

Servidor BF4

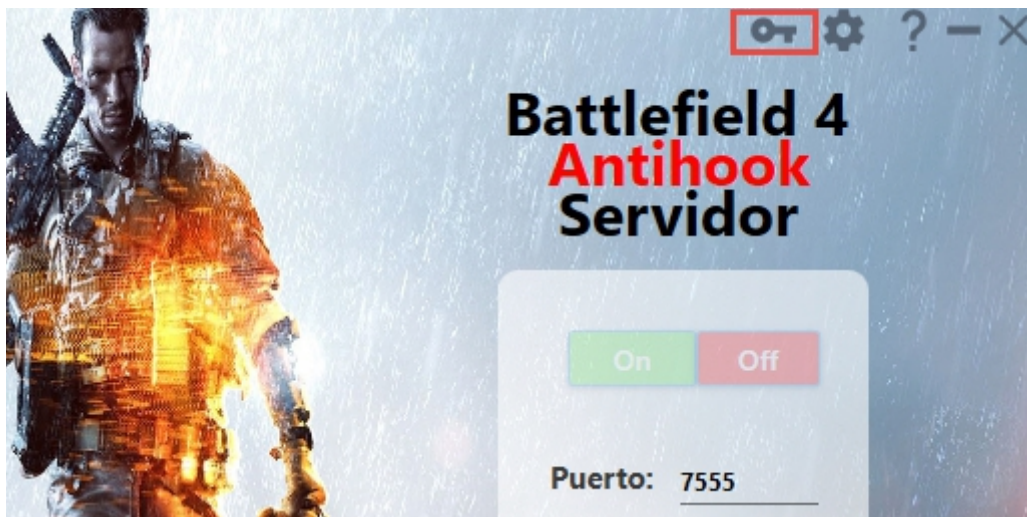
--..sequeyo..--



Hola saludos de nuevo. Hace ya mucho que no escribo nada aunque sigo al tanto de lo que se menea en el grupo CracksLatinos. Algunos ni me conocerán y otros muchos quizás me recuerden porque me centré bastante en la plataforma .NET y sobre la que escribí unos cuantos tutoriales.

Fué a raíz de un tuto escrito por Tincopasan sobre un programa en NET, lo que me animó a despolvar apuntes antiguos, herramientas de cracking y algunas neuronas. Por cierto: me estoy poniendo a escribir el tuto y caigo en la cuenta que creo que Tincopasan y yo "crecimos" juntos aquí en el grupo CraksLatinos y ambos tirábamos con el mismo lenguaje de programación que fué el Visual Basic. Y aquí seguimos desde entonces.

Bueno, pues centrándonos en el objetivo de éste tuto, el programa que estudió Tincopasan necesita ser desbloqueado mediante una clave. Si se fijan en la imagen de la cabecera, en el momento de abrir el programa vemos un par de botones que están deshabilitados. El boton verde "On" no estará accesible hasta que activemos el programa escribiendo una clave correcta. El programa tiene en su parte superior derecha una serie de iconos y el que nos interesa es el que tiene forma de llave. Al pulsarlo nos lleva al formulario de registro:



Registrar Antihook Server

Número de serie:

E435-05A2-BBEB-19CF-CB5E-A430-0A84-5964

Copiar

Licencia:

#####

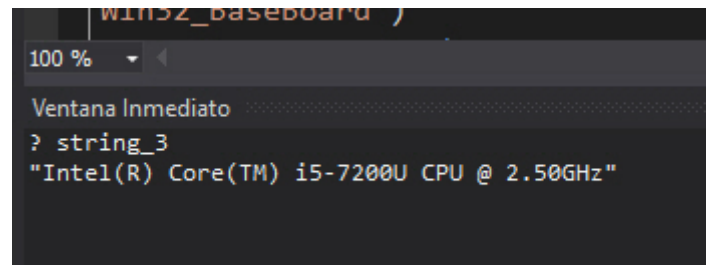
Pegar

Registrar

El objetivo de éste formulario es solicitar al usuario una licencia para desbloquear el programa. Dicho formulario nos aporta un número de serie que es único y varía en cada máquina. Quiere decir éso que sería inútil compartir con el resto del mundo una licencia porque simplemente no serviría. El método para conseguir un ID ó número de serie único e irrepetible para cada PC varía puede variar según lo considere oportuno el programador ó creador de la aplicación pero imagino que la gran mayoría utilizarán información sacada de nuestros propios ordenadores. ¿De dónde?, pues básicamente información de la placa base, del procesador, bios, disco duro, memoria ram... etc,etc. Con tan solo sacar info del procesador, ya tenemos algo que es único y que nos va a diferenciar del resto de máquinas. Si a eso le sumamos que podemos combinar la info extraída del procesador con la obtenida de

la placa base por ejemplo, ya nos estamos diferenciando todavía más del resto. Por ejemplo, mirando en mi Visual Studio y sacando info de mi máquina vemos lo siguiente con el procesador:

```
Using managementObjectSearcher As ManagementObjectSearcher = New ManagementObjectSearcher("Win32_Processor")
For Each current As ManagementBaseObject In managementObjectSearcher.[Get]()
    text = smethod_3(current("Manufacturer").ToString())
    text2 = smethod_3(current("Name").ToString())
    text3 = smethod_3(current("MaxClockSpeed").ToString())
    text4 = smethod_3(current("ProcessorID").ToString())
Try
```



The screenshot shows the 'Ventana Inmediato' (Immediate Window) in Visual Studio. It displays the value of a variable named 'string_3' as 'Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz'. The window title is 'Win32_BaseBoard'.

