

[YouTubeByClick v2.2.87 (.NET)(Crack-Patch)(dnSpy v5.0.7)]



Software	YouTubeByClick v2.2.87
WEB	http://www.youtubebyclick.com/
Protección	SERIAL - VALIDACIÓN POR INTERNET
Herramientas	<p>Windows 7 Home Premium SP1 - Build 7601 x86 Bits (SO donde trabajamos)</p> <p>dnSpy v5.0.7 (.NET assembly editor, decompiler, and debugger https://github.com/0xd4d/dnSpy/)</p> <p>ExeinfoPe v0.0.5.0 2018.03.31 MSIL OpCode Table v1.0 dUP2 Diablo's Universal Patcher v2.26</p> <p>DESCARGAR HERRAMIENTAS</p> <p>DESCARGAR TUTO+ARCHIVOS</p>
SOLUCIÓN	CRACK - PATCH
AUTOR	LUISFECAB
RELEASE	NOVIEMBRE 20 DE 2018 [TUTORIAL 013]

INTRODUCCIÓN

En este tutorial vamos a aplicar lo aprendido en nuestro anterior escrito, [1671](#), que para mí fue una maravilla, ahí pulí un poco más mis conocimientos que se traducen en **EXPERIENCIA**. Lo hicimos con ayuda de mis amigos de **PeruCrackerS**, pero que también hacen parte de la gran familia de **CrackLatinoS**; ya saben al final somos uno solo, personas que nos apasiona el **Reversing** y para mí el **Cracking** principalmente.

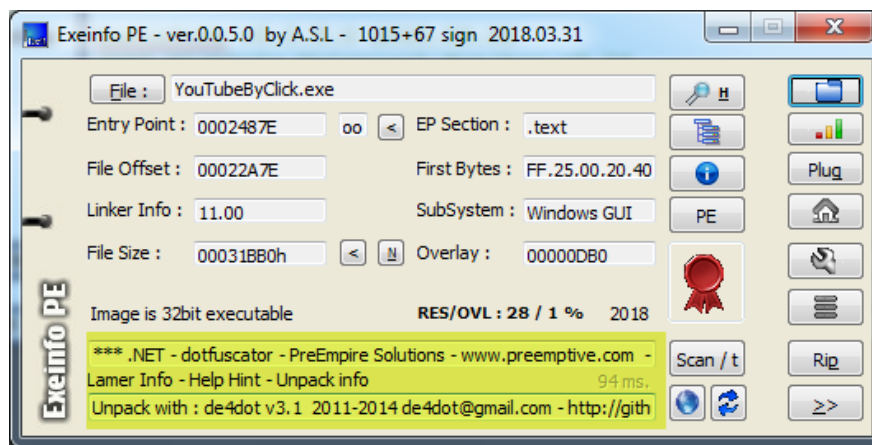
El programa que vamos a ver es el <YouTubeByClick v2.2.87>. Este programa lo intenté crackear muchos meses atrás y no pude, "*no naba pie con bola*", lo revisaba en ese entonces con el <dnSpy v3.0.7> y los cambios que intentaba no me cuajaban solo obtenía puros errores de compilación. Yo pensaba que el <dnSpy v3.0.7> ya estaba desactualizado y que a lo mejor no desesamblaba la versión del Framework del **.NET** pero con lo que pude aprender ahora creo más bien que el del problema era yo. Ahora lo veo tan claro y me pareció relativamente más fácil, como que todo lo entendía a la primera en esta oportunidad.

Bueno, el < YouTubeByClick v2.2.87> estaba condenado a caer, no pensaba que fuera tan pronto; a buena hora **Dani** puso ese **Target** en **.NET** que fue la llave a la puerta de la solución. Haremos el tuto sin las explicaciones básicas, es que ya me está pasando como a la mayoría cuando se hacen varios tutos, uno se va cansando de repetir los mismos pasos básicos pero eso sí, trataré de la mejor forma posible explicar la lógica del por qué lo hacemos de esa forma.

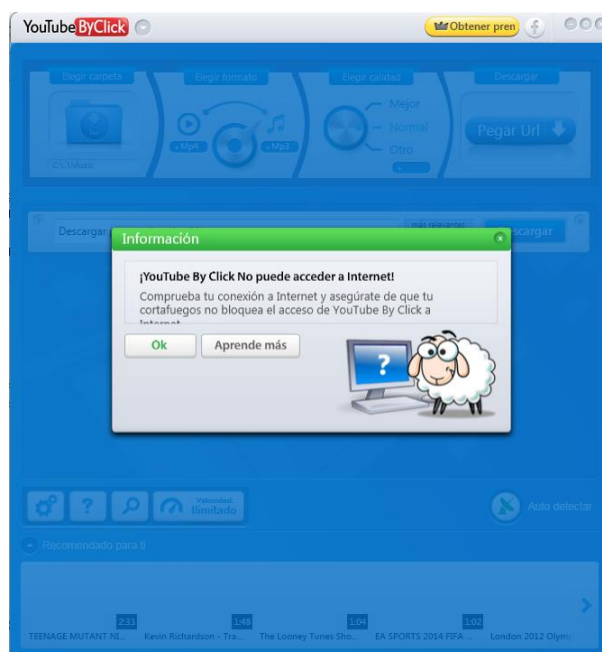
Saludos a todos y empezamos.

ANÁLISIS INICAL

Si tienes una porquería de Internet como el mío puedes desesperarte porque el < **YouTubeByClick v2.2.87** > lo debemos analizar estando conectados para saber por dónde va y poder ver que nuestros cambios son los correctos. Yo sufrí mucho por eso, pero mientras tengas un Internet estable no tendrás inconvenientes. **¿Y cómo sé que es con el Internet conectado?**, pues a medida que tu vayas traceando y reversando un programa les va pillando las marrullas de lo que hace, ya ustedes saben eso mis avezados lectores, conoce a tu enemigo y búscale sus debilidades, a lo **-El Arte de la guerra-**. Bien, la miradita de costumbre; utilizaremos el < **ExeinfoPe v0.0.5.0 2018.03.31** >.

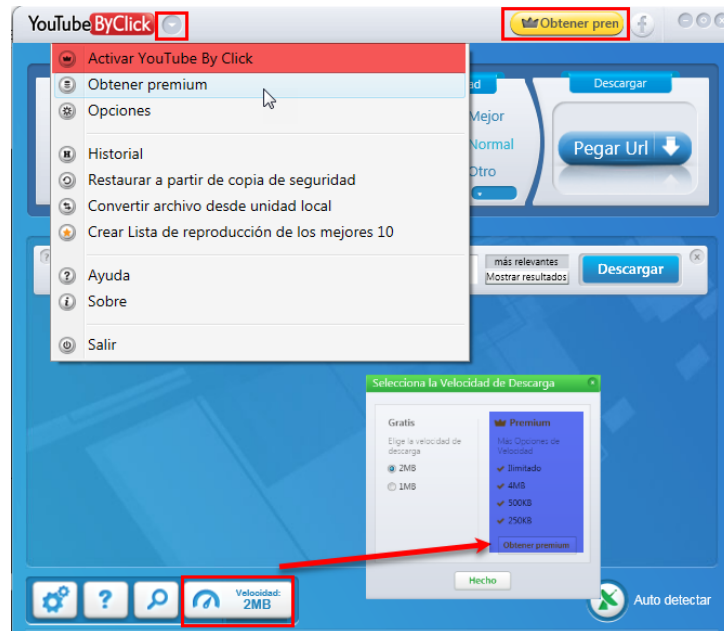


Es un **.NET** y disque está obfuscado y nos recomienda usar el < **de4dot v3.1** >. Cuando lo carguemos en nuestro < **dnSpy v5.0.7** > miraremos si es verdad. Ya es hora de abrir el < **YouTubeByClick v2.2.87** > para ir conociéndonos más de cerca. Lo ejecutamos desconectados de Internet, si lo sé, dije que debíamos estar conectados a Internet pero esto es el **ANÁLISIS INICAL** y aquí debemos ver las diferentes opciones.



