

[muvee Reveal v13.0.0.29340 Build 3157 (.NET)(Crack-Patch)(dnSpy v5.0.7)]



Software	muvee Reveal Encore v13.0.0.29340 Build 3157
WEB	https://www.muvee.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>Detect It Easy (DIE) v1.01 dUP2 Diablo's Universal Patcher v2.26 MSIL OpCode Table v1.0</p> <p>DESCARGAR HERRAMIENTAS</p> <p>DESCARGAR TUTO+ARCHIVOS</p>
SOLUCIÓN	CRACK - PATCH
AUTOR	LUISFECAB (ADVISERS: DavicoRm, nextco, Apuromafo)
RELEASE	NOVIEMBRE 1 DE 2018 [TUTORIAL 012]

INTRODUCCIÓN

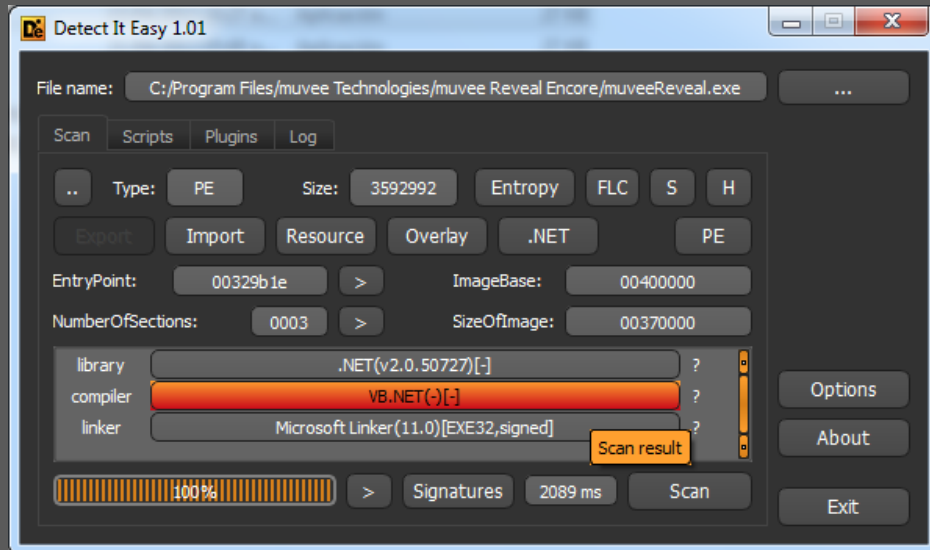
Este tutorial es el fruto de la colaboración de mis amigos de **PeruCrackerS**. **Dani** lo puso en la lista como un target para practicar y así todos podíamos compartir nuestros pocos o muchos conocimientos en reversar un **.NET**, y de paso **Dani** aprovechaba para introducirse con los **.NET**, espero especialmente que este tutorial te sea de mucha ayuda amigo **Dani**. Pasaron los días y como que esta aplicación que había propuesto **Dani** estaba olvidada en el canal, pero yo no la había olvidado cuando supe que era un **.NET** me animé a crackearla porque ya había hecho un **.NET** y me fue lo más de bien. Pensaba que era un buen momento de hacer otro y practicar.

La realidad para mi es que no fue tan sencillo como pensaba, me demoré en ir pillando el lugar de interés, este programa no se parecía en nada al otro que había hecho; así pasó más de una semana picando allí, picando allá, traciando hasta el cansancio cuando por fin llegué a una función que determina un valor para que arranque el programa como **Full** o eso pensaba yo, esto fue un gran avance para mí, yo muy contento me dije -**Ahora solo debes cambiar esa función para que me retorne el valor deseado**-. Quién dijo miedo, la voy a cambiar y tenga pa' que lleve, no pude. Estoy riéndome con ganas mientras escribo estas palabras al recordar que terminé derrotado al tratar de hacer los cambios.

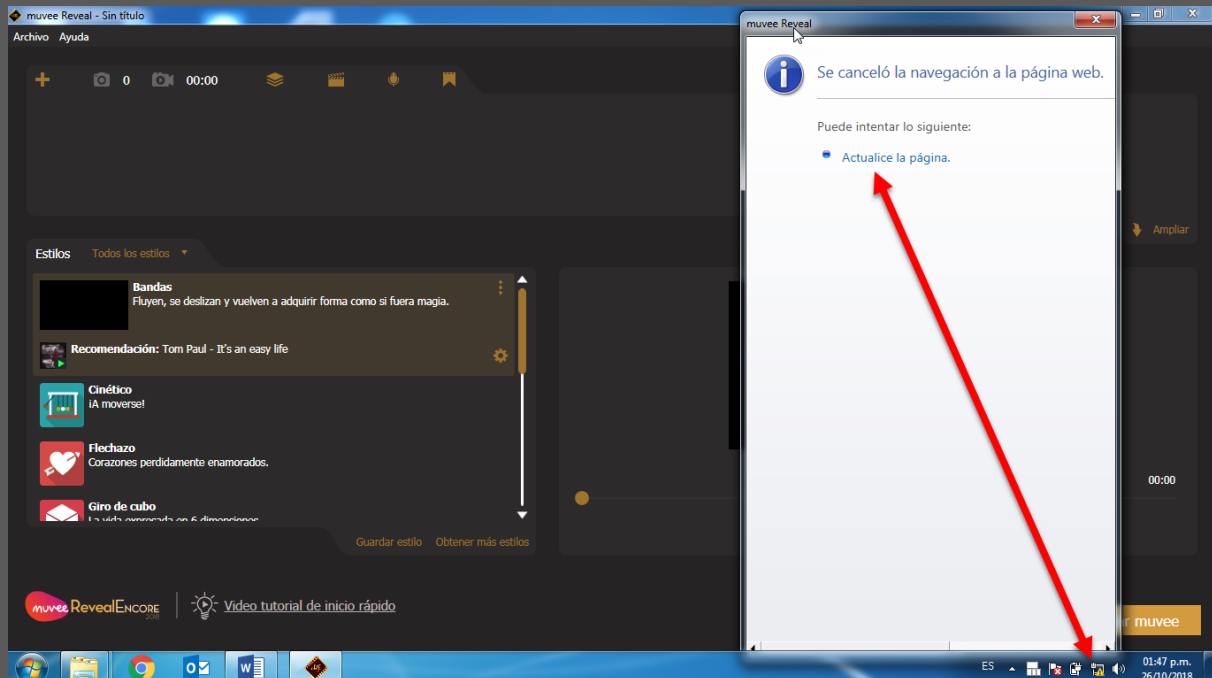
Después de que pasaran los días y que no pudiera lograr nada, me decidí a compartir mis pocos avances con mis amigos de **PeruCrackerS**, pues de ahí es que se había planteado el reto. Eso fue lo mejor que pude haber hecho porque el grupo se volcó a resolverlo con entusiasmo y de ahí se logró crackearlo, entraron en acción **Apuromafo**, **nextco** y **DavicoRm**, ellos fueron los realmente artífices de que pueda escribir este tutorial para que entre todos podamos aprender, y eso es lo que más me gusta de **PeruCrackerS**, son un grupo que está listo en ayudarte. Gracias a todos por sus buenas enseñanzas, que por mi parte son invaluable.

ANÁLISIS INICAL

Esto es nuestra rutina de siempre, echémosle una ojeadita por encima como para no perder la costumbre. Carguémoslo en el <Detect It Easy (DIE) v1.01>.

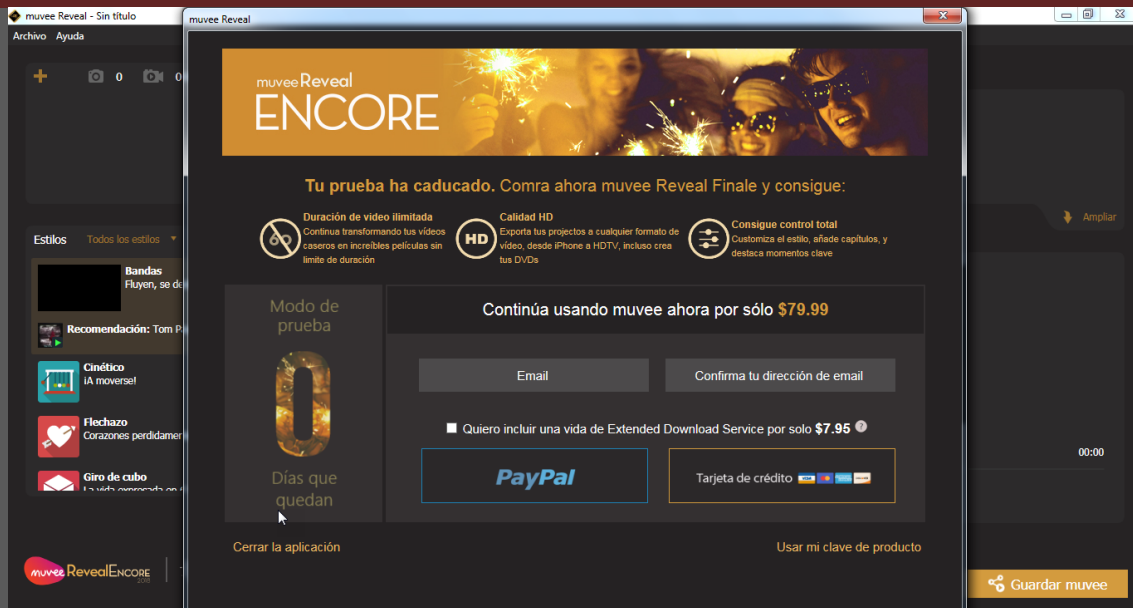


Es un **.NET**, eso ya lo sabíamos y nada de obfuscación. Ya es hora de ejecutarlo y ver qué nos muestra. Lo vamos a ejecutar con nuestro Internet desconectado.



Pues este pillo al arrancar te muestra una **NAG** que se comporta como una ventana de un navegador y como nosotros estamos desconectados de Internet no carga nada, y para rematar si nosotros cerramos esa **NAG** el programa se nos cierra, así que el programa nos obliga a que hagamos lo que la **NAG** nos diga, que por desgracia no lo sabemos por estar desconectados. Bien, conectémonos y veamos que nos muestra esta dichosa **NAG**.

[muvee Reveal v13.0.0.29340 Build 3157 (.NET)(Crack-Patch)(dnSpy v5.0.7)]



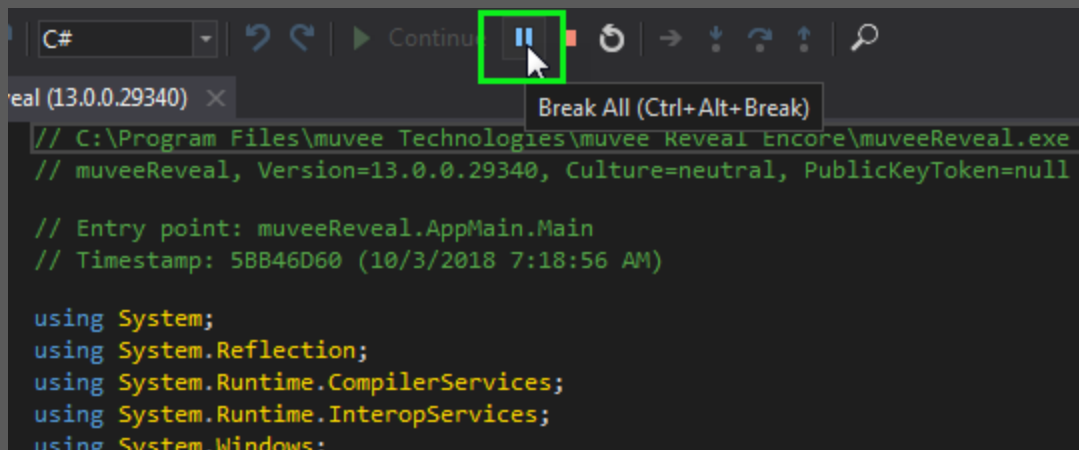
No es más que una **NAG** de aviso **TRIAL**, cuando ejecutamos el programa por primera vez y estando conectados la **NAG** nos ofrece **15** días de prueba, en mi caso ya se me pasaron mis **15** días, también nos ofrece ingresar nuestro **Serial** de activación y que será comprobado vía Internet. Eso no importa, ya verás pinche **NAG** que te mandaremos al infierno. Aquí recuerdo lo que **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.

El programa tiene un camino único que toma dependiendo del idioma que tú escojas durante la instalación, esto solo ocurre cuando lo instalas en inglés, este camino lo pilló **nextco**. Más adelante lo veremos pero quería dejar un inicio de esto aquí en nuestro **ANÁLISIS INICAL**.

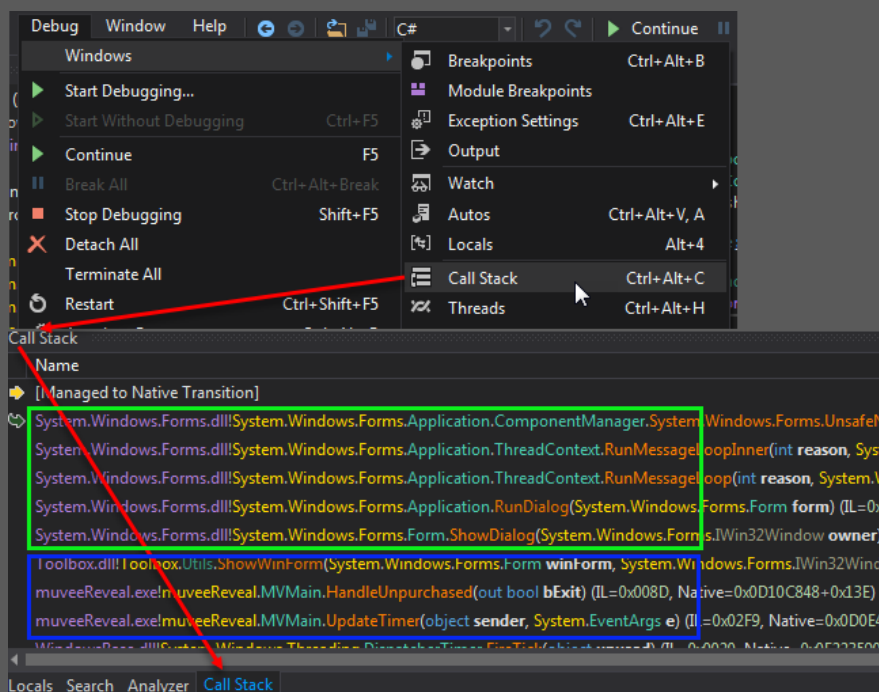
Bien, ya pudimos verlo en acción y conocer lo que nos muestra y nos pide para activarlo, ya con eso determinamos nuestro modo de ataque, le daremos desde adentro.

