

Neófito Reversando .NET [Entrega #1][LUISFECAB]



HERRAMIENTAS	Windows 7 Home Premium SP1 x32 Bits (SO donde trabajamos) Microsoft Visual Studio Professional 2017 dnSpy v6.0.5 DESCARGAR HERRAMIENTAS DESCARGAR TUTO+ARCHIVOS
AUTOR	LUISFECAB
RELEASE	Agosto 1 2019 [TUTORIAL 018]

INTRODUCCIÓN

La idea de estas futuras entregas de “**Neófito Reversando .NET**” surgió en el canal de **@PeruCrackerS** mientras hablábamos de algunos de mis tutoriales que tenían alguna relación con **.NET**. En una de esas charlas mi amigo **@AbelJM** comentada que no se defiende muy bien con los **.NET**; entonces de ahí me surgió la idea de hacer unos escritos reversando **.NET**.

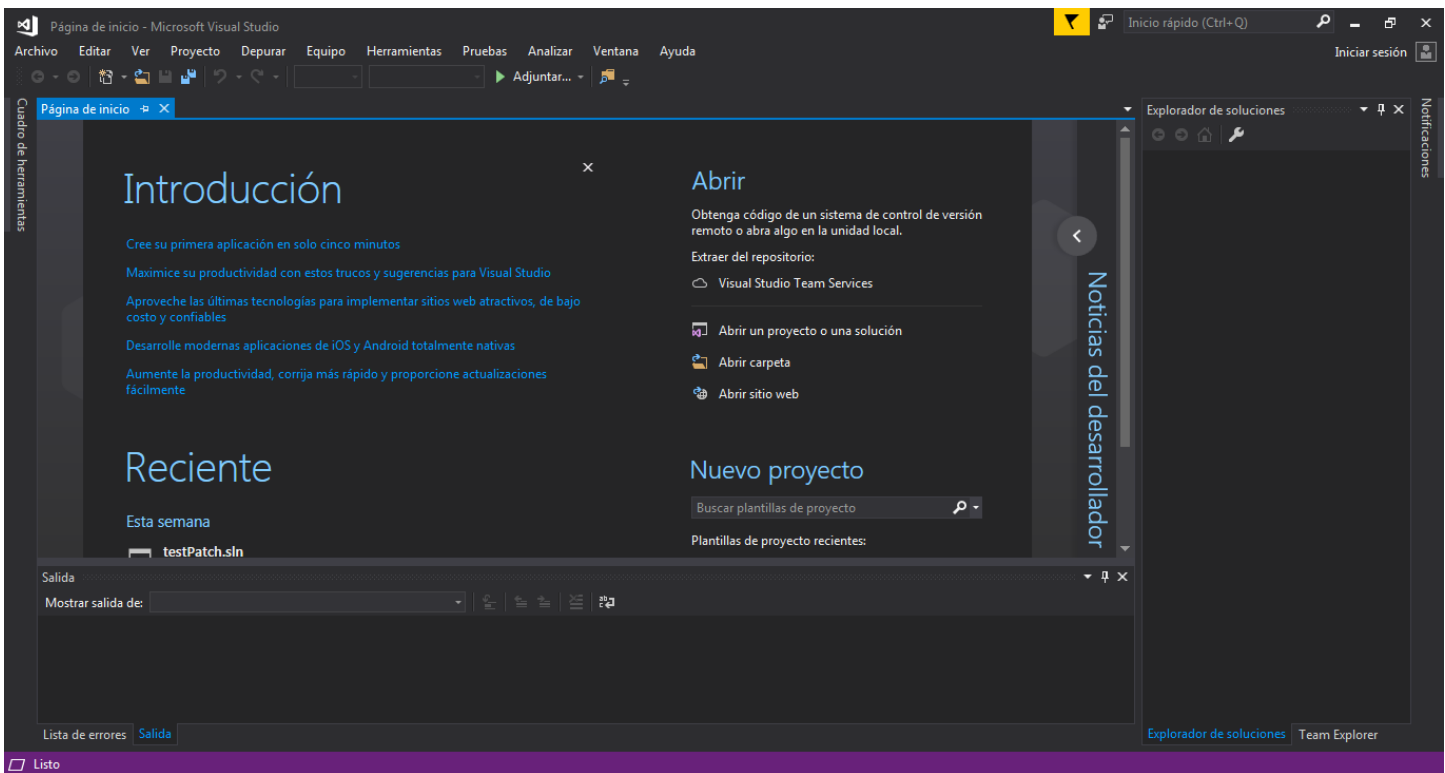
A estos escritos los llamaré “**Neófito Reversando .NET**” porque yo no soy un experto en Cracking en general y mucho menos en **.NET**, solo soy un Neófito que quiere compartir lo poco que sabe según lo entiende y comprende. Creo que no aportaré nada nuevo en cuanto a Reversar **.NET** se refiere, pero trataré de compartir mi forma de entender cómo Reversar **.NET**.

Ahora, algo que creo muy importante, que prácticamente es un prerequisite para que puedas sacarle provecho a estas lecturas, y es que tengas un conocimiento básico (como lo tengo yo) en saber algo de **.NET**, y me refiero a programar cosas sencillas en lenguajes tales como **C#.NET** o **VB.NET**. ¿Y por qué digo esto?; se debe a que el código desamplado en las herramientas especializadas para **.NET** es muy diferente a lo que venimos trabajando comúnmente con otros lenguajes como **C/C++**, **VB 6.0**, **Delphi**... que los trabajamos con las herramientas de siempre como el **<OllyDBG>** o el **<x64DBG>** en donde trabajamos con el lenguaje de bajo nivel **Assembler**. En cambio cuando estemos Reversando **.NET**, tendremos prácticamente el código fuente del programa, y si sabemos y entendemos **.NET**, nos será relativamente fácil poder Crackear aplicaciones hechas en **.NET**, así que si no tienes ni idea de **.NET**, respetuosamente te aconsejo como amigo que primero te leas algún libro que te enseñe lo básico de programar en **.NET**, preferiblemente **VB.NET** porque ese es el que manejaremos aquí para hacer nuestros pequeños CrackMe's de Neófito.

Dedicado a mis amigos de **@PeruCrackerS** y especialmente para **@AbelJM**, para ver si se anima a adentrarse en el mundo de los **.NET**.

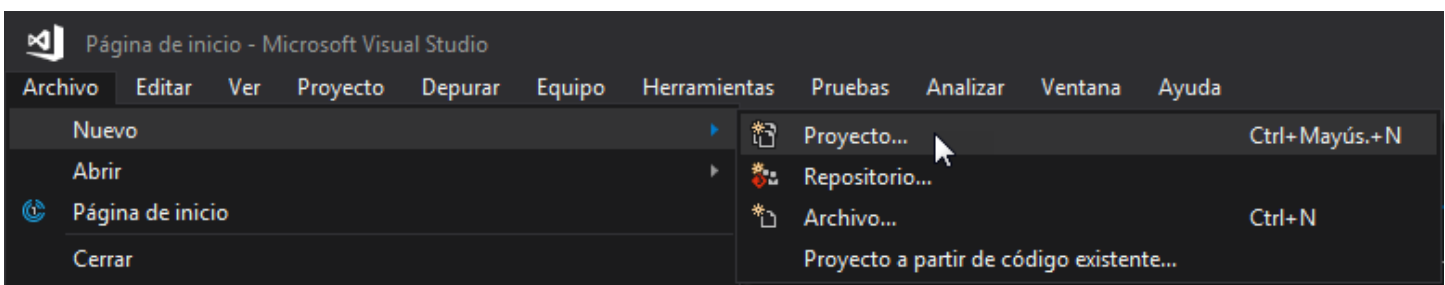
ENTREGA #1

Ya les dije en la **INTRODUCCIÓN** que es mejor saber algo de **.NET** antes de empezar pero aquí también explicaremos algo de programación, así que usaremos el **<Visual Studio Professional 2017>**.



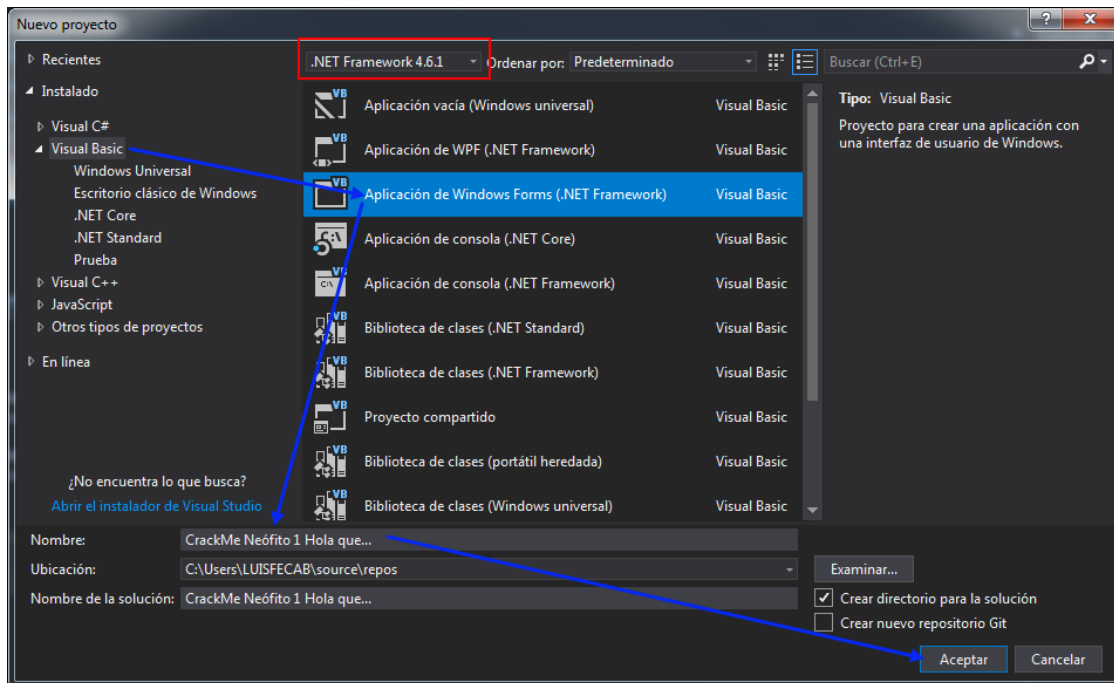
Como sabemos **C#.NET** y **VB.NET** son lenguajes de programación orientado a objetos los cuales tienen eventos en donde programamos nuestras instrucciones para realizar la tarea que nosotros queremos hacer. Lo objetos son los ya conocidos controles como Formularios, Botones, Cajas de texto, Labels y muchísimos más.

No voy a profundizar en el manejo del **IDE**, esa es tarea de ustedes. Vamos a hacer nuestro primer **<CrackMe-Neófito #1: Hola que...>**. Creamos un **<Nuevo proyecto>**.

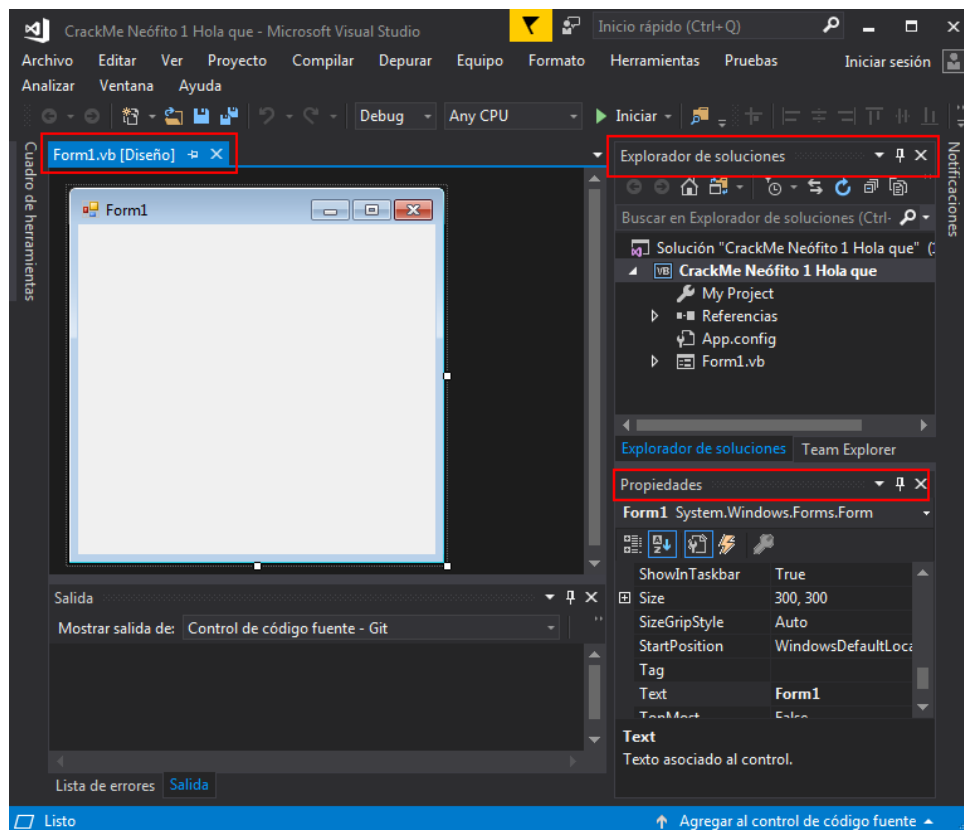


Se nos abrirá la ventana para crear nuestro proyecto, en donde hay muchas opciones. Miremos cómo creamos el nuestro.

