AGENDA MSD 2.0

(LECCION VEINTIUNO)

A pedido de una amiga VERÓNICA VILLAVERDE que actualizo su programa AGENDA MSD de una versión anterior a la nueva 2.0 y le borró el registro que tenía anteriormente, lo que hizo que buscara el crack en ASTALAVISTA pero ese crack que era un GENERADOR DE CLAVES no funcionaba, entonces me propuse hallar la clave de este programa.
Primero trate de ver el ejecutable AgendaMSD.exe en el WDASM y desensamblarlo pero no sale nada ya que esta comprimido.
Para averiguar que compresor se ha utilizado, uso el UNPACK 2000 que está bastante actualizado ya que el GETTYPE esta un poco desfasado y casi no reconoce nada.
Coloco una copia del ejecutable en la carpeta del UNPACK y salgo a DOS voy hasta el directorio donde esta el unpack en mi caso

D:\Download\herramientas crcak\unpack20

Allí tecleo

un-pack agendamsd.exe

Y después de un montón de información al final me sale

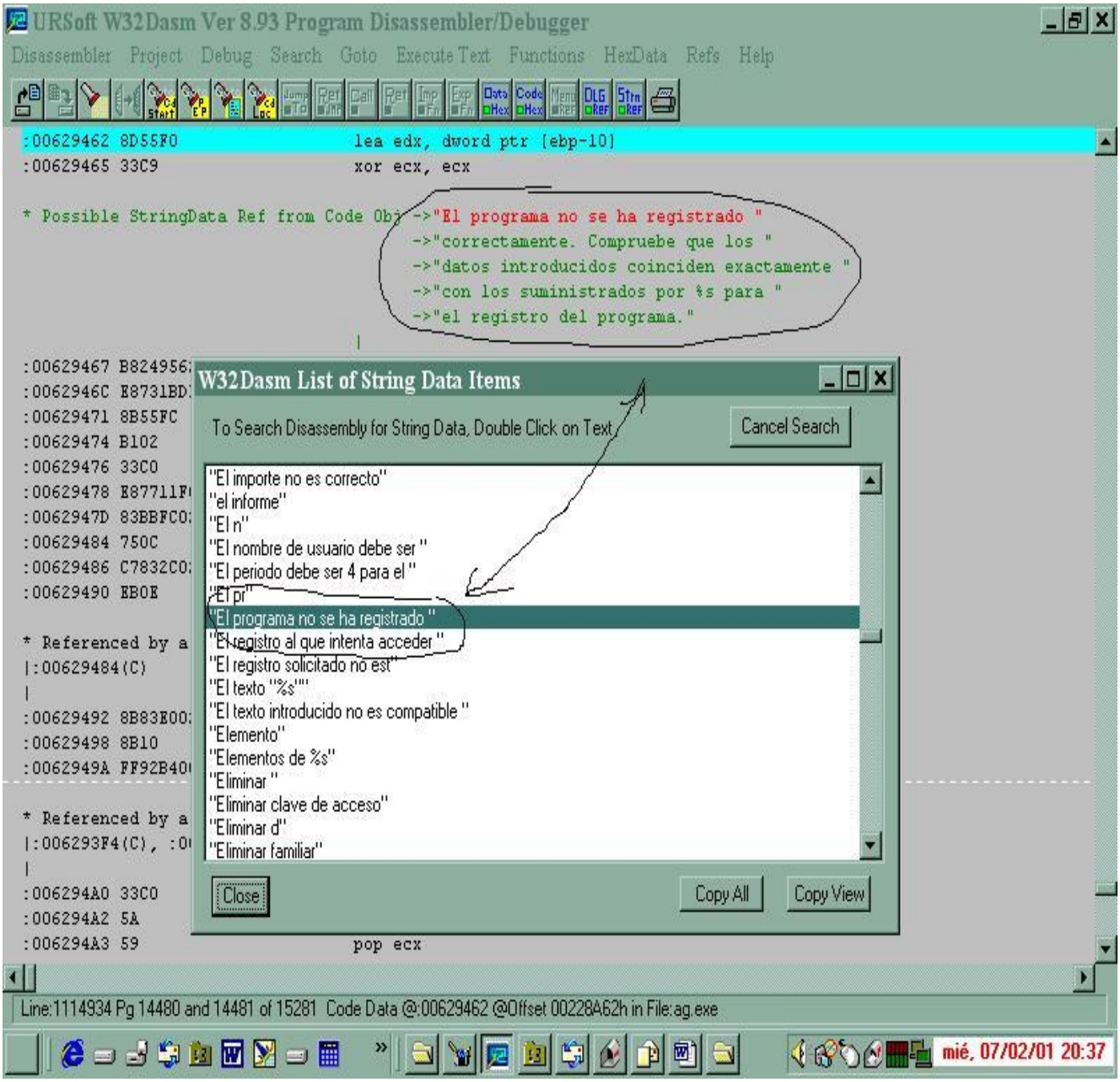
Packed by ASPack v.1.08.04 by Alexey Solodovnikov
Use UnAspack v.1.0.9.1 by BiWeiGuo for unpack

O sea que esta comprimido con ASPACK 108.04 y también me aconseja un descompresora el UNASPACK 1.0.9.1 pero como en el PROCDUMP vi que estaba esa versión del compresor lo haré con PROCDUMP.
Tengo varias versiones del PROCDUMP increíblemente la que me funciono bien es la original ya que otras actualizaciones me daban al descomprimir SCRIPT ERROR.
Bueno lo descomprimo como vimos en la LECCIÓN de descompresión automática con procdump eligiendo ASPACK108.4 y después de un rato de trabajar lo guarda, ojo que hay que quitarle la protección antidesensamblado C0000040 y cambiarla por E0000020 para poder verlo en el WDASM.
Una vez que hice todo eso lo abro con el WDASM, me salen las STRING REFERENCES y allí una muy importante que es el mensaje que aparece cuando pones una clave que no es correcta para registrarte como vemos en la imagen (EL PROGRAMA NO SE HA REGISTRADO CORRECTAMENTE....)

Haciendo doble clic en la STRING REFERENCE y después subiendo un poco la imagen vemos que viene de una

REFERENCE BY A CONDITIONAL....JUMP at address 629430

Que es un poquito mas arriba de donde estamos, allí vemos un salto en 629430 que claramente decide si vamos a estar registrados ya que si salta nos lleva a la ZONA DE CHICO MALO que estábamos antes y si no salta vemos que abajo aparece otro cartelito que dice ENHORABUENA EL PROGRAMA HA SIDO REGISTRADO SATISFACTORIAMENTE.
Bueno ya sabemos cuál es el salto que puede registrarnos, pero ¿podremos ver la clave? Entonces arranquemos la maquina con el SOFTICE y arranquemos el programa, vayamos a MANTENIMIENTO-DESBLOQUEAR EL PROGRAMA y la ventana de registración te pide que pongas un nombre de mas de ocho caracteres y separados por un espacio vacío, podría ser nombre**: narvaja ricardo** o el que quieran y pongan una clave falsa que llene todos los espacios vacíos por ejemplo **9898-9898-9898-9898-9898-9898.**



Antes de tratar de DESBLOQUEAR el programa entremos al SOFTICE y pongamos

BPX HMEMCPY

Luego volvamos a WINDOWS con CTRL.+D esperamos un poquito para ver que no vuela a romper el softice solo y aceptamos para que ingrese la clave lo que rompe en el SOFTICE nuevamente, tecleamos F12 tantas veces como sea necesario hasta que en la línea verde del SOFTICE hayamos vuelto al ejecutable AGENDA.
Vamos a poner un BPX en 62940E un poco antes de el salto para ver si aparece por ahí la clave verdadera una vez que para ahí voy traceando y haciendo D ... para ver si aparece la clave y me aparece así.
Traceo hasta 629423 allí hago D EAX y aparece mi nombre ricardo narvaja, ejecuto con F10 hasta que llego al salto, allí bajo un poco la ventana de datos para ver si hay una secuencia de números o letras de 24 cifras, si no aparece pongo X y repito el proceso, a la tercera vez a mi me apareció siempre, en 629423 hice D EAX y apareció mi nombre y seguí traceando con f10 hasta llegar al salto y allí fui bajando la ventana de datos para ver debajo de mi nombre y 20 o 30 líneas mas abajo apareció la clave de 24 letras y números, ojo que son mayúsculas y minúsculas y hay que copiarla bien, porque una i parece una I o un 1, son parecidos fíjense bien.

Bueno ya hallamos la clave para la agenda MSD 2.0 un saludo, hasta la próxima…

Para mi nombre empieza con 1jN4-9P13-11x2-0Hp6-1w12-..... y los cuatro últimos descúbranlos ustedes a ver quien manda primero la clave MIA completa.

Ricardo Narvaja

CIBER-BOSS 2.1

(LECCION VEINTIDOS)

PROGRAMA PARA CIBER CAFES

Recibí un mail de un amigo que me decía lo siguiente:

> Ricardo:
>
> Estimado amigo...estoy intentando instalarme con un CyberCafé en la ciudad de .....
>
> Y para llevar un buen control, tanto del tiempo, como el de las impresoras y Scanner, encontré un excelente programa en Español...se llama **"Ciber Boss 2.1"...** está en la siguiente dirección:
>
> http://www.jm-soft.com

Bueno ya que parece interesante fuimos a ver de qué se trata hicimos un crack y como yo no tengo CyberCafé no lo puedo probar pero como no hubo quejas supongo que funcionó bien.
La primera vez que lo ejecuto puedo saltear la identificación del usuario con aceptar solamente pero es conveniente que una vez dentro, vayamos a MAQUINA SERVIDOR - MANTENIMIENTO y allí vayamos a MANTENIMIENTO DE EMPLEADOS y editemos el que aparece allí con grado MASTER y le pongamos una clave y un login que debemos recordar ya que después de unas veces de usarlo si no lo hicimos no nos dejara entrar más, también se pueden configurar aquí los empleados que van a usar el programa y ponerles una clave y login y nivel OPERADOR, y así sucesivamente.

Una vez bajado el programa veo que el ejecutable se llama servidor.exe y que se puede desensamblar con el WDASM y busco las STRING REFERENCES de la ventana que me aparece al empezar el programa luego de colocar la clave
Dicha ventana dice:

VERSIÓN SHAREWARE DE CIBER BOSS

Usted está utilizando una versión shareware (PROBAR ANTES DE COMPRAR) de CIBER BOSS.....