AL ATAQUE

Les comentaba en la **INTRODUCCIÓN** que tracié muchísimo, picando allí y por todos lados para llegar a mi "**ZONA CALIENTE**", aquí ya entraremos de lleno desde ahí. Uno puede pillar la "**ZONA CALIENTE**" traciando con paciencia, agudizando el ojo pero quiero dejar una forma más efectiva y es usar el **CALL STACK**, así como lo usamos en mi querido <OlllyDBG>, esto no es algo nuevo ni nada, pero sí lo es para mí, ahí vamos aprendiendo cómo usar más efectivamente el <dnSpy v5.0.7>. Carguemos el programa en nuestro <dnSpy v5.0.7> para empezar el ataque, ponemos a correr la aplicación hasta que aparezca la **NAG**.



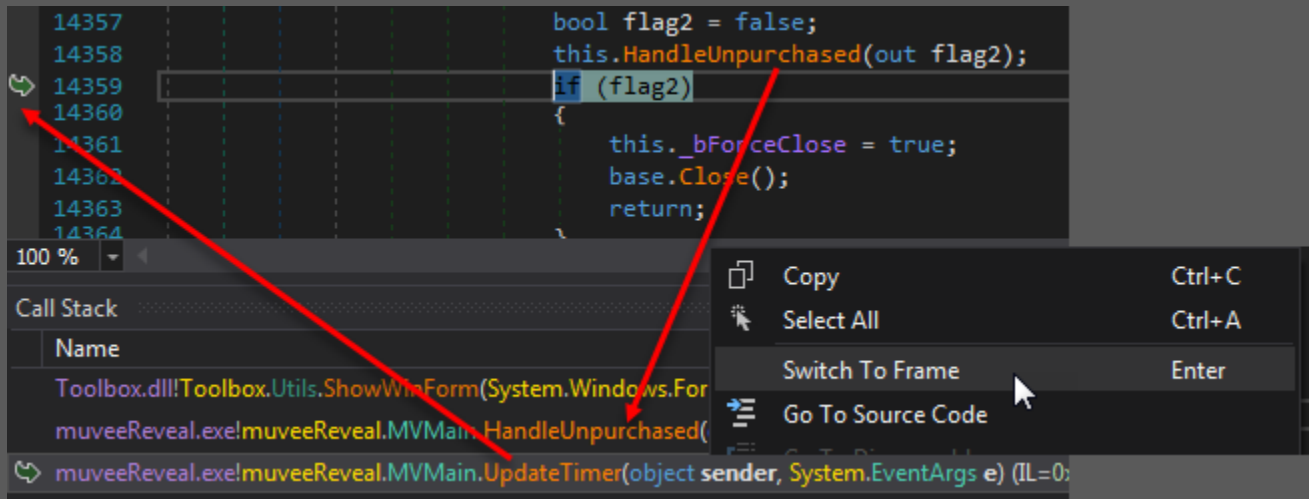
Pausamos el traceo para poder ver todas las llamadas que ha hecho, que muy seguramente en una de esas se nos carga la **NAG**. Si no tienes tu ventana **CALL STACK** abierta, la puedes activar desde <Debug->Windows->Call Stack> ó <CTRL+ALT+C>.



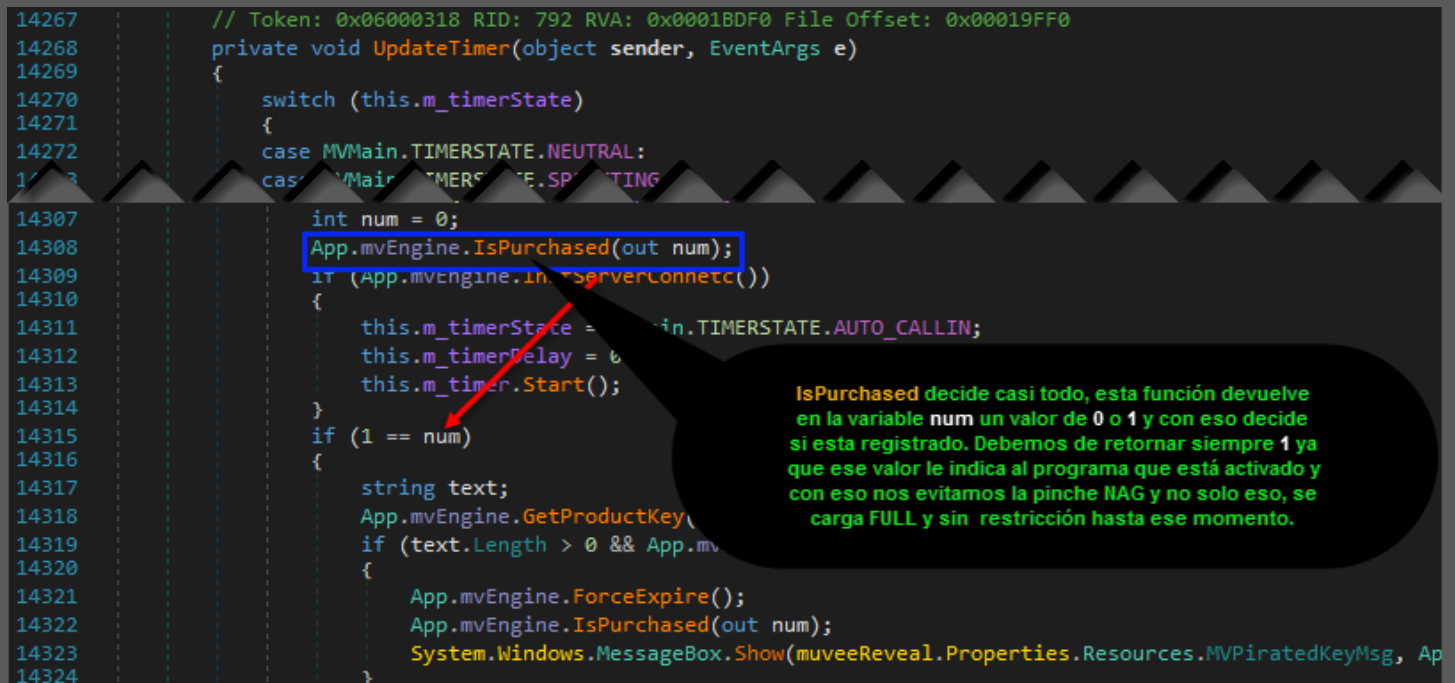
En el **RECUADRO VERDE** tenemos los últimos **CALLS's** que fueron llamados y como vemos son del sistema, estos reemplazan a las **API's** que vemos en otros lados, el lenguaje

[muvee Reveal v13.0.0.29340 Build 3157 (.NET)(Crack-Patch)(dnSpy v5.0.7)]

.NET hace uso de sus propias funciones para hacer todo lo necesario para que funcione el programa correctamente, claro que podemos usar **API's** en .NET pero se supone que para eso tiene sus propias funciones. Vemos que para mostrar nuestra **NAG** ha hecho uso de cinco **CALL's** del Sistema. Luego tenemos en el **RECUADRO AZUL** tres **CALL's** que son llamados, uno por **Toolbox.dll!Toolbox.Utils** y dos son desde nuestra aplicación, **muveeReveal.exe!muveeReveal.MVMain**. Ya es cuestión de revisar esos tres **CALL's** y ver cuál nos acerca más a la "**ZONA CALIENTE**", el análisis sigue siendo el mismo, en algún lugar habrá un salto decisivo. Recordemos que nos referimos a estos **CALL's** como funciones, que también son conocidos como **Métodos**. Pues para ir a nuestro lugar de interés, yo voy a observar el retorno del **Método UpdateTimer**, esto es igual al **Ollly**.



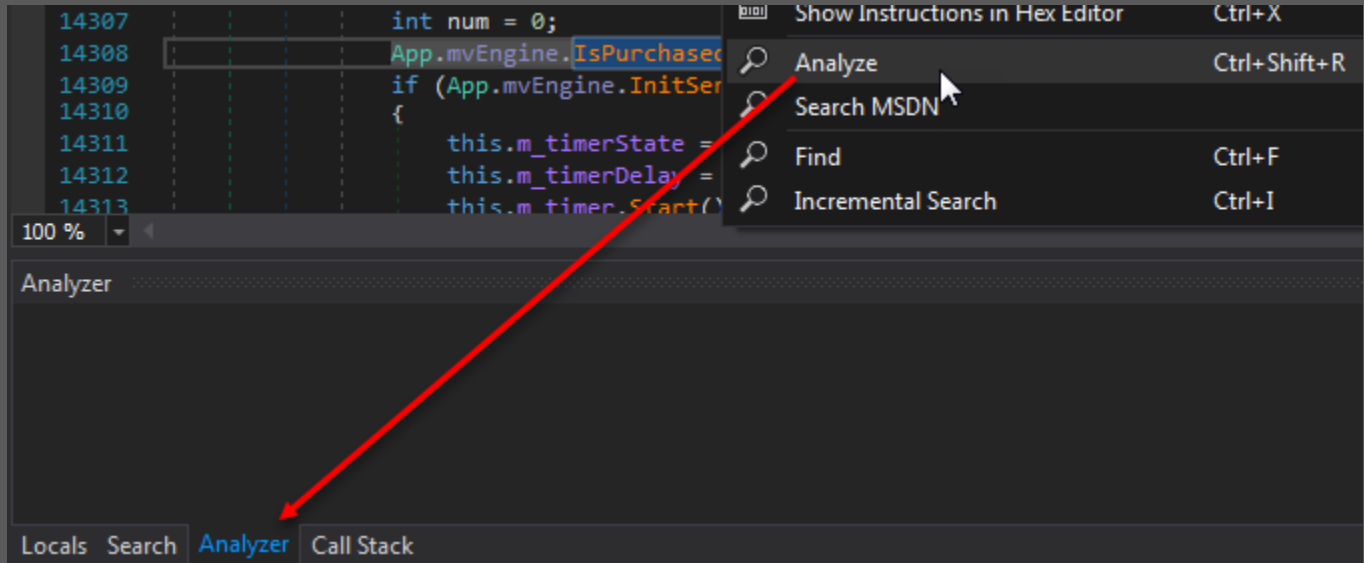
Lo ven, igual que el <**OlllyDBG**>, también con doble clic sobre el **Método** al cual queremos ver para analizar.



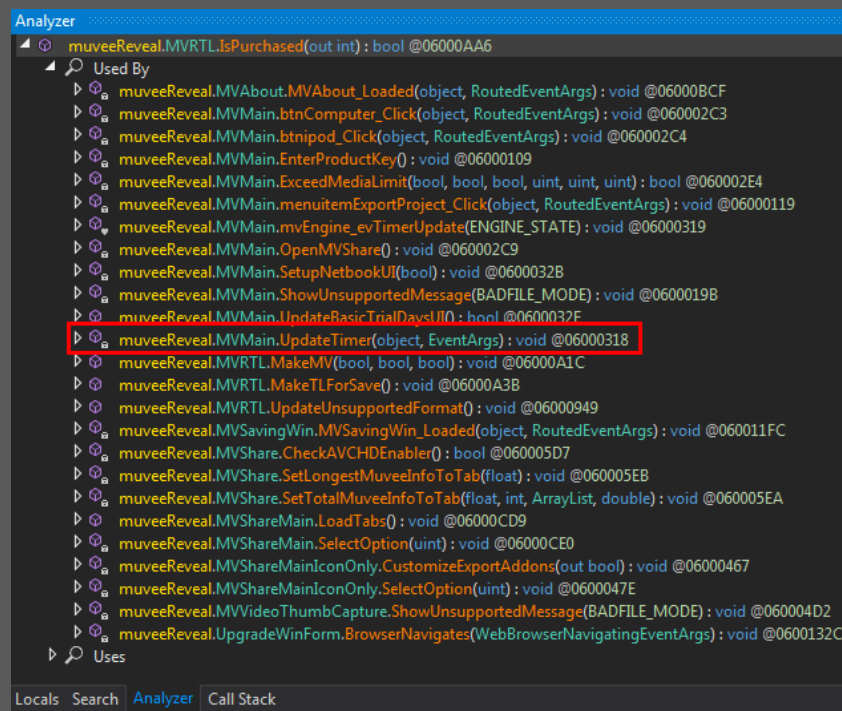
En la imagen de arriba vemos el inicio de la rutina, como es un poco larga la he cortado y he dejado la "**ZONA CALIENTE**", bueno en realidad la "**ZONA CALIENTE**" es `App.mvEngine.IsPurchased(out num);` porque es ahí donde se debe retornar el **1** para aparecer

[muvee Reveal v13.0.0.29340 Build 3157 (.NET)(Crack-Patch)(dnSpy v5.0.7)]

como activados. Más adelante entraremos en ese **IsPurchased** y haremos nuestros cambios para que retorne **1** porque **App.mvEngine.IsPurchased(out num)**; no solamente se llama desde aquí, si no como es lo normal, las validaciones siempre se hacen desde diferentes sitios y en diferentes oportunidades. Para poder ver desde dónde y cuantas veces es llamada una función podemos realizar una búsqueda de la siguiente forma con ayuda de la ventana **ANALYZER**.



