aRC-FL-Cracking 006 (01/08/2003)

Terminando lo empezado

(por Furious Logic [aRC])

Advertencia

Antes de poner en práctica el presente documento y cualquier material asociado al mismo, sea éste naturaleza tangible o intangible usted debe estar totalmente de acuerdo con todos los términos y condiciones siguientes:

Del software

Cualquier software se proporciona tal como está, sin ninguna garantía expresa ni implícita de ningún tipo.

aRC no garantiza ni asume responsabilidad alguna en cuanto a la integridad o exactitud de cualquier información contenida en el software.

Ni los miembros ni los colaboradores ni los invitados aRC se hacen responsables por el uso que se le pueda dar al software.

Al examinar, copiar, ejecutar, instalar o utilizar el software, el lector está aceptando su total conformidad con todos los términos y condiciones enunciados.

Del documento

Al abrir este documento, el lector acepta incondicionalmente su total y exclusiva responsabilidad legal, de acuerdo a las leyes vigentes en su país, por el uso de las técnicas experimentales, educativas y/o de investigación aquí vertidas en materia de programación especializada de computadoras.

En caso de discrepar con alguno de los puntos descritos, deberá eliminar inmediatamente el presente documento y todo material asociado al mismo.

Agradecimientos

A Lotus Word Pro de Lotus Corporation (TM), incluido en la suite de oficina Lotus Millenium 9.

A FontLab 3.00F de FontLab Developers Group por permitirnos asignar permiso completo a las fuentes true type protegidas contra copia y a Acrobat Distiller 5.0 de Adobe Systems por su excelente resultado en la creación del documento electrónico en formato PDF.

A Mario0_1, gran amigo colaborador de aRC. Esperamos tenerlo en el canal en algún momento.

A Jeremy Gordon por su magnífico ensamblador GoAsm actualmente en la versión 0.43 y disponible en http://www.godevtool.com. El ensamblador más sencillo, potente, veloz y reducido que hayamos podido examinar.

A nuestros grandes amigos a573r10n y Axel Silver, ambos miembros oficiales de aRC desde el viernes 14 de marzo del 2003. ¡Felicitaciones colegas! Desde aquí les deseamos grandes éxitos y mayores logros por su denodado esfuerzo, recompensado con justicia. Este número está dedicado a ellos, esperando pronto continuar con los demás proyectos que tenemos para ustedes.

Generalidades

Después de algún tiempo plagado de "dimes" y "diretes", al fin retomamos las riendas de este vuestro aRC-FL-Cracking. Nuestra palabra está empeñada y pretendemos a todas luces cumplirla. Al mismo tiempo, todos estos meses habrán servido para que hayan perfeccionado sus habilidades en los diversos lenguajes de programación que circulan, o al menos en uno solo de ellos.

Diseñaremos un pseudocódigo de un *patch* ("programa que realiza los cambios") aplicable en cualquier intérprete (Visual Basic), pseudocompilador (C#) y compilador (Delphi, C++Builder, Visual C++, Dev-C++, MASM32, GoAsm). Finalmente, proporcionaremos algunos ejemplos de código, no optimizados y con bugs, en algunos lenguajes un par de obsequios sobre VCL.

Deben acompañar a este documento los siguientes archivos:

- ✓ parchame.exe (40,960 Bytes. Archivo para prácticas.)
- ✓ parchame.zip (38,798 Bytes. Código fuente del archivo de prácticas)
- ✓ CrackVB.zip (2,506 Bytes. Código fuente de un *patch* en Visual Basic)
- ✓ CrackBD.zip (6,938 Bytes. Código fuente de un *patch* en Borland Delphi)
- ✓ CrackCB.zip (11,303 Bytes. Código fuente de un *patch* en C++Builder)
- ✓ CrackDC.zip (4,592 Bytes. Código fuente de un *patch* en Dev-C++)
- ✓ CrackVC.zip (9,591 Bytes. Código fuente de un *patch* en Visual C++)
- ✓ CrackMA.zip (2,032 Bytes. Código fuente de un *patch* en MASM32)
- ✓ CrackGA.zip (2,827 Bytes. Código fuente de un *patch* en GoAsm)
- ✓ SinvclBD.zip (1,637 Bytes. Código fuente de un *patch* en Borland Delphi sin utilizar VCL)
- ✓ SinVCLCB.zip (1,719 Bytes. Código fuente de un *patch* en C++Builder sin utilizar VCL)

Cómo para variar un poco con la música, pues muchos no conocen a Enigma ni a Deep Forest ni a Vangelis, esta vez disponemos de una selección de colección (Joe Cocker, Lionel Richie, Sting, Joe Esposito, John Lenon, Lobo, Michael Bolton, Bee Gees) que llegó a nuestras manos bajo "misteriosas circunstancias". ¡ Muchas gracias ... !).

¡Advertencia!: Puede producir efectos maniaco-depresivos en los oyentes sensibles.

Pseudocódigo

Para poder hacer un *patcher* ("programa parchador"), es indispensable contar con la siguiente información:

- ✓ Nombre del archivo a parchar
- ✓ Tamaño del archivo a parchar
- ✓ Desplazamientos de los bytes a modificar (ú *offsets*)
- ✓ Valores originales de los bytes a modificar
- ✓ Valores nuevos para los bytes a modificar

Utilicemos nuestro archivo de práctica **parchame.exe**. Debido a que nuestro propósito es realizar un *patcher* o "programa parchador", esta vez SÍ proporcionamos los desplazamientos del código, aunque esperamos que usted los ubique por sí mismo sin necesidad de leer el resto de esta sección.

1. Ejecutamos **parchame.exe** y anotamos el mensaje de error. Son 2 ocurrencias: al cargar el programa y al hacer click en el gráfico. El mensaje es:

   ```
   Aun no me has crackeado
   ```

2. Sacamos una copia, la llamamos **1.exe** y desensamblamos esta copia en W32Dasm (W32)
3. Buscamos el mensaje de error anotado arriba y notamos 2 ocurrencias del mismo:

   1ra. ocurrencia:

   ```
   ...
   :00401033 E84CC00000                call 0040D084

   * Possible StringData Ref from Data Obj ->"-=[Error]=-"
                                           ¦
   ```

```
:00401038 6814204000              push 00402014

* Possible StringData Ref from Data Obj ->"Aun no me has crackeado"
                                   ¦
:0040103D 6830204000              push 00402030
:00401042 E830020000              call 00401277
:00401047 C3                      ret
...
```

2da. ocurrencia:

```
...
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
¦:0040106F(C)
¦

* Possible StringData Ref from Data Obj ->"-=[Error]=-"
                                   ¦
:00401082 6814204000              push 00402014

* Possible StringData Ref from Data Obj ->"Aun no me has crackeado"
                                   ¦
:00401087 6830204000              push 00402030
:0040108C E8E6010000              call 00401277
...
```

Cuya llamada condicional proviene de:

```
...
:0040106C 83F801                  cmp eax, 00000001
:0040106F 7511                    jne 00401082
...
```

4. Para la primera ocurrencia, determinamos que después del `call 0040D084` en **00401033**, empiezan los **push** que serán los parámetros a pasarle al `call 00401277` en **00401042**, que a su vez es la llamada al mensaje de error. Finalmente notamos un código `C3`, un `ret`, con el que termina el procedimiento general. Reemplazaremos el código `68` con un código `C3` en **00401038**. Para la 2da ocurrencia existe un `jne 00401082` que sustituiremos por un `inc eax` y `dec eax`. Es decir, reemplazaremos los códigos `7511` con los códigos `40` en **0040106F** y `48` en **00401070**.
5. Resumiendo nuestros hallazgos en una tabla tenemos:

DESP. VIRTUAL	DESP. ABSOLUTO	BYTE ACTUAL	NUEVO BYTE
00401038h	438h	68h	C3h
0040106Fh	46Fh	75h	40h
00401070h	470h	11h	48h

Con esta información crearemos el pseudocódigo:

```
1.  // Declaración de constantes:
2.  MAXBYTES                 = 3
3.  NombreArchivo            = "parchame.exe"
4.  TamanoArchivo            = 40960
5.  Desplazamiento[MAXBYTES] = 438h,46Fh,470h
6.  ByteActual[MAXBYTES]     = 68h,75h,11h
7.  NuevoByte[MAXBYTES]      = C3h,40h,48h
8.  Titulo                   = "Crack para Parchame 1.0"
9.  Autor                    = "aDVANCED rESEARCH cOMMUNITY (c) 2003"
10.
11. // Declaración de variables:
12. HandleArchivo  : entero
13. K, ByteLeido   : byte
14. TamanoObtenido : enterolargo
```

```
15.
16. // Procedimiento general
17. MostrarMensaje(Titulo)
18. MostrarMensaje(Autor)
19. HandleArchivo = AbrirArchivo(NombreArchivo)
20. Si HandleArchivo = 0
21.    MostrarMensaje("Error:  No se pudo abrir el archivo")
22.    SalirConError 1
23. DeLoContrario
24.    TamanoObtenido = ObtenerTamanoArchivo(HandleArchivo)
25.    Si TamanoObtenido <> TamanoArchivo
26.       CerrarArchivo(HandleArchivo)
27.       MostrarMensaje("Error:  El archivo no es el apropiado")
28.       SalirConError 2
29.    DeLoContrario
30.       Desde K=1 hasta MAXBYTES hacer
31.          UbicarPosicion(Desplazamiento[K])
32.          ByteLeido = LeerByteArchivo()
33.          Si (ByteLeido = ByteActual[K])
34.             EscribirByteArchivo(NuevoByte[K])
35.          DeLoContrario
36.             CerrarArchivo(HandleArchivo)
37.             MostrarMensaje("Error:  El archivo no coincide o ya fue crackeado")
38.             SalirConError 3
39.          FinSi
40.       FinDesde
41.       CerrarArchivo(HandleArchivo)
42.       MostrarMensaje("Operacion completada con exito")
43.       SalirConError 0
44.    FinSi
45. FinSi
```

Resulta más que obvio, que será muy sencillo trabajar con este pseudocódigo si lo entendemos. Analicemos lo que hace el programa. En la línea 2 se declara una variable **MAXBYTES** que contiene la cantidad de bytes a editar. Nos servirá para definir los arreglos de las líneas 5 al 7, y para controlar el bucle de la línea 30. **NombreArchivo** se llama indirectamente en la línea 19. **TamanoArchivo** nos permite verificar si estamos abriendo el archivo indicado en las líneas 24 a 28. El bucle de las líneas 30 a la 40, verifica que el byte actual corresponda al que debe ser editado. Solo bajo esta condición, se realiza el proceso de grabación contemplado en las líneas 33 y 34. Finalmente, cerramos el archivo y emitimos un mensaje apropiado en las líneas 41 y 42.

