Software:	**Canasta 5.0**
Objetivo:	**Keygen**
Herramientas:	**OllySND, RadASM**
Cracker:	**lv!n$on**
Fecha:	**8/03/2012**
Tutorial Nº	**8**

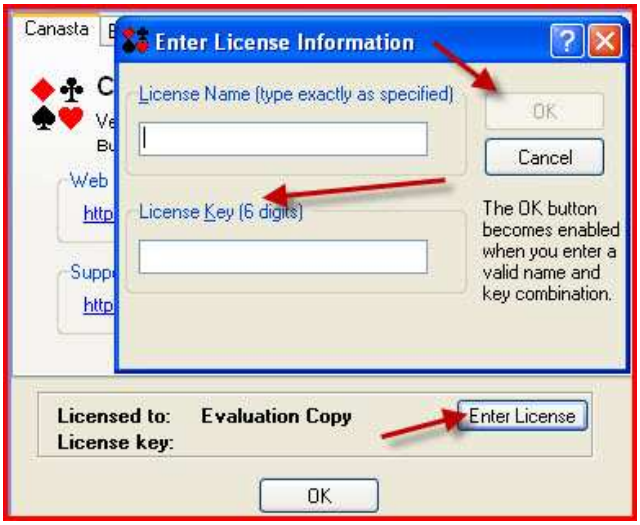
Downloads:

Canasta 5.0: **http://www.mediafire.com/?ddrlssmvzwokwjb**
Keygen Source Code: **http://www.mediafire.com/?dlbrnpm4v5aw4br**

En la introducción al Cracking hay un tuto acerca de Canasta 5.0 un juego de cartas donde se pesca el serial, pero aquí, trataremos de hacerle un Keygen. Así, vamos aprendiendo mutuamente acerca del keygening que requiere más dedicación porque hay algoritmos no aptos para newbies. ☺

No creo que éste sea el caso porque es algo sencillo. Bueno, eso creo. Veremos poco a poco como se analiza la rutina del serial y como extrapolarlo al generador de clave que trataremos de programar.

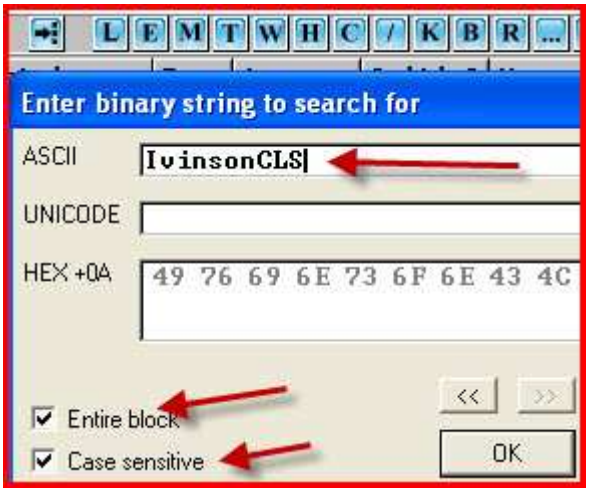
Comencemos a ver cómo nos va.



Tenemos un botón inactivo y la clave debe tener 6 dígitos.

Tipeo solo el nombre:

Luego, Voy a "M" en Olly y buscaré esa string en Memoria.



Para en:

Seleccionamos el nombre y le ponemos un Memory BreakPoint:

Luego, tipeemos cualquier cosa en el edit del serial y para en:

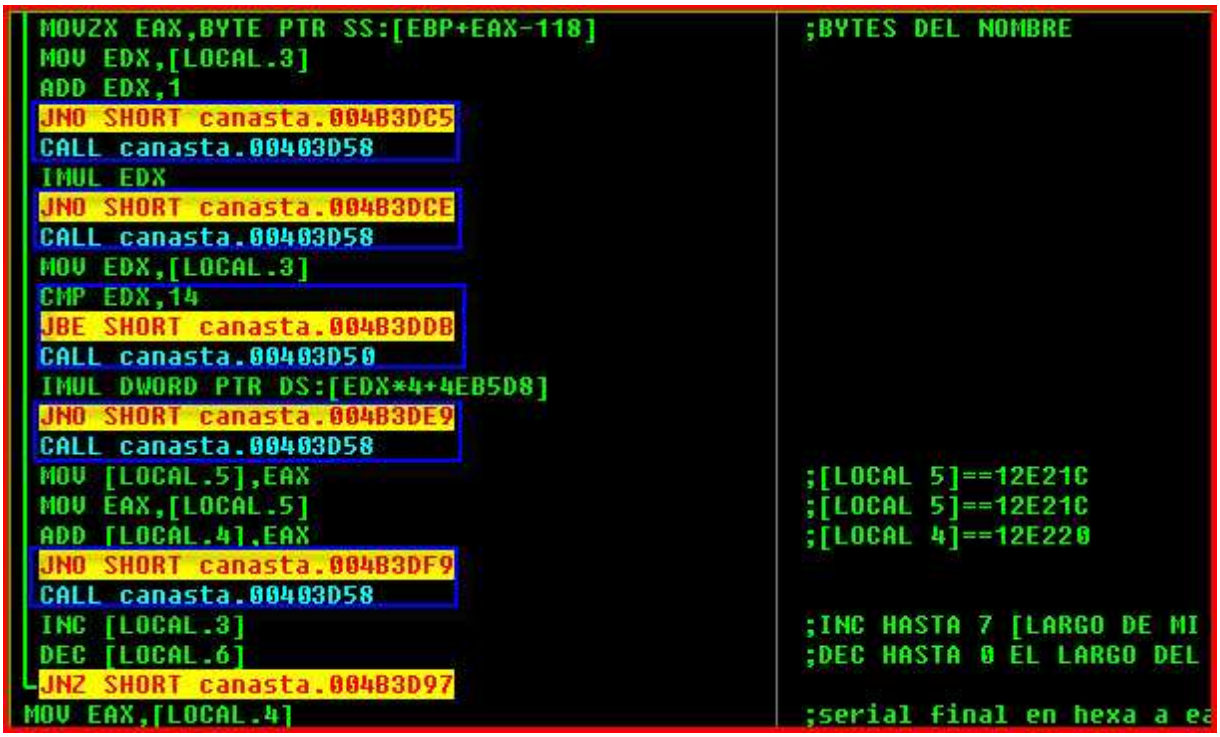
Vamos dando **F9** hasta ver el primer nombre de la lista negra: **TNO**.

Debajo de todos los nombres, está la **CALL** que se encarga de la rutina del serial.

Le damos **ENTER** para ver que hay por dentro. Y caemos aquí:

Vamos hasta la rutina para comenzar a trabajar de una vez.

Bajemos hasta ver la famosa rutina:



Esa es toda la rutina. Lo que está en rectángulos azules no lo tomaremos en cuenta para nuestro Keygen. Con la magia de la edición, veamos esa misma rutina pero sin los rectángulos azules para hacernos una idea más clara de cómo sería la rutina final.

Mueve byte a byte los cars del nombre.
Mueve contador a EAX.
Suma 1 al contador.
Mult cada carácter * su pocisión.
Mueve contador a EDX.
Mult cada car * valor constante.
Guarda result en buffer1
Mueve contador de buffer1 a EAX.
Suma EAX con Constante.
Incrementa contador1.

Decrementa contador2
Sigue
Mueve serial en hexa a EAX.

¡Qué cambio! Así, se ve mejor.

Si le ponen un **BP** al principio podrán ver como funciona. Quiten el **BPM** y den Run.

Escriban por ejemplo: Ivinson en el bloc de notas, cópienlo y péguenlo en el form de registro de Canasta. De esa manera, parará de una vez con el nombre completo. Porque si escriben uno por uno, se pausará y será más incómodo estar dando **F9** por cada carácter.

Pueden ver algunos comentarios que hice en la imagen superior.

Pero de todas maneras aquí les explicaré bien como es su funcionamiento.

Esta función usa una tabla de valores constantes para cada posición de los caracteres.

Obsérvenla:

CONSTANTES

```
004EB5D8  D9 00 00 00   63 00 00 00
004EB5E0  58 00 00 00   22 00 00 00
004EB5E8  3E 00 00 00   93 00 00 00
004EB5F0  F0 00 00 00   08 00 00 00
004EB5F8  34 00 00 00   62 00 00 00
004EB600  1B 00 00 00   BF 00 00 00
004EB608  D7 00 00 00   B9 00 00 00
004EB610  6F 00 00 00   4A 00 00 00
004EB618  5A 00 00 00   B2 00 00 00
004EB620  84 00 00 00   24 00 00 00
004EB628  11
```

Para encontrar esa tabla pueden dar Follow in Dump cuando tracen hasta el segundo **IMUL** de la rutina.

