DIFÍCIL PERO ARREGLÁNDONOS

(LECCION CINCUENTA Y UNO)

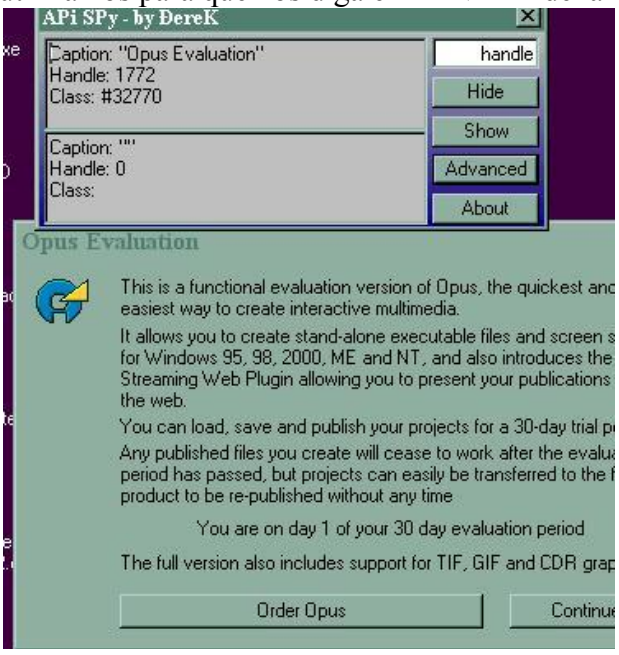
De aquí se baja la victima de hoy se llama ILLUMINATUS OPUS 2001 la versión común, no la PRO, es una versión de evaluación, tiene ciertas limitaciones ya que no se puede registrar, pero bueno, practicaremos con él a ver qué podemos hacer, esto dice en la pagina

http://www.digitalworkshop.com/downloads/evaluation.shtml

Multimedia program that combines video, audio, animation, words and pictures into stunning interactive publications.

- EXPIRE after 30 days - ANY PUBLICATIONS CREATED WITH THE EVALUATION VERSION WILL ALSO CEASE TO FUNCTION WHEN YOUR EVALUATION PERIOD ENDS.
- These publications will then only be recoverable with a full version of the program. No support GIF, TIFF or CDR files.

Bueno aquí dice que el programa expira a los 30 días y las publicaciones que hagas también, trataremos de que ambas sigan funcionando.
El programa tiene un ejecutable chiquito llamado OPUS.exe que esta comprimido con aspack que con el Caspr se puede descomprimir y reemplazar pero no afecta en nada ya que el programa no realiza nada importante desde el ejecutable.
El problema es que la ventana que dice los días que van de la evaluación, no es ni messageboxa ni dialogboxparama, ni para con drawtexta ni con nada conocido usaremos otra técnica para ver donde se cierra esta ventana desde el programa principal.
Arrancamos el programa y una vez que aparece la nag esa donde dice los días que nos quedan, arrancamos algún programa analizador de ventanas como el API-SPY o WINDOWSE, que lo utilizamos para que nos diga el HANDLE de la ventana



Como se ve en la imagen la ventana del API-SPY me da la información de la ventana sobre donde coloco el cursor, y lo coloco sobre la ventana a estudiar, la parte principal es la que dice el texto OPUS EVALUACIÓN y allí vemos que el HANDLE en mi caso es 1172.
Es importante que el cursor este bien sobre el texto principal, ya que si lo pongo en los botones, salen otros resultados.
Bueno lo que importa es ver el handle de la ventana, para mi 1172 para cada uno variara el valor y cada vez que ejecute el programa será distinto este valor.
El APY SPY te da el handle en decimal el WINDOWSE en hexa, para verificar, con la calculadora pasamos a hexa el valor 1172 y es 06ec.
Si con la ventana todavía activa entramos al softice con CTRL+D y allí teclearemos.

HWND

Con esto nos sale un listado de la información de todos los handles que están en ese momento.

Si con ENTER voy bajando hasta que en la columna OWNER comiencen a aparecer los del proceso OPUS vamos a encontrar allí lo mismo que nos dijo el API SPY el handle en hexa para mi es 06ec de la ventana esta.

Bueno la ventana todavía esta activa existe un breakpoint para que caiga en el SOFTICE cuando se cierre la ventana

BMSG handle Wm_Destroy

En mi caso

BMSG 06ec Wm_Destroy

Con CTRL+D vuelvo a WINDOWS y al hacer click en el BOTON CONTINUE y al desaparecer la ventana cae en el SOFTICE.

Hay que volver con F12 varias veces hasta que llegamos en este caso al dll ILMEDITMAIN, que aparece en la línea verde, ya que el ejecutable OPUS aquí casi no hace nada todo lo hacen las dll.

Caemos en

BDB932 Mov eax,[ebp-00dc]

Y justo en la sentencia anterior hay un Call EAX que seguro es la ejecución del cartel entero, para probar eso pongamos un BPX bdb930 y arranquemos el programa de nuevo, vemos que justo para en el BPX y no salió aun la ventana molesta (podemos mirar con F4) si hacemos F10 y ejecutamos el CALL aparece la ventana y al continuar caemos en la sentencia posterior al CALL o sea en BDB932.

Si ponemos un BPX un poquito antes en BDB8E8, cuando para allí, vamos traceando con F10, y ya que en la ventana aparece la cantidad de días que lleva de evaluación, por aquí debe cargar ese valor y eso ocurre en.

BdB8E8 justo a EAX va la cantidad de días de evaluación menos uno, ya que unas sentencias más adelante al hacer INC EAX queda justo en EAX la cantidad de días transcurridos, si modificamos en BDB8E8 para que reemplacemos la sentencia original por XOR EAX, EAX que pone EAX a cero , luego al incrementar EAX, queda siempre el valor de los días de evaluación en uno, a pesar de adelantar el reloj, siempre dice que va un solo día de evaluación.
Este es el primer paso, el segundo es hacer una publicación y ver que pasa, si la cargo con el programa en la fecha actual, la abre sin problemas, pero si adelantamos el reloj nos dice que la publicación ha expirado y la cierra.
Ya que este cartel es un BPX messageboxa, ponemos allí un breakpoint, y cuando adelantamos el reloj, para en ese break, hacemos F12, aceptamos la ventana y volvemos a caer en el SOFTICE, allí nos damos cuenta cuando volvemos al ILMEDITMAIN que arriba del lugar de donde caemos hay un salto que esquiva el cartel, y está en

C08AC9 jae C08B53

si reemplazamos ese JAE por un JMP no aparecerá el cartel que cierra las publicaciones cuando vencen, y es cierto, el problema es que cuando copiamos las cadenas para modificar el ILMEDITMAIN.dll.

EL REEMPLAZO POR **XOR EAX,EAX** EN BDB8E8 la cadena es

8B8520FFFFFFBB8051010033D2 y hay que reemplazar las negritas por
33C090909090

El salto que hay que reemplazar en c08ac9 la cadena es

0F83840000008D4DC0 reemplazar las negritas por
E98500000090

El problema es que cuando buscamos estas cadenas en el ULTRAEDIT no aparecen. por lo que el dll esta comprimido, si lo vemos con el PEEDITOR vemos que en los nombres de las secciones una se llama ASPACK, así que con el CASPR descompactamos el dll, ahora si buscamos las cadenas que aparecen, realizamos los cambios y funcionaaaaa. Con eso el programa sigue funcionando sin vencer y las publicaciones también.

Ricardo Narvaja

OTRA VEZ P-CODE PERO SIN EL DEBUGGER

(LECCION CINCUENTA Y DOS)

Hoy crackearemos un programa que tiene una parte en P-CODE y una parte normal de VISUAL BASIC. El programa se llama DIGITAL PEN 1.0 y se baja de

http://camtech2000.net/

Es un programita resencillo para hacer STATIONERYS o sea animaciones para el OUTLOOK EXPRESS.

Lo primero que dice el tipo cuando funciona es que vence en 10 usos o 10 días cualquiera de las dos cosas.

Bueno podemos analizar con el TECHFACTS 98 a ver cada vez que lo usamos que cambios hace cada vez que se ejecuta.

Arrancamos el TECHTACTS 98 y vamos a la pestaña TOOLS y allí a WATCH SYSTEM y donde dice RUN THIS PROGRAM buscamos el ejecutable del DIGITAL PEN y lo arrancamos desde abajo donde dice GO.

Luego de que hace un escaneo de todo el sistema arranca el programa se ejecuta lo abrimos para que consuma una vez y luego lo cerramos y el TECHFACTS hace otro escaneo para ver qué cambios hizo el programa y aquí está el resultado.

Registry key values changed: (12)
HKEY_CLASSES_ROOT\NSPlay32s
Value "GamesAvail": from "4" to "5"
HKEY_USERS\.DEFAULT\Software\Microsoft\HTMLUrlls
Value "Visited": from "4" to "5"
HKEY_USERS\.DEFAULT\Software\Microsoft\Windows\CurrentVersion\Explorer\StreamMRU
Value "MRUListEx": binary data changed
HKEY_USERS\.DEFAULT\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections
Value "SavedLegacySettings": binary data changed
HKEY_LOCAL_MACHINE\Software\CLASSES\NSPlay32s
Value "GamesAvail": from "4" to "5"
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Msversions
Value "WinKeys": from "4" to "5"

Bueno estas claves son donde guarda la cantidad de veces que va usando, si ahora exportamos estas claves y cada vez que usamos el programa ejecutamos los archivitos del registro para que vuelva la cuenta de la cantidad de veces usadas para atrás, con eso lo mantenemos funcionando mientras lo crackeamos.

También hay una clave del registro esta la vi. Con el REGMON y es cuando el programa expira cambia este 1 y no funciona mas

[HKEY_CURRENT_USER\Software\VB and VBA Program Settings\Digital Pen\Logged]
"Expired"="1"

Aquí también conviene exportar esta clave por si nos vence volverla a inyectar en el registro para hacer que funcione nuevamente.

Bueno si probamos tratar de verlo con el P-CODE debugger con que crackeamos el CUENTAPASOS, y vemos que arranca el programa pero no para en el debugger y se cierra sin poder hacer nada, por lo que no lo podemos utilizar.

Si descomprimimos el programa con el WDASM de VISUAL BASIC vemos que salen las STRING REFERENCES bien.

El SOFTICE se me cuelga cuando quiero usarlo con este programa así que usando el TRW2000 veremos que podemos hacer.

:00449CD2 8B7508 mov esi, dword ptr [ebp+08]

* Possible StringData Ref from Code Obj **->"This program will stop functioning "**
->"after 10 uses or 10 days."
|
:00449CD5 6824DA4000 push 0040DA24

Aquí vemos una parte del WDASM que aparece en el cartel molesto cuando arranca y dice que la versión registrada no muestra este cartelito así que trataremos de esquivarlo.

Un poco más arriba hay un salto pero si paro en el TRW2000 y lo invierto da error. Más arriba en el WDASM está este otro cartel

- **Possible StringData Ref from Code Obj ->"Expired"**

O sea que por aquí no hay que pasar si estas registrado

Sigo más arriba en el WDASM y siguen los carteles que no deben aparecer en la versión registrada.

:00449AFD 8B3D50104000 mov edi, dword ptr [00401050]

* Possible StringData Ref from Code Obj **->"This program has timed out and "**
->"will not function anymore."

Más arriba…

- Possible StringData Ref from Code Obj ->"**Program Timed Out"**

O sea que por aquí no hay que andar, siguiendo para arriba vemos que vuelven a salir mensajes de TIME OUT y EXPIRED y también los nombres de las claves del registro que guardan la cantidad de veces que se usaron, que en la versión registrada no debe utilizar.

Sigamos mas arriba

*** Possible StringData Ref from Code Obj ->"GamesAvail"**

```
                        |
:004499FC BAB8D64000          mov edx, 0040D6B8
:00449A01 8D4DE0              lea ecx, dword ptr [ebp-20]
:00449A04 FFD6                call esi
```

- **Possible StringData Ref from Code Obj ->"HKEY_CLASSES_ROOT\NSPlay32s"**

NSPlay32s es una clave del registro donde guarda la cantidad de veces, seguimos más arriba:

Más arriba siguen apareciendo claves del registro de la versión de prueba y mas EXPIRED, etc.

La conclusión es que toda esta rutina debe ser para la versión NO REGISTRADA así que vamos a seguir bastante hacia arriba a ver donde comienza.

La rutina comienza en

```
:00448A50 55                  push ebp
:00448A51 8BEC                mov ebp, esp
:00448A53 83EC0C              sub esp, 0000000C
```

Ya que hay una REFERENCE que dice que el programa viene de un salto en 4056a6 y más arriba de esto hay un RET que parece ser otra cosa.

Veamos en 4056a6 que hay

Aquí hay una zona con una tablita de saltos

```
:0040568C E90F310400          jmp 004487A0
:00405691 816C240463000000    sub dword ptr [esp+04], 00000063
:00405699 E952320400          jmp 004488F0
:0040569E 816C240433000000    sub dword ptr [esp+04], 00000033
:004056A6 E9A5330400          jmp 00448A50
:004056AB 816C240433000000    sub dword ptr [esp+04], 00000033
:004056B3 E958470400          jmp 00449E10
```

```
:004056B8 816C24045F000000      sub dword ptr [esp+04], 0000005F
:004056C0 E9AB480400            jmp 00449F70
:004056C5 816C240437000000      sub dword ptr [esp+04], 00000037
:004056CD E94E490400            jmp 0044A020
:004056D2 816C24045B000000      sub dword ptr [esp+04], 0000005B
:004056DA E9F1490400            jmp 0044A0D0
```

Si cargamos el programa con el TRW2000 vemos que traceando con T enseguida entramos en el dll de VISUALBASIC MSVBVM60.dll que es la parte de P-CODE y no podemos volver al ejecutable por más que hagamos F12 quizás porque la parte de P-CODE termina en un JMP donde no paramos con F12.

Posiblemente esa tablita de JMPs sea donde el programa salta de partes de P-CODE a VISUAL COMUN, podemos poner en todos esos JMPs unos breakpoints a ver qué pasa.

Arrancamos el TRW2000 y ponemos BPX en todos los JMPs de esa listita.

la primera vez que para lo hace en 4056a6 justo el salto que va a la parte de no registrado, seguimos ejecutando el programa y sale el cartel maldito y luego vuelve a parar en 40568c y luego arranca el programa.

Ahora que pasaría si en 4056a6 que es cuando va a ir a la parte de no registrado cambiamos el salto al de arranque del programa que está en 40568c, saltearía la parte de no registrado y arrancaría directo en teoría, probemos.

Pongo un BPX 4056a6 donde está el JMP 448a50 y cuando para allí hago A y ENTER y escribo el otro salto

JMP 40568c para que arranque el programa directo le doy a X y FUNCIONNNNAAA y no aparece la maldita ventana de desregistrado.

El cambio con el ULTRAEDIT es

E9A5330400 816C240433000000 CAMBIAR POR
E9F5300400 816C240433000000

y listo no vence mas ni lee las claves donde guarda la cantidad de veces ni las modifica nunca más lo que se puede ver con el regmon o el techfacts.

Ricardo Narvaja

UNA MOCHILA PESADA

(LECCION CINCUENTA Y TRES)

Hoy vamos a tratar un programa bastante completo llamado Cimagrafi 6.02 que se puede bajar de mi FTP (pedir dirección del mismo a ricnar22@millic.com.ar) o también lo voy a subir a mi kturn.
Allí hay una carpeta que hay que descomprimir y NO INSTALAR a partir del SETUP que hay allí.
Porque digo que no usen el SETUP, bueno, el programa es de esos que funcionan con una mochila o aparatito que se conecta a un puerto de la computadora, por supuesto yo no tuve el gusto de conocer dicho aparato, pero me llego el programa y la idea es tratar de hacerlo funcionar sin poseer ese chiche.
La mochila que usa este programa es la llamada mochila Hasp y hay varios tratados de gente que sabe de cómo emularla o como atacarla, nosotros no vamos a hacer eso, quizás porque este humilde cracker no esta a la altura de semejante desafío, solo trataremos de hacer funcionar las aplicaciones que tiene allí sin la dichosa mochila y punto.
Dentro de esa carpeta una vez descomprimida están los ejecutables de las aplicaciones que vamos a tratar de hacer funcionar, GRAFICAD.exe es una y la primera que vamos a estudiar, las otras P2p.exe , TRACE.exe y MIL.exe la protección es similar y inclusive la cadena encontrada es la misma en todas estas aplicaciones y se puede reemplazar igual en todas.
Bueno si alguno cometió el error de ejecutar SETUP.exe a pesar de que se lo advertí lo que va a hacer es instalar el HASP driver que si detecta la presencia del Softice no solo te impide que arranque sino que te renombra filas del mismo para que se cuelgue y funciona mal.
Bueno puede eliminarlo usando el editor del registro y borrando la clave

[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\HASP95]

Con eso ya eliminamos el driver molesto ese. Así que sin instalar el programa veremos qué podemos hacer y si podemos hacer que arranque.
Tomamos el archivo GRAFICAD.exe que va a ser nuestro objetivo, lo ejecutamos y zas PLUG NOT FOUND, ya nos aparece un cartelito que nos cierra el programa.
Bueno puedo poner un **BPX messageboxa** y para allí pero a pesar de que vuelvo al ejecutable este ya se cierra y no obtengo mucha información de esto.
Si voy traceando hasta que aparece la ventana de PLUG NOT FOUND y cuando llego a un CALL que al ejecutarlo aparece el cartel, y voy anotando ese CALL y luego pongo un BPX allí y paro en el SOFTICE y entro con T y sigo traceando con F10 y así sucesivamente voy a llegar a un momento en que este en este sector.

```
:00490C04 E8332CF7FF          call 0040383C
:00490C09 A1D4485700            mov eax, dword ptr [005748D4]
```

Al ejecutar ese CALL 40383C sale el cartel maldito y se cierra el programa, no pasando ya por la sentencia subsiguiente 490c09 lo que puede comprobarse poniendo un BPX 490c09 y viendo que no para allí y se cierra el programa igual.