Visual Basic

Necesitaremos un formulario o *Form*, con 2 etiquetas de texto o *Label* vacías y un botón de comando o *CommandButton* con el título &Crack. El código correspondiente es:

```
Option Base 1
--------------------------------------------------------------------------------
Private Sub Command1_Click()
   Const NombreArchivo As String = "parchame.exe"
   Const TamanoArchivo As Long = 40960
   Const MAXBYTES As Byte = 3
   Dim Desplazamiento, ByteActual, NuevoByte       'Declaradas como Variant
   Dim HandleArchivo As Integer
   Dim K As Byte, ByteLeido As Byte
   Dim TamanoObtenido As Long
   Desplazamiento = Array(&H438, &H46F, &H470)
   ByteActual = Array(&H68, &H75, &H11)
   NuevoByte = Array(&HC3, &H40, &H48)
```

```
   HandleArchivo = FreeFile()
   If HandleArchivo = 0 Then
      MsgBox "Error:  No se pudo abrir el archivo"
      Exit Sub
   Else
      Open NombreArchivo For Binary Access Read Write Lock Read Write As HandleArchivo
      TamanoObtenido = LOF(HandleArchivo)
      If TamanoObtenido <> TamanoArchivo Then
         Close (HandleArchivo)
         MsgBox "Error:  El archivo no es el apropiado"
         Exit Sub
      Else
         Get HandleArchivo, 1, ByteLeido
         For K = 1 To MAXBYTES
            Get HandleArchivo, CLng(Desplazamiento(K)) + 1, ByteLeido
            If (ByteLeido = ByteActual(K)) Then
               Put HandleArchivo, CLng(Desplazamiento(K)) + 1, CByte(NuevoByte(K))
            Else
               Close (HandleArchivo)
               MsgBox "Error:  El archivo no coincide o ya fue crackeado"
               Exit Sub
            End If
         Next
            Close (HandleArchivo)
            MsgBox "Operacion completada con exito"
      End If
   End If
End Sub
-------------------------------------------------------------------------------
Private Sub Form_Load()
    Label1.Caption = "Crack para Parchame 1.0"
    Label2.Caption = "aDVANCED rESEARCH cOMMUNITY (c) 2003"
    Form1.Caption = "Crack"
End Sub
```

El ejecutable resultante SÍ requiere librerias adicionales (`msvbvm60.dll`). Conviene estudiar la documentación de las siguientes funciones:

`Array`	:	Asigna un arreglo a una variable de tipo Variant.
`FreeFile`	:	Devuelve un handle o identificador disponible para un archivo.
`Open`	:	Abre un archivo según handle o identificador indicado.
`LOF`	:	Devuelve el tamaño en bytes de un archivo abierto.
`Close`	:	Cierra un archivo abierto según handle o identificador.
`Get`	:	Lee n bytes desde una posición indicada y hacia una variable.
`Put`	:	Graba n bytes en una posición indicada y desde una variable.
`CLng`	:	Convierte una variable en tipo entero largo.
`CByte`	:	Convierte una variable en tipo byte.

Delphi

Antes de traducir el pseudocódigo hacia delphi, necesitamos un formulario o *Form*, un par de etiquetas de texto o *Label* y un botón o *Button* con el título "&Crack". El código necesario para `Unit1.pas` sería:

```
unit Unit1;

interface

  uses
    Classes, Controls, Forms, StdCtrls, Dialogs;

  type
```

```
    TForm1    = class(TForm)
      Label1  : TLabel;
      Label2  : TLabel;
      Button1 : TButton;
      procedure Button1Click(Sender: TObject);
      procedure FormCreate(Sender: TObject);
    end;

  var
    Form1 : TForm1;

implementation
{$R *.dfm}

  procedure TForm1.Button1Click(Sender: TObject);
    const
      MAXBYTES                = 3;
      NombreArchivo  : String  = 'parchame.exe';
      TamanoArchivo  : Longint = 40960;
      Desplazamiento : Array [1..MAXBYTES] of Integer = ($438,$46F,$470);
      ByteActual     : Array [1..MAXBYTES] of Byte    = ($68,$75,$11);
      NuevoByte      : Array [1..MAXBYTES] of Byte    = ($C3,$40,$48);
    var
      HandleArchivo  : File of Byte;         // Indica archivo binario
      K, ByteLeido   : Byte;
      TamanoObtenido : Longint;
    begin
      {$I-}                                  // Desactiva control de errores I/O
      AssignFile(HandleArchivo,NombreArchivo);
      Reset(HandleArchivo);
      if (IOResult<>0) then begin
        ShowMessage('Error:  No se pudo abrir el archivo');
        Halt(1);
      end
      {$I+}                                  // Reactiva control de errores I/O
      else begin
        TamanoObtenido := FileSize(HandleArchivo);
        if (TamanoObtenido<>TamanoArchivo) then begin
          CloseFile(HandleArchivo);
          ShowMessage('Error:  El archivo no es el apropiado');
          Halt(2);
        end
        else begin
          For K:=1 to MAXBYTES do begin
            Seek(HandleArchivo,Desplazamiento[K]);
            Read(HandleArchivo,ByteLeido);
            if (ByteLeido=ByteActual[K]) then begin
              Seek(HandleArchivo,Desplazamiento[K]);
              Write(HandleArchivo,NuevoByte[K]);
            end
            else begin
              CloseFile(HandleArchivo);
              ShowMessage('Error:  El archivo no coincide o ya fue crackeado');
              Halt(3);
            end;
          end;
          CloseFile(HandleArchivo);
          ShowMessage('Operacion completada con exito');
        end;
      end;
    end;

  procedure TForm1.FormCreate(Sender: TObject);
    begin
      Label1.Caption := 'Crack para Parchame 1.0';
```

```
    Label2.Caption := 'aDVANCED rESEARCH cOMMUNITY (c) 2003';
    Form1.Caption  := 'Crack';
  end;

end.
```

Solo nos queda aclarar que se ha utilizado el más sencillo código Delphi nativo. El ejecutable resultante NO requiere de librerias adicionales. Las funciones principales a estudiar serían:

AssignFile	:	Asigna un handle o identificador a un archivo.
Reset	:	Abre un archivo (por omisión para lectura-escritura)
FileSize	:	Devuelve el tamaño de un archivo abierto indicado por un handle.
Seek	:	Ubica el puntero de archivo en la posición especificada.
Read	:	Lee un grupo de bytes en una variable y avanza el puntero de archivo.
Write	:	Escribe un grupo de bytes desde una variable y avanza el puntero de archivo.
CloseFile	:	Cierra el archivo indicado por un handle o identificador.
Halt	:	Terminación anormal del programa con código de error.

C++Builder

Existen varios compiladores de C, entre ellos el clásico C++Builder, el rapidísimo Dev-C++ y Visual C++ del monopolio de Micro$$oft. Empezaremos con C++Builder por ser muy sencillo de implementar para los recién iniciados. Luego tomaremos a Dev-C++ y a Visual C++.

