

ByteFence

--..sequeyo..--

Hola familia, saludos de nuevo. Un placer andar de nuevo escribiendo para CracksLatinos. Hablando hace un tiempo con Apuromafo sobre el MalwareBytes como protector extra en nuestros PC's, me comentó que había una alternativa que él conocía. También había que pasar por caja si queríamos una versión PRO pero que rulaba por ahí un serial que te lo registraba. Encima y para colmo me dijo que es un .NET que en su día estuvo investigando así que me puso los dientes largos y decidí pegarle una mirada. El programa en cuestión es el ByteFence y si van a la web del producto <https://es.bytefence.com/> podrán informarse un poco sobre dicho software y descargarlo si lo desean. Básicamente tiene el mismo funcionamiento que el que yo ya tengo (MalwareBytes AntiMalware): lo ejecutan, lo dejan corriendo y si detecta en algún momento que algo puede afectar su equipo, actuará en consecuencia. También pueden escanear su equipo en busca de algo que se pueda considerar sospechoso. No voy a pararme en comparar uno con otro: el ByteFence versus MalwareBytes. Si por la razón que sea, no disponen de ningún software de este tipo y quieren probar a ver si les soluciona algún percance que hayan tenido con una infección, no veo por qué no darle una oportunidad a este software. Una vez que lo descarguen y lo instalen, cuando lo abran lo tendrán PRO totalmente funcional pero solo por 14 días. Durante ese tiempo lo pueden probar, pero cuando finalice el trial, se les quedará free y con algunas funciones capadas. En mi máquina, lo tengo desde hace unos días y ya me descontaron 3 días:

The screenshot shows the ByteFence Anti-Malware Pro application window. At the top, there is a navigation bar with the following items: ByteFence Anti-Malware Pro (Prueba), INICIO (Home), ANALIZAR (Scan), NAVEGADOR (Navigator), AJUSTES (Settings), ESTADÍSTICAS (Statistics), and ACTIVAR (Activate). On the left side, there is a sidebar with three main sections: AJUSTES (Settings), GENERAL (General), and PROTECCIÓN (Protection). The AJUSTES section is currently selected and displays a gear icon. The GENERAL and PROTECCIÓN sections are shown below it. In the main content area, there is a section titled 'LICENCIAS' (Licenses). It contains the following information:

Serial:	Activar
Estado del serial:	Prueba de Versión Pro
Licencia:	Prueba de Versión Pro completa con actualizaciones ilimitadas
Restante:	La Prueba de la Versión Pro caduca en 11 días.

Bien, lo normal sería desensamblarlo y depurarlo con algún debugger apropiado para la plataforma NET y desde luego sin dudarlo, el dnSpy. Pero como ayuda extra y como consejo personal deberían utilizar (y aquí utilizaré) el DotNetTracer que a grandes rasgos lo que hace es correr o lanzar el programa víctima para capturar

todos los eventos y funciones a la que accede la víctima en su funcionamiento normal. Con ello podemos saber en todo momento cómo se llaman los métodos que se ejecutan en la carga del programa o cuando se aprieta un botón. Si recuerdan de la época del Visual Basic, es prácticamente lo que haría el famoso SmartCheck de Numega con el que podíamos ver en tiempo real las funciones y eventos que se producían en la ejecución de una víctima. Pero antes de utilizar cualquier herramienta para estudiar un NET, conviene intentar averiguar si está protegido con algún protector, packer u ofuscador. Haber hay muchos como por ejemplo el ProtectionID, pero yo en mi caso, la información la saco directamente del de4dot. Para ello coloco en la carpeta del ejecutable de4dot a la víctima:

AssemblyServer-CLR40-x04	23/03/2019
AssemblyServer-CLR40-x64.exe.config	23/03/2019
AssemblyServer-x64	23/03/2019
AssemblyServer-x64.exe.config	23/03/2019
ByteFence	27/03/2019
de4dot.blocks.dll	23/03/2019
de4dot.code.dll	23/03/2019
de4dot.cui.dll	23/03/2019
de4dot	23/03/2019
de4dot.exe.config	23/03/2019
de4dot.mdecrypt.dll	23/03/2019
de4dot-x64	23/03/2019
de4dot-x64.exe.config	23/03/2019
dnlib.dll	23/03/2019

Como ven en la imagen, copié el ejecutable principal del ByteFence, desde la carpeta donde se instaló hacia la carpeta donde está el de4dot. Como ando en máquina de 64, debo ejecutar el de4dot-x64 desde línea de comandos. Abro la ventana de símbolo de sistema y navego hasta esta carpeta que nos ocupa:

```
Administrator: Símbolo del sistema
Microsoft Windows [Versión 10.0.17763.914]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>cd C:\Herramientas.NET\de4dot (ALL VERSION)\de4dot Latest
C:\Herramientas.NET\de4dot (ALL VERSION)\de4dot Latest>de4dot-x64 -d ByteFence.exe

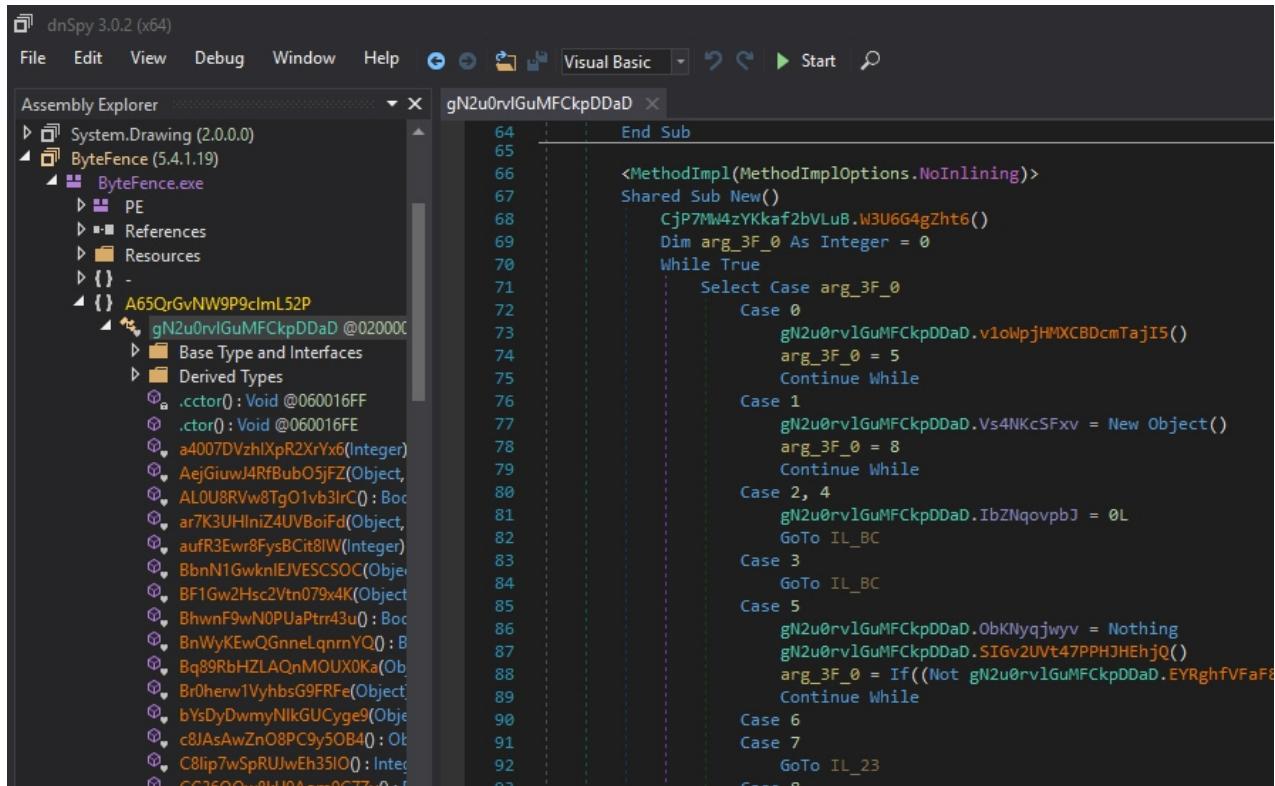
de4dot v3.1.41592.3405 Copyright (C) 2011-2015 de4dot@gmail.com
Latest version and source code: https://github.com/0xd4d/de4dot

Detected .NET Reactor (C:\Herramientas.NET\de4dot (ALL VERSION)\de4dot Latest\ByteFence.exe)

C:\Herramientas.NET\de4dot (ALL VERSION)\de4dot Latest>
```

Y una vez allí tipeo de4dot-x64 -d ByteFence.exe. El argumento “-d” es simplemente

para que de4dot solo haga la detección del protector. Nada más. Como ven en la imagen, me detectó el packer NET Reactor. ¿Por qué hacer esto? Muy simple: Si abrimos la víctima directamente con dnSpy, la simple visión del desensamblado protegido es una pesadilla y esto no hay quien lo siga o quien saque información útil. Lo vemos:



The screenshot shows the dnSpy interface with the assembly code for a protected victim binary. The assembly code is heavily obfuscated, featuring many random names and numbers. The code includes several case statements and goto labels, typical of packer detection logic. The assembly window title is "gN2u0rvlGuMFCKpDDaD". The left pane shows the assembly explorer with various namespaces and types, including "ByteFence" and "A65QrGvNW9P9clmL52P". The code itself is as follows:

```

Assembly Explorer      gN2u0rvlGuMFCKpDDaD
File Edit View Debug Window Help Visual Basic Start Search

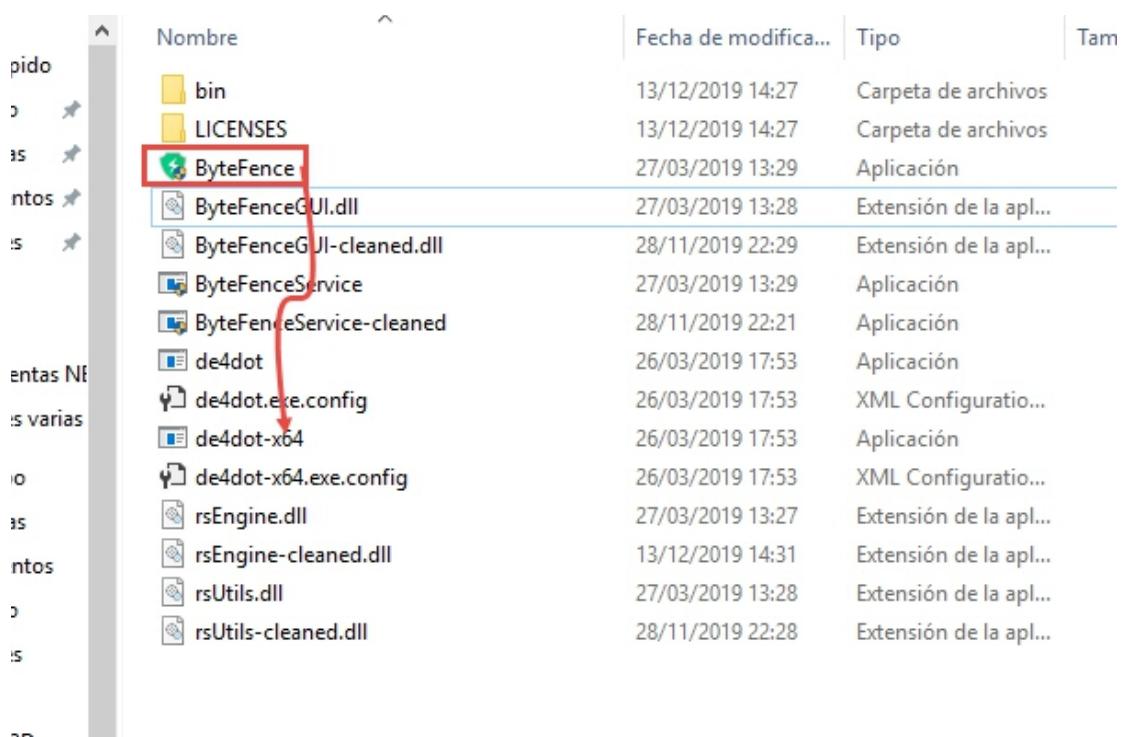
64    End Sub
65
66    <MethodImpl(MethodImplOptions.NoInlining)>
67    Shared Sub New()
68        CjP7MW4zYKkaf2bVLuB.W3U6G4gZht6()
69        Dim arg_3F_0 As Integer = 0
70        While True
71            Select Case arg_3F_0
72                Case 0
73                    gN2u0rvlGuMFCKpDDaD.v1oWpjHMXCBDcmTajI5()
74                    arg_3F_0 = 5
75                    Continue While
76                Case 1
77                    gN2u0rvlGuMFCKpDDaD.Vs4NKcSFxv = New Object()
78                    arg_3F_0 = 8
79                    Continue While
80                Case 2, 4
81                    gN2u0rvlGuMFCKpDDaD.IbZNqovpbJ = 0L
82                    GoTo IL_BC
83                Case 3
84                    GoTo IL_BC
85                Case 5
86                    gN2u0rvlGuMFCKpDDaD.ObKNyqjwyv = Nothing
87                    gN2u0rvlGuMFCKpDDaD.SIGv2UVt47PPHJHEhjQ()
88                    arg_3F_0 = If((Not gN2u0rvlGuMFCKpDDaD.EYRghfVFaF&
89                    Continue While
90                Case 6
91                Case 7
92                    GoTo IL_23
93                Case 8