Neófito Reversando .NET [Entrega #1][LUISFECAB]



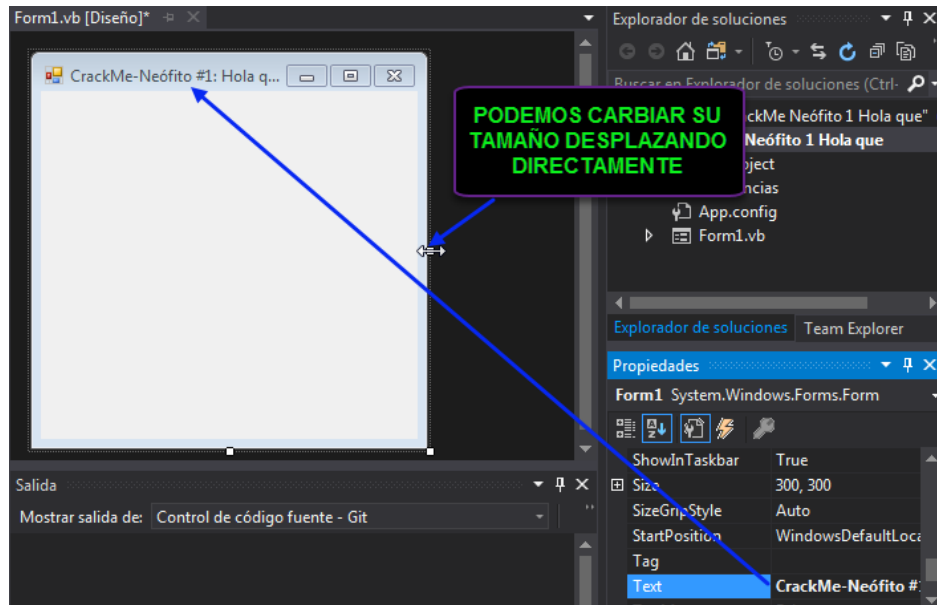
Como ven, escogí "Visual Basic" -> "Aplicación de Windows Form". Lo otro relevante es el SDK, y es el ".NET Framework 4.6.1".



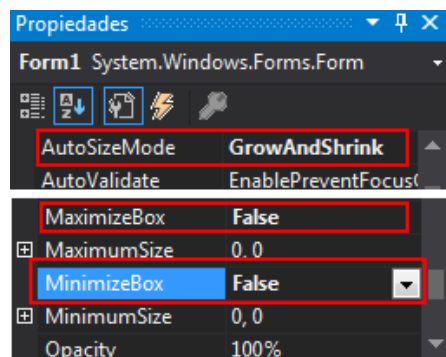
Tenemos esas tres secciones, que son las principales que nos salen para gestionar nuestro proyecto. Vamos directamente a la de "Propiedades" en donde podemos configurar todo lo relacionado con los objetos o controles. Como vemos, aparece las Propiedades del Formulario, **Form1**, ahí podemos configurarle muchas cosas, como el

Neófito Reversando .NET [Entrega #1][LUISFECAB]

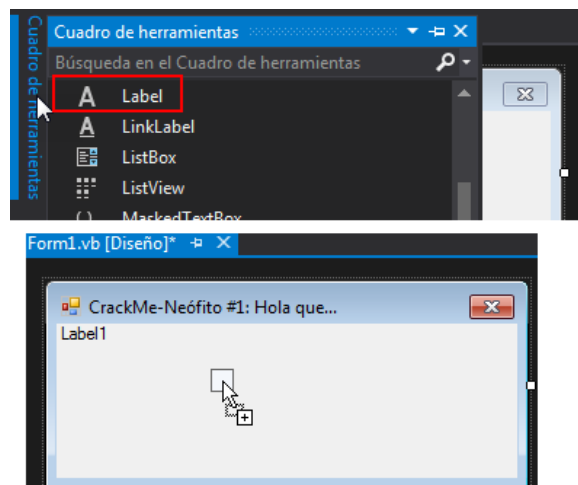
tamaño, color, ubicación, nombre y otras "ijuehil" cosas. Vamos a hacerle unos cuantos cambios desde ahí.



La Propiedad "**Text**" le cambia el nombre al Formulario (ventana). También cambiamos su tamaño directamente desde el Formulario. Cambiemos unas últimas Propiedades.

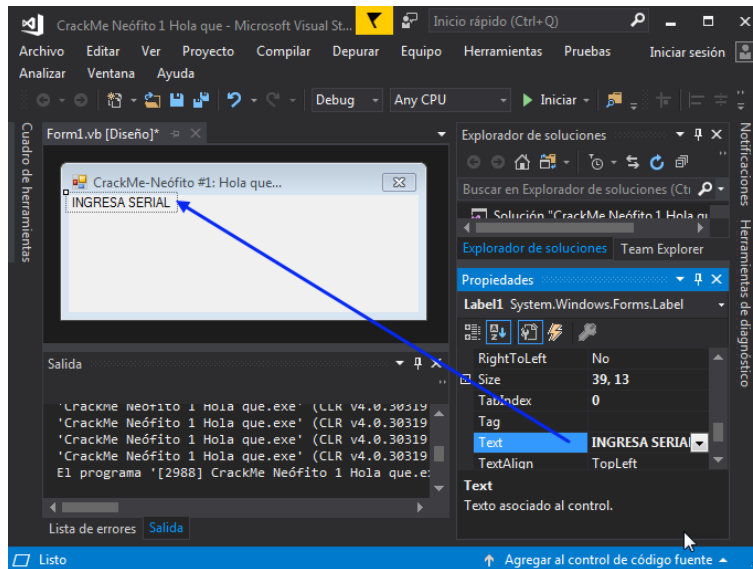


Cambiémosla como se muestran en la imagen. Con eso cambios evitamos que nuestro Formulario le puedan cambiar el tamaño. Ahora le agregamos un "**Label**".

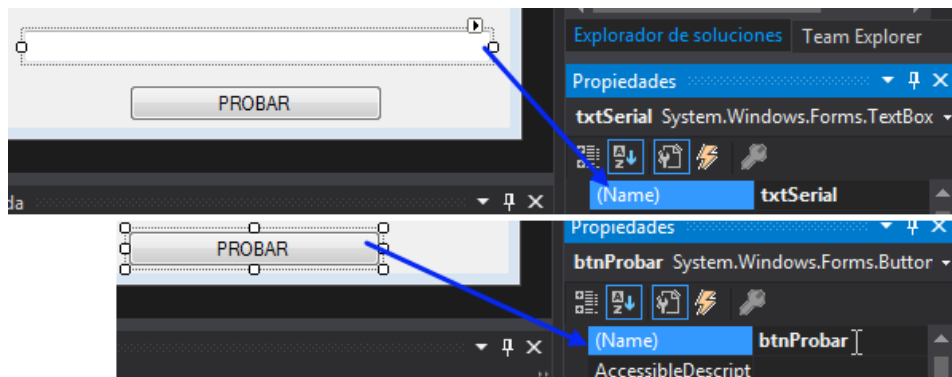


Neófito Reversando .NET [Entrega #1][LUISFECAB]

En la pestaña "**Cuadro de herramientas**", se desplegarán todos los controles. Buscamos el "**Label**", lo podemos seleccionar y arrastrarlo al "**Formulario**" o con <Doble-Clic> se puede agregar también.

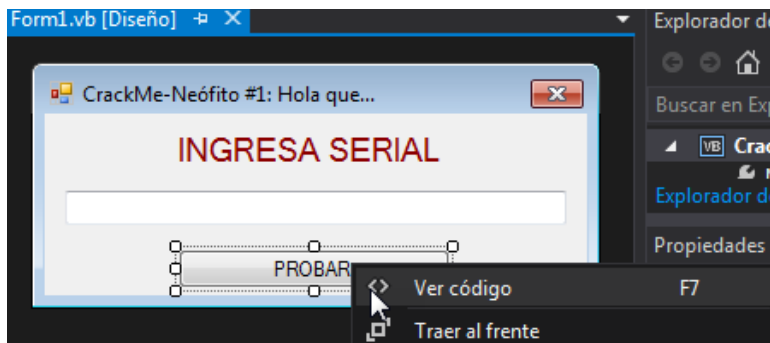


Seleccionamos el "**Label**" y cambiamos el texto que muestra en su propiedad "**Text**"="**INGRESA SERIAL**". Yo le cambiaré otras Propiedades para dejar el "**Label**" más presentable. Ahora, agregaremos un "**TextBox**" (Caja de Texto) y un "**Button**" (Botón), al Botón le cambiamos la Propiedad "**Text**"="**PROBAR**". También les cambiamos el tamaño y posición de estos. Miremos la Propiedad "**(Name)**".

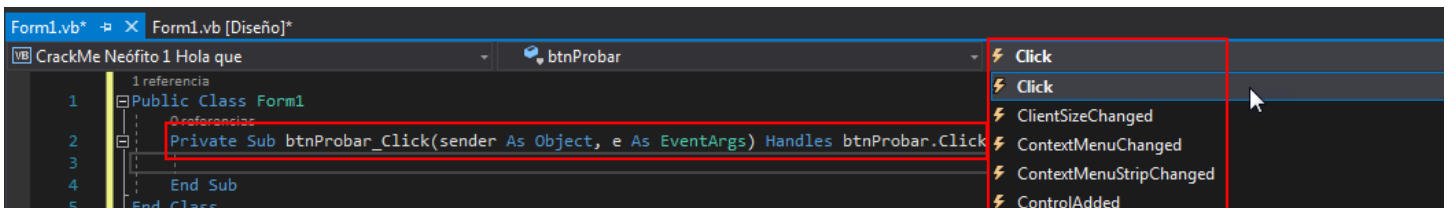


Ahí podemos cambiar el nombre de los Controles, lo cual es ideal porque con el nombre que le pongamos aquí, es que será mostrado cuando vallamos a unos de sus eventos; y una gran ventaja de esto es que podemos reconocer fácilmente en que control nos encontramos. Ya teniendo la parte gráfica de nuestro **CrackMe**, ahora vamos a programarlo. Será programar el **Evento_Click** para validar nuestro **SERIAL** ingresado. Seleccionamos el Botón "**PROBAR**".

Neófito Reversando .NET [Entrega #1][LUISFECAB]



Con **<Clic_Derecho>** y escogemos **"Ver código"**. Para acceder de forma rápida podemos usar la tecla **<F7>**. También con **<Doble_Clic>** se accede al editor de código.