Necesitamos un formulario o *Form*, un par de etiquetas de texto o *Label* y un botón o *Button* con el título "&Crack". El código necesario para **unit1.cpp** sería:

```
//---------------------------------------------------------------------------
//#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

  //---------------------------------------------------------------------------
  __fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
  { // Este es el constructor.  Aqui no va nada
  }
  //---------------------------------------------------------------------------

  void __fastcall TForm1::FormCreate(TObject *Sender)
  { Label1->Caption = "Crack para Parchame 1.0";
    Label2->Caption = "aDVANCED rESEARCH cOMMUNITY (c) 2003";
    Form1->Caption  = "Crack";
  }

  //---------------------------------------------------------------------------
  void __fastcall TForm1::Button1Click(TObject *Sender)
  { const byte MAXBYTES                = 3;
    const char *NombreArchivo          = "parchame.exe";
    const long TamanoArchivo           = 40960;
    const int Desplazamiento[MAXBYTES] = {0x438,0x46F,0x470};
    const byte ByteActual[MAXBYTES]    = {0x68,0x75,0x11};
    const byte NuevoByte[MAXBYTES]     = {0xC3,0x40,0x48};
    int HandleArchivo;
    byte K, ByteLeido;
    long TamanoObtenido;
    HandleArchivo = FileOpen(NombreArchivo,fmOpenReadWrite);
    if (HandleArchivo==-1)
    { MessageBoxA(0,"Error:  No se pudo abrir el archivo",0,MB_OK);
```

```
      exit(1);
    }
    else
    { TamanoObtenido = FileSeek(HandleArchivo,0,2);
      if (TamanoObtenido!=TamanoArchivo)
      { FileClose(HandleArchivo);
        MessageBoxA(0,"Error:  El archivo no es el apropiado",0,MB_OK);
        exit(2);
      }
      else
      { for(K=0;K<MAXBYTES;K++)
        { FileSeek(HandleArchivo,Desplazamiento[K],0);
          FileRead(HandleArchivo,&ByteLeido,1);
          if (ByteLeido==ByteActual[K])
          { FileSeek(HandleArchivo,Desplazamiento[K],0);
            FileWrite(HandleArchivo,&NuevoByte[K],1);
          }
          else
          { FileClose(HandleArchivo);
            MessageBoxA(0,""Error:  El archivo no coincide o ya fue crackeado",0,MB_OK);
            exit(3);
          }
        }
        FileClose(HandleArchivo);
        MessageBoxA(0,"Operacion completada con exito","OK",MB_OK);
        exit(0);
      }
    }
  }
  //---------------------------------------------------------------------------
  //Obs.:   - Si su version de C++Builder falla la compilacion, pruebe quitando el
  //          comentario "//" de la linea 2: //#include <vcl.h>
  //        - Desactive la opcion Menu Project, Options, Packages, Build with
  //          Runtime packages
  //        - Desactive la opcion Menu Project, Options, Compiler, Debugging, Debug
  //          information
  //        - Desactive las 3 opciones Menu Project, Options, Linker: Create debug
  //          information, Use dynamic RTL y Use debug libraries.
```

Se puede observar poca variación respecto al código en Delphi. Para empezar, tanto Delphi como C++Builder están hechos por la misma compañía Borland (¿o es Inprise?). Además, ambos compiladores utilizan un tipo de componentes llamados VCL (de *Visual Component Library*) que producen resultados similares en cuanto a tamaño y velocidad. El ejecutable resultante NO requiere de librerias adicionales. Funciones a revisar:

FileOpen	:	Abre un archivo y devuelve su handle o identificador.
FileClose	:	Cierra un archivo a partir del handle o identificador proporcionado.
MessageBoxA	:	Muestra un mensaje. Versión de 32 bits.
FileSeek	:	Ubica el puntero de archivo en la posición especificada.
FileRead	:	Lee datos desde un archivo hacia una variable y avanza el puntero.
FileWrite	:	Escribe datos hacia un archivo desde una variable y avanza el puntero.
exit	:	Termina el programa con código de error.

Dev-C++

Excelente compilador de C++ que conocimos desde su versión 4.x gracias a la recomendación de un amigo investigador del underground ya desaparecido de la escena. En su reciente versión 5.0 aún no tiene ayuda ni manual alguno, excepto una breve introducción a la programación en C++. Por lo tanto, debemos aclarar que el nivel para utilizarlo debe ser superior al nivel promedio. Si algún lector ha tenido el privilegio de disfrutar de Borland C++ (que no es C++Builder), entonces considérese aventajado. Sin embargo, no nos engañemos pensando que un compilador de C++ es simplemente mejor que otro, lo que sucede es que tanto Dev-C++ como Visual C++ utilizan las API de Windows

32, que otros compiladores también pueden emplear. No obstante, en algunos casos, Dev-C++ presenta ejecutables de tamaño superior a Visual C++, al menos en el caso del código de nuestro *patch*. Aunque en la mayoría casos, el resultado es un ejecutable más pequeño que el devuelto por otros compiladores de C++.

Este código, en todo caso, es C++ empleando las API de Windows 32, aplicable también a otros compiladores de C, además de Dev-C++.

```
#include <windows.h>

HWND hbutton;

LRESULT CALLBACK WindowProcedure (HWND, UINT, WPARAM, LPARAM);

int WINAPI WinMain (HINSTANCE hThisInstance,HINSTANCE hPrevInstance,
                    LPSTR lpszArgument,int nFunsterStil)
{ const char *NombreClase = "WinApp";
  const char *Titulo      = "Crack para Parchame 1.0";
  const char *Autor       = "aDVANCED rESEARCH cOMMUNITY (c) 2003";
  HWND hwnd;
  MSG messages;
  WNDCLASSEX wincl;
  // Nueva clase WinApp (Ventana principal)
  wincl.cbSize        = sizeof(WNDCLASSEX);
  wincl.style         = CS_HREDRAW|CS_VREDRAW;
  wincl.lpfnWndProc   = WindowProcedure;
  wincl.cbClsExtra    = 0;
  wincl.cbWndExtra    = 0;
  wincl.hInstance     = hThisInstance;
  wincl.hIcon         = LoadIcon(NULL,IDI_APPLICATION);
  wincl.hCursor       = LoadCursor(NULL,IDC_ARROW);
  wincl.hbrBackground = (HBRUSH) COLOR_BACKGROUND;
  wincl.lpszMenuName  = NULL;
  wincl.lpszClassName = NombreClase;
  wincl.hIconSm       = LoadIcon(NULL,IDI_APPLICATION);
  if (!RegisterClassEx(&wincl))
    return 0;
  hwnd = CreateWindowEx(0,NombreClase,"Crack",WS_VISIBLE|WS_SYSMENU|WS_MINIMIZEBOX|
                        WS_MAXIMIZEBOX,CW_USEDEFAULT,CW_USEDEFAULT,316,192,
                        HWND_DESKTOP,NULL,hThisInstance,NULL);
  // Clase STATIC que no necesita ser registrada (Titulo)
  CreateWindowEx(0,"STATIC",Titulo,WS_CHILD|WS_VISIBLE|SS_CENTER,16,16,273,33,hwnd,
                 NULL,hThisInstance,NULL);
  // Clase STATIC que no necesita ser registrada (Autor)
  CreateWindowEx(0,"STATIC",Autor,WS_CHILD|WS_VISIBLE|SS_CENTER,16,56,273,33,hwnd,
                 NULL,hThisInstance,NULL);
  // Clase BUTTON que no necesita ser registrada
  hbutton = CreateWindowEx(0,"BUTTON","&Crack",WS_CHILD|WS_VISIBLE|BS_PUSHBUTTON|
                           BS_CENTER|BS_VCENTER,115,104,75,25,hwnd,NULL,
                           hThisInstance,NULL);
  while (TRUE==GetMessage(&messages,NULL,0,0))
  { TranslateMessage(&messages);
    DispatchMessage(&messages);
  }
  return messages.wParam;
}

LRESULT CALLBACK WindowProcedure (HWND hwnd,UINT message,WPARAM wParam,
                                  LPARAM lParam)
{ const byte MAXBYTES                = 3;
  const char *NombreArchivo          = "parchame.exe";
  const long TamanoArchivo           = 40960;
  const int Desplazamiento[MAXBYTES] = {0x438,0x46F,0x470};
  const byte ByteActual[MAXBYTES]    = {0x68,0x75,0x11};
  const byte NuevoByte[MAXBYTES]     = {0xC3,0x40,0x48};
  HANDLE HandleArchivo;
```

```
  byte K, ByteLeido;
  DWORD TamanoObtenido, temp;
  switch (message)
  { case WM_COMMAND:
      if (hbutton==(HWND)lParam)
      { HandleArchivo = CreateFile(NombreArchivo,GENERIC_READ|GENERIC_WRITE,
                                   FILE_SHARE_READ,NULL,OPEN_EXISTING,
                                   FILE_ATTRIBUTE_NORMAL,0);
        if (HandleArchivo==INVALID_HANDLE_VALUE)
        { MessageBox(hwnd,"Error:  No se pudo abrir el archivo",0,MB_OK);
          exit(1);
        }
        else
        { TamanoObtenido = SetFilePointer(HandleArchivo,0,NULL,FILE_END);
          if (TamanoObtenido!=TamanoArchivo)
          { CloseHandle(HandleArchivo);
            MessageBox(hwnd,"Error:  El archivo no es el apropiado",0,MB_OK);
            exit(2);
          }
          else
          { for (K=0;K<MAXBYTES;K++)
            { SetFilePointer(HandleArchivo,Desplazamiento[K],NULL,FILE_BEGIN);
              ReadFile(HandleArchivo,&ByteLeido,1,&temp,NULL);
              if (ByteLeido==ByteActual[K])
              { SetFilePointer(HandleArchivo,Desplazamiento[K],NULL,FILE_BEGIN);
                WriteFile(HandleArchivo,&NuevoByte[K],1,&temp,NULL);
              }
              else
             { CloseHandle(HandleArchivo);
               MessageBox(hwnd,"Error:  El archivo no coincide o ya fue crackeado",0,MB_OK);
               exit(3);
             }
            }
            CloseHandle(HandleArchivo);
            MessageBox(hwnd,"Operacion completada con exito","OK",MB_OK);
            PostQuitMessage(0);
          }
        }
      }
      break;
    case WM_DESTROY:
      PostQuitMessage(0);
      break;
    default:
      return DefWindowProc(hwnd,message,wParam,lParam);
  }
  return 0;
}
//Obs.:   - Si en cambiamos WNDCLASSEX wincl, por WNDCLASS wincl, y utilizamos
//          RegisterClass(&wincl) en lugar de RegisterClassEx(&wincl), eliminando
//          previamente las lineas wincl.cbSize = sizeof(WNDCLASSEX) y tambien
//          wincl.hIconSm = LoadIcon(NULL,IDI_APPLICATION), observaremos un notable
//          aumento del ejecutable resultante que no ocurre con otros compiladores
//          de C++.
//        - Menu Project, Project Options, Compiler settings, Linker y en Generate
//          debugging information seleccionar No
```

Consulte el archivo *Win32 Developer's Reference* o en su defecto *Win32 Programmer's Reference*, y también sus enciclopedias de lenguaje C++, para obtener la documentación detallada de:

RegisterClassEx	:	Registra en memoria la estructura de una ventana nueva.
CreateWindowEx	:	Crea una ventana, un botón, texto, etc. standard o registrado.
GetMessage	:	Obtiene el primer mensaje que se encuentre en la cola de mensajes.
TranslateMessage	:	Traduce el mensaje hacia modo texto.
DispatchMessage	:	Despacha un mensaje hacia un procedimiento de ventana.