[YouTubeByClick v2.2.87 (.NET)(Crack-Patch)(dnSpy v5.0.7)]

Carga y nos muestra una **NAG** que nos informa que no hay Internet. Muy lógico, si es una aplicación para descargar videos de Internet, pues lo mínimo que debemos estar es conectados porque así que gracia, **-si me vas a crackear para qué me usas sin Internet-**, nos está diciendo, y yo le respondo es que quiero ver unas cositas de esta forma. Aceptamos la **NAG** con "Ok", y se nos carga el programa.



Rastros de que no estamos **Full** y las restricciones son las velocidades de descarga. El programa te da un día **TRIAL**, después te pide activarlo para poder usarlo. Bueno, no importa para eso estamos escribiendo este tuto para dejarlo sin restricciones. Lo que quería ver es determinar cómo se comparta si metemos un **Serial** de activación en la opción resaltada en **ROJO**.



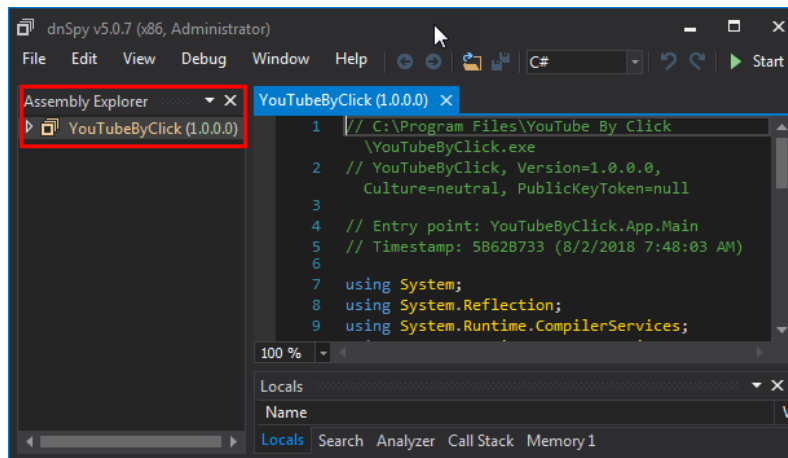
Ahí está, vuelve y nos recuerda que no hay Internet, no dice nada referente a nuestro **Serial**, le importa muy poco, y esto es prueba de que nuestro **Serial** es validado por Internet, entonces aquí no vale sacar el **Serial**, de dónde lo vamos a sacar si el programa no hace nada con él.

[YouTubeByClick v2.2.87 (.NET)(Crack-Patch)(dnSpy v5.0.7)]

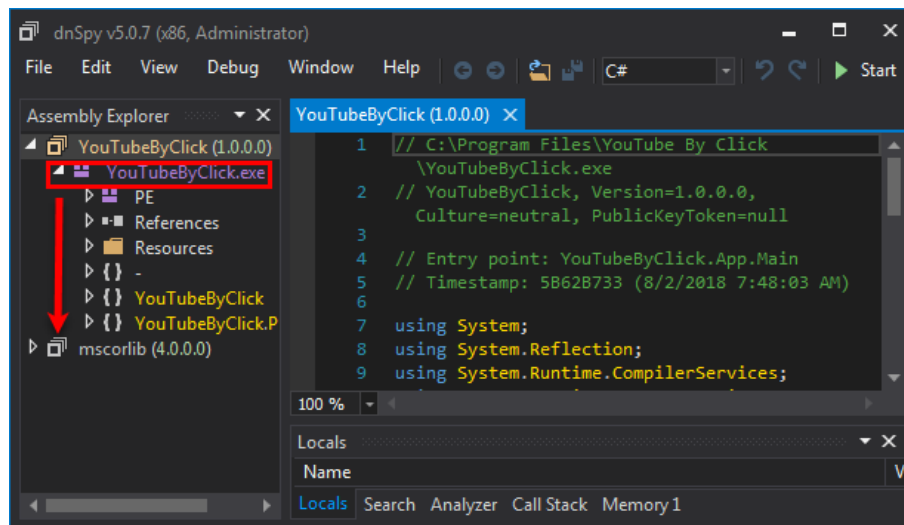
Copio y pego para recordar. **DavicoRm** ha dicho varias veces, palabras más, palabras menos, *-en la actualidad, con esto de que ya es tan común de que los dispositivos estén conectados a Internet, los programadores han aprovechado para agregar validaciones de activación por Internet haciendo necesario parchear el programa para evitar eso, con tener un serial ya no es suficiente-*, mucha razón tienes **DavicoRm**, nosotros aquí estamos frente a uno de esos casos, entonces para qué atacarle por el **Serial** que será comprobado vía Internet, si podemos derrotarlo desde adentro haciéndolo pensar que ya está activado.

Ya vimos lo queríamos y que la solución es un **Crack** y hacer un **Patch** donde busque y reemplace la "**ZONA CALIENTE**".

AL ATAQUE

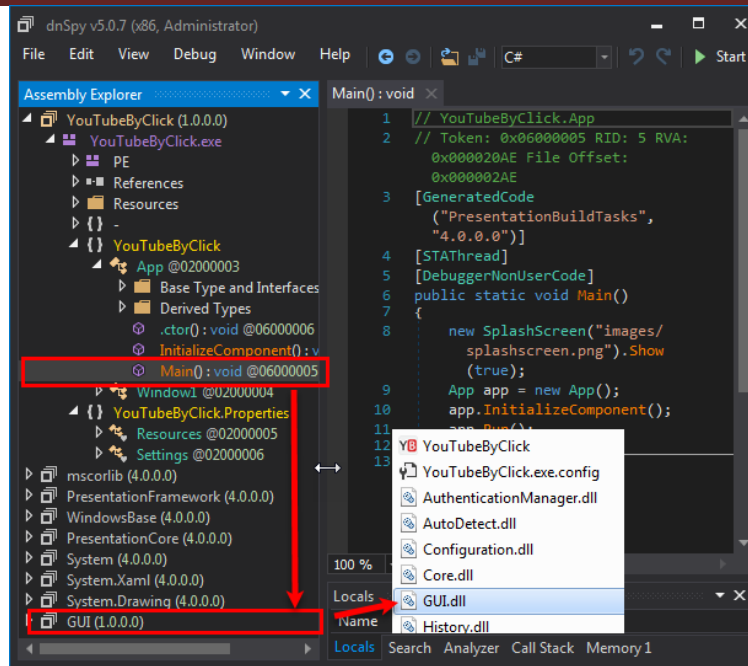


Ahí lo cargamos en nuestro <dnSpy v5.0.7>, cosas para notar y es que si observamos en la ventana "ASSEMBLY EXPLORER" resaltado con el **RECUADRO ROJO**, solo tenemos nuestro programa únicamente, no se ha cargado nada más. Luego veremos cómo se irán cargando más archivos que serán las **DLL** que usa el programa, las cargará a medida que vamos debugeando nuestro ejecutable y mientras se va descompilando le va echando mano a las librerías que usa nuestro código. Bueno, esto no es nada nuevo, en <OlllyDBG> hace lo mismo solo que aquí lo tenemos en una sola vista. exploremos nuestro ejecutable.

