¡¡¡ IMPORTANTE !!!

Si no tienes claro el crack en DOS incluidas las instrucciones en ensamblador vas a perder el tiempo leyendo esta página, te aconsejo que comiences la casa por los cimientos. No se puede construir un tejado donde no hay algo que lo sujete. Una vez que se tienen los conocimientos necesarios para crackear en DOS, resulta muy fácil hacerlo en windows (aunque parezca una paradoja), ya que disponemos de herramientas más potentes que nos facilitan y reducen mogollón el trabajo. En cambio, es imposible crackear en windows sin conocimientos básicos de ASM. No creas que esto del crackeo es cosa de unos minutos ni que siempre se localiza lo que andamos buscando. Ten siempre una copia de seguridad de tus datos más importantes, los cuelgues son habituales y después de unos cuantos siempre se pierde algo importante por el camino.

Vamos a lo que nos interesa...

UNA NOTA CURIOSA

Sabéis que para crackear en DOS es imprescindible estar familiarizado con las interrupciones (o tener un buen manual de consulta), bien pues olvidaros de ellas en windows. El crackeo en este entorno requiere del conocimiento de las API's (o un buen manual de consulta). Sólo cuando se rueda un programa de dos en windows es útil el tema de las interrupciones.

Cuando desensambléis el código de lo que sea en windows observaréis que siendo un entorno de 32 bits la memoria continua estando segmentada. No se a que es debido ésto, pero supongo que será en previsión a la demanda de memoria que tendrán los programas dentro de poco tiempo. De todos modos podréis comprobar que los segmentos son increiblemente grandes, Windows asigna a cada aplicación en ejecución la friolera de 4 gigabytes. Aunque no se disponga de tanta memoria, el entorno hace uso de la memoria virtual para ello, de tal modo que si rodamos 10 aplicaciones es como si tuvieramos 40 Gigas de ram.

También podréis observar que algunos registros llevan una E delante, este es el caso de

AX-BX-CX-DX-SI-DI-BP-SP-IP

que se convierten en

EAX-EBX-ECX-EDX-ESI-EDI-EBP-ESP-EIP

Los registros son los mismos solo que en lugar de ser de 16 bites son de 32 y, por supuesto, podemos utilizarlos como más nos convenga.

SOFTICE... (nuestra arma más poderosa. Alguien lo denominó 'La Bestia')

Con este programa se hacen maravillas que hace unos años eran inimaginables cuando se crackeaba en DOS. Antes, para crackear una protección no demasiado complicada, era necesario tracear el código durante días entreros, saltando mil y una instrucción anti-traceo (int 3 a punta pala, modificacion de las direcciones de retorno, etc). Ahora y con la ayuda de este poderoso programa se puede interrumpir la ejecución de cualquier cosa en el momento que nos interese. Para ello es necesario saber en qué lugar o momento hay que poner un BP (break point o punto de ruptura).

Si has llegado hasta aquí, pensaré que tienes los conocimientos necesarios para poder saltarme la explicación de algunas cosas (de lo contrario necesitaría muchisimo espacio web y mogollón de curro para poder explicarlo todo). Por este motivo centraré esta página en el uso de softice, y que mejor forma de empezar que con una correcta...

CONFIGURACION DE SOFTICE

El primer paso, después de la instalación es editar el archivo "**winice.dat**" y quitarle el punto y coma al menos a las siguientes lineas

EXP=c:\windows\system\kernel32.dll
EXP=c:\windows\system\user32.dll
EXP=c:\windows\system\gdi32.dll

Esto nos permitirá interceptar las llamadas de las funciones mas importantes.

