Victima	ChmEditor 1.3
Protección	Asprotect (15 días)
Herramientas	Olly,
Objetivo	Quitar la protección de tiempo de Asprotect

Introducción

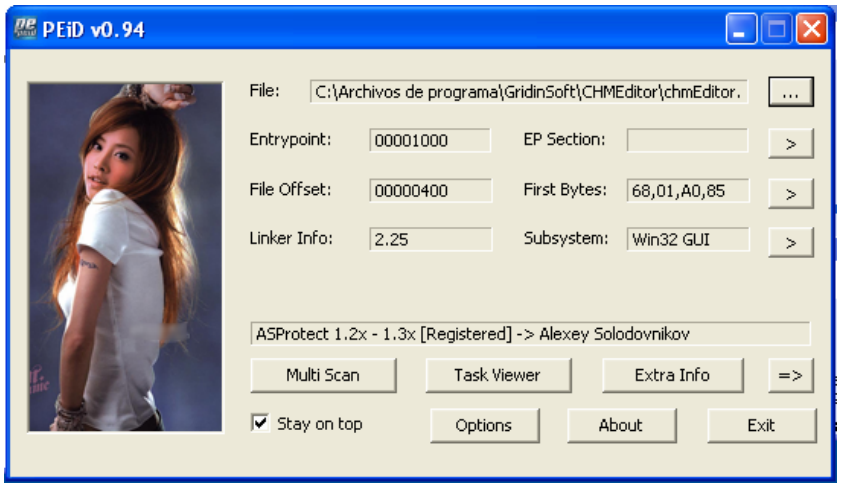
Hola a todos, después de un tiempo volvemos a la carga. En esta entrega vamos a ver como saltar la protección de tiempo de un Asprotect de los últimos.

Al parecer este packer sigue evolucionando, en este programa en concreto he visto complejo obtener un desempacado limpio, pero saltar el control de tiempo veremos que es muy sencillo.

Lo mejor de todo es que la protección de tiempo corre a cuenta del packer y muy posiblemente sea idéntica para todos los programas comprimidos con este bicho.

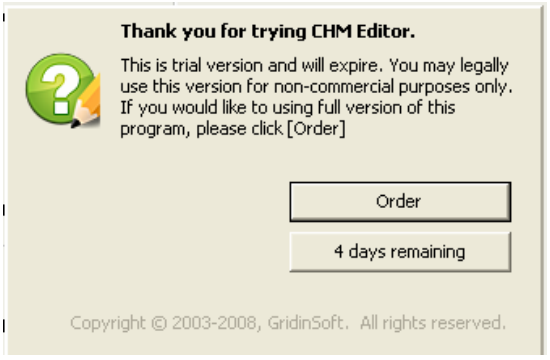
Conociendo al enemigo

Si pasamos el programa por el PEID vemos esto:



Como este PEID es viejito y no tengo mantenido las signaturas pues lo único que podemos sacar en claro es la protección pero no su versión.

Si ejecutamos el programa tenemos esto:



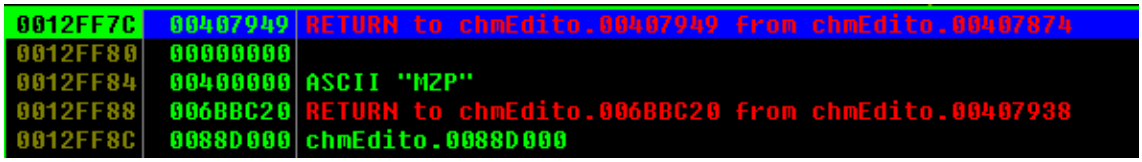
En este caso el programa es totalmente funcional dentro del período de pruebas