Ahora yo con el resultado de ésta consulta, puedo hacer lo que quiera porque al fin y al cabo es una cadena de texto lo que obtengo y puedo jugar con ella aplicando los algoritmos que se nos ocurran. Para saber todo lo que se puede hacer y toda la info que se puede sacar, deberían echar un vistazo a la documentación existente en el link:

<https://docs.microsoft.com/es-es/windows/desktop/CIMWin32Prov/computer-system-hardware-classes>

Ahí encontrarán info sobre cómo trabajar con el WMI(Windows Management Instrumentation).

Explico ésto porque la aplicación que nos ocupa trabaja así: saca info de nuestro PC y dicha info la somete a ciertos algoritmos para conformar su número de serie. Como Tincopasan, utilizo el dnSpy que me parece el mejor debugger para Net's que alguien haya creado. Desde que apareció no hago más que sorprenderme con él cada vez que lo utilizo. Lo que hice fué copiar y pegar en Visual Studio el código obtenido con dnSpy incluso manteniendo el nombre de las funciones :-D y les explico cómo funciona.

Lo primero que se hace es dimensionar ocho variables como strings:

```
Dim text As String = ""
Dim text2 As String = ""
Dim text3 As String = ""
Dim text4 As String = ""
Dim text5 As String = ""
Dim text6 As String = ""
Dim text7 As String = ""
Dim text8 As String = ""
```

Y ahora es donde entra ya en juego la recopilación de datos y las consultas mediante el WMI. Primero se pide info del procesador:

```
Using managementObjectSearcher As ManagementObjectSearcher = New ManagementObjectSearcher
("Select * from Win32_Processor")
For Each current As ManagementBaseObject In managementObjectSearcher.[Get]()
    text = smethod_3(current("Manufacturer").ToString())
    text2 = smethod_3(current("Name").ToString())
    text3 = smethod_3(current("MaxClockSpeed").ToString())
    text4 = smethod_3(current("ProcessorID").ToString())
    Try
        text5 = smethod_3(current("Revision").ToString())
    Catch ex_D3 As Exception
        text5 = smethod_3("error")
    End Try
Next
End Using
```

Se hacen consultas sobre el fabricante, el nombre del procesador, etc. Se utiliza también un bloque Try...Catch para manejar una excepción con la que el programador ya contaba. De hecho, en mi máquina también se genera la excepción porque por alguna razón, no se puede hacer la consulta "Revision" del procesador y gracias al manejo de la excepción se utiliza simplemente la palabra "error". Por cada consulta, se entra como ven en la imagen, en la función smethod_3:

```
Public Function smethod_3(string_3 As String) As String
    Dim arg_12_0 As HashAlgorithm = New MD5CryptoServiceProvider()
    Dim bytes As Byte() = Encoding.ASCII.GetBytes(string_3)
    Return smethod_4(arg_12_0.ComputeHash(bytes))
End Function
```

Bien, a ésta función se entra recurrentemente por cada consulta. Es de fácil comprensión lo que ocurre aquí. Se prepara el entorno para utilizar el hash MD5. La cadena con la que se entra en ésta función es convertida a un array de bytes. Se computa el hash MD5 de ése array de bytes y con ése resultado se entra en otra función; la smethod_4. Bien, vamos por partes. Supongamos que corremos el programa y se hace la consulta "Manufacturer" del procesador. Lo vemos en mi máquina:

```
Using managementObjectSearcher As ManagementObjectSearcher = New ManagementObjectSearcher
("Select * from Win32_Processor")
For Each current As ManagementBaseObject In managementObjectSearcher.[Get]()
    text = smethod_3(current("Manufacturer").ToString())
    text2 = smethod_3(current("Name").ToString())
    text3 = smethod_3(current("MaxClockSpeed").ToString())
    text4 = smethod_3(current("ProcessorID").ToString())
```

Ahí tengo un BP en Visual Studio justo cuando se va a hacer la consulta sobre el "Manufacturer". Previamente coloqué otro BP en la función smethod_3 así que continuó la ejecución y se detiene en la smethod_3:


```
Public Function smethod_3(string_3 As String) As String
    Dim arg_12_0 As HashAlgorithm = New MD5CryptoServiceProvider()
    Dim bytes As Byte() = Encoding.ASCII.GetBytes(string_3)
    Return smethod_4(arg_12_0.ComputeHash(bytes))
End Function
```

100 %

Ventana Inmediato

? string_3
"GenuineIntel"

Bien, a la consulta "Manufacturer" de mi procesador, se contesta con ése "GenuineIntel" como ven en la imagen. Se crea un array de bytes a partir de ésa cadena:

```
Dim arg_12_0 As HashAlgorithm = New MD5CryptoServiceProvider()
Dim bytes As Byte() = Encoding.ASCII.GetBytes(string_3)
Return smethod_4(arg_12_0.ComputeHash(bytes))
End Function
```

100 %

Ventana Inmediato

? bytes
{Length=12}
(0): 71
(1): 101
(2): 110
(3): 117
(4): 105
(5): 110
(6): 101
(7): 73
(8): 110
(9): 116
(10): 101
(11): 108

Lo siguiente es computar el MD5 del array de bytes y con ése resultado se entra en la funcion smethod_4:

```
Public Function smethod_4(ilist_0 As IList(Of Byte)) As String
    Dim text As String = String.Empty
    For i As Integer = 0 To ilist_0.Count - 1
        Dim expr_14 As Byte = ilist_0(i)
        ...
    Next
End Function
```