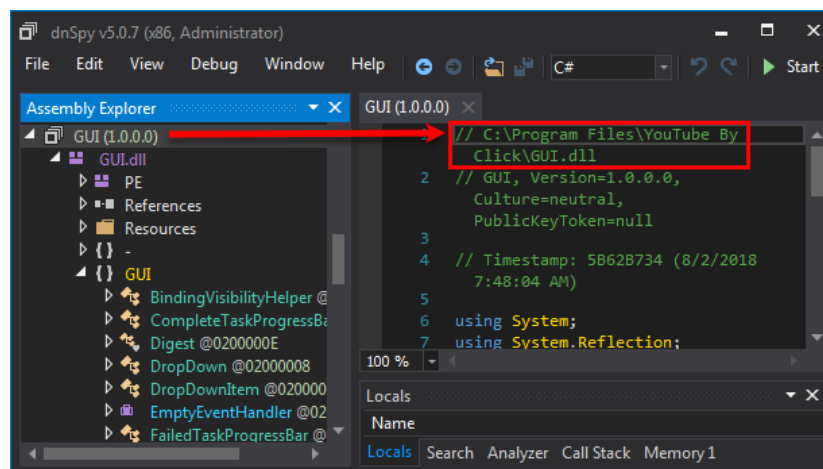


Listo, cuando exploramos el **YouTubeByClick.exe** se cargó la librería **mscorlib (4.0.0.0)** que es una **DLL** de nuestro sistema y así se seguirán cargando más a medida que lo exploramos profundamente. Voy a seguir explorando un poco más.

[YouTubeByClick v2.2.87 (.NET)(Crack-Patch)(dnSpy v5.0.7)]

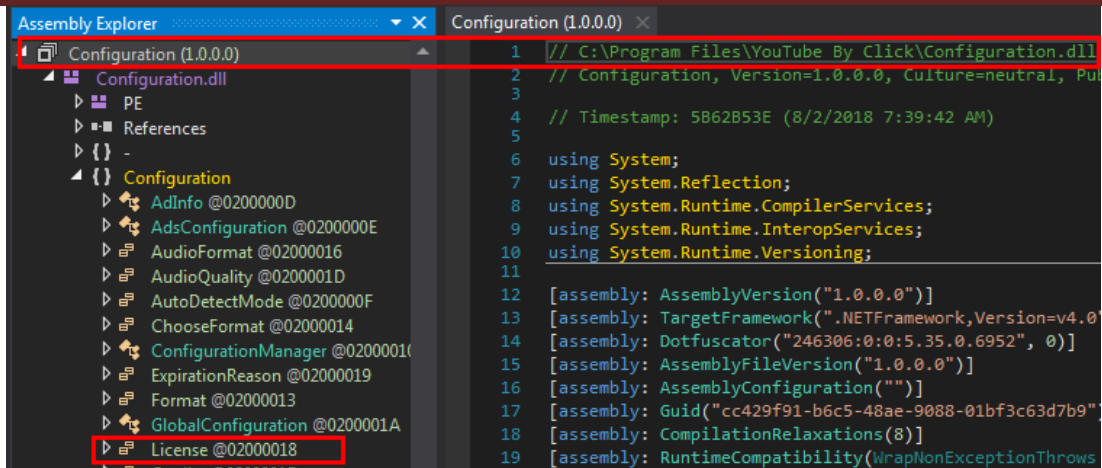


Esto es lo quería hacer notar, este programa no carga nada desde el **EXE**, prácticamente no hace nada, todo lo hacen sus propias **DLL's**. Uno puede sacar las **DLL's** propias de una aplicación **.NET** a medida que practicamos con ellos y vamos reconociendo las **DLL's** del sistema y cuáles no. Y cuando revisaba este **.NET** fue que caí en cuenta de eso. Si observamos la imagen de arriba ha cargado **GUI (1.0.0.0)**. Esta es una **DLL** propia del programa y ya con ese nombre nos dice todo, con ella va a cargar la interfaz gráfica de usuario. Seleccionémosla para ver qué trae.

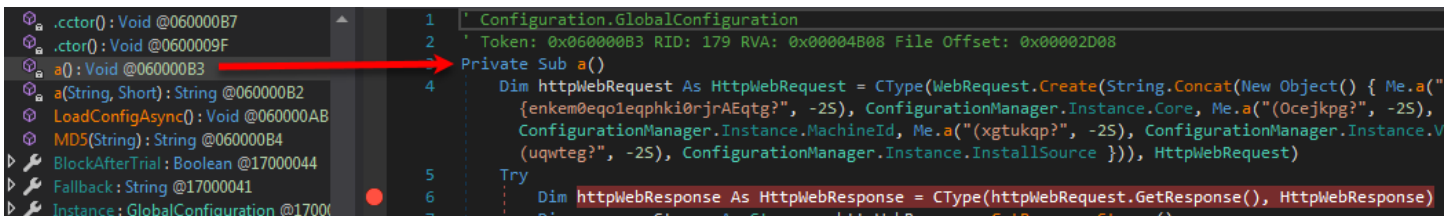


Solo es cuestión de seleccionar la **DLL** de nuestro interés y podemos obtener la información de dónde está ubicada, y como podemos observar esta hace parte de la instalación del programa. Como les dije esta carga la interfaz pero no es la que decide todo, si no es otra; aquí es saber buscar e ir revisando y si vemos nuevas librerías que se carguen y que ya sabemos no son del sistema pues las revisamos. Aclaro que este método para hallar nuestra "**ZONA CALIENTE**" no es el mejor porque nos toma más tiempo en localizar nuestros lugares de interés pero sirve para ir reconociendo y poder distinguir las **DLL's** del sistema. Mientras uno revisa las **DLL's** vemos que se cargan unas cuantas y una de esa es la **Configuration (1.0.0.0)**.

[YouTubeByClick v2.2.87 (.NET)(Crack-Patch)(dnSpy v5.0.7)]



Es una **DLL** del programa y esa es la "*mamá de los pollitos*", esta comprueba la **Licencia** y es aquí donde debemos hacer nuestros cambios para que al cargar el programa aparezca **Full**. ¿Y en dónde ocurre eso?



Esa rutina comprueba nuestra **Licencia**, y lo primero que va hacer es conectarse a Internet para validar el **Serial**. La captura de arriba muestra el inicio del procedimiento y podemos ver que se declara una consulta para Internet:

```
5 HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(string.Concat(new object[]
```

Es prácticamente lo mismo que uno hace al escribir una dirección web en un navegador, la dirección a nuestra consulta y está dada por la siguiente parte de esa instrucción.

```
7 this.a("jvvr<11nqike0{qwvwgdg{enkem0eqo1eqphki0rjrAEqtg?", -2),  
8 ConfigurationManager.Instance.Core,  
9 this.a("Ocejkgp?", -2),  
10 ConfigurationManager.Instance.MachineId,  
11 this.a("xgtukqp?", -2),  
12 ConfigurationManager.Instance.Version,  
13 this.a("wrfcvge?vtwg(uqwteg?", -2),  
14 ConfigurationManager.Instance.InstallSource
```

Podemos ver que esto será la que nos debe dar la dirección web, y por ahí no se ve nada parecido; resulta que está encriptada y es por eso que vemos esas **Strings** que no dicen nada y que hacen parte de los argumentos de la función `private string a(string A_0, short A_1)` que desencripta las **Strings**. Miremos la función con nuestra primera **String** `this.a("jvvr<11nqike0{qwvwgdg{enkem0eqo1eqphki0rjrAEqtg?", -2)` y veamos qué nos devuelve.

