEna
008FA6EA EB 12      JMP    SHORT AppIniti.008FA6FE
008FA6EC A1 3C309000 MOV    EAX,DWORD PTR DS:[@Ufrmnagscreen@frmN
008FA6F1 8B80 64030000 MOV    EAX,DWORD PTR DS:[EAX+364]

```

Bueno esto lo he echo para ordenarnos así bueno tenemos menos suciedad en la pila. ☺

Lo cual nos facilitara dar con la protección ☺

Si miramos la pila vemos esto

```

0012FE38 01FA1540
0012FE3C 01FCEA01
0012FE40 00000000
0012FE44 00000000
0012FE48 00000000
0012FE4C 00000000
0012FE50 01F332A0
0012FE54 0012FEB4
0012FE58 008FA6BC AppIniti.008FA6BC
0012FE5C 0012FE84
0012FE60 0012FE6C Pointer to next SEH record
0012FE64 008FA735 SE handler
0012FE68 0012FEB4
0012FE6C 0012FEC0 Pointer to next SEH record
0012FE70 008FA772 SE handler
0012FE74 0012FEB4
0012FE78 00000198
0012FE7C 00422264 <JMP.&AppInitialization.@Tuappinit@initialization$qgrv>
0012FE80 000000EE
0012FE84 01EF6728
0012FE88 01EF60B8
0012FE8C 01FE31C8 ASCII "Trial\071213_a\main.css"
0012FE90 00000000
0012FE94 01FE31F0 ASCII "Trial\071213_a\main.htm"
0012FE98 00000000
0012FE9C 00000000
0012FEA0 00000000
0012FEA4 0000001E
0012FEA8 01F33F08 ASCII "body { /margin: 0 0 0; /} /b { /color: #3939:
0012FEAC 01FDB268 ASCII "Trial\071213_a\"
0012FEB0 01ECC678 ASCII "<config> /htmlonly = on / activate_over_trial_mode :
0012FEB4 0012FF04
0012FEB8 008FF934 RETURN to AppIniti.008FF934 from AppIniti.@Ufrmnagscreen@Show
0012FEC0 00000000
0012FEC4 008FF970 Pointer to next SEH record
0012FEC8 0012FF04 SE handler
0012FECC 00000000

```

Buscamos de donde ha sido llamada

0012FEB8 |008FF934 RETURN to AppIniti.008FF934 from

[AppIniti.@Ufrmnagscreen@ShowNagscreen\\$gqriio](#)

Bueno hay la tenemos la dirección 008FF934

Bueno vamos hay a ver que es lo que pasa ☺

```

008FF900 59          POP     ECX
008FF90E E8 45AFFFFF CALL   AppIniti.@Ufrmnagscreen@ShowNagscree
008FF913 EB 1F       JMP     SHORT AppIniti.008FF934
008FF915 6A 00      PUSH   0
008FF917 A1 6C309000 MOV    EAX,DWORD PTR DS:[90306C]
008FF91C E8 0BECFFFF CALL   AppIniti.008FE520
008FF921 50        PUSH   EAX
008FF922 A1 6C309000 MOV    EAX,DWORD PTR DS:[90306C]
008FF927 E8 58EBFFFF CALL   AppIniti.008FE484
008FF92C 33C9     XOR    ECX,ECX
008FF92E 5A       POP    EDX
008FF92F E8 24AFFFFF CALL   AppIniti.@Ufrmnagscreen@ShowNagscree
008FF934 A1 58309000 MOV    EAX,DWORD PTR DS:[0Tuappinit@MUTEX_NA
008FF939 E8 3A18FFFF CALL   <JMP.&rt1100.@System@LStrToPChar$qq
008FF93E 50        PUSH   EAX
008FF93F 6A FF     PUSH   -1
008FF941 6A 00     PUSH   0
008FF943 E8 6C1BFFFF CALL   AppIniti.008F14B4
008FF948 A3 20239000 MOV    DWORD PTR DS:[0Tuappinit@NagMutexHand
008FF94D 33C0     XOR    EAX,EAX

```

Bueno pasa tres cuartos de lo mismo y esto a mi no me gusta así que vamos a ver si nos facilitamos las cosas

Esto me empieza a sonar bien pero muy bien si nos fijamos más