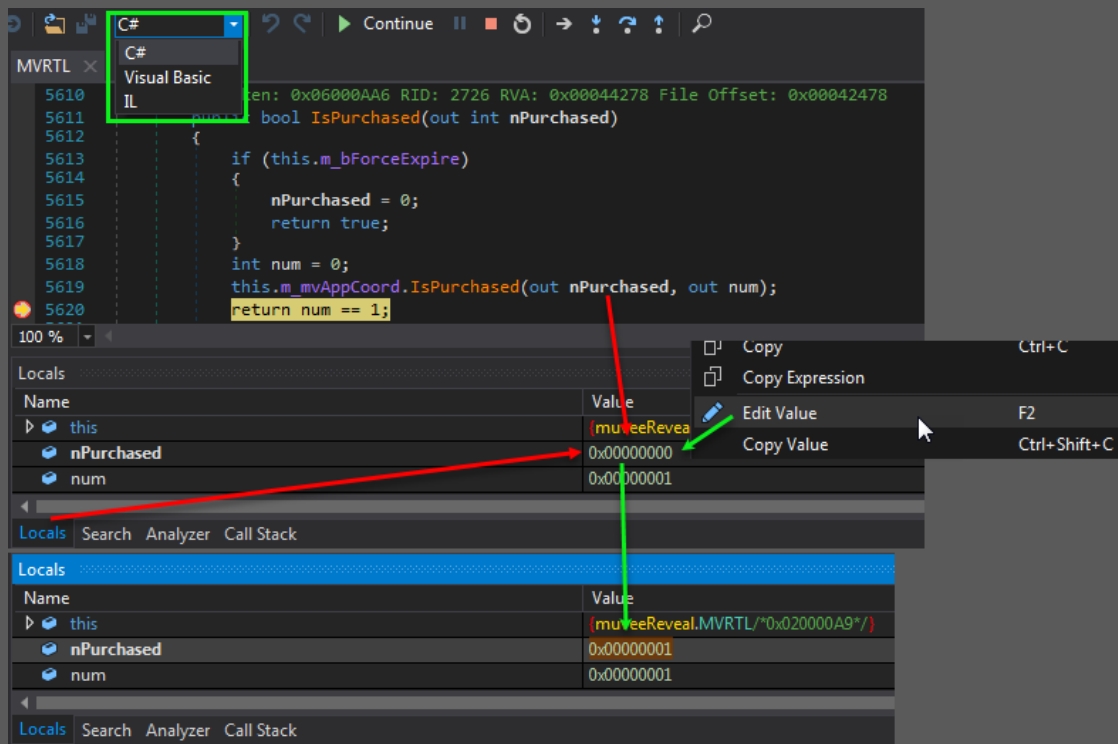
Nos posicionamos sobre la función de nuestro interés, es este caso **IsPurchased**, <**Clic Derecho**->**Analyze**>. Como vemos en la ventana **ANALYZER** no tenemos nada porque no hemos realizado la búsqueda pero una vez hecha se cargará.



Ahí podemos ver las funciones que hacen uso del **Método IsPurchased**, en el **RECUADRO ROJO** podemos ver que esta **UpdateTimer**, que es donde llegamos persiguiendo la **NAG**. Ahora si entremos a **IsPurchased**.

```
5610 // Token: 0x06000AA6 RID: 2726 RVA: 0x00044278 File Offset: 0x00042478
5611 public bool IsPurchased(out int nPurchased)
5612 {
5613     if (this.m_bForceExpire)
5614     {
5615         nPurchased = 0;
5616         return true;
5617     }
5618     int num = 0;
5619     this.m_mvAppCoord.IsPurchased(out nPurchased, out num);
5620     return num == 1;
5621 }
```

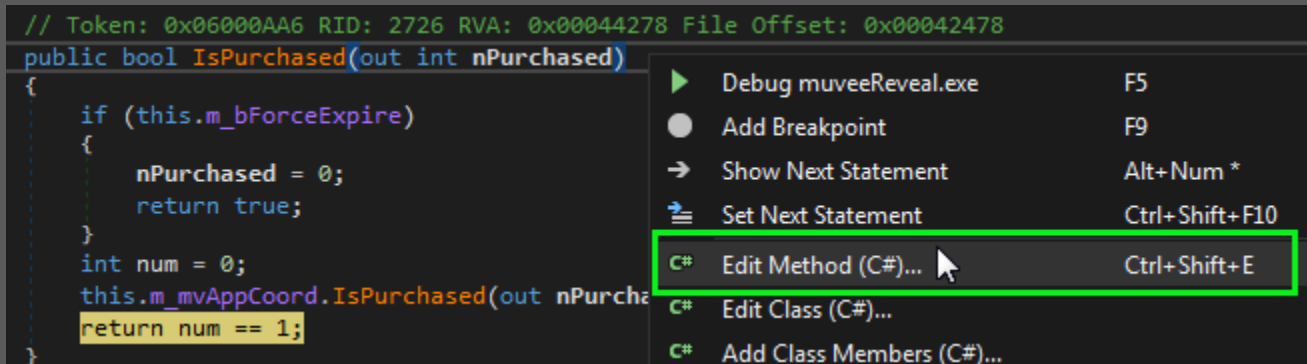
Hasta aquí fue que llegué yo solito, resulta que mis cambios no eran bien hechos y al tratar de guardarlos me daban error de compilación, y como he dicho es mis últimos tutos eso se reduce a **FALTA DE EXPERIENCIA**, pero que con cada tutorial vamos corrigiendo eso. Voy a serles franco **nextco** y **Apuromafo** me explicaron esta función de todas las formas y no pude entenderla, más o menos esta función comprueba inicialmente un número, retornando ese número y luego dependiendo de ese número hace una nueva comprobación retornando ahora un Booleano. Sabes que función del demonio no te entiendo pero lo que sí tengo claro es que la variable **nPurchased=1** para que el programa piense que esta **FULL**. Voy a colocar un <BREAKPOINT> en **5620 return num == 1;**, reinicio todo para que pare cuando haya retornado **nPurchased**.



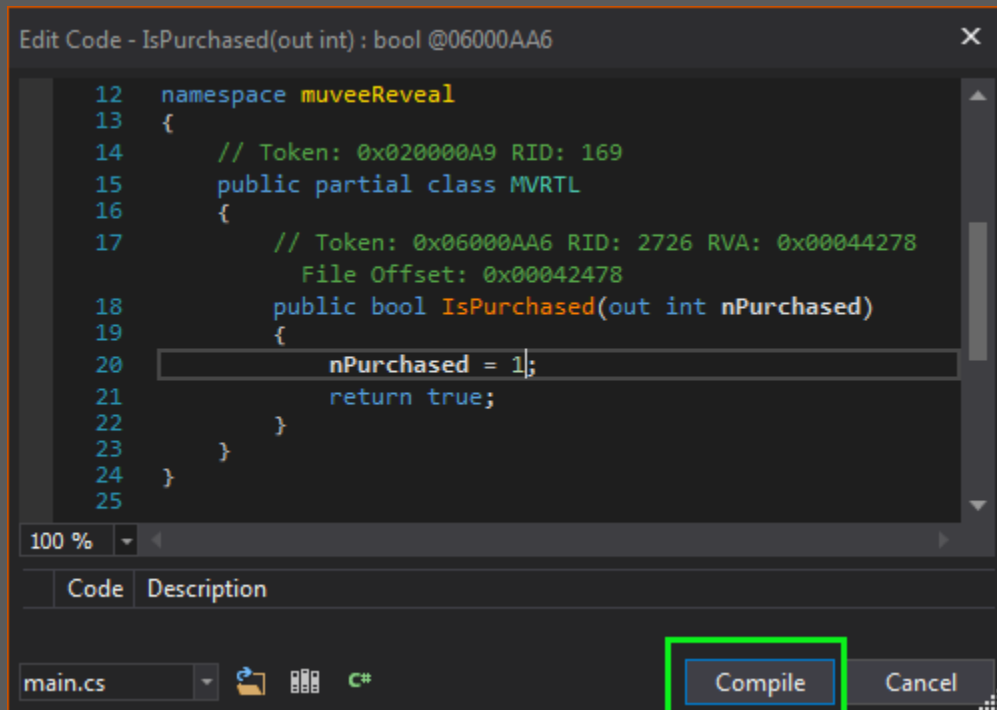
Empecemos por el **RECUADRO VERDE**, ahí podemos seleccionar la vista en la que queremos ver cómo se nos va a mostrar el código, a mí siempre me ha gustado la vista **Visual Basic** y con ella hacer mis cambios, pero resulta que al tratar de hacer los cambios con esa vista me daba error de compilación, decidí cambiar la vista a **C#** cuando me di cuenta que **nextco** la tenía en **C#**, y mira cómo es la vida, los cambios si me eran tomados, creo que es mejor trabajar con la vista **C#** de aquí en adelante. Ahora, si cambiamos manualmente nuestra variable **nPurchased=1** y seguimos la ejecución con <F5> vemos que el programa nos arranca sin mostrarnos la **NAG** y no le importa que estemos desconectados de Internet. Luego, volvemos a detenernos en el <BREAKPOINT>

[muvee Reveal v13.0.0.29340 Build 3157 (.NET)(Crack-Patch)(dnSpy v5.0.7)]

que tenemos en `IsPurchased`. Cambiemos a mano de nuevo nuestra variable `nPurchased=1`. Esto se hace un par de veces y el programa funciona muy bien hasta ahí,. Bien, cambiemos la función `public bool IsPurchased(out int nPurchased)` para que siempre nos retorne `nPurchased=1`.

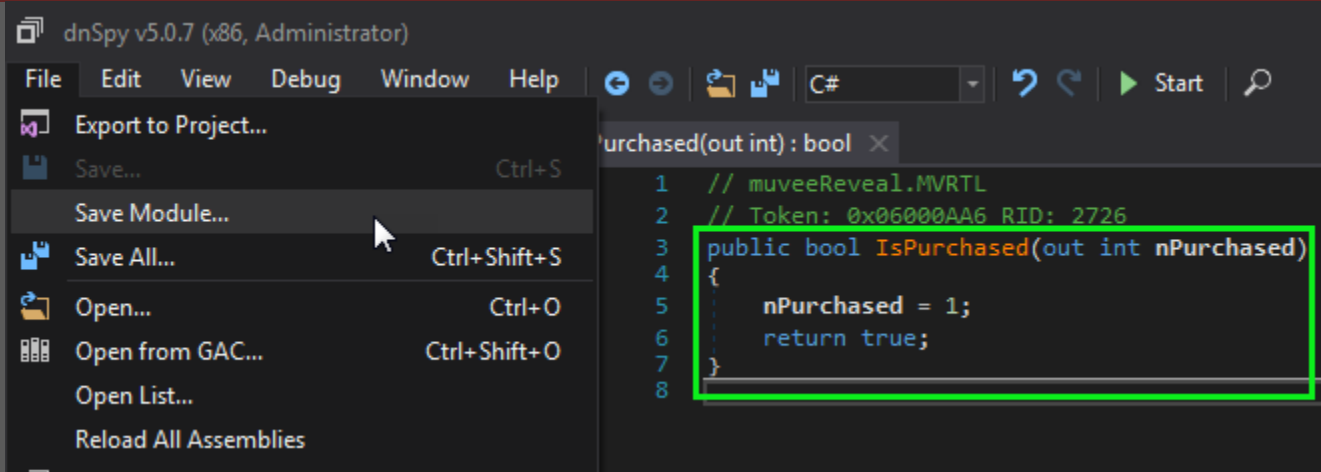


<Clic derecho>Edit Method (C#)...>. Con eso se nos abre la ventana para editar nuestro código.

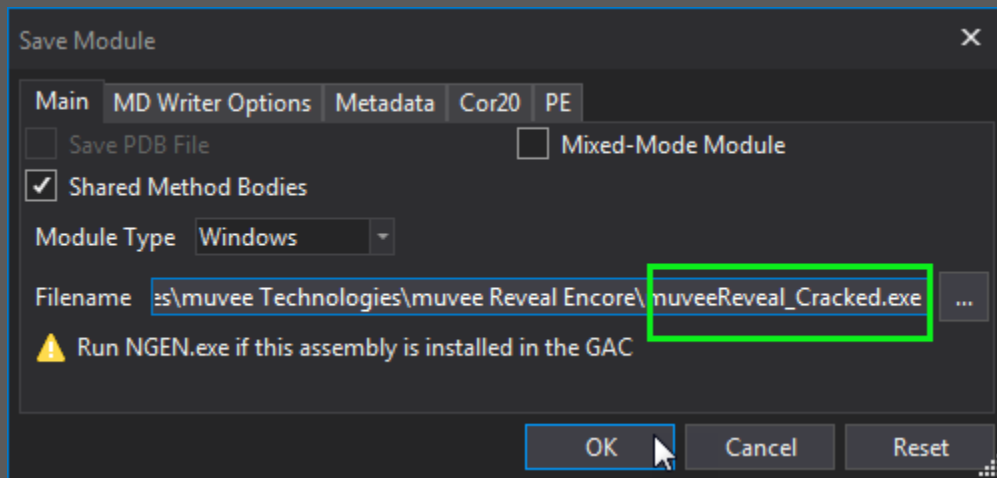


Listo el pollo, aplicamos los cambios en "Compile". Solo nos queda guardar ya en serio esos cambios.

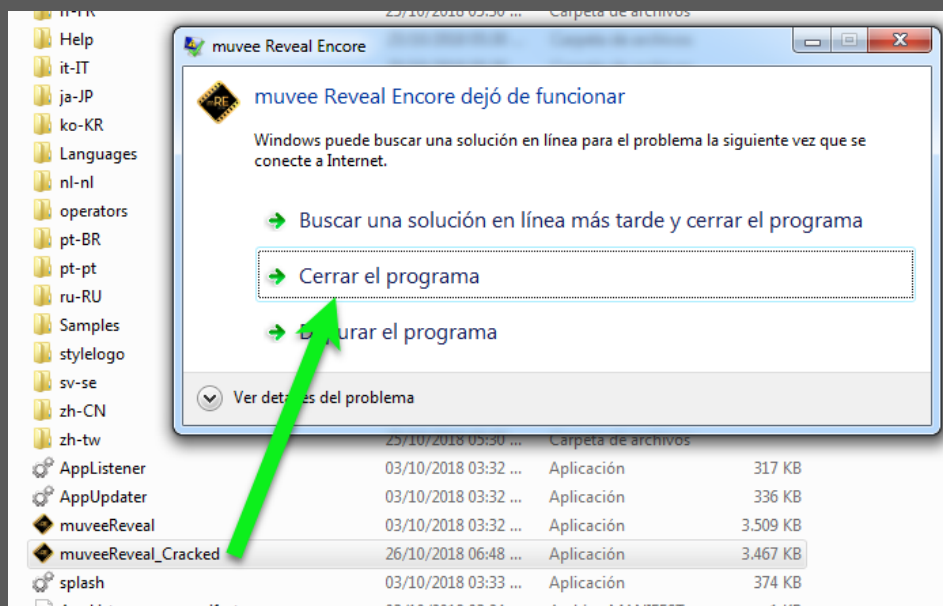
[muvee Reveal v13.0.0.29340 Build 3157 (.NET)(Crack-Patch)(dnSpy v5.0.7)]



Vamos a guardar el módulo, podemos observar que `public bool IsPurchased(out int nPurchased)` ha tomado nuestros cambios.