`CreateFile`	:	Crea o abre un archivo y devuelve su handle.
`SetFilePointer`	:	Posiciona el puntero de archivo en la ubicación especificada.
`CloseHandle`	:	Cierra un archivo a través de su handle.
`ReadFile`	:	Lee n bytes desde un archivo hacia una variable y avanza el puntero.
`WriteFile`	:	Escribe n bytes hacia un archivo desde una variable y avanza el puntero.
`PostQuitMessage`	:	Realiza un requerimiento de salida del programa.
`DefWindowProc`	:	Llama al procedimiento manejador de eventos de una ventana.

Es probable que algún lector, haya notado la similitud con el código de Visual C++. Algunos lectores habrán abierto el ejecutable resultante con un programa como eXeScope 6.30 para determinar que librerías emplea y al instante habrán descubierto la clásica librería `msvcrt.dll`, y que también es común a los programas generados con Visual C++. Así, el tamaño de la DLL, aunque ya viene con winbugs, sumado al tamaño del ejecutable resultante, nos daría algo así como 380 KiB (Recuerde la lección aRC-FL-Cracking 005 sobre las diferencias entre KB y KiB).

Dev-C++ es un magnífico compilador de C++ perteneciente al grupo de compiladores GCC (*GNU Compiler Colection* o Colección de compiladores GNU) realizado por la compañía Bloodshed Software. Por lo tanto, es gratuito desde su creación y fácilmente descargable de:

Bloddshed Software Website	: http://www.bloodshed.net
Mingw Compiler Website	: http://www.mingw.org
Dev-C++ discussion forums	: http://www.bloodshed.net/forum
Dev-C++ users mailing list	: http://www.bloodshed.net/devcpp-ml.html
Dev-C++ Resource Site	: http://www.bloodshed.net/dev

Visual C++

Como preámbulo, recordemos que Visual C++ es el mejor producto que han realizado los programadores de Silicon Valley, en lo que a productos Micro$$oft se refiere. Utiliza las librerias MFC o puede trabajar sin ellas. Es uno de los competidores del afamado Borland C++ cuya robustez y consistencia pocas veces ha tenido competidor.

Sería sumamente interesante que los programadores que desarrollaron Visual C++ fuesen "promovidos" hacia el área de desarrollo de las nuevas versiones de Windows.

Veamos el código para `project1.cpp`:

```
#include "stdafx.h"

HWND hbutton;

ATOM RegistrarVentana (HINSTANCE hInstance);
HWND MostrarVentana (HINSTANCE);
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM);
int APIENTRY WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                      LPSTR lpCmdLine, int nCmdShow)
{ const char *Titulo = "Crack para Parchame 1.0";
  const char *Autor  = "aDVANCED rESEARCH cOMMUNITY (c) 2003";
  MSG msg;
  HWND hwnd;
  RegistrarVentana(hInstance);
  hwnd = MostrarVentana(hInstance);
  // Clase STATIC que no necesita ser registrada (Titulo)
  CreateWindowEx(0,"STATIC",Titulo,WS_CHILD|WS_VISIBLE|SS_CENTER,16,16,273,33,hwnd,
                 NULL,hInstance,NULL);
  // Clase STATIC que no necesita ser registrada (Autor)
  CreateWindowEx(0,"STATIC",Autor,WS_CHILD|WS_VISIBLE|SS_CENTER,16,56,273,33,hwnd,
                 NULL,hInstance,NULL);
  // Clase BUTTON que no necesita ser registrada
  hbutton = CreateWindowEx(0,"BUTTON","&Crack",WS_CHILD|WS_VISIBLE|BS_PUSHBUTTON|
                           BS_CENTER|BS_VCENTER,115,104,75,25,hwnd,NULL,
                           hInstance,NULL);
  while (GetMessage(&msg,NULL,0,0))
  { TranslateMessage(&msg);
```

```
    DispatchMessage(&msg);
  }
  return msg.wParam;
}

ATOM RegistrarVentana (HINSTANCE hInstance)
{ const char *szWindowClass = "WinApp";
  WNDCLASSEX wcex;
  wcex.cbSize        = sizeof(WNDCLASSEX);
  wcex.style         = CS_HREDRAW | CS_VREDRAW;
  wcex.lpfnWndProc   = (WNDPROC)WndProc;
  wcex.cbClsExtra    = 0;
  wcex.cbWndExtra    = 0;
  wcex.hInstance     = hInstance;
  wcex.hIcon         = LoadIcon(hInstance,IDI_APPLICATION);
  wcex.hCursor       = LoadCursor(NULL,IDC_ARROW);
  wcex.hbrBackground = (HBRUSH)(COLOR_BACKGROUND);
  wcex.lpszMenuName  = 0;
  wcex.lpszClassName = szWindowClass;
  wcex.hIconSm       = LoadIcon(wcex.hInstance,IDI_APPLICATION);
  return RegisterClassEx(&wcex);
}

HWND MostrarVentana (HINSTANCE hInstance)
{ const char *szTitle       = "Crack";
  const char *szWindowClass = "WinApp";
  HWND hWnd;
  hWnd = CreateWindow(szWindowClass,szTitle,WS_VISIBLE|WS_SYSMENU|WS_MINIMIZEBOX|
                      WS_MAXIMIZEBOX,CW_USEDEFAULT,CW_USEDEFAULT,316,192,NULL,
                      NULL,hInstance,NULL);
  return hWnd;
}
LRESULT CALLBACK WndProc (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{ const unsigned char MAXBYTES             = 3;
  const char *NombreArchivo                = "parchame.exe";
  const long TamanoArchivo                 = 40960;
  const int Desplazamiento[MAXBYTES]       = {0x438,0x46F,0x470};
  const unsigned char ByteActual[MAXBYTES] = {0x68,0x75,0x11};
  const unsigned char NuevoByte[MAXBYTES]  = {0xC3,0x40,0x48};
  HANDLE HandleArchivo;
  unsigned char K, ByteLeido;
  DWORD TamanoObtenido, temp;
  switch (message)
  { case WM_COMMAND:
      if (hbutton==(HWND)lParam)
      { HandleArchivo = CreateFile(NombreArchivo,GENERIC_READ|GENERIC_WRITE,
                                   FILE_SHARE_READ,NULL,OPEN_EXISTING,
                                   FILE_ATTRIBUTE_NORMAL,0);
        if (HandleArchivo==INVALID_HANDLE_VALUE)
        { MessageBox(hWnd,"Error:  No se pudo abrir el archivo",0,MB_OK);
          exit(1);
        }
        else
        { TamanoObtenido = SetFilePointer(HandleArchivo,0,NULL,FILE_END);
          if (TamanoObtenido!=TamanoArchivo)
          { CloseHandle(HandleArchivo);
            MessageBox(hWnd,"Error:  El archivo no es el apropiado",0,MB_OK);
            exit(2);
          }
          else
          { for (K=0;K<MAXBYTES;K++)
            { SetFilePointer(HandleArchivo,Desplazamiento[K],NULL,FILE_BEGIN);
              ReadFile(HandleArchivo,&ByteLeido,1,&temp,NULL);
              if (ByteLeido==ByteActual[K])
              { SetFilePointer(HandleArchivo,Desplazamiento[K],NULL,FILE_BEGIN);
```

```
              WriteFile(HandleArchivo,&NuevoByte[K],1,&temp,NULL);
            }
            else
            { CloseHandle(HandleArchivo);
              MessageBox(hWnd,"Error:  El archivo no coincide o ya fue crackeado",0,MB_OK);
              exit(3);
            }
          }
          CloseHandle(HandleArchivo);
          MessageBox(hWnd,"Operacion completada con exito","OK",MB_OK);
        }
      }
    }
    break;
  case WM_DESTROY:
    PostQuitMessage(0);
    break;
  default:
    return DefWindowProc(hWnd, message, wParam, lParam);
 }
 return 0;
}
```

Solo son algunas pequeñas diferencias comparado con el código necesario para Dev-C++. Excepto que Visual C++ genera un ejecutable sumamente reducido, en este caso solo superable por el lenguaje *assembler* o ensamblador.