Por defecto el control **"Button"** nos muestra el **Evento_Click**, que para nuestro caso sería el **btnProbar_Click**, que es el en nombre que nosotros le pusimos en la Propiedad **"(Name)"**. Ahora, cuando hagamos **<Clic>** que se valide nuestro serial.

```
Private Sub btnProbar_Click(sender As Object, e As EventArgs) Handles btnProbar.Click
    'Esto es un comentario
End Sub
```

La comilla simple (') nos permite agregar comentario o convertir código en un comentario y de esa forma evitar que se ejecute, si así lo deseamos. Lo primero que debemos saber es cómo declarar variables, y en **VB.NET** tenemos.

```
Public Class Form1
    'Public declara la variable para ser usada en todo el Proyecto.
    Public PublicstrSerial As String = "Cadena de texto" ' Para String
    Public PublicstrLetra As Char = "A"c '
    Public PublicnumDecimal As Decimal
    Public PublicnumEntero As Integer
    Public PublicnumLargo As Long
    Public PublicnumDoble As Double
    Public Publicsinobool As Boolean

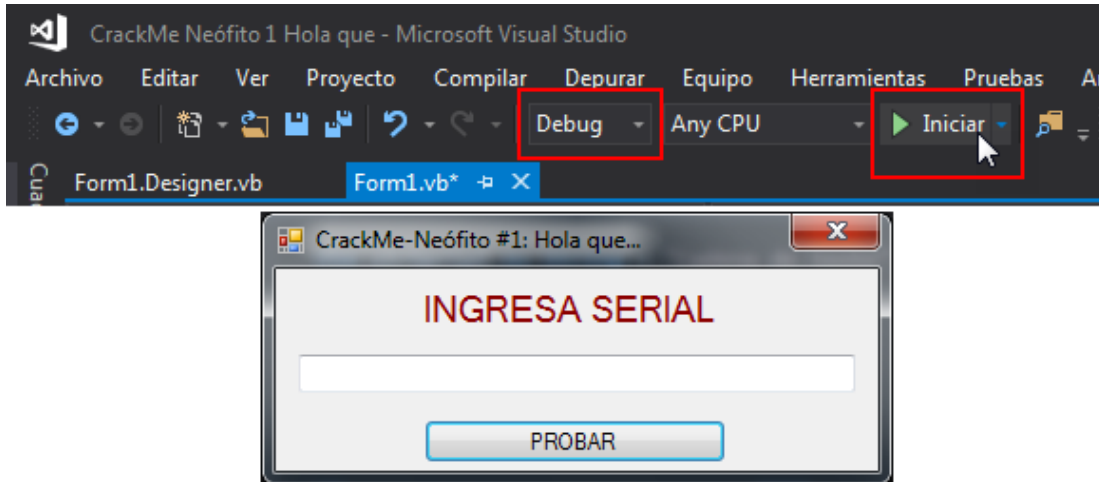
    'Private declara la variable para ser usada en todo el Módulo.
    Private PrivatestrSerial As String = "Cadena de texto" ' Para String
    Private PrivatestrLetra As Char = "A"c '
    Private PrivatenumDecimal As Decimal
    Private PrivatenumEntero As Integer
    Private PrivatenumLargo As Long
    Private PrivatenumDoble As Double
    Private Privatesinobool As Boolean

    0 referencias
    Private Sub btnProbar_Click(sender As Object, e As EventArgs) Handles btnProbar.Click
        Dim strSerial As String = "Cadena de texto" ' Para String
        Dim strLetra As Char = "A"c '
        Dim numDecimal As Decimal
```

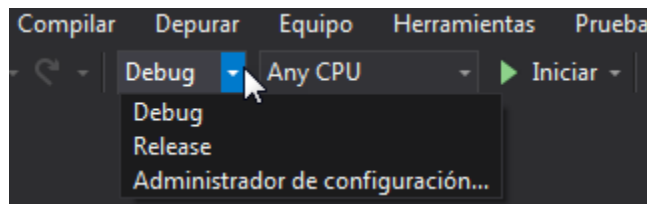

Neófito Reversando .NET [Entrega #1][LUISFECAB]

Podemos declarar una variable como **Public** para ser usada en todo el Proyecto, con **Private** se puede acceder a esa variable en todo el Módulo, y para declarar variables dentro de un procedimiento lo hacemos con **Dim**.

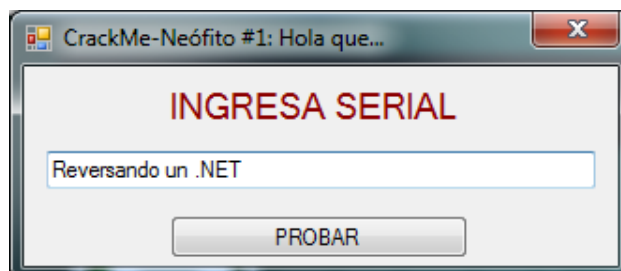
Pensaba dejar el código de la validación pero si hacía esto se perdería la gracia de hallar el **SERIAL** porque el código muestra el **SERIAL** directamente. Mejor compilamos el Proyecto para crear nuestro **CrackMe**, para de esa forma entrarle al Reversing porque la idea es enseñar lo poco que sé reversando **.NET**, y no en programar **.NET**.



El Proyecto puede ser compilado sin necesidad que se ejecute el **CrackMe** ya compilado pero lo haremos desde la opción "**Iniciar**" para que se compile y ejecute a la vez.

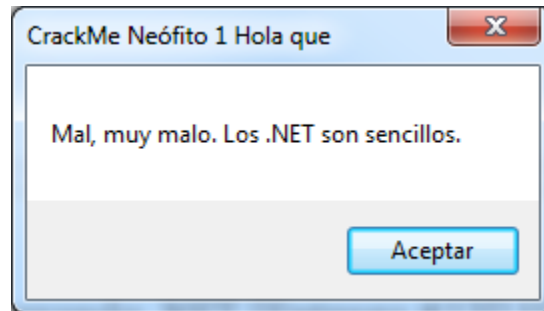


Algo que se me olvidaba y es que podemos escoger el modo de la compilación. Se supone que "**Debug**" permite dejar información útil para que en casos de errores de programación al Debugear sea más sencillo hallarlos, y creo que la opción "**Release**" optimiza la compilación evitando compilar código innecesario. Bueno, no me crean mucho lo que acabo de decir, es lo que creo que tienen de diferencia a grandes rasgos. Yo lo compilaré como "**Release**".



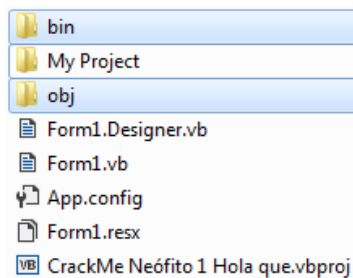
Neófito Reversando .NET [Entrega #1][LUISFECAB]

Ingresamos un **SERIAL** para "**PROBAR**". No esperen que ingrese el **SERIAL** correcto porque entonces no hay gracia, no; para eso es que estamos aprendiendo a Reversar aplicaciones hechas en **.NET**.

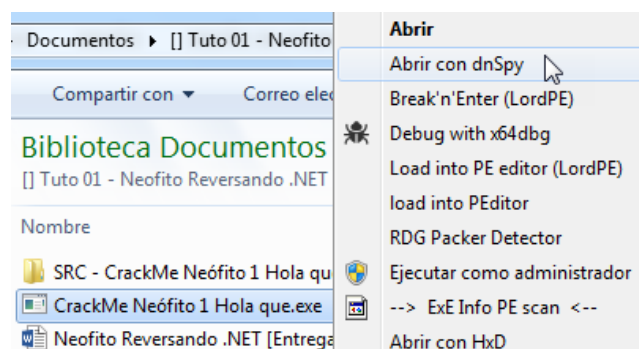


Nos sale el **CHICO MALO**. No importa ya encontraremos el **SERIAL** correcto, que es lo de menos porque el objetivo de esta primera entrega es familiarizarnos con los **.NET**. Aceptamos el mensaje y cerramos el **CrackMe**.

El binario lo podemos hallar en estas dos carpetas en donde tenemos guardado el Proyecto. Es curioso que se cree en dos carpetas distintas; no tengo ni idea el porqué de eso. Sería bueno poder averiguar la razón de eso, tal vez más adelante busque por Internet si alguien lo explica.

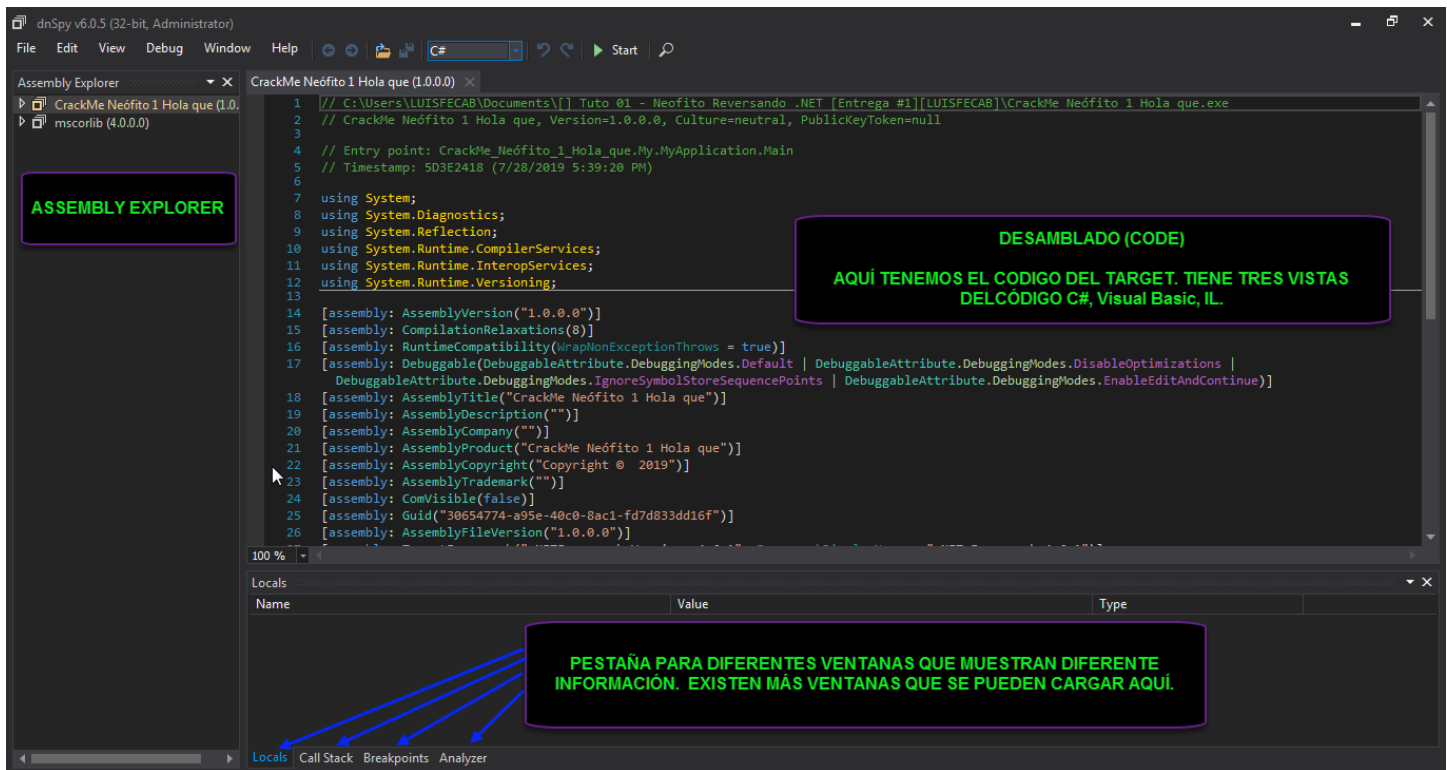


En esas dos carpetas se nos guarda el Compilado. Ya teniendo el **CrackMe** lo vamos a Debugear con el **<dnSpy v6.0.5>**. Existen otras aplicaciones pero creo que el **<dnSpy>** es muy bueno y es ideal para iniciarnos, es más, puede que algunos le hallen desventajas con respecto a otros Debugger, pero para mí con el **<dnSpy>** es más que suficiente para Crackear **.NET**.

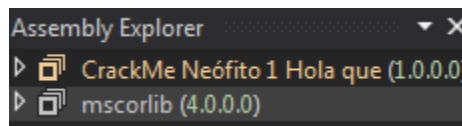


He copiado el **CrackMe** donde escribo este tutorial. Lo abrimos con el **<dnSpy>**.

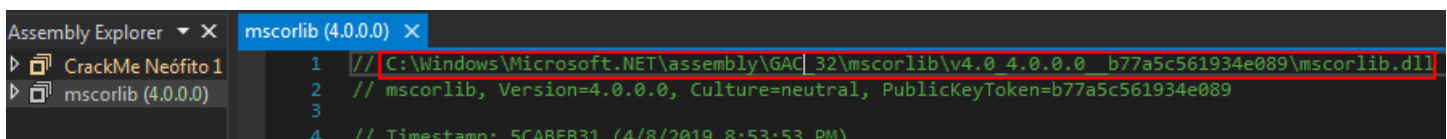
Neófito Reversando .NET [Entrega #1][LUISFECAB]



Tenemos esas tres secciones principales. La primera sería el "**ASSEMBLY EXPLORER**" y esta nos muestra todos los elementos que componen nuestro **TARGET**, que sería nuestro **Crackme**, como también todas aquellas **DLL** que usa para su funcionamiento.



Como vemos tenemos dos elementos principales que sería nuestro "**CrackMe Neófito 1 Hola que**" y la **DLL** "**mscorlib**". Seleccionemos esa **DLL**.