Veamos lo en el Olly

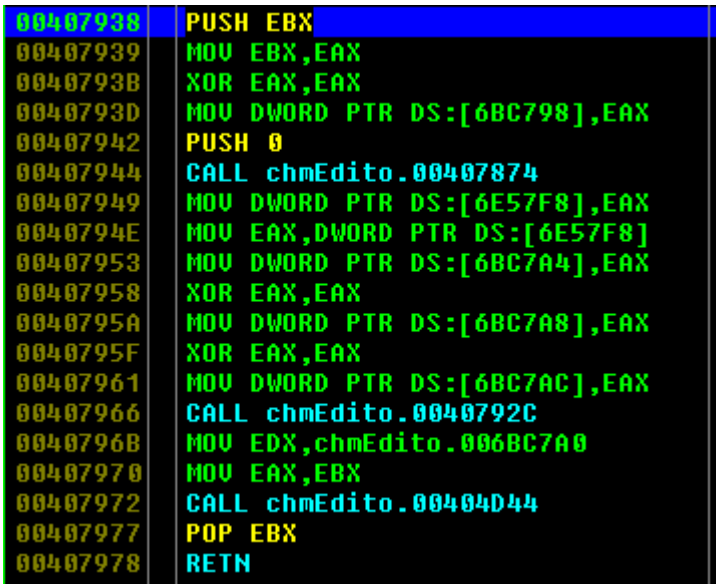
Bueno esto nos confirma que es Asprotect, tiene la estructura típica estructura del PE.

Este Asprotect es de los últimos y no responde al truco de las excepciones, lo más parecido es el tuto de Solid *Asprotect 2.1 - 2.2 SKE - peleando contra el mutante_por Solid teoría 1111* en él Solid toma el truco de poner un BP en una API que se llame cerca del OEP. Hay muchos métodos pero el de poner un BP en el ret de la API GetModuleHandleA en este caso es muy rápido, hagámoslo.

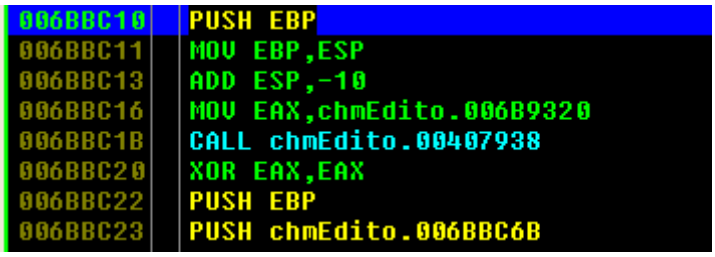
Y vamos pasando con F9 hasta ver algo como esto en la pila



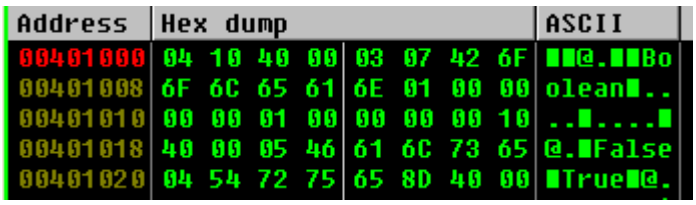
Como vemos esta llamada se realizo desde la sección de código vallamos a ver que tenemos.



Aparentemente tiene poca pinta de OEP, veamos la otra dirección la 6BBC20

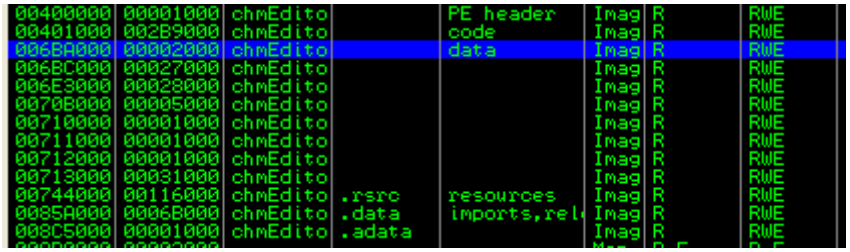


Esto es otra cosa y parece un Delphi, vemos como se encuentra la sección code.