```

Fíjense qué horror. Aquí no hay quien se aclare y esto es imposible de seguir. Mucho menos sacar algo en claro de toda esta pesadilla. Por eso hay que averiguar si la víctima está protegida. Si no lo está, mejor: menos trabajo y si lo está, hay que actuar en consecuencia. Yo hace tiempo me bajé de algún sitio un recopilatorio de los diferentes mods que se hicieron del de4dot para diferentes protectores .NET. Muchos dirán que eso de andar con unpackers no es trabajo fino y que hay que romperse el cerebro desempacando. Que debería ser uno mismo el que haga todo ese trabajo. Bien, yo lo respeto. Pero para ser sinceros, si ya alguien se preocupó de crear herramientas para tal efecto y para facilitarnos la tarea a los demás, pues es de agradecer y para eso se crearon: para que nos aprovecháramos de ellas. En la siguiente imagen les muestro una parte del recopilatorio de dichas mods específicas para cada packer. Ojo, no quiere decir esto que sean perfectas. En la mayoría de los casos te solucionan el trabajo en un 90% de las veces y puede que se requiera de algún comando extra a la hora de romperles la protección, pero en líneas generales funcionan muy bien:

Nombre
.Net Guard Unpacker
de4dot (B@S)
de4dot (Crypto,Phoenix,Reactor,OrangeHeap)
De4dot (DorAEm0nKiNG)
de4dot (FamilyFree v2)
de4dot (IvancitoOz)
de4dot (kao)
de4dot (String Deob.)
de4dot (Syklon) Phoenix Protector
de4dot (TheProxy)
de4dot (Wuhensoft)
de4dot .Net Reactor (Unknown)
de4dot .NET Reactor 5.0
de4dot 2.0.3
de4dot 3.1.0
de4dot 3.1.4
de4dot 4.9 (PC-RET)
de4dot 5.8 (Eazfuscator Support)

Ok, yo en este caso particular que nos ocupa, para romperle la protección al ByteFence, utilicé el de4dot para NET Reactor 5.0. La forma de hacer es copiar el ejecutable víctima, desde la carpeta donde se instaló el programa y pegarlo en la carpeta donde se encuentra de4dot. Una vez que se copió, se le arrastra directamente encima de la herramienta:



	Nombre	Fecha de modifica...	Tipo	Tam
pido	bin	13/12/2019 14:27	Carpeta de archivos	
ntos	LICENSES	13/12/2019 14:27	Carpeta de archivos	
is	ByteFence	27/03/2019 13:29	Aplicación	
entes N	ByteFenceGUI.dll	27/03/2019 13:28	Extensión de la apl...	
is varias	ByteFenceGUI-cleaned.dll	28/11/2019 22:29	Extensión de la apl...	
io	ByteFenceService	27/03/2019 13:29	Aplicación	
rs	ByteFenceService-cleaned	28/11/2019 22:21	Aplicación	
ntos	de4dot	26/03/2019 17:53	Aplicación	
o	de4dot.exe.config	26/03/2019 17:53	XML Configuratio...	
is	de4dot-x64	26/03/2019 17:53	Aplicación	
ntos	de4dot-x64.exe.config	26/03/2019 17:53	XML Configuratio...	
o	rsEngine.dll	27/03/2019 13:27	Extensión de la apl...	
is	rsEngine-cleaned.dll	13/12/2019 14:31	Extensión de la apl...	
ntos	rsUtils.dll	27/03/2019 13:28	Extensión de la apl...	
o	rsUtils-cleaned.dll	28/11/2019 22:28	Extensión de la apl...	

Justo cuando se arrastra, se abre una ventana de comandos y se ve el progreso y lo que hace la herramienta:

```
C:\Herramientas.NET\de4dot (ALL VERSION)\de4dot .NET Reactor 5.0\de4dot-x64.exe

de4dot v3.1.41592.3405 Copyright (C) 2011-2015 de4dot@gmail.com
Latest version and source code: https://github.com/0xd4d/de4dot

[*] Support .NET Reactor 5.0

Detected .NET Reactor (C:\Herramientas.NET\de4dot (ALL VERSION)\de4dot .NET Reactor 5.0\ByteFence.exe)
Cleaning C:\Herramientas.NET\de4dot (ALL VERSION)\de4dot .NET Reactor 5.0\ByteFence.exe
WARNING: Could not find all arguments to method System.Boolean aa34qNsunpq3hGd6xtN.CjP7MW4zYKkaf2bVLuB::em.Int32) (06003909), instr: IL_000F: ldarg
Renaming all obfuscated symbols
Saving C:\Herramientas.NET\de4dot (ALL VERSION)\de4dot .NET Reactor 5.0\ByteFence-cleaned.exe
Ignored 294 warnings/errors
Use -v/-vv option or set environment variable SHOWALLMESSAGES=1 to see all messages

Press any key to exit...
```

Ahí se ven las distintas fases del comportamiento de de4dot. Pulsar cualquier tecla para salir de consola. Pese a algún warning o aviso, parece que hizo su trabajo y dejó en la misma carpeta a la víctima limpia renombrada con su nombre original más la coletilla “-cleaned”. Yo por las pruebas que estuve haciendo, no detecté mal funcionamiento alguno en los cleaned a la hora de ejecutarse la víctima. Vemos el cleaned en carpeta:

Nombre	Fecha
bin	13/07/2015 10:45:40
LICENSES	13/07/2015 10:45:40
ByteFence	27/07/2015 10:45:40
ByteFence-cleaned	17/07/2015 10:45:40
ByteFenceGUI.dll	27/07/2015 10:45:40
ByteFenceGUI-cleaned.dll	28/07/2015 10:45:40
ByteFenceService	27/07/2015 10:45:40

Bien. Yo normalmente lo que hago es guardar los originales de las víctimas en una carpeta creada para tal fin por si sale algo mal y en su lugar, coloco los cleaned renombrándolos obviamente:

« Archivos de programa > ByteFence		
	Nombre	Fec
	Logs	15/
	ORIGINALES	16/
	rtop	15/
	Scans	15/
	x64	15/

En la carpeta donde se instaló la aplicación víctima, creo otra donde ir guardando los originales. En dicha carpeta, meto el ejecutable ByteFence.exe original y en su lugar meto la cleaned en la principal:

« Archivos de programa > ByteFence		
	Nombre	Fecha de modifica... Tipo
do	Logs	15/12/2019 14:20 Carpeta
dos	ORIGINALES	16/12/2019 0:37 Carpeta
	rtop	15/12/2019 14:36 Carpeta
	Scans	15/12/2019 14:20 Carpeta
	x64	15/12/2019 14:20 Carpeta
	x86	15/12/2019 14:20 Carpeta
	ByteFence.exe.config	27/03/2019 13:26 XML Config
tas NB	ByteFence-cleaned	17/12/2019 18:23 Aplicación
varias	ByteFenceGUI.dll	27/03/2019 13:28 Extensión
	ByteFenceScan	27/03/2019 13:30 Aplicación
	ByteFenceService	27/03/2019 13:29 Aplicación
	ByteFenceService.exe.config	27/03/2019 13:26 XML Config

Ya hice el cambio y ahora solo hay que renombrarla para que quede como si fuese la original, es decir, renombrarla quitándole la coletilla “-cleaned”. Lo hago:

« Archivos de programa > ByteFence		
	Nombre	Fech
	Logs	15/1.
	ORIGINALES	16/1.
	rtop	15/1.
	Scans	15/1.
	x64	15/1.
	x86	15/1.
	ByteFence	17/1.
	ByteFence.exe.config	27/0
	ByteFenceGUI.dll	27/0
	ByteFenceScan	27/0

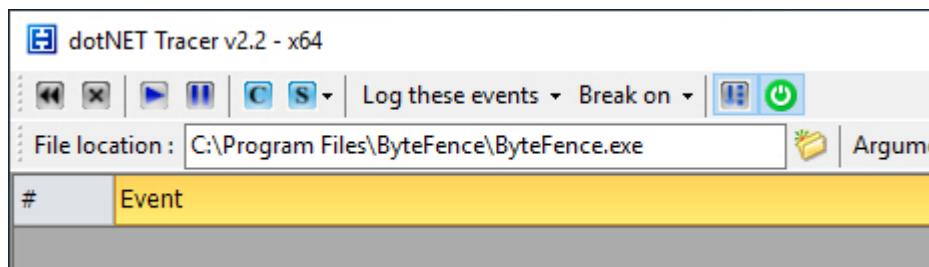
Ya está. El programa sigue trabajando de igual forma cuando lo ejecuto y no observo

ninguna incidencia en su funcionamiento normal. Si ahora vuelvo a abrir dnSpy, la cosa cambia de forma bestial y ya podemos ir mirando teniendo las cosas muchísimo más claras:

The screenshot shows the dnSpy interface with the Assembly Explorer on the left and the code editor on the right. The Assembly Explorer tree shows the project structure, including the main executable and a specific form named frmChromeExtensionOfferScan. The code editor displays the C# source code for this form, which includes various event handlers and properties. The code uses reflection and localization, as indicated by the numerous method and property names starting with 'smethod' or 'Class' followed by numbers.