En la linea **PHYSMB=XX** cambiar XX por el número de megabytes de memoria ram que tengais disponible.
En la línea **DRAWSIZE=XXXX** cambiar XXXX por el número de Kb. de memoria de video que tengais.
En esta otra **INIT="X;"** podréis personalizar el arranque del softice intercalando entre **INIT="** y **X;"** lo siguiente
LINES ?;
donde "?" es el número de lineas que queremos que tenga la pantalla (60 es una buena cantidad)
CODE ON;
esto permite que a la izquierda del código podamos ver, en modo hexadecimal, los bytes que componen cada instrucción
WC ?
"?" = número de lineas que queremos que muestre el código.
WR
nos mostrara el el estado de los registros.
La linea deberia de quedar así (más o menos):
INIT="code on;lines 60;wr;wl;wc 30;X;"

Winice.dat debería quedar más o menos así

```
NMI=ON
SIWVIDRANGE=ON
LOWERCASE=OFF
MOUSE=ON
NOLEDS=OFF
NOPAGE=OFF
PENTIUM=ON
THREADP=ON
VERBOSE=ON

PHYSMB=128
SYM=1024
HST=256
DRAWSIZE=8192
TRA=8

INIT="CODE ON;LINES 60;WC 30;WD 20; WW 10;WATCH EAX;WATCH
*EAX;WATCH EBX;WATCH *EBX;"
INIT="WATCH EDX;WATCH *EDX; X;"

F1="h;"
```

```
F2="^wr;"
F3="^src;"
F4="^rs;"
F5="^x;"
F6="^ec;"
F7="^here;"
F8="^t;"
F9="^bpx;"
F10="^p;"
F11="^G @SS:ESP;"
F12="^p ret;"
SF3="^format;"
CF8="^XT;"
CF9="TRACE OFF;"
CF10="^XP;"
CF11="SHOW B;"
CF12="TRACE B;"
AF1="^wr;"
AF2="^wd;"
AF3="^wc;"
AF4="^ww;"
AF5="CLS;"
AF8="^XT R;"
AF11="^dd dataaddr->0;"
AF12="^dd dataaddr->4;"
CF1="altscr off; lines 60; wc 32; wd 8;"
CF2="^wr;^wd;^wc;"

EXP=c:\windows\system\kernel32.dll
EXP=c:\windows\system\user32.dll
EXP=c:\windows\system\gdi32.dll
EXP=c:\windows\system\comdlg32.dll
EXP=c:\windows\system\shell32.dll
EXP=c:\windows\system\advapi32.dll
EXP=c:\windows\system\msvbm50.dll

WDMEXPORTS=OFF
MONITOR=0
; WINICE.DAT
; (SIW95\WINICE.DAT)
; for use with SoftICE Version 3.2 (Windows 95)
; 14 July 1997
;
*****************************************************************************
; If your have MORE than 32MB of physical memory installed, change
; the PHYSMB line to the correct # of Megabytes.
; If you have LESS than 32MB you can save a bit of memory by
; specifying the correct # of Megabytes
; Example: PHYSMB=32
;
*****************************************************************************
; ***** Examples of sym files that can be included if you have the SDK *****
;    Change the path to the appropriate drive and directory
;LOAD=c:\windows\system\user.exe
;LOAD=c:\windows\system\gdi.exe
;LOAD=c:\windows\system\krnl386.exe
;LOAD=c:\windows\system\mmsystem.dll
;LOAD=c:\windows\system\win386.exe
; ***** Examples of export symbols that can be included *****
;    Change the path to the appropriate drive and directory
```

```
;EXP=c:\windows\system\vga.drv
;EXP=c:\windows\system\vga.3gr
;EXP=c:\windows\system\sound.drv
;EXP=c:\windows\system\mouse.drv
;EXP=c:\windows\system\netware.drv
;EXP=c:\windows\system\system.drv
;EXP=c:\windows\system\keyboard.drv
;EXP=c:\windows\system\toolhelp.dll
;EXP=c:\windows\system\shell.dll
;EXP=c:\windows\system\commdlg.dll
;EXP=c:\windows\system\olesvr.dll
;EXP=c:\windows\system\olecli.dll
;EXP=c:\windows\system\mmsystem.dll
;EXP=c:\windows\system\winoldap.mod
;EXP=c:\windows\progman.exe
;EXP=c:\windows\drwatson.exe
; ***** Examples of export symbols that can be included for Windows 95 *****
;    Change the path to the appropriate drive and directory
;EXP=c:\windows\system\shell232.dll
;EXP=c:\windows\system\comctl32.dll
;EXP=c:\windows\system\crtdll.dll
;EXP=c:\windows\system\version.dll
;EXP=c:\windows\system\netlib32.dll
;EXP=c:\windows\system\msshrui.dll
;EXP=c:\windows\system\msnet32.dll
;EXP=c:\windows\system\mspwl32.dll
;EXP=c:\windows\system\mpr.dll
```

Con esta pequeña modificación ya estamos preparados para utilizarlo. Resetea el ordenador para que la nueva configuración tenga efecto.