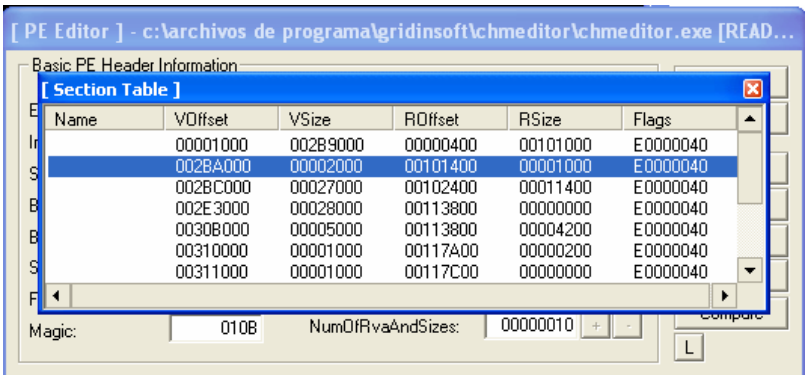


Pues si esto confirma que estamos ante un Delphi y que lo que parece el OEP es la dirección **6BBC10.**

Bueno hasta aquí parece todo correcto pero si nos da por comprobar en que sección se encuentra nuestro OEP tenemos una sorpresa.



Parece que esta en la sección data. Si os fijáis este packer ha dejado muchísimas secciones he inicialmente estaban todas con Lectura/Escritura/Ejecución, vamos a verlo con el LordPE



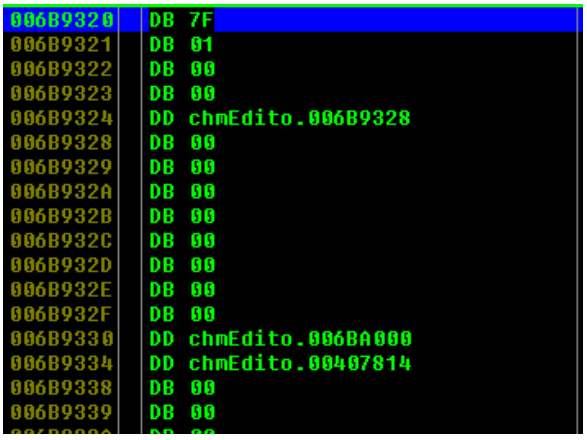
Podemos ver que muchas de las secciones no están en el disco físicamente, ya que tiene un RSize de 0, es decir solo existen en memoria.

Personalmente pienso que es un truco sucio del Asprotect para despistar, que haya subdivido la sección .code real en 2, pero bueno son simples suposiciones. Solid me llego a comentar otro punto que el pensaba que podía ser más OEP ya que pensaba que no estaba totalmente desempacado la sección de código donde yo lo tome.

Como esto es un Delphi según nuestro OEP tenemos que tener una instrucción del tipo:

mov eax, DIRECCION DE LA INIT TABLE

En este caso es mov eax, 006B9320. Vamos a revisar como anda



Como podemos ver esta totalmente destrozada.

¿Y la IAT como estará?

```
00407871   LEA EAX,DWORD PTR DS:[EAX]
00407874   CALL 00FE0004
00407879   DB DB
0040787A   MOV EAX,EAX
0040787C   JMP DWORD PTR DS:[70BDF0]      kernel32.LocalAlloc
00407882   MOV EAX,EAX
00407884   JMP DWORD PTR DS:[70BDEC]      kernel32.TlsGetValue
0040788A   MOV EAX,EAX
0040788C   JMP DWORD PTR DS:[70BDE8]      kernel32.TlsSetValue
00407892   MOV EAX,EAX
```

Podemos ir viendo los famosos CALL del Asprotect de los cuales Solid (entre otros) ha hecho unos cuantos script para Olly para su reparación.