Bueno, este cartelito aparece solo en esta versión sin registrar así que si encontramos la STRING REFERENCE quizás podamos ver donde el programa decide si tiene que aparecer esta ventanita o no tiene que aparecer (REGISTRADO).

Abrimos el WDASM y vamos a IMP FN (IMPORTED FUNCION) y allí vemos que las referencias son todas al archivo MSVBVM60 que es de VISUAL Basic así que cerramos este WDASM y abrimos el que tenemos modificado para ver STRINGS REFERENCES de Visual Basic. Allí en las STRING REFERENCES aparece esta:

"USTED ESTA UTILIZANDO UNA VERSI" obviamente la escondió un poco pero no lo suficiente, hago doble clic encima de esta STRING y aparecemos en 447b81 y arriba justo la STRING está.
Vamos mirando hacia arriba de esa posición y el primer salto condicional que encontramos que puede evitar este cartelito ya que lo saltea por encima se encuentra en 447aa0.

447a97 movsx edx, word ptr [4b2094]
447a9e test edx, edx
447aa0 je 447da5

O sea que según el valor que hay en 4b2094 el cual es transferido a EDX, luego compara edx si es cero y si es cero salta por encima del cartel a 447da5

Lo que tenemos que hacer aquí para registrarnos es modificar un poquito estas sentencias para mover a 4b2094 un valor de cero para que más adelante el programa si vuelve a evaluar si estamos registrados encuentre un cero allí y crea que lo estamos.

Podemos probar con el SOFTICE abrir el SYMBOL LOADER y arrancar el programa y poner un BPX en 447a97 y cuando para allí, hacemos D 4b2094 y vemos que en las dos primeras cifras hay **FF FF** hacemos clic con el mouse en la primera y reemplazamos ambas por **00 00** ya que testea las dos.
ejecutamos con f10 y llegamos hasta el salto y vemos que marca JUMP o sea que va a saltar, tecleamos X y el programa arranca normalmente como si estuviéramos registrados y no aparece la ventana maldita ni tampoco cuando cerramos el programa nos aparece la otra ventana que dice si lo queremos comprar, entonces ya está listo.
Lo que tenemos que hacer es una vez que para en 447a97 teclear A y ENTER y escribir estas dos sentencias

447a97 mov word PTR [4b2094],0000
447aa0 jmp 447da5

y si lo ejecutamos realiza el mismo trabajo de guardar el doble cero en 4b2094 y arrancar como REGISTRADO.
Vemos que usamos solo dos sentencias ya que son más largas que las originales.
Copiamos la cadena que comienza en 447a97

66C70594204B000000E900030000

Y la original era

0FBF1594204B0085D20F84FF020000

Esta última la buscamos en el ultra edit, la reemplazamos por la de arriba y programa crackeado.

Ricardo Narvaja

UNA FACIL

(LECCION VEINTITRÉS)

Ya que esta semana trabaje mucho y estoy muuuy cansado, voy en esta lección a hacer un crack fácil dentro de todo. Es el programa **TURBONOTE 4.5** se baja de

http://turbonote.com/

Y una vez que lo abrimos y vamos a GENERAL SETTINGS está la opción REGISTER TURBO NET donde se puede poner una clave y al ACEPTAR abajo si no es correcta desaparece la ventana sin salir ningún cartelito para buscar en el WDASM. Abro el WDASM y trato de abrir el ejecutable tbnote.exe y se cuelga la máquina hmmm, que cosa, podría ponerme a descomprimirlo pero voy a usar el PUPE para rápidamente hacer un archivo que pueda desensamblar con el WDASM.
Ejecuto el Turbonote y cuando ya arranco abro el PUPE, con el elijo en la lista de procesos el que corresponde a TURBONOTE y hago clic derecho y elijo CAJA DE HERRAMIENTAS, allí vuelvo a elegir entre los módulos el ejecutable TBNOTE y hago clic derecho para cargarlo y después voy a VOLCADOR TOTAL- ALINEAR y VOLCAR y guardo el ejecutable descomprimido que aunque no funcione me va a servir para ver en el WDASM.
Por supuesto antes tengo que usar el PROCDUMP para cambiar las características de las secciones y eliminar la protección Antidesensamblado como vimos en las lecciones anteriores aquí hay que cambiar 60000020 por E0000020 y ya se desensambla con el WDASM.
Una vez que se desensambla miramos las STRING REFERENCES y si miramos en el listado del programa bajando encontramos con un poco de paciencia en **412d08 "THANKS FOR REGISTERING TURBONOTE+ "**,lo cual es una punta para empezar.
Vemos que eso viene de una REFERENCED BY... que está un poco más arriba que dice 412cb4, vamos allí con GOTO CODE LOCATION.
Y vemos el salto que nos llevaría supuestamente a REGISTRARNOS pero si probamos poner un BPX en el SOFTICE ,y ponemos una clave de registro , no para allí quiere decir que aquí no llega por alguna razón, sigamos hacia atrás.
El REFERENCED que esta arriba de donde nos encontramos dice 412c85 vamos allí y vemos también un salto que si no salta nos aparecería un cartelito que dice THERE IS A PROBLEM SAVING THE REGISTRY KEY, sigamos más hacia arriba, bastante más arriba en 412bf3 hay una comparación de EAX con 14 justo después de una función que se ve en azul LSTRLENA que es una función que te dice el largo de una STRING o sea que esto puede ser que verifica la cantidad de cifras de nuestra clave y si no es mayor de 14 hexadecimal o sea 20 decimal nos tira fuera de esta parte de registración.
Vamos al programa y probemos poner una clave de 20 o más cifras y ahora al ACEPTAR dice **I'M SORRY THATS NOT A VALID REGISTRATION CODE, bueno ya logramos algo saber que la clave tiene 20 o mas cifras.**
Ahora podemos poner un BPX en 412c85 para invertir el salto a ver que pasa con el SOFTICE y cuando invierto el salto me aparece el CARTEL de THANKS FOR REGISTERING pero no nos registra sigue diciendo UNREGISTERED por todos lados, allí mismo en la ventana de registro y en ABOUT también.

Generalmente cuando ocurre esto es porque hay que mirar el CALL que esta antes del salto en 412c7e , entramos dentro del CALL poniendo en el SOFTICE un BPX en 412c7e y cuando para allí entro con T y traceo por allí hasta que encuentro un salto, lo invierto a ver qué pasa y voy probando los distintos saltos que encuentro a ver si alguno me registra, y también mirando dentro de los CALLS que hay dentro e invirtiendo los saltos, al final al llegar a

44BC29 75 4E jne 44bc79

Este es el salto clave si lo invierto me aparece el cartelito de registrado, pero si dejo el BPX allí en 44bc29 veo que accede al salto en varios lugares del programa por los que haremos un cargador con el RISC PROCESS PATCHER para que cargue el programa y modifique ese salto en la memoria así siempre estaremos registrados.
Tengo que hacer que en 44bc29 no realice el salto o sea que tengo que reemplazar los bytes originales 75 y 4e por 90 90 para que pase de largo. El script del RISC es este:

;INTERNET DESK

F=tbnote.exe:
O=loaderturbonote.exe:
P=44bc29/75,4e/90,90:
;
$;end of script

Y listo cargamos el programa con el LOADER y vamos a registro y ponemos un numero de más de 20 cifras y ya está, lo que si hay que cargar siempre el programa desde el LOADER así que hay que borrar en INICIO-PROGRAMAS-INICIO la referencia a TURBONOTE para que no arranque cuando inicia WINDOWS y poner un acceso directo al LOADER si queremos que arranque, y borrar los accesos directos del escritorio al ejecutable y hacer accesos directos al LOADER, inclusive se puede cambiar el icono del ACCESO DIRECTO para que tenga el mismo icono que el TBNOTE para que quede más lindo.

Ricardo Narvaja

EL MALDITO ASPROTECT

(LECCION VEINTICUATRO)

La verdad ya me tenía bastante podrido el ASPROTECT este, hace un par de meses me habían alcanzado un programa protegido con este compresor maldito y a pesar de encontrar un par de saltos que invirtiéndolos se REGISTRARIA el programa, no pude hacer nada.
PORQUE?
Porque este compresor diseñado por el mismo que hizo el ASPACK es una fortaleza llevada al máximo de su expresión.
EJEMPLOS:
Protección ANTISOFTICE
Protección Antidesensamblado
Compresor no reconocido por la mayoría de los programas identificadores de compresores ya que este loco hizo de la misma versión miles de variaciones y existe una colección de distintos ASPROTECTS que hacen difícil su reconocimiento y varias versiones distintas con muchísimas modificaciones, es mas este programa yo lo pude crackear y todavía no se verdaderamente que versión de ASPROTECT es la que utiliza.
Si lo descomprimís manualmente como hicimos en la LECCIÓN sobre ese tema, te sale un ejecutable que convenientemente modificado (E0000020) puede ser desensamblado pero no funciona, ya que destruye la tabla de importaciones y da error.
Bueno dirá alguno si no se puede descomprimir le aplicamos un LOADER como el RISC PROCESS PATCHER, craso error si haces eso, te aparece un ERROR DE PROTECCIÓN 15 y no arranca si lo arrancaste desde un LOADER, probé con otros PARCHEADORES como el PELG que dice que funciona en ASPROTECT y nada no lo detecta pero tampoco lo parchea.
Probé muchísimas herramientas que había por ahí y ninguna funcionaba, no quería caer en esos tutoriales complicados que leí sobre el tema que dicen que tenés que rescribir medio ejecutable o inyectar a mano la tabla de importación que es un lío, igual lo intente hacer y no me funciono el ejecutable resultante.
Así que hoy un tiempo después me encontré con otro engendro diabólico de ese ASPROTECT el programa ADVANCED DIRECT REMAILER versión 2.0, desde ya los cracks que hay en Internet a mi no me funcionaron, hay un LOADER por ahí que dicen que funciona pero a mi me sigue diciendo UNREGISTERED y eso no me gusta , no esta muy bien hecho.
Bueno manos a la obra después de tan extenso preámbulo para que conozcan al rival, es un rival de fuste quizás demasiado para mi verdaderamente y es probable que me noquee en el primer round pero quizás como DAVID Y GOLIAT este humilde cracker lo pueda hacer caer y vencerlo de alguna forma quizás mas por insistencia que por conocimiento.
Lo primero es detectarlo, ni el unpack 2000 lo detecta, a este lo detecta el LANGUAJE 2000, el único que dice algo que te puede orientar.
Lo cargo con el LANGUAJE 2000 pongo ANALIZAR y me sale como resultado:

ASProtect/ASPack

Autor: Alexey Solodovnikov

No es mucho, no dice la versión ni nada y no distingue si es ASPack o ASProtect pero dado que ya vimos que los tipos ASPack los detecta el Unpack fácilmente y a este no, concluimos que es ASProtect y no sé qué versión será.