[YouTubeByClick v2.2.87 (.NET)(Crack-Patch)(dnSpy v5.0.7)]

```
197     private string a(string A_0, short A_1)
198     {
199         int num = Convert.ToInt32(char.MaxValue);
200         int num2 = Convert.ToInt32('\0');
201         char[] array = A_0.ToCharArray();
202         for (int i = 0; i < array.Length; i++)
203         {
204             int num3 = Convert.ToInt32(array[i]) + (int)A_1;
205             if (num3 > num)
206             {
207                 num3 -= num;
208             }
209             else if (num3 < num2)
210             {
211                 num3 += num;
212             }
213             array[i] = Convert.ToChar(num3);
214         }
215         return new string(array);
216     }
217 }
```

Name	Value
array	[char[0x00000030]]
[0]	0x0068 'h'
[1]	0x0074 't'
[2]	0x0074 't'
[3]	0x0070 'p'
[4]	0x003A ':'
[5]	0x002F '/'
[6]	0x002F '/'
[7]	0x006C 'l'
[8]	0x006F 'o'

Podemos notar que retorna una **String** y que es una dirección web, no la mostramos completa porque nos quedaría una captura muy grande, pero lo relevante lo dejamos explicado y es que por esa función pasarán todas nuestras **Strings** encriptadas. Con colocar un **<BREAKPOINT>** en el retorno de la función, **255 return new string(array)**, podemos conocer la **String** desencriptada. Terminemos de hacer esa consulta y miremos toda la consulta primero.

```
232     try
233     {
234         HttpWebResponse httpWebResponse = (HttpWebResponse)httpWebRequest.GetResponse();
235         Stream responseStream = httpWebResponse.GetResponseStream();
236         Encoding encoding = Encoding.GetEncoding("utf-8");
237     }
```

Name	Value
httpWebRequest	(System.Net.HttpWebRequest/*0x02000109/)
Aborted	false
Accept	null
Address	{http://logic.youtubebyclick.com/config.php?Core=3443448&Machine=BFEBFBFF000206520C97DE2B 005056C00001&version=2.2.87&updatec=true&source=MAIN}

Ahí está la dirección completa, solicitará información a su servidor y con esa respuesta que obtenga determinará el tipo de **Licencia**. Hagamos un paréntesis aquí y recordemos que el **<ExeinfoPe v0.0.5.0 2018.03.31>** nos informaba que estaba obfusado, yo supongo que esas **Strings** encriptadas pueden ser parte de la obfuscación pero que no sirve de mucho porque el código así como está se entiende completamente. Bien, sigamos por donde veníamos y miremos la dirección completa.

```
http://logic.youtubebyclick.com/config.php?Core=3443448&Machine=BFEBFBFF000206520C97DE2B|005056C00001&version=2.2.87&updatec=true&source=MAIN
```

Que escalofríos me da, envía información única de cada computadora. Yo no sé exactamente qué información nuestra pueden obtener con esos datos pero lo que si

estoy seguro es que pueden saber las veces que ejecuto su programa y tenerme un registro. Bueno, Supongo que ellos en sus registros guardan el número de **Core** y **MachineID** y con eso obtiene mi **Licencia** que será determinada por la respuesta que recibamos. Estas consultas siempre se hacen dentro de un manejador de excepciones porque si no hay conexión por cualquier motivo se procese ese error, en este caso si no hay conexión el error es usado para retornar nuestra **Licencia** con otros valores que nos dejarán sin la versión **FULL**.

```
public enum License
{
    // Token: 0x04000055 RID: 85
    Unknown,
    // Token: 0x04000056 RID: 86
    NoConnection,
    // Token: 0x04000057 RID: 87
    Trial,
    // Token: 0x04000058 RID: 88
    TrialEnded,
    // Token: 0x04000059 RID: 89
    License,
    // Token: 0x0400005A RID: 90
    LicenseExpired,
    // Token: 0x0400005B RID: 91
    NoConnectionToServer
}
```

```
234 HttpWebResponse httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
235 Stream responseStream = httpResponse.GetResponseStream();
236 Encoding encoding = Encoding.GetEncoding("utf-8");
237 string xml = new StreamReader(responseStream, encoding).ReadToEnd();
238 httpResponse.Close();
239 XmlDocument xmlDocument = new XmlDocument();
240 xmlDocument.LoadXml(xml);
```

CARGA RESPUESTA COMO UN DOCUMENTO XML

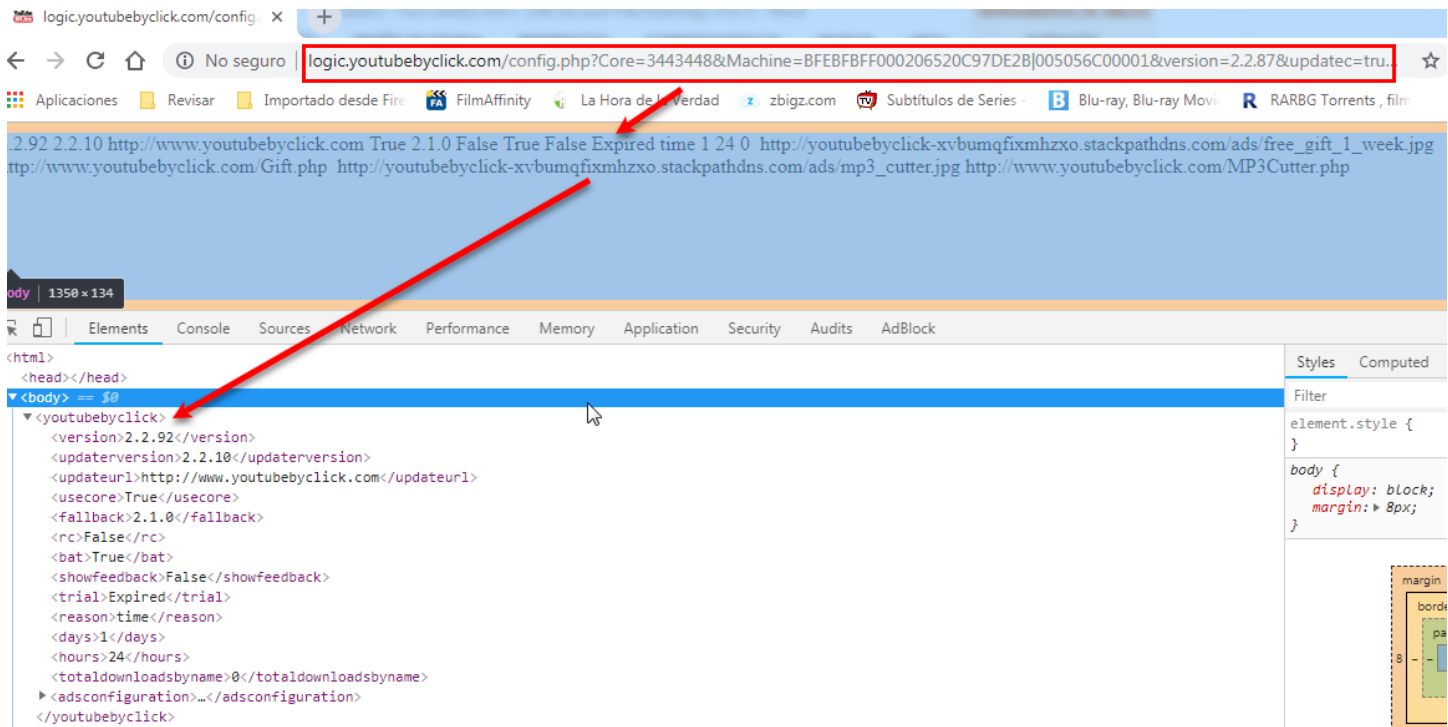
```
<YouTubeByClick>\r\n\t<Version>2.2.92</Version>\r\n\t<UpdaterVersion>2.2.10</UpdaterVersion>\r\n\t<UpdateUrl>http://www.youtubebyclick.com</UpdateUrl>\r\n\t<UseCore>True</UseCore>\r\n\t<Fallback>2.1.0</Fallback>\r\n\t<RC>False</RC>\r\n\t<BAT>True</BAT>\r\n\t<ShowFeedback>False</ShowFeedback>\r\n\t<Trial>Expired</Trial>\t<Reason>time</Reason>\t<Days>1</Days>\t<Hours>24</Hours>\t<TotalDownloadsByName>0</TotalDownloadsByName><AdsConfiguration>\r\n\t<AD>\r\n\t\t<Image>\r\n\t\t\t\thttp://youtubebyclick-xvbumqfixmhzo.stackpathdns.com/ads/free\_gift\_1\_week.jpg\r\n\t\t\t</Image>\r\n\t\t\t<Link>http://www.youtubebyclick.com/Gift.php</Link>\r\n\t\t</AD>\r\n\t\t<AD>\r\n\t\t\t<Image>\r\n\t\t\t\t\thttp://youtubebyclick-xvbumqfixmhzo.stackpathdns.com/ads/mp3\_cutter.jpg\r\n\t\t\t\t\t</Image>\r\n\t\t\t\t\t<Link>http://www.youtubebyclick.com/MP3Cutter.php</Link>\r\n\t\t\t\t</AD>\r\n\t\t</AdsConfiguration>\t\r\n\r\n</YouTubeByClick>
```