Address	Value	Comment
0070BDB8	7C813559	kernel32.FindFirstFileA
0070BDBC	AAB64240	
0070BDC0	7C81CAA2	kernel32.ExitProcess
0070BDC4	C94E5CBE	
0070BDC8	7C81082F	kernel32.CreateThread
0070BDCC	7C80D293	kernel32.CompareStringA
0070BDD0	D556BCA2	
0070BDD4	7C862B8A	kernel32.UnhandledExceptionFilter
0070BDD8	7C947A40	ntdll.RtlUnwind
0070BDDC	7C81EAE1	kernel32.RaiseException
0070BDE0	EE1F0540	
0070BDE4	00000000	
0070BDE8	7C809BF5	kernel32.TlsSetValue
0070BDEC	7C809750	kernel32.TlsGetValue
0070BDF0	7C8099BD	kernel32.LocalAlloc
0070BDF4	59509FE6	
0070BDF8	00000000	
0070BDFC	77D21AD5	USER32.CreateWindowExW
0070BE00	77D2190B	USER32.CreateWindowExA
0070BE04	77D1C57E	USER32.WindowFromPoint

Por otra parte vemos que la IAT también está completamente destruida.

En este punto intenté con algunos script la reparación de la IAT pero sin éxito y teniendo en cuenta la cantidad de secciones que tiene me rondaba la cabeza que pudiera tener una VM, así que opte por olvidar el desempacado y centrarme en la comparación de tiempo.

Buscando la zona mágica

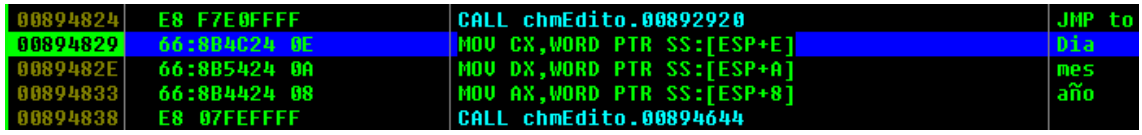
En este punto estuve un poco dando palos de ciego. Si vemos como arranca el programa aparece su Splash screen, luego la ventana principal del programa y un startup con los proyectos abiertos recientemente donde tenemos el cartel de UNREGISTER VERSION. Luego en un breve espacio de tiempo aparece nuestra NAG.

Con este proceso la verdad es que pensaba que la NAG pertenecía al programa y que por lo tanto la comprobación del tiempo aparecería después de nuestro OEP.

Con este razonamiento una vez parado en el OEP puse un parte de HE (hardware breakpoint) en GetLocalTime y GetSystemTime, pero sin resultado alguno. Siendo un Delphi llege a pensar que usaran algún método de la VCL que terminara tomando el tiempo de alguna otra forma.

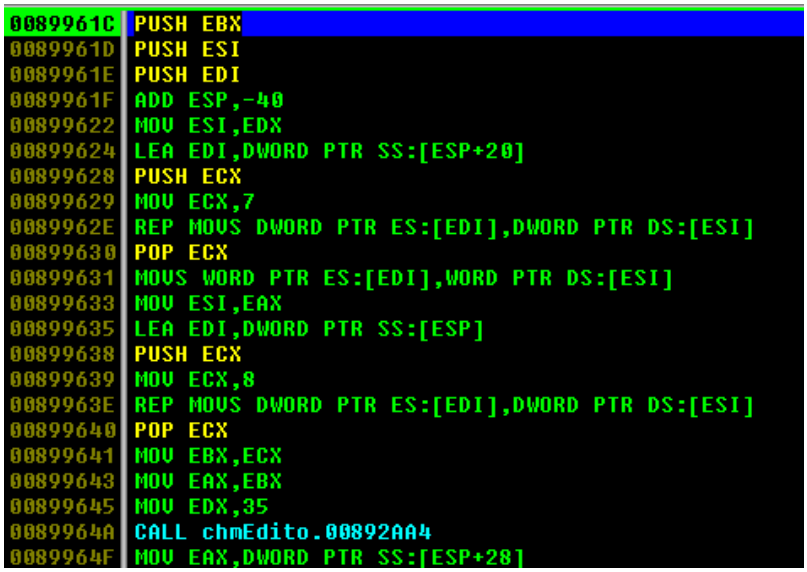
Nada más lego de la realidad en unos de los reinicios al partir del EP me para aquí.