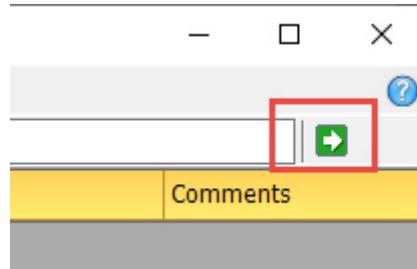
```
dnSpy 3.0.2 (x64)
File Edit View Debug Window Help Visual Basic Start
Assembly Explorer frmChromeExtensionOfferScan
dnSpy (3.0.2.0)
ByteFence (5.4.1.19)
ByteFence.exe
PE
References
Resources
{}
ByteFence
frmChromeExtensionOfferScan @
Base Type and Interfaces
Derived Types
.ctor() : Void @060000C6
addscanDetectionObject(Scan)
askClose(Boolean) : Void @060
cmdApply_Click(Object, Event)
cmdCancel_Click(Object, Even
Dispose(Boolean) : Void @060C
frmAlertsRTP_FormClosing(Obj
frmCEOS_Shown(Object, Even
getActiveAlertPaths() : List(Of
handleIgnore(AlertRTP.Class22
hideModalOverlay() : Void @060
InitializeComponent() : Void @
method_400 : Integer @060000C
method_410 : Void @060000D
method_420 : Void @060000E
Try
Me.clickInstall.setImage(TImage.FromStream(New Me
```

En fin, sobran las palabras después de ver esta imagen. Entenderán que ahora, es mucho más fácil y ameno investigar las tripas de la víctima. Una opción para buscar la o las zonas calientes sería entretenerte en buscarlas siguiendo como pistas nombres de métodos, funciones, eventos click de botones o cadenas significativas que hagan referencia a registro de serial o parecidos. En un programa grande como éste eso te puede llevar mucho tiempo y si dispones de él y no te importa pues adelante. Pero si utilizamos alguna herramienta, como por ejemplo la que les comenté antes, la DotNetTracer, podemos ver en tiempo real qué se está ejecutando e incluso qué se ejecuta cuando hacemos click en uno u otro botón. Hagamos la prueba. Abro DotNetTracer y cargo la víctima:

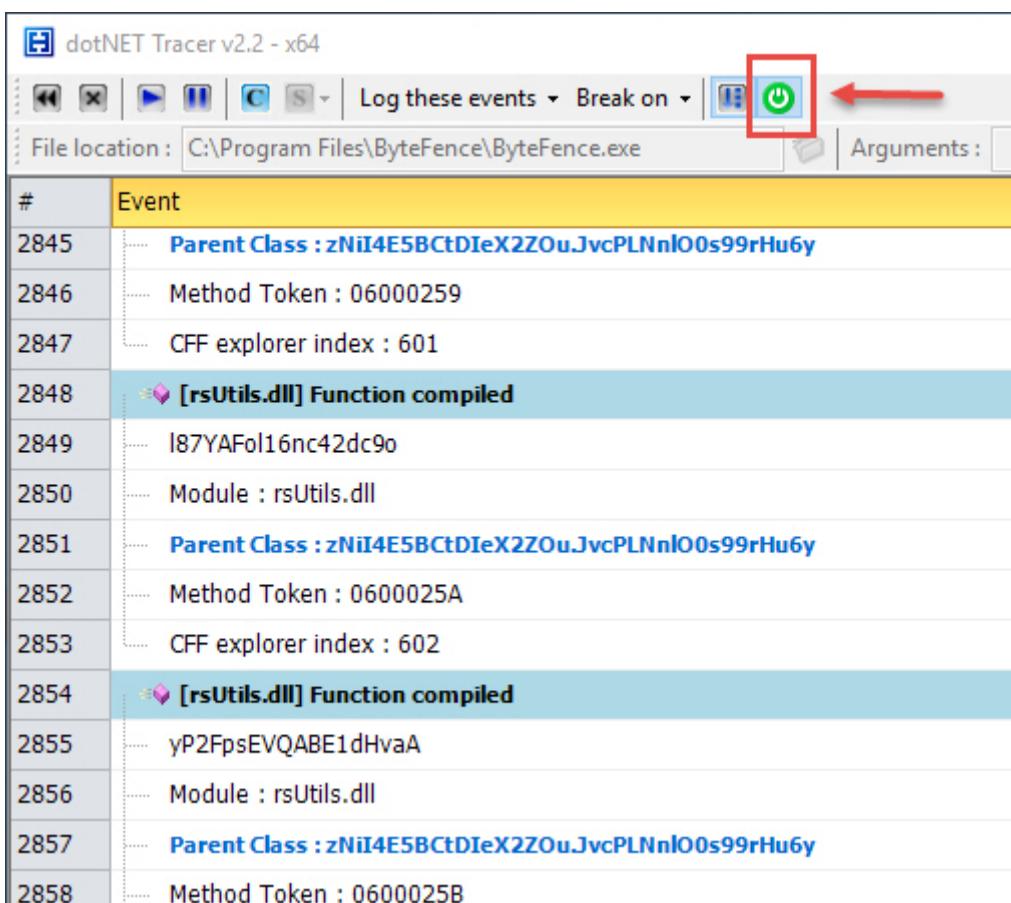


Les comento antes de seguir, que encontré esta herramienta solo para x64 y las anteriores que yo ya tenía no me funcionaban (podían tracear x32). Bueno, me

supongo que será solo cuestión de tiempo que saquen alguna con funcionalidad en x32 y Windows 10. Pues una vez cargada la víctima (recuerden que es la cleaned pero renombrada), solo hay que pulsar el botón verde con la flechita blanca que hay en la parte derecha de la herramienta para que comience a tracear:



Y esperamos que se ejecute ByteFence:



Ya le di a comenzar el traceo y verán en la imagen que está capturando todo lo que se está ejecutando, funciones, así como las distintas dll's que entran en juego. Tenemos la opción de inhabilitar en el botoncito verde el traceo mientras se carga la víctima y eso nos ahorra llenar la ventana de resultados. Se puede limpiar dicha ventana de resultados y volver a activar la opción de traceo justo cuando vayamos a pulsar un botón y de esa forma los resultados son más concisos y centrados en lo que ocurra cuando se haya apretado un botón. Después de un tiempito, la víctima ya se ha ejecutado del todo:

ByteFence Anti-Malware Pro (Prueba)

 ByteFence	 INICIO	 ANALIZAR	 NAVEGADOR	 AJUSTES	 ESTADÍSTICAS	ACTIVAR
---	--	--	---	---	--	---------

LICENCIA	● Prueba de Versión Pro registrada. Su HP es seguro	Mi licencia
PROGRESO DEL ANÁLISIS	● No está funcionando (no programado)	Gestionar
 PROTECCIÓN Riesgos Bloqueados: 0	● Activo y protegiendo su Computadora.	Administrar
 NOTIFICACIÓN	¡Necesita realizar un análisis!	



Último análisis rápido: (nunca)

Último análisis completo del sistema: (nunca)

Ya está cargado y lanzado gracias al DotNetTracer. Ahí indica que la versión PRO está registrada obviamente mientras dure el trial de 14 días. Vuelvo al traceador y limpio la ventana de resultados pues de momento, no quiero nada de ahí del arranque de la víctima. Se hace desde el ícono con la letra C de Clear:

dotNET Tracer v2.2 - x64

#	Event
6013	Module : rsEngine.dll
6014	Parent Class: Reason.rsEngine.Scan.OnAccess
6015	Method Token : 060008BC
6016	CFF explorer index : 2236
6017	[ByteFenceGUI.dll] Function compiled
6018	qrift5N3S2qOSUWfdCqZ
6019	Module : ByteFenceGUI.dll
6020	Parent Class : ByteFence.GUI.BaseControls.Button
6021	Method Token : 060012A1
6022	CFF explorer index : 4769
6023	[rsEngine.dll] Function compiled
6024	.ctor
6025	Module : rsEngine.dll
6026	Parent Class : Reason.rsEngine.Utilities.ExpiringList`1

Pulso dicho icono y me limpia la ventana de resultados y recuerden que sigue deshabilitado el trazado para que no se llene con eventos espurios. Navego por ByteFence y me voy hasta la ventana desde donde se hace la activación por serial y lo relleno con cualquier cosa teniendo en cuenta el patrón que debe seguir y del que informan en la misma ventana de registro:

INGRESE EL NÚMERO SERIAL DE SU LICENCIA



5555-6666-7777-8888

Ingrese su número serial de su licencia arriba y haga clic en el botón 'Activar' a continuación.

Su número serial de su licencia luce así: 1111-2222-3333-4444

Contiene sólo números y se puede introducir con o sin guiones.

[ta comprar una licencia?](#)

Activar

Aquí indican el patrón que debe tener el serial y de ese modo le coloqué uno cualquiera. Vuelvo al traceador, vuelvo a habilitar el trazado en tiempo real y pulso en ByteFence el botón de Activar:

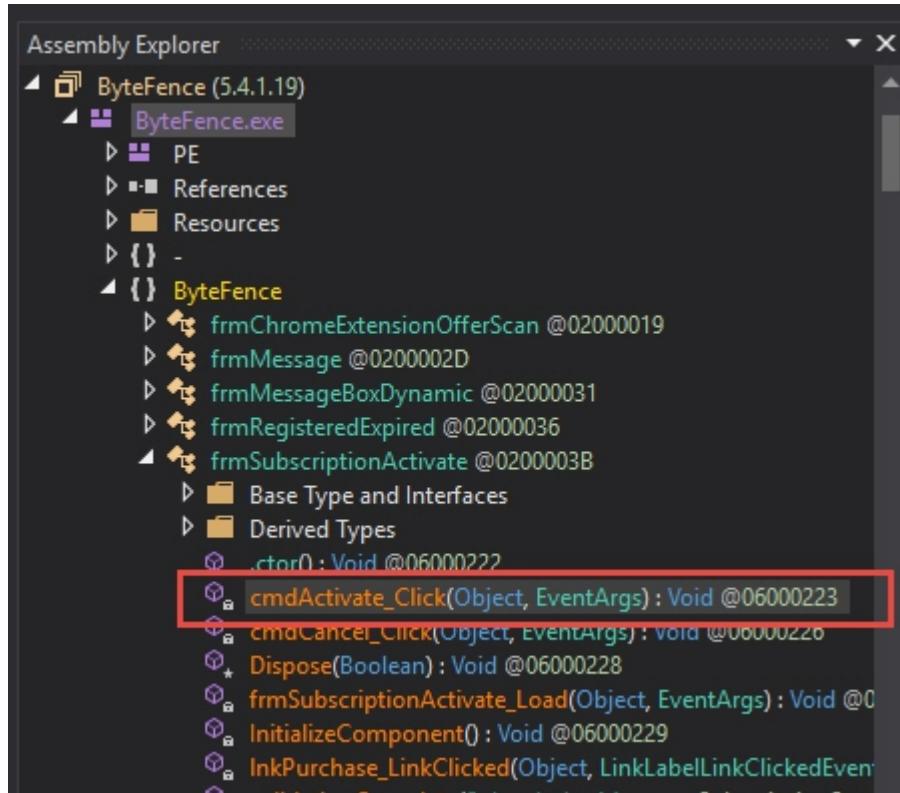


Obviamente me avisa que el serial no es correcto. Después de haber esperado un poco hasta que el traceador me capture todo, investigo un poco en la ventana de resultados:

#	Event
55	◆ [ByteFenceGUI.dll] Function compiled IYgeb53AOnpxo5CWfa4 Module : ByteFenceGUI.dll Parent Class : ByteFence.GUI.BaseControls.Button Method Token : 060012A3 CFF explorer index : 4771
61	◆ [ByteFence.exe] Function compiled cmdActivate_Click Module : ByteFence.exe Parent Class: ctlFeature_Subscription_Activate Method Token : 06000A0A CFF explorer index : 2570
67	◆ [ByteFence.exe] Function compiled .ctor

No tengo que navegar demasiado hacia abajo y me encuentro algo bastante

descriptivo. Hace referencia al cmdActivate_Click del ejecutable. Bien, si me voy de nuevo a dnSpy ya no tengo que gastar tiempo en buscar casi a ciegas. Ahora ya puedo ir directo al tema:



```

1  ' ByteFence.frmSubscriptionActivate
2  Private Sub cmdActivate_Click(sender As Object, e As EventArgs)
3      Try
4          Dim [class] As frmSubscriptionActivate.Class28 = New frmSubscriptionActivate.Class28()
5          [class].frmSubscriptionActivate_0 = Me
6          If Me.txtLicense.Text.Trim().Length = 0 Then
7              Me.panellicenseKey.BackColor = ColorTable.ColorAccentRisk
8              Me.txtLicense.[Select]()
9          Else
10             Me.qqiqqNkcss.Enabled = False
11             Me.Cursor = Cursors.WaitCursor
12             Me.loadingActivate.Visible = True
13             Application.DoEvents()
14             [class].string_0 = Me.txtLicense.Text
15             ThreadPool.QueueUserWorkItem(AddressOf [class].method_0)
16         End If
17     Catch ee As Exception
18         Logger.LogError("ByteFence", "frmSubscriptionActivate", "cmdActivate_Click", "Wrapper", ee)
19     End Try
20 End Sub
21 
```

Aquí en este bloque Try lo que se hace primeramente es verificar con el IF si el largo del txtLicense.Text (donde escribimos nuestro serial trucho) es igual a 0 después de hacerle un Trim. Si eso fuera así, el programa no hace nada y se queda a la espera de que se le meta otro serial. Si no fuera ese el caso, pasariamos al Else en el que se le asigna a la variable string_0 el contenido del txtLicense y se crea un subproceso o hilo de ejecución con el espacio de nombres ThreadPool.QueueUserWorkItem al que

se le asigna el trabajo que tiene que hacer: en este caso el method_0 que es una subrutina. La vemos en la siguiente imagen:

```
Friend Sub method_0(object_0 As Object)
    Try
        Dim flag As Boolean = False
        Dim subscriptionStatuses As SubscriptionManager.SubscriptionStatuses = Class121.validateSubscription(Me.string_0, flag)
        If Not Me.frmSubscriptionActivate_0.IsDisposed Then
            Me.frmSubscriptionActivate_0.Invoke(AddressOf Me.frmSubscriptionActivate_0.validationComplete, New Object()
                { subscriptionStatuses, flag })
        End If
    Catch ee As Exception
        Logger.LogError("ByteFence", "frmSubscriptionActivate", "cmdActivate_Click", "Thread Wrapper", ee)
    End Try
End Sub
```

Aquí en esta subrutina se inicializan dos variables: una, la flag que será un booleano a la que ya directamente se le asigna False y otra la subscriptionStatuses que será del tipo SubscriptionStatuses:

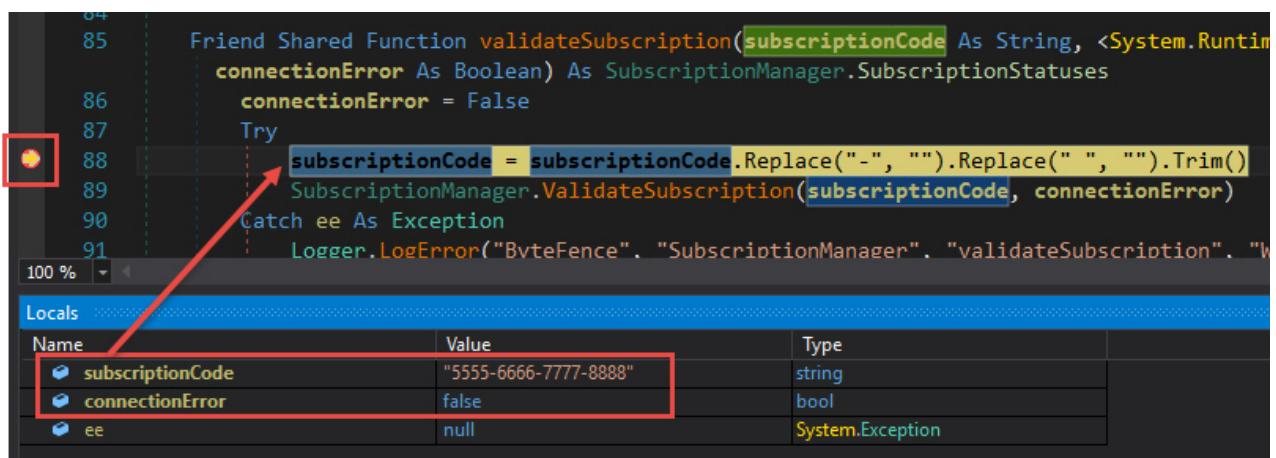
```
Friend Enum SubscriptionStatuses
    ' Token: 0x02000015 RID: 21
    None
    Trial
    TrialExpired
    Registered
    RegisteredExpired
End Enum
```

Es decir, que la variable subscriptionStatuses tomará uno de los valores de esta enumeración. Dicha variable tomará el valor que le devuelva la función validateSubscription a la que se le aportan los argumentos string_0 y el flag:

```
Friend Shared Function validateSubscription(subscriptionCode As String, <System.Runtime.InteropServices.OutAttribute> connectionError As Boolean) As SubscriptionManager.SubscriptionStatuses
    connectionError = False
    Try
        subscriptionCode = subscriptionCode.Replace("-", "").Replace(" ", "").Trim()
        SubscriptionManager.ValidateSubscription(subscriptionCode, connectionError)
    Catch ee As Exception
        Logger.LogError("ByteFence", "SubscriptionManager", "validateSubscription", "Wrapper", ee)
    End Try
    Return Class121.getSubscriptionStatus()
End Function
```

Esta es la función validateSubscription. De nuevo en un bloque Try lo que se hace es reemplazar al serial que le metimos, que está en la variable subscriptionCode, todos los guiones que tenga por una string nula gracias al "" además de quitarle cualquier espacio que pudiera contener. Acto seguido se entra en la función ValidateSubscription con el serial trucho sin guiones y con connectionError en False puesto que se inicializó con ese valor. No debemos confundirnos con el nombre de las funciones porque si observan, una empieza en minúscula y otra en mayúscula.

Podríamos seguir si se quiere, con este análisis en estático y sacaríamos mucha información, pero podría ya ser buen momento para depurar a la víctima y vamos viendo in situ qué valores van tomando las distintas variables. Para los que ya andamos desde hace ya algún tiempo con los .NET, el funcionamiento de dnSpy lo conocemos con soltura y para aquellos que no estén familiarizados, les invito que lean algunos de los tutoriales que tenemos en CracksLatinos para la plataforma .NET y en la que se describe con algo más de detalle el modo de trabajo con dnSpy. Bien, como ya tenemos la víctima cargada, ponemos un BP en la línea donde, por ejemplo, se le eliminan los guiones al serial truco. Corremos el entorno y se ejecuta el BP:



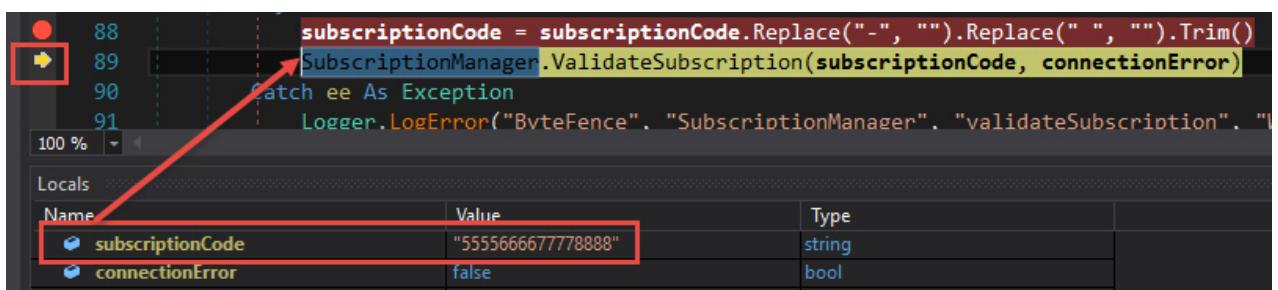
```

84
85     Friend Shared Function validateSubscription(subscriptionCode As String, <System.Runtime.InteropServices>
86         connectionError As Boolean) As SubscriptionManager.SubscriptionStatuses
87         connectionError = False
88         Try
89             subscriptionCode = subscriptionCode.Replace("-", "").Replace(" ", "").Trim()
90             SubscriptionManager.ValidateSubscription(subscriptionCode, connectionError)
91             Catch ee As Exception
92                 Logger.LogError("BvteFence", "SubscriptionManager", "validateSubscription", "Validation failed for serial code: " & ee.Message)
93             End Try
94         End Function

```

Locals			
Name	Type	Value	
subscriptionCode	string	"5555-6666-7777-8888"	
connectionError	bool	false	
ee	System.Exception	null	

Como ven, gracias al BP y después de pulsar el botón para comprobar el serial, el entorno se interrumpió en la línea donde justo se van a eliminar guiones y espacios. Además, tenemos los que contiene cada variable en la ventana de locales. Hago un F10 para ejecutar esa línea y ya se eliminaron los guiones:



```

88     subscriptionCode = subscriptionCode.Replace("-", "").Replace(" ", "").Trim()
89     SubscriptionManager.ValidateSubscription(subscriptionCode, connectionError)
90     Catch ee As Exception
91         Logger.LogError("BvteFence", "SubscriptionManager", "validateSubscription", "Validation failed for serial code: " & ee.Message)
92     End Try
93 End Function

```

Locals			
Name	Type	Value	
subscriptionCode	string	"5555666677778888"	
connectionError	bool	false	