Es recomendable revisar la documentación de Visual C++ y las siguientes API:

`RegisterClassEx`	:	Registra en memoria la estructura de una ventana nueva.
`CreateWindowEx`	:	Crea una ventana, un botón, texto, etc. standard o registrado.
`GetMessage`	:	Obtiene el primer mensaje que se encuentre en la cola de mensajes.
`TranslateMessage`	:	Traduce el mensaje hacia modo texto.
`DispatchMessage`	:	Despacha un mensaje hacia un procedimiento de ventana.
`CreateFile`	:	Crea o abre un archivo y devuelve su handle.
`SetFilePointer`	:	Posiciona el puntero de archivo en la ubicación especificada.
`CloseHandle`	:	Cierra un archivo a través de su handle.
`ReadFile`	:	Lee n bytes desde un archivo hacia una variable y avanza el puntero.
`WriteFile`	:	Escribe n bytes hacia un archivo desde una variable y avanza el puntero.
`PostQuitMessage`	:	Realiza un requerimiento de salida del programa.
`DefWindowProc`	:	Llama al procedimiento manejador de eventos de una ventana.

MASM32

Utilizar lenguaje ensamblador, es lo mejor en cuanto a tamaño y velocidad de código. El listado que mostramos a continuación, básicamente es el mismo código que para Dev-C++ y Visual C++ solo que aquí se emplean exclusivamente instrucciones Win32 y *assembler*.

El código para `patch.asm` es:

```
; Reemplace h:\masm32\ por una ruta adecuada
;   a su compilador MASM32

.386
.model flat, stdcall
option casemap :none

include h:\masm32\include\windows.inc
include h:\masm32\include\user32.inc
include h:\masm32\include\kernel32.inc

includelib h:\masm32\lib\user32.lib
includelib h:\masm32\lib\kernel32.lib
```

```
WinMain PROTO :DWORD,:DWORD,:DWORD,:DWORD
WndProc PROTO :DWORD,:DWORD,:DWORD,:DWORD

.const
  NombreClase    db "WinApp",0
  WinTitle       db "Crack",0
  ButTitle       db "&Crack",0
  Label1         db "STATIC",0
  Button1        db "BUTTON",0
  MAXBYTES       dd 3
  NombreArchivo  db "parchame.exe",0
  TamanoArchivo  dd 40960
  Desplazamiento dd 438h,46Fh,470h
  ByteActual     db 68h,75h,11h
  NuevoByte      db 0C3h,40h,48h
  Titulo         db "Crack para Parchame 1.0",0
  Autor          db "aDVANCED rESEARCH cOMMUNITY (c) 2003",0
  Error1         db "Error:  No se pudo abrir el archivo",0
  Error2         db "Error:  El archivo no es el apropiado",0
  Error3         db "Error:  El archivo no coincide o ya fue crackeado",0
  MsgOK          db "Operacion completada con exito",0
  TitOK          db "OK",0

.data?
  hbutton        dd ?
  HandleArchivo  dd ?
  TamanoObtenido dd ?
  ByteLeido      dd ?
  ByteAGrabar    dd ?
  temp           dd ?

.data
  K              dd 0

.code
start:
  INVOKE GetModuleHandle,NULL
  INVOKE WinMain,EAX,NULL,NULL,SW_SHOWDEFAULT
  INVOKE ExitProcess,EAX

  WinMain proc hInst:HINSTANCE, hPrevInst:HINSTANCE, CmdLine:LPSTR, CmdShow:DWORD
    LOCAL wc   : WNDCLASSEX
    LOCAL msg  : MSG
    LOCAL hwnd : HWND
    MOV  wc.cbSize,SIZEOF WNDCLASSEX
    MOV  wc.style,CS_HREDRAW or CS_VREDRAW
    MOV  wc.lpfnWndProc,OFFSET WndProc
    MOV  wc.cbClsExtra,NULL
    MOV  wc.cbWndExtra,NULL
    PUSH hInst
    POP  wc.hInstance
    INVOKE LoadIcon,NULL,IDI_APPLICATION
    MOV  wc.hIcon,EAX
    INVOKE LoadCursor,NULL,IDC_ARROW
    MOV  wc.hCursor,EAX
    MOV  wc.hbrBackground,COLOR_WINDOW
    MOV  wc.lpszMenuName,NULL
    MOV  wc.lpszClassName,OFFSET NombreClase
    MOV  wc.hIconSm,0
    INVOKE RegisterClassEx,ADDR wc
    INVOKE CreateWindowEx,NULL,ADDR NombreClase,ADDR WinTitle,WS_VISIBLE or   \
                          WS_SYSMENU or WS_MINIMIZEBOX or WS_MAXIMIZEBOX,     \
                          240,110,316,192,NULL,NULL,hInst,NULL
    MOV hwnd,EAX
    ; Clase STATIC que no necesita ser registrada (Titulo)
```

```
    INVOKE CreateWindowEx,NULL,ADDR Label1,ADDR Titulo,WS_CHILD or WS_VISIBLE \
                          or SS_CENTER,16,16,273,33,hwnd,NULL,hInst,NULL
    ; Clase STATIC que no necesita ser registrada (Autor)
    INVOKE CreateWindowEx,NULL,ADDR Label1,ADDR Autor,WS_CHILD or WS_VISIBLE  \
                          or SS_CENTER,16,56,273,33,hwnd,NULL,hInst,NULL
    ; Clase BUTTON que no necesita ser registrada
    INVOKE CreateWindowEx,NULL,ADDR Button1,ADDR ButTitle,WS_CHILD or         \
                          WS_VISIBLE or BS_PUSHBUTTON or BS_CENTER or         \
                          BS_VCENTER,115,104,75,25,hwnd,NULL,hInst,NULL
    MOV hbutton,EAX
    .WHILE TRUE
      INVOKE GetMessage, ADDR msg,NULL,0,0
      .BREAK .IF (!EAX)
      INVOKE TranslateMessage, ADDR msg
      INVOKE DispatchMessage, ADDR msg
    .ENDW
    MOV EAX,msg.wParam
    RET
  WinMain endp

  WndProc proc hWnd:HWND, uMsg:UINT, wParam:WPARAM, lParam:LPARAM
    .IF uMsg==WM_COMMAND
      MOV EAX,lParam
      .IF hbutton==EAX
        INVOKE CreateFile,ADDR NombreArchivo,GENERIC_READ or GENERIC_WRITE,   \
                          NULL,NULL,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL
        MOV HandleArchivo,EAX
        .IF HandleArchivo==INVALID_HANDLE_VALUE
          INVOKE MessageBox,hWnd,ADDR Error1,0,MB_OK
          INVOKE PostQuitMessage,1
        .ELSE
          INVOKE SetFilePointer,HandleArchivo,0,NULL,FILE_END
          .IF EAX!=TamanoArchivo
            INVOKE CloseHandle,HandleArchivo
            INVOKE MessageBox,hWnd,ADDR Error2,0,MB_OK
            INVOKE PostQuitMessage,2
          .ELSE
            MOV EDX,MAXBYTES
            .WHILE K<EDX
              MOV EDX,K
              MOV CL,2
              SHL EDX,CL
              INVOKE SetFilePointer,HandleArchivo,Desplazamiento[EDX],NULL,   \
                                    FILE_BEGIN
              INVOKE ReadFile,HandleArchivo,ADDR ByteLeido,1,ADDR temp,NULL
              MOV EDX,K
              XOR ECX,ECX
              MOV CL,BYTE PTR ByteActual[EDX]
              .IF ByteLeido==ECX
                MOV CL,2
                SHL EDX,CL
                INVOKE SetFilePointer,HandleArchivo,Desplazamiento[EDX],NULL, \
                                      FILE_BEGIN
                MOV EDX,K
                XOR ECX,ECX
                MOV CL,BYTE PTR NuevoByte[EDX]
                MOV ByteAGrabar,ECX
                INVOKE WriteFile,HandleArchivo,ADDR ByteAGrabar,1,ADDR temp,  \
                                 NULL
              .ELSE
                INVOKE CloseHandle,HandleArchivo
                INVOKE MessageBox,hWnd,ADDR Error3,0,MB_OK
                INVOKE PostQuitMessage,3
              .ENDIF
              INC K
```

```
              MOV EDX,MAXBYTES
             .ENDW
             INVOKE CloseHandle,HandleArchivo
             INVOKE MessageBox,hWnd,ADDR MsgOK,ADDR TitOK,MB_OK
             INVOKE PostQuitMessage,NULL
           .ENDIF
         .ENDIF
       .ENDIF
     .ELSEIF uMsg==WM_DESTROY
       INVOKE PostQuitMessage,NULL
     .ELSE
       INVOKE DefWindowProc,hWnd,uMsg,wParam,lParam
       RET
     .ENDIF
    RET
  WndProc endp

END start
; Obs.:   - El arreglo Desplazamiento dd 438h,46Fh,470h es una variable de
;           12 bytes (dd=Define Double Word).  Por lo tanto, para acceder a cada
;           uno de sus elementos, recuerde que esto es assembler y trabajamos con
;           desplazamientos de memoria y nunca indices estilo VB, Delphi o C++ asi:
;
;           Desplazamiento[0] = 438h cuando K vale 0
;           Desplazamiento[4] = 46Fh cuando K vale 1
;           Desplazamiento[8] = 470h cuando K vale 2
;
;         - No obstante, la variable K avanza de uno en uno.  El mejor metodo para
;           multiplicarla por 4 y obtener el desplazamiento correcto es:
;
;           MOV EDX,K     ; Copiar K en un registro
;           MOV CL,2      ; 1=x2; 2=x4; 3=x8; 4=x16; ...
;           SHL EDX,CL    ; EDX = EDX * (2^CL)
;
;         - Si no le agrada este metodo, puede optar por el clasico MUL
```

Como hemos podido observar, realmente no se diferencia mucho del código en Dev-C++ ni del código en Visual C++. Solo fueron necesarios algunos ajustes relacionados con los caprichos del propio MASM32. Quienes prefieran TASM, deberán cambiar las llamadas **INVOKE** por un **CALL** y pasar sus parámetros con **PUSH** en orden inverso, además de otras modificaciones menores.