Es decir llamado desde el código de Asprotect. Vamos a seguirlo. Para ello visualizamos la dirección donde guardará los datos. En el commandline hacemos
d 0012FEF0 y saltamos la API hasta llegar aquí

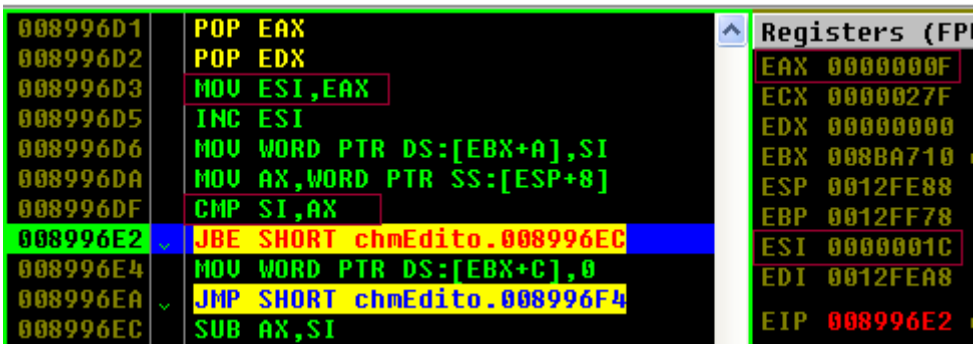


La idea es seguirle la pista.

Tras muchísimas vueltas y operaciones llegamos a la siguiente dirección



Este es el principio de la rutina que se encarga del control del tiempo del Asprotect.

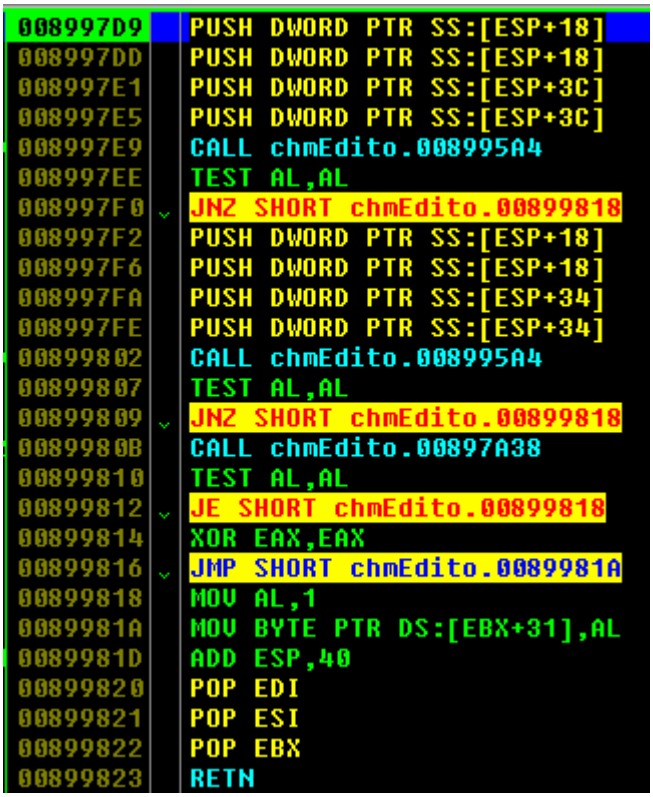


En este punto se comprueban los valores AX con SI.

En AX tenemos el valor F (15 en decimal) y en SI 1C (28 dec). En este punto modifique el reloj para que aparezca como terminado el tiempo de prueba.

En ese caso se ejecuta la instrucción 8996E4 poniendo una variable a 0.