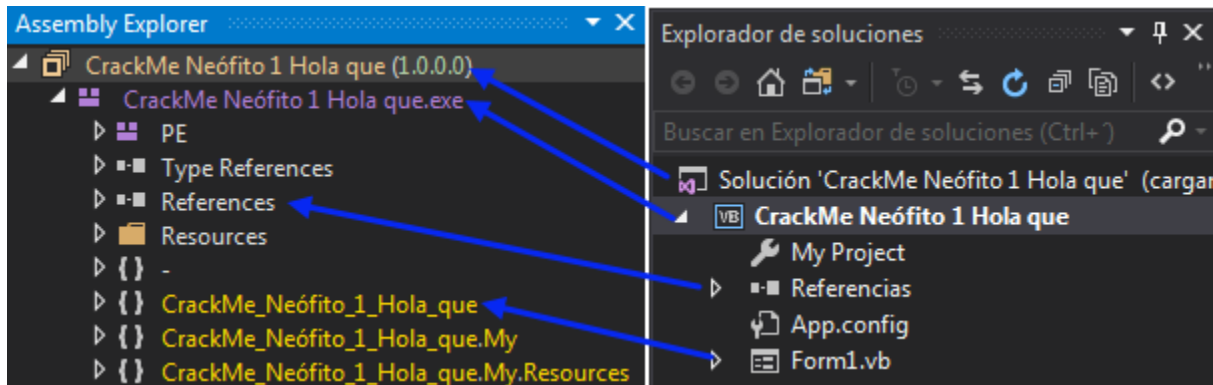


Como vemos es una **DLL** de Microsoft y si recordamos, venimos acostumbrados a que las **DLL** de Windows las tenemos en System32, donde se tiene toda la implementación de las **APIs Win32**, ese entorno lo conocemos como **COM**, pero como **.NET** es un entorno completamente nuevo desde su base pues este usa sus propias librerías. En teoría **.NET** es mejor porque ya no hace uso del **COM**, el cual supuestamente presenta más fallas. Entonces **.NET** deja atrás el uso de **APIs Win32** y crea sus propias funciones, claro que si deseamos podemos hacer uso de las **APIs**.

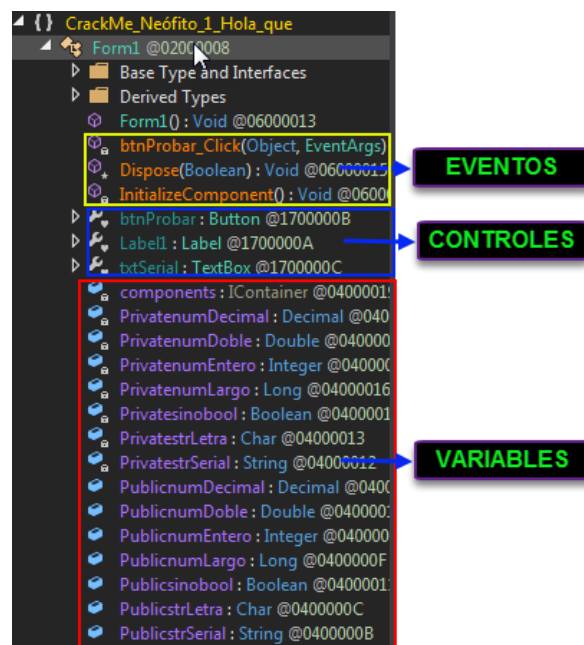
Recordemos que muchas veces los programas utilizan sus propias **DLL** para validar su estado de Activación, y es ahí en el "**ASSEMBLY EXPLORER**" que se cargará la **DLL** cuando sea utilizada por el **TARGET**. El truco aquí es ir conociendo los nombres de las **DLL** del sistema y cuáles no, y como ya vimos solo con seleccionarla podremos

Neófito Reversando .NET [Entrega #1][LUISFECAB]

saber si hace parte de **.NET Framework**. Miremos qué tenemos dentro de nuestro **CrackMe** en el "**ASSEMBLY EXPLORER**".



A la izquierda tenemos el **ASSEMBLY EXPLORER** del <dnSpy> y a la derecha vemos el "**Explorador de soluciones**" del <Visual Studio Professional 2017>. Como vemos tienen mucho parecido. Si observamos bien, la última flecha que señala el **Form1**, no tienen el mismo nombre pero si desplegamos en el **ASSEMBLY EXPLORER**.

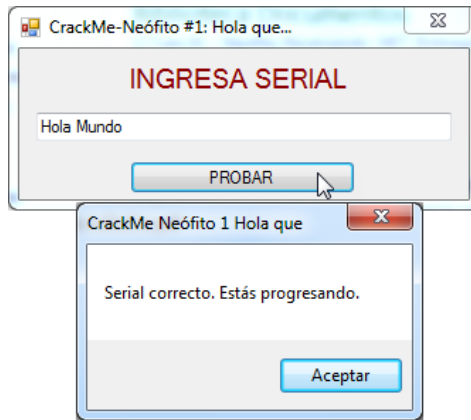


Ahí tenemos todo lo que el **Form1** contiene, están las Variables que declaramos, están los tres Controles que agregamos y los Eventos. El Evento **btnProbar_Click**, tiene la comprobación de nuestro **SERIAL**; eso no es ningún secreto, si aquí mismo programamos el **CrackMe**. Miremos el **btnProbar_Click**.

```
btnProbar_Click(Object, EventArgs) : Void
1 ' CrackMe_Neófito_1_Hola_que.Form1
2 ' Token: 0x06000014 RID: 20 RVA: 0x00002247 File Offset: 0x0000447
3 Private Sub btnProbar_Click(sender As Object, e As EventArgs)
4     If Operators.CompareString(Me.txtSerial.Text, "Hola Mundo", False) = 0 Then
5         Interaction.MsgBox("Serial correcto. Estás progresando.", MsgBoxStyle.OkOnly, Nothing)
6         Return
7     End If
8     Interaction.MsgBox("Mal, muy malo. Los .NET son sencillos.", MsgBoxStyle.OkOnly, Nothing)
9 End Sub
```

Neófito Reversando .NET [Entrega #1][LUISFECAB]

Se compara nuestro **SERIAL** ingresado y que está almacenado en **Me.txtSerial.Text** con la String **"Hola Mundo"**. Si son iguales vamos al **CHICO BUENO**. Entonces sabemos que nuestro **SERIAL** correcto es **"Hola Mundo"**.



Bueno, ya tenemos el **SERIAL** correcto pero falta mucho por analizar. Comparemos con el código original.

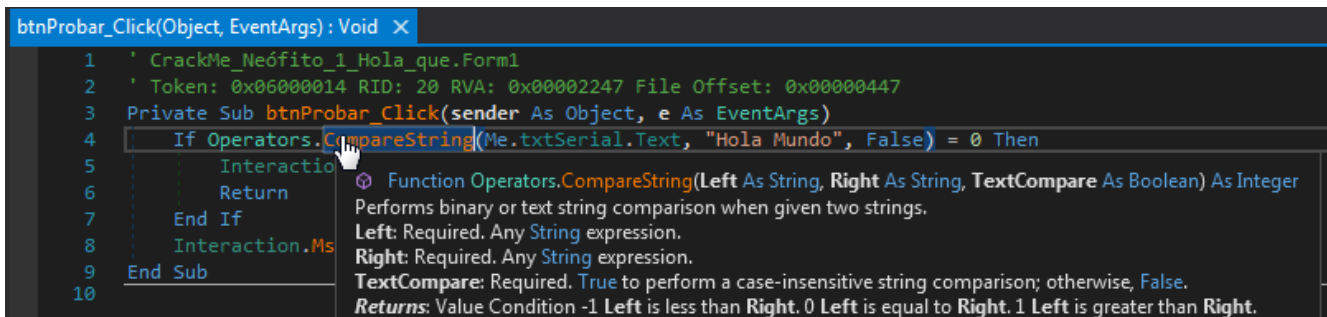
```
Private Sub btnProbar_Click(sender As Object, e As EventArgs) Handles btnProbar.Click

    Dim strSerial As String = "Cadena de texto" ' Para String
    Dim strLetra As Char = "A"c '
    Dim numDecimal As Decimal
    Dim numEntero As Integer
    Dim numLargo As Long
    Dim numDoble As Double
    Dim sinobool As Boolean

    If txtSerial.Text = "Hola Mundo" Then
        MsgBox("Serial correcto. Estás progresando.")
    Else
        MsgBox("Mal, muy malo. Los .NET son sencillos.")
    End If

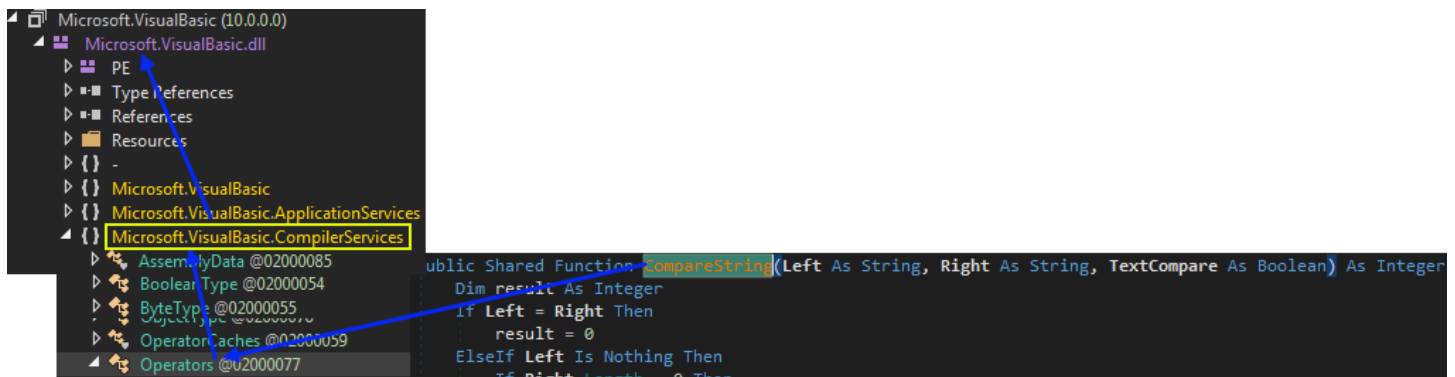
End Sub
```

Los códigos son muy parecidos, nosotros utilizamos el igual ("=") para averiguar si nuestro **SERIAL** ingresado es correcto. Al compilar el Proyecto nuestra igualdad de Strings se debe hacer de alguna forma, y eso se hace con el Método **CompareString**, que no es otra cosa que una función que retorna un valor entero, **0** si hay igualdad. Entonces los Métodos son simplemente Funciones, y estas Funciones son las que reemplazan a las **APIs** que usan otros lenguajes como su base principal. Si colocamos el mouse sobre el Método.

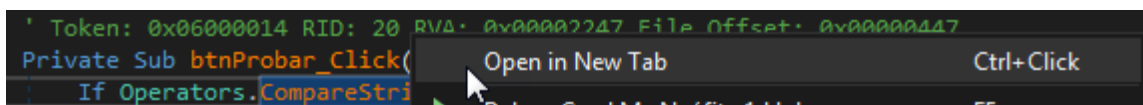


Neófito Reversando .NET [Entrega #1][LUISFECAB]

Tenemos información muy precisa del Método, lo que hace, sus parámetros y lo que retorna. Y es mucho mejor, si hacemos clic sobre cualquier Método, Función o Procedimiento podemos ir directamente a su ejecución.

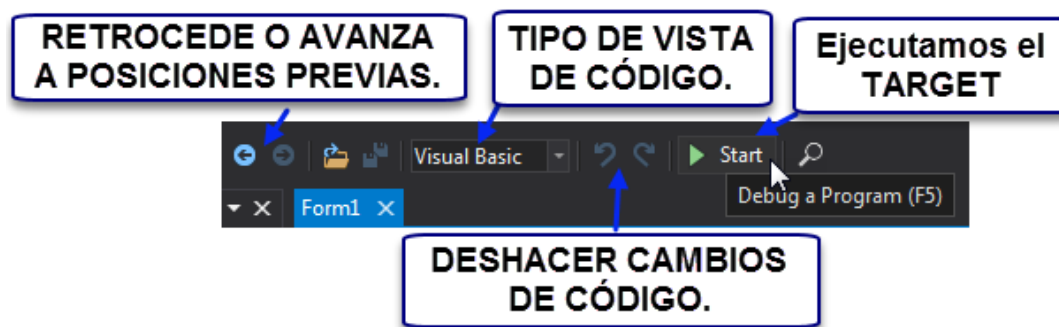


Como vemos esa Función está en **Microsoft.VisualBasic.dll**. De esta misma forma iremos a procedimientos que sean de nuestro interés. También podemos abrir los procedimientos en nuevas pestañas.

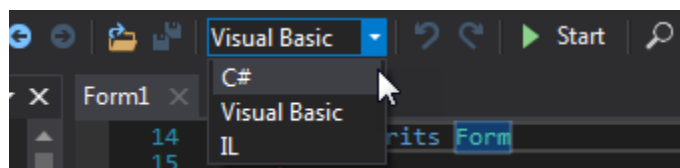


Esta opción es muy buena porque así tenemos todos los procedimientos y con mejor acceso a todos los códigos que sean de nuestro interés.

Explicaremos más opciones que tiene el <dnSpy> y que nos serán útiles a medida que avancemos.



En la barra de herramientas tenemos las opciones que son vitales para nosotros cuando estemos Reversando aplicaciones más complejas y no tan inocentes como nuestro **CrackMe**. Bueno, cuando me refiero a complejas no es para asustarnos, mientras no tengamos los **.NET** protegidos estos son más crackeables que otros programas codificados en otros lenguajes. Miremos las vistas del código.