Bueno un buen primer paso sería analizar dentro de ese CALL 40383c.

```
:0040383C 5A                pop edx
:0040383D 54                push esp
:0040383E 55                push ebp
:0040383F 57                push edi
:00403840 56                push esi
:00403841 53                push ebx
:00403842 50                push eax
:00403843 52                push edx
:00403844 54                push esp
:00403845 6A07              push 00000007
:00403847 6A01              push 00000001
:00403849 68DEFAED0E   push 0EEDFADE
:0040384E 52                push edx

* Reference To: kernel32.RaiseException, Ord:0000h
                            |
:0040384F E91CDAFFFF    Jmp 00401270
:00403854 C3                ret
```

Bueno no vemos mucho aquí solo que pone algunos valores en los registros y luego hace un JMP a Raiseexception que no sé exactamente que hace pero no me gusta nada, probemos que pasa si en 40383c ponemos directamente un RET y evitamos que siga por este camino que nos lleva al cartel maldito y a cerrarse el programa.
Hagamos
A 40383c ENTER
y escribamos RET

A ver que pasa
Luego salgamos con Escape y sigamos ejecutando el programa con X.
ARRANCAAAA
No se si en forma perfecta pero sale el Cartel de Cimagrafi y la pantalla principal y las herramientas para dibujar y al parecer funcionan bien, si hay algún error se arregla individualmente pero yo no encontré ninguno aun.

Bueno la cadena que debemos buscar en el ULTRAEDIT para reemplazar es
5A 54 55 57 56 53 50 52 54 6A 07 6A 01 68 DE

Y ALLI SOLO REEMPLAZAR EL **54** POR **C3** QUE CORRESPONDE A LA SENTENCIA RET.
Lo mas divertido es que la misma cadena la podemos encontrar en los otros ejecutables y reemplazar el 54 por el c3 y hacer funcionar todos.
Ojo si la buscamos en el Wdasm la posición de memoria en uno y otro programa no es la misma, pero si buscamos en el ULTRAEDIT la cadena esta aparecerá y solo modificando el 54 por c3 arrancaran perfectamente sea el TRACE, el P2p, El MIL, el Graficad etc.

Bueno ahora ya que arranco podemos registrarlo si hacemos clic en ABOUT vemos que dice DEMO que está entre las STRINGS REFERENCES del Wdasm para DELPHI ya que este programa está hecho en DELPHI.

Esta parte es la del mensaje de DEMO

```
:00503DCC E85BD7F8FF      call 0049152C
:00503DD1 85C0             test eax, eax
:00503DD3 7C4B             jl 00503E20
:00503DD5 8D55D8           lea edx, dword ptr [ebp-28]
:00503DD8 8BB3D0020000     mov esi, dword ptr [ebx+000002D0]
:00503DDE 8BC6             mov eax, esi
:00503DE0 E8FBF5F2FF       call 004333E0
:00503DE5 FF75D8           push [ebp-28]
```

- **Possible StringData Ref from Code Obj ->" DEMO (More "**

Ahí vemos el cartel de DEMO y un poco antes un CALL 49152c que según el valor de EAX que sale aparece el cartel de DEMO o no.
Si entramos dentro del CALL 49152c

```
:0049152C E873FEFFFF       call 004913A4
:00491531 0FBE05C3485700     movsx eax, byte ptr [005748C3]
:00491538 C3               ret
```

Vemos que entra dentro de otro CALL 4913a4 veamos allí dentro. Bueno allí empieza una larga parte que es donde decide si esta registrado o no.

```
:004913A4 53               push ebx
:004913A5 BBAC485700       mov ebx, 005748AC
:004913AA C6431000         mov [ebx+10], 00
:004913AE C6431100         mov [ebx+11], 00
:004913B2 C6431200         mov [ebx+12], 00
:004913B6 C6431300         mov [ebx+13], 00
:004913BA C6431400         mov [ebx+14], 00
:004913BE C6431500         mov [ebx+15], 00
:004913C2 C6431600         mov [ebx+16], 00
:004913C6 C6431700         mov [ebx+17], 00
:004913CA C6431800         mov [ebx+18], 00
:004913CE C6431900         mov [ebx+19], 00
:004913D2 C6431A00         mov [ebx+1A], 00
:004913D6 C6431B00         mov [ebx+1B], 00
:004913DA 8B5008           mov edx, dword ptr [eax+08]
```

Bueno sigue y es mucho más larga pero esos son los valores que siempre testea más adelante par a ver si esta registrado o no, me refiero a EBX+10, EBX+11 y asi sucesivamente.
Probé cambiar estos valores por 01 y no salió registrado, probé luego cambiarlos por FF uno por uno y desapareció el CARTEL DEMO.
O sea que deberíamos buscar esta cadena en el ULTRAEDIT

C6431000
C6431100
C6431200
C6431300
….
……
hasta
C6431B00

y asi sucesivamente y cambiar los 00 por FF y listo queda registrado.

Quedaria

C64310FF
C64311FF
C64312FF
C64313FF

Seguir cambiando los 00 por FF hasta

C6431BFF

Y listo el programa funciona bastante bien, quizás halla que corregir algún error suelto que aparezca pero ya el hecho de que funcione sin la mochila y lo podamos registrar es un gran avance, luego los retoques se pueden ir hacienda a medida que se van encontrando.

Ricardo Narvaja

ROBADA DE MI AMIGO SILVER STORM. Y DEL MISMO MARMOTA (ESPERO QUE ME PERDONEN JAJA)

(LECCION CINCUENTA Y CUATRO)

Este tute está basado en uno de mi amigo SILVER STORM y otro de MARMOTA que lo hizo ya que quería demostrar las utilidades del nuevo gestor de ventanas de Marmota WINGESTOR que está en mi FTP para bajar y que mejor que estos tutes.
El primero es el ya famoso ULTRAEDIT, como hacerlo funcionar sin cambiar ni una coma. (Con el WINGESTOR) Ojo esto hay programas donde se puede hacer otros que no pues estamos aprovechando un fallo en la implementación del programa ULTRA EDIT que obvio no todos los programas tienen, pero es muy lindo experimentar un poco y este tute basado de los de ambos es para hacer conocida la herramienta y más adelante ya haré uno mío propio.
El ULTRA EDIT lo conocemos todos casi todos ya lo usamos y lo tenemos registrado, pero así como viene sin registrar adelantamos el reloj un par de meses para vencerlo y aparece una ventana para registrarnos y poner el nombre y el número de registro, y si no ponemos el correcto y cancelamos se cierra el programa y chau.
Bueno arrancamos el WINGESTOR y el ULTRAEDIT y una vez que aparece la ventana para registrarnos no tocamos nada lo dejamos ahí.
Vamos al WINGESTOR y en la lista aparecen muchas ventanas activas enseguida por el texto ubicamos la ventana de registro del ULTRAEDIT buscamos una ventana con la clase #32770 y el texto (Autorización), y hacemos clic en la línea que encontramos en el WINGESTOR y entre las opciones que aparecen elegimos OCULTAR y listo desapareció, ahora nos queda la ventana principal que no recibe nada del teclado esta como muerta, a esa hay que habilitarla vamos otra vez al WINGESTOR y por el texto que tiene (UltraEdit-32 - [Editar1] encontramos la ventana principal y hacemos clic y elegimos habilitar y listo funciona a pesar de estar vencido.
Con el WINGESTOR nos podemos divertir haciendo clic en muchas opciones y ocultando cosas y activando otras que están inactivas.
Ese es el otro caso el del tute de Silver Storm

Ftp://ftp.novastor.com/pub/demos/BACKUP/WIN95/nsw9end.exe

Aquí copio del mismo SILVER STORM (perdón amigo)

> Hola.
> Este tute es facilito. Es convertir un programa en su propio
> Uno más con esta herramienta de marmota :)
>
> Ftp://ftp.novastor.com/pub/demos/BACKUP/WIN95/nsw9end.exe
>
> Se trata de nova backup la última versión :) bajamos el programa, lo instalamos.. entramos al programa... luego about.. comprar... y entramos donde nos dice convertir la demostración.. ahí nos pide un número de tarjeta.. ponemos el numero que queramos 10 dígitos... por ejemplo 1234567890 en donde nos pide código lo dejamos en blanco.. si observamos el botón de OK. esta desactivado...abrimos el gestor de ventanas y nos

vamos a la ventana que nos interesa (verificación de numero de acceso.) la abrimos.. y nos paramos sobre el OK. Damos dos click y le damos Habilitar.
Jejeje.. que genial.. se activa el botón de Ok..lo presionamos...y nos salen unas pantallas de agradecimientos... etc.. le decimos que ok a todo.. y retorna a la ventana de entrada de datos... entramos de nuevo donde dice "convertir la demostración" y jejeje... que les parece ... aparece el código ya puesto... y lo mejor del caso .. nos lo muestra.. el programa original sin ninguna modificación se convirtió en un keygen..
jeje ami me dio 579138HZKP… Gracias. Marmota and Jack.. Good Job ;)

Espero que por ser ladrón no me metan preso jaja. los invito con el WINGESTOR a probar y aunque no todo se pueda registrar tan fácil como estos dos ejemplos hay botones deshabilitados, barras que activar y mucha diversión por delante, prueben y verán.

Ahh me olvidaba si la ventana tiene asteriscos al hacer clic con el WINGESTOR aparece el significado de esos asteriscos y además podemos cambiar el texto de las ventanas en la memoria.

Bastante lindo para divertirse…

Ricardo Narvaja

FLASH FXP POR MR GANDALF
(GRACIAS)

(LECCION CINCUENTA Y CINCO)

Introducción Por Ricardo Narvaja: Dado que me he tomado el mes de enero de vacaciones después de 54 lecciones duras, y que a Mr Gandalf le pedí si podía reemplazarme durante el mes de enero haciendo algún tute o los que pudiera, bueno aquí está el tute hecho por el, bien prolijo (eso si que no lo aprendió de mi jaja), además que siempre cuando quiera colaborar con algún tute suyo, o de quien siga el curso y quiera colaborar bienvenido sea.

Programa: FlashFXP Version 1.3 Build 770.

Download: www.FlashFXP.com

Protección: Serial. Trial de 30 días. Nag screen.

Dificultad: Sí lo he hecho yo...

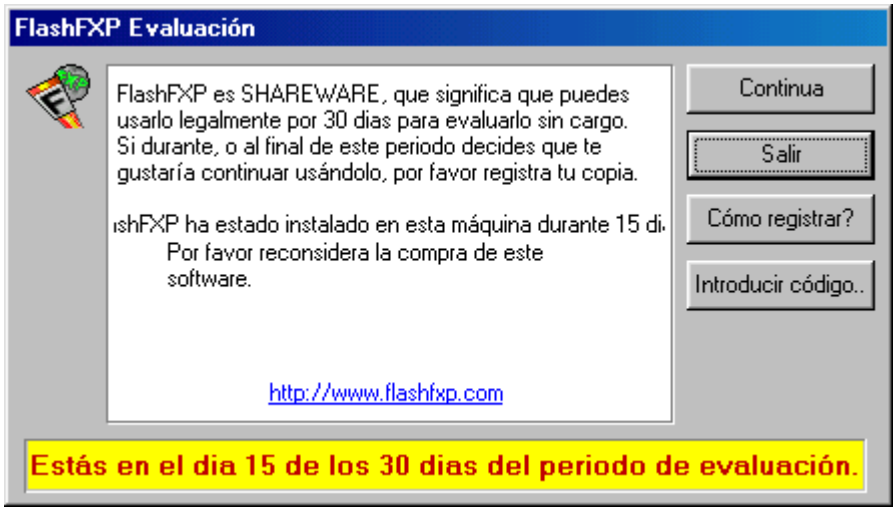
Herramientas: SoftICE 4.0. Languaje 2000. W32Dasm 8.93. Dede 2.50. Hiew 6.76. Advanced Registry Tracer.

Cracker: Mr Gandalf.

Fecha: 30/12/2001

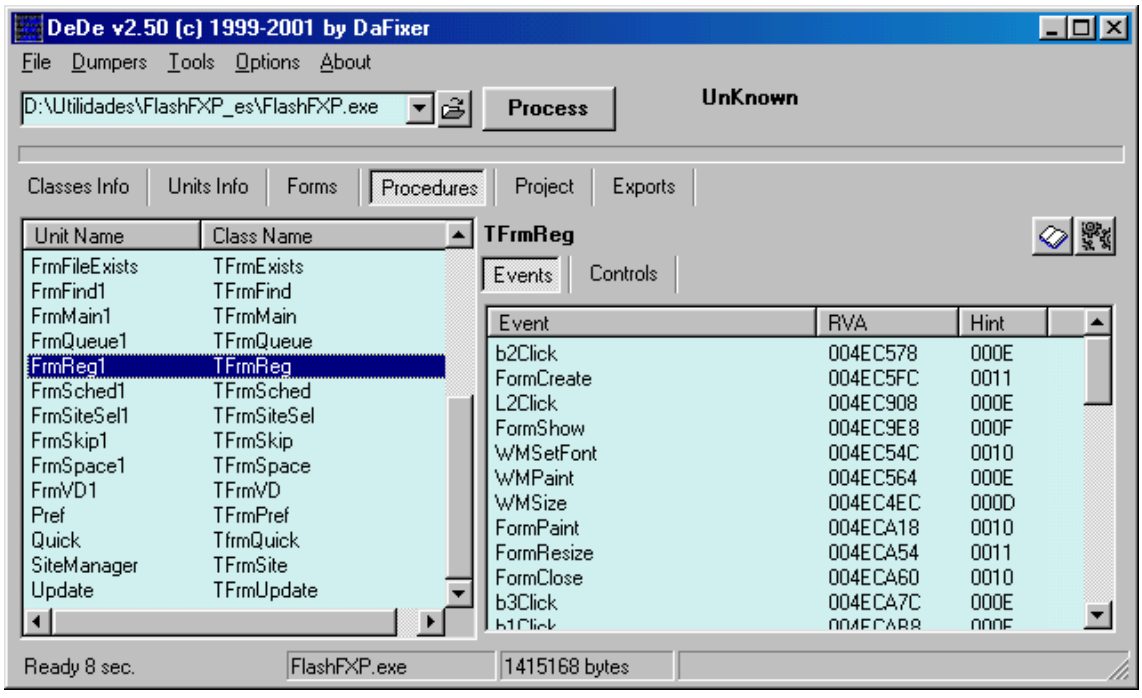
Introducción: Se trata de un conocido programa para FTP`s, que en la versión traducida por Pedator, es el que yo uso para chupar del disco duro de Ricardo, entre otros. Inicialmente me gustó más que el CuteFTP y, aunque este tenía su crack en Astalavista, empecé a usarlo. Conforme avanzaban los días se iba consumiendo el trial, así que me decidí a intentar el asalto. Primero conseguí parar el conteo basándome en un tutorial de Karpoff sobre el MemTurbo 1.5. Así que lo estuve utilizando durante mucho tiempo de esta manera, aunque al inicio del programa salía una molesta ventana recordándome que estaba en el día 0 (bendita inocencia). Finalmente deje el asunto aparcado en mi carpeta de cracks hasta que Ricardo Narvaja me instó a escribir un tutorial para llenar un mes de enero que se prometía silencioso. Volví sobre el tema de la nag y esta vez todo pareció mucho más fácil.

Desguaze: De lo primero que nos percatamos al iniciar el programa es de esa molesta nag que nos aparece diciéndonos los días que han pasado desde la instalación y que son 30 los que tenemos para evaluarlo. Esta nag tiene 4 botones y uno de ellos es para introducir un serial que parece será largo por que abre una gran caja de dialogo y por que ofrece la posibilidad de traerlo desde el portapapeles. Si pulsamos continuar entramos en la versión demo que es completamente funcional y que también tiene un apartadito desde donde podemos intentar introducir el serial, solo que esta vez si nos equivocamos ya no nos dejará más oportunidades de intentarlo durante esa misma sesión.



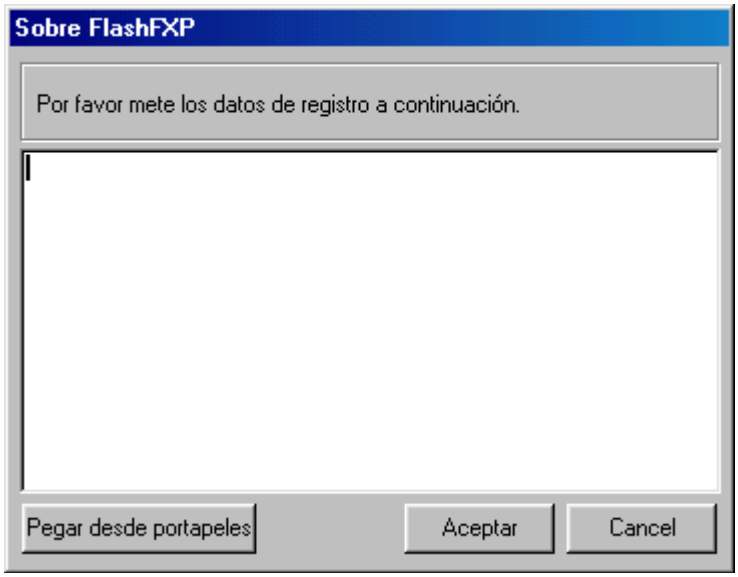
Lo primero es lo primero...

Antes que nada tenemos que saber que nos traemos entre manos. Con el Languaje 2000 vemos que es un programa no empacado escrito en Delphi. Desensambla muy bien con el Dede y vemos que hay un FrmReg1 con un b2click que tiene como referencia el texto "Introducir código". Si arrancamos con el Sice y ponemos un Bpx en 4EC578 nos sorprenderá comprobar que aquí no para. Los otros events tampoco parece que contengan información determinante.



Encontrar el serial...

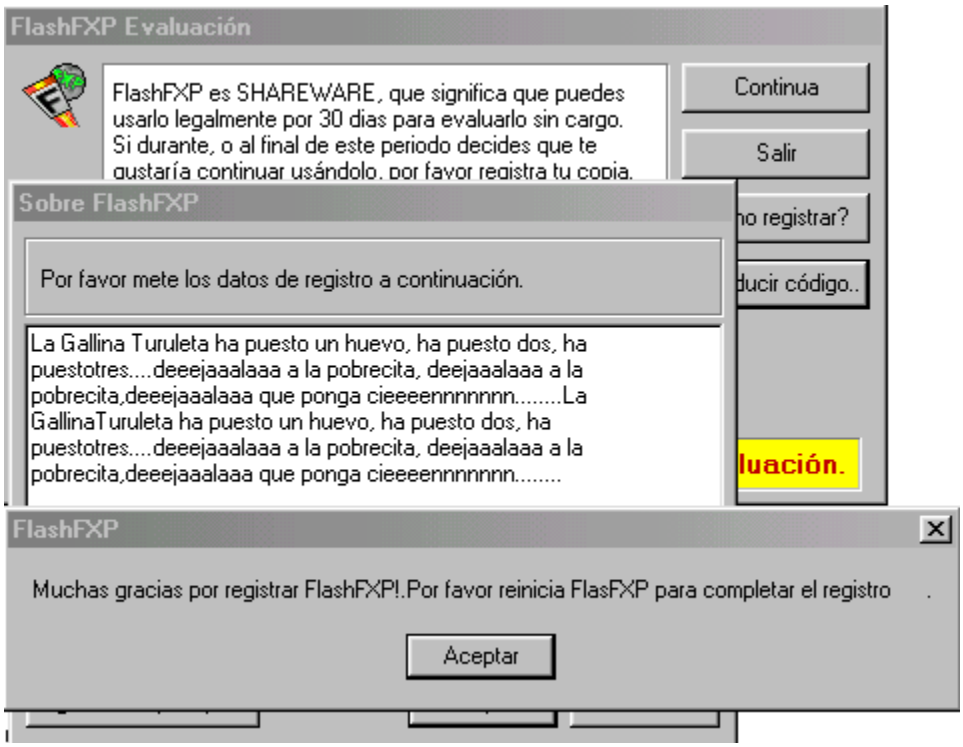
Ya no nos podemos aguantar más las ganas de entrar a saco con el Sice y metemos un Bpx Hmemcpy en la misma entrada del serial.