[YouTubeByClick v2.2.87 (.NET)(Crack-Patch)(dnSpy v5.0.7)]

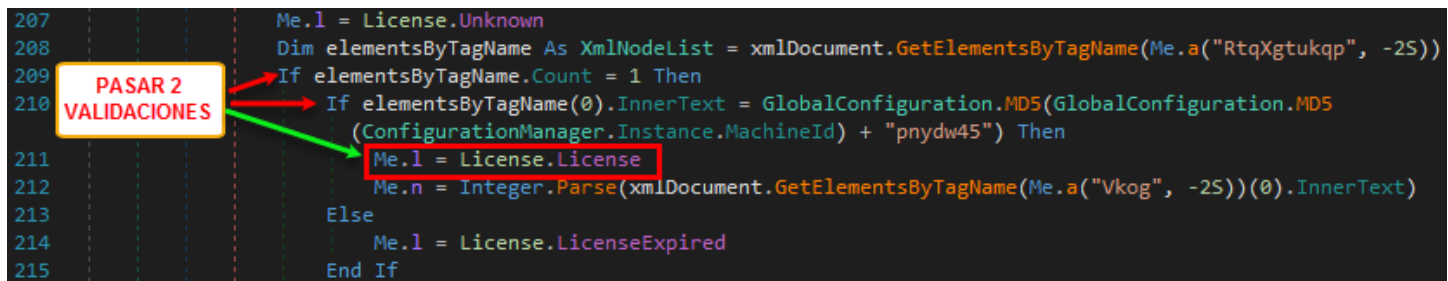
La respuesta de arriba es almacenada en:

```
Dim xml As String = New StreamReader(responseStream, encoding).ReadToEnd()
```

También podemos hallar todo esto colocando la dirección en nuestro navegador.



Lo que hará el programa para determinar nuestra **Licencia** es buscar en ese documento xml si existen etiquetas que solo se podrían tener si el servidor respondería si nosotros tuviéramos una **Licencia** registrada en sus servidores. Entonces debemos obligarlo a que siga el camino de nuestra **Licencia Full**.

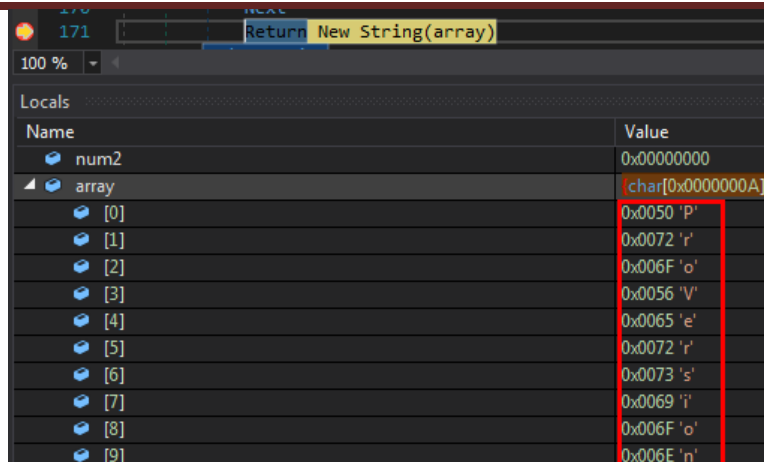


La primer validación comprueba si tenemos una **TAG** específica que esté una solo vez, **209 If elementsByTagName.Count=1 Then**, ¿y cuál es esa TAG?, pues esta definida en la instrucción anterior.

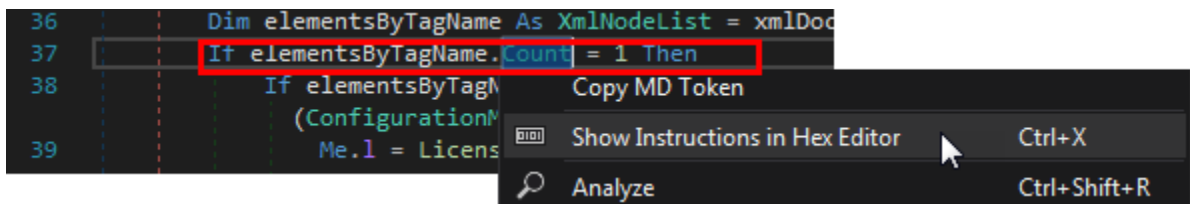
```
208 Dim elementsByTagName As XmlNodeList = xmlDocument.GetElementsByTagName(Me.a(\"RtqXgtukqp\", -2S))
```

Podemos saber la **TAG** a contar si miramos la **String** desencriptada.