Si provocamos el salto seguimos traceando y llegamos a esta zona



En esta zona se comprueba que el reloj no se haya atrasado entre otras cosas, si todo va bien debe de llegar a la zona 899814 XOR EAX,EAX.

En definitiva tenemos 2 cambios a realizar para que se crea que siempre le quedan los 15 días de evaluación.

Modificamos la dirección **8996D3 por XOR SI,SI**
Y el salto de **8997F0 lo cambiamos por un JMP 899814**
Con esos simples cambios ya saltamos el control de tiempo.

El control de CRC

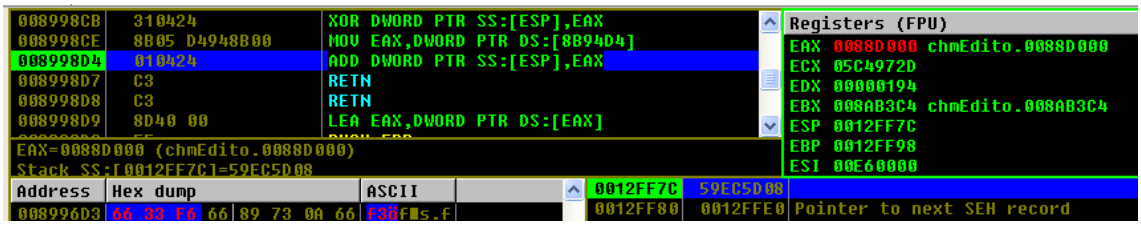
Ahora como siempre viene un pero. Si hacemos los cambios en caliente con el Olly y damos a RUN el programa termina en una excepción.

En antiguos Asprotect el CRC se comprobaba abriendo el fichero original y comprobando con lo que hay en memoria, en este caso solo trabaja con la memoria.

Vamos a poner un HW on Access sobre los bytes modificado de la dirección 8996D3 y veamos donde los lee y nos para aquí

En BL se tiene el primer byte (66)

Sigamos la ejecución traceando con cuidado.



Tras varias operaciones con los bytes de la zona llega a esta dirección en donde si nos fijamos en el ESP tenemos el numeraco de la operación, en EAX se da el valor de una dirección de Asprotect (a saber si es fijo o calculado). Al sumar los 2 debe de dar la dirección siguiente para seguir con la ejecución, como los datos no solo los correctos porque hicimos el parche se provoca la excepción.

Bueno esto tampoco parece que sea un gran problema, podemos intentar que los cambios vengan ya puestos con un inline parching, y luego al finalizar la ejecución intentar reponer los valores originales antes de que pase al CRC y listo.

Así que lo siguiente que queda sería seguir el tuto de Aboslom1 *Inline Patch Asprotect como si fuese un UPX*.

La técnica es sencilla pero muy laboriosa, pones en la sección un HW on WRITE y vamos tirando del hilo hasta dar con la zona dentro del programa donde se encuentra los bytes que seguramente estén encriptados.

Personalmente buscando me perdí y tras 1 hora, y viendo los cambios a realizar opté por probar el hacer un Loader con mi componente DebugerCTRL en MASM ya que este Asprotect al menos no tiene checkeo de debuger jejeje.

El Loader

Como loader la cosa es aun más sencilla, lo que haremos es poner 2 HW on execute, una en la dirección 8996D3h en el mov esi,eax y cambiaremos el valor de EAX a 0 (NO TOCAMOS EL CODIGO SOLO EL VALOR DEL REGISTRO).