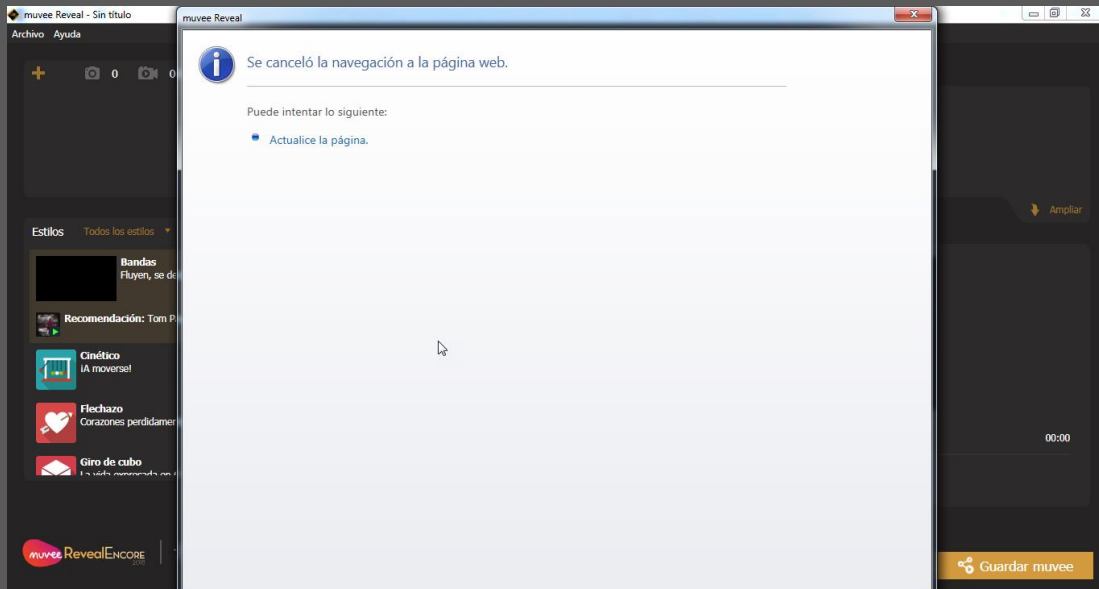


Listo, lo he guardado como **muveeReveal_Cracked.exe**. Ahora probémoslo, a ver qué tal.

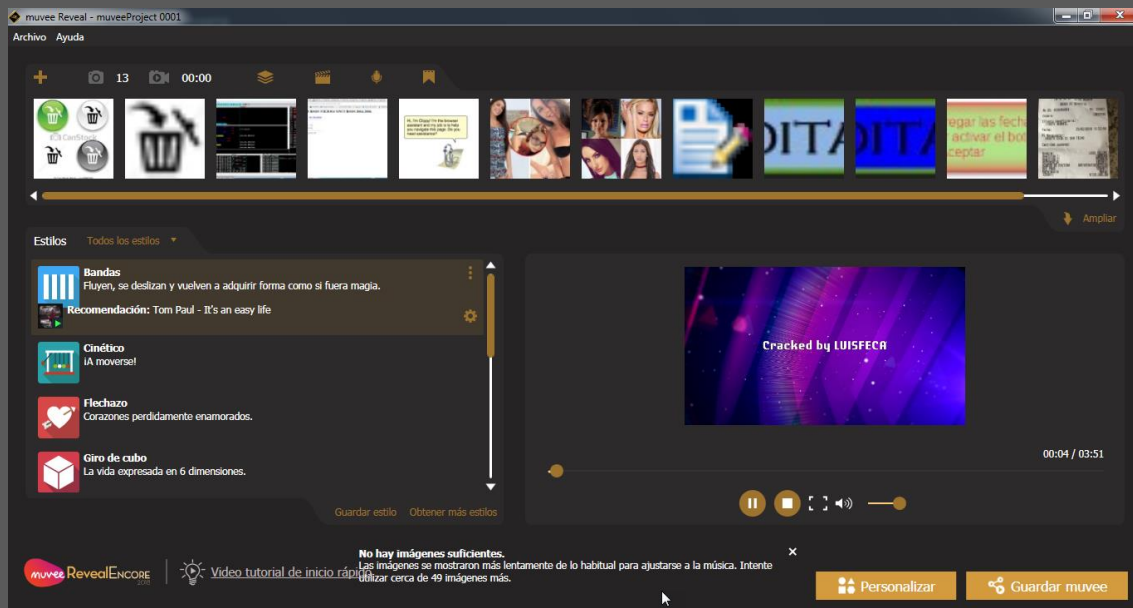


[muvee Reveal v13.0.0.29340 Build 3157 (.NET)(Crack-Patch)(dnSpy v5.0.7)]

Si tratamos de ejecutar nuestro **EXE** crackeado con ese nombre **muveeReveal_Cracked.exe**. El programa no se ejecuta, da un error; sabemos que es muy frecuente que los programas verifiquen si tienen su nombre original para funcionar de lo contrario fallan, así que debemos renombrarlo como el original, **muveeReveal.exe**.

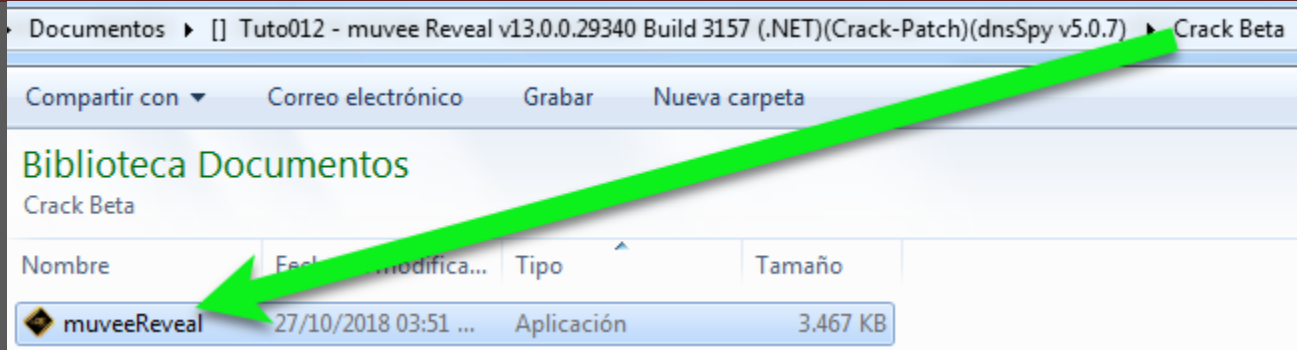


Ja! No me funcionó, esto está muy raro, parecía que íbamos bien. El desconcierto aumenta cuando lo cargo en el **<dnSpy v5.0.7>** y ahí sí me arranca bien.

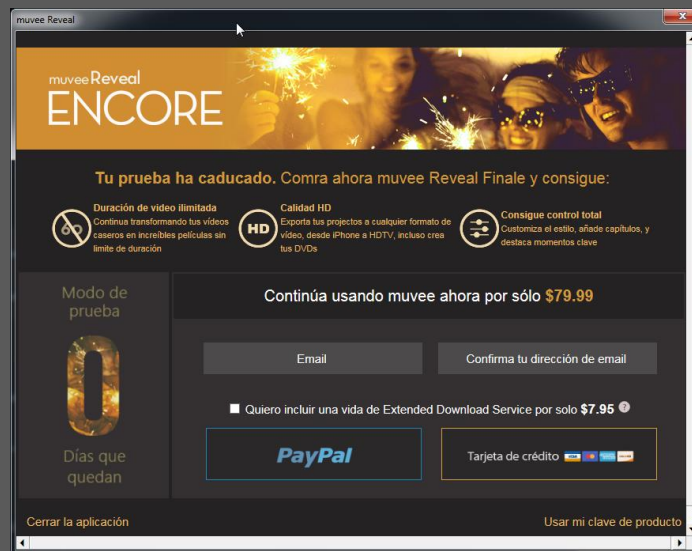


La verdad no tengo las más mínima idea de por qué diablos ocurre eso. Opté por desinstalar completamente el programa, limpie el **Registro de Windows** con el **<CCleaner>**, borré la carpeta, he hice todo de nuevo y ahí sí, mi archivo crackeado me funciona muy bien, podemos decir que ya crackeamos el programa y que hemos hecho el **Crack**.

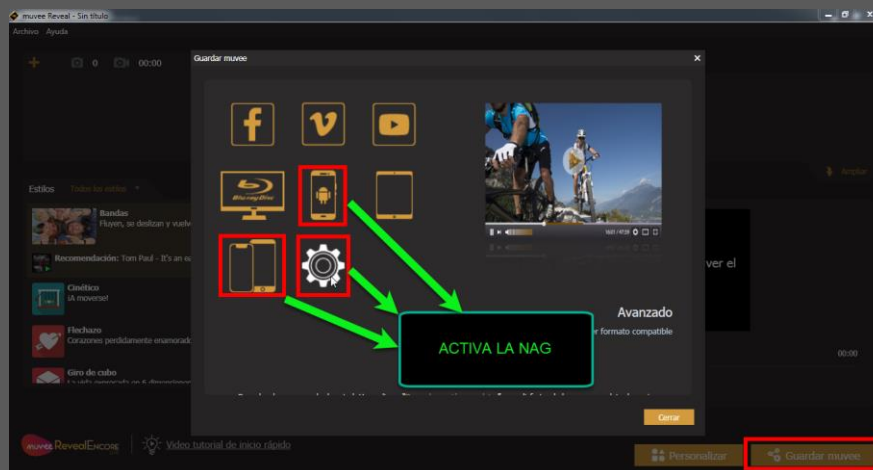
[muvee Reveal v13.0.0.29340 Build 3157 (.NET)(Crack-Patch)(dnSpy v5.0.7)]



Ya probando la aplicación crackeada para ver cómo se porta o si salta algún aviso que nos salga con el cuento de versión **TRIAL** y si sale con eso, pues que no cantemos victoria tempranamente. Pues la realidad es que nuestro primer **Crack** está incompleto, nuestra pinche **NAG** aparece de nuevo.



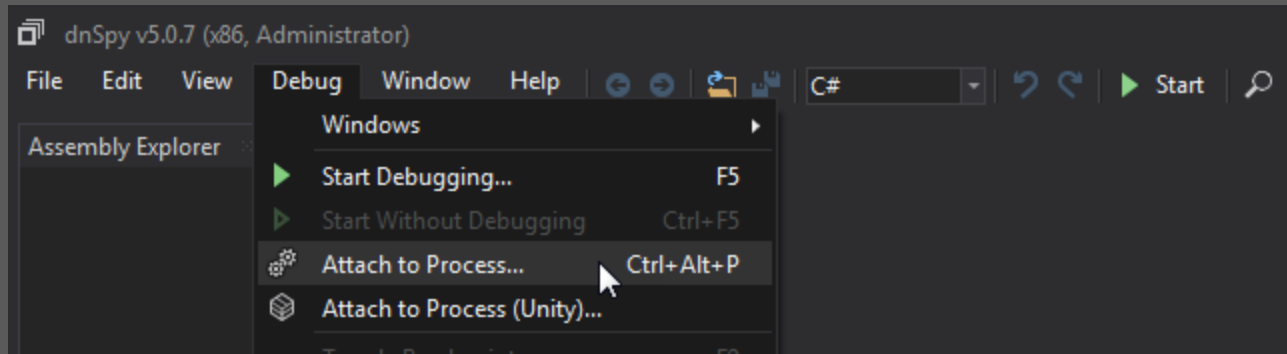
Esa **NAG** aparece en un par de casos especiales que ya abordaremos. Algo muy importante para resaltar es que si cerramos la **NAG**, ya no se nos cierra el programa. Ahora sí, miremos cuáles son los casos especiales que nos muestran la **NAG**.



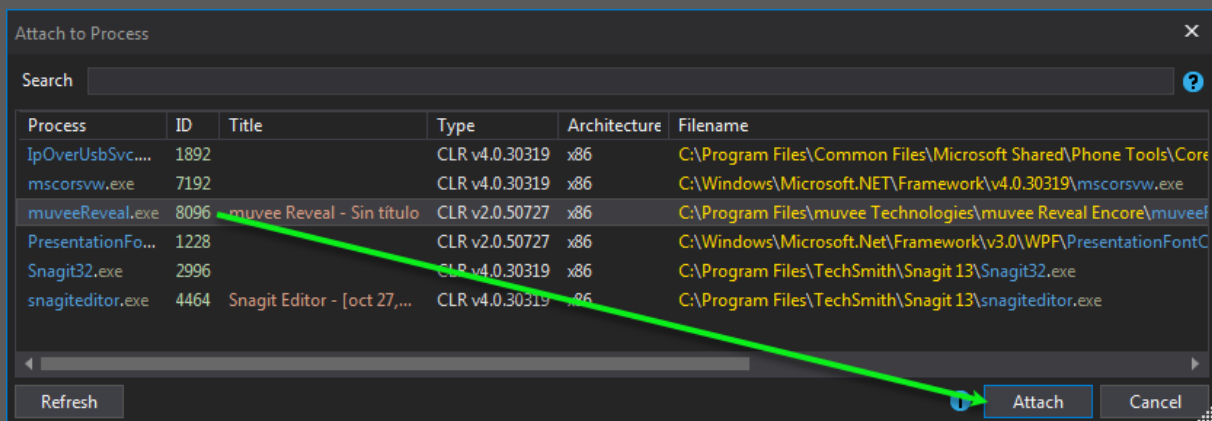
Cuando vamos a guardar un video, se nos abre la ventana para escoger qué tipo de video vamos a guardar, si escogemos una de esas tres opciones resaltadas con un **RECUADRO ROJO** nos falla el **Crack** y nos muestra la **NAG**. Para hallar el lugar de

[muvee Reveal v13.0.0.29340 Build 3157 (.NET)(Crack-Patch)(dnSpy v5.0.7)]

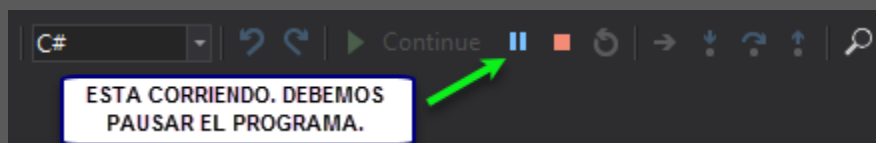
dónde es cargada la **NAG**, hacemos el mismo procedo inicial del **CALL STACK**. El programa lo he ejecutado solo fuera del **<dnSpy v5.0.7>**, así que vamos a attachearlo, **<Debug->Attach to Process...>** o **<CTRL+ALT+P>**



Se nos abrirá una ventana donde se mostrarán los procesos que están corriendo en ese momento.



Escogemos nuestro **muveeReveal.exe** y lo attacheamos con **"Attach"**. Veremos que no se carga nada pero si miramos nuestra barra de herramientas.