Por todos nosotros es conocido el hecho que, al programar en *assembler* o ensamblador propio de MASM32, deben respetarse ciertas restricciones básicas tales como:

- ✓ Nunca asumir valores por defecto en un registro.
- ✓ No se pueden realizar movimientos de memoria hacia memoria.
- ✓ Se debe distinguir perfectamente entre un **ADDR** y un **OFFSET**.
- ✓ Determinar correctamente el tamaño de una variable al declararla.
- ✓ Si se utilizan valores constantes definirlos en una sección **.CONST**. Esto protegerá aquellas direcciones de memoria contra posibles ediciones.
- ✓ Si se declaran variables sin inicializar, hacerlo en la sección **.DATA?**
- ✓ Indentar con claridad el código relacionado a las condicionales, bucles, procedimientos y macros.

Revisar detalladamente la documentación de las API Win32

GetModuleHandle	:	Obtiene el identificador de la instancia de la aplicación.
RegisterClassEx	:	Registra en memoria la estructura de una ventana nueva.
CreateWindowEx	:	Crea una ventana, un botón, texto, etc. standard o registrado.
GetMessage	:	Obtiene el primer mensaje que se encuentre en la cola de mensajes.
TranslateMessage	:	Traduce el mensaje hacia modo texto.
DispatchMessage	:	Despacha un mensaje hacia un procedimiento de ventana.
CreateFile	:	Crea o abre un archivo y devuelve su handle.
SetFilePointer	:	Posiciona el puntero de archivo en la ubicación especificada.

CloseHandle	:	Cierra un archivo a través de su handle.
ReadFile	:	Lee n bytes desde un archivo hacia una variable y avanza el puntero.
WriteFile	:	Escribe n bytes hacia un archivo desde una variable y avanza el puntero.
PostQuitMessage	:	Realiza un requerimiento de salida del programa.
DefWindowProc	:	Llama al procedimiento manejador de eventos de una ventana.
ExitProcess	:	Cierra la instancia de un programa devolviendo un valor.

GoAsm

Para cerrar con broche, no de oro sino de titanio, tenemos a nuestro favorito de los ensambladores. GoAsm contiene lo mejor de MASM32 unido a la facilidad de NASM y la velocidad de TASM. GoAsm nos evita los problemas de siempre:

- ¿Por qué es **ADDR** y no es **OFFSET**?
- ¿Cuándo es **EAX** y cuando es **[EAX]**?
- ¿Por qué a veces no puedo pasar cómo parámetro una cadena?
- **Stack**, **FLAT**, **casemap**, **stdcall**, **include**, etc.

El código de **PatchGo.asm** es el siguiente:

```
;-------------------------------------------------------------------------------
; Estructura para la clase ventana
WNDCLASSEX STRUCT
  cbSize             DD ?
  style              DD ?
  lpfnWndProc        DD ?
  cbClsExtra         DD ?
  cbWndExtra         DD ?
  hInstance          DD ?
  hIcon              DD ?
  hCursor            DD ?
  hbrBackground      DD ?
  lpszMenuName       DD ?
  lpszClassName      DD ?
  hIconSm            DD ?
ENDS

; Constantes para la ventana
WS_MAXIMIZEBOX           =     10000h
WS_MINIMIZEBOX           =     20000h
WS_SYSMENU               =     80000h
WS_VISIBLE               =  10000000h

; Constantes para los botones y etiquetas
SS_CENTER                =         1h
BS_CENTER                =       300h
BS_VCENTER               =      0C00h
WS_CHILD                 =  40000000h

; Mensajes a procesar
WM_DESTROY               =        2h
WM_COMMAND               =      111h

; Color de fondo para la ventana
COLOR_WINDOW             =        5h

; Estilos de clases para la ventana
CS_VREDRAW               =        1h
CS_HREDRAW               =        2h

; Varios
INVALID_HANDLE_VALUE     =       -1h
```

```
FILE_BEGIN              =        0h
FILE_END                =        2h
FILE_SHARE_READ         =        1h
FILE_SHARE_WRITE        =        2h
OPEN_EXISTING           =        3h
FILE_ATTRIBUTE_NORMAL   =       80h
IDI_APPLICATION         =    32512d
IDC_ARROW               =    32512d
GENERIC_READ            = 80000000h
GENERIC_WRITE           = 40000000h


;-------------------------------------------------------------------------------
CONST SECTION
  MAXBYTES       DD 3
  NombreArchivo  DB "parchame.exe",0
  TamanoArchivo  DD 40960
  Desplazamiento DD 438h,46Fh,470h
  ByteActual     DB 68h,75h,11h
  NuevoByte      DB 0C3h,40h,48h
  Titulo         DB "Crack para Parchame 1.0",0
  Autor          DB "aDVANCED rESEARCH cOMMUNITY (c) 2003",0
  MESSAGES       DD WM_DESTROY, Destroy          ; Mensajes windows y sus
                 DD WM_COMMAND, Click            ; direcciones de codigo
  NombreClase    DB "WinApp",0


;-------------------------------------------------------------------------------
DATA SECTION
  K              DD 0
  MainMsg        DD 7 DUP 0                      ; Estructura con los mensajes
                                                 ; hWnd,
                                                 ; +4=message,
                                                 ; +8=wParam,
                                                 ; +C=lParam,
                                                 ; +10h=time,
                                                 ; +14h/18=pt
  hbutton        DD ?
  HandleArchivo  DD ?
  ByteLeido      DD ?
  ByteAGrabar    DD ?
  temp           DD ?
  HandleApp      DD ?                            ; Handle de la aplicacion
  HandleWin      DD ?                            ; Handle de la ventana
  wc             WNDCLASSEX


;-------------------------------------------------------------------------------
CODE SECTION

  ; Responde al mensaje WM_DESTROY
  ; ------------------------------
  Destroy:
    INVOKE PostQuitMessage,0                     ; Requerimiento de terminacion
    STC                                          ; Procesar DefWindowProc
    RET


  ; FUNCION MANEJADORA DE EVENTOS:
  ; ------------------------------
  WndProc FRAME hwnd,uMsg,wParam,lParam          ; Crear pila y obtener parametros
    MOV EAX,[uMsg]                               ; Mover mensaje windows hacia EAX
    MOV ECX,SIZEOF MESSAGES/8                    ; Obtener cantidad de mensajes
    MOV EDX,ADDR MESSAGES                        ; Direccion del mensaje
    FOUND:
```

```
    DEC ECX                                 ; Si es negativo
    JS >.notfound                           ;   entonces ir a .notfound
    CMP [EDX+ECX*8],EAX                     ;   Verificar mensaje correcto
    JNZ FOUND                               ;   Si no, siguiente mensaje
    CALL [EDX+ECX*8+4]                      ; Procesar el mensaje
    JNC >.exit
  .notfound
    INVOKE DefWindowProcA,[hwnd],[uMsg],[wParam],[lParam]
  .exit
    RET
  ENDF


; Responde al mensaje WM_COMMAND
; ------------------------------
Click:
  USEDATA WndProc                           ; Utilizar parametros de WndProc
    USES EBX,EDI,ESI
    MOV EAX,[lParam]
    CMP [hbutton],EAX
    JNE >QuitClick
    .OK1
      INVOKE CreateFileA,ADDR NombreArchivo,GENERIC_READ ¦ GENERIC_WRITE,0,0,   \
                         OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,0
      MOV [HandleArchivo],EAX
      CMP D[HandleArchivo],INVALID_HANDLE_VALUE
      JE >ShowError1
    .OK2
      INVOKE SetFilePointer,[HandleArchivo],0,0,FILE_END
      CMP EAX,[TamanoArchivo]
      JNE >ShowError2
    .OK3
      MOV EDX,[MAXBYTES]
      .Repetir
        CMP D[K],EDX
        JGE >>.FinRepetir
        MOV EDX,[K]
        MOV CL,2
        SHL EDX,CL
        INVOKE SetFilePointer,[HandleArchivo],[Desplazamiento+EDX],0,FILE_BEGIN
        INVOKE ReadFile,[HandleArchivo],ADDR ByteLeido,1,ADDR temp,0
        MOV EDX,[K]
        XOR ECX,ECX
        MOV CL,[ByteActual+EDX]
        CMP [ByteLeido],ECX
        JNE >ShowError3
        .OK4
          MOV CL,2
          SHL EDX,CL
          INVOKE SetFilePointer,[HandleArchivo],[Desplazamiento+EDX],0,FILE_BEGIN
          MOV EDX,[K]
          XOR ECX,ECX
          MOV CL,[NuevoByte+EDX]
          MOV [ByteAGrabar],ECX
          INVOKE WriteFile,[HandleArchivo],ADDR ByteAGrabar,1,ADDR temp,0
          INC D[K]
          MOV EDX,[MAXBYTES]
          JMP <.Repetir
      .FinRepetir
        INVOKE CloseHandle,[HandleArchivo]
        INVOKE MessageBoxA,[HandleWin],"Operacion completada con exito","OK",0
        INVOKE PostQuitMessage,0
        JMP >QuitClick
    ShowError1:
      INVOKE MessageBoxA,[HandleWin],"Error:  No se pudo abrir el archivo",0,0
```

```
      INVOKE PostQuitMessage,1
      JMP >QuitClick
    ShowError2:
      INVOKE CloseHandle,[HandleArchivo]
      INVOKE MessageBoxA,[HandleWin],"Error:  El archivo no es el apropiado",0,0
      INVOKE PostQuitMessage,2
      JMP >QuitClick
    ShowError3:
      INVOKE CloseHandle,[HandleArchivo]
      INVOKE MessageBoxA,[HandleWin],"Error:  El archivo no coincide o ya fue crackeado",0,0
      INVOKE PostQuitMessage,3
    QuitClick:
    RET
  ENDU


; PROGRAMA PRINCIPAL:
; -------------------
START:
  INVOKE GetModuleHandleA,0
  MOV [HandleApp], EAX

  MOV D[wc.cbSize],SIZEOF WNDCLASSEX
  MOV D[wc.style],CS_HREDRAW | CS_VREDRAW
  MOV [wc.lpfnWndProc],ADDR WndProc
  MOV D[wc.cbClsExtra],0
  MOV D[wc.cbWndExtra],0
  PUSH [HandleApp]
  POP [wc.hInstance]
  INVOKE LoadIconA,[HandleApp],IDI_APPLICATION
  MOV [wc.hIcon],EAX
  INVOKE LoadCursorA,0,IDC_ARROW
  MOV [wc.hCursor],EAX
  MOV D[wc.hbrBackground],COLOR_WINDOW
  MOV D[wc.lpszMenuName],0
  MOV [wc.lpszClassName],ADDR NombreClase
  MOV D[wc.hIconSm],0

  INVOKE RegisterClassExA, ADDR wc
  INVOKE CreateWindowExA,0,ADDR NombreClase,"Crack",WS_MAXIMIZEBOX |            \
                         WS_MINIMIZEBOX | WS_SYSMENU | WS_VISIBLE,240,110,     \
                         316,192,0,0,[HandleApp],0
  MOV [HandleWin],EAX
  ; Clase STATIC que no necesita ser registrada (Titulo)
  INVOKE CreateWindowExA,0,"STATIC",ADDR Titulo,WS_CHILD | WS_VISIBLE |        \
                         SS_CENTER,16,16,273,33,[HandleWin],0,[HandleApp],0
  ; Clase STATIC que no necesita ser registrada (Autor)
  INVOKE CreateWindowExA,0,"STATIC",ADDR Autor,WS_CHILD | WS_VISIBLE |         \
                         SS_CENTER,16,56,273,33,[HandleWin],0,[HandleApp],0
  ; Clase BUTTON que no necesita ser registrada
  INVOKE CreateWindowExA,0,"BUTTON","&Crack",WS_CHILD | WS_VISIBLE | BS_CENTER |\
                         BS_VCENTER,115,104,75,25,[HandleWin],0,[HandleApp],0
  MOV [hbutton],EAX

  .bucle
    INVOKE GetMessageA, ADDR MainMsg,0,0,0    ; Obtener puntero del mensaje
    OR EAX,EAX                                ; mas eficiente que CMP EAX,0
    JZ >.quit                                 ;   entonces .QUIT
    INVOKE TranslateMessage, ADDR MainMsg     ; Convertir mensaje a cadena
    INVOKE DispatchMessageA, ADDR MainMsg     ; Entregar mensaje al manejador
   JMP <.bucle
  .quit
    INVOKE UnregisterClassA,ADDR NombreClase,[HandleApp]
    INVOKE ExitProcess, [MainMsg+8h]          ; Finalizar devolviendo wParam
; Obs.:   - Para poder compilar este programa utilice los siguientes comandos:
```

```
;
;           c:\goasm patchgo.asm
;           c:\golink patchgo.obj gdi32.dll kernel32.dll user32.dll
;
;         - El primer comando genera el archivo patchgo.obj, mientras que el
;           segundo comando crea el ejecutable.
;         - El directorio de GoAsm debe estar en la ruta
```