Después de tracear un poquito venimos aquí:

4BCC73 CMP EAX,8
4BCCBC CMP EAX,100
4BCCC7 JLE 4BCD9B <-La longitud de la clave debe ser mayor de 100!

Sí forzamos este salto a ejecutarse, o simplemente metemos una cadena mayor de 100, obtendremos un mensaje de éxito instándonos a reiniciar el programa para completar el registro.



(He de reconocer que soy fan acérrimo de Fofo, Miliki y Fofito... son de mi época, sniff, sniff)

Al reiniciarlo veremos una ventanita que dice "Registrarse Fallido" y vuelta al cartelito con los días que nos faltan...

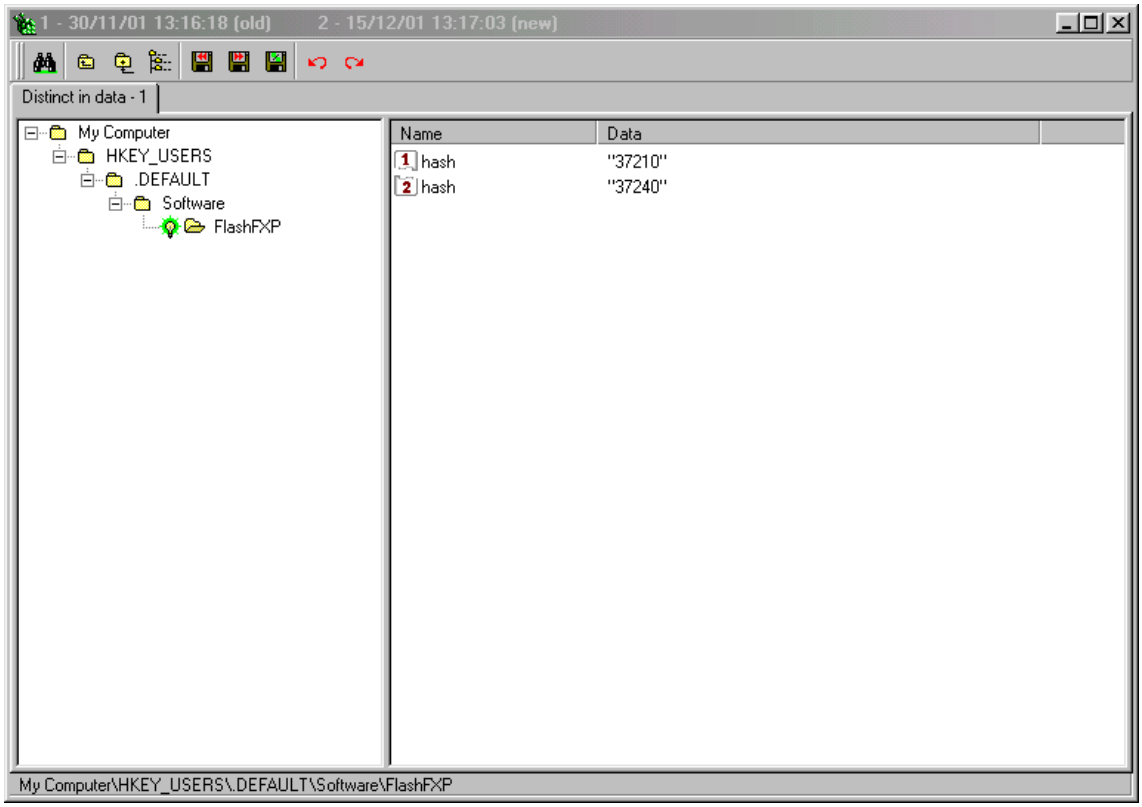
Detener el conteo...

Ya que no podemos engañar al programa con que nos hemos registrado intentemos hacerle creer que el tiempo no pasa para nosotros. Lo primero es averiguar que método utiliza el señor Charles de Wise (autor del programa) para llevar el conteo de los días. Las protecciones basadas en la evaluación de un programa por un tiempo durante el cual podemos utilizarlo plenamente y si nos gusta registrarnos, ofrecen una vulnerabilidad adicional que es precisamente el conteo del tiempo. Dicho conteo se realiza con una variedad de API`s como GetSystemTime, GetLocalTime, etc... Una vez que el programa comprueba la fecha, hace sus cálculos para saber sin nos quedan más días de evaluación y si no es así nos saca un mensaje de "Evaluación Finalizada" o algo así y puede que realice cambios irreversibles para que ya no pueda utilizarse más una versión no registrada. La cuestión no es saber cómo nos cuenta los días, bien guardando la fecha en que fue instalado, bien guardando los días que nos quedan o de otra forma aún más rebuscada, sino DONDE guarda esta información. Básicamente hay tres posibilidades: En el registro, en un archivo que utiliza el programa con este propósito o en el propio ejecutable. Para conocer cuál de estos métodos es el que nos interesa en nuestro caso, avanzamos un día el reloj, corremos el TechFacts 98 y vemos que modificaciones se producen. Hay 11 ramas del registro que se modifican:

[HKEY_LOCAL_MACHINE\Software\CLASSES\.FlashFXP_PI]
[HKEY_LOCAL_MACHINE\Software\CLASSES\.fqf]
[HKEY_LOCAL_MACHINE\Software\CLASSES\Applications\FlashFXP.exe]
[HKEY_LOCAL_MACHINE\Software\CLASSES\Applications\FlashFXP.exe\shell]
[HKEY_LOCAL_MACHINE\Software\CLASSES\Applications\FlashFXP.exe\shell\open\command]
[HKEY_LOCAL_MACHINE\Software\CLASSES\FlashFXP.Document]
[HKEY_LOCAL_MACHINE\Software\CLASSES\FlashFXP.Document\shell]
[HKEY_LOCAL_MACHINE\Software\CLASSES\FlashFXP.Document\shell\open\command]
[HKEY_LOCAL_MACHINE\Software\CLASSES\FlashFXP_PI]
[HKEY_LOCAL_MACHINE\Software\CLASSES\FlashFXP_PI\DefaultIcon]
[HKEY_USERS\.DEFAULT\Software\FlashFXP]

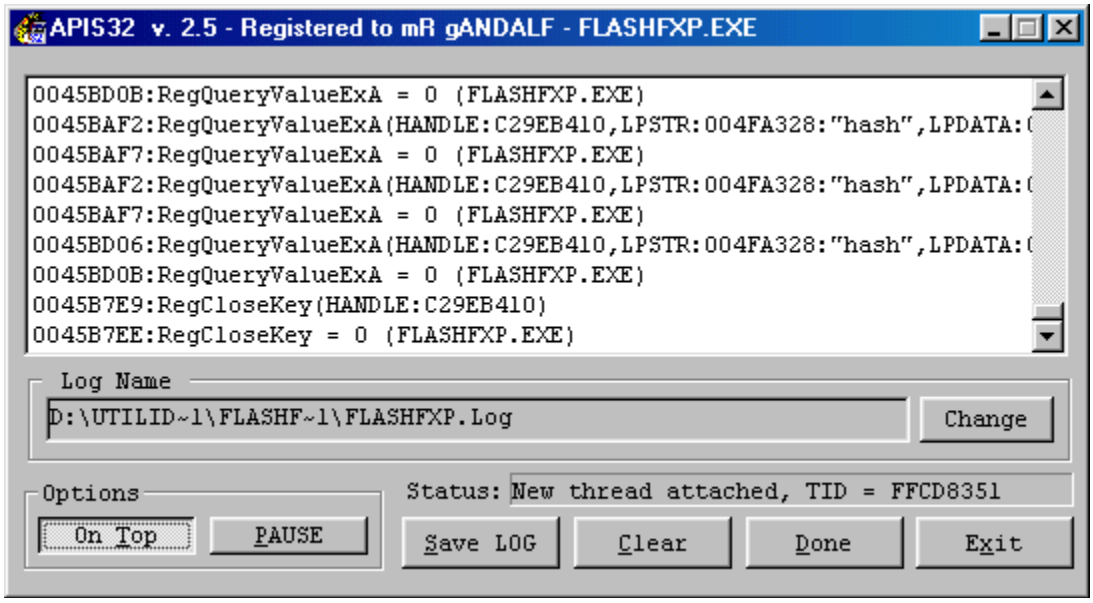
Como puede verse no todo es entrar en el Regedit y pedirle que busque FlashFXP, pues la 2º rama de esta colección no la encontraría, aunque en nuestro caso eso sería irrelevante. En algunos programas la clave de registro que interesa está escondida, encriptada y además no alude para nada al nombre de nuestro programa, por lo que es conveniente monitorizarlo. Yo tengo la costumbre de hacerlo con el TechFacts en los momentos claves, durante la instalación y cuando muevo el reloj o hay algún cambio que

me parece interesante. En alguna ocasión puede ser interesante volver al archivito de instalación y ver que hizo el programa. Una vez que sé que es en el registro donde se guarda esta información suelo correr el Advanced Registry Tracer que nos permite jugar un poco más con el registro.



Vemos que adelantando el reloj del día 15 al día 30 el valor de “hash” cambia de 37210 a 37240, e igual le sucede a “MaxIcon” (que no sale en la figura). De las dos clave es MaxIcon la que nos interesa. Esto se averigua por el simple procedimiento de ensayo y error, tan util para los que como yo, no estamos tocados por la varita del Zen Cracking. Posiblemente estas cifras representen formas más o menos crípticas de representar la fecha, pero simplemente poniendo MaxIcon a 0 nos basta para tener el contador siempre a 0.

Otro camino posible que se nos abre al conocer cual es la clave que gobierna el conteo es monitorizar los accesos del programa a dicha clave y llegar hasta el corazón mismo de la comparación de fechas, dejando que el complejo proceso de conteo avance pero sin que el programa sé de cuenta. Esta es una bonita maniobra que viene muy bien explicada en el tutorial de Karpoff sobre MemTurbo 1.5. Tomamos el APISpy y seleccionamos todas las API`s que vienen con ADVAPI32, o mejor aún solo la mitad última de ellas que hacen referencia al registro, y tendremos un listado limpio:



El listado de FlashFXP.log, que es donde ApiSpy guarda toda esa información que hace scroll en esta ventanita, es un poco más largo que este, pero yo he seleccionado la parte interesante:

0045BAF2:RegQueryValueExA(HANDLE:CAA0B560,LPSTR:004FB218:"**MaxIcon**",LPDATA:00000000,LPDATA:007AFBDC,LPDATA:00000000,LPDATA:007AFBF8)
0045BAF7:RegQueryValueExA = 0 (FLASHFXP.EXE)
0045BAF2:RegQueryValueExA(HANDLE:CAA0B560,LPSTR:004FB218:"**MaxIcon**",LPDATA:00000000,LPDATA:007AFBBC,LPDATA:00000000,LPDATA:007AFBD8)
0045BAF7:RegQueryValueExA = 0 (FLASHFXP.EXE)
0045BD06:RegQueryValueExA(HANDLE:CAA0B560,LPSTR:004FB218:"**MaxIcon**",LPDATA:00000000,LPDATA:007AFBD4,LPDATA:0196D740,LPDATA:007AFBE4)
0045BD0B:RegQueryValueExA = 0 (FLASHFXP.EXE)
0045B7E9:RegCloseKey(HANDLE:CAA0B560)
0045B7EE:RegCloseKey = 0 (FLASHFXP.EXE)

Aquí vemos desde que parte del código se "lee" la clave MaxIcon. Podríamos ir a mirar con el Wdasm y probar unos saltos con el Sice, pero creedme, no merece la pena tanto refinamiento para esta parte del crack. Si adelantamos la fecha más allá del tope del trial veremos que lo único que pasa es que donde quedaba el botón continuar ahora sale uno de esos contadores de tiempo que te hace esperar 10 segundos antes de continuar. Luego lo que verdaderamente nos limita es la ventana en sí y no el conteo. De todas formas metiendo esta clave en el registro lo conseguimos parar:

```
[HKEY_LOCAL_MACHINE\Software\CLASSES\FlashFXP_PI\DefaultIcon]
@="D:\\UTILIDADES\\FLASHFXP_ES\\FLASHFXP.EXE,1"
"MaxIcon"="0"
```

Como podéis ver yo tengo instalado el FlashFXP en D:\Utilidades\FlashFXP_es.

Vamos a por esa molesta nag...

Como yo soy un chico muy aplicado voy a seguir los pasos explicados en el tutorial de Dek-Oin sobre Nag`s, que es sobre todo un método, la mejor arma que podemos tener en esta guerra contra las protecciones.

Método de Anticipación: Consiste en anticiparnos a la salida de la Nag. Simplemente ponemos un Bpx en todas las API`s que suelen utilizarse para hacer Nag`s y esperamos a que el Sice salte. Allí donde caigamos es chico malo y a partir de hay que tracear hacía atrás hasta encontrar un salto que invertir o un call que nopear. Las API`s que nos interesan vienen en la lección 3 de Ricardo y que son:

bpx MessageBox
bpx MessageBoxExA
bpx MessageBeep
bpx SendMessage
bpx DialogBoxParamA
bpx CreateWindow
bpx CreateWindowEx
bpx ShowWindow
bpx UpdateWindow
bpx GetDlgItemText
bpx GetDlgItemInt
bpx GetWindowText
bpx GetWindowWord
bpx GetWindowInt

Lamentablemente el Sice no para en ninguna de estas, luego el señor Charles de Wise utiliza otro sistema para crear la Nag, vaya usted a saber cual.

Método de Salida: Consiste en identificar el punto donde la ventana se destruye para encontrar desde donde fue llamada. El comando de Sice es Bmsg *handle* wm_destroy, donde el handle, o manipulador, lo podemos encontrar con el wingestor de Crack el destripador y Marmota, o bien con el Sice haciendo task y buscando. Con este wingestor es cosa de niños eliminar nags, pero no nos sirve para hacerlo definitivamente. Siguiendo con el método de salida encuentro que es especialmente difícil tracear hacía atrás la nag en las ventanas con varios botones, pero igual son manías.

Existe un tercer método descrito por Dek-Oin que se me antoja similar al del wingestor, que consiste en in visibilizar la ventana utilizando el Resource Hacker y el exescope.

Llegados aquí seguimos sin tener solución para nuestra nag, por lo que habrá que improvisar. Afortunadamente y gracias a DaFixer (alabado sea) tenemos una herramienta llamada DeDe que incluso cuando más inútil parece nos termina sacando las castañas del fuego. Sí volvemos a FrmReg1 y ponemos Bpx al azar en alguno de estos eventos y probamos a tracear con F10 veremos desde donde se llama a la nag:

```
* Reference To: user32.PostMessageA, Ord:0000h
                    |
:004FA1DE E8B1D3F0FF          Call 00407594
:004FA1E3 A180E05200          mov eax, dword ptr [0052E080]
:004FA1E8 8B00                mov eax, dword ptr [eax]
:004FA1EA 8B10                mov edx, dword ptr [eax]
:004FA1EC FF92D0000000        call dword ptr [edx+000000D0]<- Aquí  se llama a la nag;
:004FA1F2 B201                mov dl, 01
:004FA1F4 A130B64500          mov eax, dword ptr [0045B630]
:004FA1F9 E87215F6FF          call 0045B770
```

Y un poco mas adelante se produce otro llamada:

```
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:004FA27B(U)
|
:004FA26D 8B45F8              mov eax, dword ptr [ebp-08]
:004FA270 E8AF8CF0FF          call 00402F24
:004FA275 C3                  ret


:004FA276 E94993F0FF          jmp 004035C4
:004FA27B EBF0                jmp 004FA26D
:004FA27D A180E05200          mov eax, dword ptr [0052E080]
:004FA282 8B00                mov eax, dword ptr [eax]
:004FA284 E8E736F5FF          call 0044D970<- Aquí se llama de nuevo a la nag;
```

Traceando un poquito hacia arriba encontramos:

```
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:004FA006(C)
|
:004FA036 803DDCDE520000      cmp byte ptr [0052DEDC], 00
:004FA03D 0F8546020000        jne 004FA289 <- Bin Laden haciendo aeromodelismo?
:004FA043 8B0D80E05200        mov ecx, dword ptr [0052E080]
:004FA049 A13CE35200          mov eax, dword ptr [0052E33C]
:004FA04E 8B00                mov eax, dword ptr [eax]
```

Sí hacemos **jmp 004FA289** evitamos limpiamente la nag**.**

Hacia el corazón de la protección...

Y como no podía ser menos en un programa hecho en Delphi termina saliendo todo el tomate. La cuestión obvia es: Ese **cmp byte ptr [0052DEDC], 00** ¿será una Flag?

Abrimos nuestro Hiew y a buscar. Salen solamente 45 líneas en las que aparece un cmp byte ptr [0052DEDC], 00 y siempre se sigue de un JE o un JNE.
Aquí hay material para andar probando que hace cada una y dejar el programa limpio como una patena, pero la protección ya está vencida y hay que empezar a prepararse para la fiestaaaaaaaaa!!!!!!!!!!

Agradecimientos en especial para mi MAESTRO, Ricardo Narvaja, de cuyos tutoriales he aprendido todo lo que sé.

OTRA DE MI AMIGO GANDALF

(LECCION CINCUENTA Y SEIS)

Introducción de Ricardo Narvaja: Otro tute magnifico de MR GANDALF que mientras yo me repongo del STRESS y se me vuelven a conectar los cables pelados, durante mis vacaciones de enero del curso, ya Gandalf hizo dos magníficos tutes los cuales le agradezco mucho y espero que les gusten como a mí, y si les gustan por favor háganselo saber así le dan fuerzas para hacer mas, no saben qué lindo es que te digan que un esfuerzo que uno hizo es reconocido por los demás.

GRACIAS A MR GANDALF

Programa: Internet Browser E-Mail Capturer 1.0.

Download: http://www.tecnomarketing.com/

Protección: Serial/Name. ASPACK. ShareWare Funciones limitadas.