Podemos ver que el programa se está ejecutando, debemos es pausarlo y cuando hagan eso, ahí se cargará todo como si lo hubiéramos hecho de la forma tradicional pero mucho mejor porque aquí hemos cargado el programa en ejecución estando con la **NAG** cargada, y no en el **<ENTRY POINT>**, **EP**. Ahora sí, podemos buscar en el **CALL STACK** el método que nos hace saltar la **NAG** y ver si podemos evitar eso.

```
663     private bool IsLimited(MVShare mShare)
664     {
665         string empty = string.Empty;
666         if (mShare.GetFeatureLimitName(out empty))
667         {
668             int num = 0;
669             if (empty == "MobileAddOnLimit")
670             {
671                 App.mvEngine.IsLimited(6, out num);
672                 if (num == 1)
673                 {
674                     return this.ShowUpgradeWinForm(FEATURE_LIMIT.LIMIT_MOBILEADDON);
675                 }
676             }
677             else if (empty == "iPodAddOnLimit")
678             {
679                 App.mvEngine.IsLimited(7, out num);
680                 if (num == 1)
681                 {
682                     return this.ShowUpgradeWinForm(FEATURE_LIMIT.LIMIT_IPODADDON);
683                 }
684             }
685             else if (!(empty == "HdSaveAddOnLimit"))
686             {
687                 if (empty == "DvdBurningAddOnLimit")
688                 {
689                     return this.ShowUpgradeWinForm(FEATURE_LIMIT.LIMIT_DVDADDON);
690                 }
691             }
692         }
693     }
694 }
```

Call Stack

Name
Toolbox.dll!Toolbox.Utils.ShowWinForm(System.Windows.Forms.Form winForm, System.Windows.Forms.IWin32Window parent)
muveeReveal.exe!muveeReveal.MVShareMainIconOnly.ShowUpgradeWinForm(muveeReveal.MVShareMainIconOnly mvAppSDKLib.FEATURE_LIMIT fl) (IL=0x0048, Native=0x00000000)
muveeReveal.exe!muveeReveal.MVShareMainIconOnly.IsLimited(muveeReveal.MVShare mShare) (IL=0x0030, Native=0x00B06550)

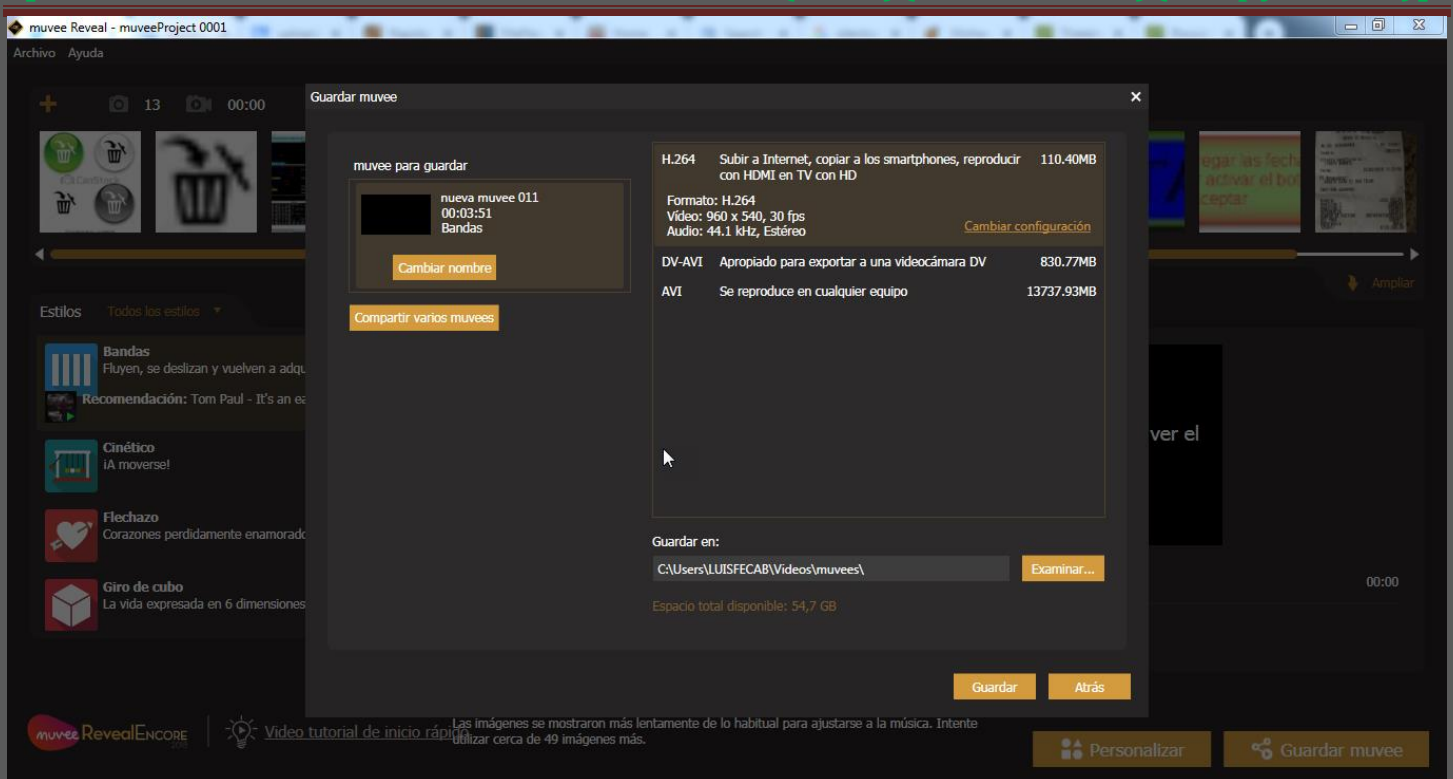
Tools Search Analyzer Call Stack

Ese CALL, `private bool IsLimited(MVShare mShare)`, es el que me carga la **NAG**, resulta que va comparando la variable `empty` con diferentes **Strings**, y si la variable `empty` es igual a una de esas variables se nos carga la **NAG**; ¿y cuándo la variable `empty` es igual?, pues cuando hemos seleccionado una de esas tres opciones de video, lo recuerdan, arriba colocamos la imagen donde teníamos resaltadas esas opciones. Entonces por cada opción que escojamos `empty` retornará con una String a comparar. En nuestro caso son tres, pero uno no sabe más adelante, puede que cualquiera de las otras comparaciones la active, así que lo que hare será cambiar esas **Strings** por otra que nunca me dará la igualdad; la **String** será **"PeruCrackerS"**, qué mejor que dejar la firma del grupo.

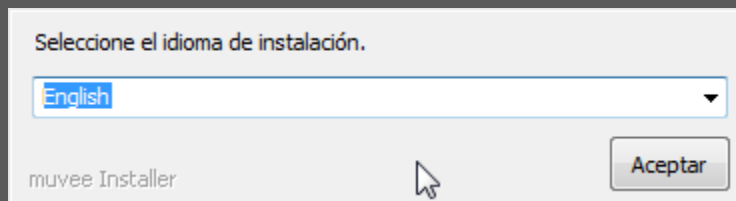
```
private bool IsLimited(MVShare mShare)
{
    string empty = string.Empty;
    if (mShare.GetFeatureLimitName(out empty))
    {
        int num = 0;
        if (empty == "PeruCrackerS")
        {
            App.mvEngine.IsLimited(6, out num);
            if (num == 1)
            {
                return this.ShowUpgradeWinForm(FEATURE_LIMIT.LIMIT_MOBILEADDON);
            }
        }
        else if (empty == "PeruCrackerS")
        {
            App.mvEngine.IsLimited(7, out num);
            if (num == 1)
            {
                return this.ShowUpgradeWinForm(FEATURE_LIMIT.LIMIT_IPODADDON);
            }
        }
        else if (empty == "PeruCrackerS")
        {
            App.mvEngine.IsLimited(8, out num);
            if (num == 1)
            {
                return this.ShowUpgradeWinForm(FEATURE_LIMIT.LIMIT_DVDADDON);
            }
        }
    }
}
```

Ahí, una muestra pequeña de nuestro cambio ya realizado. Solo queda guardar los cambios y este será mi nuevo **Crack**. A probarlo, estoy cruzando los dedos.

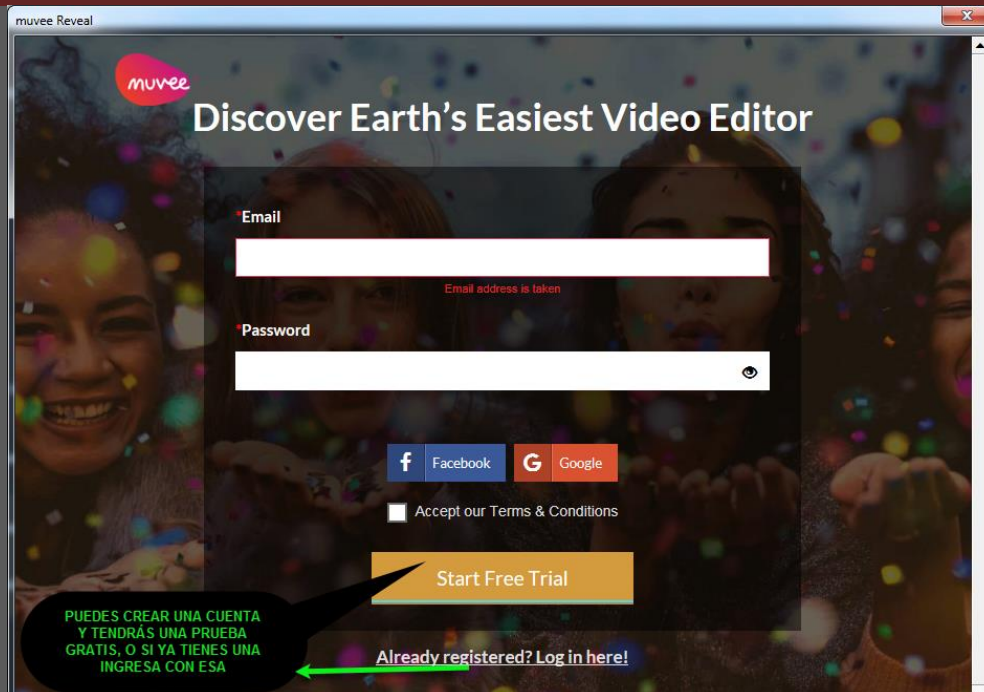
[muvee Reveal v13.0.0.29340 Build 3157 (.NET)(Crack-Patch)(dnSpy v5.0.7)]




Hasta aquí perfecto, ya puedo guardar en esas opciones que me mostraban la **NAG**. He revisado el programa y me funciona muy bien, creo que ya está **Full**. Ahora debemos revisar el descubrimiento de **nextco** que nos carga la **NAG** cuando lo hemos instalado con el idioma inglés. Voy a desinstalar el programa y lo instalaré escogiendo el inglés.



El resto es lo mismo, instalamos normalmente y lo ejecutamos estando conectados a Internet para ver con qué nos sale la **NAG**. Ya verán que tenemos cosas nuevas, la verdad no me equivoqué hace mucho rato cuando escribí que este programa es un pillo. Si tú instalas en inglés te muestra una información diferente, te trata de forma diferente, qué elitista este programa.



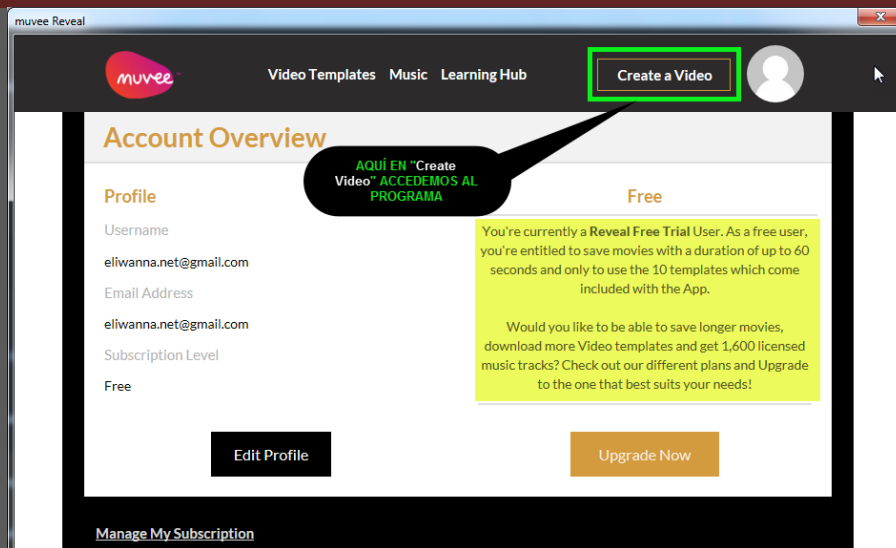
Aquí tenemos algo completamente diferente, no nos aparece información de que nuestra prueba gratis terminó, aquí parece que podemos volver a tener de nuevo una prueba del programa, pero que debemos crear una cuenta e ingresar con ella para poder disfrutar de esos **15** días. He creado mi cuenta y voy a ingresar.

We've got a plan for you! → 

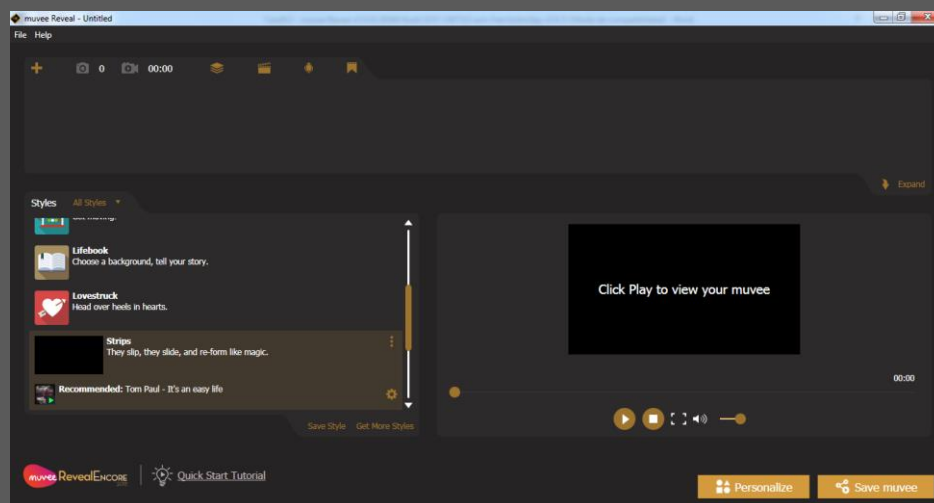
	Studio \$49.99 /monthly	Plus \$19.99 /monthly	Solo \$6.99 /monthly
No contract. No fine print. Cancel Anytime.	START CREATING	START CREATING	START CREATING
No watermark. No limits.	✓	✓	✓
Lifetime Upgrades. Always.	✓	✓	✓
Members-only Training Content	✓	✓	✓
Same-day FastTrack Support	✓	✓	✓
Video Style Templates	ALL (62)	10	6
Cinematic Title Themes	ALL (23)	13	5
Monthly New Themes & Templates	Unlimited	Any 25	
Music Library (1,600 tracks)	✓		

FAQs

Cuando ingreso me muestra una opción para que contrate un plan, no muchas gracias yo estoy detrás de los supuestos **15** días gratis, aquí no pide **Serial** de activación, si quieres la versión **Full** debes contratar un plan ligado a tu cuenta. Voy a cerrar eso para ver con qué me sale ahora.