En este caso específico y frente a un código de por sí reducido, nos encontramos con un curioso hecho. Los ejecutables producto de MASM32 y GoAsm tienen exactamente el mismo tamaño. Aunque si los analizamos, su código lleva diferencias destacables.

Como es acostumbrado, recomendamos leer la documentación de:

`GetModuleHandle`	:	Obtiene el identificador de la instancia de la aplicación.
`RegisterClassExA`	:	Registra en memoria la estructura de una ventana nueva.
`CreateWindowExA`	:	Crea una ventana, un botón, texto, etc. standard o registrado.
`GetMessage`	:	Obtiene el primer mensaje que se encuentre en la cola de mensajes.
`CreateFileA`	:	Crea o abre un archivo y devuelve su handle.
`SetFilePointer`	:	Posiciona el puntero de archivo en la ubicación especificada.
`CloseHandle`	:	Cierra un archivo a través de su handle.
`ReadFile`	:	Lee n bytes desde un archivo hacia una variable y avanza el puntero.
`WriteFile`	:	Escribe n bytes hacia un archivo desde una variable y avanza el puntero.
`PostQuitMessage`	:	Realiza un requerimiento de salida del programa.
`DefWindowProc`	:	Llama al procedimiento manejador de eventos de una ventana.
`TranslateMessage`	:	Traduce el mensaje hacia modo texto.
`DispatchMessage`	:	Despacha un mensaje hacia un procedimiento de ventana.
`UnregisterClassA`	:	Elimina una clase registrada con `RegisterClassExA`.
`ExitProcess`	:	Cierra la instancia de un programa devolviendo un valor.

Optimización

Mejorar un código fuente, ya sea para hacerlo más rápido o tal vez para reducir su tamaño, es una tarea delicada. Algunas pautas que podríamos aplicar para cualquiera de los lenguajes expuestos:

- El código carece de una verificación de la correcta escritura de los bytes en el archivo a crackear.
- Solo se ha considerado la modificación de 1 archivo por vez.
- Los mensajes explicativos son escasos.
- El fondo es tan importante como la forma. Estos códigos fuente requieren algo de "maquillaje".

Delphi sin VCL

Aquí un obsequio para los Delphimaníacos que quieren un patcher sin utilizar VCL. Cree un archivo de texto plano con el Notepad, aunque les recomiendo utilizar Dana 1.14 descargable desde http://www.rimarts.co.jp y cuyo crack lo pueden encontrar en http://fl.web1000.com, bajo el nombre de aRC-Crack Engine). El archivo a crear debe tener extension .DPR.

El contenido del programa `patcher.dpr` sería el siguiente:

```
// Compile con:  c:\>dcc32 patcher.dpr
program
  patcher;

uses
  windows, messages;

var
  wc                : TWndClassEx;
  msg               : TMsg;
  HandleWin, hbutton : HWND;
```

```
procedure crackear;
  const
    MAXBYTES                = 3;
    NombreArchivo  : String  = 'parchame.exe';
    TamanoArchivo  : Longint = 40960;
    Desplazamiento : Array [1..MAXBYTES] of Integer = ($438,$46F,$470);
    ByteActual     : Array [1..MAXBYTES] of Byte    = ($68,$75,$11);
    NuevoByte      : Array [1..MAXBYTES] of Byte    = ($C3,$40,$48);
  var
    HandleArchivo  : File of Byte;
    K, ByteLeido   : Byte;
    TamanoObtenido : Longint;
  begin
    {$I-}                                          // Desactivar control de errores I/O
    AssignFile(HandleArchivo,NombreArchivo);
    Reset(HandleArchivo);
    if (IOResult<>0) then begin
      MessageBox(0,'Error:  No se pudo abrir el archivo','Error',0);
      Halt(1);
    end
    {$I+}                                          // Reactivar control de errores I/O
    else begin
      TamanoObtenido := FileSize(HandleArchivo);
      if (TamanoObtenido<>TamanoArchivo) then begin
        CloseFile(HandleArchivo);
        MessageBox(0,'Error:  El archivo no es el apropiado','Error',0);
        Halt(2);
      end
      else begin
        For K:=1 to MAXBYTES do begin
          Seek(HandleArchivo,Desplazamiento[K]);
          Read(HandleArchivo,ByteLeido);
          if (ByteLeido=ByteActual[K]) then begin
            Seek(HandleArchivo,Desplazamiento[K]);
            Write(HandleArchivo,NuevoByte[K]);
          end
          else begin
            CloseFile(HandleArchivo);
            MessageBox(0,'Error:  El archivo no coincide o ya fue crackeado','Error',0);
            Halt(3);
          end;
        end;
        CloseFile(HandleArchivo);
        MessageBox(0,'Operacion completada con exito','OK',0);
        PostQuitMessage(0);
      end;
    end;
  end;

function WndProc(hWnd: HWND; uMsg: UINT; wParam: wParam; lParam: LParam): lresult;
                stdcall;
  begin
    Result := 0;
    case uMsg of
      WM_DESTROY:
        PostQuitMessage(0);
      WM_COMMAND:
        if lParam=hbutton then crackear;
      else
        Result := DefWindowProc(hWnd, uMsg, wParam, lParam);
    end;
  end;

// Programa principal
begin
```

```
  wc.cbSize        := SizeOf(TWndClassEx);
  wc.style         := CS_HREDRAW or CS_VREDRAW;
  wc.lpfnWndProc   := @WndProc;
  wc.cbClsExtra    := 0;
  wc.cbWndExtra    := 0;
  wc.hInstance     := hInstance;
  wc.hIcon         := LoadIconA(hInstance,IDI_APPLICATION);
  wc.hCursor       := LoadCursorA(0,IDC_ARROW);
  wc.hbrBackground := COLOR_WINDOW;
  wc.lpszMenuName  := NIL;
  wc.lpszClassName := 'WinApp';
  wc.hIconSm       := 0;
  RegisterClassExA(wc);
  HandleWin := CreateWindowExA(0,'WinApp','Crack',WS_MAXIMIZEBOX or
                               WS_MINIMIZEBOX or WS_SYSMENU or WS_VISIBLE,
                               240,110,316,192,0,0,hInstance,NIL);
  // Clase STATIC que no necesita ser registrada (Titulo)
  CreateWindowExA(0,'STATIC','Crack para Parchame 1.0',WS_CHILD or WS_VISIBLE
                  or SS_CENTER,16,16,273,33,HandleWin,0,hInstance,NIL);
  // Clase STATIC que no necesita ser registrada (Autor)
  CreateWindowExA(0,'STATIC','aDVANCED rESEARCH cOMMUNITY (c) 2003',WS_CHILD or
                  WS_VISIBLE or SS_CENTER,16,56,273,33,HandleWin,0,hInstance,NIL);
  // Clase BUTTON que no necesita ser registrada
  hbutton := CreateWindowExA(0,'BUTTON','&Crack',WS_CHILD or WS_VISIBLE or
                             BS_CENTER or BS_VCENTER,115,104,75,25,HandleWin,
                             0,hInstance,NIL);
  While GetMessage(msg,0,0,0) do begin
    TranslateMessage(msg);
    DispatchMessage(msg);
  end;
  ExitCode := msg.wParam;
end.
```