La instrucción es ésta:

IMUL DWORD [**EDX***4+**4EB5D8**]

Como **IMUL** multiplica **EDX** por **EAX** y guarda en **EAX** y en el caso que el resultado sea mayor a un **DWORD**, guarda el resto en **EDX**. Arriba vemos que multiplica **EDX** que será la posición del carácter por 4 más la posición **4EB5D8**.

Veamos, por ejemplo, como funciona la instrucción arriba mencionada **IMUL DWORD [EDX*4+4EB5D8]** con el segundo carácter de mi nombre.

De acuerdo a la tabla de constantes, a la posición 2 le corresponde el número **63**h. Aquí, tienes los 2 primeros **DWORD**'s de la tabla. Vemos el **D9**h para la primera posición y el **63**h para la segunda.

004EBB5D8 D9 00 00 00 **63** 00 00 00 00

Entonces, cuando multiplique el primer carácter del nombre, en mi caso, en **AL** estará el **49**h ["I"] y como es el primer carácter, **EDX** (contador) valdrá 1. **[EDX*4+4EB5D8]** sería **1*4==4 + 4EB5D8** (Inicio de la tabla de constantes) el resulta será el contenido del primer **DWORD** o sea, **D9**h. Por lo tanto, al ser **IMUL**, multiplicará **D9**h * **49**h, y cuando **EDX** valga 2, **[EDX*4+4EB5D8]** apuntará al segundo **DWORD 63**h y así hasta llegar al final del largo del nombre. Como mi nombre tiene 7 letras, loopeará 7 veces llegando hasta **F0**h que es el 7mo **DWORD** de la tabla de constantes.

Función de la rutina:

Miren como usa los valores de la tabla de constantes.

En **rojo** las constantes y en **azul** los caracteres de mi nombre.

1) MULT 1er car "**I**" [49] * **1**==[**49**] ;[Posición 1]
2) MULT 1er car "**I**" [49] * **D9**==[**3DE1**] ;[1er Byte del DWORD 1]

3) MULT 2ndo car "**v**" [76] * **2**==[**EC**] ;[Posición 2]
4) MULT EC * **63**==[**5B44**] ;[2ndo Byte del DWORD 2]

5) SUMA **5B44 + 3DE1**==[**9925**]

6) MULT 3ER car "**i**" [69] * **3**==[**13B**] ;[Posición 3]
7) MULT 13B * **58**==[6C48] ;[3er Byte del DWORD 3]

8) SUMA **6C48 + 9925**==[**1056D**]

9) MULT 4to car "**n**" [6E] * **4**==[**1B8**] ;[Posición 4]
10) MULT **1B8** * **22**==[**3A70**] ;[4to Byte del DWORD 4]

11) SUMA 1056D + 3A70==[13FDD]

12) MULT 5to car "s" [73] * 5==[23F] ;[Posición 5]
13) MULT 23F * 3E==[8B42] ;[5to Byte del DWORD 5]

14) SUMA 13FDD + 8B42==[1CB1F]

15) MULT 6to car "o" [6F] * 6==[29A] ;[Posición 6]
16) MULT 29A * 93==[17E6E] ;[6to Byte del DWORD 6]

17) SUMA 1CB1F + 17E6E==[3498D]

18) MULT 7mo car "n" [6E] * 7==[302] ;[Posición 7]
19) MULT 302 * F0==[2D1E0] ;[7mo Byte del DWORD 7]

20) SUMA 2D1E0 + 3498D==[61B6D]

RESULTADO FINAL==61B6D
En decimal==400237

Nombre: Ivinson
Serial: 400237

ALGUNOS DETALLES.

Probando el keygen, me di cuenta que el máximo de caracteres para el nombre es de 18 y que a partir de ciertos dígitos en el nombre el serial tenía un largo de 7 y el cual tiene que tener 6 dígitos solamente. Y si borraba el primero o el último carácter que me daba el keygen para cada nombre dejando solo 6, funcionaba perfectamente porque se habilitaba el botón “OK”.

Ej. Serial: 1234567. Si borro el 1, queda 23456. Serial definitivo.