Neófito Reversando .NET [Entrega #1][LUISFECAB]

Tenemos tres tipos de vista que nos permiten ver el código. Las vistas **C#** y **Visual Basic** son código de alto nivel que en teoría ya conocemos, aquí lo nuevo, como lo fue para mí, es la vista **IL**.

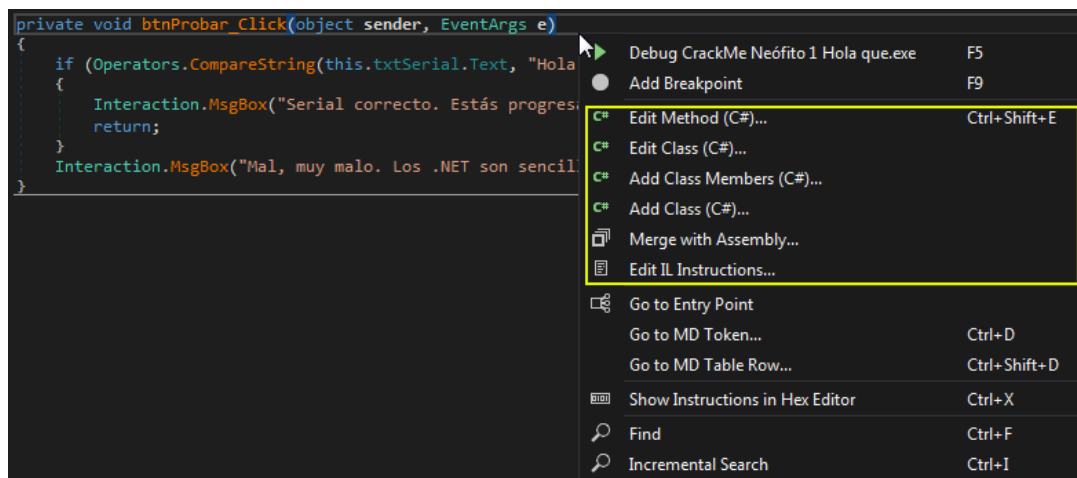
```
.method private
instance void btnProbar_Click (
    object sender,
    class [mscorlib]System.EventArgs e
) cil managed
{
    // Header Size: 1 byte
    // Code Size: 52 (0x34) bytes
    .maxstack 8

    /* 0x00000448 02          */ IL_0000: ldarg.0
    /* 0x00000449 6F1B00006   */ IL_0001: callvirt instance class [System.Windows.Forms]System.Windows.Forms.TextBox
    CrackMe_Neófito_1_Hola_que.Form1::get_txtSerial()
    /* 0x0000044E 6F400000A   */ IL_0006: callvirt instance string [System.Windows.Forms]
    System.Windows.Forms.TextBox::get_Text()
    /* 0x00000453 726B00070   */ IL_000B: ldstr      "Hola Mundo"
    /* 0x00000458 16          */ IL_0010: ldc.i4.0
    /* 0x00000459 28410000A   */ IL_0011: call      int32 [Microsoft.VisualBasic]
    Microsoft.VisualBasic.CompilerServices.Operators::CompareString(string, string, bool)
    /* 0x0000045E 2D0E       */ IL_0016: brtrue.s   IL_0026

    /* 0x00000460 728100070   */ IL_001B: ldstr      "Serial correcto. Estás progresando."
    /* 0x00000465 16          */ IL_001D: ldc.i4.0
    /* 0x00000466 14          */ IL_001E: ldnull
    /* 0x00000467 28420000A   */ IL_001F: call      valuetype [Microsoft.VisualBasic]Microsoft.VisualBasic.MsgBoxResult
    [Microsoft.VisualBasic]Microsoft.VisualBasic.Interaction::MsgBox(object, valuetype [Microsoft.VisualBasic]
    Microsoft.VisualBasic.MsgBoxStyle, object)
```

Este es el **Lenguaje Ensamblador IL**, que es el código en bajo nivel del **.NET**. También es llamado **CIL (Common Intermediate Language)**. A Medida que practiquemos con **.NET** iremos familiarizándonos con este.

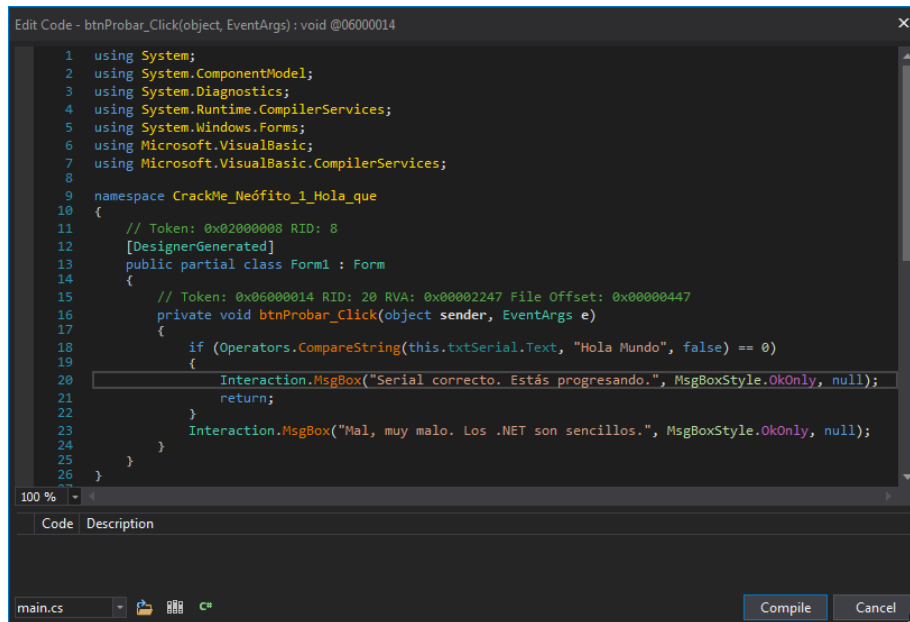
Ahora, si queremos hacer algún cambio al código de algún procedimiento. Con **<Click-Derecho>** se nos despliega un menú con muchas opciones.



En la imagen de arriba tenemos el procedimiento **btnProbar_Click**, y está con la vista de código **C#**. Nos enfocaremos en las opciones del **RECUADRO AMARILLO**. **<Edit Method (C#)...>** nos abrirá el editor de código mostrando solo el Método a editar. **<Edit Class (C#)...>** se mostrará todo el código del Módulo en donde nos encontramos. También podemos agregar nuevos Módulos con las opciones **<Add Class Members (C#)...>** y **<Add Class (C#)...>**, hasta este momento no he utilizado esas opciones para Crackear los pocos **.NET** que he hecho, solo con editar el código me ha sido suficiente. La opción **<Merge With Asembly...>** tampoco la he utilizado. La última opción **<Edit IL Instructions...>** pues nos sirve para editar código pero en **IL**.

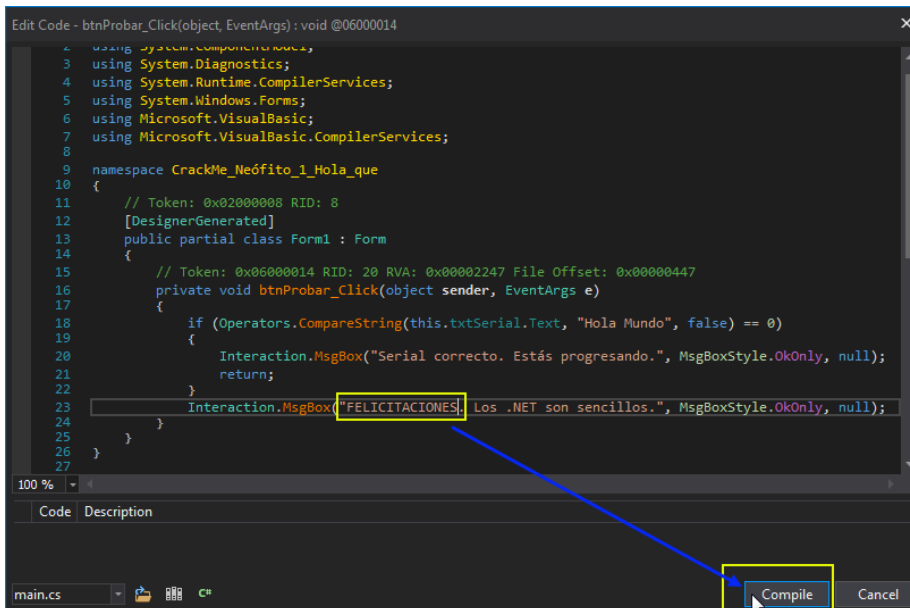
Neófito Reversando .NET [Entrega #1][LUISFECAB]

Bueno amigos para animar este escrito porque hasta yo me estoy pasmando de escribir y no hacer nada. Editemos el **btnProbar_Click**, para volver el "CHICO MALO" en un "CHICO BUENO". Ya sabemos <Click Derecho->Edit Method (C#)...



```
1 using System;
2 using System.ComponentModel;
3 using System.Diagnostics;
4 using System.Runtime.CompilerServices;
5 using System.Windows.Forms;
6 using Microsoft.VisualBasic;
7 using Microsoft.VisualBasic.CompilerServices;
8
9 namespace CrackMe_Neófito_1_Hola_que
10 {
11     // Token: 0x02000008 RID: 8
12     [DesignerGenerated]
13     public partial class Form1 : Form
14     {
15         // Token: 0x06000014 RID: 20 RVA: 0x00002247 File Offset: 0x00000447
16         private void btnProbar_Click(object sender, EventArgs e)
17         {
18             if (Operators.CompareString(this.txtSerial.Text, "Hola Mundo", false) == 0)
19             {
20                 Interaction.MsgBox("Serial correcto. Estás progresando.", MsgBoxStyle.OkOnly, null);
21                 return;
22             }
23             Interaction.MsgBox("Mal, muy malo. Los .NET son sencillos.", MsgBoxStyle.OkOnly, null);
24         }
25     }
26 }
27
```

Ahí tenemos la ventana para editar nuestro código. Cambiaremos la parte del mensaje que nos dice "Mal, muy malo" por "FELICITACIONES".



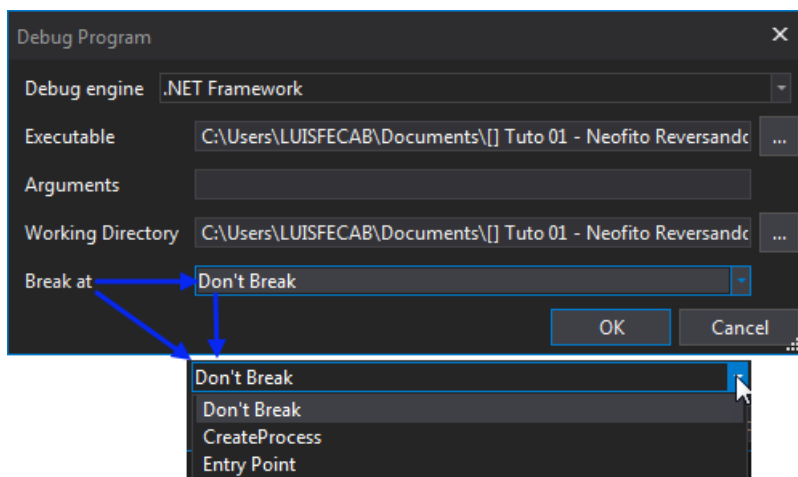
```
1 using System;
2 using System.ComponentModel;
3 using System.Diagnostics;
4 using System.Runtime.CompilerServices;
5 using System.Windows.Forms;
6 using Microsoft.VisualBasic;
7 using Microsoft.VisualBasic.CompilerServices;
8
9 namespace CrackMe_Neófito_1_Hola_que
10 {
11     // Token: 0x02000008 RID: 8
12     [DesignerGenerated]
13     public partial class Form1 : Form
14     {
15         // Token: 0x06000014 RID: 20 RVA: 0x00002247 File Offset: 0x00000447
16         private void btnProbar_Click(object sender, EventArgs e)
17         {
18             if (Operators.CompareString(this.txtSerial.Text, "Hola Mundo", false) == 0)
19             {
20                 Interaction.MsgBox("Serial correcto. Estás progresando.", MsgBoxStyle.OkOnly, null);
21                 return;
22             }
23             Interaction.MsgBox("FELICITACIONES. Los .NET son sencillos.", MsgBoxStyle.OkOnly, null);
24         }
25     }
26 }
27
```

Sencillamente borramos lo que no queremos y agregamos nuestras "FELICITACIONES", nada del otro mundo como si estuviéramos programando. Luego compilamos con "Compile", y si todo se compila bien tendremos nuestro código ya editado en la sección **CODE** que yo llamo **DESAMBLADO**; digo esto porque a veces he editado código y me compila mal, y cuando esto sucede en la sección **CODE** lo que era nuestro código original desaparece. Miremos nuestros cambios.