Por último otro HW on execute en la dirección 8997F0h el salto y cambiamos el EIP a la dirección 899814h, es decir al XOR EAX,EAX

Para empezar el código debe de tener los siguientes incluyes

```
.386
.model flat, stdcall  ;32 bit memory model
option casemap :none  ;case sensitive

include Loader.inc

include ..\Comunes\Funciones.inc        ; <= Requerido por el componente
include ..\Comunes\ListaDoble.inc       ; <= Requerido por el componente
include ..\Comunes\DebugerCTRL.inc  ; <= EL componente
.code

start:

        invoke GetModuleHandle,NULL
        mov    hInstance,eax

        invoke InitCommonControls
        invoke DialogBoxParam,hInstance,IDD_DIALOG1,NULL,addr DlgProc,NULL
        invoke ExitProcess,0

;#########################################################################

DlgProc proc hWin:HWND,uMsg:UINT,wParam:WPARAM,lParam:LPARAM
        LOCAL MyD:DWORD
        mov                     eax,uMsg
        .if eax==WM_INITDIALOG
                invoke ShowWindowAsync,hWin,SW_HIDE
                invoke CrearDepurador,addr szVictima,NULL,NULL,hWin,NULL,NULL
                .if eax != NULL
                    mov hDBG,eax   ; <= Se crea el Depurador bien
                .else
                    invoke MessageBox,NULL,NULL,NULL,MB_OK ; <= Error creando el depurador
                .endif
        .elseif eax == WM_CREATEPROCESS_DBG  ; 1º Mensaje enviado por el depurador
                invoke GetProcessEntryPoint,hDBG ; Obtenemos la dirección del EP
                invoke SetHWBreakPointDBG,hDBG,eax,HW_EJECUCION,HW_BYTE  ; Ponemos un HW on Execute en el EP
        .elseif eax == WM_HW_BREAKPOINT_DBG ; Mensaje que se genera cuando se alcanza un HW
                invoke GetProcessEntryPoint,hDBG
                mov ebx,wParam
                .if ebx == eax
                        ; Estamos en el EP
                        invoke BorrarHWBP_DBG,hDBG,eax ; borramos el HW del EP
                        invoke SetHWBreakPointDBG,hDBG,8996D3h,HW_EJECUCION,HW_BYTE ; Cambiar EAX a 0
                        invoke SetHWBreakPointDBG,hDBG,8997F0h,HW_EJECUCION,HW_BYTE ; Cambiar el salto
                .elseif ebx ==  8996D3h
                        invoke BorrarHWBP_DBG,hDBG,8996D3h
                        invoke Set_EAX,hDBG,0
                .elseif ebx == 8997F0h
                        invoke BorrarHWBP_DBG,hDBG,8997F0h
                        invoke Set_EIP,hDBG,899814h

                .endif
        .elseif eax == WM_EXITPROCESS_DBG ; Se cerró el programa víctima
                invoke PostMessage,hWin,WM_CLOSE,0,0 ; Importante que sea PostMessage para dar tiempo a cerrar el debugger
        .elseif eax==WM_CLOSE ; se ha cerrado el Loader
                .if hDBG != NULL
                        invoke CerrarDepurador,hDBG
                .endif
                invoke EndDialog,hWin,0
        .else
                mov                     eax,FALSE
                ret
        .endif
        mov                     eax,TRUE
        ret
DlgProc endp
end start
```

Fichero Loader.inc

```
include advapi32.inc ; <= requerido por el control para escalar privilegios
include windows.inc
include kernel32.inc
include user32.inc
include Comctl32.inc
include shell32.inc

includelib ADVAPI32.LIB
includelib kernel32.lib
includelib user32.lib
includelib Comctl32.lib
includelib shell32.lib

DlgProc                         PROTO     :HWND,:UINT,:WPARAM,:LPARAM

.const

IDD_DIALOG1                             equ 101

IDC_BTN1 equ                    1001

;#########################################################################

.data

szVictima    db "victima.exe",0  ;<=Nombre del proceso víctima

hDBG         dd 0
.data?

hInstance                       dd ?

;#########################################################################
```

Sencillo verdad, pues esto fue todo.

Seguramente podamos sacar una firma a la zona del control de tiempo y casi seguro que los Asprotect de esta versión caen igual sin requerir desempacado jejeje.

Bueno hasta la próxima.