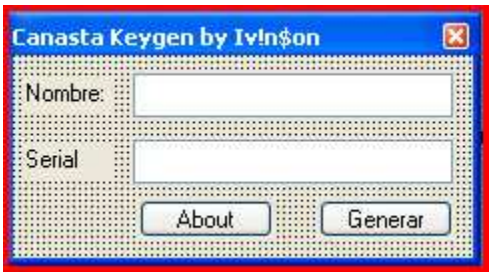
Por lo que, la solución que se me ocurrió fue después que el keygen escribía la clave correcta en el edit2, llamara a GetDlgItemTextA,

cogiera el largo y lo compara con 6 y si era mayor, moviera la string de la clave a **ESI** y hacía **INC ESI**. Luego, lo pusiera en el edit2 de nuevo. ☺

KEYGENING

Como siempre, abramos **RadASM** y creamos un nuevo proyecto.

Y hacemos la interface.



ARCHIVO .ASM

Como siempre les digo. El único código que agregué está en fondo azul. Comentarios en ;rojo.

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

start:
        invoke GetModuleHandle,NULL
        mov         hInstance,eax
    invoke InitCommonControls
        invoke         DialogBoxParam,hInstance,IDD_DIALOG1,NULL,addr
DlgProc,NULL
        invoke ExitProcess,0

DlgProc proc
hWin:HWND,uMsg:UINT,wParam:WPARAM,lParam:LPARAM

        mov         eax,uMsg
        .if eax==WM_INITDIALOG

        .elseif eax==WM_COMMAND
;*************************************************************************
;Escribimos la tabla de constantes.

MOV [buffer_const]    ,0D9h
MOV [buffer_const+4h] ,63h
MOV [buffer_const+8h] ,58h
MOV [buffer_const+0Ch],22h
MOV [buffer_const+10h],3Eh
MOV [buffer_const+14h],93h
MOV [buffer_const+18h],0F0h
MOV [buffer_const+1CH],08h
```

```
MOV [buffer_const+20h],34h
MOV [buffer_const+24h],62h

MOV [buffer_const+28h],1Bh
MOV [buffer_const+2Ch],0BFh
MOV [buffer_const+30h],0D7h
MOV [buffer_const+34h],0B9h
MOV [buffer_const+38h],6Fh
MOV [buffer_const+3Ch],4Ah
MOV [buffer_const+40h],5Ah
MOV [buffer_const+44h],0B2h
MOV [buffer_const+48h],84h
MOV [buffer_const+4Ch],24h
MOV [buffer_const+50h],11h
;*************************************************************************
        mov eax,wParam  ;Pasamos el control a EAX.
        .if eax!=0              ;¿Hay algo en EAX?

.if eax==1005               ;Si presionamos el botón "Generar".
invoke GetDlgItemText,hWin,1001,addr nombre,20 ;Coge el texto.
mov [largo_del_nombre],eax      ;Guardamos el largo.
CMP [largo_del_nombre],4   ;Compara el largo con 4.
JBE MAS_CAR  ;Salta a mostrar el MsgBox "min 5, max 18"

CMP [largo_del_nombre], 19  ;Compara largo con 19.
JAE MAS_CAR   ;Si es igual o mayor a 19. MsgBox "min 5, max 18".
        LEA ESI,[nombre] ;Si no, mueve el nombre a ESI.

prueba_letras:   ;Bucle para ver si hay solo letras.
MOV AL,BYTE PTR DS:[ESI]
INC ESI
TEST AL,AL
JE rutina1
CMP AL,41h    ;41h == "A" si un car es menor que 41h, no es una letra.
JB solo_letras   ;Salta a MsgBox "Solo letras".
JMP prueba_letras    ;Sigue chequeando los cars.
;*************************************************************************
rutina1:
XOR EAX,EAX ;Limpia EAX.
 ;D96358223E93F00834621BBFB96F4A5AB2842411---Constantes

MOV [buffer1],0  ;Inicia buffer1 a  "0".
```

```
 rutina:

 LEA ESI,[nombre]  ;Mueve el nombre a ESI.