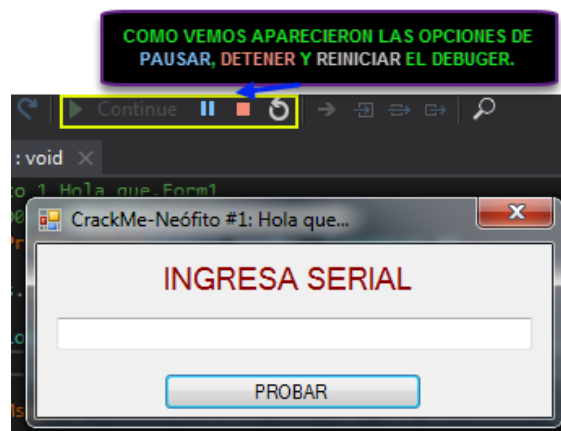
Neófito Reversando .NET [Entrega #1][LUISFECAB]

```
btnProbar_Click(object, EventArgs) : void X
1 // CrackMe_Neófito_1_Hola_que.Form1
2 // Token: 0x06000014 RID: 20
3 private void btnProbar_Click(object sender, EventArgs e)
4 {
5     if (Operators.CompareString(this.txtSerial.Text, "Hola Mundo", false) == 0)
6     {
7         Interaction.MsgBox("Serial correcto. Estás progresando.", MsgBoxStyle.OkOnly, null);
8         return;
9     }
10    Interaction.MsgBox("FELICITACIONES. Los .NET son sencillos.", MsgBoxStyle.OkOnly, null);
11 }
```

Tome pa' que lleve, ahora ya son dos "**CHICOS BUENOS**". Iniciemos el **CrackMe** con **Start** o con <F5>. Y le ponemos un **SERIAL** incorrecto, yo pongo mi favorito "**MUYDIFICIL**".

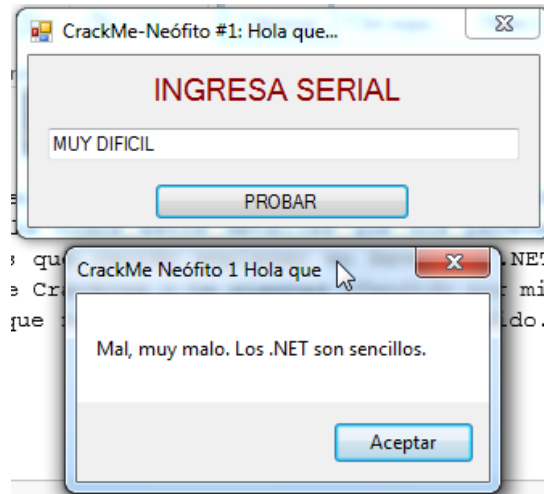


Antes nos aparece para configurar el proceso de Debugear el **TARGET**, aquí lo que debemos escoger es si queremos que se detenga o no; nosotros escogeremos que no pare, así que lo dejamos en <Don't Break" e iniciamos con "OK".

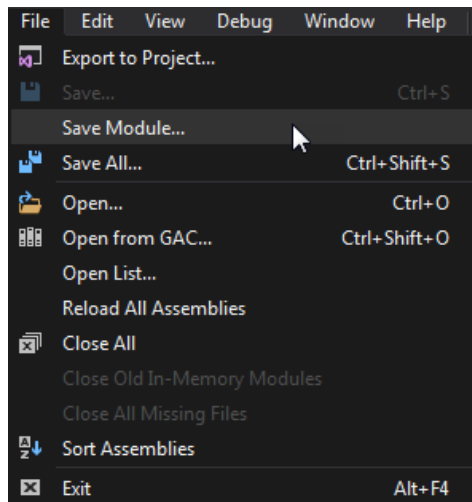


Bien, parece que ahora si voy a ingresar mi Serial "**MUYDIFICIL**", es que hay que dejar lo más claro posible todos estos detalles que nos parecen obvios pero esto está dirigido a aquellos que recién empiezan en Reversar **.NET**. Mi querido lector si tienes buenas bases de Cracking y te sientes ofendido por mis obviedades te pido disculpas pero recuera que nadie de nosotros nació aprendido. Bueno, a probar el **SERIAL**.

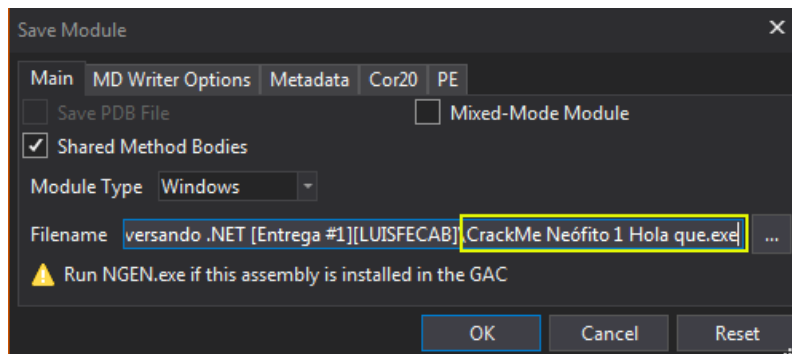
Neófito Reversando .NET [Entrega #1][LUISFECAB]



He tomado la captura sobre este tutorial, estamos escribiéndolo sobre la marcha. "Chanfle", nos apareció el **CHICO MALO**, pero si hicimos el cambio, yo esperaba que me dijera "**FELIITACIONES**". Pues resulta que si venimos acostumbrados a como ocurre con el <OllYDBG> o el <x64DBG> que si cambiamos las instrucciones y ejecutamos, estas son ejecutadas pero aquí con el <dnSpy> no; aquí debemos guardar los cambios primero y luego ejecutar de nuevo.

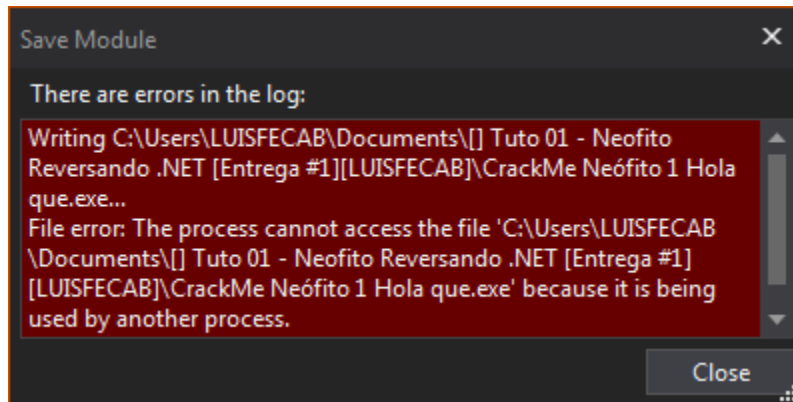


Hay varias opciones que veremos a medida que avancemos. Para nosotros guardar los cambios vamos a <**File->Save Module...**>.

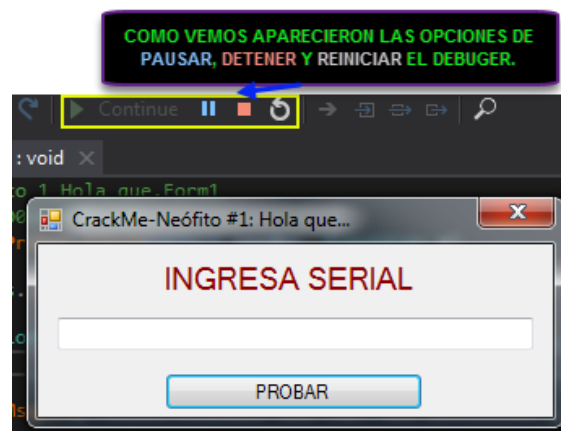


Neófito Reversando .NET [Entrega #1][LUISFECAB]

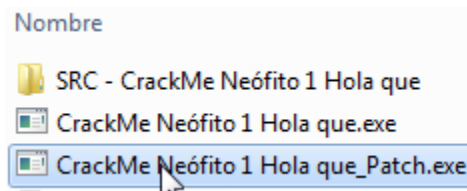
Tenemos gran cantidad de opciones cuando queramos guardar los cambios que hagamos, aquí nosotros no cambiaremos nada, lo guardaremos tal cual. Bueno, debemos guardar el Módulo, o sea el **CrackMe** con otro nombre. ¿Pero y que pasa si lo guardamos con el mismo nombre?, hagámoslo, solo por curiosidad.



Tenemos un error. Yo siempre me equivoco con esto, como queremos sobrescribir el **TARGET**, pero no podemos debido a que nosotros tenemos ejecutado este mismo desde el <dnSpy>. Recordemos que venimos desde esta ejecución.

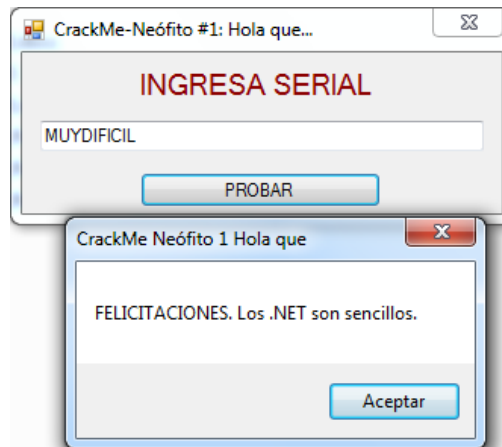


Con detener la ejecución ya podríamos guardarlo sobrescribiendo el original. Cierro ese mensaje de error e intento guardarlo de nuevo pero con otro nombre <**CrackMe Neófito 1 Hola que_Patch.exe**>. Ejecutémoslo directamente y lo probamos.

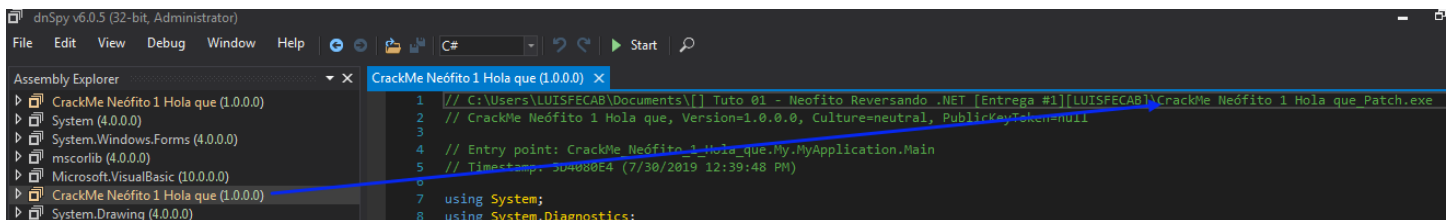


Meteremos nuestro **SERIAL** "MUYDIFICIL" y veremos si estábamos en lo cierto, que con el <dnSpy> debemos guardar los cambios primero para ver si funcionan.

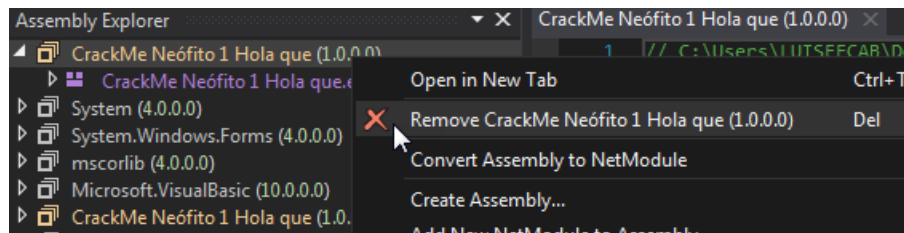
Neófito Reversando .NET [Entrega #1][LUISFECAB]



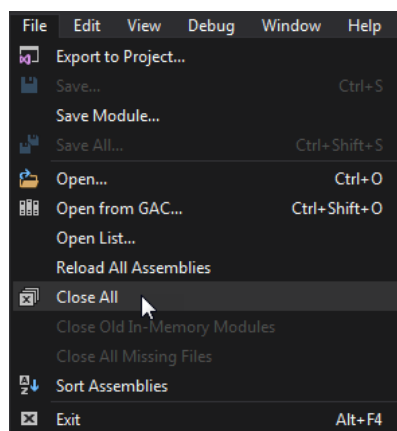
Perfecto, ya tendremos un bonito mensaje cada vez que ingresemos un **SERIAL** chueco. Abramos este **CrackMe** Parcheado con otro <dnSpy>.



Tenemos los dos **CrackMe** y las **DLLs** del sistema que se cargaron durante la ejecución. Es muy aconsejable mantener lo más limpio posible el **ASSEMBLY EXPLORER** para evitar confundirnos. Recordemos que tengo dos <dnSpy> abiertos, así que cierro el del **CrackMe** original.