Dificultad: Newbie.

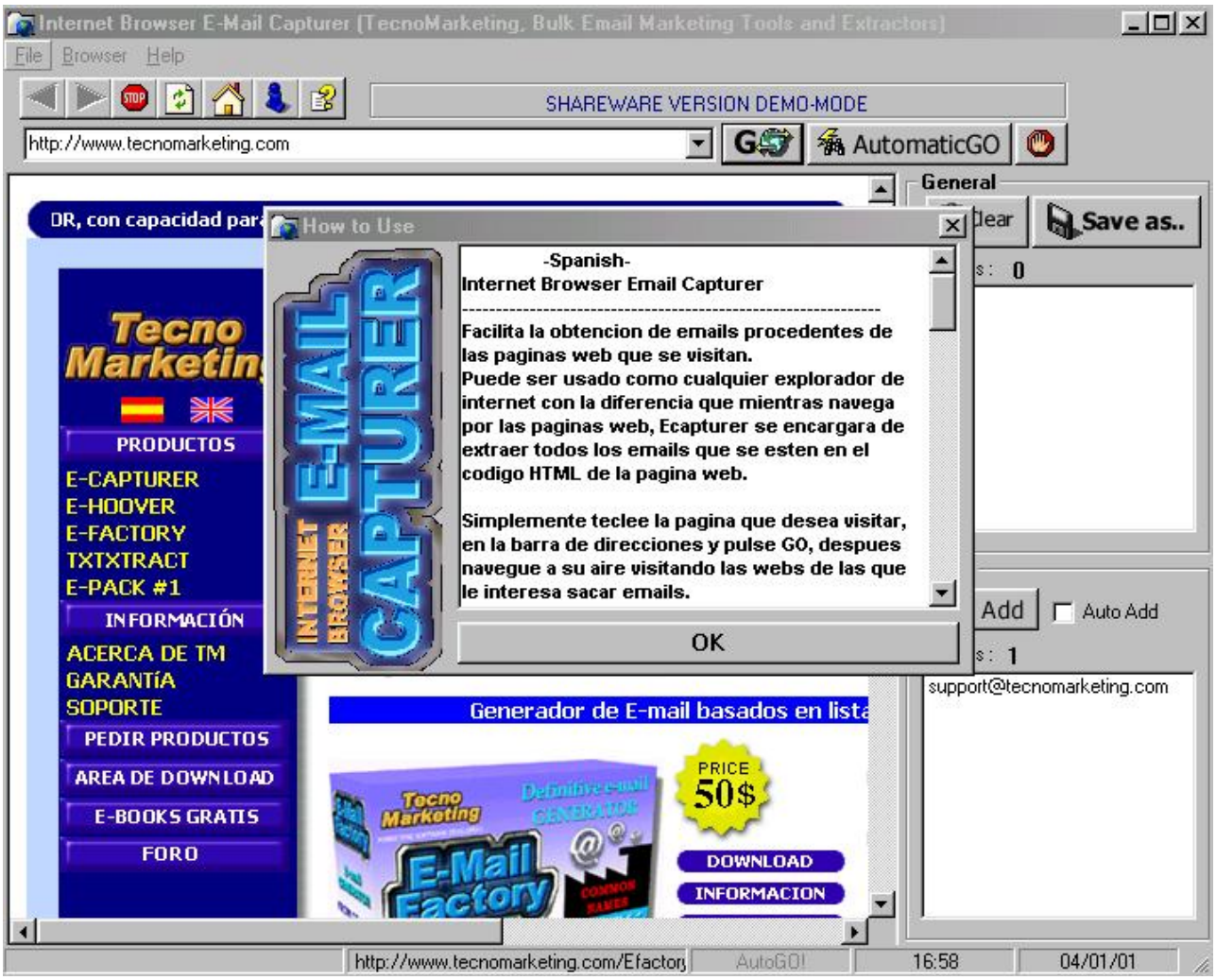
Herramientas: SoftICE 4.0. FrogsIce 1.09b4. IceDump 6.0.2.5. PEditor 1.5. Import Reconstructor 1.2. W32Dasm 8.93. Ppatcher 4.0.

Cracker: Mr Gandalf.

Fiche: 4/01/2002

Introducción: Casualmente encontré esta petición de Just_me en es.comp.cracks.

Francamente, el programa en sí no me interesaba mucho, pero quise probar y al comprobar que estaba hecho en Visual Basic pensé que sería tarea fácil. La función del programa parece ser facilitar la obtención de emails procedentes de las páginas web que se visitan, o al menos así versa.



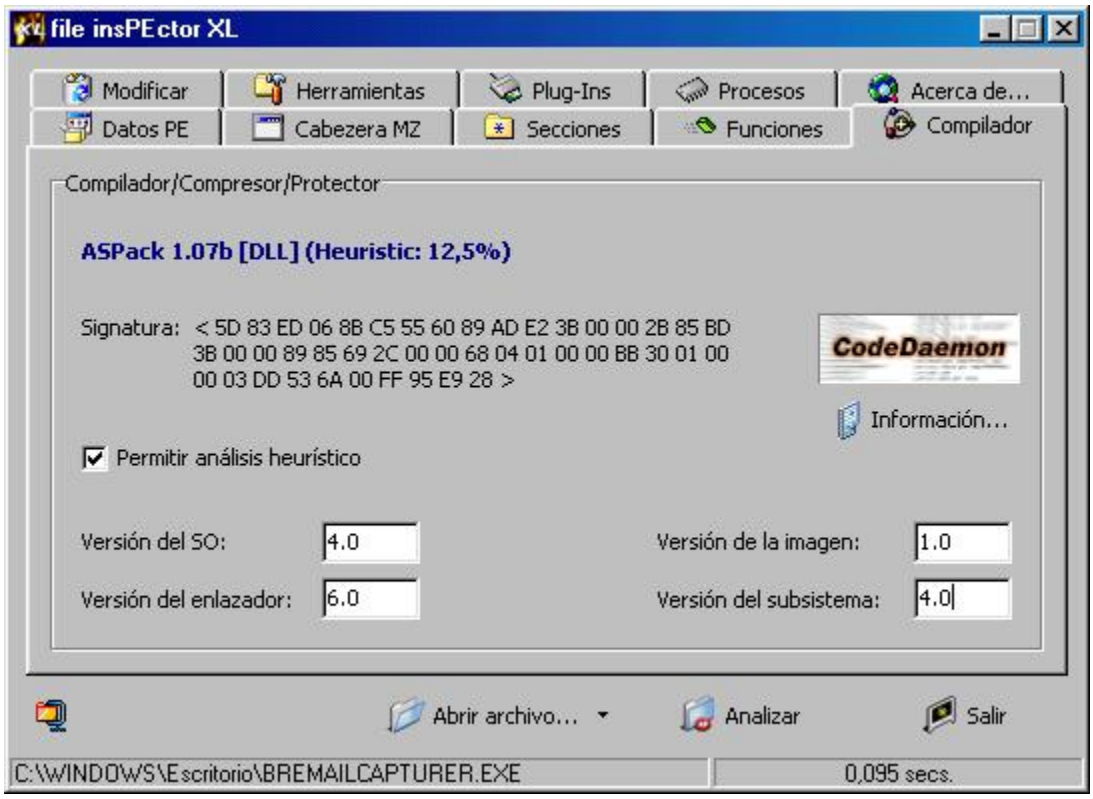
Desguaze:

En realidad no es tan fácil como arrancar el SmartCheck y leer la clave. De entrada está comprimido y protegido contra los depuradores de Numega, incluido el Sice. El FileInspector de Vipper fue el analizador que mejor y más información aportó.

Por lo visto es ASPack 1.07b y misteriosamente ningún desempacador de los que probé pudo con él, por lo que tuve que desempacarlo manualmente.

Lo primero es repasarse las lecciones 6, 34 y 35 del curso. Después todo va como la seda. Resumiendo, un programa protegido con un compresor/encriptador comercial como ASPack comienza ejecutando una rutina que descomprime/desencripta el programa para que pueda ejecutarse justo a continuación.

Por tanto, si desensamblamos el programa tal cual no obtendremos nada, o lo que obtendremos será erróneo. ASPack además trae una protección antiSICE, es decir, además de las rutinas de descompresión se ejecuta un código que busca la presencia de Sice en memoria y sí lo encuentra no permite que se inicie el programa o bien provoca un cuelgue de *windown*, como es el caso. Para obtener un ejecutable desprotegido de ASPack debemos copiarlo de la memoria una vez descomprimido/desencriptado, cambiar el entry point para que señale al comienzo del *programa en sí* y no al inicio de ASPack, y reconstruir la tabla de secciones. Siguiendo estos pasos...



Dumpeado fácil...

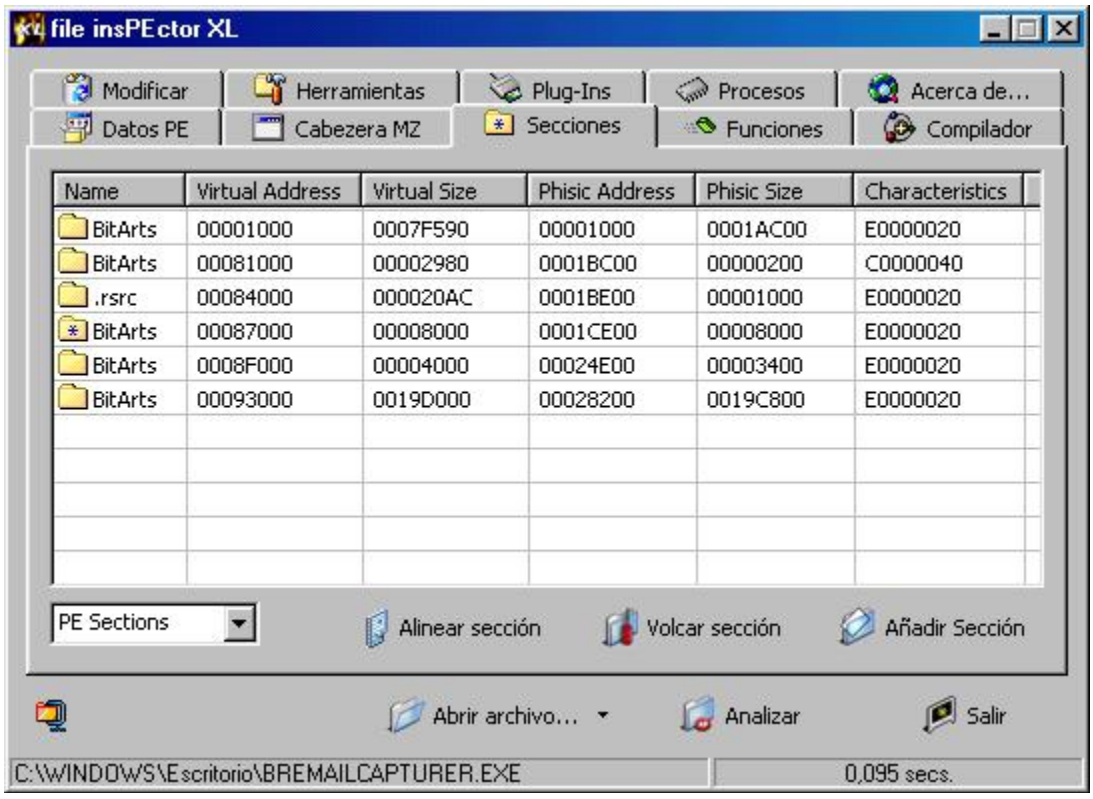
Para bajarnos el programa de la memoria necesitamos detener la ejecución justo en el inicio de la primera instrucción del *programa en sí* y entonces copiar el trozo de memoria donde está. La primera instrucción del *programa en sí* suele estar contenida en la primera sección de las que se muestran en la tabla de secciones. Calculamos el inicio y el final de esa primera sección y ponemos un *Bpr inicio final* con el Sice.

Inicio = Image Base + Virtual Offset
Final = Inicio + Virtual Size -1

	A	B	C	D	E
1	Image Base=	400000			
2	Virtual Offset=	1000			
3	Virtual Size=	7F590			
4	Raw Size=	1ac00			
5	Entry Point =	401900			
6	Size of Image=	230000			
7					
8	COMPRESIÓN	SI			
9	Inicio 1° sección=	401000			
10	Final 1° sección=	48058F			
11	Offset entry Point=	1900			
12	Incio del DUMP=	400000			
13	Fín del DUMP=	630000			

Hoja1 / Hoja2 / Hoja3

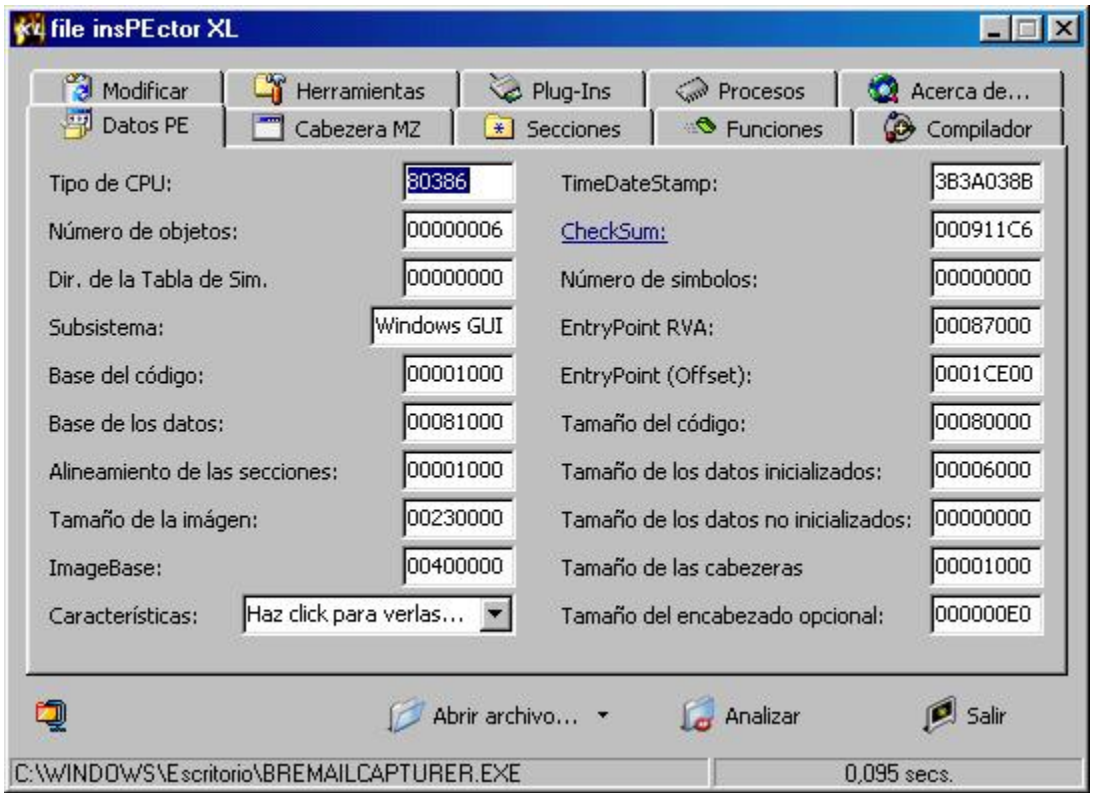
Yo utilizo esta pequeña utilidad en Excel para ahorrarme abrir la calculadora científica.



Entonces arrancamos el FrogsIce con las opciones Hide FPLoader, Protect Sice Files y Autoscan on exit/start-up. Cargamos IceDump y arrancamos el programa con el Symbol Loader (podría hacerse también desde el FrogsIce con la opción Hook int03h) y entramos al programa (es el principio de ASPack). Ponemos Bpr 018F:401000 018F:48058F r if (eip>=401000)&&(eip<=48058F), pulsamos F5 y después de unos 15 seg. aparecemos en 401900 Push 40F610.

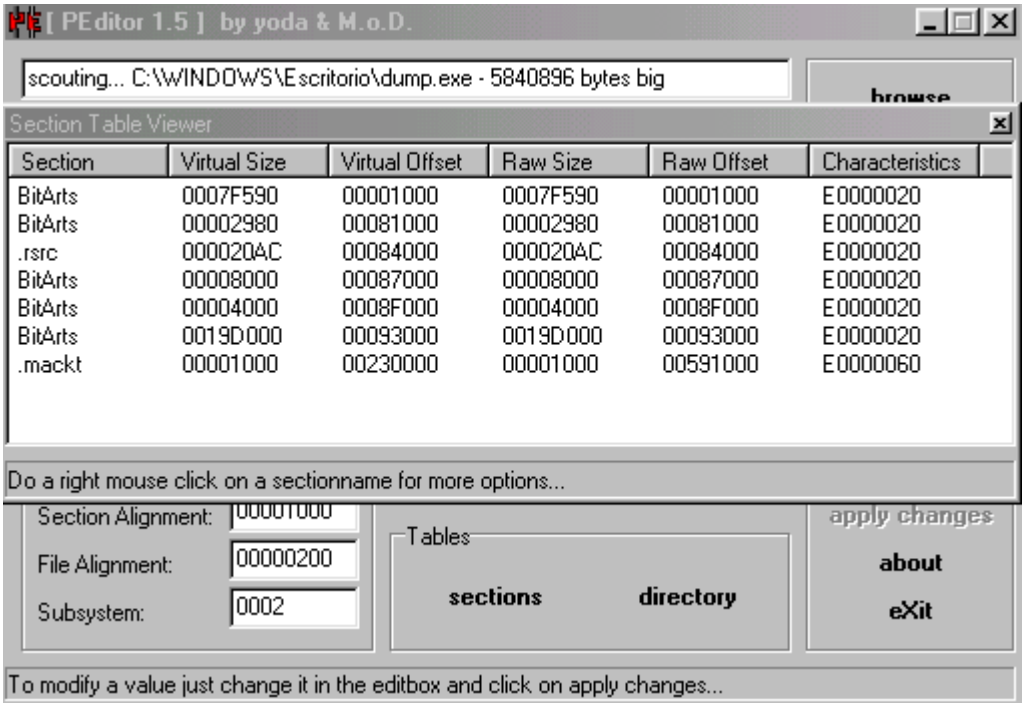
Esta es la primera instrucción del programa que dumpearemos. Podría ser un falso entry point, p.ej podría volver a ASPack, incluso varias veces. Sabemos que es el verdadero por que si seguimos traceando con F10 un buen rato vemos que ya nunca vuelve a ASPack sino que carga las dll (en este caso MSVBVM60) y comienza a ejecutarse el *programa en sí*.