Vamos primero a ver, no se puede desensamblar, así que arranco el programa, arranco el PUPE y como vimos en las lecciones anteriores pongo ALINEAR y hago un VOLCADO TOTAL, quito la protección Antidesensamblado (RECORDAR E0000020 de lecciones anteriores) y aunque no funciona al ejecutarlo, se puede desensamblar con el WDASM.
Vamos al programa , hay una parte para registrarlo que hay que poner una clave, pongo una clave falsa a ver que dice, sale un cartel que se mueve para todos lados (QUE RARO NO?) que me dice

THE CODE YOU'VE ENTERED IS INVALID

Bueno pongo cualquier clave y pongo un BPX HMEMCPY (EL SOFTICE MODIFICADO CON EL BACKDOOR KEEPER NO ES DETECTADO), y cuando hago ACEPTAR, caigo en el SOFTICE hago F12 hasta que vuelvo al ejecutable ADR, eso es en

430601

Voy a probar si falta mucho para llegar al cartelito de que el código es invalido , tengo apretada F10 hasta que salte el cartelito, una vez que aparece el cartel, pongo ACEPTAR y vuelvo al SOFTICE, a ver donde salió el cartelito, o sea la sentencia anterior a donde estoy,

(ACLARACIÓN EN EL SOFTICE NO CONVIENE TENER APRETADO MUCHO TIEMPO SEGUIDO SINO UN RATITO Y SOLTAR Y ASI, YA QUE SI UNO TIENE MUCHO TIEMPO APRETADO CUANDO SUELTA SIGUE EJECUTANDO SOLO MUCHAS SENTENCIAS Y NO PARA)
Bueno la sentencia anterior adonde estoy es

430836 CALL [0043c3c4]

Puedo borrar con BC* los BPX anteriores y poner un BPX allí en ese CALL y volver al programa a la parte de REGISTRO para ver si para antes de que salga justo la ventanita maldita.
Pongo BPX 430836 y hago X enter y vuelvo al programa voy a REGISTRO pongo otra clave falsa y cuando acepto veo la ventana agitarse y justo cuando va a salir el mensaje cae en el SOFTICE jeje.

Bueno un poco más arriba hay un RET y un poco más arriba un CALL un TEST y un salto que saltea justo ese RET lo que puede hacer que si no salta no llegue al cartel maldito y se desvié el programa en el RET hacia otro lado, probemos.

4307fA jz 430827

Poniendo un BPX en 4307fa

Y volvamos a repetir el proceso, cuando el softice para allí en el salto, me indica JUMP o sea que va a saltar y tirarme a la zona del cartel maldito o sea que si hago
R EIP=4307fC para que siga ejecutándose en la siguiente sentencia y no salte

Hago X y ENTER

THANKS YOU FOR REGISTERING ADR

quiere decir que vamos por el camino correcto aunque el programa no nos registra solo aparece el cartel pero sabemos que aquí está la clave del asunto.

Bueno ya vimos que hay un CALL en 4307f0 que es el CALL comparador que devuelve EAX=0 si es la clave incorrecta y UNO si es correcta si cambio **R EAX=1** antes de la sentencia TEST también va al cartel de REGISTRADO.

No ingreso a buscar la clave porque está bastante encriptada y como soy vago, jaja , hago lo más fácil supuestamente.
Voy al WDASM y voy a GOTO CODE LOACION 4307f0 y veo que si ingreso dentro del CALL hay referencias de tres lugares distintos que llaman a este CALL posiblemente para ver si estás bien REGISTRADO, podríamos en vez de parchear todos los saltos que hay en esos sitios, parchear directamente en algún lado dentro de este CALL para que siempre me devuelva UNO sea lo que sea lo que compare, y que mejor que justo cuando finaliza el CALL justo antes del RET agregar algunas cosillas para que de aquí salga siendo siempre EAX=1 y en cualquier lado que llame a este CALL nos diga que estamos registrados.
RECORDAR QUE SI EAX=1 al salir del CALL significa REGISTRADOS, vemos que el RET donde termina este CALL está en 4303EA RET y que después hay varios NOPs (CINCO EXACTAMENTE) que bueno lugar vacío, jajaja.
Puedo entonces allí que hay lugar escribir en

4303EA ADD eax, 01
4303EF RET

Lo que cabe justo ya que ocupa los seis lugares que había y le suma uno a eax y después hace el ret, o sea que si eax aquí llego siendo cero, le sumo uno y siempre valdrá uno al salir del call, lo lea desde donde lo lea.

O sea que si el programa se pudiera parchear con el ultraedit con buscar la cadena c39090909090 y reemplazarla por 0501000000c3 estaría resuelto el crack.
Pero como demonios podemos modificar este ejecutable si esta comprimido, y no podemos modificarlo tampoco desde un loader como el risc process patcher o similar.

Estamos a 27 de febrero del 2001 cuando yo escribo este tutorial, ya mencione a ese RUSO que escribió el ASProtect se ve que es bastante bueno en esto, pero como él es bueno y a mí me derroto, hay otros programadores que también son buenos en hacer herramientas para batir al ASProtect, y esta que uso yo, salió solo hace 5 días en la versión que me funciono y descomprimió bastante bien el ASPROTECT.

Es el CASPR 1.00 lo busque y lo encontré entre muchísimos que probé, es un utilitario muy simple

Aquí transcribo lo que dice el README del CASPR

ASProtect is a very powerful win32 protector. It has compression,encryption, anti-debugging code,anti-disassembler code,crc checking,anti-dumping, etc. features. It uses also strong cryptographic algorithms like Blowfish,Twofish, TEA,etc, and also uses RSA1024 - very strong public keys cryptographic algorithm - as registration keys generation. It communicates with the application through API hooks including import hooks(GPA hook) and export hooks. Furthermore, since v1.1 ASProtect uses the Win95.Marburg's polymorphic engine.ASProtect v1.11c was added anti-Apihook code(stolen from Win32.Crypto) and the BPE32's polymorphic engine.
The big question: Is it a super virus or a commercial protector?

CASPR is a cool unpacker for ASProducts. It is designed for experienced crackers, not for newbies. Sorry, its source code can't be provided.

II. Features :
^^^^^^^^^^^^^^

- **Support for ASProtect v0.95b - v1.2x and ASPack v1.00b - v2.11x**
- **Own decryption and decompression code**
- **Import table rebuilding engine**
- **Export table rebuilding engine**
- **Resources recovering engine**
- **x86 CPU emulator**
- **Win32(Win9x/Me/NT/2k/XP) compatible**

III. Usage :
^^^^^^^^^^^^

CASPR has no GUI, you must run it under command line:

CASPR [-|/L] <input> [[-|/P] output]

Bueno ya vemos allí todo lo que tiene el ASProtect y después las versiones que soporta, inclusive soporta varias versiones de ASPACK

Es muy fácil de usar hay que copiar el ejecutable a descomprimir al directorio del CASPR entrar en MODO DOS y allí teclear

CASPR ADR.exe

Y listo sale esto en la pantalla de DOS

TECLEE

>caspr adr.exe

CASPR v1.000 - A cool unpacker for ASProducts
Copyright (c) 2000,2001 Coded by SAC/UG2001! aLL rIGHTS rEVERSED!

Unpacking Adr.exe...OK!

Don't forget to see readme.txt. It's helpful to reverse ASProducts.

D:\Download\herramientas crcak\ASPACK\caspr100>

Y listo me hizo un archivito llamado ADR.ex que si lo renombro ya esta limpiado de toda la protección ASProtect y funciona solo hay un detalle que arreglar. El programa arranca y sale una messagebox que dice CRYPT API NOT FOUND, pero es una messagebox colocada allí por si alguien llega a descomprimir el ejecutable, hay dos de estas una cuando comienza el programa y si parcheamos esa el programa arranca, la otra esta cuando entro a GENERAL SETTINGS.
No hay problema con el SOFTICE ponemos BPX MESSAGEBOXA y vemos que en la primera después de aceptar y volver con F12 vemos que el CALL al MESSAGEBOXA esta en 42f660 y el salto para que pase por encima esta en
42f651 75 16 jnz 42f669
Con el ULTRA EDIT cambiamos este salto por un JMP o sea el 75 por un EB ahora lo podemos hacer porque el programa esta descomprimido. Y ARRANCA perfectamente y si le hacemos el PARCHE en el CALL que habíamos estudiado en
4303ea
Arranca registrado perfectamente. Queda solamente con el mismo mecanismo que con el primer cartelito entrar a GENERAL SETTINGS y cuando rompe en el MESSAGEBOXA parchear el salto que esquiva el CALL este está en 425c2b, y lo reemplazamos por un
JMP 425cb6
Lo editamos con el ULTRA EDIT y funciona perfecto. Obviamente pudimos con el gracias a dos cosas EL CASPR y la persistencia en buscar una solución que siempre tenemos que tener los crackers, ser tozudos, buscar y buscar y buscar que de alguna forma se va a abrir y va a poder ser crackeado.

PUUFFF… CAYÓ GOLIAT.

Ricardo Narvaja

SORPRESA CON EL ADR 2.0

(LECCION VEINTICINCO)

Bueno como vimos en la LECCIÓN 24 ya parecía que el ADR 2,0 estaba completamente crackeado , pero nos tenia deparada una sorpresa, a pesar de que decía REGISTRADO por todos lados y que parecía que estaba 100 % funcional, no era así, ya que a los quince días de usarlo cuando mandabas un mail por el OUTLOOK EXPRESS o cualquier programa similar si tenias configurado el ADR para trabajar mandando los mails, aparecía un cartel de que estaba vencido que ya habían pasado los quince días de prueba y que tenias que comprarlo para seguir usándolo y se cerraba el programa sin enviar el mail.