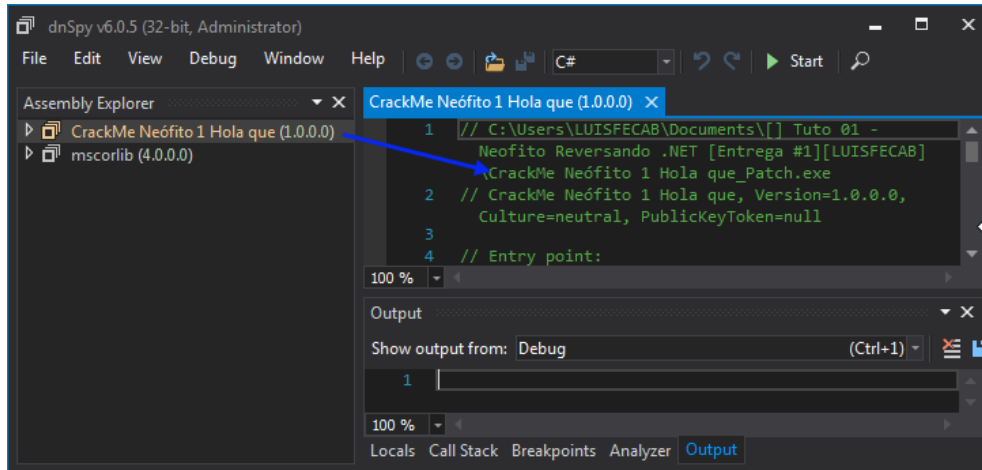


Seleccionando el elemento y <Click-Derecho> podemos remover el elemento o con la tecla <DELETE> O <SUPRIMIR> que son las mismas solo que depende del idioma.

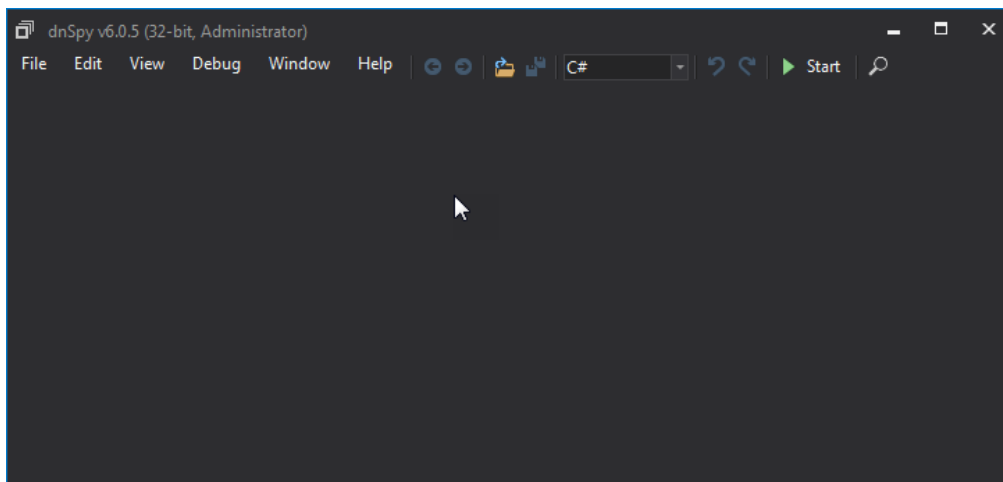


Neófito Reversando .NET [Entrega #1][LUISFECAB]

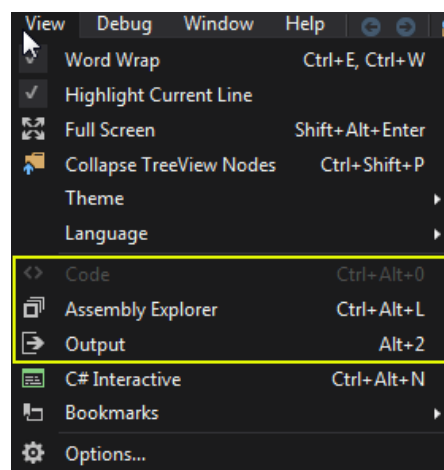
Si queremos cerrar todos los elementos para que la cosa nos rinda más, lo podemos hacer desde <**File->Close All**>. Como cerré todo, vuelvo y cargo el **CrackMe** parcheado.



Como andamos cerrando cosas, también podemos cerrar las ventanas principales que yo he llamado secciones.



Nos quedó limpio, no tenemos sobre qué trabajar. Podemos cargar de nuevo todo lo que queramos desde <**View**>.

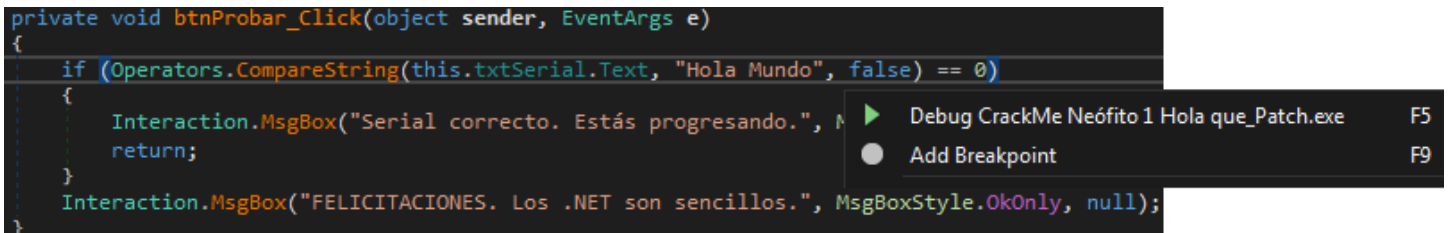


Neófito Reversando .NET [Entrega #1][LUISFECAB]

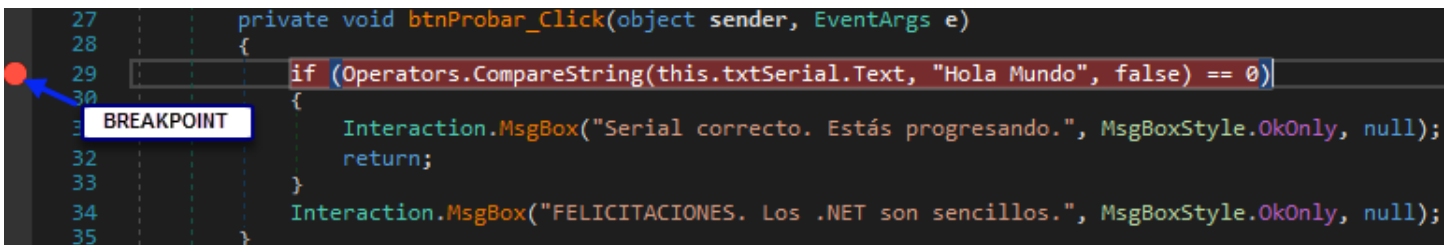
En el **RECUADRO AMARILLO** tenemos las tres secciones que venimos utilizando. Como vemos la sección **<Code>** está deshabilitada y esto se debe a que debemos tener algún ensamblado seleccionado para tener algo de código que mostrar. Esto de poder cerrar alguna sección principal es muy bueno en el caso que necesitemos más espacio para visualizar otras cosas que lo requieran más.

Ahora entraremos a colocar los **<BREAKPOINTS>**, cosa muy importante para poder detenernos en los lugares de nuestro interés. Vallamos de nuevo a **btnProbar_Click**.

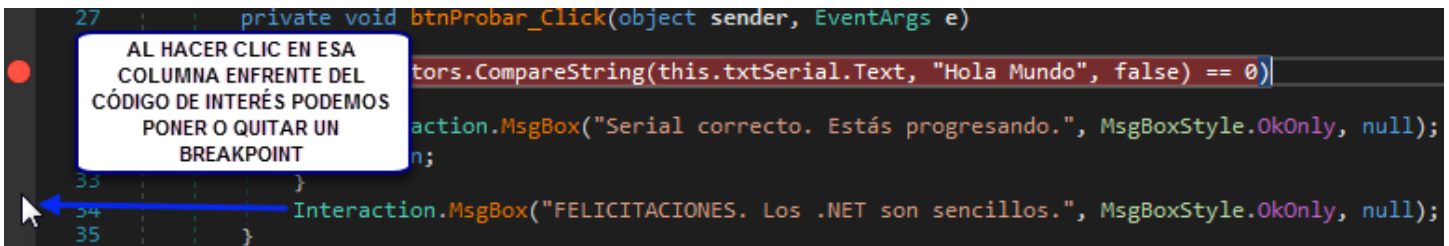
```
private void btnProbar_Click(object sender, EventArgs e)
{
    if (Operators.CompareString(this.txtSerial.Text, "Hola Mundo", false) == 0)
    {
        Interaction.MsgBox("Serial correcto. Estás progresando.", null);
        return;
    }
    Interaction.MsgBox("FELICITACIONES. Los .NET son sencillos.", MsgBoxStyle.OkOnly, null);
}
```



Nos posicionamos sobre la línea de código en donde queremos colocar el **<BREAKPOINT>** y con **<Click-Derecho->Add Breakpoint>** o con su tecla de acción rápida **<F9>**.

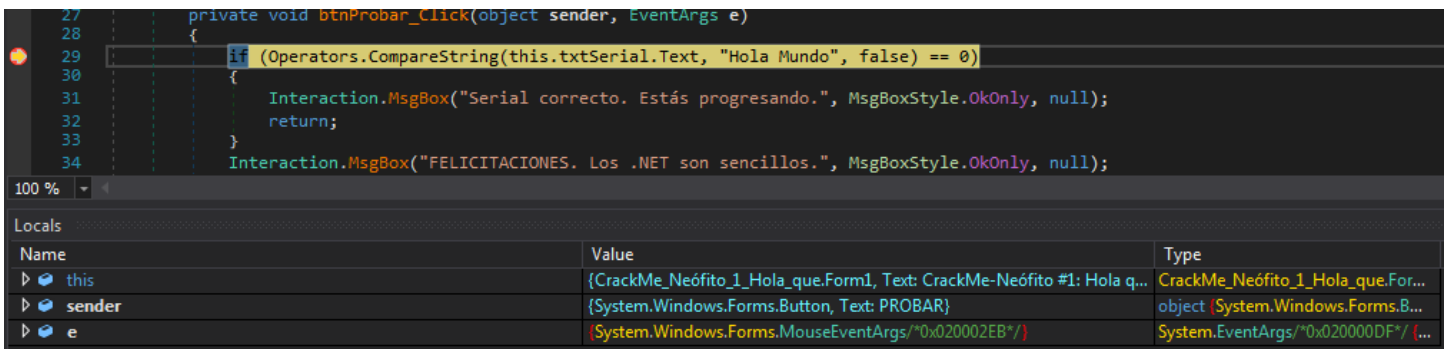


Otra manera, la cual me gusta mucho, es hacer **<Click>** sobre la columna donde queremos ponerlo.

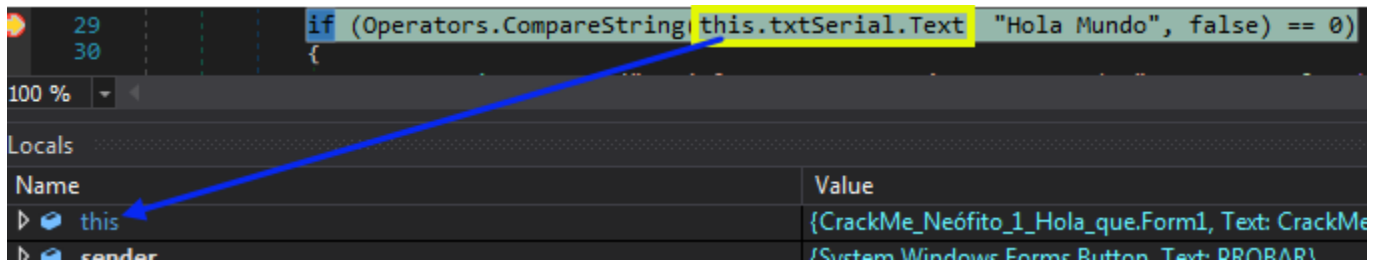


Para mi esa es la mejor manera de colocar un **<BREAKPOINT>**. Bien, ahora ejecutamos con **<F5>**. El **CrackMe** se abrirá y debemos ingresar un **SERIAL**, y ahí **<Click>** en **"PROBAR"** para detenernos en nuestro **<BREAKPOINT>**. Les recuerdo yo ingresé **"MUYDIFICIL"** como **SERIAL**. Aquí ya entraremos a ver y analizar los diferentes parámetros y valores que carga el **<dnSpy>** mientras Debuguea. Aquí podemos cambiar estos valores para que tomemos un camino u otro de ejecución de código.