Si ya lo tenemos instalado, el siguiente paso es probarlo, así que Ctrl+D para entrar y F5 para salir... que... ¿funciona?... bien ahora vamos a aprender a utilizarlo

Observar los registros

Entrar en softcite Ctrl+D y escribir "D EAX". En la ventana de datos aparecerá el contenido en ascii del registro EAX. Hacer la misma prueba con el resto de registros y observareis que la ventana cambia indicandonos lo que hay en cada uno de ellos pero ojo, no nos enseña lo que hay en EAX sino lo que hay en el lugar al que apunta EAX, si EAX=04043567 estaremos viendo los datos que hay en la posición de memoria 04043567.
Si hacemos "D *EAX" softcite cojerá el valor de los primeros 4 bytes a los que apunta EAX y nos enseñará lo que hay en esa dirección. Se que es un poco lioso pero hacer pruebas hasta que comprendáis lo que está sucediendo. Veamos un ejemplo práctico:
Supongamos que en la ventana de registros observamos que EAX = 0050F608
entonce, si hacemos "D EAX" veremos que la ventana de datos apunta a la dirección 0050F608 y veremos los bytes que contiene. Estos pueden ser por ejemplo 11-01-00-00-FF-FF-FF-FF etc...
Ahora vamos a hacer "D *EAX" ¿ que hace softice ? pues carga los primeros cuatro bytes (11-01-00-00), los coloca en orden inverso (00-00-01-11) y nos muestra el contenido de esta dirección (00000111). ¿ lo habeis entendido ?. Este dato es muy importante ya que las funciones son llamadas pasandoles como parámetros una dirección concreta y otras veces son llamadas diciendoles en qué dirección se encuentra la dirección donde están los datos. Sería como decir "la dirección es 00000111" y "la dirección se encuentra donde apunta EAX"
Se que es un poco complicado pero practicar con esto hasta que entendáis este detalle y no paséis a otro punto hasta tenerlo bien claro.

Break Point

El break point (bp) hay que entenderlo como un lugar (el que a nosotros nos interese) donde haremos que el programa se detenga y softice nos de el control enseñandonos el código desensamblado o descompilado. Para hacer una prueba sobre esto hacer lo siguiente:

1.- Dentro de windows pulsar Ctrl + D para entrar en softice
2.- Escribir "BPX GETWINDOWTEXTA"
acabamos de decirle que cuando se ejecute la función de capturar el texto de una ventana interrumpa ese programa y nos de el control.
3.- Salir de softice con F5
4.- Ejecutar cualquier programa que tenga una ventana donde se escriba alguna cosa (digamos, por ejemplo, la introducción de un número de serie)
5.- Escribir lo que sea y pulsar aceptar.
6.- Si en este momento estáis dentro de softice es que vuestro bp ha funcionado, si no es así es que el programa que habeis arrancado no utiliza "getwindowtexta" para la captura de texto, así que probar con "getwindowtext", "getgdlgitemtexta" o "getdlgitemtext".
7.- Estamos antes de que el programa capture el texto que hemos introducido, para que termine la función y saber quien la ha llamado pulsar F12 y F12 otra vez.
8.- Ahora estamos después del código que ha capturado el texto introducido... y a partir de este momento hay que trabajarselo.
9.- A partir de aquí, el programa hará una serie de comprobaciones de la cadena introducida. Esta comprobación varía dependiendo del programador que ha hecho la aplicación. Es posible que realice varias operaciones con el valor de cada byte que compone la cadena (sumas, restas, multiplicaciones, operaciones XOR, Etc.) y lo compare con el número que se supone es el adecuado (de aquí que varios números de serie sean validos para registrar un mismo programa). También es posible (aunque rara vez sucede) que se compruebe la cadena con otra cadena (la buena) en cuyo caso veremos la buena y la introducida por nosotros en la memoria, no estarán muy alejadas la una de la otra. En cualquier caso, despues de la comparación siempre hay una instrucción que nos envía al código que continua la ejecución normal del programa o nos tira a la calle en caso de que la comparación no sea la adecuada. Estas instrucciones son mas o menos así:

```
cmp dword ptr tal_y_cual, pascual      compara un número con otro
je 00123456                            salta al código bueno si es igual
jmp 00345678                           salta al código de tirarnos a la calle
```