```

008FF900 . 59          POP     ECX
008FF90E . E8 45AFFFFF CALL   AppIniti.@Ufrmnagscreen@ShowNagscree
008FF913 . EB 1F       JMP     SHORT AppIniti.008FF934
008FF915 > 6A 00      PUSH   0
008FF917 . A1 6C309000 MOV    EAX,DWORD PTR DS:[90306C]
008FF91C . E8 0BECFFFF CALL   AppIniti.008FE520
008FF921 . 50        PUSH   EAX
008FF922 . A1 6C309000 MOV    EAX,DWORD PTR DS:[90306C]
008FF927 . E8 58EBFFFF CALL   AppIniti.008FE484
008FF92C . 33C9     XOR    ECX,ECX
008FF92E . 5A       POP    EDX
008FF92F . E8 24AFFFFF CALL   AppIniti.@Ufrmnagscreen@ShowNagscree
008FF934 > A1 58309000 MOV    EAX,DWORD PTR DS:[0Tuappinit@MUTEX_NA
008FF939 . E8 3A18FFFF CALL   <JMP.&rt1100.@System@LStrToPChar$qq
008FF93E . 50        PUSH   EAX
008FF93F . 6A FF     PUSH   -1
008FF941 . 6A 00     PUSH   0
008FF943 . E8 6C1BFFFF CALL   AppIniti.008F14B4
008FF948 . A3 20239000 MOV    DWORD PTR DS:[0Tuappinit@NagMutexHand
008FF94D > 33C0     XOR    ECX,ECX
008FF94F . 5A       POP    EDX
008FF950 . 59          POP     ECX
008FF952 . 64:8910   MOV    DWORD PTR FS:[EAX],EDX
008FF955 . 68 77F98F00 PUSH  AppIniti.008FF977
008FF95A > 8D45 C8   LEA   EAX,DWORD PTR SS:[EBP-38]
008FF95D . E8 AE17FFFF CALL   <JMP.&rt1100.@System@LStrClr$qqrpv>
008FF962 . 8D45 D4   LEA   EAX,DWORD PTR SS:[EBP-2C]
008FF965 . 5A       POP    EDX

```

Arg3 =
Arg2 =
Arg1 =
AppIniti

Jumps from 008FF6E1, 008FF730, 008FF77F, 008FF7C6

Y eso de donde salio, bueno si hubierais ido viendo las rutinas con el desamplador nos hubiéramos dado cuenta pero buen

También a plena vista se notaba debido a esas 4 instrucciones que me resultaron sospechosas a si sin más.

Bueno como vemos en la foto hay 4 referencias y a nosotros nos interesa la primera.

```

008FF977 . 8BE5     MOV    ESP,EBP
008FF978 . 5D       POP    EDI

```

Jumps from 008FF6E1, 008FF730, 008FF77F, 008FF7C6

Address	Hex dump
00423080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00423088	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Copy pane to clipboard
Go to JNZ from 008FF6E1

0012FE6
0012FE6
0012FE6

Y caemos aquí

```

008FF6C0  .  5A 5C          PUSH 0
008FF6C2  .  6A 00         PUSH 0
008FF6C4  .  49           DEC ECX
008FF6C5  .  75 F0         JNZ SHORT AppIniti.008FF6C0
008FF6C7  .  33C0         XOR EAX,EAX
008FF6C9  .  55           PUSH EBP
008FF6CA  .  68 70F98F00  PUSH AppIniti.008FF970
008FF6CF  .  64:FF30     PUSH DWORD PTR FS:[EAX]
008FF6D2  .  64:8920     MOV DWORD PTR FS:[EAX],ESP
008FF6D5  .  A1 6C309000  MOV EAX,DWORD PTR DS:[90306C]
008FF6DA  .  E8 E9E7FFFF  CALL AppIniti.008FDEC8
008FF6DF  .  84C0         TEST AL,AL
008FF6E1  .  0F85 66020000 JNZ AppIniti.008FF940
008FF6E7  .  68 84F98F00  PUSH AppIniti.008FF984

```

Ya lo tenemos ese salto evita las nag y no hay nada que evite ese salto bueno si un Humano con un Depurador así

Que a parchear se a dicho lo cambiamos por un JMP

```

008FF6DA  .  E8 E9E7FFFF  CALL AppIniti.008FDEC8
008FF6DF  .  84C0         TEST AL,AL
008FF6E1  .  0F85 66020000 JMP AppIniti.008FF940
008FF6E6  .  90           NOP
008FF6E7  .  68 84F98F00  PUSH AppIniti.008FF984
008FF6EC  .  8D55 F0     LEA EDX,DWORD PTR SS:[EBP-10]
008FF6EF  .  23C0         XOR EAX,EAX

```

Y comprobamos que no removió la instrucción siguiente así que lo guardamos y lo probamos.

☺ Realmente Funciona.

Y con esto hemos eliminado la protección claro nos quedaría el info. Pero a mi personalmente me da igual he he ☺

Bueno espero que hoy hayas divertido y aprendido algo muy poco pero bastante para saber como eliminar un NAG nada más con la pila☺.

-RECORDAR: PRACTICAR Y REPRACTICAR Y ESCRIBIR

RECUERDA

SI CREES QUE UNA APLICAIÓN TE ES UTIL, ESTA BIEN PROGRAMADA Y ADEMAS TE GUSTA, NO SEAS GOLFO Y REGISTRATE, AYUDARÁS A SU PROGRAMADOR.

Agradecimientos: A toda la lista ☺ especialmente a la parte activa y a SOLID, RICARDO MEK Y C_C NCR Y Dapaf (hace mucho que no apareces) stxwei guan de dio y los que tengo en correo y en el MESSENGER si tu eres uno de ellos Gracias.

Aparte me gustaría agradecer a una persona que me metió en esto y me hizo despertar y salir de un camino equivocado por el cual me llevaba a ningún destino. Un profesor que me ayuda, me enseña y me forma día a día para llegar a uno de mis sueños.

Gracias, a todos ☺