Neófito Reversando .NET [Entrega #1][LUISFECAB]



Nos detuvimos en la comparación con el **if**. Vemos que tenemos la ventana "**Locals**", en esta tenemos toda la información que se origina mientras se ejecuta, tenemos todas las variables, información de objetos, de controles...etc. En este momento en "**Locals**" tenemos tres grupos principales de valores cargados, los dos últimos son "**sender**" y "**e**", los cuales son los parámetros de **btnProbar_Click(object sender, EventArgs e)**; lo cual de una vez descartamos. Ahora miremos con más atención esto.



Como sabemos en **this.txtSerial.Text** se encuentra nuestro **SERIAL** ingresado. Bien, y qué viene siendo **this**, pues yo diría que ahí tenemos toda la información del **TARGET** en este caso nuestro **CrackMe**, ahí está los detalles del **Form1**, los controles, las variables y sus propiedades. Entonces, dentro de **this** encontraremos nuestro **SERIAL** ingresado, que debe estar contenido en el control **txtSerial.Text**. La cuestión es buscarlo.

txtSerial	{System.Windows.Forms.TextBox, Text: MUYDIFICIL}
AcceptsReturn	false
AcceptsTab	false
AccessibilityObject	{ControlAccessibleObject: Owner = System.Windows...
AccessibleDefaultActionDescription	null

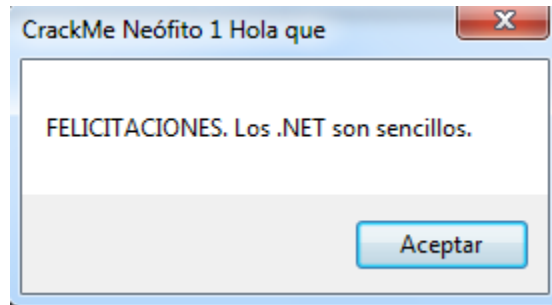
Debemos bajar sin miedo hasta llegar a los elementos que empiezan por "**T**" y como sabemos el "(name)" de nuestro "**TextBox**" es **txtSerial**. Paso seguido es buscar dentro del **txtSerial** en donde se guardó nuestro **SERIAL** ingresado "**MUYDIFICIL**".

Tag	null
Text	"MUYDIFICIL"
TextAlign	Left
TextLength	10
Top	45

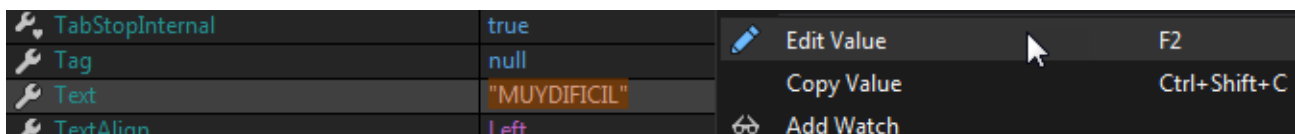
En **Text** se guarda nuestro **SERIAL** ingresado. Ahí podemos cambiarlo por cualquier otra String. La idea de esto es saber cómo cambiar algún parámetro para obtener un

Neófito Reversando .NET [Entrega #1][LUISFECAB]

resultado diferente. Como sabemos estamos con el **CrackMe** parcheado y si ejecutáramos como venimos tendríamos el siguiente mensaje.



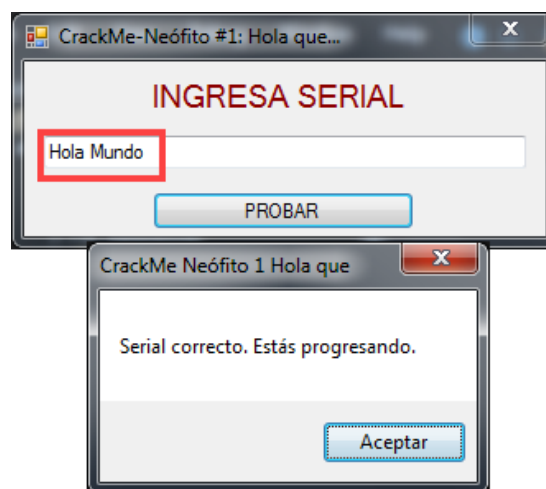
El de "**FELICITACIONES**" que era antes un "**CHICO MALO**", pero si cambiamos el "**MUYDIFICIL**" por el "**Hola Mundo**" deberíamos obtener el original "**CHICO BUENO**". Hagamos el cambio.



<Clic-Derecho>**Edit Value**>, tecla de acción rápida <F2> o con <Doble-Clic> podemos editar el valor. Aclaremos, que no es que podamos editar todo lo que queramos, solamente donde se nos sea permitido.



Ahí hicimos el cambio. Continuemos con la ejecución del **CrackMe** para ver el resultado.

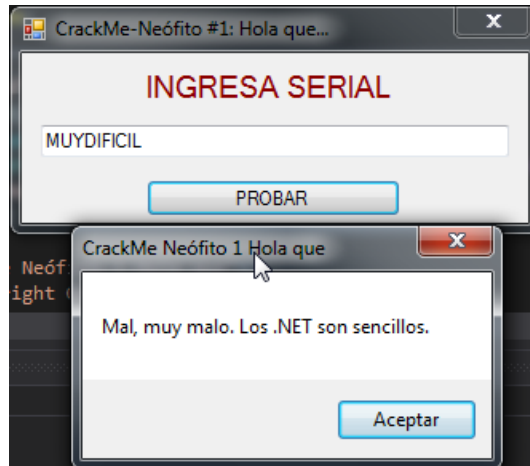


Como ven obtuvimos el "**CHICO BUENO**" original y cambió hasta el **SERIAL** ingresado, recordemos que era "**MUYDIFICIL**".

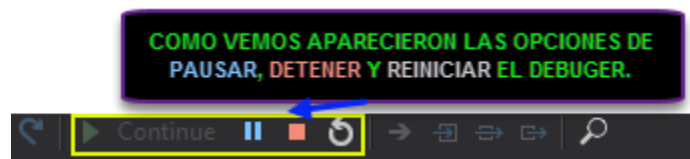
Recordando un poco, siempre decimos que debemos hallar la "**ZONA CALIENTE**" porque ahí es donde podemos saber la rutina de Activación de un programa y poder Crackearlo

Neófito Reversando .NET [Entrega #1][LUISFECAB]

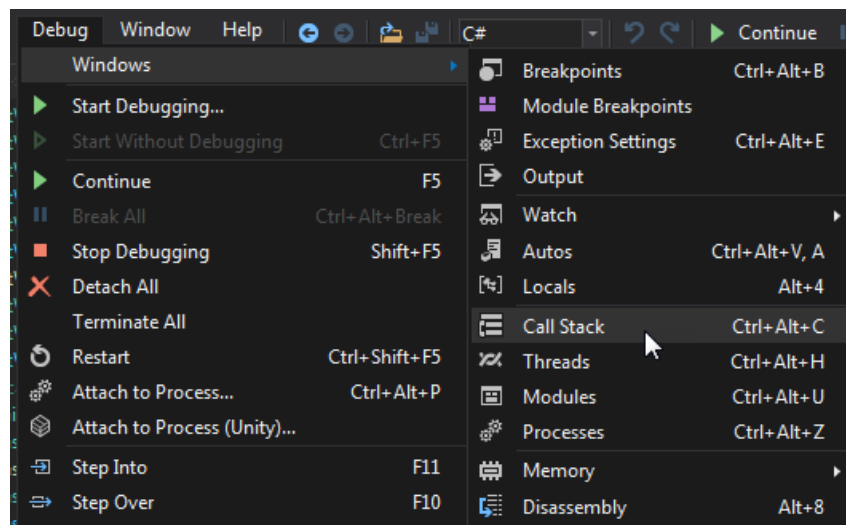
ahí. Para hallar esta "ZONA CALIENTE" en un .NET podemos utilizar el truco del **CALL STACK** que hemos conocido, recordemos que mientras la aplicación ejecuta procedimientos estos los va guardando a medida que se van ejecutando estos. Si uno pausa la ejecución entonces podemos ver el listado de todos los procedimientos que se están ejecutando hasta ese momento. Estos los veremos en la ventana **CALL STACK**. Cerremos este <dnSpy> que tiene el CrackMe Parcheado y abrimos uno con el CrackMe original.



Le metemos un **SERIAL** chueco y lo probamos para obtener al "CHICO MALO". Estando parados con ese mensaje pausamos la ejecución. Dejemos la imagen de nuevo.

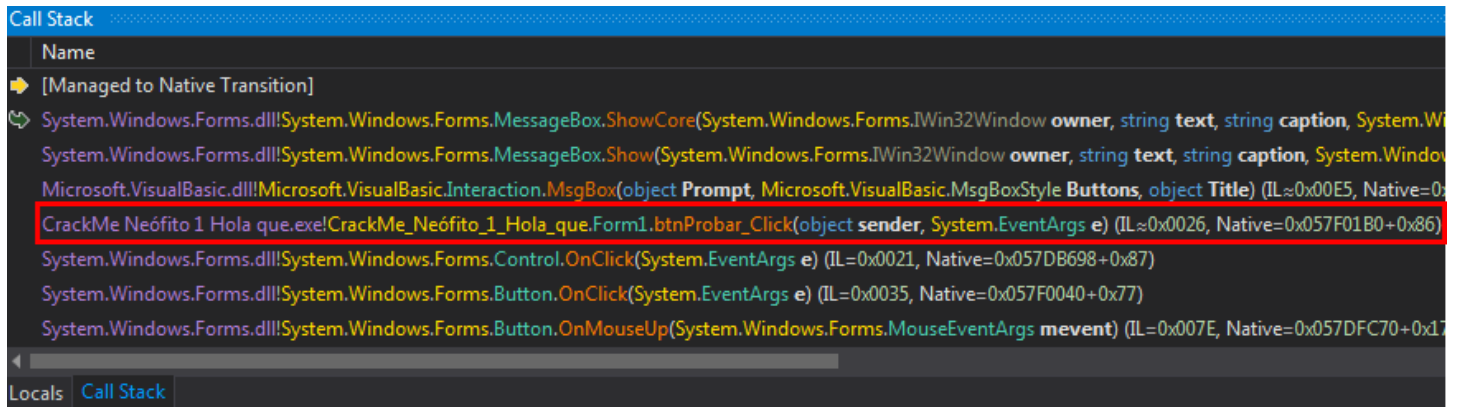


Una vez pausemos nos detendremos en la Clase **MessageBox**, que pertenece al sistema. Lo importante aquí es ver en la ventana **CALL STACK**. Deberían verla donde está la ventana "Locals"; si no la encuentran como es mi caso, la abrimos desde <Debug->Windows->Call Stack>.



Neófito Reversando .NET [Entrega #1][LUISFECAB]

Como vemos tenemos muchas ventanas para muchas cosas, tenemos para ver los **<BREAKPOINTS>** que ya aprendimos a colocar. Miremos que tenemos en la ventana **CALL STACK**.



```
Call Stack
Name
[Managed to Native Transition]
System.Windows.Forms.dll!System.Windows.Forms.MessageBox.ShowCore(System.Windows.Forms.IWin32Window owner, string text, string caption, System.W
System.Windows.Forms.dll!System.Windows.Forms.MessageBox.Show(System.Windows.Forms.IWin32Window owner, string text, string caption, System.Wind
Microsoft.VisualBasic.dll!Microsoft.VisualBasic.Interaction.MsgBox(object Prompt, Microsoft.VisualBasic.MsgBoxStyle Buttons, object Title) (IL=0x00E5, Native=0
CrackMe Neófito 1 Hola que.exe!CrackMe_Neófito_1_Hola_que.Form1.btnProbar_Click(object sender, System.EventArgs e) (IL=0x0026, Native=0x057F01B0+0x86)
System.Windows.Forms.dll!System.Windows.Forms.Control.OnClick(System.EventArgs e) (IL=0x0021, Native=0x057DB698+0x87)
System.Windows.Forms.dll!System.Windows.Forms.Button.OnClick(System.EventArgs e) (IL=0x0035, Native=0x057F0040+0x77)
System.Windows.Forms.dll!System.Windows.Forms.Button.OnMouseUp(System.Windows.Forms.MouseEventArgs mevent) (IL=0x007E, Native=0x057DFC70+0x17)
```

Como vemos todo empieza desde nuestro **CrackMe**. La realidad es que los programas normalmente hacen muchas vueltas para validar su estado de Activación, y si utilizamos el truco del **CALL STACK** podemos llegar fácilmente a la "**ZONA CALIENTE**".

Listo, creo que hasta aquí es suficiente como para tener un inicio.

PARA TERMINAR

Creo que con lo que hemos escrito hasta aquí es un buen principio. He tratado de explicar recordando de cómo fui aprendiendo a hacer uso del <dnSpy>. Poco a poco iremos aplicando estas cosas aprendidas en programas de verdad.

El <dnSpy> es una herramienta muy buena, y con que aprendamos a usarlo lo mejor posible, podremos extrapolar lo aprendido aquí a otras herramientas similares a esta.

Espero pronto escribir la segunda entrega, veré por donde me enfoco.

@LUISFECAB