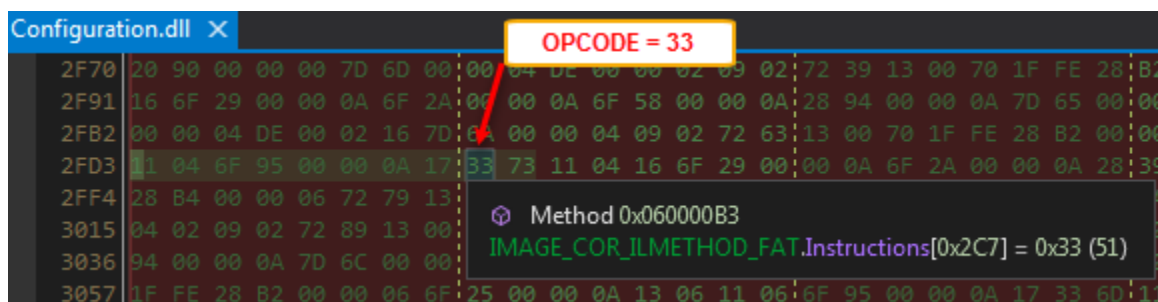
[YouTubeByClick v2.2.87 (.NET)(Crack-Patch)(dnSpy v5.0.7)]



Va a contar la **TAG <ProVersion>** y si la buscamos en la respuesta dada por el servidor nos daremos cuenta que no la tenemos, así que nosotros obtendremos un **0** para comparar con el **1** y como no hay igualdad pues nos vamos por el mal camino. Pueden haber varias soluciones, una sería cambiar el **1** por **0** para que haya igualdad, otra sería cambiar la comparación de la igualdad por "<" ya que al comparar **0** con **1** tendremos **0** menor que **1** y por consiguiente es correcto. Yo optaré por la segunda, cambiar el "=" por "<", esos cambios corresponde a la vista en **Visual Basic**, si la tuviéramos en la vista **C#** sería "==" por "<". El **OPCODE=33** para "=". Busquémoslo en la vista **HEX EDITOR**.



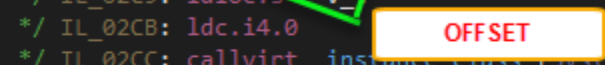
Si apenas estamos empezando con los **.NET** no será muy claro en un principio pero es mientras nos vamos familiarizando. Miremos la ventana **HEX EDITOR**.



Se nos muestran unos **BYTES** resaltados que conforman esa instrucción y como vemos antes del **OPCODE=33** tenemos unos cuantos **BYTES** que nos pueden confundir pero a medida que practiquemos entenderemos mejor y hallaremos los **OPCODES** más fácilmente, además un truco para pillarlos es que lo puedes hallar en esa posición de penúltimo. También lo pillamos más fácilmente en la vista **IL**.

[YouTubeByClick v2.2.87 (.NET)(Crack-Patch)(dnSpy v5.0.7)]

```
/* 0x00002FDB 3373 */ IL_02C7: bne.un.s IL_033C
/* 0x00002FDD 1104 */ IL_02C9: ldloc.s V_4
/* 0x00002FDF 16 */ IL_02CB: ldc.i4.0
/* 0x00002FE0 6F290000A */ IL_02CC: callvirt instance class [System.
System.Xml.XmlNodeList::get_ItemOf(int32)
/* 0x00002FE5 6F2A0000A */ IL_02D1: callvirt instance string [System
/* 0x00002FEA 283900006 */ IL_02D6: call class Configuration.Con
Configuration.ConfigurationManager::get_Instance()
/* 0x00002FEF 6F3D00006 */ IL_02DB: callvirt instance string Configu
/* 0x00002FF4 28B400006 */ IL_02E0: call string Configuration.Gl
/* 0x00002FF9 7279130070 */ IL_02E5: ldstr "pnydw45"
```

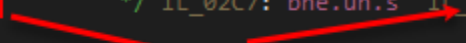


Aquí está mucho mejor y si miramos la instrucción está compuesta por dos **BYTES** 3373, el primer **BYTE** es el que representa el "=" (**OPCODE=33**) que en el lenguaje **MSIL** sería **bne.un.s** y de acuerdo al resultado saltará al **Offset** dado por el segundo **BYTE** 73, entonces si no hay igualdad saltará **73 BYTES** más adelante contados a partir del siguiente **Offset**.

MSIL OpCode Table v1.0 (miroslav.stampar@gmail.com)			
Name	Value	Size	Info
bne.un	40	1	Branches to specified offset if value1 isnt equal to value2 (unsigned or unordered)
bne.un.s	33	1	Branches to specified offset if value1 isnt equal to value2 (unsigned or unordered)
box	8C	1	Converts value type to object reference

Nuestra tablita nos ayuda mucho, nos explica que saltaremos si no son iguales. Como no tenemos una **Licencia** válida siempre saltaremos. Miremos los **Offset** para saber a dónde saltarían.

```
/* 0x00002FDB 3373 */ IL_02C7: bne.un.s IL_033C
/* 0x00002FDD 1104 */ IL_02C9: ldloc.s V_4
/* 0x00002FDF 16 */ IL_02CB: ldc.i4.0
```



Como sabemos saltaremos **73 BYTES** a partir del **IL_02C9**, al sumar **73 + 02C9 = 033C** que será la longitud del salto tomado. Un poco de análisis para futuros crackeos.

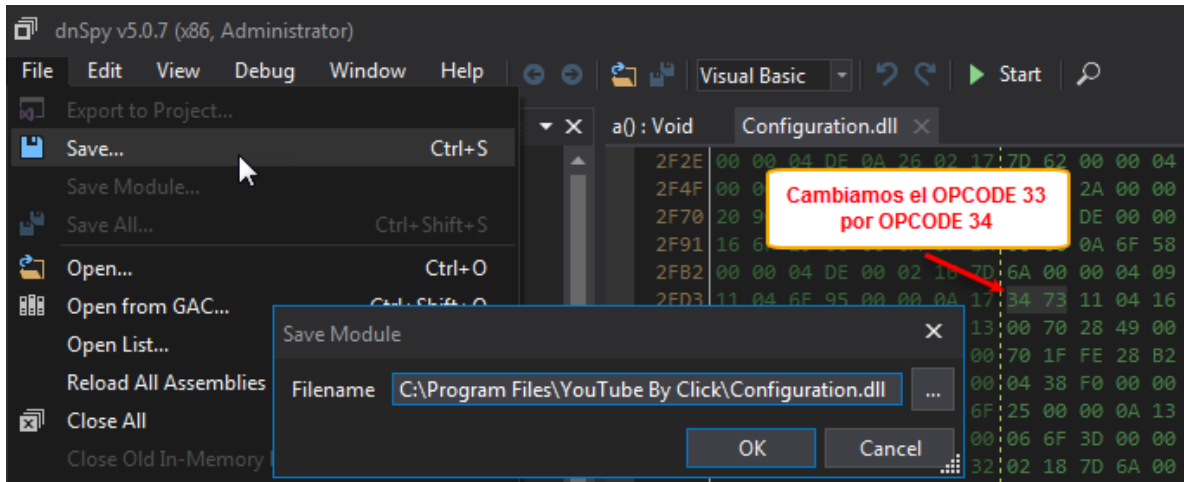
Void	Configuration.dll
2FB2	00 00 04 DE 00 02 16 70 6A 00 00
2FD3	11 04 6F 95 00 00 0A 17 33 73 11
2FF4	28 B4 00 00 06 72 79 13 00 70 28
3015	04 03 00 02 73 00 13 00 70 15 5F

Si lo vemos desde el **HEX EDITOR** a partir de la vista **IL** se nos resaltan solamente los dos **BYTES** y no como hace rato que eran más. Mostrando diferentes formas de ver lo mismo. Como ya tenemos el **OPCODE** a cambiar solo nos falta cambiarlo por el **OPCODE** "<" que es el **OPCODE=34**.