Sin salir de Sice ni movernos de EIP 401900, ponemos */DUMP inicio fin c:\Temp./dump.exe*. Ahora inicio-fin son los del programa entero, no los de la primera sección. Estos nos los dice de nuevo el File inspector. El inicio es la image base y el tamaño es el size of image (por lo tanto fin = image base + size of image). Todo este procedimiento viene muy bien explicado en las lecciones 34 y 35 de RICARDO, pero como nunca viene mal un poco de practica, podéis coger el programa y seguir estos pasos. Escribimos /DUMP 400.000 630000 c:\Temp./dump.exe y tenemos el programa dumpeado con un tamaño de 5,57 MB (por 1,76 del ejecutable comprimido!).



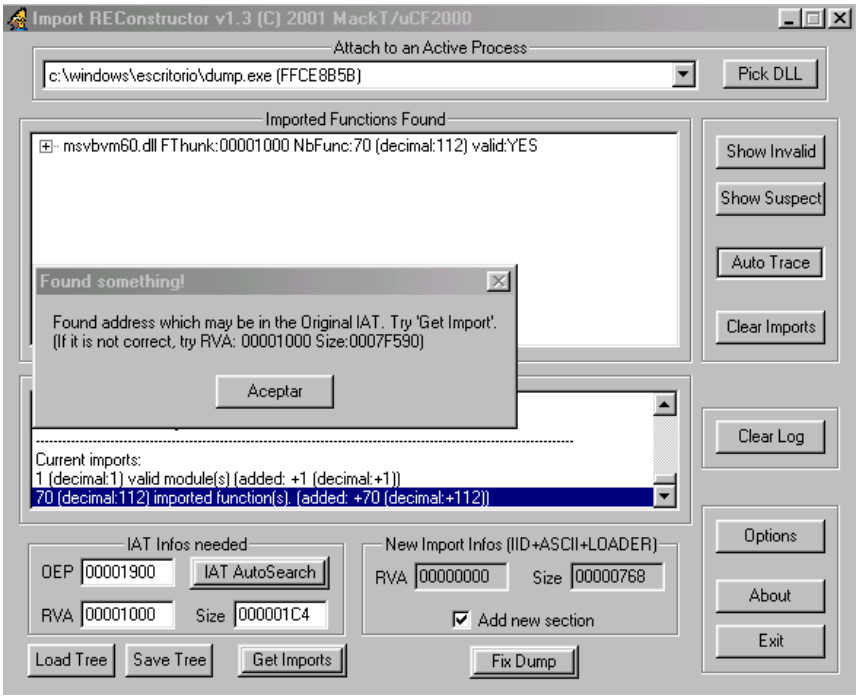
Arreglando el *dump*...

Ahora tenemos que "arreglar" el volcado, ya que así no nos serviría de nada. Abrimos el PEditor y empezamos los primeros cambios:



Cargamos el programa en browse, abrimos la tabla de secciones en *sections* y pulsando con el botón derecho del ratón se abre un menú en el que elegimos dumpfixer, con lo que se iguala el virtual size y raw size por un lado, y el virtual offset y raw offset por otro (queda como en la figura). También podemos aprovechar para habilitar poniendo E0000020 en *características* de todas las secciones.

Después hay que calcular el IAT ardes y el IAT length, para lo cual existe un método manual, pero yo probé el Import Reconstructor 1.2 y va de perlas. Hay que tener ejecutándose el programa original cuando arranquemos el Import Reconstructor y poner en OEP el offset del nuevo entry point, ósea, 1900 que ya teníamos calculado con nuestra hojita de Excel (RVA-Image Base = 401900-400000 = 1900).



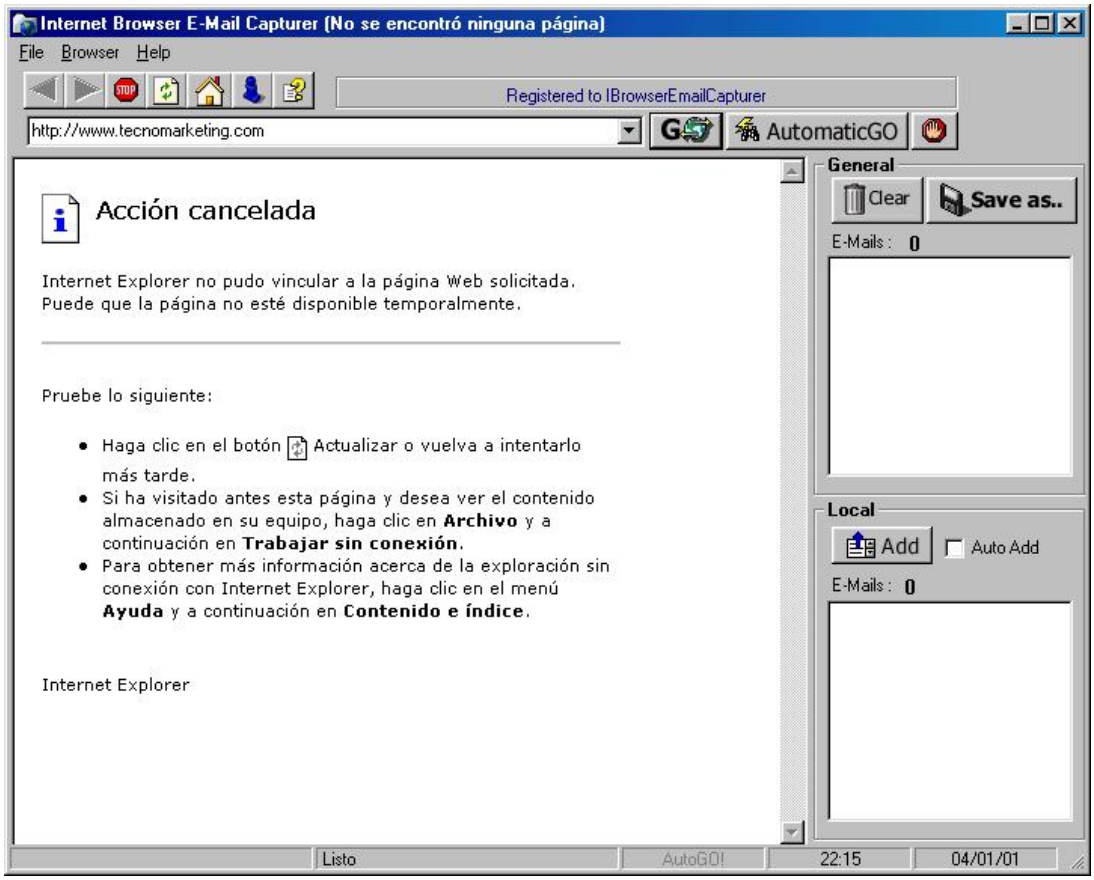
Entonces pulsamos IAT reconstructor y se rellenan RVA y Size, avisándonos de que si estos valores no nos sirven podemos usar otros. Pulsamos Get Imports y en imported functions aparece msvbsm60.dll (...) valid:YES, que indica que vamos bien. Ahora pulsamos Fix Dump, seleccionamos el archivo dumpeado cuando nos lo pida y *Viola!* Programa descomprimido y sin ASPack que podemos *debbugear* sin FrogsIce y desensamblar con sus String references y todo. Sí con este Fix dump no nos hubiera quedado un volcado decente, o se colgara el programa, podriamos llevar los datos de RVA y Size de la izquierda al Revirgín, como en la lección 34.

Listo para el almuerzo...

Así vencido y desnudito el pobre queda que da pena romperlo. Pero nosotros a lo nuestro, nos ha dado su buen trabajito y ahora hay que entrar a matar. El SmartCheck sigue sin funcionar pero el Wdasm preparado para VB es más que suficiente.

Recordáis que lo primero que sale cuando arrancamos el programa es un cartelito que dice "Unregistered versión working in demo mode"?. Un poco más arriba del cartelito aparece 47A611 JE 47A6AA. Sí cambiáis este por JNE entrareis registrados como Ibrowseremailcapturer. Sin cartelito, ni límites ni nada. Para el ejecutable comprimido hice el loader con el Ppatcher, que me gusta porque solo actúa cuando la víctima está ya confiada y por su nombre castizo.

Que aproveche.



PD: Gracias a Ricardo, de cuyas lecciones aprendí todo lo que sé y espero seguir aprendiendo (hay que pincharle para que haga más).

Ricardo Narvaja

(LECCION CINCUNTA Y SIETE)

Introducción de Ricardo Narvaja: Otro tutazo de MR GANDALF que cada día escribe mejor, muy bueno y completo con imágenes aclaratorias, para las listas donde no salen los gráficos sepan que este tute los tiene, y seguimos descansando el mes de enero, en la playa (jaja ojala) tirado en la cama y trabajando muerto de calor, jua.
Muy bueno GANDALF gracias por ayudarme, tu nivel es de primera y estoy reagradecido.

Programa: Simstat 1.3.

Download: http://www.simstat.com/

Protección: Trial de 30 días o 7 usos. ASPACK.

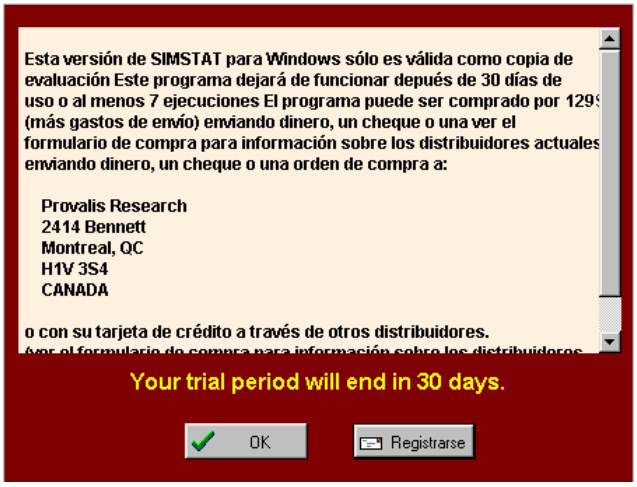
Dificultad: Newbie.

Herramientas: SoftICE 4.0. FileINspector XL. CASPR v1.012. ProcDump32 1.6.2. TechFacts 98. Advanced Registry Tracer 1.43. DeDe 2.50. W32Dasm 8.93. Ppatcher 4.0.

Cracker: Mr Gandalf.

Fecha: 10/01/2002

Introducción: Simstat 1.3 es un programa de la empresa canadiense Provalis Research que permite realizar detallados análisis estadísticos y, en opinión del Dr.Jorge R. Fernández, es uno de los mejores que hay. Fue precisamente a petición suya en la lista de cracks latinos que me decidí a intentarlo. Nada mas arrancar el programa y mientras se carga vemos una ventanita que nos dice que no estamos registrados, desaparece y, acto seguido, se nos muestra otra "ventanita" informándonos de que en versión demo solo puede usarse durante 30 días o 7 usos, lo cual es verdaderamente un comienzo prometedor.

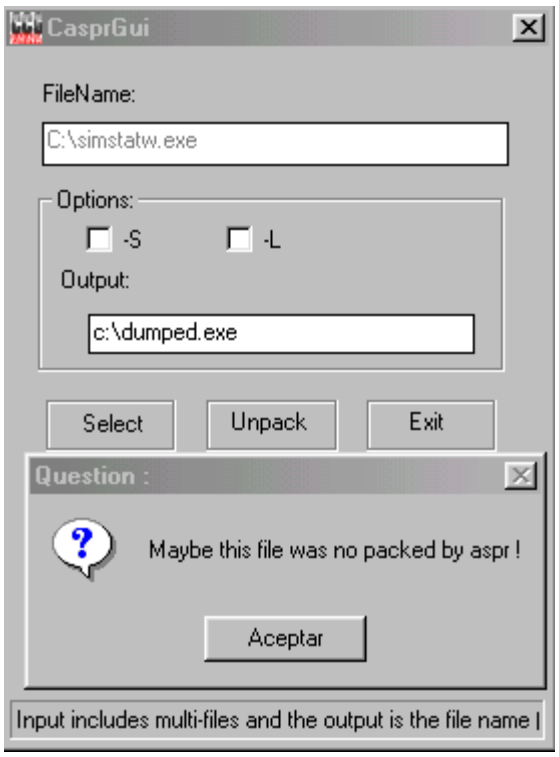


En el botón registrase entramos en una página de ayuda en la que nos explica las múltiples formas que tenemos de pagar para poder registrarnos. Según esta info lo recibiremos por correo cuando soltemos la pasta. Después de pulsar OK entramos en el programa que ya no muestra ningún signo de que estemos ante una versión share ni ofrece la posibilidad de introducir un pass o serial para registrarnos. Por otra parte, si pasan los 30 días pero no lo hemos usado 7 veces, el programa nos permitirá consumir estas oportunidades.

Siempre lo primero que hay que plantearse es el...

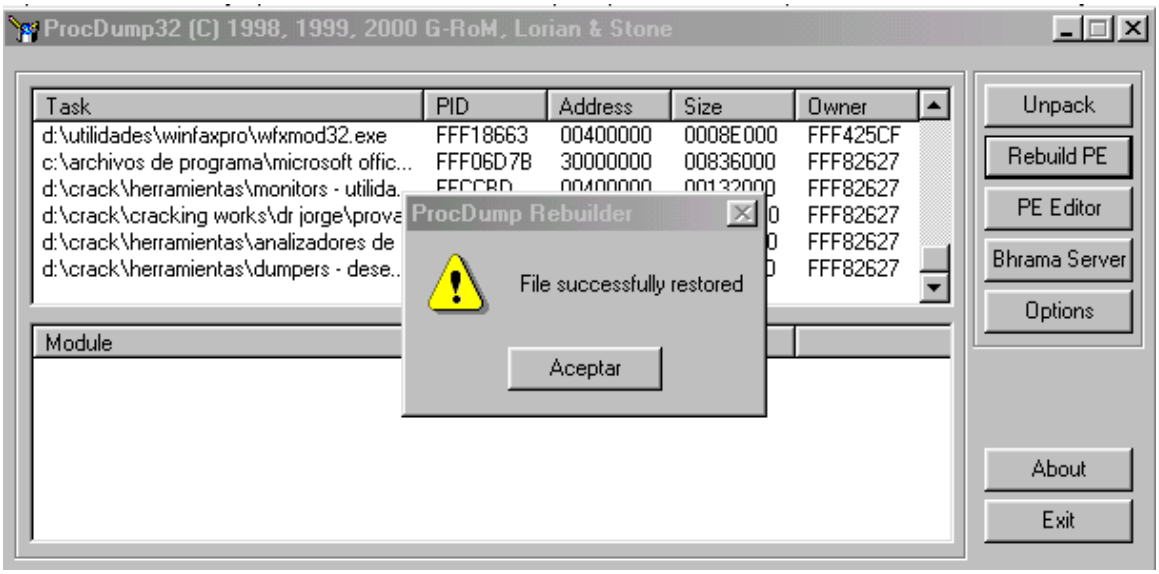
Desguaze...

Sí abrimos el ejecutable simstatw.exe con el **FileInspector** de VIPER nos encontraremos con que se trata de ASPack 1.08.04. En esta ocasión Un-Pack 2000 tampoco funcionó, pero teníamos a Caspr que lo desempacó sin dificultad, aunque nos muestra un cartel que dice que tal vez este archivo no esté comprimido por ASPr.



Antes de poder desensamblar tenemos que arreglar el encabezado del archivo con ProcDump. Para ello pulsamos rebuild PE y seleccionamos el volcado (c:\dumped.exe).

Después de esto editamos la tabla de secciones en PE Editor y ponemos características como E0000020 para todas las secciones.



Si a este archivo lo analizamos de nuevo con FileInspector nos llevaremos una sorpresa. Nos dice que esta compilado con Applok 95. La pregunta clave es: Mandeeeé??!!??! Pero como somos crackers de recursos nos iremos al **Languaje 2000** que nos dice que está hecho en Delphi...ufff, que susto...Después de esta información es obligado pasarse por el DeDe. Aquí veremos algunas cosas interesantes. Como siempre "pescar" en el Dede depende del olfato de cada uno y aunque a todos nos mosquean palabras como Tregister, puede haber otras tanto o más interesantes. Siempre es conveniente dedicar todo el tiempo necesario a revisar la estructura del programa en el DeDe y aunque a veces no encontremos nada de nuestro gusto y probemos a ver las string references en el Wdasm o a tracear las API`s en el Sice, es a menudo útil volver al DeDe con la información recopilada de otras herramientas y buscar aquellas direcciones que nos parecieron interesantes. En este caso, la verdad es que no hay nada destacable salvo un Tregister y un Shware, pero en ninguno de los dos para el Sice.

Como decíamos en el tutorial sobre el Flash FXP, el programa debe llevar la cuenta de los días que faltan en algún sitio: El registro, un archivo auxiliar o en el propio ejecutable. También "sabe" sí movemos el reloj hacia atrás, pues si hacemos esto nos sale un cartelito diciendo que la fecha del sistema ha sido cambiada e instándonos a registrarnos. Después de esto el programa ya no funciona más. Es de suponer que además de llevar la cuenta de los días que quedan también lleva la del día que instalamos el programa y si no concuerdan nos castiga. Sí nos damos un paseito por el Techfacts veremos que hay 4 archivitos y una rama de registro que se modifican cada vez que adelantamos un día el reloj:

c:\WINDOWS\APPLOG\APPLOG.ind
c:\WINDOWS\msds32.dll
c:\WINDOWS\simst32.ini
c:\WINDOWS\SYSTEM\mstx32.dll
HKEY_LOCAL_MACHINE\Software\Microsoft\Active Setup\Installed Components\{44BBA840-CC51-11CF-AAFA-00AA00B6015C}.Restore Value "DontAsk": from "15794178" to "15532034"

Sin embargo, si corremos el Advanced Registry Tracer y lo hacemos después de haber cerrado ese montón de aplicaciones que corren en secundario, veremos que ningún valor del registro cambia como consecuencia de adelantar el reloj. De entre los 4 archivos son msds32.dll y mstx32.dll, ambos creados durante la instalación del programa, los que llevan el conteo de los días y los usos, al parecer de forma duplicada. Borrando esos 2 archivitos volvemos a 0 usos y 30 días de Trial, salvo que el que nos haya vendido el ordenador sea el mismo que se lo vendió al Dr. Jorge R. Fernández, en cuyo caso es mejor ponerse una cataplasma fresquita en el occipucio, por lo que pueda pasar. Con esto podemos decir que lo tenemos cogido y va a ser muy difícil que se nos escape.

Si echamos un vistazo en el **APISs32** a las API`s que escriben en fichero, veremos que WriteFile es llamada en dos ocasiones durante el arranque del programa, una para escribir en msds32.dll y otra para escribir en mstx32.dll. Además esta API no se utiliza en ninguna otra parte del programa, como podemos comprobar poniendo un Bpx en el Sice y corriendo todas las opciones. Esta API viene muy bien documentada en las Crackers Notes, traducidas gentilmente por TXELI.