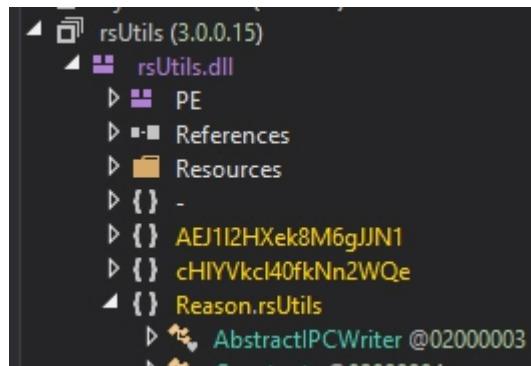
La flechita amarilla indica la línea de ejecución y ven ahí que ya se eliminaron los guiones al serial. Ahora, justo se entraría con ese serial y el false de connectionError a la función de validación. Si le pegamos una mirada:

```

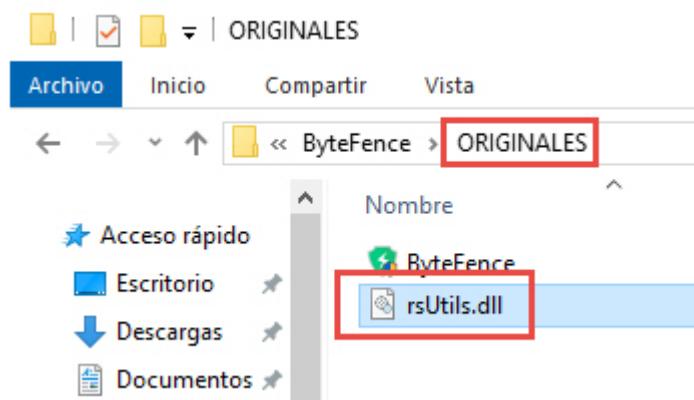
<MethodImpl(MethodImplOptions.NoInlining)>
Friend Function ValidateSubscription(SubscriptionCode As String, <System.Runtime.InteropServices>
    connectionError As Boolean) As SubscriptionManager.SubscriptionStatuses
    Return CType(Nothing, SubscriptionManager.SubscriptionStatuses)
End Function

```

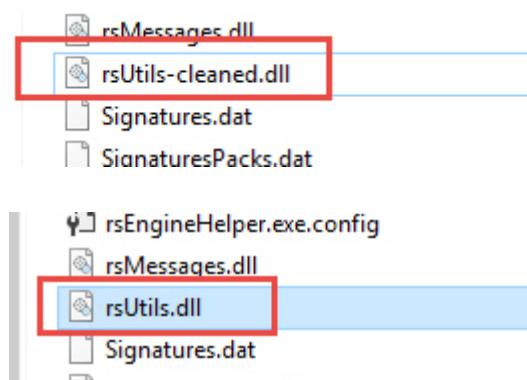
Uuupsss, ¿qué pasa aquí? No se ve nada jajajaja. Según esto, aquí prácticamente no se hace nada, pero eso no es así. Eso ocurre porque está ofuscado/protectoro. Esta función pertenece a la dll rsUtils que es utilizada por el ejecutable principal de ByteFence:



Esta dll obviamente también se instaló con ByteFence junto a otras. Lo que toca ahora es meterla en el de4dot para que la limpie del NetReactor y poder seguir con claridad. Cierro todo, la limpio con de4dot y la dll resultante la meto en la carpeta donde se instaló. Guardo la rsUtils original dentro de la carpeta que creé para los originales y continúo con el trabajo:



Meto la dll original en la carpeta creada para tal fin para cubrirnos un poco las espaldas por si algo sale mal. Ya limpié la dll con de4dot y la meto en la carpeta de instalación y renombro quitándole la coletilla “-cleaned” como ya expliqué antes:



Listo. Ya renombré y podemos seguir investigando. Vuelvo a abrir dnSpy y echamos un vistazo:

```
Friend Function ValidateSubscription(SubscriptionCode As String, <System.Runtime.InteropServices.O
Boolean) As SubscriptionManager.SubscriptionStatuses
    connectionError = False
    If Not Manager.CheckIsInitialized(True, "ValidateSubscription") Then
        Return SubscriptionManager.SubscriptionStatuses.None
    End If
    Try
        SubscriptionCode = SubscriptionCode.Replace("-", "").Replace(" ", "").Trim()
        If SubscriptionManager.ValidSubscriptionCode(SubscriptionCode) Then
            Dim flag As Boolean = False
            Dim dateT As DateTime = DateTime.MinValue
            Dim flag2 As Boolean = Manager.Product = "BF" OrElse Manager.Product = "AR"
            Try
                SubscriptionCode = SubscriptionCode.Replace("-", "").Replace(" ", "").Trim()
                If Not SubscriptionCode.Contains("-") AndAlso SubscriptionCode.Length = 16 Then
                    SubscriptionCode = String.Concat(New String() { SubscriptionCode(0).ToString()
                        SubscriptionCode(2).ToString(), SubscriptionCode(3).ToString(), "-", Subscri
                        (5).ToString(), SubscriptionCode(6).ToString(), SubscriptionCode(7).ToString()
                        SubscriptionCode(9).ToString(), SubscriptionCode(10).ToString(), Subscriptio
                        SubscriptionCode(12).ToString(), SubscriptionCode(13).ToString(), Subscription
                        (15).ToString() })
                End If
                Dim text As String = ""
                T
```

Esto ya es otra cosa amigos jejeje. Así se trabaja a gusto. Bien, gracias al limpiado podemos ver el código en todo su esplendor. En esta función se vuelve a eliminarle guiones y espacios al serial. Digo yo, qué trabajo inútil porque a esta función precisamente, ya se le entrega el serial limpio sin guiones y posibles espacios, pero bueno: cosas del programador que quiso asegurarse y ponerle doble filtro. En fin, que la premisa aquí es que al inicio del bloque Try, se evalúa en la sentencia si el serial es válido y de eso se encarga la función ValidSubscriptionCode:

```
Friend Function ValidSubscriptionCode(SubscriptionCode As String) As Boolean
    If Not Manager.CheckIsInitialized(True, "ValidSubscriptionCode") Then
        Return False
    End If
    SubscriptionCode = SubscriptionCode.Replace("-", "").Replace(" ", "").Trim()
    Return SubscriptionCode.Length = 16 AndAlso Not(SubscriptionCode = "1111222233334444") AndAlso Not(SubscriptionCode =
        "00000000000000") AndAlso Not(SubscriptionCode = "44443333222111") AndAlso Not(SubscriptionCode = "0001222200034444")
    AndAlso Not(SubscriptionCode = "122212221222122") AndAlso Not(SubscriptionCode = "1144114411441144") AndAlso Not
    (SubscriptionCode = "5566556655665566") AndAlso Not(SubscriptionCode = "058451373329119") AndAlso StringCrypt.vsc
    (SubscriptionCode)
End Function
```

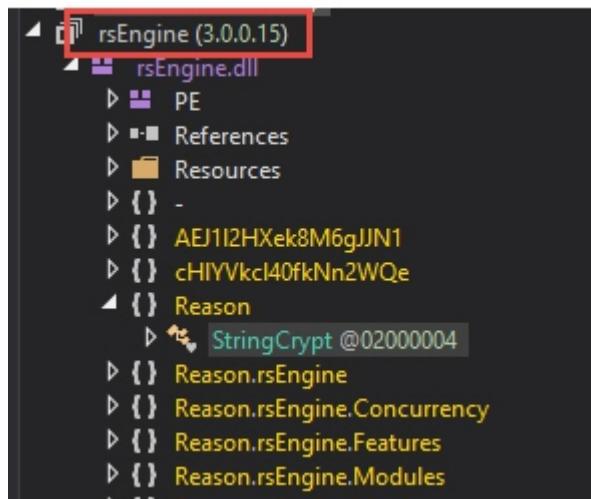
Es una función que devuelve un booleano. Se entra en ella recordarán con el serial sin guiones y se utiliza AndAlso Not primero como cortocircuito de modo que si detecta que el serial (sin guiones) y que está en la variable SubscriptionCode es igual a los que van apareciendo ahí, no seguiría comprobando el resto de seriales y saldría directamente con False. Como no coincide obviamente con ninguno, el efecto cortocircuito no hace efecto y como última comprobación, entrar en la función vsc con la cadena numérica que en este momento es “5555666677778888”. Echamos un vistazo a la función vsc:

```

<MethodImpl(MethodImplOptions.NoInlining)>
Friend Shared Function vsc(c As String) As Boolean
    Return True
End Function

```

Nos volvemos a topar con lo de antes: no se ve nada. Otra protección de NetReactor. La función vsc está en otra dll utilizada por el programa: la rsEngine.dll:



Así que repito los pasos anteriores de limpieza y sustitución y probamos de nuevo:

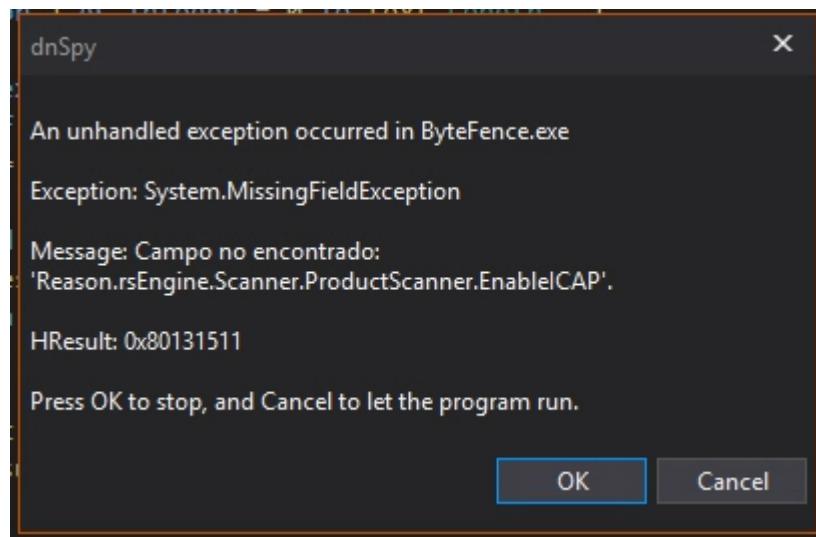
```

Friend Shared Function vsc(c As String) As Boolean
Try
    c = c.Replace("-", "").Replace(" ", "")
    Dim source As IEnumerable(Of Char) = c.ToArray().Reverse(Of Char)()
    Dim num As Integer = 0
    For i As Integer = 0 To source.Count(Of Char)() - 1
        Dim num2 As Integer = Convert.ToInt32(source.ElementAt(i).ToString())
        If(i + 1) Mod 2 = 0 Then
            Dim text As String = (num2 * 2).ToString()
            num2 = 0
            For j As Integer = 0 To text.Length - 1
                num2 += Convert.ToInt32(text.ElementAt(j).ToString())
            Next
        End If
        num += num2
    Next
    If num Mod 10 = 0 Then
        Dim result As Boolean = True
        Return result
    End If
Catch
    Dim result As Boolean = False
    Return result
End Try
Return False
End Function