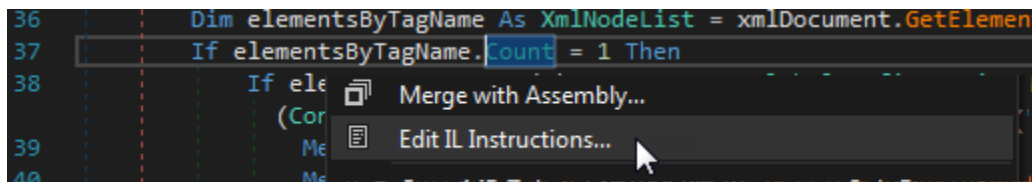
[YouTubeByClick v2.2.87 (.NET)(Crack-Patch)(dnSpy v5.0.7)]

Name	Value	Size	Info
bge.un	41	1	Branches to specified offset if value1 is greater than or equal to value2 (unsigne...
bge.un.s	34	1	Branches to specified offset if value1 is greater than or equal to value2 (unsigne...
bgt	3D	1	Branches to specified offset if value1 is greater than value2

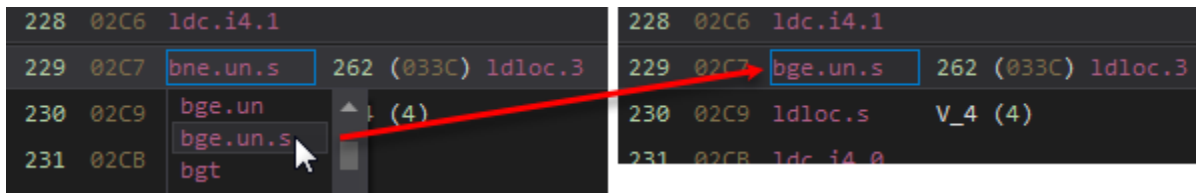
Como vemos la Info dice que salta si el value1 mayor o igual, como sabemos value1 es siempre cero así que nunca saltaremos ya que $0 < 1$. Podemos cambiar el **OPCODE=34** desde el **HEX EDITOR** y guardamos ese primer cambio.



También podemos cambiarlo desde <Clic Derecho>Edit IL Instructions...>.



Y como vimos en nuestra tabla <MSIL OpCode Table v1.0> la instrucción **MSIL** sería **bge.un.s** que es la que debemos colocar ahora.



Seleccionamos la instrucción a cambiar para que se despliegan las diferentes opciones y buscamos la que queremos colocar, solo nos queda guardar los cambios y no olviden guardar una copia del original. Probemos, lleguemos a ese **IF** y miremos si pasamos derecho a la otra instrucción que también es un **IF** pero que debemos sortearla de otra forma porque aquí compara dos **Strings** que se hallan en ejecución.

[YouTubeByClick v2.2.87 (.NET)(Crack-Patch)(dnSpy v5.0.7)]

```
209 If elementsByTagName.Count < 1 Then
210     If elementsByTagName(0).InnerText = GlobalConfiguration.MD5(GlobalConfiguration.MD5
        (ConfigurationManager.Instance.MachineId) + "pnydw45") Then
211         Me.l = License.License
212         Me.n = Integer.Parse(xmlDocument.GetElementsByTagName(Me.a("Vkog", -25))(0).Inne
213     Else
```

Efectivamente no saltamos en la línea 209 que es la primera validación. La siguiente validación origina un error debido a que la **String** que debería tener `elementsByTagName(0).InnerText` no existe porque la **TAG <ProVersion>** no está presente en el documento xml. Resulta que esta instrucción va en realidad a comparar la String de `lementsByTagName(0).InnerText` con un **HASH MD5** los cuales deben ser iguales. Aquí podemos darnos cuenta que en `lementsByTagName(0).InnerText` estaría el **HASH MD5** que retornaría el servidor, calculado inicialmente con nuestro **MachineID** que luego concatena con la **String** "pnydw45" para sacar finalmente el **HASH MD5**. Volvamos al análisis del error, lo que ocurre pasa al manejador de excepciones.

```
240     End If
241     Catch
242         Me.c = ConfigurationManager.Instance.Version
243         Me.f = ConfigurationManager.Instance.Version
244         Me.j = ConfigurationManager.Instance.Version
245         Me.h = False
246         Me.i = False
247         Me.k = False
248         Me.d = True
249     Try
250         Dim httpWebResponse2 As HttpWebResponse = CType(CType(WebRequest
        HttpWebRequest).GetResponse(), HttpWebResponse)
251         If httpWebResponse2.StatusCode = HttpStatusCode.OK Then
252             Me.l = License.NoConnectionToServer
```

Ahí hace más comprobaciones pero que si seguimos por este camino no llegaremos a buen puerto. La solución es **NOPEAR** esa validación de la línea 210 y todo aquello que sobre, que sea basura. Esto lo podemos hacer ya que ese **IF** no tiene ninguna otra instrucción importante. Para que funcione debemos hacerlos desde **<Click Derecho>Edit IL Instructions...** sobre nuestro **IF** a **NOPEAR**.

229	02C7	bge.un.s	262 (033C) ldloc.3	NOP Instructions	N
230	02C9	ldloc.s	V_4 (4)	Invert Branches	I
231	02CB	ldc.i4.0		Convert to Unconditional Branches	B
232	02CC	callvirt	instance class [System.Xml]System.Xml.XmlNode [System.Xml]System.Xml.XmlNodeList::get_ItemOf(int32)		
233	02D1	callvirt	instance string [System.Xml]System.Xml.XmlNode::get_InnerText()		
234	02D6	call	class Configuration.ConfigurationManager Configuration.ConfigurationManager::get_Instance()		
235	02D8	callvirt	instance string Configuration.ConfigurationManager::get_MachineId()		
236	02E0	call	string Configuration.GlobalConfiguration::MD5(string)		
237	02E5	ldstr	"pnydw45"		
238	02EA	call	string [mscorlib]System.String::Concat(string, string)		
239	02EF	call	string Configuration.GlobalConfiguration::MD5(string)		
240	02F4	call	bool [mscorlib]System.String::op_Equality(string, string)		
241	02F9	brfalse.s	258 (0330) ldarg.0		

NOPEAMOS todo lo que sale seleccionado por defecto que representa todo esa sección del **IF**. Luego guardamos esos cambios y miremos cómo nos queda.