Si lo abrías de nuevo, ahí recién enviaba el mail anterior pero si enviabas uno nuevo, de nuevo el cartelito molesto y el programa se cerraba de nuevo, tornándose molesto. Bueno crackearlo no fue nada fácil y hubo que improvisar y salirse del método ortodoxo de crackeo (CUAL SERA NO?) pero bueno lo pudimos crackear y costó bastante, vamos a aprender otra forma de encontrar el salto para parchear, el método es un poco manual y de prueba y error pero funciono y eso es lo que importa. (TODO SE VALE EN LA GUERRA Y EN LOS CRACKS, jaja).

Veamos el cartel dichoso, no es una messageboxa, puesto que es más complejo que un simple cartelito de mensaje, tiene varios botones para conectarse para registrarse, poner la clave, etc, tiene como cuatro o cinco botones no recuerdo bien, así que messageboxa no es, podría ser una dialogboxparama , puesto que este tipo de cartelitos pueden ser un poco más complejos, que mejor que utilizar el softice para comprobarlo, ponemos BPX DIALOGBOXPARAMA y cuando aparece el cartelito caemos en el softice, lo que quiere decir que si es DIALOGBOXPARAMA.

Igual vemos que cuando aparece el cartelito ya desapareció la flecha del ADR en la barra de tareas así que el programa va camino a cerrarse, y cuando hacemos F12 en la ventana se cierra el programa después de unas poquitas sentencias, lo que hace que no podamos llegar al RET y terminar el CALL y volver al programa principal, lo que ocurre es que cuando detecta que está vencido toma un camino el programa sin retorno, sale la ventana y después se cierra así que tenemos que ir hacia atrás en el programa hasta donde todavía este la flechita en la BARRA DE TAREAS y tome la decisión de que aparezca la ventana o que siga funcionando normalmente.

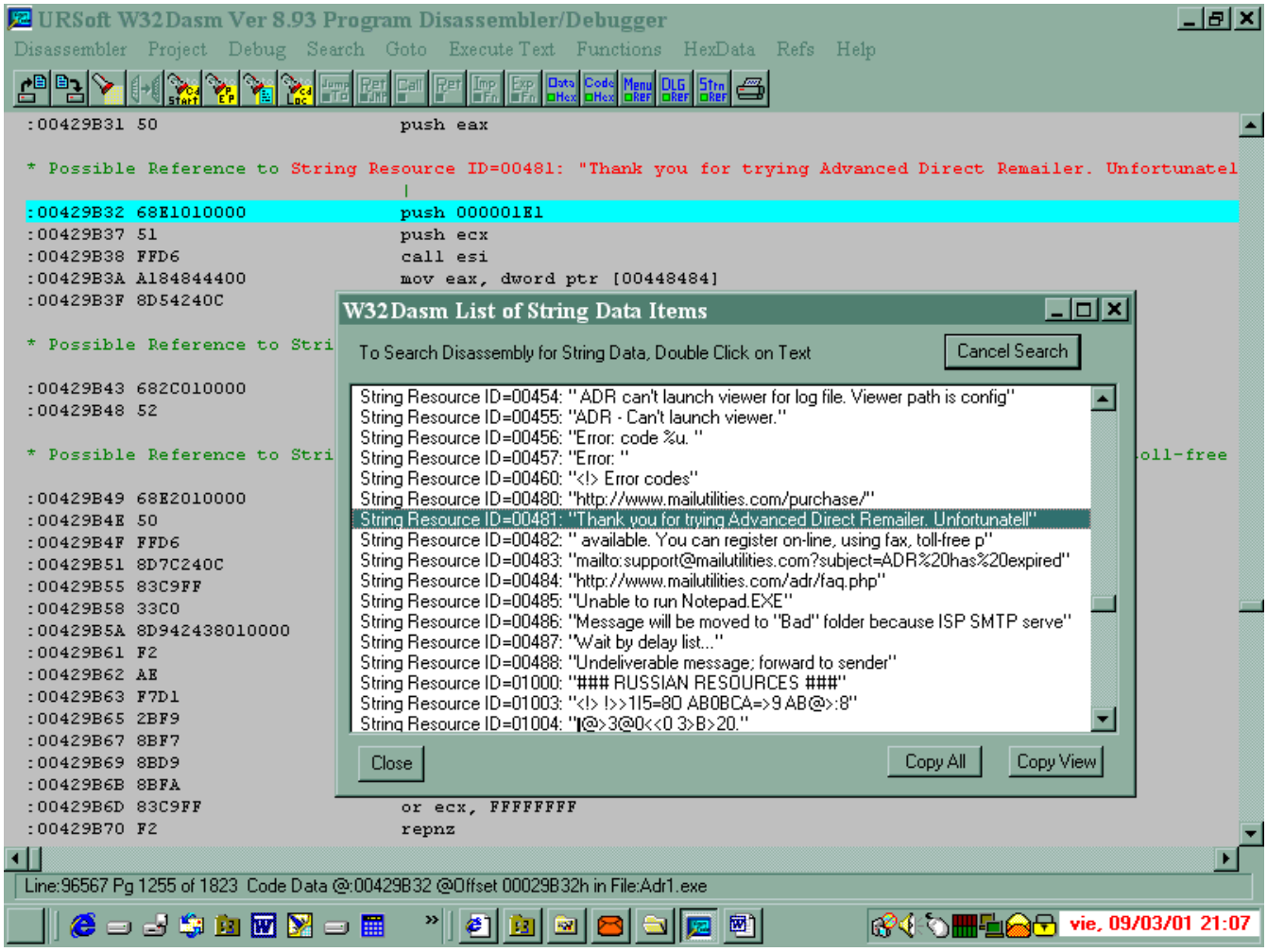
Esto es hacia atrás en el programa, y parece que bastante, veremos. Ya que el programa esta descomprimido podemos ver si la STRING que aparece en la ventana aparece en el WDASM dentro de las STRING REFERENCE, y allí esta la STRING **"THANK YOU FOR TRYING ADVANCED DIRECT REMAILER- UNFORTUNATE** … se te venció, jaja", y chau se cerró el programa.

Si hago clic encima de esa STRING aparezco donde muestra la imagen, en rojo se ve exactamente donde el programa utiliza la STRING esa. Si subimos un poco en el WDASM encontramos el REFERENCED de donde viene el programa es la dirección

*** Referenced by a CALL at Address:**
|:00429CD9
|

```
:00429B10    8B0D84844400          mov ecx, dword ptr [00448484]
:00429B16    81EC84030000          sub esp, 00000384
:00429B1C    8D84242C010000        lea eax, dword ptr [esp+0000012C]
:00429B23    53                    push ebx
:00429B24    56                    push esi
:00429B25    8B358CC34300          mov esi, dword ptr [0043C38C]
:00429B2B    57                    push edi
:00429B2C    858020000             push 00000258
:00429B31    50                    push eax
```

- **Possible Reference to String Resource ID=00481: "Thank you for trying Advanced Direct Remailer. Unfortunatell"**



Es decir, viene de 429CD9. Vamos a GOTO CODE LOCATION 429CD9 y vemos las siguientes líneas

```
:00429CD9   E832FEFFFF          call 00429B10
:00429CDE   83C40C              add esp, 0000000C
:00429CE1   C21000              ret 0010
```

Vemos que allí no hay ningún salto antes de ese CALL entonces seguimos hasta el REFERENCED BY A CALL que está un poco más arriba

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

00429C9E

Vamos a GOTO CODE LOCATION 429C9E y hay un salto

```
:00429C9E 742A              je 00429CCA
```

Si ponemos en el SOFTICE un BPX 429C9E y cuando mandamos un email desde el outlook express nos cae dentro del softice y para justo antes de que aparezca la ventana, y encima para varias veces en el mismo salto antes de que salga la ventana maldita, pero igual nos damos cuenta que el salto este no nos sirve ya que cuando para por primera vez allí, si miramos con f4 como está la pantalla de windows, vemos que ya desapareció la flechita en la barra de tareas, y nosotros tenemos que buscar un salto que cuando pare allí todavía la flechita este y no haya desaparecido, por lo tanto debemos ir todavía más atrás en el programa, volvemos al softice con f4 nuevamente y volvemos al WDASM. Miramos en el Wdasm un poco más arriba del salto anterior y vemos otro REFERENCED...

```
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:00429C0C(C)
|
:00429C71 C3                ret
```

Pero si el programa viniese de allí encontraría como siguiente sentencia un RET y no llegaría al salto, estamos perdidos, el programa desensamblado no nos dice de donde viene y no podemos seguir hacia atrás.

COMO PUEDE SER ESTO

Es muy fácil el programa tiene una sentencia que es la que lleva a esos NOP que están arriba del salto pero el WDASM no muestra una REFERENCED..... ya que puede ser una sentencia tipo

CALL eip o alguna sentencia en la cual no está determinado el valor, por lo tanto cuando encuentra esta sentencia el WDASM no puede crear una REFERENCE ya que no sabe

cuánto va a valer EIP, eso solo se puede saber ejecutando el programa o con el SOFTICE el WDASM no sabe los valores de los registros. Es decir que para crear una REFERENCE el WDASM debería encontrar una sentencia tipo

CALL 45890
JMP 34789

En la cual está determinado a donde va a ir el programa en cambio las sentencias.

CALL EIP
CALL (ESI+EDI)
JMP EAX

O sentencias donde no sabemos cuánto vale EIP, ESI +EDI o EAX, no podemos saber donde va a ir salvo que pongamos un BPX en el softice y al parar en esas sentencias veamos el valor que no conocemos. (Estas sentencias indeterminadas son muy usadas para despistar crackers)

Por lo tanto si con el WDASM no podemos ir mas atrás ¿cómo podemos hacer?

Aquí usaremos métodos menos ortodoxos pero que pueden servir. Tenemos que poner un **BPX 429C9E** y cuando para allí, utilizamos un comando del SOFTICE muy importante para vencer estos truquitos que no nos dejan ver lo que pasa hacia atrás en el programa.