WriteFile

La función WriteFile escribe datos a un archivo y está diseñado para la operación sincrónica y asíncrona. La función comienza escribiendo datos al archivo en la posición indicado por el indicador del archivo. Después de que la operación de escritura se ha completado, el indicador del archivo es ajustado realmente por el número de bytes escritos, excepto cuando el archivo es abierto con FILE_FLAG_OVERLAPPED. Si el manipulador de archivos se creara para la entrada y salida solapada (I/O), la aplicación debe ajustar la posición del indicador del archivo después de que la operación de escritura es terminada.

```
BOOL WriteFile(
  HANDLE  hFile,                    // el manipulador de archivo para escribir
  LPCVOID  lpBuffer,                // la dirección de datos para escribir al archivo
  DWORD  nNumberOfBytesToWrite,     // el número de bytes a escribir
  LPDWORD  lpNumberOfBytesWritten,  // la dirección del número de bytes escritos
  LPOVERLAPPED  lpOverlapped        // la direc. de estructura necesaria para I/O solapada
  );
```

Returns
Si la función tiene éxito, el valor de retorno es TRUE.

Si la función falla, el valor del retorno es FALSE. Para conseguir información extendida del error, llama a GetLastError.

Lo que en nuestro listado de Wdasm corresponde a:

```
:00408B96 6A00                  push 00000000
:00408B98 8D442404              lea eax, dword ptr [esp+04]

:00408B9C 50                    push eax
:00408B9D 57                    push edi
:00408B9E 56                    push esi
:00408B9F 53                    push ebx

* Reference To: KERNEL32.WriteFile, Ord:0000h
                                 |
:00408BA0 E82BE4FFFF            Call 00406FD0
:00408BA5 85C0                  test eax, eax
:00408BA7 7507                  jne 00408BB0
```

Es en ESI donde se pasa LPCVOID, que es un puntero a la dirección de los datos que van a ser escritos, ósea, el número de días que faltan. Ahora solo tenemos que cambiar ESI por otra cosa y habremos conseguido que el programa, después de haber hecho múltiples cálculos y comprobaciones para ver si estamos registrados, termine guardando lo que nosotros queramos. Si cambiamos 408B9E PUSH ESI por PUSH EDI, ósea 0x408B9E/0x56/0x57 en el Process Patcher, habremos conseguido que el trial dure indefinidamente. Ahora solo nos queda quitarnos de en medio la nag.

Por curiosidad, si echamos una ojeada en **FileMon** (que tiempos aquellos de Mortadelo) veremos que el programa intenta acceder a un archivo simstat.lic. Lic de licencia, se supone. Si creamos un archivo simstat.lic entonces vemos que lo abre, lee algo, y luego sigue como si tal. Evidentemente no le convenció lo que le ofrecimos. Este era otro camino para romper la protección pero al ver que se complicaba la cosa lo dejé como 2º opción.

Para eliminar la nag ya hemos visto cuales son los procedimientos habituales en el tute del FlashFXP. Ninguno de estos funciona tampoco en el Simstat. La nag se forma elemento a elemento, es decir fondo rojo (428171 CALL 0042FC0) y el resto dentro de un bucle, texto (4281A8 CALL 406FD0 la vez 43 y 44 que se ejecuta) botón registrar (el mismo CALL a la 45º vez) y botón ok (46º vez).

```
:004281A6 8B03                  mov eax, dword ptr [ebx]----------
:004281A8 E89B210000            call 0042A348
:004281AD 8B03                  mov eax, dword ptr [ebx]
:004281AF 80787C00              cmp byte ptr [eax+7C], 00
:004281B3 740F                  je 004281C4
:004281B5 8B45FC                mov eax, dword ptr [ebp-04]
:004281B8 C780500100000200000   mov dword ptr [ebx+00000150], 00000002
:004281C2 EB14                  jmp 004281D8
```

- Referenced by a (U)nconditional or (C)onditional Jump at Address:

```
|:004281B(C)
|
:004281C4 8B45FC                mov eax, dword ptr [ebp-04]
```

```
:004281C7 83B85001000000        cmp dword ptr [eax+00000150], 00000000
:004281CE 7408                  je 004281D8
:004281D0 8B45FC                mov eax, dword ptr [ebp-04]
:004281D3 E814FAFFFF            call 00427BEC
```

- Referenced by a (U)nconditional or (C)onditional Jump at Addresses:

```
|:004281C2(U), :004281CE(C)
|
:004281D8 8B45FC                mov eax, dword ptr [ebp-04]
:004281DB 8B8050010000          mov eax, dword ptr [eax+00000150]
:004281E1 85C0                  test eax, eax
:004281E3 74C1                  je 004281A6 --------------------------
```

La cosa no pinta facilona. Si rompemos el bucle, por ejemplo, haciendo 004281E3 jne 004281A6, parecerá que hemos vencido. No sale la nag, el programa aparentemente funciona, pero... no sale la ventana de about (en Help) ni el Tools Setup (en Tools). Parece que ese bucle hacia más cosas que dar la lata. También podemos probar el método de tracear con F10. Por este método veremos que el que da entrada a esta parte del código es un 5F0395 CALL 005F0000. Este CALL es responsable de un procedimiento, según vemos en DeDe, al parecer el responsable de toda la seguridad del programa. Nopeando este CALL queda completamente crackeado el programa, no hace falta más cambios. De todos los loader que probé, nuevamente solo el **Process Patcher** funcionó correctamente.

mR gANDALF

PD: Gracias a mi maestro, RICARDO, de cuyos tutoriales aprendí todo lo que sé y espero seguir aprendiendo.

Ricardo Narvaja

(LECCION CINCUENTA Y OCHO)

Introducción de Ricardo Narvaja: quería agradecerle yo, ya que no ha habido muchos agradecimientos en la lista a mr gandalf un verdadero amigo y como han visto ya todo un cracker, la ayuda que me ha dado durante este mes de enero, y decirle que siempre que quiera colaborar en cualquier momento con un tute de él, será bienvenido ya que a mí me gustan mucho sus tutes, la opinión de los demás no la sabemos ya que nadie dijo nada, pero bueno yo lo aliento porque yo sé lo que es estar horas detrás de una computadora tratando de crackear un programa para ayudar a alguien o para enseñar, con solo el rédito de la satisfacción por ayudar a los demás sin esperar nada.
Por eso amigo gandalf: muchísimas gracias y quisiera que este no fuera tu ultimo tute como vos me escribiste, sea por el motivo que sea me gustaría que alguna vez cuando puedas y te dé el tiempo, mandes una colaboración para despuntar el vicio tuyo y mío, el cracking.

UN ABRZO GANDALF SABES QUE ACA TENES UN AMIGO
RICARDO NARVAJA

Programa: SciTech GLDirect Versión 2 (28 Jun 2001).

Download: www.scitechsoft.com

Protección: Trial de 21 dias. Registración por name y serial.

Dificultad: Fácil.

Herramientas: SoftICE 4.0. W32Dasm 8.93. Hiew 6.76.

Cracker: mR gANDALF.

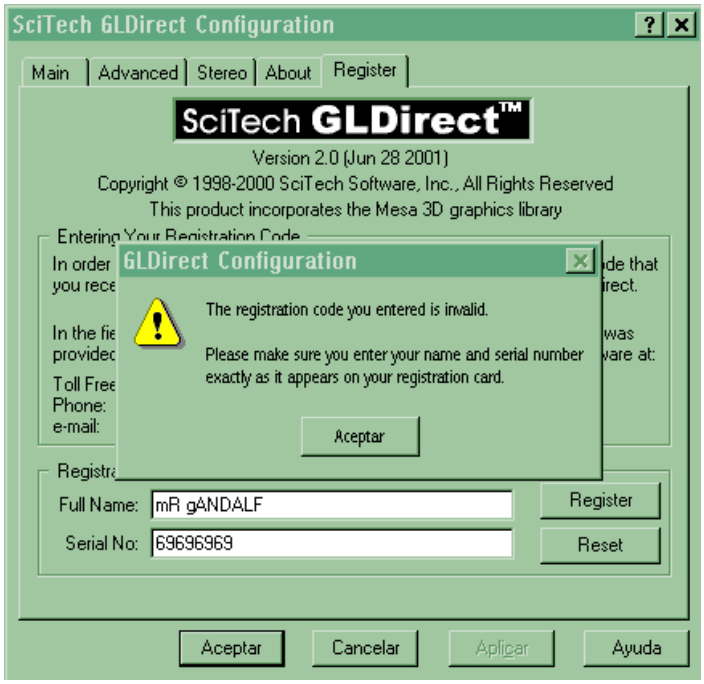
Fecha: 30/01/2002

Introducción: Después de un tiempo al margen de la lista debido a problemas técnicos (estallido de un CD en la unidad de DVD en periodo de garantía del ordenador, con lo que hubo que llevarlo a la tienda con el consiguiente cashondeo y "vuelvaustedmañaneo") el humilde aprendiz de crackeador y amigo vuestro, mR gANDALF, resurge como el Ave Fenix para atender una petición de la lista. Se trata de noseman, muy interesado en este programita que para funcionar necesita DirectX 7 o superior, ojo!, que podéis bajar de la red, pero no utilicéis para ello la utilidad que acompaña al programa, es mi consejo.

Según reza la ayuda del programa (in english, of course) SciTech GLDirect is the utility package for Windows 95/98/Me that combines the power of the OpenGL API with the wide availability of Direct3D hardware drivers. It accomplishes this by enabling OpenGL based games and applications to access 3D hardware acceleration through the Direct3D 7.x drivers provided by your graphics hardware manufacturer. The OpenGL API is the cross-platform, high performance standard for 3D graphics applications..

Desguaze: Como siempre empezando por el principio es como mejor se llega al final. Lo primero es instalarlo bajo estrecha monitorización de TechFacts guardando el informe en un archivo por si acaso luego hay que buscar algo interesante. El primer uso también lo monitorizo con TechFacts.

Después hay que echarle un vistazo con el FileInspector de VIPPER y ver con quien nos medimos: Parece no empacado y además es Visual C++ 6.0, buena señal. Si echamos un vistazo en la pestaña que dice sections veremos que en características nos pone E0000020, que es lo que nos permite desensamblarlo con Wdasm. Por tanto cambiémoslo y veamos el "listado muerto" con Wdasm. Aquí aparecen cosas interesantes, como el texto de los mensajes de chico malo que nos salen al hacer un intento de registrarnos:



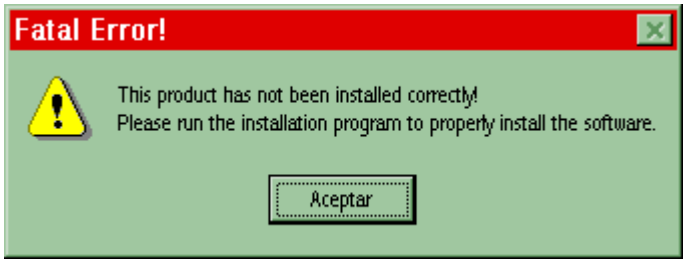
Francamente esto va tomando buena pinta. Si ahora ponemos un Bpx Hmencpy antes de pulsar el botón Register y traceamos hacia delante con F10 llegaremos al cartel de chico malo en 405AFC CALL dword ptr [041E054] "The registration code you entered is invalid". Fijense que hay distintos carteles de chico malo. En todos pone arriba: "The registration code you entered, etc... ". Sin embargo abajo el mensaje es distinto, según que no sea correcta la longitud u otros parámetros que el programa mira en el serial. Ahora el procedimiento es sencillo, se trata simplemente de intentar buscar un salto que evite estos mensajes y que nos registre.
Después de un par de intentos facilones sin éxito mirando en el listado de Wdasm, me fui al Sice que me gusta más. Con un Bpx en Hmencpy y sabiendo cual es el salto de chico malo fui traceando hacia delante, mirando por aquí y por allá como el programa toma mi name y mi serial y se pone a hacer malabarismos binarios para encriptarlos y que no sea obvia la comparación. Lo más "limpio" aquí seria intentar desenmascarar la rutina encriptadora y hacer un keygen. Pero si te encuentras perezoso y piensas que con registrarlo es suficiente, entonces solo sigue traceando hacia delante. Verás que en sucesivos *jes* precedidos de los respectivos *Coles* (de Bruselas) va realizando cada una de las comprobaciones pertinentes cuyos mensajes de error encontraras en las String References de Wdasm:

```
:00408E80 E88BF6FFFF          call 00408510      <- Aquí vuelve eax 0
:00408E85 83C41C              add esp, 0000001C
:00408E88 85C0                test eax, eax
:00408E8A  750B                          jne  00408E97          <- Aquí evita el cartel
"Please make sure you entered the name..."
:00408E8C 5F                  pop edi
:00408E8D B802000000          mov eax, 00000002
:00408E92 5E                  pop esi
:00408E93 83C408              add esp, 00000008
:00408E96 C3                  ret

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:00408E8A(C)
|
:00408E97 837C240801          cmp dword ptr [esp+08], 00000001
:00408E9C 760B                jbe 00408EA9       <- Aquí evita el cartel "Please make sure
you obtain the ..."
:00408E9E 5F                  pop edi
:00408E9F B802000000          mov eax, 00000002
:00408EA4 5E                  pop esi
:00408EA5 83C408              add esp, 00000008
:00408EA8 C3                  ret

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:00408E9C(C)
```

Aquí sigue con la secuencia que muestra el cartel de registrado (...)

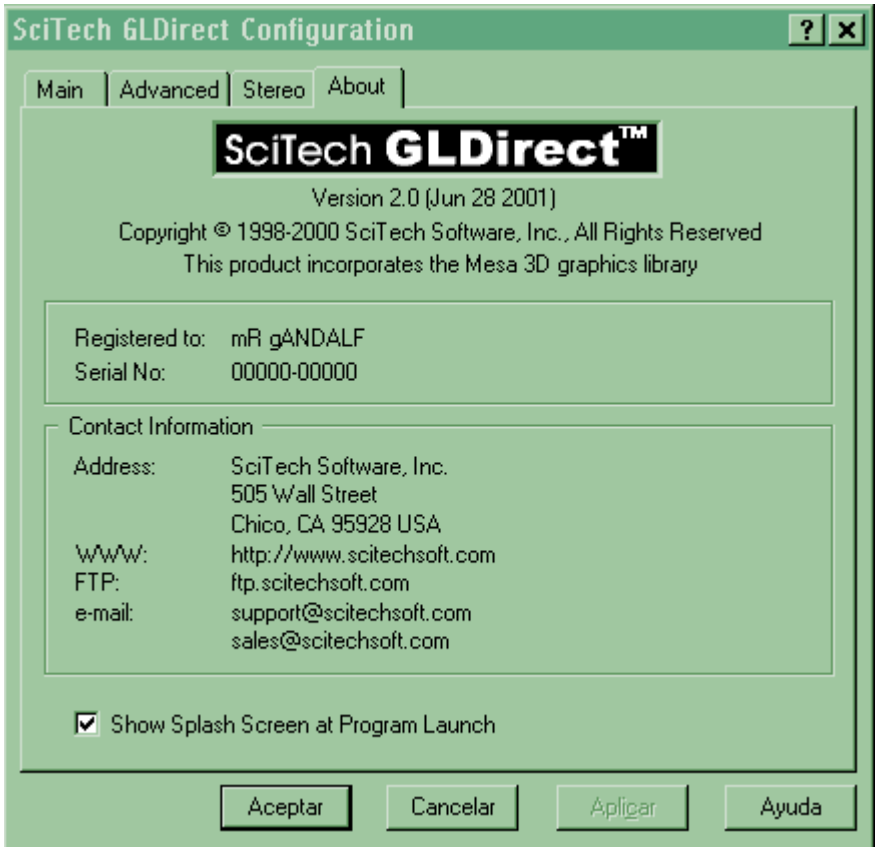
Y diréis... ¡Pues que fácil, jope!... Pues NO!!, jope NO que es de pijo, lesheee

Probad a reiniciarlo y .... ooohhhhh!!!!....



que maric!!... Con esto aprenderemos algo importante. El método en esto del crackeo es fundamental, como en cualquier cosa. Sí hubiéramos seguido un buen método, considerando la posibilidad de que el programa realizara mas comprobaciones o marcará nuestro ordenador de algún modo para que después de detectar un intento de pirateo no arrancara más, por ejemplo monitorizando con TechFacts (y mejor si guardamos también un "snapshoot" en ART), ahora todo seria coser y cantar. Probablemente el programa guarde nuestra clave o un engendro encriptado y escondido de nuestra clave en el registro o en un archivito perdido en el disco duro, realice una comprobación al iniciarse y se de cuenta del pastel que le hemos preparado. Como nos hemos confiado y no hemos hecho esto, lo cual nos facilitaría mucho las cosas, ahora solo nos queda el procedimiento de buscar de donde sale esta ventanita y quitarla, a ver que pasa. Traceando hacia delante vemos que sale de:

```
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:00407D2D(C)
|
:00407D6D E84E0D0000            call 00408AC0
:00407D72 85C0                test eax, eax
:00407D74                   jz 407D8F    <-       Aquí se evita el CALL
:00407D75 1968BC              sbb dword ptr [eax-44], ebp
:00407D78 124200              adc al, byte ptr [edx+00]
:00407D7B E84E0D0000            call 408AC0  <- Aquí sale el cartelito malvado
:00407D7C D01B                rcr byte ptr [ebx], 1
:00407D7E 0000                add byte ptr [eax], al
:00407D80 83C404              add esp, 00000004
:00407D83 33C0                xor eax, eax
:00407D85 5F                  pop edi
:00407D86 5E                  pop esi
:00407D87 5D                  pop ebp
:00407D88 5B                  pop ebx
:00407D89 83C438              add esp, 00000038
:00407D8C C21000              ret 0010
```



A pesar de todo el programa nos lo puso fácil. Este crack puede hacerse con cualquier patcher o loader pero adolecerá de un elemento que puede ser importante: La caducidad. Si lo aplicamos a un programa que ya ha caducado posiblemente no funcione. Yo ya no puedo dar marcha atrás pues aunque lo desinstale no me da la posibilidad de volver a entrar en modo share y estudiar el modo de vencer el límite de tiempo, que no puede ser muy difícil, visto lo visto. Por eso lo dejo para otra mente más aguda o que siga más escrupulosamente el método, tan importante en esto del cracking como ya dijo el maestro Descartes...

mR gANDALF

PD: Y a mi amigo Ricardo Narvaja, maestro también, un fuerte abrazo y mucho ánimo para que siga con este curso estupendo que ayudará, sin duda, a muchos a iniciarse en esto del cracking.