Ventana Inmediato

```
? ilist_0.ToArray
{Length=16}
(0): 228
(1): 53
(2): 196
(3): 52
(4): 102
(5): 144
(6): 10
(7): 72
(8): 154
(9): 152
(10): 174
(11): 216
(12): 187
(13): 253
(14): 107
(15): 235
```

Ahí está ya computado el hash MD5 con el que se entra en smethod_4. En ésta función se van tomando mediante un bucle For...Next todos los bytes y se hace alguna que otra operación. Obviamente para no alargar el tutorial, no voy a transcribir qué se hace con cada byte. Con que lo hagamos con dos de ellos será suficiente para entenderlo. En la primera vuelta del bucle, la variable expr_14 tomará el el primer byte de la lista (228) y las variables num y num2 van tomando valores que son resultados de unas operaciones:

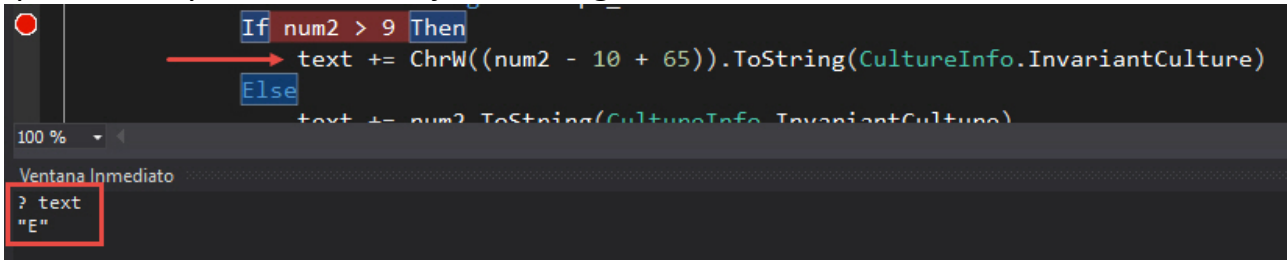
```
Public Function smethod_4(ilist_0 As IList(Of Byte)) As String
    Dim text As String = String.Empty
    For i As Integer = 0 To ilist_0.Count - 1
        Dim expr_14 As Byte = ilist_0(i)
        Dim num As Integer = CInt((expr_14 And 15))
        Dim num2 As Integer = expr_14 >> 4 And 15
        If num2 > 9 Then
            text += CharW((num2 - 10 + 65)) ToString(CultureInfo.InvariantCulture)
        End If
    Next
End Function
```

Ventana Inmediato

```
? expr_14
228
? num
4
? num2
14
```

Para la variable num2, se utiliza tambien un desplazamiento lógico hacia la derecha de cuatro bites (4) antes de hacerle el And 15.

Ahí ven en la imagen anterior lo que va tomando cada variable. Lo siguiente que se hace es consultar el valor que tienen ambas variables (num y num2) y dependiendo si son o no mayores que 9, se toman unas decisiones a la hora de ir conformando una cadena de texto. Partiendo de la base que aún seguimos con el primer byte y que num=4 y num2=14, se ejecuta la siguiente línea:

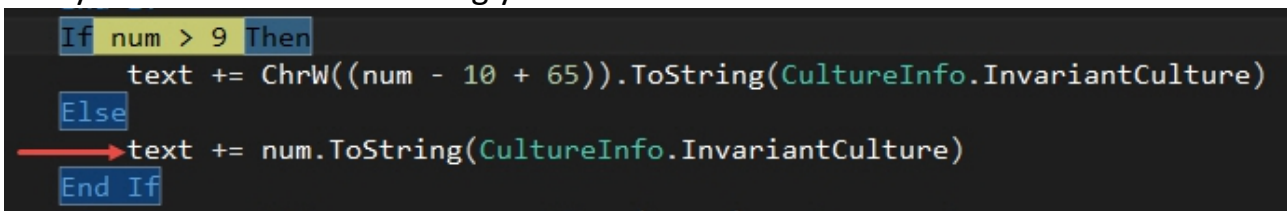


```
If num2 > 9 Then
    text += ChrW((num2 - 10 + 65)).ToString(CultureInfo.InvariantCulture)
Else
    text += num2.ToString(CultureInfo.InvariantCulture)
End If
```

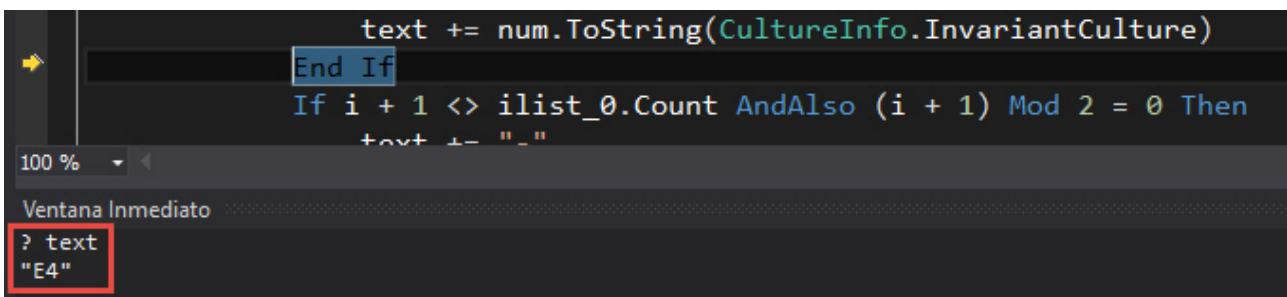
Ventana Inmediato

? text
"E"