BPR INICIO FINAL T

Es decir INICIO Y FINAL deberían ser las direcciones de inicio y final del programa pero si uno sabe más o menos en que valores está trabajando el programa, se puede acotar a un valor menor en este caso yo puse

BPR 420000 430000 T

Lo que hace la letra T que le agregamos al BPR es que el SOFTICE memoriza las sentencias anteriores al BPX en donde pare o sea que si esta activo

BPR 420000 430000 T

Y

BPX 429C9E

Cuando pare en este ultimo BPX tendremos en la memoria de la maquina las sentencias anteriores que vino ejecutando el programa una por una.
Una vez que para en el BPX tecleo

SHOW 1

Y me va a mostrar la sentencia exacta donde estoy o sea 429c9e y si presiono la flecha de subir voy a ver las sentencias que ejecuto el programa antes puedo subir y bajar con las flechas y recorrer estas sentencias.
Por supuesto al softice no le alcanza la memoria para guardar todas las sentencias desde que se inicia el programa pues no tiene tanta memoria pero guarda muchísimas y si las que guardo no alcanzan se pone un BPX en la primera de todas y se vuelve a repetir el proceso para ir mas atrás en la lista de sentencias que ejecuto el programa.

Ahora aquí hay varias posibilidades es obvio que hay que anotar en un papel estas sentencias ya que no se pueden imprimir, hay crackers que anotan todas las sentencias (MUYYY LARGO YA QUE SON MUCHAS), hay crackers que anotan solo los saltos condicionales por donde paso el programa (BUENA OPCION Y MAS BREVE), hay quien anota los saltos y los CALLS y hay quien como yo anota una sentencia cada tanto, las que ve interesantes.

Estas son las anotaciones de puntos interesantes por donde pasa el programa antes de llegar al BPX ese, otra persona puede llegar a anotar muchas más o menos sentencias cada cual vera.

429C97
429D15
429D0C
422460
42C5C4

42BEB7
42BE80
42BE7A
42C553

42F2AC (AQUI ENCONTRE LO INTERESANTE)
42C3F1
42C7BC

Además se observa que el programa al ir hacia atrás las primeras sentencias son casi un ciclo o sea que se repiten varias veces la misma secuencia una vez que vamos mas atrás y salimos de ese ciclo repetitivo, encontramos lo interesante.

Ahora el tema es poner un BPX antes de que desaparezca la flechita del ADR de la Barra de tareas y probando en lo valores que anote veo que en

42f2AC si pongo un BPX para allí antes de que desaparezca la flecha y esta sentencia es un CALL

:0042F2AC FF1580C24300 call dword ptr [0043C280]

Cuando ejecuto ese CALL desaparece justo la flechita del ADR o sea que por aquí esta la cosa, ahora la cuestión es buscar un salto que evite llegara este CALL ya que una vez que el programa ingresa allí, KAPUTTT. Vemos que más arriba hay un REFERENCE que nos dice que puede venir el programa desde

414c33 o de 42f264

Ponemos un BPX en ambas direcciones de memoria para ver desde cual de las dos viene el programa, mandamos un email y para en 414c33, pum caímos en el softice.

Ya casi llegamos probamos invertir el salto que está un poquito más arriba en 414bf7 y no sirve, ya casi llegamos uff.

Hay más arriba un REFERENCED que viene de varios lugares

```
* Referenced by a CALL at Addresses:
|:00415B08   , :00415CE5   , :00415DC4   , :00415E21   , :00416158
|:004161CE
```

Ponemos un BPX en cada uno de ellos para ver de dónde viene el programa y mando otro email (POBRE AL QUE SE LO ESTE MANDANDO YA LO LLENE DE MAILS)

y para en **416158**

bueno vamos allí a ver que hay y vemos esta parte

```
:00416147     7515                      jne 0041615E  (SALTO A PARCHEAR)
:00416149     8B442404                  move exam, word per [esp+04]

* Possible Reference to Menu: MenuID_0001
                     |
* Possible Reference to Dialog: DialogID_0001
                     |
* Possible Reference to String Resource ID=00001: "Advanced Direct Retailer"
                     |
:0041614D     C705186A440001000000      move word per [00446A18], 00000001
:00416157     50                            push exam
:00416158 E893EAFFFF                    call 00414BF0
```

Habíamos caído en 416158 y justo arriba hay un salto, así que adivinen lo invierto y SORPRESA el programa saltea al call maldito manda el mail perfectamente y no sale cartelito alguno ni falla ni se cierra, jeje.

Espero que no tenga este programa ninguna otra sorpresita más así que por ahora programa crackeado, nos vemos en la lección 26.

Ricardo Narvaja

WINALUM 2.0.1

(LECCION VEINTISEIS)

Bueno después del ADR (ufff) vamos a un crack en DELPHI, el Winalum 2.0.1 es un programa para gente que hace trabajos en aluminio, ventanas, marcos, etc. El programa se baja de

http://www.coproinf.com/winalum.htm

Y viene en tres archivos (HAY QUE BAJAR LOS TRES) Winalum.exe, Winalum.r00 y Winalum.r01 y al ejecutar el exe se descomprimen. Primero de todo lo desensamblamos con el WDASM ya que no está comprimido, no hay problema, no tiene protección anti desensamblado ni anti softice, ni nada, que bueno.

Lo desensamblamos y nos damos cuenta que está hecho en DELPHI, ya que entre las STRING REFERENCES aparece una referencia a SOFTWARE\BORLAND\DATABASE ENGINE a veces puede decir SOFWARE\BORLAND\LOCALES o BORLAND solo, la mejor prueba que podemos hacer ya que esta descomprimido, es cargarlo con el DEDE, el cual ya va por la versión 2.50 actualmente, para no dar tantas vueltas.

Abrimos el DEDE y cargamos el ejecutable del WINALUM y lo carga por lo tanto esta en DELPHI, además buscamos aquí ya que la STRING REFERENCE que aparece cuando pones una clave falsa no aparece por ningún lado en el WDASM.

Una vez que desde el DEDE se arranca el programa, cerramos el mismo haciendo clic en la ventanita que dice que hagamos clic cuando el programa está completamente funcionando y vamos al DEDE, a PROCEDURES y allí buscamos algo que tenga que ver con REGISTRO y encontramos **UedtRegistrePrg** que parece tener que ver.

Marcamos y vamos a EVENTS y allí aparecen tres botones que seguro corresponden a los que tiene la ventana el primero que hay en la ventana es para ACEPTAR la clave (Dácord), el segundo es DEMOSTRACIÓN y el tercero es CANCELAR, todo eso lo podemos ver haciendo clic derecho encima de los botones y eligiendo SHOW ADITIONAL DATA lo que nos muestra en CAPTION el texto del botón ,por supuesto vamos al de ACEPTAR y hacemos clic derecho y elegimos DESENSAMBLAR (DISSASEMBLE) y allí nos aparece la parte del programa que se ejecuta cuando hacemos clic en ACEPTAR cuando ponemos una clave falsa.

Si ponemos un BPX en el SOFTICE donde comienza la rutina en 593700, (podemos tener que poner un BPX HMEMCPY primero y cuando entra en el SOFTICE volver al ejecutable y allí borrar el BPX HMEMCPY con BC* y poner el BPX 593700), y entonces ahora si vuelvo a la ventana de registro pongo un numero de varias cifras y hago clic en ACEPTAR y para en 593700

```
:00593758   A130246000      mov eax, dword ptr [00602430]
:0059375D   DC18            fcomp qword ptr [eax]
:0059375F   DFE0            fstsw ax
:00593761   9E              sahf
:00593762   741A            je 0059377E                 (SALTO a invertir)
:00593764   8D55F4          lea edx, dword ptr [ebp-0C]
:00593767   A14C1C6000      mov eax, dword ptr [00601C4C]
:0059376C   E84B35E7FF      call 00406CBC
:00593771   8B45F4          mov eax, dword ptr [ebp-0C]
:00593774   E89378EAFF      call 0043B00C               (cartel de chico malo)
```

Si vamos ejecutando con F10 vemos que llegamos al CALL que esta en 593774 y al pasar por encima de ese CALL sale el cartelito de que la clave no es correcta (chico malo), y justo antes esta el salto que puede pasar por encima ese CALL y si probamos invirtiéndolo vemos que se registra , el problema es que cuando arranca el programa de nuevo se vuelve a desregistrar, así que debe haber algún chequeo al comienzo del programa, por lo pronto ya tenemos el primer salto a modificar, para registrarse

:00593762 741A je 0059377E

cambiar a

:00593762 EB1A jmp 0059377E

con lo que acepta cualquier clave.

Ahora veamos la verificación al principio del programa, seguro que el CALL anterior al salto es el responsable del chequeo de la clave, así que veamos de donde es llamado es CALL para ver si desde alguna otra parte del programa es utilizado.

Si quieren divertirse vean desde que otros lugares entra a ese CALL

*** Referenced by a CALL at Addresses:**

|:0042648B , :004A5E2D , :004BE17A , :004E38A7 , :004E86B3
|:004E86E6 , :004E8D9C , :004E8DCF , :004F2098 , :004F20CB
|:004F2135 , :004F2168 , :004F2198 , :004F21CB , :004F21FE
|:004F23C7 , :004F2452 , :004F24DD , :004F2568 , :004F25F3
|:004F267E , :004F2900 , :00516404 , :0051649F , :005164C3
|:0052F513 , :0052F51F , :005334F9 , :005336A2 , :005336D9
|:00533710 , :00533744 , :00533778 , :00536DE8 , :00536E74
|:00536EEE , :00537007 , :00537059 , :005370B8 , :00537117
|:00537176 , :00537DE9 , :00537E0E , :00537E3B , :00537E60
|:0053804F , :0053806C , :0053B0CF , :0053B0F0 , :0053B111
|:0053B45A , :0053B47B , :0053B49C , :0053B4BD , :0053B4DE
|:0053CAC6 , :0053CBBF , :0053CC98 , :0053CE14 , :0053CE6C

