

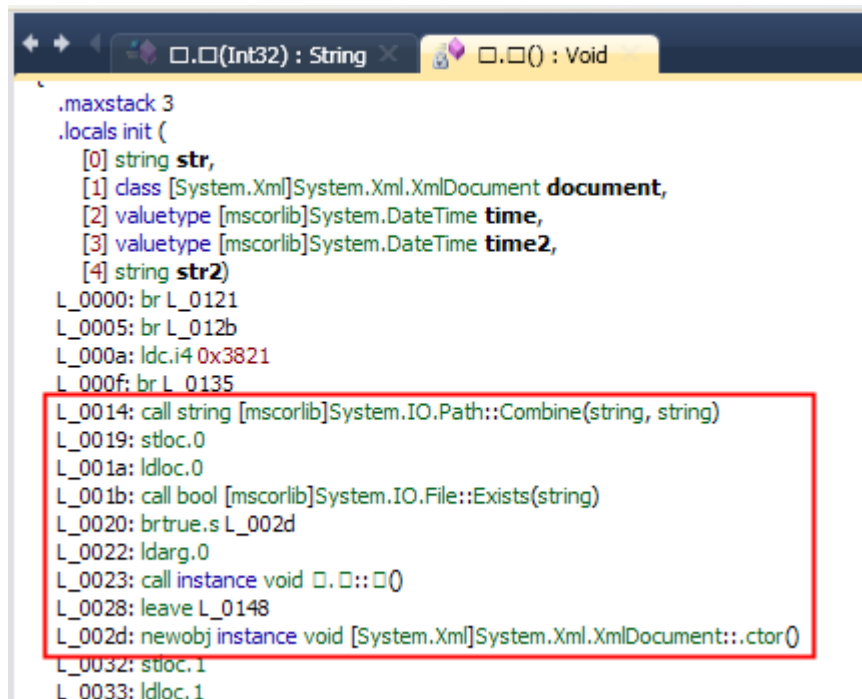
Acerca de Jamcast version 1.5.1.111



Solucion final

Saludos gente. Si leyeron el tuto que escribí hace unos dias sobre este software, recordaran que me quedó pendiente el asunto del archivo de licencia que contenía una estructura típica XML. De esa estructura se toman los datos para saber si el software está en periodo trial, si está vencido o si por el contrario cuenta con una licencia valida para utilizarla bastante tiempo.

Del tuto anterior, en esta zona del programa se trabaja con el archivo de licencia:

A screenshot of a debugger's assembly view. The window has a title bar with two tabs: "□.□(Int32) : String" and "□.□() : Void". The assembly code is listed on the left, with a red rectangle highlighting a specific block of instructions. The instructions include stack frame setup, variable declarations, and calls to system methods like Path.Combine and File.Exists.

```
.maxstack 3
.locals init (
    [0] string str,
    [1] class [System.Xml]System.Xml.XmlDocument document,
    [2] valuetype [mscorlib]System.DateTime time,
    [3] valuetype [mscorlib]System.DateTime time2,
    [4] string str2)
L_0000: br L_0121
L_0005: br L_012b
L_000a: ldc.i4 0x3821
L_000f: br L_0135
L_0014: call string [mscorlib]System.IO.Path::Combine(string, string)
L_0019: stloc.0
L_001a: ldloc.0
L_001b: call bool [mscorlib]System.IO.File::Exists(string)
L_0020: brtrue.s L_002d
L_0022: ldarg.0
L_0023: call instance void □.□::□()
L_0028: leave L_0148
L_002d: newobj instance void [System.Xml]System.Xml.XmlDocument::.ctor()
L_0032: stloc.1
L_0033: ldloc.1
```

El programa tomaba una ruta hacia el archivo jamcast.lic, verificaba si dicho archivo existía y finalmente, si lo encontraba, trabajaba con él. Cargaba el contenido del archivo jamcast.lic y trabajaba con él como documento XML.