Nos aparece el resumen de nuestra cuenta, y como vemos somos un suscripción **Free**, y resaltado en **AMARILLO** todas las restricciones, con esas restricciones da lo mismo que no ofrecieran nada. Mirando por ahí no noto nada que nos cuente los supuestos **15** días gratis de prueba, puede ser que aquí la puedas usar **Free** todo el tiempo, claro que con esas restricciones no sirve para nada. Ahora, me voy por la opción **"Create Video"**. Ayyy! Jue madre se me cierra el programa, como que si toma mis **15** días y como yo ya los utilicé pues a la porra me manda, bueno estoy suponiendo eso. Todo esto ha sido probando el programa recién instalado y en inglés, ahora lo vamos a crackear con nuestro último **Crack**, pero primero creamos una copia del **EXE** original y luego si reemplazarlo con nuestro **Crack**. Lo ejecutamos para ver si lo dicho por **nextco** es cierto. Efectivamente **nextco** no se equivocó, me apareció la misma **NAG** donde me pide que ingrese con un **e-mail** y un **Password** para disfrutar de una **FREE TRIAL**. Ingreseemos de nuevo para ver si se nos cierra la aplicación.



Una maravilla, el programa no se cierra y funciona **Full**. Nuestra **Crack** funciona muy bien, podríamos dejar el **Crack** así y si lo compartimos debemos dejar explicado que debes crear una cuenta para poder usar el programa **Full** después de reemplazar con nuestro **Crack** si hemos realizado la instalación en inglés.

Mejor seguimos puliendo el **Crack** y hagamos que no salga esa **NAG**, gracias a **nextco** ya sabemos dónde está o podemos usar de nuevo nuestro **CALL STACK**.

[muvee Reveal v13.0.0.29340 Build 3157 (.NET)(Crack-Patch)(dnSpy v5.0.7)]

```
14375 }
14376 if (CultureInfo.CurrentUICulture.Name == "en-US")
14377 {
14378     if (this._dpiScaleX <= 0.0 || this._dpiScaleY <= 0.0)
14379     {
14380         this.GetDPIScalingFactor(out this._dpiScaleX, out this._dpiScaleY);
14381     }
14382     UpgradeWinForm upgradeWinForm = new UpgradeWinForm(TYPE_OF_HTML.EXPIRED_A);
14383     upgradeWinForm.SetOwnerForEnterKeyDialog(this);
14384     upgradeWinForm.ScaleDialog(this._dpiScaleX, this._dpiScaleY);
14385     upgradeWinForm.SetExitConfirm(true);
14386     Utils.ShowWinForm(upgradeWinForm, this);
14387     if (!upgradeWinForm.m_bCloseByButton)
14388     {
14389         ...
14390     }
14391 }
```

Call Stack

Name
> muveeReveal.exe!muveeReveal.MVMain.UpdateTimer(object sender, System.EventArgs e) (IL=0x03C5, Native=0x0EA67608+0x9FD)
WindowsBase.dll!System.Windows.Threading.DispatcherTimer.FireTick(object unused) (IL=0x0020, Native=0x0F6B9398+0x48)

Locals Search Analyzer Call Stack

Vemos en la línea 14376 `if (CultureInfo.CurrentUICulture.Name == "en-US")`, donde se compara si estamos en inglés y si es así, entramos a ese procedimiento para mostrarnos la **NAG** en 14386 `Utils.ShowWinForm(upgradeWinForm, this);`. La solución ya fue dada por **nextco**, cambiar la string "en-US" por una que nunca de igualdad, y ya deben suponer cuál voy a poner, pues efectivamente "**PeruCrackers**".

```
14376 if (CultureInfo.CurrentUICulture.Name == "PeruCrackers")
14377 {
14378     if (this._dpiScaleX <= 0.0 || this._dpiScaleY <= 0.0)
14379     {
14380         ...
14381     }
14382 }
```

Guardamos los cambios y este viene siendo nuestro **Crack** final. Lo fui a probar y no me sirvió, me salía la **NAG** en inglés pero ya sabía que no era fallo de mi **Crack**, solo con volver a instalar todo y de colocar el **Crack**, con eso caso resuelto. Listo, podemos decir que llegamos al final de la primera fase, hacer un **Crack**; lo siguiente será crear un **Patch**. Como utilicé el <dnSpy v5.0.7> no puedo utilizar la opción de comparar del <Hex Workshop Hex Editor v6.7 (6.8.0.5419)>, ya me lo decía DavicoRm que usara el <DotNet Resolver v3.3.0.1> que solo guarda nuestros cambios y no recompilado como lo hace el <dnSpy v5.0.7>. Bueno, no importa lo haré comparando los dos archivos abriéndolos cada uno con el <dnSpy v5.0.7>.

<pre>public bool IsPurchased(out int nPurchased) { if (this.m_bForceExpire) { nPurchased = 0; return true; } int num = 0; this.m_mvAppCoord.IsPurchased(out nPurchased, out num); return num == 1; }</pre>	ORIGINAL	<pre>public bool IsPurchased(out int nPurchased) { nPurchased = 1; return true; }</pre>	CRACKED
--	-----------------	---	----------------

Luego cambiaré la vista que tengo en **C#** por la **IL** que ahí me muestra los **BYTES**, y así hallaremos los **BYTES** para parchear y como utilizaremos el <dup2 Diablo's Universal Patcher v2.26> con el módulo [Search and Replace Patch], entonces que busque nuestros **BYTES** del procedimiento original y los reemplace por nuestros **BYTES** crackeados.

```
instance bool IsPurchased (
    [out] int32& nPurchased
) cil managed
{
    // Header Size: 12 bytes
    // Code Size: 37 (0x25) bytes
    // LocalVarSig Token: 0x1100001F RID: 31
    .maxstack 3
    .locals init (
        [0] int32
    )

    /* 0x00042484 02 */ /* IL_0000: ldarg.0
    /* 0x00042485 7B89060004 */ /* IL_0001: ldfld bool
    muveeReveal MVRTL::m_bforceExpire
    /* 0x0004248A 2C05 */ /* IL_0006: brfalse.s IL_000D
    /* 0x0004248C 03 */ /* IL_0008: ldarg.1
    /* 0x0004248D 16 */ /* IL_0009: ldc.i4.0
    /* 0x0004248E 54 */ /* IL_000A: stind.i4
    /* 0x0004248F 17 */ /* IL_000B: ldc.i4.1
    /* 0x00042490 2A */ /* IL_000C: ret
```

ORIGINAL

PROCEDIMIENTO CONSTA DE 37 BYTES

Ahí tenemos la función original que tiene una longitud de **37 BYTES**, **Code Size: 37 (0x25) bytes**. No tengo muy claro esto, **Header Size: 12 bytes**, pero debe ser los **BYTES** que definen el procedimiento como por ejemplo sus parámetros y lo que retorna. Recordemos esos dos datos el **Header Size** y **Code Size**. El procedimiento es más largo pero en el **RECUADRO ROJO** hacia abajo tenemos todos nuestros **BYTES** originales que vendrían siendo:

BYTES ORIGINALES - **IsPurchased**:

027B890600042C05031654172A160A027B7C0600040312006F7005000A06172E02162A172A

Esos **BYTES** son los que debemos parchear por los nuevos **BYTES** del **IsPurchased** modificado. Miremos en la imagen nuestros nuevos **BYTES**.

```
// Token: 0x06000AF9 RID: 2809 RVA: 0x000077DF File Offset: 0x000059DF
.method public hidebysig
    instance bool IsPurchased (
        [out] int32& nPurchased
    ) cil managed
{
    // Header Size: 1 byte
    // Code Size: 5 (0x5) bytes
    .maxstack 8

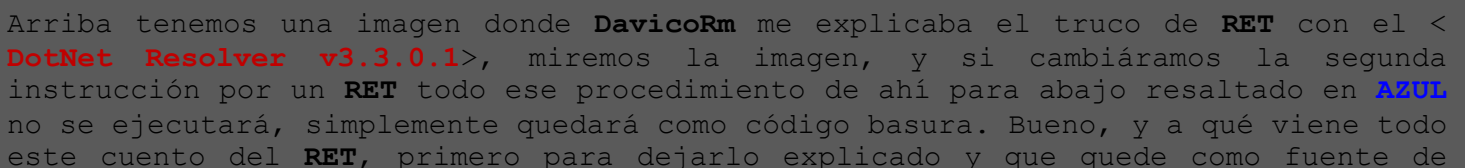
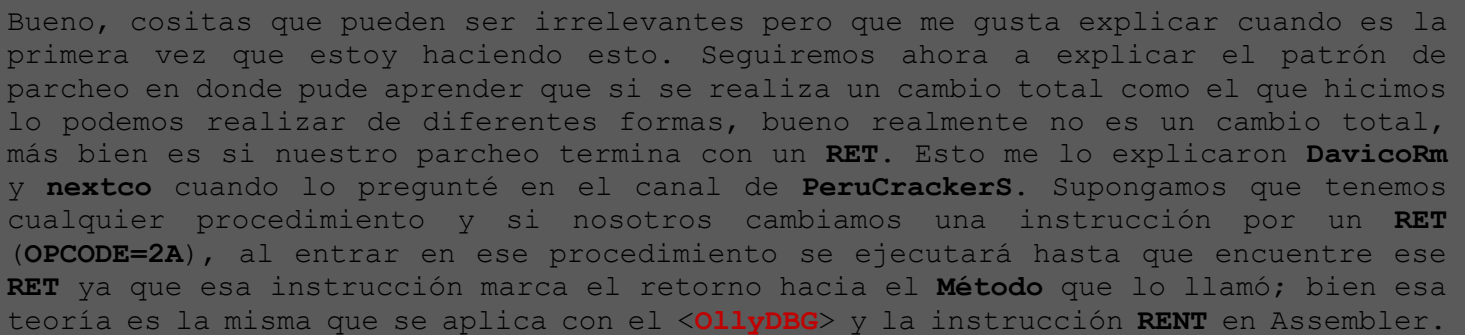
    /* 0x000059E0 03 */ /* IL_0000: ldarg.1
    /* 0x000059E1 17 */ /* IL_0001: ldc.i4.1
    /* 0x000059E2 54 */ /* IL_0002: stind.i4
    /* 0x000059E3 17 */ /* IL_0003: ldc.i4.1
    /* 0x000059E4 2A */ /* IL_0004: ret
} // end of method MVRTL::IsPurchased
```

Ahí tenemos nuestro **IsPurchased** modificado, y este es mucho más corto solo son **5 BYTES**, **Code Size: 5 (0x5) bytes**. Curioso que el **Header**, **Header Size: 1 byte**, yo pensaba que este no cambiaría, debe ser que mi suposición del **Header** que planteé arriba sea errónea. Bueno, aquí mis nuevos **BYTES** son:

BYTES NUEVOS - **IsPurchased** MODIFICADO:

031754172A

Se me olvidaba comentarles que podemos copiar los **BYTES** desde el **<dnSpy v5.0.7>**, seleccionando todo el código y **<Clic Derecho->Copy IL Bytes>**.



[muvee Reveal v13.0.0.29340 Build 3157 (.NET)(Crack-Patch)(dnSpy v5.0.7)]

aprendizaje para quien lea el tutorial y que no sepa bien esto del **RET**, y lo segundo porque podemos aplicar la teoría del **RET** para nuestro parcheo, solo debemos agregar al <**dUP2 Diablo's Universal Patcher v2.26**> un patrón con los **BYTES** originales para que pueda hallar la concordancia y parchear desde el inicio con nuestros nuevos **5 BYTES**, y como el parcheo termina con un **RET** (**OPCODE=2A**) no es necesario parchear el total de los **37 BYTES** originales ya que después de parchear quedan sobrando gracias a nuestro **RET**.

```
instance bool IsPurchased (
    [out] int32& nPurchased
) cil managed
// Token: 0x06000AF9 RID: 2809 RVA: 0x000077DF File
.method public hidebysig
    instance bool IsPurchased (
        [out] int32& nPurchased
    ) cil managed
// Header Size: 12 bytes
// Code Size: 37 (0x25) bytes
// LocalVarSig Token: 0x1100001F RID: 31
.maxstack 3
.locals init (
    [0] int32
)
/* 0x00042484 02          */ IL_0000: ldarg.0
/* 0x00042485 7B89060004    */ IL_0001: ldfld
muveeReveal.MVRTL::m_bforceExpire
/* 0x0004248A 2C05          */ IL_0006: brfalse.s
/* 0x0004248C 03          */ IL_0008: ldarg.1
/* 0x0004248D 16          */ IL_0009: ldc.i4.0
/* 0x0004248E 54          */ IL_000A: stind.i4
/* 0x0004248F 17          */ IL_000B: ldc.i4.1
/* 0x00042490 2A          */ IL_000C: ret
/* 0x000059E0 03          */ IL_0000: ldarg.1
/* 0x000059E1 17          */ IL_0001: ldc.i4.1
/* 0x000059E2 54          */ IL_0002: stind.i4
/* 0x000059E3 17          */ IL_0003: ldc.i4.1
/* 0x000059E4 2A          */ IL_0004: ret
} // end of method MVRTL::IsPurchased
```

Esa es la explicación del patrón de parcheo que **DavicoRm** hizo, quedando así:

BYTES ORIGINALES - IsPurchased:

027B890600042C05031654172A160A027B7C0600040312006F70 -> CON ESOS BYTES HAY CONCORDANCIA.

BYTES NUEVOS - IsPurchased MODIFICADO:

031754172A?? -> RET (OPCODE=2A).