|:0053CED7 , :0053CF42 , :0053CFAD , :0053D47D , :0053D4A2
|:0053D4C7 , :0053D4EC , :0053D511 , :0053D649 , :0053D66E
|:0053D693 , :0053D8D0 , :0053D8F5 , :0053D922 , :0053D947
|:0053D9D0 , :0053D9F5 , :0053DA22 , :0053DA47 , :0053E076
|:0053E0A2 , :0053E0CE , :0053E0FA , :0053E126 , :0053E2BE
|:0053E31D , :0053E340 , :0053E363 , :0053FEEC , :0053FF0F
|:00540057 , :005400FB , :005401AF , :0054020A , :0054024B
|:00540336 , :005403F6 , :0054129C , :005412BF , :00541407
|:005414AB , :0054155F , :005415BA , :005415FB , :005416E6
|:005417C2 , :005424B3 , :00542557 , :0054260B , :00542666
|:005426A7 , :0054302B , :005430CF , :00543183 , :005431DE
|:0054321F , :00543367 , :00543380 , :005433A3 , :005433BC
|:00543559 , :005435A9 , :00543621 , :00543647 , :005437A8
|:005437DA , :0054381A , :005438C4 , :005439A6 , :005439FF
|:00543A54 , :00543B25 , :00543B51 , :00543BA6 , :0054554E
|:00545580 , :005455D9 , :00545656 , :00545743 , :00546009
|:0054607A , :005460EB , :0054615C , :005461CD , :005462EB
|:00546349 , :0054637B , :0054660A , :0054666E , :00546718
|:005467FC , :0054682E , :0054689E , :005468D0 , :005471CB
|:005471FD , :00547256 , :005472D3 , :005473C0 , :00547BC1
|:00547C25 , :00547C96 , :00547D07 , :00547D78 , :00547E22
|:00547F06 , :00547F38 , :00547FA8 , :00547FDA , :00548325
|:00548389 , :00548433 , :00548517 , :00548549 , :005485B9
|:005485EB , :0054949C , :005494FC , :0054956F , :005495CA
|:0054960B , :0055543A , :0055549E , :0055550F , :00555580
|:005555F1 , :005556D4 , :00555718 , :00555783 , :005557EE
|:0055586E , :00555968 , :0055599A , :005559CC , :005559FE
|:00555A30 , :00555E23 , :00555E87 , :00555F6A , :00555FB0
|:0055601B , :00556086 , :00556106 , :005610E9 , :0056114C
|:005611F5 , :0056121D , :00561351 , :0056137C , :005613A7
|:00561438 , :00561476 , :005614B4 , :0056155C , :0056159F
|:0056177D , :005617B3 , :00561C46 , :00561D3F , :00561E18
|:00561F94 , :00561FEC , :00562057 , :005620C2 , :0056212D
|:005778BB , :00577913 , :0057904C , :005790A4 , :00579109
|:0057916E , :005791D3 , :00579373 , :0057939F , :005793CB
|:005793F7 , :00579423 , :005797F6 , :0057985A , :00579F22
|:00579F4F , :00579F7C , :00579FA9 , :00579FDC , :0057A34D
|:0057A3E1 , :0057A54F , :0057BDD7 , :0057BE04 , :0057BE31
|:0057BE64 , :0057BE97 , :0057C0DA , :0057C425 , :0057C56F
|:0057C60B , :0057C7AE , :0057EEDD , :0057EFD6 , :0057F0AF
|:0057F22B , :0057F27D , :0057F2E8 , :0057F353 , :0057F3BE
|:005807A7 , :00583A59 , :00583A7D , :00583AA1 , :00583B46
|:005847FD , :00584875 , :005848ED , :00584965 , :00584AD5
|:00584B23 , :00584B64 , :00585309 , :00585381 , :005853F9
|:00585471 , :005854F9 , :00585547 , :00585588 , :00585D1F
|:00585D7A , :00585DBB , :005873CC , :0058742C , :0058748C
|:00588D45 , :00588D69 , :00588D8D , :00588E32 , :00593753

|:005AAC1C , :005AAC7C , :005AACDC , :005B789D , :005B7915
|:005B798D , :005B7A05 , :005B7A8D , :005B7ADB , :005B7B1C
|:005B82E5 , :005B8334 , :005B8A89 , :005B8B01 , :005B8B79
|:005B8BF1 , :005B907D , :005B90CC , :005D3CCC , :005DF594
|:005DF904 , :005E1D4D , :005E1DDB , :005E1E69 , :005E1EF7
|:005E1F9A , :005E2022 , :005E20AA , :005E2132 , :005E277A
|:005E27E7 , :005E2A40 , :005E2AAD , :005E613C , :005E6150
|:005E65C1

Creo que si ponemos un BPX en cada uno nos morimos de aburrimiento por lo tanto hagamos otra cosa. Este es el CALL que estamos investigando entero desde que comienza hasta el RET

```
:0040B5F8       push ebx
:0040B5F9       add esp, FFFFFFEC
:0040B5FC       mov ebx, eax
:0040B5FE       mov eax, ebx
:0040B600       call 00404414
:0040B605       mov edx, esp
:0040B607       xor ecx, ecx
:0040B609       call 0040FC90
:0040B60E       test al, al
:0040B610       jne 0040B62B
:0040B612       mov dword ptr [esp+0C], ebx
:0040B616       mov [esp+10], 0B
:0040B61B       lea edx, dword ptr [esp+0C]
:0040B61F       mov eax, dword ptr [00601AB8]
:0040B624       xor ecx, ecx
:0040B626       call 00409CA8

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:0040B610(C)
|
:0040B62B         fld tbyte ptr [esp]
:0040B62E         add esp, 00000014
:0040B631         pop ebx
:0040B632         ret
```

Bueno lo transcribimos todo pero lo único que vamos a hacer es poner un BPX en el RET para ver cuando el programa llama a este CALL y sale de aquí a donde va. Pongo un BPX en 40b632 y arranco el programa de nuevo, la primera vez que para allí, salgo del RET con F10 y no veo ningún salto condicional que puede decidir desregistrar o no el programa, así que hago X y vuelve a parar en el RET, repetimos esto hasta que al salir del RET aparecemos en 5E6155

:005E6150 call 0040B5F8
:005E6155 mov eax, dword ptr [00602430] (al salir del RET caemos aqui)
:005E615A fcomp qword ptr [eax]
:005E615C fstsw ax
:005E615E sahf
:005E615F jne 005E618D (salto importante)

Que es una secuencia muy parecida a la que había en donde parcheamos el primer salto, las sentencias son casi idénticas, y tenemos un salto condicional en 5E615f llegamos hasta el haciendo F10 y vemos que aquí va a saltar (JUMP) o sea que tenemos que invertirlo para que no salte.

R eip= 5e6161

Si quitamos todos los BPX con BC* el programa arranca normalmente, y si no lo habías registrado pones cualquier clave la ACEPTAS te registra y al volver a arrancar no te desregistra mas ya que parcheamos la comprobación al principio del programa.

:005E615F jne 005E618D (salto importante)

cambiar a

:005E615F Nop
005E6160 Nop

O bien

:005E615F 752C por 9090 que es lo mismo

O sea parcheando los dos saltos el de 593762 y el de 5e615f

Ricardo Narvaja

UN ESPIA EN TU COMPUTADORA

(LECCION 27)

Se trata del programa STARR 007 versión 2.0 que se baja de

http://www.iopus.com/starr.htm

Es un verdadero espía que lo podes configurar para que vea todo lo que se tipea, claves, programas, usos, y lo mejor de que cuando está funcionando el monitor espía, si haces CTRL+ALT+DEL no aparece en la lista de procesos activos , ni allí ni con ningún programa que muestre la lista de procesos (PROCDUMP, PUPE, ETC), o sea que es casi imposible que la persona que usa la computadora sepa que el programa esta monitoreando todo lo que se hace, cuando uno accede a la computadora pide el REPORTE y te saca un reporte detallado de todo lo que se hizo en la misma.

Es ideal para computadoras que comparten varias personas y uno quiere saber que hacen los otros con la computadora. Bueno basta de promoción una vez instalado el programa vamos a la pestaña REGISTER y ponemos una clave falsa y al hacer clic en REGISTER sale allí mismo el cartel **WRONG REGISTRATION CODE ENTERED- PLEASE TRY AGAIN.**

Bueno abrimos el ejecutable con el WDASM y lo desensambla perfectamente, igual conviene trabajar con una copia del ejecutable que este en otro lugar, por si acaso. Allí en STRINGS REFERENCES aparece **WRONG REGISTRATION CODE ENTERED.** Si vamos subiendo vemos diversos carteles que aparecen y algunas REFERENCED… pero que son referidos a saltos apenas más arriba, seguimos más arriba y vemos:

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
:**004195D0**

y si vemos ese salto con GOTO CODE LOCATION **4195d0**

Nos damos cuenta que ese salto en 4195d0 es el que decide si te registra o no.

```
:004195D0 0F8EAB000000          jle 00419681
:004195D6 6A40                   push 00000040

* Possible StringData Ref from Data Obj ->"Registration"
                                  |
:004195D8 68680A4500             push 00450A68

* Possible StringData Ref from Data Obj ->"Your 007 STARR registration was "
                                        ->"successful - Thank you ! "
```

O sea que si no salta en 4195do lo que nos lleva a la zona de CHICO MALO, sigue aquí y nos lleva a **YOUR 007 STARR REGISTRATION WAS SUCESSFULL** o sea que estaríamos registrados, si invertimos este salto con el SOFTICE vemos que nos aparece el cartel de REGISTRADO y todo, pero cuando iniciamos el programa nos desregistra de nuevo. Analicemos el CALL antes del salto donde se realiza la comparación generalmente este está justo antes del salto

:004195B5 E8FC550000 call 0041EBB6

según el resultado de este CALL salta o no, el tema es que vemos que según el resultado que tiene EAX al salir de este CALL salta o no salta, además también vemos que este CALL es llamado desde varios lugares así que en vez de parchear todos los lugares desde donde es llamado el CALL podemos modificar el final de esa rutina para que siempre nos de al salir de ella el valor de EAX que hace que estés registrado, lo llame de donde lo llame.
Si entramos al CALL en el WDASM, posicionándonos encima y haciendo clic en el icono CALL entramos dentro del CALL, bajamos bastante hasta el RET donde termina ese CALL y este esta en

:0041EF28 C3 ret

:0041EF29 CC int 03
:0041EF2A CC int 03
:0041EF2B CC int 03
:0041EF2C CC int 03
:0041EF2D CC int 03
:0041EF2E CC int 03
:0041EF2F CC int 03

Además después del RET vemos algunas posiciones de memoria que no se usan donde podemos escribir las modificaciones que queramos.

Si probamos que dándole a EAX el valor de UNO al salir del CALL, el programa se registra, entonces lo modificamos y quedaría de esta forma

:0041EF28 B801000000 **mov eax, 00000001**
:0041EF2D C3 **ret**

Movemos a EAX el valor uno y después ejecutamos el RET, esto funcionaria supuestamente mejor que parchear el salto solamente pues aquí al modificar la rutina de comparación en cualquier punto del programa que se le ocurra chequear si es correcta la registración al llamar a este CALL le va a devolver EAX=1 o sea registrado SIEMPRE.