Ricardo Narvaja

(LECCION CINCUENTA Y NUEVE)

Programa: McAfee Firewall 3.0 (versión en inglés de 30 días).
McAfee Virus Scan 6.0 Pro (versión en ingés de 30 días)
McAfee Virus Scan 6.0 (versión en español de 30 días)
McAfee QuickClean 2.0 (versión en castellano de 30 días)
McAfee Internet Security 4.0 (version en ingles de 90 días)

Protección: Trial de 30 días.

Download: CD que acompaña a la revista PC Actual de Diciembre del 2001.

Dificultad: Mediana.

Herramientas: SoftICE 4.0. TechFacts 98. Advanced Registry Tracer 1.43. Hiew 6.76.

Cracker: mR gANDALF.

Fecha:03/02/2002

Introducción: Se trata del completo y conocido paquete de aplicaciones de McAfee sobre seguridad informática, a saber, Firewall, VirusScan, QuickClean e Internet Security (que incluye los dos últimos), que en su versión shareware, completamente operativa durante un periodo variable de tiempo, ha sido recientemente distribuida en un CD gratuito por algunas revistas informáticas. En este caso se trata del número de Diciembre del 2001 de PC Actual, que generosamente aporta tan suculenta y, a juzgar por la búsqueda en Astalavista, pocas veces escogida víctima.

Por otra parte y por aquello del qué dirán, tengo que recordarles que este tutorial se publica en la lista de cracks latinos únicamente con fines educativos y que si les interesa el programa deben comprarlo.

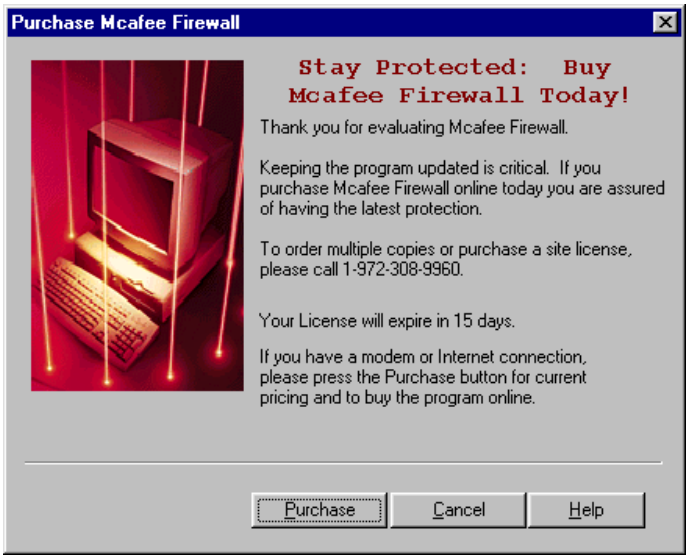
Desguaze:

Digamos que para crackear este programa es suficiente con emplear un poco de lógica y ser escrupulosos con cada paso que damos, haciendo uso exhaustivo del "método Cartesiano" del que hablábamos en la lección 58 (verdad que sí, noseman?). Sepamos que se trata de un programa Visual C++ 6.0, no empacado ni encriptado, con múltiples librerías distribuidas por varias carpetas del directorio C:\Archivos de Programa\McAfee.

... lo primero SIEMPRE es el planteamiento...

Nuestro objetivo es conocer como lleva el programa la cuenta del tiempo. De un modo general esto puede hacerse de varias maneras: Guardando la fecha en que fue instalado, la fecha en la que caduca, el número de días que se llevan gastados, el número de días que faltan o una combinación cualquiera de estos o de todos ellos. Por otra parte, el sitio físico donde esta información se guarda puede ser una clave del registro (habitualmente; muy interesante el tutorial de Karpoff sobre el MemTurbo 1.5), uno o varios archivos escondidos (véase lección 57, Simstatw), en el propio ejecutable (hasta ahora nunca lo vi) o mediante una combinación cualquiera. Para saber cual de estas es/son la/las que el programa emplea deberemos monitorizar que cambios se producen en el disco duro y en el registro de Windows mientras modificamos la fecha del sistema. Aquí es donde el TechFacts y el ART, cuyo manejo no voy a repetir porque ya lo hizo Ricardo (lecciones 32 y 19 respectivamente), son fundamentales.

Nos centraremos en el McAfee Firewall 3.0, aunque adelanto que el sistema de protección es común a todas las aplicaciones y que crackeada una, crackeadas todas. Este programa tiene una limitación de 30 días, aunque ya llegando al 15 empieza a avisarnos de que el tiempo corre que se las pela.



Pasado el día 30 nos muestra un cartelito similar avisándonos que el trial concluyó y el programa simplemente se cierra, eso sí, siempre nos da la opción “Purchase”.

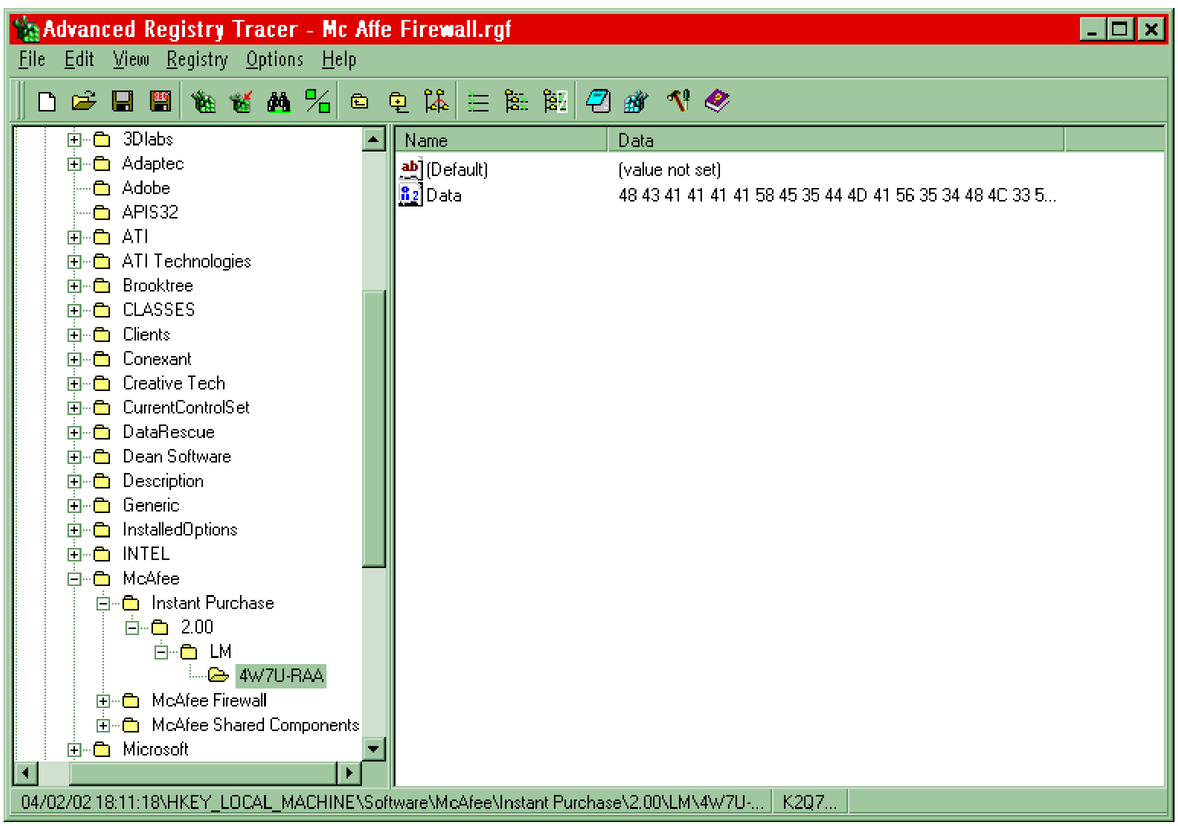
... el trabajo de campo ...

Utilizando el TechFacts y el ART llegamos a la conclusión de que es en el registro donde se guarda la fecha en la que el programa fue instalado y a partir de ahí hace sus cálculos. Normalmente los programas suelen guardar también un conteo de días y cuando se percatan de que les toquetean el reloj hacen vencer el trial (lección 55, Flash FXP). Además, cuando desinstalamos un programa NO se borran las claves del registro que informan de la caducidad del mismo, por lo que reinstalarlo es inútil (verdad noseman?). Ni siquiera reinstalar Windows sirve, pues el programa de instalación recupera los archivos donde asienta el registro ,system.dat y user.dat, que guardan información tan molesta como que programas han caducado. Un conocimiento algo más profundo del registro de Windows es necesario y para ello recomiendo el excelente tutorial de Karlitox que podréis encontrar en la pagina de Karpoff, como siempre.

Lógicamente, si atrasamos el reloj una vez el programa ha caducado, podrá volver a funcionar, cosa que no sucede en la mayoría de programas, como ya dijimos. Gracias al ART podemos borrar esas incomodas huellas del registro, para volver a instalar el programa con una fecha distinta. También podemos crear archivos *.reg con los que engañarlo sobre la fecha de instalación. Este es un paso fundamental para llegar a saber cual es la clave del registro que guarda la información, que es lo que nos llevará hasta la “madriguera del oso”. Veamos: Antes de instalar la 1º vez tomamos una “foto” del registro, llameémosla F1. Hacemos la 1º instalación con fecha de Febrero, p.ej, y tomamos otra foto, F2. Si comparamos ambas y extraemos un archivo desinstalación.reg con el comando “Undo” podremos borrar toda huella de instalaciones previas. Además, si hacemos un archivo Reinstalar.reg con el comando “Redo”, conseguiremos el efecto opuesto. Veamos: Sí hacemos una nueva instalación (tomamos foto, F3) una vez “limpio” el registro, con fecha de Abril, p.ej, y ejecutamos el archivo reinstalacion.reg que obtuvimos de comparar F1 y F2, veremos que el programa se caduca, pues le hemos introducido una clave de registro que le informa de que fue instalado en Febrero. Este efecto se invierte con una archivo “Redo” obtenido de comparar F2 y F3.

El cerco se estrecha. Ya tenemos todos los elementos necesarios para localizar la clave que buscamos. Si abrimos un archivo reinstalar.reg, p.ej, veremos que se compone de una sección delete que engloba una serie de ramas del registro y de claves que son borradas y de una sección add, donde son añadidas. A nosotros nos interesa, lógicamente, la sección add. En esta sección hay innumerables ramas y claves y una o varias de ellas son las que guardan la fecha. Habrá menos ramas cuanto mas “pulcros” hayamos sido a la hora de usar el ART. Necesitaremos un mataprocesos como el de ProcDump para limpiar el sistema de aplicaciones que no vienen al caso y que hacen uso simultaneo del registro. Tomando dos archivos reinstalar.reg que guarden información de dos fechas distintas y eliminando sucesivamente una clave tras otra para ver si el archivo reinstalar.reg conserva sus propiedades, llegamos a la conclusión de que es esta curiosa clave la que guarda la información que nos interesa:

[HKEY_LOCAL_MACHINE\Software\McAfee\Instant Purchase\2.00\LM\4W7U-RAA]
"Data"=hex:48,43,41,41,41,41,58,45,35,44,4D,41,56,35,34,48,4C,33,53,51,46,57,\4C,4B,35,51,44,4B,47,57,38,55,4C,50,32,59,35,41,41,41,41,41,41,41,41,41,41,\41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,59,50,41,41,\41,41,41,41,41,41,41,41,41,5A,4C,41,41,41,41,41,58,35,48,41,41,41,41,41,41,41,\41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,56,58,4A,41,4C,\41,41,52,41,41,36,41,41,4B,36,41,41,43,41,56,45,41,41,59,57,41,4B,38,50,43,41,\41,43,4C,41,56,41,41,41,37,41,41,41,36,41,48,36,41,41,46,4B,56,4E,41,41,41,41,\41,35,50,47,41,35,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,\41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,\41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,\41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,\41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,\41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,36,41,41,41,41,41,43,4C,\41,41,41,41,56,41,41,41,41,41,41,41,50,46,4B,35,4D,4C,39,36,33,44,56,52,56,4C,\39,35,50,59,56,52,56,4C,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,\41,41,41,41,41,41,41,41,41,41,4B,41,41,41,41,41,41



Esta clave tiene como fecha de instalación 2/Febrero/2002 y por si sola se basta para hacer que el programa se comporte como si hubiera sido instalado en esa fecha aunque no sea así. El resto de claves que componen el archivo reinstalar.reg contienen otro tipo de información, no por ello accesoria.

Aquí hay que evitar la tentación de intentar descifrar la clave ya que, obviamente, esta encriptada en hex (429 cifras). Si fuera más obvia, habría una bonita manera de crakearlo basada en la introducción de código en el espacio libre que queda en los encabezados PE y que nos puede servir para modificar la información del registro cada vez que se carga el programa, adecuándolo a nuestros intereses de uso infinito (tutorial espléndido de un miembro del grupo WKT, perdónenme que ahora no recuerdo). Como no es el caso, deberemos acercarnos a la rutina de comprobación del trial basándonos en la información que tenemos.

... olfateando con LA BESTIA ...

Generalmente prefiere usarse el APISpy para localizar de que línea parte la llamada del registro que lee la clave que nosotros sabemos guarda la información necesaria para comprobar si aún estamos autorizados para utilizar la aplicación. ¿Cómo se hace esto?, pues muy fácil, monitorizando la aplicación con las APIS del registro que están en ADVAPI32. En este caso, no sé muy bien por que, APYSpy no mostró que el registro leyera la clave [HKEY_LOCAL_MACHINE\Software\McAfee\Instant Purchase\2.00\LM\4W7U-RAA], lo cual no es posible como se demuestra monitorizando simultáneamente con Regmonitor, viendo que efectivamente esta clave se lee después de unas 12, las últimas de las cuales son Software\McAfee\Instant Purchase\2.00\EconWr y Software\McAfee\McAfee shared components\Updater. Pero tenemos a la BESTIA (alguien dijo algo de TRW2000?). Lo primero es saber que API vamos a tracear para saber cómo recibe los valores de la pila (aquí recomiendo el magnífico tute de Karpoff, "usos y abusos de la pila en las sesiones de depuración"). Una muy buena ayuda son las crackers notes de la lamentablemente desaparecida Inmortal Descendants:

RegOpenKeyExA / RegOpenKeyExW

La función RegOpenKeyEx abre la clave especificada.

```
LONG RegOpenKeyEx(
  HKEY  hKey,                 // el manipulador de clave abierta
  LPCTSTR  lpszSubKey,        // la dirección del nombre de la subclave a abrir
  DWORD  dwReserved,          // reservado
  REGSAM  samDesired,         // máscara de acceso de seguridad
  PHKEY  phkResult            // la dirección del manipulador de la clave abierta

);
```

Returns
Si la función tiene éxito, el valor de retorno es ERROR_SUCCESS.

Si la función falla, el valor de retorno es un valor de error.

Vemos que es en el penúltimo elemento en el que se pasa el nombre de la clave que se va a abrir a la pila, HKEY hkey (se pasan de derecha a izquierda). Ponemos un BPX en esta API y observamos los siguientes breaks:

1º CPDCLNT

```
    push 401219     <- Software\McAfee\McAfee Firewall
    push (...)
401219 call RegOpenKeyExA
```

2º CPDCLNT

```
    push 407330     <- Software\McAfee\McAfee Shared components
    push (...)
401BA9 call RegOpenKeyExA
```

3º CPDCLNT

```
    push 407330     <- Software\McAfee\McAfee Shared components\central
    push (...)
401BCF call RegOpenKeyExA
```

4º central

```
    push   40034C6C   <- Software\McAfee\McAfee Shared components\central settings
    push (...)
4016D9 call RegOpenKeyExA
```

5º Ole32!txt

```
    push  7FF4E38E  <-  ClSID
    push (...)
7FF4E38E call RegOpenKeyExA
```

6º Ole32!txt

```
    push  7FF4E3DA  <-  Software\Microsoft\Ole
    push (...)
7FF4E3DA call RegOpenKeyExA
```

7º Wininet

```
    push  7612FB20  <-  Remote Access\Addresses
    push (...)
7613BE15 call RegOpenKeyExA
```

8º RNR20

```
    push   7828FAE0   <-   System\Current   Controlset\Control\Service Provider\Service Types
    push (...)
7828BE0 call RegOpenKeyExA
```

9º RNR20
push 78287996 <- System\Current Controlset\Control\Service
push (...)
78287996 call RegOpenKeyExA

10º FWHOME
push ecx <- Software\McAfee\Instant Purchase\2.00\EconWr
push (...)
010E1D76 call RegOpenKeyExA

11º FWHOME
push edi <- Software\McAfee\McAfee shared components\Updater
push (...)
010E1C89 call RegOpenKeyExA

12º Recapi
push edx <- ??? 00,00,11,08,00,00,71 (valores hex)
push (...)
019C3A7B call RegOpenKeyExA

Llegados aquí nos aparece en la ventana de comandos del Sice el siguiente mensaje de depuracion: ---- 02/05/2002 17:29:57 ----------[Entering ECLicense Function, szProduct= MFW, szlanguaje = (null) ECAPI---

Si volvemos a pulsar F5 (recuerden que hacemos un BPX RegOpenKeyExA) sale la ventana de trial acabado, lo cual, junto con el conocimiento por el regmonitor del orden en que se leen las claves, nos hace pensar que esta 12º es la que contiene la que a nosotros nos interesa, [HKEY_LOCAL_MACHINE\Software\McAfee\Instant Purchase\2.00\LM\4W7U-RAA].