[YouTubeByClick v2.2.87 (.NET)(Crack-Patch)(dnSpy v5.0.7)]

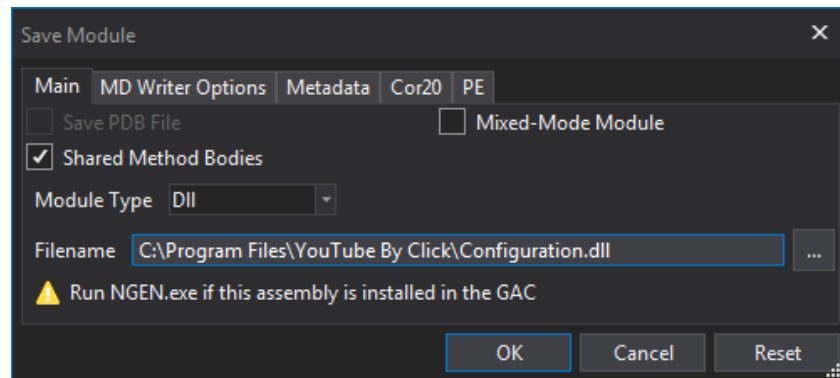
```
Dim elementsByTagName As XmlNodeList = xmlDocument.GetElementsByTagName(Me.a("RtqXgtukqp", -2S))
If elementsByTagName.Count < 1 Then
    If elementsByTagName(0).InnerText = GlobalConfiguration.MD5(GlobalConfiguration.MD5
        (ConfigurationManager.Instance.MachineId) + "pnydw45") Then
        Me.l = License.License
        Me.n = Integer.Parse(xmlDocument.GetElementsByTagName(Me.a("Vkog", -2S))(0).InnerText)
    Else
        Me.l = License.LicenseExpired
    End If
Else
```

ANTES DEL CAMBIO

```
Dim elementsByTagName As XmlNodeList = xmlDocument.GetElementsByTagName(Me.a("RtqXgtukqp", -2S))
If elementsByTagName.Count < 1 Then
    Me.l = License.License
    Me.n = Integer.Parse(xmlDocument.GetElementsByTagName(Me.a("Vkog", -2S))(0).InnerText)
Else
```

DESPUÉS DEL CAMBIO

Como vemos **DESPUÉS DEL CAMBIO** solo tenemos cosas buenas y nuestra **Licencia** pasa ser tomada como la **Licencia** buena, la que debe ser. Guardamos estos cambios que serían los últimos.



Como vemos nuestro **Crack** es la **DLL Configuration.dll**. Existen versiones más recientes del programa, supongo este **Crack** debe seguir sirviendo y lo mismo será para el **Patch** que ya mismo vamos a hacer como lo hicimos en el tuto anterior con el uso del módulo **[Search and Replace Patch]**. Ya explicamos cómo hacerlo, busquemos nuestro patrón para reemplazarlo comparando el original y el crackeado para tenerlo como guía.

ORIGINAL				CRACKED			
268	/* 0x00002FD1	1304	*/ IL_02BD: stloc.s V_4	269	/* 0x0000305B	1104	*/ IL_028F: ldloc.s V_4
269	/* 0x00002FD3	1104	*/ IL_02BF: ldloc.s V_4	270	/* 0x0000305D	6F950000A	*/ IL_02C1: callvirt inst
270	/* 0x00002FD5	6F950000A	*/ IL_02C1: callvirt instance int32	271	/* 0x00003062	17	*/ IL_02C6: ldc.i4.1
271	/* 0x00002FDA	17	*/ IL_02C6: ldc.i4.1	272	/* 0x00003063	344D	*/ IL_02C7: bge.un.s IL_0
272	/* 0x00002FDB	3373	*/ IL_02C7: bne.un.s IL_033C	273			
273				274	/* 0x00003065	00	*/ IL_02C9: nop
274	/* 0x00002FDD	1104	*/ IL_02C9: ldloc.s V_4	275	/* 0x00003066	00	*/ IL_02CA: nop
275	/* 0x00002FDF	16	*/ IL_02CB: ldc.i4.0	276	/* 0x00003067	00	*/ IL_02CB: nop
276	/* 0x00002FE0	6F290000A	*/ IL_02CC: callvirt instance class System.Xml.XmlNodeList :get_ItemOf(int32)	277	/* 0x00003068	00	*/ IL_02CC: nop
277	/* 0x00002FE5	6F2A0000A	*/ IL_02D1: callvirt instance string System.Xml.XmlNodeList :get_ItemOf(int32)	278	/* 0x00003069	00	*/ IL_02CD: nop
278	/* 0x00002FEA	2839000006	*/ IL_02D6: call class Configuration.ConfigurationManager::get_Instance()	279	/* 0x0000306A	00	*/ IL_02CE: nop
279	/* 0x00002FEF	6F3D000006	*/ IL_02DB: callvirt instance string Configuration.ConfigurationManager::get_Instance()	280	/* 0x0000306B	00	*/ IL_02CF: nop
280	/* 0x00002FEA	2839000006	*/ IL_02E0: call string Configuration.ConfigurationManager::get_Instance()	281	/* 0x0000306C	00	*/ IL_02D0: nop
				282	/* 0x0000306D	00	*/ IL_02D1: nop
				283	/* 0x0000306E	00	*/ IL_02D2: nop

Buscamos nuestro patrón original y nuestro patrón a reemplazar.

[YouTubeByClick v2.2.87 (.NET)(Crack-Patch)(dnSpy v5.0.7)]

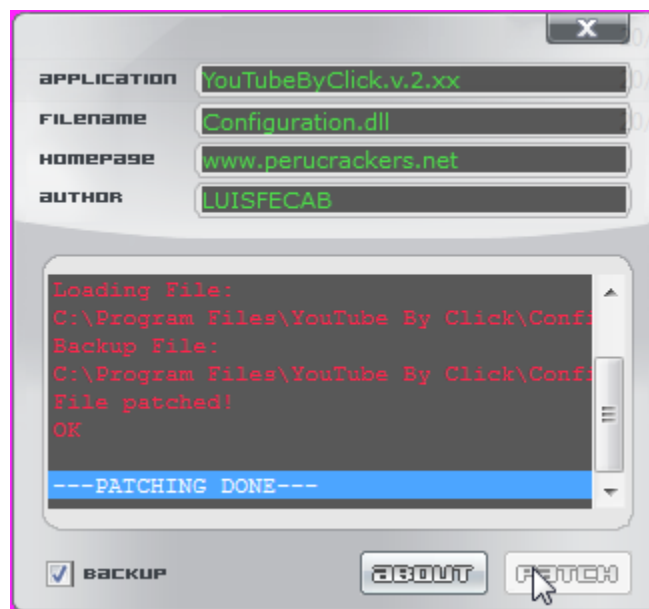
BYTES ORIGINALES:

```
33731104166F2900000A6F2A00000A28390000066F3D00000628B4000006727913007028490000  
0A28B4000006284700000A2C35
```

BYTES NUEVOS:

```
34??000000000000000000000000000000000000000000000000000000000000000000000000  
0000000000000000000000000000
```

Si miran el primer **BYTE** lo cambiamos por nuestro **OPCODE=34** (<) el segundo **BYTE** es la longitud del salto que no cambia, luego **NOPEAMOS** los **BYTES** que conforman el **IF** no deseado. Creamos nuestro **Patch** con el <dUP2 Diablo's Universal Patcher v2.26> y lo probamos.



Me funciona bien, ya con esto tenemos **Crack** y **Patch**. Miremos el programa en acción.

[YouTubeByClick v2.2.87 (.NET)(Crack-Patch)(dnSpy v5.0.7)]



Con todas las opciones activadas y sin ningún vestigio de que pida activarlo.

PARA TERMINAR

Programa básico que creo que no debe muy usado pero que para nuestro propósito de practicar lo aprendido con los **.NET** nos resulta un muy buen **Target**. Quedaría faltando pasarlo por el <de4dot> para ver si quita las **Strigs** encriptadas, cosa que no hice porque en realidad no es necesario porque todo el código se entiende perfectamente.

Mis saludos y agradecimientos especiales para **Dani**, **DavicoRm**, **nextco**, **AbelJM**, **SoftDat**, **lior**, **Apuromafo** y al maestro **Ricardo**.

Se despide su amigo,

@LUISFECAB