Es lógico pensar que si variamos la instrucción **je 00123456** por **jne 00123456** ó **jmp 00123456** el código se ejecutará como si el programa fuera legal o registrado y no importa el tipo de protección que tenga, nos da lo mismo que sea un disco llave, una mochila o un número de serie. Da lo mismo el tipo de protección que utilice el programador y lo sofisticada que ésta sea, a nosotros nos importa un pepino, tan solo tenemos que dar con la instrucción que la comprueba y parchearla.

Podemos poner un BP en cualquier función de las API's de Windows. Las que capturan el texto introducido en una ventana son:

BPX GETWINDOWTEXT
BPX GETWINDOWTEXTA (para la versión de 32 bites)
BPX GETDLGITEMTEXT
BPX GETDLGITEMTEXTA (para la versión de 32 bites)

Hay que tener en cuenta que cuando softice detiene la ejecución del programa por el BPX nos encontraremos ANTES DE QUE ESTA FUNCION SE EJECUTE, así que tendremos que pulsar F12 para ejecutarla y ver quien la ha llamado ya que la comparación se efectuará después. Este fundamento es la base para destripar cualquier protección del tipo nº de serie. Os aseguro

que una vez comprendido el mecanismo es muy fácil saltarse la mayoría de estas protecciones, tan sólo tendremos que dar con el fragmento de código que realiza esta tarea..

Puntos de ruptura importantes

Además de los puntos de ruptura ya comentados y dependiendo del comportamiento de la aplicación, podemos probar los siguientes:

Supongamos que despues de introducir un número de serie no válido nos sale una ventan advirtiendonos de ello, bien pues deberemos detener el programa en algún sitio, así que porbaremos con alguna de estas funciones:

MESSAGEBEEP
SENDMESSAGE
MESSAGEBOX
SHOWWINDOW
GETPRIVATEPROFILESTRING
GETPRIVATEPROFILEINT
HMEMCPY

Es importante observar el comportamiento de cada programa para meterle mano por un sitio u otro. Colocando un BPX en la función que utilice el programador para mostrarnos el mensaje de error (tendremos que probarlas todas) estaremos en la parte del código que ya HA HECHO la comprobación de la protección, con lo que estaremos muy cerca del código que deberemos parchear, sólo tendremos que tener en cuenta que éste código está detras de lo que estamos traceando. Para no perdernos sobre un millón de paradas que nos hará Softice sobre las funciones interceptadas, hay que colocar los BPX SOLAMENTE antes de que el código llame a la función en el momento oportuno, o sea, que no hay que colocar los BPX antes de rodar la aplicación, hay que ponerlos en el preciso instante en el que sabemos que va a ser llamada para comprobarla. Supongamos que rodamos la aplicación y estamos sobre la ventana de introducción del numerito, pues introducimos el numerito y antes de pulsar sobre el botón de aceptar es cuando colocamos el BPX, de esta forma sabemos que softice dentendrá la aplicación en el momento que a nosotros nos interesa.