;Ejemplo: ESI== "Ivinson"

 proceso:   ;Comienza el proceso de creación de la clave.
 MOVZX EAX,BYTE ptr [ESI]  ;Mueve cada byte a EAX.
 INC ESI   ;Prepara el siguiente carácter.
 MOV EDX, [buffer1] ;Buffer1 o contador1 a EDX.
 ADD EDX,1   ;Le suma 1.
 IMUL EDX ;Multiplica EDX x EAX.
 MOV EDX, [buffer1] ;Buffer1 o contador1 a EDX.
 IMUL [EDX*4+buffer_const] ;Mult result * constant de la tabla.
 MOV [suma1],EAX  ;Va guardando.
 MOV EAX, [suma1] ;Mueve a EAX para luego sumar.
 ADD  [suma2],EAX ;Va sumando y guardando en var. Suma2.
 MOV EAX,  [largo_del_nombre] ;Mueve largo a EAX.
 DEC EAX          ;Resta uno al largo.
 MOV   [largo_del_nombre],EAX  ;Guarda el result en esa var.
 INC buffer1    ;Incrementa el contador.
 CMP [largo_del_nombre],0 ;¿Se acabaron los cars.?
 JNZ proceso ;Si no, sigue procesando cars.

 MOV EDX,[suma2] ;Mueve la suma total a EDX.
;Llama a wsprintf para pasarlo a decimal.
invoke wsprintf,ADDR serialbuffer ,ADDR formatodec ,edx
;Lo escribe en el edit2 "Serial".
INVOKE SetDlgItemText,hWin,1002,addr [serialbuffer]
XOR EAX,EAX  ;Limpia EAX.
;Coge la escrito en edit2 con GetDlgItemTextA.
INVOKE GetDlgItemText,hWin,1002,addr largo_final,20
CMP EAX,6      ;¿Tiene 6 dígitos el serial?
JBE SEGUIR   ;Si es menor o igual a 6, sigue normal.
LEA ESI,[serialbuffer] ;Si es mayor a 6, lo mueve a ESI.
INC ESI ;Le quita el primer dígito al serial final.
INVOKE SetDlgItemText,hWin,1002,addr [ESI] ;Lo escribe de nuevo.

SEGUIR:
MOV [suma2],0 ;Limpia los buffer, por si metemos otro nombre.
MOV [suma1],0 ;Limpia los buffer, por si metemos otro nombre.
```

```
.elseif eax==1006 ;Si presionamos el botón "About"...

;Nos muestra el MsgBox.
invoke MessageBox,hWin,addr titulomsg ,addr textomsgbox,MB_OK
.endif
        .endif
        jmp fin  ;Salta al final.



        MAS_CAR:   ;MsgBox "Más caracteres"
        INVOKE MessageBox,hWin, addr texto,addr titulo,MB_OK
        RET

        solo_letras:     ;MsgBox "Más letras".
        INVOKE MessageBox,hWin, addr texto2,addr titulo,MB_OK
RET

fin:
;*************************************************************************
        .elseif eax==WM_CLOSE
                invoke EndDialog,hWin,0
        .else
                mov          eax,FALSE
                ret
        .endif
        mov          eax,TRUE
        ret

DlgProc endp

end start
```

ARCHIVO .INC

```
include windows.inc
include kernel32.inc
include user32.inc
include Comctl32.inc
include shell32.inc
includelib kernel32.lib
includelib user32.lib
includelib Comctl32.lib
includelib shell32.lib

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

.const

IDD_DIALOG1                 equ 101

.data?

hInstance               dd ?
;*****************************************************************************
.data

nombre dd 20 dup (0)
largo_del_nombre dd 2 dup (0)
buffer1 dd 2 dup (0)
suma1 dd 4 dup (0)
suma2 dd 4 dup (0)
buffer_const dd 50 dup (0)

buffer2 dd 4 dup (0)
largo_final dd 5 dup (0)

formatodec      db '%lu',0
serialbuffer    dd 20 dup (0)
```

```
titulo db "Ivinson",0
texto db "Nombre: min 5 letras y max 18. Y solo letras",0
texto2 db "Solo letras en el nombre :)",0
;MsgBox About
textomsgbox  db "About-- Author:Iv!n$on",0
titulomsg    db "GrEetZ to: All CLS members",0
;*****************************************************************
```

PROBANDO EL KEYGEN

PROBANDO EL SERIAL

Se habilitó el botón. Aceptemos.

Por fin, terminamos. Keygen listo. Gracias por leer.

Contacto: ipadilla63@gmail.com

Frases célebres:

Solo sé que no crackeo nada.

Crackear o no crackear.

Primero crackeo, luego existo.

Hasta el tuto 9. See you.