Conforme traceamos hacia delante nos vamos dando cuenta de que nos adentramos en las secuencias de lectura de clave. A continuación se ejecutan RegQueryValueExA y RegCloseKey. Mas adelante vemos que se ha metido en eax un puntero al inicio de la secuencia de valores hexadecimales que componen la clave. Posteriormente realiza un buen numero de operaciones complejas en un bucle de 267 ciclos que contiene a su vez varios bucles más pequeños, que empieza en 19C32EE y termina en 19C3379, que es donde se "cuece el tomate" de la clave.

```
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:10003379(U)
|
:100032EE 8B7C2418            mov edi, dword ptr [esp+18]
:100032F2 85FF                test edi, edi
:100032F4 740C                je 10003302
:100032F6 8B44241C            mov eax, dword ptr [esp+1C]
:100032FA 3B08                cmp ecx, dword ptr [eax] <- ecx es el contador, dword ptr [eax]= 10B
:100032FC 0F878D000000            ja 1000338F  <- por aquí se sale del bucle cuando ecx=10C
```

```
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:100032F4(C)
|
:10003302 83FA08              cmp edx, 00000008
:10003305 7D51                jge 10003358

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:1000334F(C)
|
:10003307 85F6                test esi, esi
:10003309 7448                je 10003353
:1000330B 8A4D00              mov cl, byte ptr [ebp+00]
:1000330E 45                  inc ebp
:1000330F 4E                  dec esi
:10003310 80F920              cmp cl, 20    - "marcas" especificas en la clave; marca 20.
:10003313 7437                je 1000334C
:10003315 80F909              cmp cl, 09    - "marca 09"
:10003318 7432                je 1000334C
:1000331A 80F92D              cmp cl, 2D    - "marca 2D"
:1000331D 742D                je 1000334C
:1000331F 80F961              cmp cl, 61    - "marca 61"
:10003322 7C08                jl 1000332C
:10003324 80F97A              cmp cl, 7A    - "marca 7A"
:10003327 7F03                jg 1000332C
:10003329 80C1E0              add cl, E0

* Referenced by a (U)nconditional or (C)onditional Jump at Addresses:
|:10003322(C), :10003327(C)
|
:1000332C 33C0                xor eax, eax

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:1000333C(U)
|
:1000332E 388810F90110        cmp byte ptr [eax+1001F910], cl
:10003334 7408                je 1000333E
:10003336 40                  inc eax
:10003337 83F820              cmp eax, 00000020
:1000333A 7D42                jge 1000337E
:1000333C EBF0                jmp 1000332E

* Referenced by a (U)nconditional or (C)onditional Jump at Address:

|:10003334(C)
|
:1000333E 83F820              cmp eax, 00000020
```

```
:10003341 7D3B                jge 1000337E
:10003343 8BCA                mov ecx, edx
:10003345 D3E0                shl eax, cl
:10003347 0BD8                or ebx, eax
:10003349 83C205              add edx, 00000005

* Referenced by a (U)nconditional or (C)onditional Jump at Addresses:
|:10003313(C), :10003318(C), :1000331D(C)
|
:1000334C 83FA08              cmp edx, 00000008
:1000334F 7CB6                jl 10003307
:10003351 EB05                jmp 10003358

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:10003309(C)
|
:10003353 83FA08              cmp edx, 00000008
:10003356 7C33                jl 1000338B

* Referenced by a (U)nconditional or (C)onditional Jump at Addresses:
|:10003305(C), :10003351(U)
|
:10003358 85FF                test edi, edi
:1000335A 7407                je 10003363
:1000335C 881F                mov byte ptr [edi], bl
:1000335E 47                  inc edi
:1000335F 897C2418            mov dword ptr [esp+18], edi

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:1000335A(C)
|
:10003363 8B4C2414            mov ecx, dword ptr [esp+14]
:10003367 33C0                xor eax, eax
:10003369 41                  inc ecx
:1000336A 8AC7                mov al, bh
:1000336C 83EA08              sub edx, 00000008
:1000336F 894C2414            mov dword ptr [esp+14], ecx
:10003373 85F6                test esi, esi
:10003375 8BD8                mov ebx, eax
:10003377 7416                je 1000338F
:10003379 E970FFFFFF          jmp 100032EE  <- Vuelve al inicio del bucle.

* Referenced by a (U)nconditional or (C)onditional Jump at Addresses:
|:1000333A(C), :10003341(C)
```

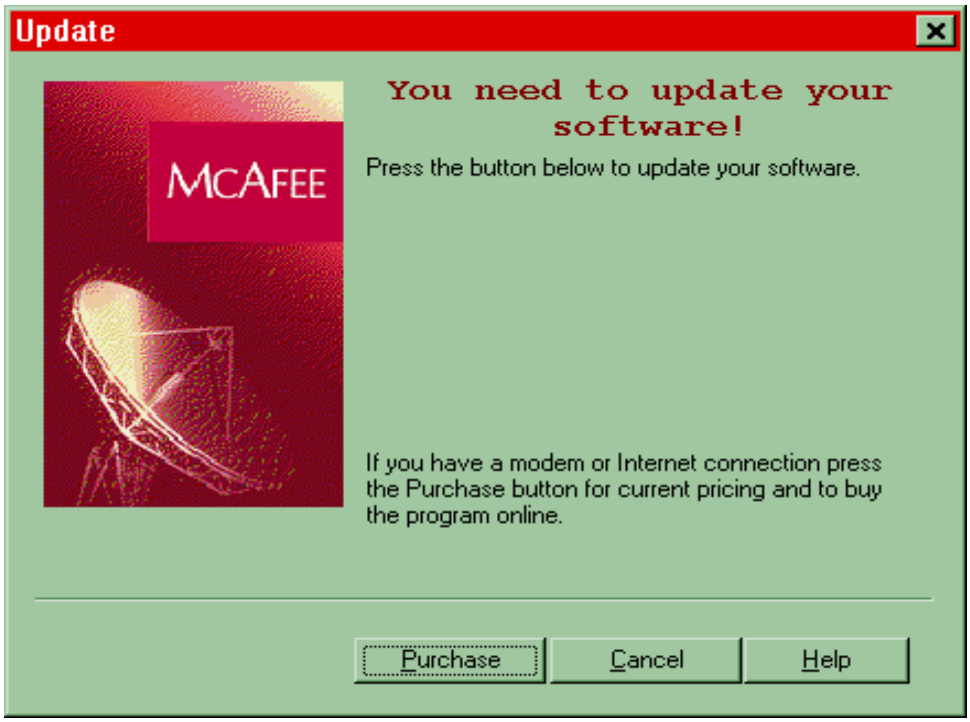
Dentro del bucle va cargando cada elemento hexadecimal de la clave dentro de cl, y realiza varias comprobaciones de cada uno de los elementos con valores "marca", de tal forma que si estan presentes se fuerza la salida del bucle. Varios de estas "marcas" tienen el valor de permitir un uso indefinido del programa, aunque a menudo haga falta reiniciarlo o aparezcan molestas nags, como la de QuickClean de la imagen. Posiblemente estos saltos nos lleven por una región del programa que modifique el registro en el sentido de permitir un uso indefinido, pero no investigué esta posibilidad por encontrarme ya muy cerca de la solución definitiva.

Hacia el inicio del bucle ecx tiene función de contador, y así sabemos cuantas ciclos realiza, conocimiento necesario para saber por donde sale y a donde se dirige. Una vez fuera del bucle, siempre dentro de Recapi, que es una dll incluida dentro de la carpeta C:\Archivos de programa\McAfee\McAfee Shared Components\Instant Updater, llegamos a este interesante trozo de código:

```
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:1000489A(U)
:1000489E 8BC8                  mov ecx, eax
:100048A0 889C2420010000        mov byte ptr [esp+00000120], bl
:100048A7 8906                  mov dword ptr [esi], eax
:100048A9 E802F1FFFF            call 100039B0
:100048AE 8B0E                  mov ecx, dword ptr [esi]
:100048B0 8D442414              lea eax, dword ptr [esp+14]
:100048B4 50                    push eax
:100048B5 E876F5FFFF            call 10003E30
:100048BA 85C0                  test eax, eax  <- ¿Permitir uso indefinido?
:100048BC 745B                  je 10004919    <- No: salta fuera de la zona de "registro"
:100048BE 6824020000            push 00000224
```

```
:100048C3 E8D9F40000            call 10013DA1
:100048C8 83C404                add esp, 00000004
:100048CB 8944240C              mov dword ptr [esp+0C], eax
:100048CF 3BC3                  cmp eax, ebx
:100048D1 C684242001000002      mov byte ptr [esp+00000120], 02
:100048D9 740E                  je 100048E9
:100048DB 8D4C2414              lea ecx, dword ptr [esp+14]
:100048DF 51                    push ecx
:100048E0 8BC8                  mov ecx, eax
:100048E2 E859EEFFFF            call 10003740
:100048E7 EB02                  jmp 100048EB
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:100048D9(C)
```

Nopear este salto vence el límite de una forma mucho más mas limpia que los anteriores, solo deja una ventana que nos insta a realizar un update y que tenemos que cancelar para continuar con el programa.

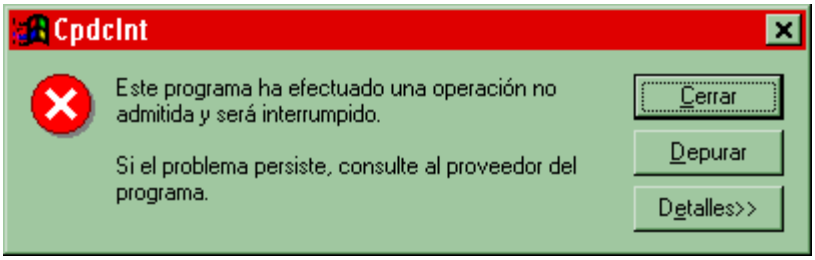


Traceando un poquito vemos que sale de:

```
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:1000539D(C)
|
:10005366 33FF                  xor edi, edi
:10005368 EB4E                  jmp 100053B8
:1000536A 8B0E                  mov ecx, dword ptr [esi]
```

```
:1000536C E8BFECFFFF          call 10004030
:10005371 85C0               test eax, eax
:10005373 8BCE                mov ecx, esi
:10005375 741F               je 10005396   <- Forzando este salto se evita
:10005377 E8F4F6FFFF          call 10004A70 <- Nag
:1000537C 8BF8                mov edi, eax
:1000537E EB38                jmp 100053B8
:10005380 8B0E                mov ecx, dword ptr [esi]
:10005382 E8A9ECFFFF          call 10004030
:10005387 85C0               test eax, eax
:10005389 8BCE                mov ecx, esi
:1000538B 7409               je 10005396   <- Forzando este salto se evita

:1000538D E8DEF6FFFF          call 10004A70  <- de nuevo la nag
:10005392 8BF8               mov edi, eax
:10005394 EB22               jmp 100053B8
```

Oséa, que cambiando estos 3 saltos en Recapi.dll hemos conseguido una versión plenamente funcional de McAfee Firewall 3.0 de uso permanente y exenta de nags. Sin embargo aún queda un apartado que corregir. Hemos evitado la comprobación del trial por que posiblemente entramos al programa por donde este evalúa si las librerías que posee están actualizadas o no, de ahí que nos apareciera esta nag. Posiblemente por esto algunos elementos necesarios para que el programa realice una actualización no se cargan o se cargan mal y como consecuencia al pulsar sobre "Check for a Firewall Update" aparece una molesta ventanita.



Esta ventanita puede localizarse fácilmente poniendo un Bpx Hmencpy y traceando un poquito. Vemos que sale de Rupdate.dll que está en la carpeta C:\McAfee\McAfee Shared Components\Instant Updater, igual que Recapi.dll.

```
* Referenced by a (U)nconditional or (C)onditional Jump at Addresses:
|:10001564(U), :100015A8(U), :100015C9(U), :100015E7(U), :10001605(U)
|:10001623(U)
|
:10001641 E8CA010000          call 10001810
:10001646 50                 push eax

* Reference To: USER32.CreateDialogParamA, Ord:004Fh
                 |
```

```
:10001647 FF1538F30110          Call dword ptr [1001F338]
:1000164D 8BF0                  mov esi, eax
```

Vemos que es un CreateDialogParamA. Viene referenciado de 1001564, que a su vez puede evitarse cambiando un jnz.

```
* Possible StringData Ref from Data Obj ->"Home"
                        |
:10001546 68F0340210            push 100234F0
:1000154B 57                    push edi
:1000154C 33F6                  xor esi, esi
:1000154E E85DDD0000            call 1000F2B0
:10001553 85C0                  test eax, eax
:10001555 7512                  jne 10001569   <- Cambiamos por jmp
:10001557 50                    push eax
:10001558 8B442414              mov eax, dword ptr [esp+14]
:1000155C 68A06E0010            push 10006EA0
:10001561 50                    push eax
:10001562 6A6F                  push 0000006F
:10001564 E9D8000000            jmp 10001641   <- llama al DialogParamA
```

Con esto evitamos que el pulsar sobre update tenga efecto alguno. Hay que decir que en la versión "virgen" pulsar update abre una ventana en la que metemos nuestros datos para ser conectados al servidor de McAfee que nos registra como paso previo a bajar los ficheros de actualización. Registrar un programa no tiene mucho sentido en una lista de cracks, así que esta pequeña amputación del programa no creo que sea echada en falta. Tal vez pueda hacerse algo con el fichero de actualización bajado de internet por otros medios, como se hacia con la actualización del Norton Antivirus (lección 17), pero eso es ya otra guerra.

Muchos procesos se han obviado de este tute. Por ejemplo, si ponemos las características en la tabla de secciones a E0000020 y desensamblamos Recapi.dll, veremos interesantes String References como "Mode evaluation" o "Mode registered" y muchas otras, pero su valor es limitado ya que muchas son flags de depuración del programa (curiosamente incluye mas elementos de depuración de lo habitual, como visteis cuando Sice hace el 12º BPX en RegOpenKeyExA y como veréis muchos más si traceais vosotros el programa por esa zona). Con todo esto no es un programa muy difícil de crackear, los avances van cayendo como fruta madura hasta que termina cantando la gallina.

Lo mejor de crackear el Mc Afee Firewall es que parcheando definitivamente Recapi.dll quedan también automáticamente crackeados el QuickClean y el VirusScan, aunque esta conclusión resulto un poquito ardua de demostrar. Nuevas instalaciones de productos de McAfee "machacan" el archivo parcheado, por lo que debe guardarse en otra carpeta un Recapi.dll parcheado o fabricarse un parcheador.

mR gANDALF

PD: Mis sinceros agradecimientos a Ricardo por su ayuda y apoyo. Deseo que este tute le sirva para medir su labor de enseñanza del craking y sobre todo le deseo que obtenga de la vida todo lo bueno que se merece.

Un fuerte abrazo Ricardo, y espero que este tute sea el último de una serie, que lo será por estupendos motivos para mí.

Ricardo Narvaja

(LECCION SESENTA)

Introducción: Aquí tenemos un Nuevo colaborador a Noseman que con sus 16 años ha hecho un buen tute, de cómo quitar el vencimiento al programa REAL PLAYER ONE en su última versión y está muy bien hecho, y funciona así que agradecemos su colaboración que pasa a engrosar las lecciones del curso.

MUCHAS GRACIAS NOSEMAN

Programa: Real Player One

Protección: No sé, pero después de Agosto 1 de 2002 ya no se sigue ejecutando porque pide un update

Dificultad: Fácil.

Herramientas: SoftICE 4.05.334, W32Dasm 8.93, , HexWorkShop 3.11

Cracker: noseman

Fecha: 06/02/2002

Introducción: Después de mi primer día en la universidad y con las pupilas dilatadas y con la imagen en mi cabeza de que de ahora en adelante voy a tener que utilizar lentes, aquí estoy con mi primer "tutorial" y hasta sorprendido estoy con todo lo que he progresado en esto del cracking, gracias a Ricardo por supuesto, he seguido todas sus lecciones menos la de los programas que sobrepasan las 15Mb (Que pereza bajarme eso). El programa que aquí se trata es el famoso Real Player de Real Networks en su ultima versión que es RealPlayerOne (6.0.10.446), lo crackee por cuenta propia porque me disgustaba que el programita pidiera un update después del 1 de agosto de 2002 y no seguía funcionando mas.

Desguaze: Empezando como siempre analizando el exe del Real Player con el File Inspector XL y el Language 2000, no muestran nada de ponerle atención puesto que no esta empaquetado ni nada por el estilo. Ahora miremos como funciona el programa antes del 1 de agosto.

Vemos que el programa no tiene ningún problema al arrancar y reproduce todo lo que uno le diga (los formatos que soporta), ahora adelantemos el reloj al 1 de agosto.

Vemos que dice que esa versión del real player ha expirado, que te conectes a internet para bajar automáticamente la ultima versión, pero yo estoy muy contento con esta versión, me funciona perfectamente para lo cual esta diseñada y no quiera bajar otra (por pereza en mi caso, o por lo que quieran), que aparte de ser lo mismo pesa mas. Entonces que hacemos???, pues una persona normal no le quedaría de otra que updatear el programa pero como nosotros sabemos algo de esto le encontraremos solución al problema. O no???

Abrimos el exe con el Peditor y vemos que sections en el .data esta en C0000040 entonces ya sabemos que hacer, cambiarlo por E0000020 para abrirlo con el w32dasm, al abrirlo desensambla perfectamente, sin ningún error pero cuando vamos a las string data references solo vemos algo sospechoso que es un "update" pero al darle ahí no nos lleva a ningún lugar interesante, entonces como no estamos orientados habrá que utilizar al sice, no hay de otra.

Lo mas lógico aquí seria un getlocaltime, si traceamos con F12 vemos que para en demasiadas partes, unas en el kernel, otras en dlls del realplayer y muchas mas, simplifiquemos las cosas con otro breakpoint, por ejemplo el hmemcpy que al tracearlo con F12 no para en tantas partes, traceamos hasta que salga el cartelito ese diciendo que actualicemos el programa y cuando le damos close vuelve al sice en esta dir 60001E21 que si vemos no pertenece exactamente al exe del realplayer sino a un dll del mismo que es el rpap3260.dll, vemos que para en test eax, eax y encima hay un call (sospechoso no?), volvamos al sice y ponemos un breakpoint en esta call haber si para, vemos que si para y si entramos al call (F8) para tracearlo con calma hasta que salga el aviso del RP y le damos close vemos que resulta en otro test eax, eax y encima un call (60022AE1), pongamos un bpx alli para ver si por casualidad para, bingo!!!! Si paro, y si lo ejecutamos con F10 de una sale el aviso nefasto ese, o sea que por aquí debe estar la clave del asunto, entremos al call a ver que pasa, nos lleva a un lugar que dice INVALID por todo lado, pero si volvemos a dar F10 nos lleva a nuestras hermosas sentencias, pero si están perezosos como yo y no quieren analizar todo ese código tan largo volvemos al call del principio (60022AE1) y probamos a nopearlo pero al hacerlo sale error por todas partes y tenemos que reiniciar el ordenador, entonces observamos que después de la call sigue un test eax, eax, que tal si la call la cambiamos por un xor eax, eax?, sí si funciona, procedemos a copiar la cadena para ir a buscarla al editor hexadecimal y tendríamos que buscar lo siguiente:

E8 9A 3C 00 00 85 C0 0F 85 B8 06 00 00
Y reemplazar **E8 9A 3C 00 00 => que es el call**
Por
Esto **33 C0 90 90 90 => que es xor eax, eax**

Y listo, no hay que preocuparse mas por el vencimiento del programa.

El método anterior no pordrá ser muy ortodoxo pero sirvió para el propósito que era. Agradecimientos a Ricardo por su gran curso, y disculpas para mr gandalf por haber copiado el esquema de las lecciones de él porque es que en realidad no sabia como hacer el mío.

ATT:

NOSEMAN

Ricardo Narvaja