Un documento basico en formato XML sería este:

```
<?xml version="1.0" encoding="utf-8"?>
<Contenido>
</Contenido>
```

Esto ya sería suficiente para que el programa lo cargara, pero obviamente, faltan datos. Para darle algo de personalidad y se haga mencion a la utilidad que va a tener este documento XML, lo pongo así:

```
<?xml version="1.0" encoding="utf-8"?>
<Licencia>
</Licencia>
```

Para saber qué datos faltan, solo hay que mirar el código:

```

L_0035: callvirt instance void [System.Xml]System.Xml.XmlDocument::Load(string)
L_003a: ldloc.1
L_003b: callvirt instance class [System.Xml]System.Xml.XmlElement [System.Xml]System.Xml.XmlDocument::get_DocumentElement()
L_0040: callvirt instance class [System.Xml]System.Xml.XmlAttributeCollection [System.Xml]System.Xml.XmlNode::get_Attributes()
L_0045: ldc.i4 0x3832
L_004a: call string [System.Xml]System.Xml.XmlAttributeCollection::get_ItemOf(int32)
L_004f: callvirt instance class [System.Xml]System.Xml.XmlAttribute [System.Xml]System.Xml.XmlAttributeCollection::get_ItemOf(string)
L_0054: callvirt instance string [System.Xml]System.Xml.XmlNode::get_Value()
L_0059: call valuetype [mscorlib]System.DateTime [mscorlib]System.Convert::ToDateTime(string)
L_005e: stloc.2
L_005f: ldloc.2
L_0060: call valuetype [mscorlib]System.DateTime [mscorlib]System.DateTime::get_UTCNow()
L_0065: call bool [mscorlib]System.DateTime::op_LessThanOrEqual(valuetype [mscorlib]System.DateTime, valuetype [mscorlib]System.DateTime)
L_006a: brfalse.s L_0077
L_006c: ldarg.0
L_006d: call instance void [System.Xml]System.Xml.XmlDocument::Load(string)
L_0072: leave L_0148
L_0077: ldloc.1

```

Una vez que se cargó el documento, entre otras cosas, en la línea L_0040 se piden los atributos del documento. Los atributos son como características o propiedades de los elementos en el documento. Si la info fuera de un auto, sería algo así:

```

<?xml version="1.0" encoding="utf-8"?>
<Auto color="Rojo" antigüedad="Tres años" kilometros="5000">
</Auto>

```

Volviendo al código en Reflector, se toman los atributos y se carga un entero que recuerden es para descryptar una cadena de texto que está en formato Base64, en los recursos que utiliza Jamcast para su funcionamiento. El primer entero que se carga, es el hexadecimal 0x3832. Se entra en la llamada encargada de descryptar y la cadena resultante es "expires".

De nuevo hago referencia al tuto que escribí para que recuerden cómo trabajaba la llamada.

Bien, teniendo la cadena "expires", en la línea L_004f se toma el ítem "expires" y en la siguiente línea se toma su valor. Si vamos conformando el documento XML, ya tendría que estar así:

```

<?xml version="1.0" encoding="utf-8"?>
<Licencia expires="31/12/2011">
</Licencia>

```

El valor del ítem "expires" es una fecha y si se fijan le tengo puesto el "31/12/2011". Bien, una vez tomada esa string, en la línea L_0059 se convierte en una estructura típica de fecha. En la línea L_0060 se toma la fecha actual y en la siguiente línea L_0065 se comparan las fechas. System.DateTime::op_LessThanOrEqual nos da una idea de lo que se compara. Si la fecha obtenida del XML es menor que o igual al día actual, obtendríamos un booleano True o Verdadero y el salto de la línea L_006a NO se produciría y nos metería de cabeza en la llamada que nos saca el mensaje de chico malo en la línea L_006d. Como en este caso, del XML extrajo el 31-12-2011, no es ni menor ni igual que la fecha actual y el booleano sería False. Entonces, el salto sí saltaría por encima del mensaje de chico malo y nos iríamos a la línea L_0077. La vemos:

```

L_0077: ldloc.1
L_0078: callvirt instance class [System.Xml]System.Xml.XmlElement [System.Xml]System.Xml.XmlDocument::get_DocumentElement()
L_007d: callvirt instance class [System.Xml]System.Xml.XmlAttributeCollection [System.Xml]System.Xml.XmlNode::get_Attributes()
L_0082: ldc.i4 0x383f
L_0087: call string [System.Xml]System.Xml.XmlAttributeCollection::get_ItemOf(int32)
L_008c: callvirt instance class [System.Xml]System.Xml.XmlAttribute [System.Xml]System.Xml.XmlAttributeCollection::get_ItemOf(string)
L_0091: callvirt instance string [System.Xml]System.Xml.XmlNode::get_Value()
L_0096: call valuetype [mscorlib]System.DateTime [mscorlib]System.Convert::ToDateTime(string)