```

Podemos seguir. En esta pequeña maraña de código lo que se hace resumiendo es

crear un array de la cadena numérica, pero dándole la vuelta con Reverse de modo que nos quedaría por ejemplo “8888777766665555”. Acto seguido, viene un bucle For--Next en el que la variable i va tomando los valores desde 0 hasta el largo total del serial recién volteado menos 1, es decir, 15 iteraciones. A cada vuelta del bucle principal, la variable num2 va a ir tomando el valor de cada carácter de la cadena principal pero pasados a Int32. Recordemos que es una string y de uno en uno serían caracteres. Luego viene un bloque IF que lo que hace principalmente es contar de dos en dos gracias a la operación Mod 2. Así de esta forma se va a trabajar con los índices pares de la cadena. En la primera vuelta del bucle principal, el cuento empieza en 0: $0+1=1$ y $1 \text{ Mod } 2$ NO es cero así que, en la siguiente vuelta del bucle, cuando la variable i sea 1, $1+1=2$ y $2 \text{ Mod } 2$ SÍ es 0 por lo tanto, se toma la variable text y se le asigna el valor que vaya conteniendo la variable num2 multiplicada por 2 y convertida a string. Centrándonos en los índices del bucle principal+1, “**8888777766665555**” que señalé en verde, la variable text va a ir tomando los valores 16,16,14,14,12,12,10 y 10. Me hubiera encantado mostrarles en dnSpy cómo van tomando los diferentes valores las variables, pero ésta última dll que pase por el limpiador para NetReactor, parece ser que no lo pudo limpiar bien y cuando arranco dnSpy me salta este mensaje de error:



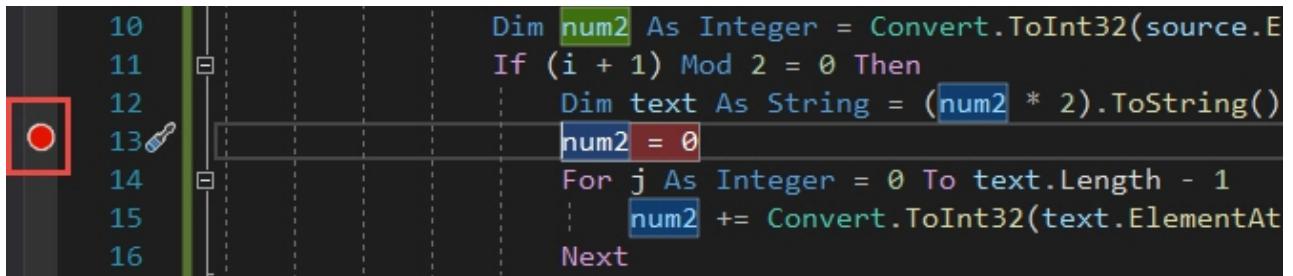
Se ve que la limpieza no se hizo bien y falta un campo. Después de investigar un poco me di cuenta que no solo es uno, sino varios campos los que faltan. Sería un trabajo enorme ir añadiendo uno a uno los campos faltantes y veo que no merece la pena. Llegado a este punto, tuve que volver a restaurar la dll original y olvidarme de utilizar la rsEngine-cleaned en dnSpy puesto que no puedo depurar. Se puede mantener también la cleaned en dnSpy para su estudio en estático. Todo este código lo extrapolé a Visual Studio y allí si podemos ver el funcionamiento del mismo:

```

Sub Main()
    Try
        Dim c As String = "555566667778888"
        c = c.Replace("-", "").Replace(" ", "")
        Dim source As Char() = c.Reverse.ToArray
        Dim num As Integer = 0
        For i As Integer = 0 To source.Count - 1
            Dim num2 As Integer = Convert.ToInt32(source.ElementAt(i).ToString())
            If (i + 1) Mod 2 = 0 Then
                Dim text As String = (num2 * 2).ToString()
                num2 = 0
                For j As Integer = 0 To text.Length - 1
                    num2 += Convert.ToInt32(text.ElementAt(j).ToString())
                Next
            End If
            num = num + num2
        Next
        If num Mod 10 = 0 Then
            Dim result2 As Boolean = True
        End If
    Catch
        Dim result3 As Boolean = False
    End Try
    Dim result4 As Boolean = False
End Sub

```

Aquí ya exporté el código en Visual Studio y para que funcione, le metí el serial trucho con el que venimos trabajando. Coloqué un BP en la línea 13:



Si se fijan en la vista general del código, el resultado final debe ser True en la variable `result2` y eso solo ocurrirá si el contenido de la variable `num Mod 10` es igual a 0. Según sea el índice del serial trucho con el que se trabaje, hará unas operaciones u otras. Cuando se ejecute el código y se llegue a la comprobación de la variable `num`, se decidirá si el retorno es True (debe ser True). Lo vemos en la imagen en Visual Studio:

```

    Next
    If num Mod 10 = 0 Then
        Dim result2 As Boolean = True
    End If
    Catch

```

Variables locales:

Nombre	Valor	Tipo
c	"5555666677778888"	String
num	84	Integer
result4	False	Boolean
source	{Length=16}	Char()

Después de tanta operación, la variable num = 84 y obviamente el resultado del Mod 10 NO es cero y por lo tanto no obtenemos el deseado True de retorno. Llegados a este punto, nos valdría como serial que pasaría el filtro, cualquiera que después de trabajarla nos diera en la variable num 30, 40, 50, etc. y aquí ahora mismo tenemos 84. Ok, si queremos por ejemplo en la variable num un 80, restamos de uno en el serial y ven el resultado:

```

    Next
    If num Mod 10 = 0 Then
        Dim result2 As Boolean = True
    End If
    Catch
        Dim result3 As Boolean = False
    End Try

```

Variables locales:

Nombre	Valor	Tipo
c	"5555666677778884"	String
num	80	Integer
result4	False	Boolean
source	{Length=16}	Char()

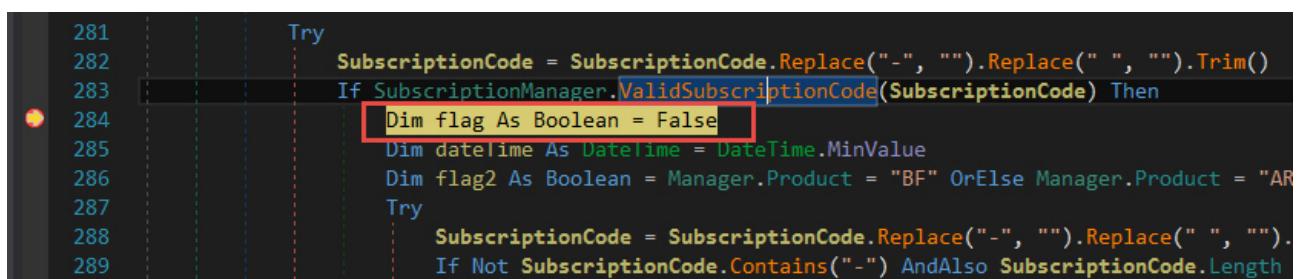
Bien, modifíco el serial trucho y le resto 4 al último porque recuerden, en el código se le da la vuelta y se opera primero con él. Por ello, obtengo un 80 y al hacerle un Mod 10, obvio obtengo 0 y ahora sí, tengo el True que me faltaba. Por si se han perdido un poco y creo que sí, éste True nos hacía falta aquí:

```

Friend Function ValidSubscriptionCode(SubscriptionCode As String) As Boolean
    If Not Manager.CheckIsInitialized(True, "ValidSubscriptionCode") Then
        Return False
    End If
    SubscriptionCode = SubscriptionCode.Replace("-", "").Replace(" ", "").Trim()
    Return SubscriptionCode.Length = 16 AndAlso Not(SubscriptionCode = "1111222233334444") AndAlso Not(SubscriptionCode =
        "0000000000000000") AndAlso Not(SubscriptionCode = "444433332221111") AndAlso Not(SubscriptionCode = "0001222200034444")
        AndAlso Not(SubscriptionCode = "1222122212221222") AndAlso Not(SubscriptionCode = "1144114411441144") AndAlso Not
        (SubscriptionCode = "5566556655665566") AndAlso Not(SubscriptionCode = "0584511373329119") AndAlso StringCrypt.vsc
        (SubscriptionCode)
End Function

```

Recuerden, StringCrypt.vsc. Se cumple con True y no es ninguno de esos serials. Bien, hasta ahora hemos estado un poco con Visual Studio y buscando un serial aceptable de modo que el serial trucho que en principio pasaría el filtro de ByteFence sería “5555666677778884”. Abro dnSpy, cargo ByteFence y le meto dicho serial:



En la imagen, pasó con éxito la linea 283 pasando el filtro con ese serial y se comienzan a hacer otro tipo de acciones. Se prepara para conectarse al servidor del propietario del software:

```

Dim text2 As String
If Manager.Product.ToUpper() = "BF" Then
    text2 = "https://license.bytefence.com/api/v1/license/"
ElseIf Manager.Product.ToUpper() = "AR" Then
    text2 = "https://license.anti-ransomware.co/api/v1/license/"
ElseIf Manager.Product.ToUpper() = "MB" AndAlso Settings.GetSettingString("AVCLAM") = "1" Then
    text2 = "https://license.malwarebusterpremium.reasonsecurity.com/api/v1/license/"
ElseIf Manager.Product.ToUpper() = "MB" Then
    text2 = "https://license.malwarebuster.reasonsecurity.com/api/v1/license/"
Else
    text2 = "https://license.reasonsecurity.com/api/v1/license/"
End If
Dim text3 As String = text2 + SubscriptionCode + "/ruserid/" + Manager.UUID

```

flag2	true
V_3	'4'
text	"
webClientEx	{Reason.rsEngine.Utilities.NetUtils+WebClientEx}
currentProxyType	None
text2	"https://license.bytefence.com/api/v1/license/"
text3	"https://license.bytefence.com/api/v1/license/5555-6666-7777-8884/ruserid/e9f28b95-6f7f-4690-8c04-7be910cb9248"

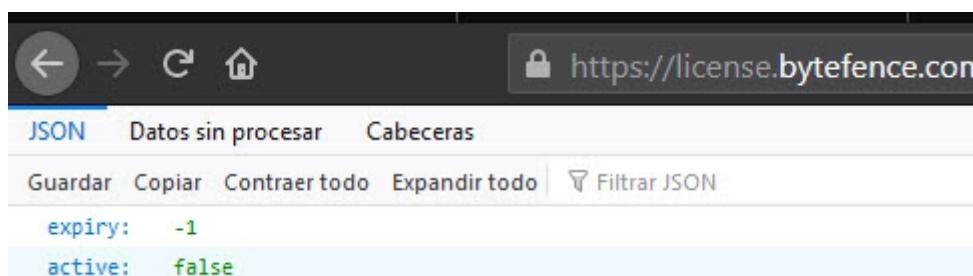
Toma una string compuesta por la dirección de conexión al servidor, el serial que le pusimos y ciertos datos que toma del registro. Hace la petición y se obtiene la respuesta por parte del servidor:

```

325     text = webClientEx.DownloadString(New Uri(text3)).Trim()
326     If Manager.DebugMode Then
327
100 %
Locals
Name Value
text "{\"expiry\":-1,\"active\":false}"
webClientEx {Reason.rseEngine.Utilities.NetUtils+WebClientEx}
currentProxyType None
text2 "https://license.bytefence.com/api/v1/license/"
text3 "https://license.bytefence.com/api/v1/license/5555-6666-7777-8884/ruserid/e9f28b95-6f7f-4690-8c0"
manualProxy null

```

Si se coloca la consulta en el navegador, lo vemos igual:



Seguimos algo mas con el código:

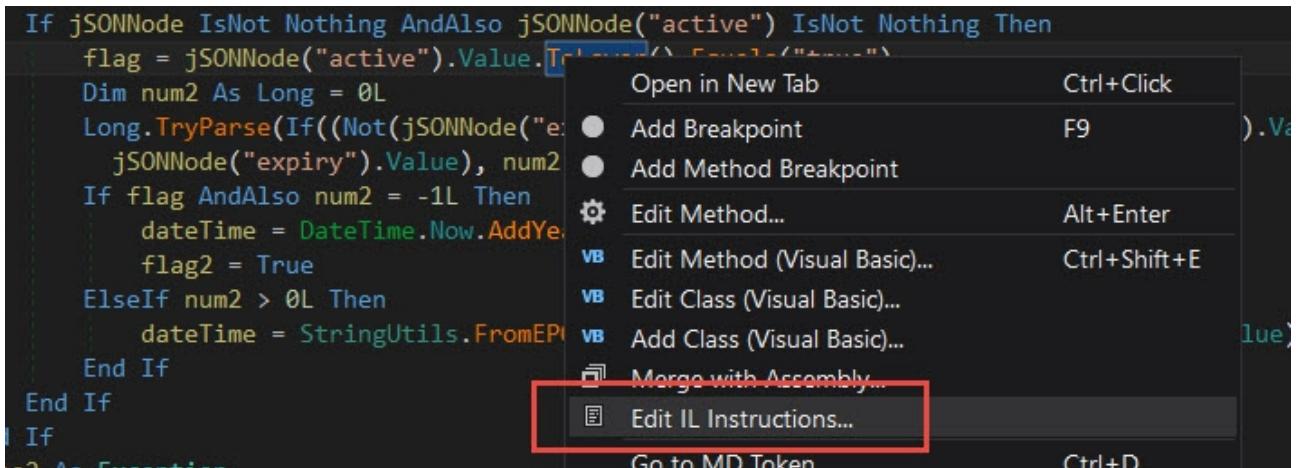
```

362 Dim jSONNode As JSONNode = JSON.Parse(text)
363 If jSONNode IsNot Nothing AndAlso jSONNode("active") IsNot Nothing Then
364     flag = jSONNode("active").Value.ToLower().Equals("true")
365     Dim num2 As Long = 0L
366     Long.TryParse>If((Not(jSONNode("expiry") IsNot Nothing) OrElse jSONNode("expiry").Value.Length <= 0), "-1",
367     jSONNode("expiry").Value), num2)
368     If flag AndAlso num2 = -1L Then
369         dateTime = DateTime.Now.AddYears(30)

```

Como la respuesta del servidor es un JSON, se prepara una variable para tal fin y se hace la comprobación de si el valor del nodo “active” es igual a “true” y el resultado se le asigna como booleano al flag. Ahora mismo, en la respuesta del servidor está en false así que el flag quedará en false. En las lineas 367 y 368 se comprueba si el flag es true (ahora mismo NO) y además, si en la variable num2 tenemos -1 (cosa que sí). Para que se cumpla por completo y al dateTime se le añadan 30 años, debe estar en True y solo se me ocurre modificar el código. Lo hago parando la depuración (muy importante para que los cambios se reflejen):

Botón derecho de la rata, edito la linea (código IL) y guardo los cambios:



```
432 04E3 callvirt instance string [rsEngine]Reason.rsEngine.Utilities.JSONUtils.JSONNode::get_Value()
433 04E8 callvirt instance string [mscorlib]System.String::ToLower()
434 04ED ldstr    "true"
435 04F2 callvirt instance bool [mscorlib]System.String::Equals(string)
436 04F7 bnttrue s 439 (04EC) ldc i4 1
```

```
431 04DE callvirt instance class [rsEngine]Reason.rsEngine.Utilities.JSONUtils.JSONNode
432 04E3 callvirt instance string [rsEngine]Reason.rsEngine.Utilities.JSONUtils.JSONNode::get_Value()
433 04E8 callvirt instance string [mscorlib]System.String::ToLower()
434 04ED ldstr    "false"
435 04F2 callvirt instance bool [mscorlib]System.String::Equals(string)
```

Ya modifiqué el código IL simplemente reescribiendo a false. Guardo los cambios y vuelvo a ejecutar.

```
363
364     If jSONNode IsNot Nothing AndAlso jSONNode("active") IsNot Nothing Then
365         flag = jSONNode("active").Value.ToLower().Equals("false")
366         Dim num2 As Long = 0L
367         Long.TryParse(If((Not(jSONNode("expiry") IsNot Nothing) OrElse jSONNode(
368             jSONNode("expiry").Value), num2)
369             If flag AndAlso num2 = -1L Then
370                 dateTime = DateTime.Now.AddYears(30)
```

Ahora, justo en el breakpoint, el flag será True porque efectivamente el valor de la respuesta del servidor lo es. Ejecuto dicha linea:

```
363
364     If jSONNode IsNot Nothing AndAlso jSONNode("active") IsNot Nothing Then
365         flag = jSONNode("active").Value.ToLower().Equals("false")
366         Dim num2 As Long = 0L
367         Long.TryParse(If((Not(jSONNode("expiry") IsNot Nothing) OrElse jSONNode("expiry").Value), num2)
368             If flag AndAlso num2 = -1L Then
369                 dateTime = DateTime.Now.AddYears(30)
```

Locals

Name	Value
flag	true
dateTime	01/01/0001 0:00:00

Bueno, vamos obteniendo los resultados que esperábamos. Sigo ejecutando linea a linea.....:

The screenshot shows a debugger interface with two code snippets and their corresponding local variable tables.

Code Snippet 1:

```
367
368
369 If flag AndAlso num2 = -1L Then
    dateTime = DateTime.Now.AddYears(30)
```

Locals Table 1:

Name	Value
SubscriptionCode	"5555-6666-7777-8884"
connectionError	false
flag	true
num2	-1
V_17	{01/01/0001 0:00:00}

Code Snippet 2:

```
367
368
369 If flag AndAlso num2 = -1L Then
    dateTime = DateTime.Now.AddYears(30)
    flag2 = True
```

Locals Table 2:

Name	Value
SubscriptionCode	"5555-6666-7777-8884"
connectionError	false
flag	true
dateTime	{26/01/2050 3:27:15}

Como se cumple la condición, se añaden 30 años a la fecha actual. Sigo ejecutando:

The screenshot shows a debugger interface with a code snippet and its corresponding local variable table.

Code Snippet:

```
384     Settings.SetSetting("SBAXD", StringCrypt.es(DateTime.Now.ToString("F")))
385 End If
386 Settings.SetSetting("SBEXD", StringCrypt.es(dateTime.ToString("F")))
387 Settings.setSetting("SBAKF", +flag2)
388 Settings.DeleteSetting("SBTMR")
389 Settings.DeleteSetting("SBTMPC")
```

Se dispone a guardar datos en el registro de Windows y lo hace de modo encriptado. Concretamente los guarda en:

Equipo\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\ByteFence			
	Nombre	Tipo	Datos
> VideoLAN	LNRYRUN	REG_SZ	22/11/2019 17:39:31
> Windows	LASTDLT	REG_SZ	1579991738
> World Machine Software LLC	LastSetting	REG_SZ	Subscription
WOW6432Node	LFFEI	REG_SZ	63710047721904722
> Adobe	LFMAIN	REG_SZ	637156047576598965
> Apple Inc.	LPMSDT	REG_SZ	637156037652342613
> ASIO	LRSQR	REG_SZ	26/01/2020 0:15:38
ASIO4ALL	LSCKTS	REG_SZ	24/11/2019 12:01:44
> ATI	MinimizedNotice	REG_SZ	1
ATI Technologies	MOLOG	REG_SZ	0
Autodesk	PINSTP	REG_SZ	/S
AviSynth	PMASS	REG_SZ	1
ByteFence	PMCP	REG_SZ	1
RTOP	PMDDP	REG_SZ	0
Caphyon	PSSET	REG_SZ	
Classes	RCFLH	REG_SZ	26/01/2020 3:30:59
Client	SBAXD	REG_SZ	p+PT2XdxZ?SlduFSMdSMVIDUf1OnIW3e75bdmifT3FVhh5mP+XFrE79zos6NkF... gAixn/6Ku7caHTl871P9hwz1MJ+wTjcgivl3eQ7azgXwPFAPM1V9vkKplDJT9tSh...
Cyberlink	SBEXD	REG_SZ	Fc7lCEhg1Ktrm+meMsbwWeNRNtqxCsReJSjvBeC/1qkQpLbn70rzlcgnT8ddPqD... 63710138200384945
DSPRobotics	SBLC	REG_SZ	
Ffmpeg for Audacity	SBTMRP	REG_SZ	
FileZilla 3			
FileZilla Client			

Aquí tienen la ruta en el registro donde la víctima manipula el registro. La fecha que se muestra en la página anterior, convertido a ticks es la siguiente:

Name	Value
OADateMaxAsDouble	2958466
OADateMinAsDouble	-657435
OADateMinAsTicks	312413760000000000
Second	15
Ticks	646623772357562512
TicksCeiling	4611686018427387904
Time	

Si esto lo extrapolo a Visual Studio:

```
Sub Main()
    Try
        Dim fecha As DateTime = New DateTime(646623772357562512)
    
```

Variables locales

Nombre	Valor	Tipo
bytes	Nothing	Byte()
fecha	#1/26/2050 3:27:15 AM#	Date
passwordDeriveBytes	Nothing	System.S
result4	False	Boolean

Y lo paso a String("F") tal y como hace la víctima:

"F" o "f"	Punto fijo	Resultado: dígitos integrales y decimales con signo negativo opcional.	1234.567 ("F", en-US) -> 1234.57
		Compatible con: todos los tipos numéricos.	1234.567 ("F", de-DE) -> 1234,57
		Especificador de precisión: número de dígitos decimales.	1234 ("F1", en-US) -> 1234.0
		Especificador de precisión predeterminado: Definido por NumberFormatInfo.NumberDecimalDigits .	1234 ("F1", de-DE) -> 1234,0
		Más información: Especificador de formato de punto fijo ("F") .	-1234.56 ("F4", en-US) -> -1234.5600 -1234.56 ("F4", de-DE) -> -1234,5600

Dim result As String

```
Sub Main()
    Try
        Dim fecha As DateTime = New DateTime(646623772357562512)
        Dim c As String = fecha.ToString("F")
    
```

Variables locales

Nombre	Valor	Tipo
bytes	Nothing	Byte()
c	"miércoles, 26 de enero de 2050 3:27:15"	String
fecha	#1/26/2050 3:27:15 AM#	Date

```
Dim c As String = fecha.ToString("F")
Dim bytes As Byte() = Encoding.Unicode.GetBytes(c)
Dim passwordDeriveBytes As PasswordDeriveBytes = New PasswordDeriveBytes("chr15j0ne5", New Byte() {73, 118, 97, 110, 32, 77, 101, 100, 118, 101, 100, 101, 118})
Using memoryStream As MemoryStream = New MemoryStream()
    Dim rijndael As Rijndael = rijndael.Create()
    rijndael.Key = passwordDeriveBytes.GetBytes(32)
    rijndael.IV = passwordDeriveBytes.GetBytes(16)
    Using cryptoStream As CryptoStream = New CryptoStream(memoryStream, rijndael.CreateEncryptor(), CryptoStreamMode.Write)
        cryptoStream.Write(bytes, 0, bytes.Length)
        cryptoStream.Close()
    End Using
    result = Convert.ToBase64String(memoryStream.ToArray())
    memoryStream.Close()
End Using
```

Ventana Inmediato

```
? result
?pNgL6kxat+4RCIPj6EkdeZ3MRgJeHiDccZOH6BoE+1PcA7hFjIJ8Z1Vwg1kmw3/500qrzXuwVszyvCI/+ci5Q6v8wQjhZ1woq2yMwzurp8c=
```

El valor de result es el valor que se va a guardar en el registro, ahí donde les señalé en la pagina anterior.