Comandos del SoftIce

BREAK POINTS (puntos de ruptura)

COMANDO	EFECTO QUE PROVOCA	EJEMPLO
BPM, BPMB, BPMW, BPMD	Breakpoint de acceso a memoria	BPM 127:00040100
BPR	Breakpoint de rango de memoria	
BPIO	Breakpoint de acceso a puerto I/O	BPIO RW 278
BPINT	Breakpoint de interrupción	BPINT 16
BPX	Breakpoint de ejecución	BPX MESSAGEBEEP

BMSG	Breakpoint de mensaje de Windows	
BSTAT	Breakpoint estadísticas	
CSIP	Selecciona CS:EIP rango calificado	

MANIPULACIÓN BREAK POINTS

COMANDO	EFECTO QUE PROVOCA	EJEMPLO
BPE	Edita breakpoint	
BPT	Usa breakpoint como plantilla	
BL	Lista los breakpoints actuales	
BC	Borrar breakpoint	
BD	Desactiva breakpoint	
BE	Activa breakpoint	
BH	Breakpoint historial	

MOSTRAR/CAMBIAR MEMORIA

COMANDO	EFECTO QUE PROVOCA	EJEMPLO
R	Muestra/cambia el contenido de los registros	
U	Desemsambla instrucciones	
D, DB, DW, DD, DS, DL, DT	Muestra memoria	
E, EB, EW, ED, ES, EL, ET	Edita memoria	
PEEK	Lee de dirección física	
POKE	Escribe a dirección física	
PAGEIN	Carga un página en la memoria física (nota: no siempre)	
H	Ayuda sobre una función especifica	
?	Evalua una expresión	
VER	Versión de SoftICE	
WATCH	Añade reloj	
FORMAT	Cambia el formato de datos de la ventana	
DATA	Cambia datos de la	

	ventana	

MOSTRAR INFORMACIÓN DEL SISTEMA

COMANDO	EFECTO QUE PROVOCA	EJEMPLO
GDT	Muestra tabla de descripción general	
LDT	Muestra tabla de descripción local	
IDT	Muestra tabla de descripción de interrupciones	
TSS	Muestra el estado de los segmento de la tarea	
CPU	Muestra información de los registros de la CPU	
PCI	Muestra información de los dispositivos PCI	
MOD	Muestra la ventana de lista de modulos	
HEAP	Muestra la ventana de memoria global	
LHEAP	Muestra la ventana de la memoria local	
VXD	Muestra ventana del mapa VxD	
TASK	Muestra la ventana de lista de tareas	
VCALL	Muestra las llamadas VxD	
WMSG	Muestra la ventana de mensajes	
PAGE	Muestra información de tabla de pagina	
PHYS	Muestra todas las direciones virtuales de direcciones físicas	
STACK	Muestra llamadas a la pila	
XFRAME	Muestra los marcos de la excepción activos	
MAPV86	Muestra mapa de memoria v86	
HWND	Muestra ventana de información handle	
CLASS	Muestra ventana de información de clases	
VM	Muestra información de la máquina virtual	
THREAD	Muestra información	

	thread	
ADDR	Muestra/cambia el contexto de dirección	
MAP32	Muestra mapa de sección de 32 bits	
PROC	Muestra información del proceso	
QUERY	Muestra los procesos del mapa de direciones virtuales	
WHAT	Identifica el tipo de una expresión	
OBJDIR	Muestra información sobre el directorio objeto	
DEVICE	Muestra información sobre un dispositivo	
DRIVER	Muestra información sobre un driver	
FOBJ	Muestra información sobre un archivo objecto	
IRP	Muestra información sobre un IRP	
COMANDOS DE PUERTOS I/O		
I, IB, IW, ID	Introducir datos de un puerto I/O - Ix (x es el tamaño del dato): - B => Byte, W => Word, D => Doble Word	
O, OB, OW, OD	Manda datos a un puerto I/O	
COMANDOS DE CONTROL DE FLUJO		
X	Regresa al administrador debugger o programa	
G	Ir a dirección	
T	Solo pasa una instrucción	
P	Pasa saltando llamadas, interrupciones, etc.	
HERE	Ir a la actual posición del cursor	
EXIT	Forzar y salir al actual DOS/Windows programa	
GENINT	Generar una interrupción	
HBOOT	Resetea el sistema (reseteo total)	

CONTROLES DE MODO

I1HERE	Dirije INT1 al SoftICE	
I3HERE	Dirije INT3 al SoftICE	
ZAP	Desintegre incluido en INT1 o INT3	
FAULTS	Activa/Desactiva la falta entrampando SoftIce	
SET	Cambia una variable interna	