```

Aquí, se vuelve a hacer lo mismo que antes: se toma un entero que en este caso es el 0x383f que correspondería a la cadena encriptada "created". Bien, se vuelve a tomar el documento XML y se busca el ítem "created", se toma su valor y se convierte también a estructura de fecha. Por lo que llevo viendo del programa, es como si hiciera referencia a cuando se instaló el programa y yo en mi caso, le puse al archivo XML la fecha "24-6-2011". Dicho documento XML ya tendría este aspecto:

```
<?xml version="1.0" encoding="utf-8"?>
<Licencia expires="31/12/2011" created="24/6/2011">
</Licencia>
```

Bien seguimos con el código:

```
L_009d: callvirt instance class [System.Xml]System.Xml.XmlElement [System.Xml]System.Xml.XmlDocument::get_DocumentElement()
L_00a2: callvirt instance class [System.Xml]System.Xml.XmlAttributeCollection [System.Xml]System.Xml.XmlNode::get_Attributes()
L_00a7: ldc.i4 0x384c
L_00ac: call string [System.Xml]System.Xml.XmlAttribute::get_Value()
L_00b1: callvirt instance class [System.Xml]System.Xml.XmlAttribute [System.Xml]System.Xml.XmlAttributeCollection::get_ItemOf(string)
L_00b6: callvirt instance string [System.Xml]System.Xml.XmlNode::get_Value()
L_00bb: stloc.s str2
L_00bd: ldarg.0
L_00be: ldftld class [System.Windows.Forms]System.Windows.Forms.Label [System.Windows.Forms]System.Windows.Forms.Control::set_Text(string)
L_00c3: ldloc.s str2
L_00c5: callvirt instance void [System.Windows.Forms]System.Windows.Forms.Control::set_Text(string)
```

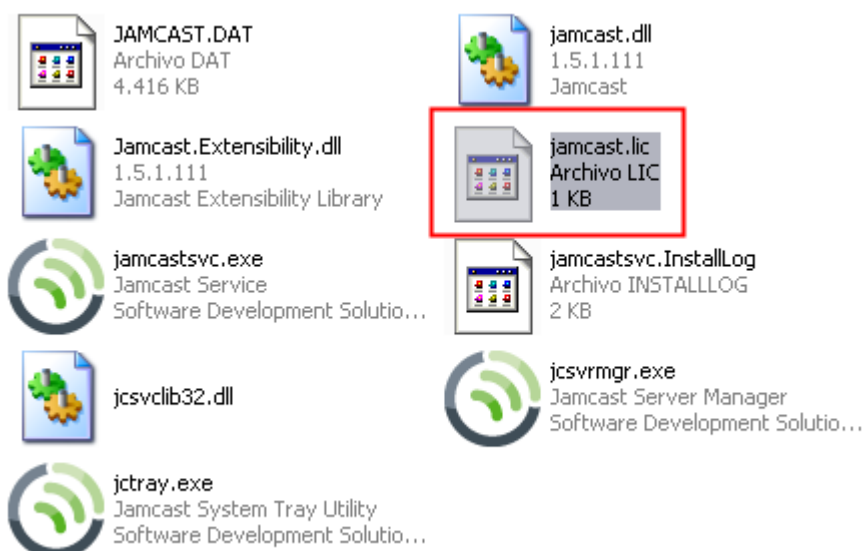
Aquí lo mismo: se toma el documento XML y se utiliza el entero 0x384c que hace referencia a la cadena encriptada "email" y se toma su valor. Vuelvo a modificar el documento XML para que el programa pueda trabajar con el:

```
<?xml version="1.0" encoding="utf-8"?>
<Licencia expires="31/12/2011" created="24/6/2011" email="sequeyo@gmail.com">
</Licencia>
```