Al ser la variable num2 (14) mayor que 9 se realiza una operación con dicha variable y se convierte a carácter con ChrW con lo que obtenemos el carácter "E" y la variable text va acumulando ése carácter para ir de a poco a poco construyendo una cadena de texto. Seguidamente se hace una consulta de la variable num que recordemos tiene un valor de 4 y ya que NO es mayor que 9, se ejecuta la instrucción dentro del Else y se convierte ése 4 a string y se le concatena al contenido de text:



```
If num > 9 Then
    text += ChrW((num - 10 + 65)).ToString(CultureInfo.InvariantCulture)
Else
    text += num.ToString(CultureInfo.InvariantCulture)
End If
```

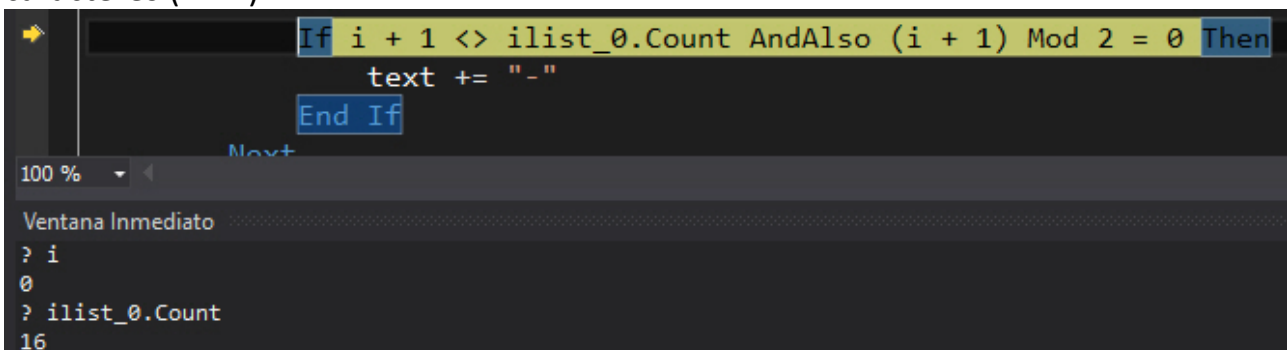


```
text += num.ToString(CultureInfo.InvariantCulture)
End If
If i + 1 <> ilist_0.Count AndAlso (i + 1) Mod 2 = 0 Then
    text += "-"
End If
```

Ventana Inmediato

? text
"E4"

Bien, lo siguiente que se hace es una especie de conteo para que cada valor par que arroje i+1, se le añada a la variable text un guión ("-"). Lo explico: como aún seguimos en la primera vuelta del bucle, i=0. Ya tenemos en la variable text, dos caracteres ("E4"):

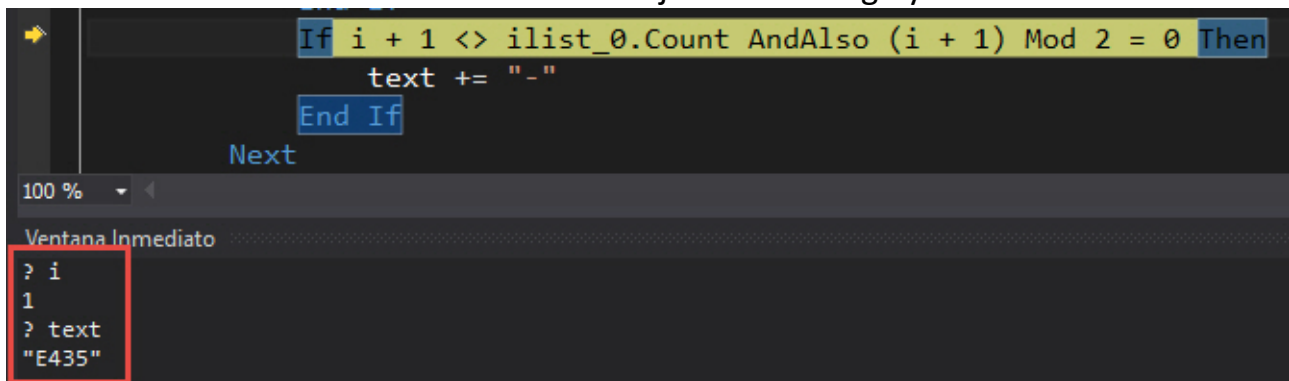


```
If i + 1 <> ilist_0.Count AndAlso (i + 1) Mod 2 = 0 Then
    text += "-"
End If
```

Ventana Inmediato

? i
0
? ilist_0.Count
16

Si $i+1$ es diferente de `ilist_0.Count` (16) y además el módulo de la división entre $i+1$ es cero, entonces se inserta un guión en lo que contenga la variable `text` pero de momento, solo se cumple la primera expresión puesto que ahora mismo $i+1=1 \neq 16$ y $1 \bmod 2$ no es cero. Para la siguiente vuelta del bucle `For...Next`, $i=1$. En la variable `text` tendremos otros dos caracteres más. Ejecuto el código y lo vemos:



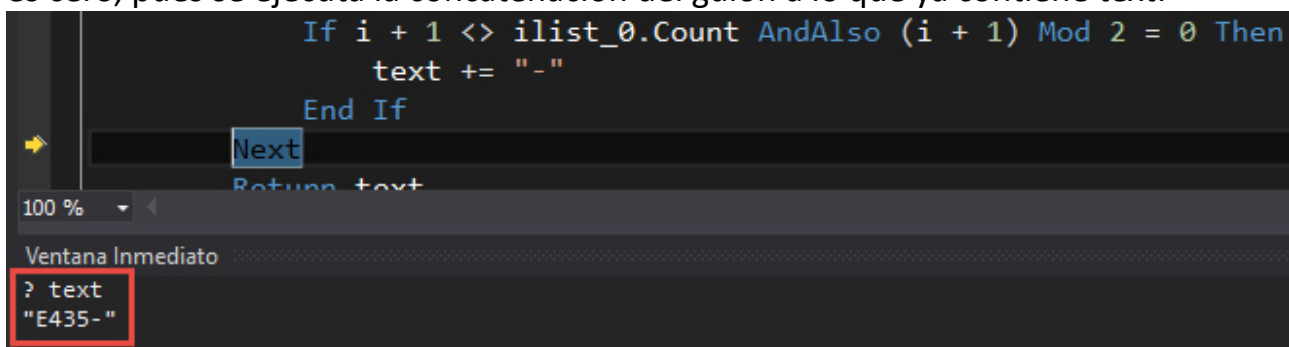
The screenshot shows a code editor with the following code:

```
If i + 1 <> ilist_0.Count AndAlso (i + 1) Mod 2 = 0 Then
    text += "-"
End If
Next
```

The Immediate Window (Ventana Inmediato) shows the current state of variables:

```
? i
1
? text
"E435"
```

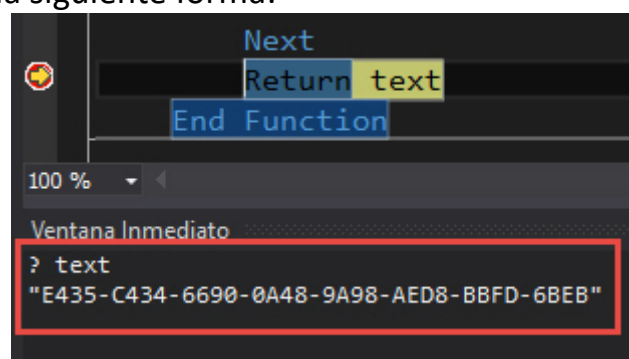
Ya estamos en la segunda vuelta del bucle. $i=1$ y la variable `text` ya va conteniendo "E435". Bien, puesto que $i+1=2$, es diferente de 16 y además, $2 \bmod 2$ efectivamente es cero, pues se ejecuta la concatenación del guión a lo que ya contiene `text`:



The screenshot shows the code editor with the same code as before, but the execution has moved to the `Next` statement. The Immediate Window (Ventana Inmediato) shows the updated state of variables:

```
? text
"E435-"
```

Bien, ahora que ya sabemos cómo funciona el código, podemos ejecutar todo el bloque hasta que se completen todas las operaciones con los bytes restantes. Cuando se han recorrido todos, la función devuelve el contenido total de la variable `text` y que queda de la siguiente forma:



The screenshot shows the code editor with the following code:

```
Next
Return text
End Function
```

The Immediate Window (Ventana Inmediato) shows the final state of the `text` variable:

```
? text
"E435-C434-6690-0A48-9A98-AED8-BBFD-6BEB"
```

Lo que sigue, es más de lo mismo y vuelta a empezar. Ahora le toca el turno a la variable `text2` y para ello y recuerden, se juega con el nombre de mi procesador:


```

For Each current As ManagementBaseObject In managementObjectSearcher.[Get]()
    text = smethod_3(current("Manufacturer").ToString())
    text2 = smethod_3(current("Name").ToString())
    text3 = smethod_3(current("MaxClockSpeed").ToString())
    text4 = smethod_3(current("ProcessorID").ToString())
    Try
        text5 = smethod_3(current("Revision").ToString())
    
```

Como comprenderán de nuevo, no es plan de volver a repetir aquí todo el proceso anteriormente expuesto. Manejando la info del procesador se rellenan las variables restantes text2, text3, text4 y text5. Se los resumo y les pongo los resultados:

```

        text4 = smethod_3(current("ProcessorID").ToString())
    Try
        text5 = smethod_3(current("Revision").ToString())
    Catch ex_D3 As Exception
        text5 = smethod_3("error")
    End Try
Next
End Using

```

Ventana Inmediato

```

? text2
"05A2-F967-EAA8-BA01-9DF7-9CC1-E7B1-C195"
? text3
"BBEB-0C1B-1FD4-4E39-2C7C-E2FD-BD13-7E87"
? text4
"19CF-DCE9-CF80-A4C6-A689-FC0F-A43E-E245"
? text5
"CB5E-100E-5A9A-3E7F-6D1F-D975-1221-5282"

```

Ya se terminó de extraer info del procesador y los resultados son éstos que ven en la imagen. Un detalle, para la variable text5 se intenta extraer info de la revision del procesador pero se utiliza un bloque Try...Catch para manejar la excepción y reconducir la situación y que sea el código del programador el que maneje dicha excepción. En mi caso particular, no puede extraer dicha info y entra en excepción por lo que se ejecuta el Catch y simplemente se utiliza la palabra "error" para generar la cadena de texto para text5. Parece ser que el programador de ésta aplicación contaba con ése detalle y actuó en consecuencia inteligentemente. Muy bien por él. Algún avisado, en éste momento, se habrá dado cuenta de un detalle: los cuatro primeros caracteres de cada variable son los que componen el numero de serie que el programa saca de mi máquina. Recuerden:

Registrar Antihook Server

Número de serie:

E435-05A2-BBEB-19CF-CB5E-A430-0A84-5964

Copiar

Llegados a este punto, ya podemos concluir que faltan tres consultas para completar el numero de serie. Dichas consultas se almacenarán en las variables text6 (la Bios), text7 y text8 (la Placa Base):

```
Using managementObjectSearcher2 As ManagementObjectSearcher = New
ManagementObjectSearcher("Select * from Win32_BIOS")
Using enumerator As ManagementObjectCollection.ManagementObjectEnumerator =
managementObjectSearcher2.[Get]().GetEnumerator()
While enumerator.MoveNext()
    text6 = smethod_3(enumerator.Current("SMBIOSBIOSVersion").ToString())
End While
End Using
End Using
Using managementObjectSearcher3 As ManagementObjectSearcher = New
ManagementObjectSearcher("Select * from Win32_BaseBoard")
For Each expr_189 As ManagementBaseObject In managementObjectSearcher3.[Get]()
    text7 = smethod_3(expr_189("SerialNumber").ToString())
    text8 = smethod_3(expr_189("Product").ToString())
Next
End Using
```

Y siguiendo el método antes descrito, las variables quedan así:

```
Ventana Inmediato
? text6
"A430-B706-819B-1F21-6D4E-3D2E-0993-E916"
? text7
"0A84-6953-7E6C-5B48-2ABF-A665-A8FB-DB9A"
? text8
"5964-BD2D-7E4D-98F2-AFDB-5165-CCE4-767E"
|
```

Ya casi se termina con la obtención del numero de serie. Ahora lo que toca es lo siguiente:

```
Return String.Concat(New String() {text.Remove(5), text2.Remove(5), text3.Remove
(5), text4.Remove(5), text5.Remove(5), text6.Remove(5), text7.Remove(5), text8.Remove
(5)}).Remove(39)
```

Con ésto se consigue crear una nueva string que resulta de concatenar los cinco primeros caracteres de cada variable gracias a la función Remove al que se le aporta un entero que no es otra cosa que el índice desde el que queremos remover ó eliminar caracteres hacia adelante, índice incluido. Un ejemplo:

```
Ventana Inmediato
? text.Remove(5)
"E435-"
? text2.Remove(5)
"05A2-"
? text3.Remove(5)
"BBEB-"
? text4.Remove(5)
"19CF-"
```

Y con el Remove(39) lo que se hace es quitar el último guión que contiene la variable

text8. El resultado final es el siguiente:

```
Ventana Inmediato
? String.Concat(New String() {text.Remove(5), t
"E435-05A2-BBEB-19CF-CB5E-A430-0A84-5964"
```

Bien, ya hemos visto cómo se consigue sacar info de una PC y obtener un numero de serie único e irrepetible. Pero claro, la gran pregunta es de dónde y cómo se saca una licencia válida para que el programa quede totalmente funcional. Ok, de eso se encarga la función smethod_0 a la que se le aporta la cadena de texto recién obtenida:

```
Public Function smethod_0(string_3 As String) As String
    Dim array As String() = string_3.Split(New Char() {"-"c})
    array(0) = smethod_3(array(0)).Remove(5)
    array(1) = smethod_3(array(1)).Remove(5)
    array(2) = smethod_3(array(2)).Remove(5)
    array(3) = smethod_3(array(3)).Remove(5)
    array(4) = smethod_3(array(4)).Remove(5)
    array(5) = smethod_3(array(5)).Remove(5)
    array(6) = smethod_3(array(6)).Remove(5)
    array(7) = smethod_3(array(7)).Remove(5).Remove(4)
    Return smethod_2(String.Concat(New String() {array(0), array(1), array(2),
array(3), array(4), array(5), array(6), array(7)}))
End Function
```

```
100 %
Ventana Inmediato
? string_3
"E435-05A2-BBEB-19CF-CB5E-A430-0A84-5964"
```

En ésta función, se dimensiona un array como string, es decir, que va a contener subcadenas de texto. Tantas subcadenas como las que resulten de dividir la cadena principal utilizando el delimitador del guión y para ello se utiliza la función Split. Si troceamos la cadena principal utilizando los guiones como delimitador, es fácil imaginar cómo quedará el array con sus diferentes subcadenas:

```
Ventana Inmediato
? array
{Length=8}
(0): "E435"
(1): "05A2"
(2): "BBEB"
(3): "19CF"
(4): "CB5E"
(5): "A430"
(6): "0A84"
(7): "5964"
```

Las líneas siguientes se van a encargar de sobrescribir dicho array de cadenas. Se entrará sucesivamente en smethod_3 al que se le aporta precisamente dicho array por índices(array(0),array(1),etc..etc.):

```

array(0) = smethod_3(array(0)).Remove(5)
array(1) = smethod_3(array(1)).Remove(5)
array(2) = smethod_3(array(2)).Remove(5)
array(3) = smethod_3(array(3)).Remove(5)
array(4) = smethod_3(array(4)).Remove(5)
array(5) = smethod_3(array(5)).Remove(5)
array(6) = smethod_3(array(6)).Remove(5)
array(7) = smethod_3(array(7)).Remove(5).Remove(4)