Esperamos que sea de utilidad para nuestros amigos delphimaníacos, recordándoles que revisen la documentación de las funciones y constantes de la API del winbugs.

C++Builder sin VCL

No podíamos dejar de lado a los programadores de C++Builder sin VCL, así que estamos seguros apreciarán este modesto código que presentamos. Cree un archivo de texto plano y guardelo con extensión .BPR. El archivo **patcher.bpr** quedaría así:

```
// Compile con:  c:\>bcc32 -tW patcher.bpr
#include "windows.h"

#define MAXBYTES 3
HWND hbutton;

void crackear(void)
{ const char *NombreArchivo          = "parchame.exe";
  const unsigned long TamanoArchivo  = 40960;
  const int Desplazamiento[MAXBYTES] = {0x438,0x46F,0x470};
  const byte ByteActual[MAXBYTES]    = {0x68,0x75,0x11};
  const byte NuevoByte[MAXBYTES]     = {0xC3,0x40,0x48};
  HANDLE HandleArchivo;
  byte K, ByteLeido;
  unsigned long TamanoObtenido, temp;
  HandleArchivo = CreateFile(NombreArchivo,GENERIC_READ | GENERIC_WRITE,
                             FILE_SHARE_READ,NULL,OPEN_EXISTING,
                             FILE_ATTRIBUTE_NORMAL,NULL);
  if (HandleArchivo==INVALID_HANDLE_VALUE)
  { MessageBoxA(0,"Error:  No se pudo abrir el archivo",0,MB_OK);
    exit(1);
  }
```

```
  else
  { TamanoObtenido = SetFilePointer(HandleArchivo,0,NULL,FILE_END);
    if (TamanoObtenido!=TamanoArchivo)
    { CloseHandle(HandleArchivo);
      MessageBoxA(0,"Error:  El archivo no es el apropiado",0,MB_OK);
      exit(2);
    }
    else
    { for(K=0;K<MAXBYTES;K++)
      { SetFilePointer(HandleArchivo,Desplazamiento[K],NULL,FILE_BEGIN);
        ReadFile(HandleArchivo,&ByteLeido,1,&temp,NULL);
        if (ByteLeido==ByteActual[K])
        { SetFilePointer(HandleArchivo,Desplazamiento[K],NULL,FILE_BEGIN);
          WriteFile(HandleArchivo,&NuevoByte[K],1,&temp,NULL);
        }
        else
        { CloseHandle(HandleArchivo);
          MessageBoxA(0,"Error:  El archivo no coincide o ya fue crackeado",0,MB_OK);
          exit(3);
        }
      }
      CloseHandle(HandleArchivo);
      MessageBoxA(0,"Operacion completada con exito","OK",MB_OK);
      PostQuitMessage(0);
    }
  }
}

LRESULT CALLBACK WndProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{ LRESULT Result = 0;
  switch (uMsg)
  { case WM_DESTROY:
      PostQuitMessage(0);
      break;
    case WM_COMMAND:
      if (hbutton==(HWND)lParam)
        crackear();
      break;
    default:
      Result = DefWindowProc(hwnd, uMsg, wParam, lParam);
  }
  return Result;
}

// Programa principal
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
                   LPSTR lpszCmdParam, int nCmdShow)
{ WNDCLASSEX wc;
  MSG msg;
  HWND HandleWin;
  wc.cbSize        = sizeof(WNDCLASSEX);
  wc.style         = CS_HREDRAW | CS_VREDRAW;
  wc.lpfnWndProc   = (WNDPROC)WndProc;
  wc.cbClsExtra    = 0;
  wc.cbWndExtra    = 0;
  wc.hInstance     = hInstance;
  wc.hIcon         = LoadIconA(0,IDI_APPLICATION);
  wc.hCursor       = LoadCursorA(0,IDC_ARROW);
  wc.hbrBackground = (HBRUSH)COLOR_WINDOW;
  wc.lpszMenuName  = 0;
  wc.lpszClassName = "WinApp";
  wc.hIconSm       = 0;
  RegisterClassExA(&wc);
  HandleWin = CreateWindowExA(0,"WinApp","Crack",WS_MAXIMIZEBOX |
                              WS_MINIMIZEBOX | WS_SYSMENU | WS_VISIBLE,
```

```
                                240,110,316,192,0,0,hInstance,0);
  // Clase STATIC que no necesita ser registrada (Titulo)
  CreateWindowExA(0,"STATIC","Crack para Parchame 1.0",WS_CHILD | WS_VISIBLE
                  | SS_CENTER,16,16,273,33,HandleWin,0,hInstance,0);
  // Clase STATIC que no necesita ser registrada (Autor)
  CreateWindowExA(0,"STATIC","aDVANCED rESEARCH cOMMUNITY (c) 2003",WS_CHILD |
                 WS_VISIBLE | SS_CENTER,16,56,273,33,HandleWin,0,hInstance,0);
  // Clase BUTTON que no necesita ser registrada
  hbutton = CreateWindowExA(0,"BUTTON","&Crack",WS_CHILD | WS_VISIBLE |
                            BS_CENTER | BS_VCENTER,115,104,75,25,HandleWin,
                            0,hInstance,0);
  while (TRUE==GetMessage(&msg,0,0,0))
  { TranslateMessage(&msg);
    DispatchMessage(&msg);
  }
  return msg.wParam;
}
```

En realidad no es el código que esperábamos realizar. Lamentablemente nos vimos forzados a reemplazar las funciones de manejo de archivos de C++Builder por sus equivalentes API. Apreciaríamos muy de veras, nos hagan llegar sus sugerencias acerca del include apropiado para poder emplearlas. Las funciones de archivos para las que necesitamos el include apropiado son:

- FileOpen
- FileClose
- FileRead
- FileWrite
- FileSeek

Apéndice

Para aquellos interesados en el código fuente de nuestro archivo **parchame.exe**, se adjunta el archivo **parchame.zip** con todo lo necesario (38,798 Bytes). Descomprimirlo obligatoriamente en **c:\test** y compilar en una ventana DOS con:

```
c:\test>gorc rsrc.rc
c:\test>goasm parchame.asm
c:\test>golink parchame.obj rsrc.obj kernel32.dll user32.dll gdi32.dll lz32.dll winmm.dll
```

parchame.exe hace uso de recursos que han sido comprimidos, luego incluidos en un archivo RC tradicional. Estos recursos son cargados luego de ser descomprimidos en tiempo real. Hemos incluido una simple animación vía un temporizador o *Timer* no muy eficiente y también un efecto de sonido para la carga. Estudien los métodos empleados cuyos comentarios se han colocado apropiadamente.

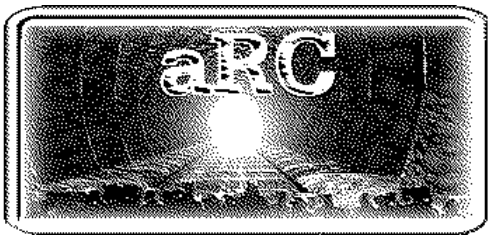
Derechos de autor

Las preguntas, consultas, sugerencias y correcciones son todas bienvenidas aunque las respuestas puedan tardar unos días en llegarles.

El autor puede ser contactado en:

IRC CyberArmy /server -m irc.cyberarmy.com: #arc3000, #arc
IRC DalNet: #arc3000
Email: furiouslogic@eml.cc

Furious Logic
Cofounder of



aDVANCED rESEARCH cOMMUNITY
http://arc3000.web1000.com
Lista: http://beam.to/arc3000
IRC CyberArmy: #arc3000, #arc
IRC DalNET: #arc3000

"Porque buscamos la libertad que sólo en el conocimiento podemos encontrar"