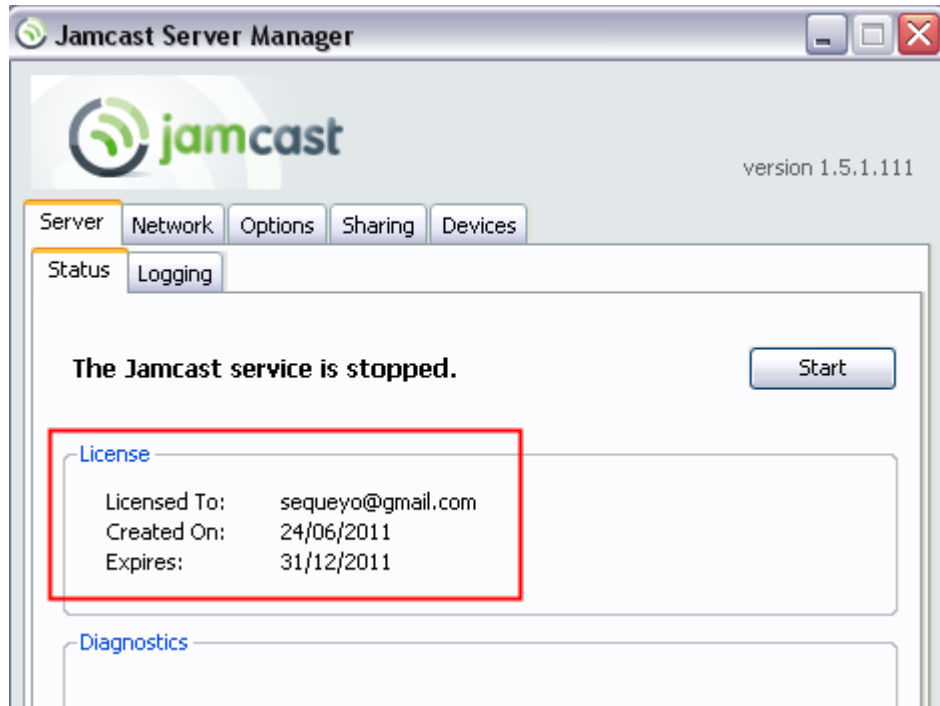
En el resto del código, como ven, se trabajan con etiquetas o Label propias del aspecto que tendrá el programa:

```
L_00ef: ldloc.s time
L_00f1: call instance string [mscorlib]System.DateTime::ToShortDateString()
L_00f6: br.s L_0102
L_00f8: ldc.i4 0x3855
L_00fd: call string [System.Windows.Forms]System.Windows.Forms.Control::set_Text(string)
L_0102: ldarg.0
L_0108: ldftld class [System.Windows.Forms]System.Windows.Forms.Panel [System.Windows.Forms]System.Windows.Forms.Control::set_Visible(bool)
L_010d: ldc.i4.1
L_010e: callvirt instance void [System.Windows.Forms]System.Windows.Forms.Control::set_Visible(bool)
L_0113: ldarg.0
L_0114: ldftld class [System.Windows.Forms]System.Windows.Forms.Label [System.Windows.Forms]System.Windows.Forms.Control::set_Visible(bool)
L_0119: ldc.i4.0
L_011a: callvirt instance void [System.Windows.Forms]System.Windows.Forms.Control::set_Visible(bool)
```

Aspectos del formulario que se hacen visibles y que antes, por culpa del trial expirado no se veían. Pues bien, así de sencillo es solucionar esto. El pequeño documento en formato XML lo meto dentro del archivo jamcast.lic y lo coloco en la carpeta donde se instaló el programa:



Abro el Jamcast.....



Para no pasarme tampoco mucho, le puse que expira en el 31 de Diciembre de este año. Como ven, licenciado con mi email. Ya cada uno, si quieren jugar con el archivo de licencia y ponerlo para que funcione hasta el año 3000 pues adelante, pueden probar.

Yo aconsejo, para que el autor no se enfade con nosotros, que lo pongan para que les trabaje unos días mas y luego le compren al muchacho una licencia y así pasen a utilizarlo legalmente. Yo, para que no se enfade conmigo, le comento que lo voy a desinstalar, pues yo no le voy a dar utilidad ninguna y no lo quiero de momento.

Espero que le sirva a alguien, para probarlo solo unos días mas.

Por otra parte, espero le sirva a otros para aprender algo mas de los .NET's y su funcionamiento.

Me despido hasta la proxima, un saludo

S E Q U E Y O