OPTIMIZAR COMANDOS

PAUSE - Muestra los controles en modo scroll
ALTKEY - Selecciona la secuencia de teclas para invocar la ventana
FKEY - Muestra/selecciona la función de las teclas
DEX - Muestra/asigna las expresiones de datos de la ventana
CODE - Muestra los bytes de instrucciones en la ventana de codigo
COLOR - Muestra/selecciona los colores de la pantalla
ANSWER - Auto-responde y remite la consola al modem
DIAL - Redireciona la consola al modem
SERIAL - Redirecciona la consola
TABS - Selecciona/Muestra las opciones de la etiqueta
LINES - Selecciona/Muestra el numero de lineas en pantalla
WIDTH - Selecciona/Muestra el numero de columnas en pantalla
PRN - selecciona el puerto de la impresora
Tecla PRINT-SCREEN - Patalla del dump a la impresora
MACRO - Define un nombre para un comando de macro

COMANDOS DE UTILIDAD

A - Ensambla el codigo
S - Busca datos
F - Rellena la memoria con datos
M - Mueve datos
C - Compara dos bloques de datos

TECLAS DE USO EN LA LINEA DEL EDITOR

(Up) - Llama a la anterior linea de comandos
(Down) - Llama a la proxima linea de comandos
(Right)- Mueve el cursor a la derecha

(Left) - Mueve el cursor a la izquierda
BKSP - Vuelve encima del último caracter
HOME - Comienzo de linea
END - Final de linea
INS - Modo de inserción
DEL - Elimina caracter
ESC - Cancela el comando actual

USO DE LAS TECLAS DE DESPLAZAMIENTO

PageUp - Muestra la anterior pagina del historial
PageDn - Muestra la proxima pagina del historial
Alt-(Up) - Desplaza la ventana de datos una linea hacia abajo
Alt-(Down) - Desplaza la ventana de datos una linea hacia arriba
Alt-PageUp - Desplaza la ventana de datos una página hacia abajo
Alt-PageDn - Desplaza la ventana de datos una página hacia arriba
Ctrl-PageUp - Desplaza la ventana de codigo una página hacia abajo
Ctrl-PageDn - Desplaza la ventana de codigo una página hacia arriba
Ctrl-(Up) - Desplaza la ventana de codigo una linea hacia abajo
Ctrl-(Down) - Desplaza la ventana de codigo una linea hacia arriba

VENTANA DE COMANDOS

WC - Toggle ventana de codigo
WD - Toggle ventana de datos
WF - Toggle punto flotante de la ventana de la pila
WL - Toggle ventana locals
WR - Toggle ventana de registros
WW - Toggle ventana de reloj
EC - Activa/Desactiva ventana de codigo
. - Localiza la intrucción actual

CONTROL DE LA VENTANA

CLS - Limpia ventana
RS - Restaura la pantalla de programa
ALTSCR - Cambia a la alternativa de despliegue
FLASH - Restaura la pantalla durante P y T

COMANDOS DE SIMBOLOS/ORIGEN

SYM - Muestra simbolos
SYMLOC - Relocaliza la base de simbolos
EXP - Muestra los simbolos explortados
SRC - Toggle entre el origen, mixto & codido
TABLE - Selecciona/elimina la tabla de simbolos
FILE - Cambia/muestra la actual fila de origen
SS - Buscar una cadena en el modulo de origen
TYPES - Lista todos los tipos, o muestra la definición de los tipos
LOCALS - Muestra los locals actualmente al alcance

COMANDOS DE TRACEO

SHOW - Muestra el buffer de traceo
TRACE - Introduce backtrace en modo simulación
XT - Paso en trace en modo simulación
XP - Programa paso en trace en modo simulación
XG - Go to address in trace en modo simulación
XRSET - Resetea backtrace buffer del historial

OPERACIONES ESPECIALES

. - Precediendo un número decimal se especifica un número de linea
$ - Precediendo una dirección se especifica la dirección de segmento
# - Precediendo una dirección se especifica la dirección del selector
@ - Precediendo una dirección se especifica indirección

Continuara...