```

Por ejemplo, el array(0) contiene en éste momento "E435" y quedará sobrescrito con lo que resulte del retorno de la función smethod_3 a la que se entre precisamente cn dicha cadena "E435". Bien para no aburrirles de nuevo con tanta parafernalia, el funcionamiento de smethod_3 está descrito desde la pagina 5 hasta la 8. A dicho retorno de smethod_3 se le remueven caracteres desde el índice 5 en adelante. Les pongo ejemplo de cómo queda el retorno en la variable:

```

Ventana Inmediato
? smethod_3(array(0))
"ABA0-5BDC-EFDF-54FA-570D-7322-49FF-DC5E"
? smethod_3(array(0)).Remove(5)
"ABA0-"

```

Sigo ejecutando linea a linea hasta que se completen todas las sobreescrituras:

```

array(7) = smethod_3(array(7)).Remove(5).Remove(4)
Return smethod_2(String.Concat(New String() {array(0), array
(1), array(2), array(3), array(4), array(5), array(6), array(7)}))
End Function

```

100 %

```

Ventana Inmediato
? array
{Length=8}
(0): "ABA0-"
(1): "FB0D-"
(2): "C2AC-"
(3): "DB8D-"
(4): "9F4C-"
(5): "0B96-"
(6): "0B35-"
(7): "C67B"

```

El punto de ejecución se encuentra detenido ahí como ven justo cuando se va a proceder a la concatenación de los diferentes arrays. Por cierto, el array(7) si se fijan, primero se le aplica el Remove(5) y luego el Remove(4) para que se elimine el guión. Todo concatenado queda así:

```

Ventana Inmediato
? String.Concat(New String() {array(0), array
"ABA0-FB0D-C2AC-DB8D-9F4C-0B96-0B35-C67B"

```


Ése resultado de la concatenación se va a meter en la función smethod_2, que es previo al retorno así que ejecuto la línea y sigo hasta que entro en smethod_2:

```
Public Function smethod_2(string_3 As String) As String
    Dim result As String
    Using mD As MD5 = MD5.Create()
        Dim bytes As Byte() = Encoding.ASCII.GetBytes(string_3)
        Dim arg_1F_0 As Byte() = mD.ComputeHash(bytes)
        Dim stringBuilder As StringBuilder = New StringBuilder()
        Dim array As Byte() = arg_1F_0
        For i As Integer = 0 To array.Length - 1
            Dim b As Byte = array(i)
            stringBuilder.Append(b.ToString("X2"))
        Next
        result = stringBuilder.ToString().ToLower()
    End Using
    Return result
End Function
```

Ya dentro de la función, en string_3 tenemos la cadena obtenida en el paso anterior. Aquí resumiendo un poco, con la variable mD se prepara el entorno para trabajar con el hash MD5. Se obtienen los bytes del contenido de la variable string_3 y se computa el MD5 de dichos bytes. Ejecuto esas líneas y tenemos computado el hash:

```
Ventana Inmediato
? arg_1F_0
{Length=16}
(0): 73
(1): 60
(2): 249
(3): 119
(4): 130
(5): 52
(6): 222
(7): 178
(8): 247
(9): 4
(10): 32
(11): 191
(12): 178
(13): 113
(14): 142
(15): 249
```

Ahora en las siguientes líneas se prepara una variable (array) para que almacene los bytes recién computados y con el bucle For...Next se va creando una cadena de texto con stringBuilder.Append y que tendrá el formato "X2", es decir, valores hexadecimales de dos dígitos. Por ejemplo y tomando el primer byte (73), en la primera vuelta del bucle, la cadena que se irá construyendo, contendrá la representación hexadecimal de 73 (h49):


```
Ventana Inmediato
? b.ToString("X2")
"49"
|
```

Para la siguiente vuelta del bucle, se toma el byte 60 y se formatea:

```
Ventana Inmediato
? b.ToString("X2")
"3C"
|
```

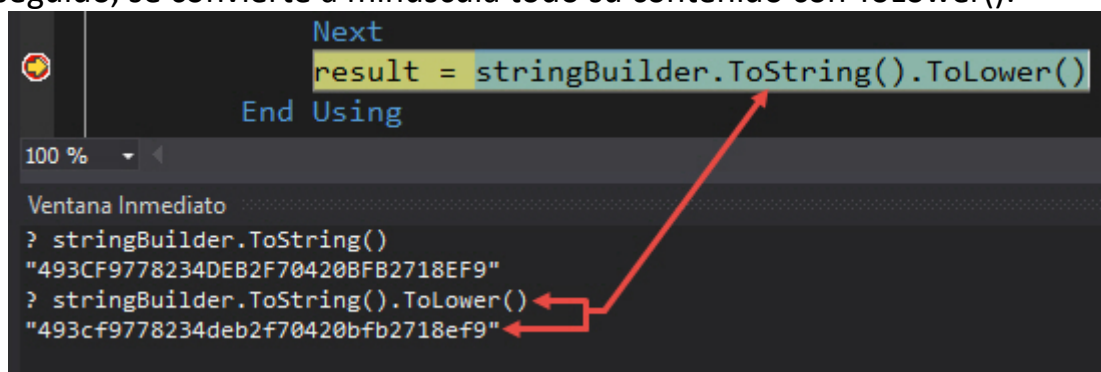
En la tercera vuelta, y con ésto finalizo la explicación, se toma el byte 249 y con el formateo:

```
Ventana Inmediato
? b.ToString("X2")
"F9"
|
```