Realizamos esta modificación con el ULTRAEDIT, y arrancamos el programa nuevamente y ponemos una clave trucha y la acepta y si lo cerramos y volvemos a iniciar nos sigue diciendo que estamos REGISTRADOS, lo cual significa que lo vencimos.

Igual queda un detalle en otro archivo, en el archivo ESPIA llamado wsys.exe que esta en la misma carpeta, cuando arrancamos el MONITOR nos sale un cartelón diciendo que nos registremos, y recién después arranca el monitor espía, evidentemente esto fue puesto para que uno si no modifica esto, no le sirva para nada el programa, porque de que me sirve un espía que me avisa con un cartelón que está comenzando el trabajo de espía, hay que quitarlo.

Es fácil ya que el cartel es un dialogboxparama así que con el SOFTICE ponemos un BPX DIALOGBOXPARAMA y cuando aparece el cartel para allí, y cuando vuelve nos muestra donde está en el programa ese cartelón.

En el WDASM vemos esa parte

```
:00412C8A       Cmp dword ptr [0043228C], 01
:00412C94       jz 00412CA8
:00412C96       mov al, byte ptr [6A004128]
:00412C9B       add byte ptr [edx+74], ch
:00412C9E       mov edx, dword ptr [ebp+08]
:00412CA1       push edx

* Reference To: USER32.DialogBoxParamA, Ord:0093h
                 |
:00412CA2       Call dword ptr [004273A8]

* Referenced by a (U)nconditional or (C)onditional Jump at Addresses:
|:00412C88(C), :00412C94(U)
|
:00412CA8       push 00000000
```

Allí vemos la comparación y el salto justo antes a la llamada al cartelón que está en **412ca2** y es un dialogboxparama, el salto anterior en 412c94, saltea el cartelón y hace que no aparezca, podemos reemplazar el JZ por un JMP y listo saltaría por encima del cartel, pero por si acaso realice el mismo test más adelante vamos a MOVER a 43228c el valor de uno que es el que compara aquí y hace que salte.

Quedaría así

```
:00412C8A C7058C22430001000000   mov dword ptr [0043228C], 00000001
:00412C94 EB12                   jmp 00412CA8
:00412C96 A02841006A             mov al, byte ptr [6A004128]
:00412C9B 006A74                 add byte ptr [edx+74], ch
:00412C9E 8B5508                 mov edx, dword ptr [ebp+08]
:00412CA1 52                     push edx

* Reference To: USER32.DialogBoxParamA, Ord:0093h
                 |
:00412CA2 FF15A8734200       Call dword ptr [004273A8]
```

* Referenced by a (U)nconditional or (C)onditional Jump at Addresses:
|:00412C88(C), :00412C94(U)
|
:00412CA8 6A00 push 00000000

Hacemos con esto que además de que salte por encima del cartel y que no caiga en el DIALOGBOXPARAMA, nos guarde en 43228c el valor uno que es lo que tiene que tener para estar REGISTRADO.

CON ESTO PROGRAMA CRACKEADO.

Ricardo Narvaja

LABEL MATRIX 5.0

(LECCION VEINTIOCHO)

Siguiendo con el crackeo a pedido, este programa me lo pidieron y por eso lo crackee, sirve para hacer etiquetas con códigos de barras e imprimirlas, y varias cosas más, se baja de

http://www.strandware.com/prod/labelmatrix.html

Para el que sabe inglés esto dice la página que hace el programa:

Label Matrix provides advanced design, printing and database features that give you the flexibility to meet your most demanding labeling needs. With hundreds of thermal and thermal transfer custom printer drivers, you can maximize your printer's performance and print large runs of labels in a very short time. Whether you need address labels for your small business or thousands of inventory labels for a multi-site warehousing plant, Label Matrix can do the job.

Bueno instalamos el programa y lo iniciamos y vemos ventanas que nos recuerdan que es una versión demo , vemos en una ventana exactamente la STRING entre paréntesis **(Demo Versión)**, así que si abrimos el ejecutable con el WDASM y vemos que allí no esta, pero si ponemos algún Breakpoint en el SOFTICE como **BPX Drawtexta** o **BPX Getversion**, al volver de la función vemos que el ejecutable enseguida le pasa el control a otro archivo y es este el que sigue con la protección es un dll , se llama Lmw500v.dll y es un archivo de cuatro megas y pico donde prácticamente se encuentra todo el programa.

Abrimos con el Wdasm éste dll que se encuentra en el directorio donde se instalo el programa generalmente

C:\Program Files\lmwdemo

Se desensambla perfectamente ya que no está comprimido ni nada, y allí entre las STRINGS REFERENCES encontramos rápidamente (Demo Version) , al hacer doble click nos lleva a esta sección del código

:1023F66F E82C7EE1FF **call 100574A0**

:1023F674 2540010000 and eax, 00000140

:1023F679 85C0 test eax, eax

:1023F67B 7444 **je 1023F6C1**

- **Possible Reference to String Resource ID=07453: "(Demo Version)"**

Este salto nos haría registrarnos, si lo invertimos o por lo menos saltear la aparicion de la palabra DEMO VERSIÓN, pero miremos dentro del CALL anterior que es el que decide si salta o no, porque según el valor de EAX que sale de ese CALL decide saltar, y puede ser que ese CALL sea el verificador de si estas registrado o no, y también desde otros lugares salte allí en distintas partes del programa para verificar si estas registrado.

Entremos dentro del **CALL 100574a0**

Las references a este CALL son muchísimas no quiero llenar la hoja con ellas pero este CALL es llamado desde muchísimas partes del programa probablemente a verificar si estas registrado a cada rato, así que como no podemos parchear cientos de lugares desde donde es llamado el CALL, parchearemos el contenido para que siempre que sea llamado al final salga siendo **EAX=0** que es lo que al salir del CALL verifica , testea y hace invertir el salto, cualquier otro valor de EAX hace que vaya a DEMO VERSIÓN.

Veamos el CALL

```
:100574A0 55                push ebp

:100574A1 8BEC              mov ebp, esp

:100574A3 51                push ecx

:100574A4 894DFC            mov dword ptr [ebp-04], ecx

:100574A7 8B45FC            mov eax, dword ptr [ebp-04]

:100574AA 8B8020010000      mov eax, dword ptr [eax+00000120]

:100574B0 8BE5              mov esp, ebp

:100574B2 5D                pop ebp

:100574B3 C3                ret

:100574B8 CC                int 03

:100574B9 CC                int 03

:100574BA CC                int 03

:100574BB CC                int 03

:100574BC CC                int 03

:100574BD CC                int 03
```

En negrita esta marcado el RET donde termina el CALL hay que cambiar ese **ret** por **MOV eax,00** y después RET para que vuelva, o sea que independientemente de lo que haga antes, salga moviéndose a EAX el valor cero y recién después haga el RET para volver.

Quedaría así

:100574B2 5D pop ebp

:100574B3 B800000000 mov eax, 00000000

:100574B8 C3 ret

:100574B9 CC int 03

:100574BA CC int 03

:100574BB CC int 03

Cambiamos con el ULTRAEDIT estos valores buscamos la cadena, y la reemplazamos por estos valores de abajo y PROGRAMA REGISTRADO.

Aclaración: El programa tiene muchísimas limitaciones cuando está en DEMO, con este crack ya no vence y las limitaciones se van, igual yo no pude probar todas las funciones del programa, para saber si queda alguna , si a alguien le ocurre que le surge alguna limitación , que avise y lo seguimos en la LECCIÓN 29.

Igual todas esas llamadas al CALL para ver si estas registrado o no, es obvio que las hace cuando el programa tiene que ver si te limita o no, y al salir EAX=0 del CALL, siempre te habilita a usarlo, como si estuvieras registrado.

Ricardo Narvaja

PD: Luego de probar el programa los que lo utilizan se comprobó que tiene todavía algunas protecciones más que yo no me había dado cuenta, no referente al vencimiento, eso funciona perfecto, sino al funcionamiento, por lo tanto terminaremos esos detalles en la LECCIÓN 29.

COMO HACER BIEN EL CRACK DEL LABEL MATRIX 5.0

(LECCIÓN VEINTINUEVE)

Luego de hacer el crack según la LECCIÓN anterior, Mi amigo NIKI se dio cuenta de varios errores que generaba en el funcionamiento del programa, por ejemplo cuando ibas a LABEL PROPERTIES no aparecían los contenidos de las pestañas , y a veces se colgaba dando error, y se cerraba el programa.

Bueno lo que podemos rescatar de la LECCIÓN anterior es que la rutina de comprobación de si estas registrado es esa pero estaba mal parcheada y generaba errores. El error estaba en que yo considere esa CALL como que solamente servía para comprobar si estabas registrado o no, y si salía valiendo EAX=0 siempre estarías registrado.

Allí estaba el error ya que ese CALL además de para eso se usaba para muchas otras cosas, y si ponías EAX=0 siempre, corrompías esas otras partes del programa y daba error por todos lados. Ahora si funciona todo ya corregí el error del crack anterior y funciona 100 % Una vez que habíamos aplicado el primer crack había quedado modificado así

```
:100574A0 55                  push ebp
:100574A1 8BEC                mov ebp, esp
:100574A3 51                  push ecx
:100574A4 894DFC              mov dword ptr [ebp-04], ecx
:100574A7 8B45FC              mov eax, dword ptr [ebp-04]
:100574AA 8B8020010000        mov eax, dword ptr [eax+00000120]
:100574B0 8BE5                mov esp, ebp
:100574B2 5D                  pop ebp
:100574B3 B800000000          mov eax, 00000000
:100574B8 C3                  ret
```

Lo que está en negrita es lo que habíamos agregado y debe quedar para que funcione perfecto de esta forma

```
:100574A0 55                  push ebp
:100574A1 8BEC                mov ebp, esp
:100574A3 51                  push ecx
:100574A4 894DFC              mov dword ptr [ebp-04], ecx
:100574A7 8B45FC              mov eax, dword ptr [ebp-04]
:100574AA C6056043130200      mov byte ptr [02134360],00
:100574B0 8B8020010000        mov eax,[eax+120]
:100574B2 8BE5                mov esp,ebp
:100574B3 5D                  pop ebp
:100574B8 C3                  ret
```

Lo que está en azul es lo nuevo que agregamos. En realidad al funcionamiento solo le agregamos

mov byte ptr [02134360],00

Lo tenemos que intercalar justo allí, porque en la sentencia subsiguiente mueve el valor de EAX+120 a eax y por lo tanto, como cuando el programa verifica la registración EAX+120 es igual a 2134360, previmos que allí haya un cero para que EAX salga valiendo cero solo en ese caso, ahora cuando EAX+120 vale otra cosa, la rutina funciona como cuando no estaba parcheado, o sea que ponga un cero en 2134360 no afecta ya que después mueve el contenido de EAX+120 a eax y este es el valor que usa el programa normalmente, lo que no lo afecta para nada y funciona bien 100%.