El código lo copié a Visual Studio directamente de la víctima:

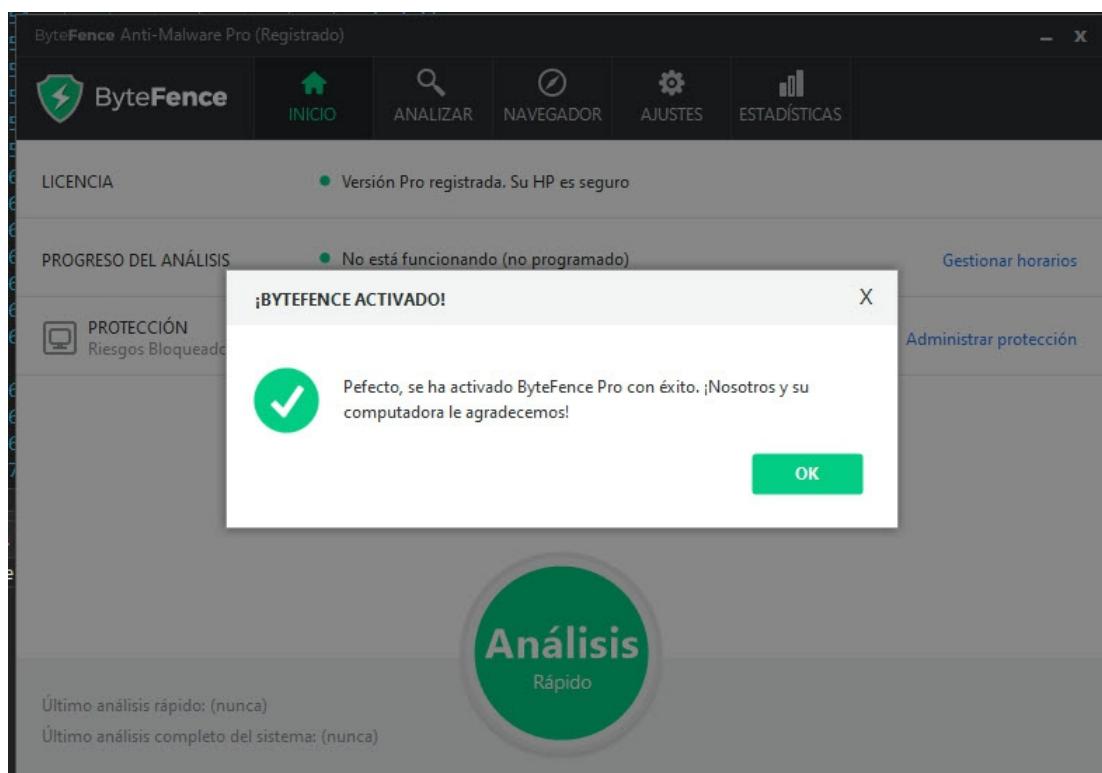
```
SubscriptionManager StringCrypt.x
10  Friend Class StringCrypt
11      ' Token: 0x02000004 RID: 4
12      Private Shared Function smethod_0(string_0 As String) As String
13          If String.IsNullOrEmpty(string_0) Then
14              Return ""
15          End If
16          If string_0.Length = 0 Then
17              Return ""
18          End If
19          Dim result As String = ""
20          Try
21              Dim bytes As Byte() = Encoding.Unicode.GetBytes(string_0)
22              Dim passwordDeriveBytes As PasswordDeriveBytes = New PasswordDeriveBytes("chr15j0ne5", New Byte() { 73, 118, 97, 110, 32,
23                  77, 101, 100, 118, 101, 100, 101, 118 })
24              Using memoryStream As MemoryStream = New MemoryStream()
25                  Dim rijndael As Rijndael = Rijndael.Create()
26                  rijndael.Key = passwordDeriveBytes.GetBytes(32)
27                  rijndael.IV = passwordDeriveBytes.GetBytes(16)
28                  Using cryptoStream As CryptoStream = New CryptoStream(memoryStream, rijndael.CreateEncryptor(), CryptoStreamMode.Write)
29                      cryptoStream.Write(bytes, 0, bytes.Length)
30                      cryptoStream.Close()
31                  End Using
32                  result = Convert.ToBase64String(memoryStream.ToArray())
33                  memoryStream.Close()
34          End Try
```

Ejecuto la linea donde estaba parado el depurador en dnSpy:

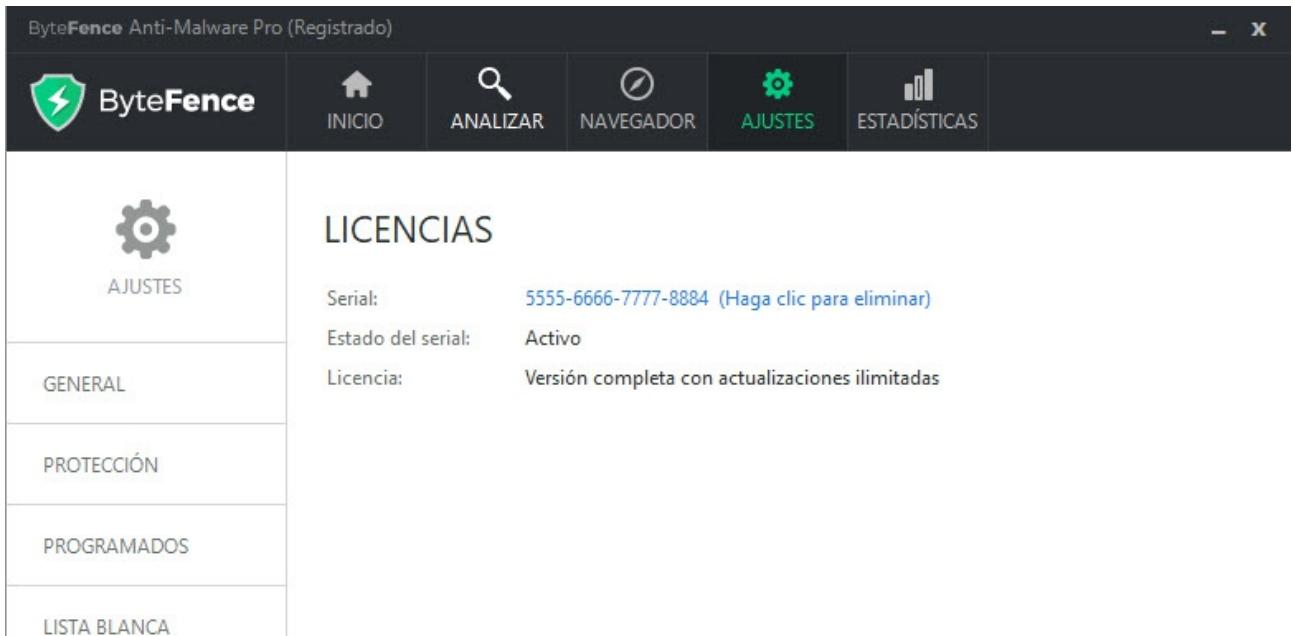
```
386 Settings.SetSetting("SBEXD", StringCrypt.es(dateTime.ToString("F")))
387 Settings.SetSetting("SBARF", flag2)
388 Settings.DeleteSetting("SBTMR")
```

ab SBAXU	REG_SZ	p+P12XdxZ2SlduHSMdSMVIDUH1OnJW3e/bbdmif13FYbb3mP+XErE/YzosbNkf...
ab SBEXD	REG_SZ	pNgL6kxat+4RCIPj6EkdeZ3MRgJeHiDccZOH6BoE+IPcA7hFjIJ8Z1VWg1kmW3/...
ab SBLC	REG_SZ	Fc7lCEHg1Ktrm+meMsbWeNRNtqxCsReJSJvBoC/1qkQpLbn70rzJcgnt8ddPqD...

Justo lo que les mostré en V.Studio qué se iba a guardar encriptado. Termino de ejecutar el resto del código y la víctima:



Cierro dnSpy, cierro también ByteFence incluso de la barra de tareas, pues después de cerrado se queda ejecutando de fondo, lo abro de nuevo y efectivamente arranca como se esperaba:



Bueno y ya sí que podemos dar por terminado el trabajo. Si alguien va a utilizar este software, recuerden después de hacer las modificaciones al binario ó binarios, deshabilitar que lo actualice. Si modifican una dll por ejemplo y resulta que la siguiente actualización se lo carga y reemplaza pues.....mal asunto: a empezar de nuevo.

Otro dato que descubrí mirando las entrañas de la víctima, es que en el caso de que sigas en modo Trial y te queden pocos días (imaginén que te queden tres o cuatro), el programador o programadores de ByteFence incluyeron una opción curiosa:

```
ElseIf Me.string_0 = "RESTARTMYTRIAL!" Then
    Settings.DeleteSetting("SBAXD")
    Settings.DeleteSetting("SBEXD")
    Settings.DeleteSetting("SBLIC")
    Settings.DeleteSetting("SBARF")
    Settings.DeleteSetting("SSEREM")
    SubscriptionManager.SetTrialStart(True)
    Class130.setSubscriptionUpdate()
```

Ese trozo de código hace que, si en el campo de texto donde se coloca el serial para activarlo, se pone la palabra “RESTARTMYTRIAL!” sin las comillas obviamente, ByteFence se te pone de nuevo con sus 14 días originales de trial y tener otra vez así más tiempo de testeo. La verdad es que desconozco si el fabricante, bajo pedido, te amplía éste trial por una módica limosna. No lo sé, la verdad.

Otra sugerencia, es que mientras juegueten con ByteFence en dnSpy y quieran hacer sus experimentos, es que deshabiliten en los servicios precisamente a ByteFence:

 BTDevManager	REALTEK Bluetooth Se...	En ejec...	Automático
 ByteFence Anti-Malware Service	ByteFence Anti-Malw...		Deshabilitado
 ByteFence Real-time Protection	The ByteFence Real-ti...		Deshabilitado

El tema está que ByteFence, aunque lo cierren, cuando pasa un tiempo se vuelve a reactivar. Es un problema eso porque mientras estudias la víctima, para cuando te des cuenta se a vuelto a ejecutar en segundo plano y te llegas a liar un poco con lo que estás depurando y lo que ya se ejecutó. Cuando lo tengan ya finalmente funcionando y modificado a sus gusto, repongan el estado de los servicios y ya está. Y por ahora, que yo sepa, no recuerdo nada más así de interés que deba compartirles puesto que al final, cuando estudias una víctima, llegas a ver bastantes cosas sobre su funcionamiento.

Ya por último, quien no tenga las herramientas con las que suelo andar, lo dicen bien por el grupo de CracksLatinos ó por Telegram (donde estoy hace relativamente poco tiempo) y ya busco la forma de colgarlas en algún sitio para compartirlas.
Un saludo a todos/as, ánimos y seguir bien!!