La cadena que se está construyendo, hasta ahora está conformada de ésta manera:

```
Ventana Inmediato
? stringBuilder.ToString()
"493CF9"
```

Si ejecuto todo el bloque hasta la última vuelta obtenemos ya la cadena completa y acto seguido, se convierte a minúscula todo su contenido con `ToLower()`:



```
Next
result = stringBuilder.ToString().ToLower()
End Using
```

100 %

```
Ventana Inmediato
? stringBuilder.ToString()
"493CF9778234DEB2F70420BFB2718EF9"
? stringBuilder.ToString().ToLower()
"493cf9778234deb2f70420bfb2718ef9"
```

Y ésta cadena pasada a minúscula es precisamente la licencia que necesitamos para desbloquear el programa. Recuerden que ésta cadena solo es válida en mi máquina y que en otras máquinas obviamente será diferente:

Servidor BF4 Resolver

ID Usuario

E435-05A2-BBEB-19CF-CB5E-A430-0A84-5964

Contraseña

493cf9778234deb2f70420bfb2718ef9

Generar Clave

Registrar Antihook Server

Número de serie:

E435-05A2-BBEB-19CF-CB5E-A430-0A84-5964

Copiar

Licencia:

493cf9778234deb2f70420bfb2718ef9

Pegar

Registrar

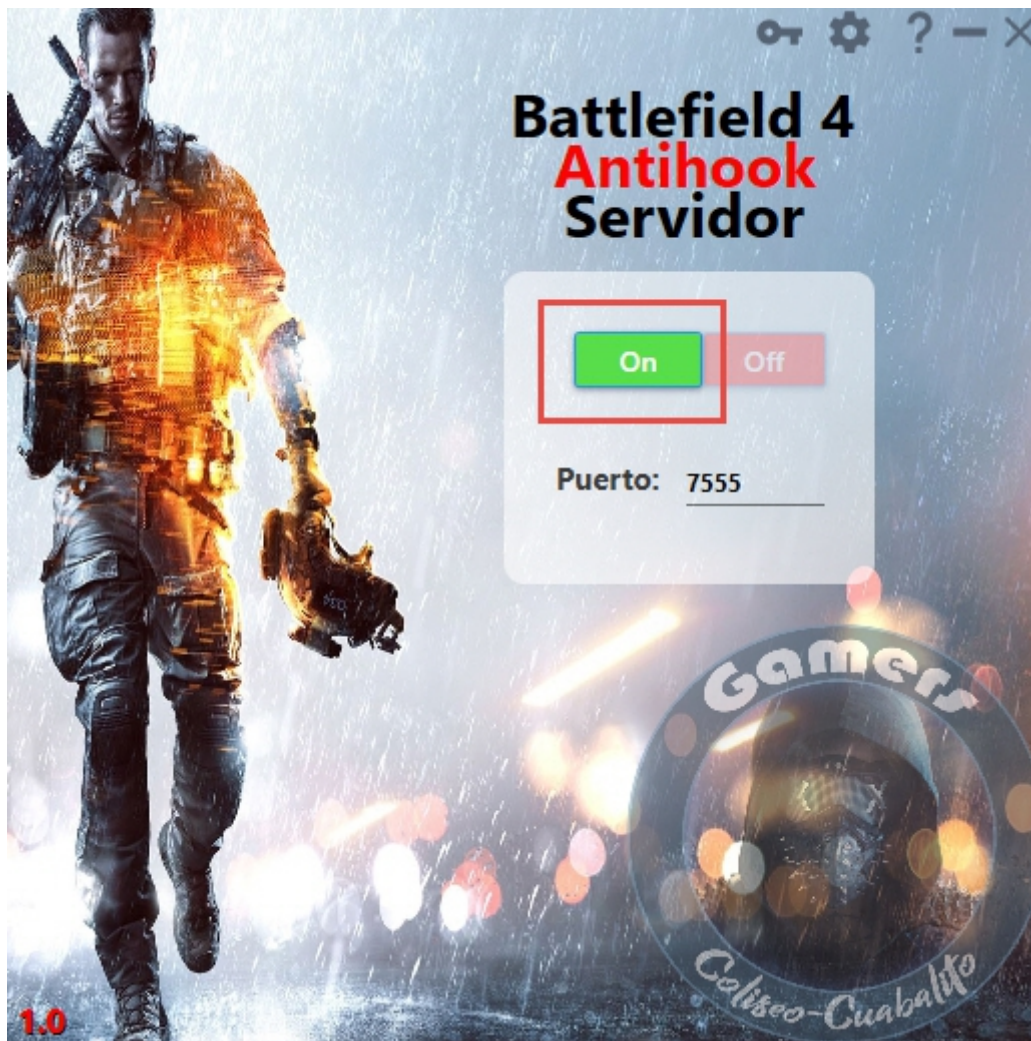
Licencia:

493cf9778234deb2f70420bfb2718ef9

Pegar

Registrar

✓



El programa acepta como buena la licencia generada y gracias a ello, se activa el botón verde para que el programa haga lo que tenga que hacer. Yo no tengo ni idea lo que hace porque no tiro de juegos on-line o lo que quiera que haga esto. Y con este último paso se puede dar por finalizado el estudio de este software y descubrir así cómo licenciarlo. Espero que no les haya sido pesado este tuto. Los que ya me conocen de antes sabrán que prefiero escribir unas cuantas paginas más de la cuenta si con ello logro que lo entiendan lo mejor posible.

Un saludo muy cordial a toda la lista de CracksLatinos y hasta la próxima.