Ricardo Narvaja

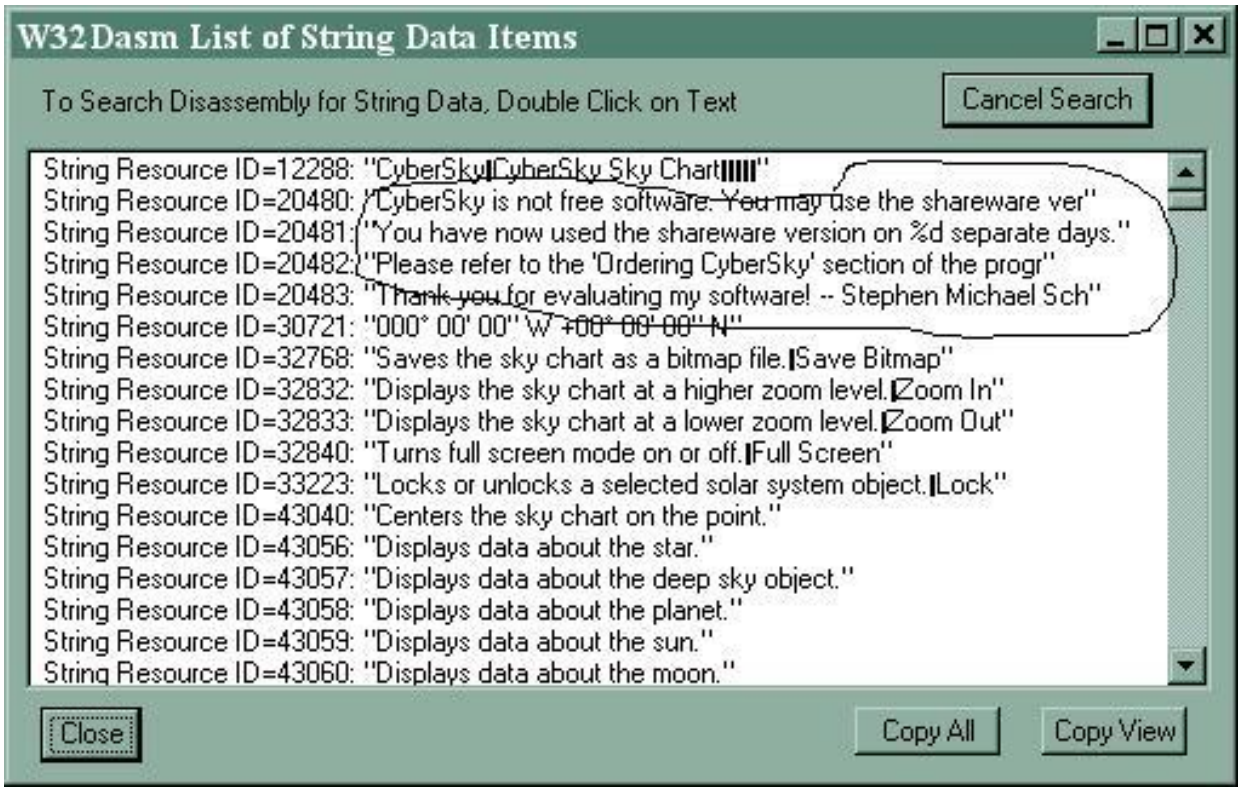
EL MOVIMIENTO DE LAS ESTRELLAS: CIBERSKY

(LECCIÓN TREINTA)

Esta lección se refiere al programa CIBERSKY 3.2.1 que se baja de

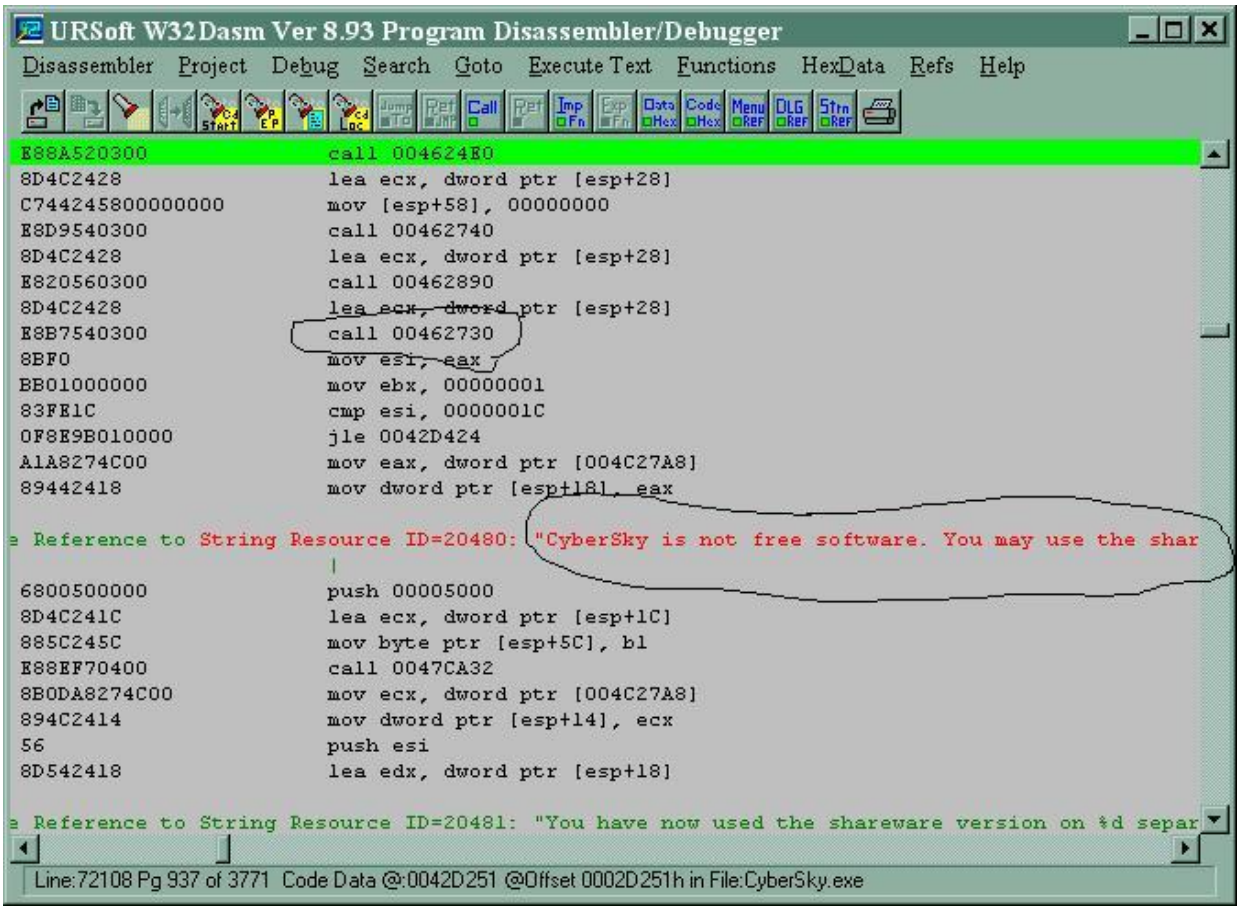
http://www.cybersky.com/

Que muestra las constelaciones y como se van moviendo las estrellas es muy lindo para la persona que está interesada en astronomía. Es bastante fácil de crackear así que será bastante breve la LECCIÓN ya que no está ni comprimido, ni tiene protección ANTISOFTICE, ni nada, lo único molesto que tiene es que sale un cartel diciéndonos que nos registremos a los 15 días de usarlo y a los 60 días ese cartel te tapa todo y no te deja ver nada.
Lo verdaderamente que cansa en este programa es que si quiero adelantar el reloj para ver ese cartel, no sale, o sea que el programa suma día por día que vas usando, y no le importa la fecha del reloj, solo es un día más, aunque diga AÑO 3005, así que vamos a desensamblar con el WDASM el ejecutable.



Aquí en el grafico vemos las STRINGS REFERENCES del cartel molesto, TANKS FOR EVALUATING MY SOFTWARE y antes CIBERSKY IS NOT FREE SOFWARE, etc etc etc.

Hacemos clic en la primera, y aparecemos en el WDASM en



Este es uno de los lugares donde cae, si hacemos clic de nuevo vamos a ver que aparecen estos mensajes en varios lugares distintos del programa, pero antes siempre está el CALL que está marcado en el dibujo CALL 462730, que es donde el programa cuenta los días, y sale EAX valiendo los días que usaste el programa. Miremos dentro de ese CALL poniéndonos encima y haciendo clic en el ICONO CALL.

```
:00462730 8B4124          mov eax, dword ptr [ecx+24]
:00462733 8B5114          mov edx, dword ptr [ecx+14]
:00462736 2BC2            sub eax, edx
:00462738 C3              ret
```

Si modificamos este CALL y hacemos que eax salga valiendo un valor fijo es como si claváramos los días que usamos en un valor fijo probemos esto.

```
:00462730 8B4124          mov eax, dword ptr [ecx+24]
:00462733 8B5114          mov edx, dword ptr [ecx+14]
:00462736 2BC2            sub eax, edx
:00462738 B800000000      mov eax, 00000000
:0046273D C3              ret
```

Con lo que siempre sale EAX valiendo cero y la comparación que hay al salir este call hace que el salto condicional siguiente saltee el cartel maldito. Como este CALL es siempre el que decide la aparición del CARTEL MALDITO en todo el programa, por lo tanto ya no aparece nunca más en ningún caso.

PROGRAMA CRACKEADO

Ricardo Narvaja