Bien y esos signos de pregunta a qué vienen, los signos significan que esos **BYTES** no son tomados en cuenta que ahí no se parcha nada y por supuesto se debe completar la longitud del patrón con **??=1 BYTE**. El patrón de parcheo de arriba muestra que **DavicoRm** es un cracker avezado en los **Patch**, seguro si uno se descuida **DavicoRm** hasta lo parchea a uno. Como dijimos podemos tener patrones diferentes a parchear, por ejemplo nopear una parte del inicio, luego agregar nuestros nuevos **BYTES** Y nopear o dejarlos iguales.

BYTES ORIGINALES - IsPurchased:

027B890600042C05031654172A160A027B7C0600040312006F70 -> CON ESOS BYTES HAY CONODANCIA.

BYTES NUEVOS – IsPurchased MODIFICADO:

```
000000031754172A????????????????????????????????????? -> RET (OPCODE=2A).
```

```
000000031754172A000000000000000000000000000000 -> RET (OPCODE=2A) .
```

Podemos hacer esto mismo con el total de los **37 BYTES**. Claro que entre menos **BYTES** se supone que más liviano y eficiente.

BYTES ORIGINALES - IsPurchased:

027B890600042C05031654172A160A027B7C0600040312006F7005000A06172E02162A172A

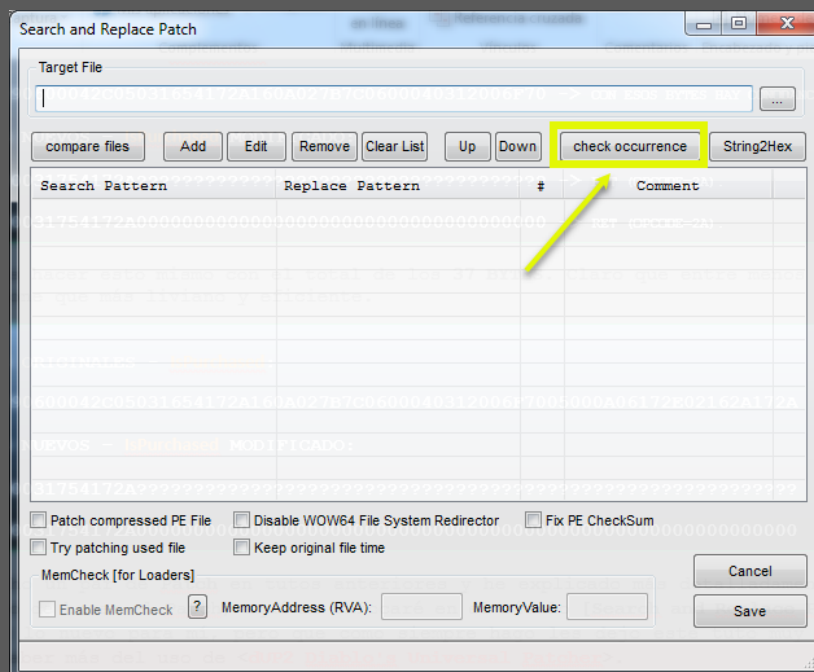
BYTES NUEVOS – IsPurchased MODIFICADO:

```
000000031754172A????????????????????????????????????????????????????????????
```

```
000000031754172A000000000000000000000000000000000000000000000000000000000000
```

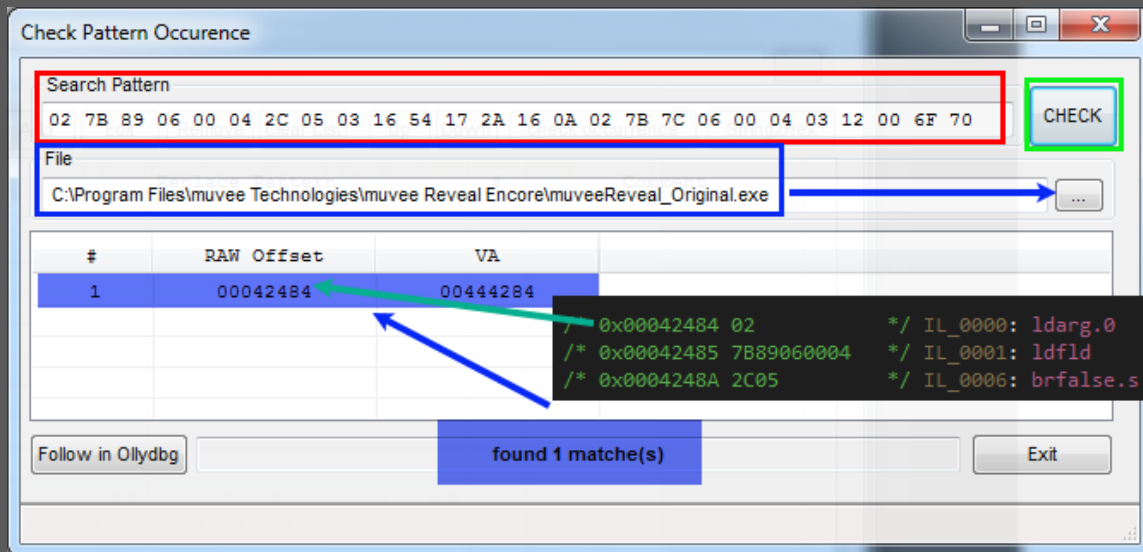
He hecho un par de **Patch** en tutoriales anteriores y he explicado más detalladamente el proceso de hacer el **Patch**, aquí me enfocaré en el módulo **[Search and Replace Patch]** que es lo nuevo para mí, pero que como siempre hago les dejo este tutorial muy bueno para saber más del uso de **<DUP2 Diablo's Universal Patcher>**.

Abramos nuestro módulo para empezar a agregar nuestros patrones de reemplazo.

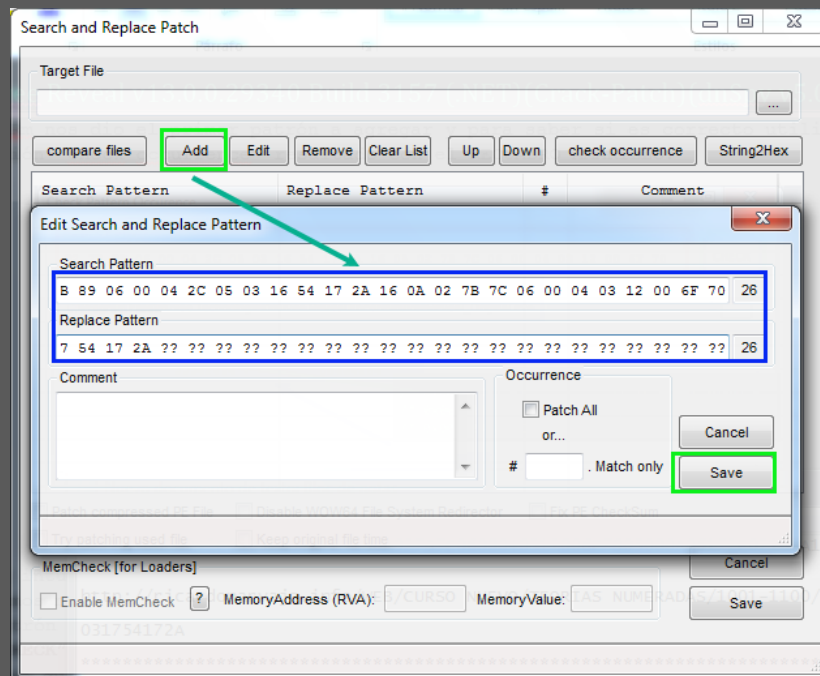


[muvee Reveal v13.0.0.29340 Build 3157 (.NET)(Crack-Patch)(dnSpy v5.0.7)]

DavicoRm nos dio el primer patrón a agregar y para saber si es correcto utilizemos la opción "check occurrence" resaltado con el **RECUADRO AMARILLO**.



Es muy intuitivo, se entiende fácilmente lo que debemos hacer, en el **RECUADRO ROJO** agregamos nuestro patrón a buscar y en el **RECUADRO AZUL** cargamos el archivo a buscar ese patrón que en nuestro caso es el **EXE** original; luego buscamos las ocurrencias con "CHECK". Podemos ver que tenemos una concordancia y la dirección de inicio es la misma que de la función **IsPurchased** en el <dnSpy v5.0.7>. Para nuestro parcheo solo debe ser una sola vez. Ahora sí, agreguémoslo con la opción "Add".

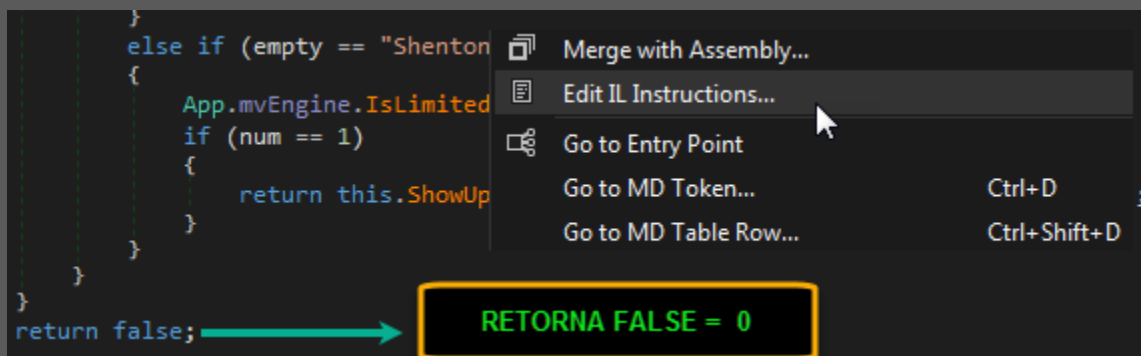


Bien, agregamos el patrón de búsqueda y el de reemplazo, los guardamos con "Save". Ese sería nuestro primer cambio y recordemos que nosotros realizamos dos cambios más, así que debemos hacerlo lo mismo con esos también y agregarlos al **Patch**. El segundo cambio fue en **private bool IsLimited(MVShare mShare)**.

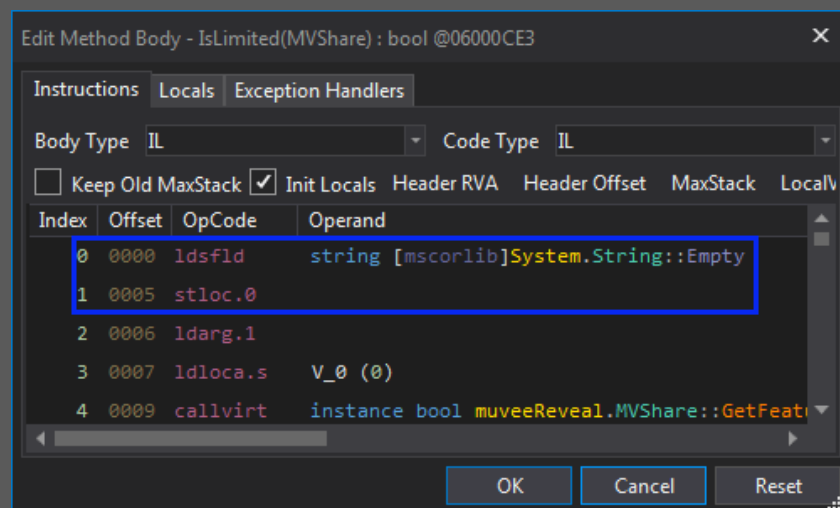
```
private bool IsLimited(MVShare mShare)
{
    string empty = string.Empty;
    if (mShare.GetFeatureLimitName(out empty))
    {
        int num = 0;
        if (empty == "PeruCrackerS")
        {
            App.mvEngine.IsLimited(6, out num);
            if (num == 1)
            {
                return this.ShowUpgradeWinForm(FEATURE_LIMIT.LIMIT_MOBILEADDON);
            }
        }
        else if (empty == "PeruCrackerS")
        {

```

Ahora que miro esa función, la idea es que no haga ninguna comparación, así que aquí podemos aplicarle el truco del **RET**. Entonces agreguemos un **RET**. Algo que debo aclarar y que es muy importante decirlo como complemento a la teoría, es que aquí los **RET** deben devolver un valor **false=0** o **true=1** porque recordemos que **IsPurchased** e **IsLimited** son funciones que devuelven un valor booleano, entonces antes de poner el **RET** (OPCODE=2A) debemos ponerle el valor de retorno.



Aquí tenemos el final de la función y vemos el **RET** (`return false;`) que retorna **false=0**, debemos colocar ese **RET** pero al inicio. Lo haremos editando las instrucciones **IL**, <Click Derecho->Edit IL Instructions...>.



Debemos cambiar las dos primeras instrucciones. La primera instrucción sería para cargar el valor de retorno que es **0**. Usemos nuestra tabla <**MSIL OpCode Table v1.0**> para que vean la instrucción a utilizar con su explicación.

[muvee Reveal v13.0.0.29340 Build 3157 (.NET)(Crack-Patch)(dnSpy v5.0.7)]

Name	Value	Size	Info
ldc.i4	20	1	Pushes specified 32-bit value
ldc.i4.0	16	1	Pushes the literal value, 0
ldc.i4.1	17	1	Copies size bytes from srcAddr to destAddr in me...

La primera instrucción es **ldc.i4.0** (OPCODE=16), que coloca **0**, el cual lo toma el **RET** como **false**. Paso seguido es agregar el **RET**.

Name	Value	Size	Info
rem.un	5E	1	Computes remainder of dividing value1 by valu...
ret	2A	1	Returns control from current method to caller
rethrow	FE1A	2	Only valid within a catch block; this instruction...

Ahí está el **RET** que no necesita más explicación, ahora solo nos queda agregar esos cambios al código **IL**, pero primero voy a dejar los **BYTES** originales porque de ahí sacamos nuestro patrón de búsqueda.

BYTES ORIGINALES - **IsLimited**:

```
7E1F00000A0A0312006FFE0500063939010000160B06728275007028AC00000A2C1D7EA80400041C12016FA40A0006
2607174015010000021C28E40C00062A0672A475007028AC00000A2C1D7EA80400041D12016FA40A000626071740EB
000000021D28E40C00062A06724674007028AC00000A3AD30000000672C275007028AC00000A2C1F7EA80400041F09
12016FA40A000626071740B0000000021F0928E40C00062A0672EC75007028AC00000A2C1F7EA80400041F0A12016F
A40A00062607174084000000021F0A28E40C00062A06721E76007028AC00000A2C1C7EA80400041F0B12016FA40A0
06260717335B021F0B28E40C00062A06725876007028AC00000A2C1C7EA80400041F0C12016FA40A00062607173332
021F0C28E40C00062A06727476007028AC00000A2C1C7EA80400041F0D12016FA40A00062607173309021F0D28E40C
00062A162A
```

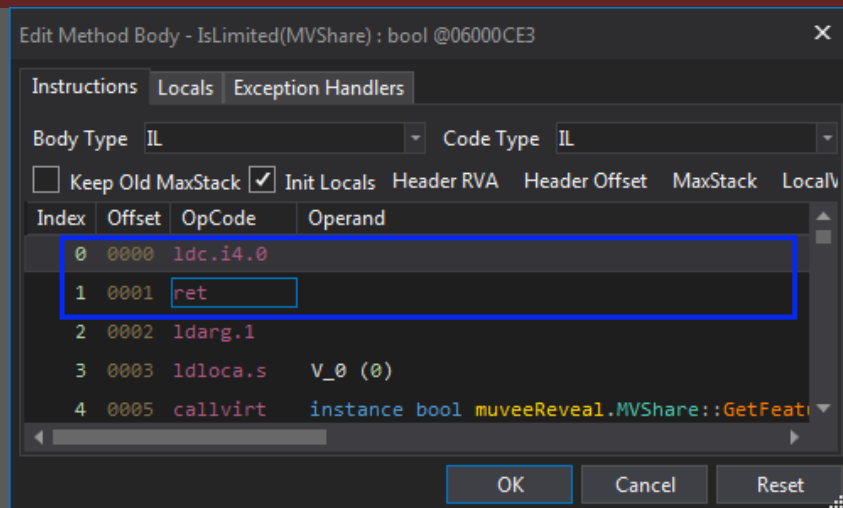
Son un montón de **BYTES** (Code Size: 334 (0x14E) bytes), pero solo debemos realizar la búsqueda con un pedazo que nos dé una concordancia y reemplazarlo con nuestros cambios al inicio para que el resto de **BYTES** no sean tomados en cuenta. Ya sabemos nuestros cambios.

BYTES ORIGINALES - **IsLimited** MODIFICADO:

162A

Hagamos los cambios en el <**dnSpy v5.0.7**> para practicar porque ya averiguamos los **BYTES** a cambiar.

[muvee Reveal v13.0.0.29340 Build 3157 (.NET)(Crack-Patch)(dnSpy v5.0.7)]



Podemos seguir analizando más cosas y ver cómo se comporta los cambios cuando los hacemos desde la vista **IL**.

```
instance bool IsLimited (
    class muveeReveal.MVShare mShare
) cil managed

// Header Size: 12 bytes
// Code Size: 330 (0x14A) bytes
// LocalVarSig Token: 0x11000033 RID: 51
.maxstack 3
.locals init (
    [0] string,
    [1] int32
)

/* 0x00054874 16          */ IL_0000: ldc.i4.0
/* 0x00054875 2A          */ IL_0001: ret

/* 0x00054876 03          */ IL_0002: ldarg.1
/* 0x00054877 1200        */ IL_0003: ldloca.s V_0
/* 0x00054879 6F47060006  */ IL_0005: callvirt instance
/* 0x0005487E 3939010000  */ IL_000A: brfalse IL_0148
```

En el **RECUADRO AZUL** están nuestros cambios y en el **RECUADRO ROJO** el código original que no será usado para nada. Solo nos queda hacer el patrón para el **Patch**.

BYTES ORIGINALES - IsLimited:

```
7E1F00000A0A0312006FFE0500063939010000160B06728275007028AC00000A2C1D7EA80400041C12016FA40A00062607174015010
000021C2881
```

BYTES ORIGINALES - IsLimited MODIFICADO:

```
162A????????????????????????????????????????????????????????????????????????????????????????????????????????
????????????
```

Me quedó largó el patrón, supongo que se podrá resumir más pero no soy muy "ducho" en Parcheo y menos de esta forma. Bueno, agreguemos este reemplazo también. Solo nos queda agregar el último reemplazo que fue lo descubierto por **nextco** que con el idioma inglés nos carga la **NAG**. Aquí, el patrón es muy similar, solo que colocaremos un retorno sencillo un simple **RET** porque es un procedimiento sencillo que no retorna nada, no requiere cargar un valor previo como los dos procedimientos anteriores que

[muvee Reveal v13.0.0.29340 Build 3157 (.NET)(Crack-Patch)(dnSpy v5.0.7)]


si recordamos eran funciones que retornan un valor Boleano, recordemos de nuevo que las funciones obligatoriamente retornan con un valor.

```
' Token: 0x06000318 RID: 792 RVA: 0x0001BDF0 File Offset: 0x00019FF0
Private Sub UpdateTimer(sender As Object, e As EventArgs)
    Select Case Me.m_timerState
        Case MVMain.TIMERSTATE.NEUTRAL, MVMain.TIMERSTATE.SPLITTING,
```

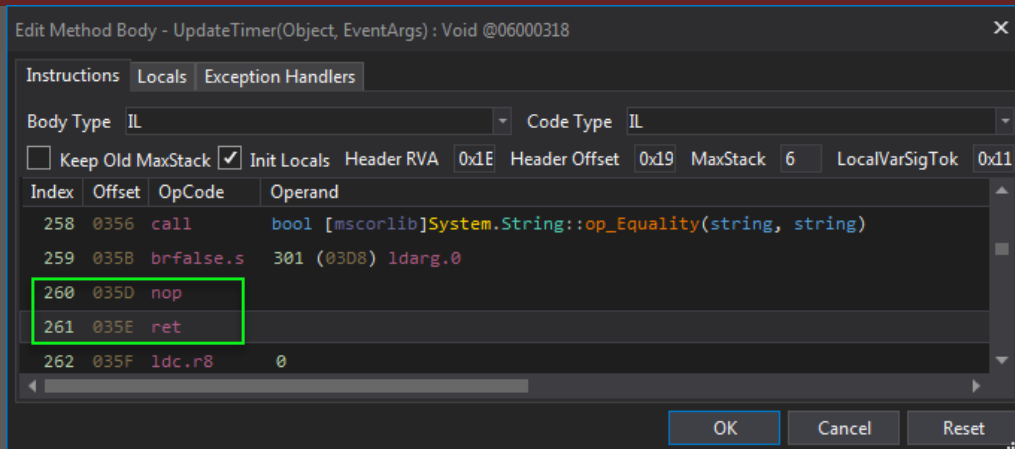
Como vemos es un procedimiento general no una función. Bueno, el **RET** lo haremos en la parte donde se nos va a mostrar a mostrar la **NAG** y lo colocaremos al inicio para que retorne y no se ejecute esa parte del código.

```
88      If CultureInfo.CurrentUICulture.Name = "en-US" Then
89          If Me._dpiScaleX <= 0.0 OrElse Me._dpiScaleY <= 0.0 Then
90              Me.GetDPIScalingFactor(Me._dpiScaleX, Me._dpiScaleY)
91          End If
92          Dim upgradeWinForm As UpgradeWinForm = New UpgradeWinForm(TYPE_OF_HTML.EXPIRED_A)
93          upgradeWinForm.SetOwnerForEnterKeyDialog(Me)
94          upgradeWinForm.ScaleDialog(Me._dpiScaleX, Me._dpiScaleY)
95          upgradeWinForm.SetExitConfirm(True)
96          Utils.ShowWinForm(upgradeWinForm, Me)
97          If Not upgradeWinForm.m_bCloseByButton Then
98              Me._bForceClose = True
99              MyBase.Close()
100             Return
101         End If
102     End If
```

Sabemos que no debemos entrar ahí, y también sabemos que si instalamos con cualquier otro idioma diferente del inglés, no habrá igualdad y nunca entraremos, logrando con eso evitar la **NAG** pero con inglés ahí perdemos el año. Mi solución fue colocar el **RET** al inicio después de entrar **88 If CultureInfo.CurrentUICulture.Name = "en-US" Then**, supongo que podríamos colocar nuestro **RET** encima del **88 If CultureInfo.CurrentUICulture.Name = "en-US" Then**, evitando esa rutina del **IF**. Cabe resaltar que podemos hacer este cambio porque ya tenemos nuestro programa cargado completamente.

<pre>88 If CultureInfo.CurrentUICulture.Name = "en-US" Then 89 If Me._dpiScaleX <= 0.0 OrElse Me._dpiScaleY <= 0.0 Then 90 Me.GetDPIScalingFactor(Me._dpiScaleX, Me._dpiScaleY) 91 End If 92 Dim upgradeWinForm As UpgradeWinForm = New UpgradeWinForm(TYPE_OF_HTML.EXPIRED_A) 93 upgradeWinForm.SetOwnerForEnterKeyDialog(Me) 94 upgradeWinForm.ScaleDialog(Me._dpiScaleX, Me._dpiScaleY) 95 upgradeWinForm.SetExitConfirm(True) 96 Utils.ShowWinForm(upgradeWinForm, Me) 97 If Not upgradeWinForm.m_bCloseByButton Then 98 Me._bForceClose = True 99 MyBase.Close() 100 Return 101 End If 102 End If</pre>		<pre>88 If CultureInfo.CurrentUICulture.Name = "en-US" Then 89 Return 90 End If</pre>
--	---	---

Los cambios los hice mediante <Clic Derecho>Edit IL Instructions...>.



Primero podemos ver la comparación del **IF** que aquí consta de dos instrucciones, luego agregué mis cambios y lo primero que hice fue nopear con la instrucción **NOP** para evitar que mi retorno no devolviera nada, y ahí si puse mi **RET**. Voy a comparar esos cambios en la vista **IL** para hacer mi patrón.

ORIGINAL

```
0x0001A34D 72EB270070 */ IL_0351: ldstr "en-US"
0x0001A352 28AC00000A */ IL_0356: call bool [mscorlib]System.String::op_Equality(string, string)
0x0001A357 2C7F */ IL_035B: brfalse.s IL_03DC

0x0001A359 02 */ IL_035D: ldarg.0
0x0001A35A 7B3E010004 */ IL_035E: ldflld float64 m
0x0001A35F 230000000000000000 */ IL_0363: ldc.r8 0.0
0x0001A368 3111 */ IL_036C: ble.s IL_037F

0x0001A36A 02 */ IL_036E: ldarg.0
```

PARCHEADO

```
0x00023E79 72EB270070 */ IL_0351: ldstr "en-US"
0x00023E7E 28AB00000A */ IL_0356: call bool [mscorlib]System.String::op_Equality(string, string)
0x00023E83 2C7B */ IL_035B: brfalse.s IL_03D8

0x00023E85 00 */ IL_035D: nop
0x00023E86 2A */ IL_035E: ret

0x00023E87 230000000000000000 */ IL_035F: ldc.r8 0.0
0x00023E90 3111 */ IL_0368: ble.s IL_037B
```

Solo quiero hacer notar que podemos ver en el **PARCHEADO** nuestros **BYTES** nuevos y más abajo los **BYTES** sobrantes del **IF** que no serán ejecutados.

BYTES ORIGINALES - UpdateTimer:

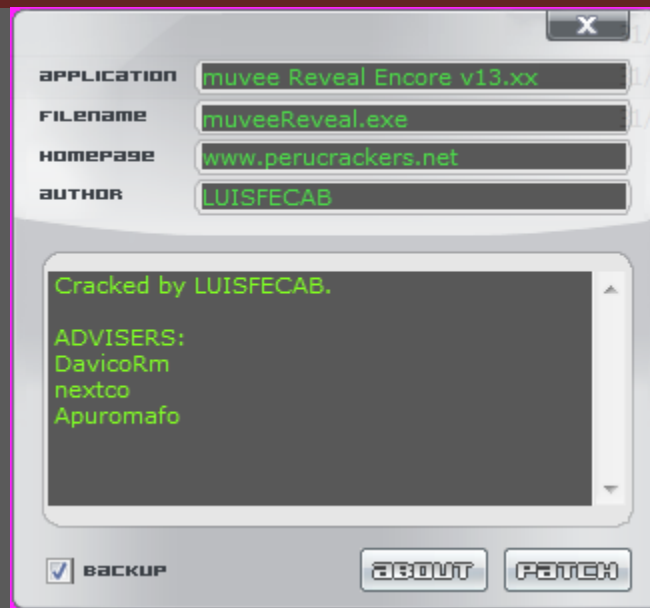
2C7F027B3E0100042300000000

BYTES NUEVOS - IsPurchased MODIFICADO:

????002A0000??0000????????

Listo, agregamos nuestros tres reemplazos como lo explicamos arriba, creamos nuestro **Patch** y lo probamos. Funciona muy bien. Parcheo esas tres partes que no me dejaban tenerlo **Full**.

[muvee Reveal v13.0.0.29340 Build 3157 (.NET)(Crack-Patch)(dnSpy v5.0.7)]



He probado el **Crack** de **nextco** y el **Patch** de **DavicoRm** y juntos no me funcionan completamente, yo creo que han quedado incompletos o puede que el programa que tengo yo sea diferente. Voy a comentarles dónde me fallan.

Empiezo por el **Patch** de **DavicoRm**. Cuando instalo en inglés, no me parchea la rutina de la **NAG** y esta me aparece, y al cerrarla me cierra la aplicación, repito, ocurre esto cuando instalamos en inglés. Otro fallo ocurre cuando iniciamos con otro idioma, ahí se carga **Full** pero en las opciones de guardar se activa la **NAG** en tres casos especiales, abajo en la imagen están señalados. Aquí permite cerrar la NAG sin cerrar la aplicación.



Ahora el **Crack** de **nextco**. Solo falla al seleccionar los tres casos de video, ver la imagen de arriba. Mi **Patch** corrige esos errores, por lo menos en el programa que yo tengo, y espero que funcione en sus programas también. Creo que con eso vamos terminando esta sección que me salió un poco extensa pero si miramos hay muchas cosas nuevas que aprendimos.

PARA TERMINAR

Tremendo target que propuso **Dani**, quién iba pensar que tuviera mucho por aprenderle, por lo menos yo lo veo así porque aprendí a saber cómo cambiar código de mejor manera, hice mi primer **Patch** buscando y reemplazando, también mejoré el uso del **<dnSpy v5.0.7>**.

Algo que me alegra mucho es el trabajo en equipo con mis amigos de **PeruCrackers** y que gracias a eso tenemos como resultado este tutorial. Definitivamente cuando se trabaja en equipo se logran buenas cosas y en Cracking ni hablar, aquí tenemos el resultado.

Bueno, me despido, ya se me acabaron las palabras y solo quiero terminar para poder compartir el tuto con todos ustedes pero no sin antes saludar a la lista de todos **CracksLatinos**, que también esto es para ustedes. Mis saludos y agradecimientos especiales para **Dani**, **RavicoRm**, **nextco** y **Apuromafo**.

Se despide su amigo,

@LUISFECAB