SHUB-NIGURRATH, ZYZYGY, GOOGLEPLEX

Primer on Reversing .NET Applications

Version 1.2
October 2006

Table of Contents

Editor: Shub-Nigurrath

1. Forewords

Finally we have time to publish a decent basic tutorial on .NET. I should thanks zyzygy and GooglePlex who submitted to us their tutorials about .NET.

I thought to merge them into a unique tutorial, and also thought to apply to it the new template we are going to use for our tutorials.

The world of .NET applications has seen a tremendous improvement and finally instruments are mature enough to successfully patch applications. Of course this is a first tutorial for which we will try to introduce the procedures and instruments one can use to decompile and patch applications.

Several extremely good tutorials has been published on this subject but mainly for laziness we didn't have one available..till now ☺

Anyway I added at the end a complete or rich at least list of references for further reading and improving your understanding of this world. Do not underestimate the difficulties you might find.

Ok, time to go! GooglePlex and zyzygy will describe how to patch two applications using decompilation, modifying the MSIL source code (few MSIL details will be given) and then recompiling the new program. I will show instead at the end on another application how to do the same without recompilation, using the MSIL bytecode specifications. zyzygy at the end again, will use *PeBrosePro* to live debugging a .NET application.

Have phun,
Shub

Disclaimers

Verification

ARTeam.esfv can be opened in the ARTeamESFVChecker to verify all files have been released by ARTeam and are unaltered. The ARTeamESFVChecker can be obtained in the release section of the ARTeam site: http://releases.accessroot.com

Table of Contents

1. Reversing .NET and a License File Check, GooglePlex

1.1. Abstract

The purpose of this tutorial is to learn how to reverse a .NET-application, which also is protected by a license file check.

The application that we are going to reverse is EngiSSol's 2D Frame Analysis Dynamic Edition (or just 2D Truss – you'll see later why).

The tools used for this are Lutz Roeder's .NET Reflector, Microsoft .NET SDK, Notepad, and - of course – brains. No OllyDbg, though. It can't reverse .NET-applications ;(

1.2. How to crack this nut

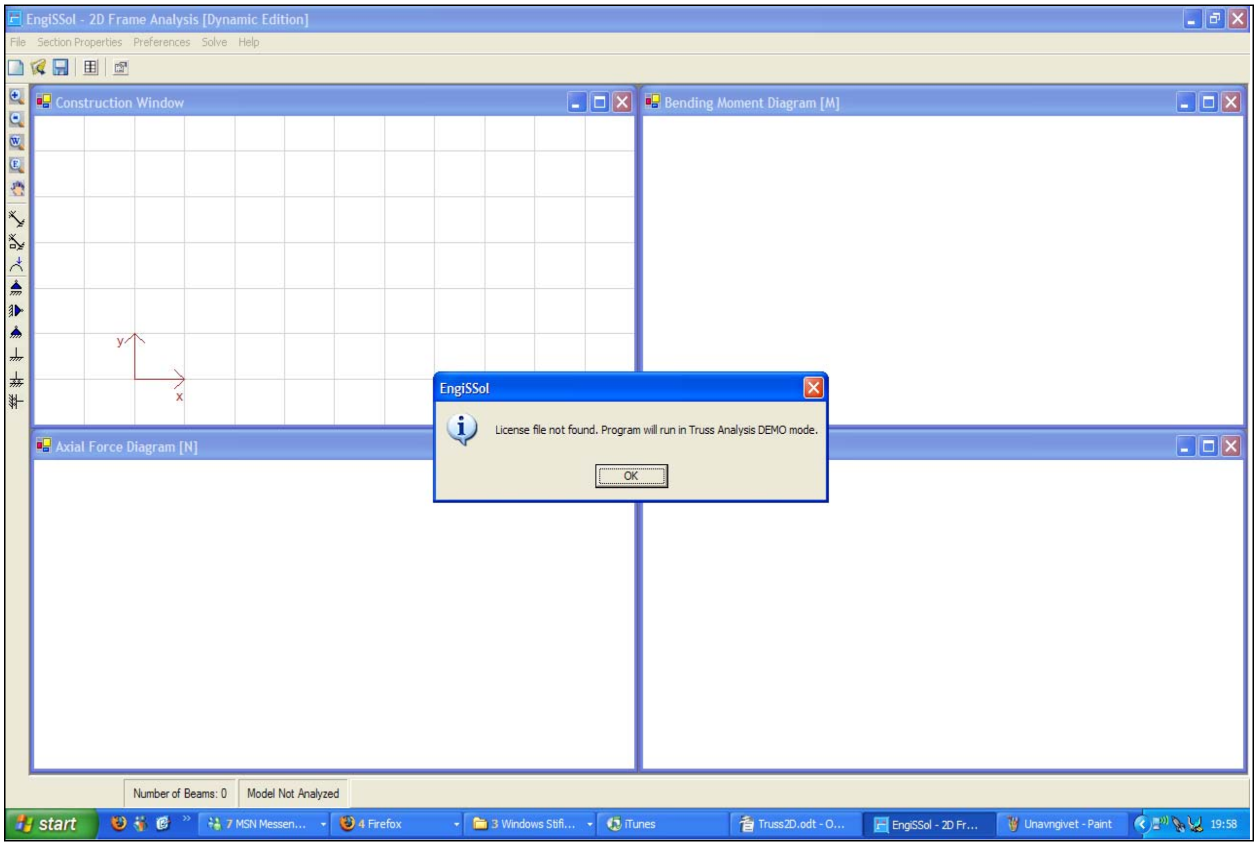
1.2.1 Preparation

This time, I've done some preparations for you. I've had opened our target in PeiD and found out that it was a .NET application. This forces us to use other software than good ol' Olly, but it really makes our job easier (thanks, Microsoft) because we now can decompile our executable to IL – Intermediate Language – which is much more readable than assembly. We can also recompile the IL to EXE (thanks again, Microsoft).

1.2.2 Checking out the target

It is always a good idea to see how the program reacts because the program's reactions to your inputs are often important flags to search for.

So, let's try to fire up our target and see what happens (Figure 1):


EngiSSol - 2D Frame Analysis [Dynamic Edition]
File Section Properties Preferences Solve Help
Construction Window
Bending Moment Diagram [M]
Axial Force Diagram [N]
EngiSSol
OK
Number of Beams: 0
Model Not Analyzed
start

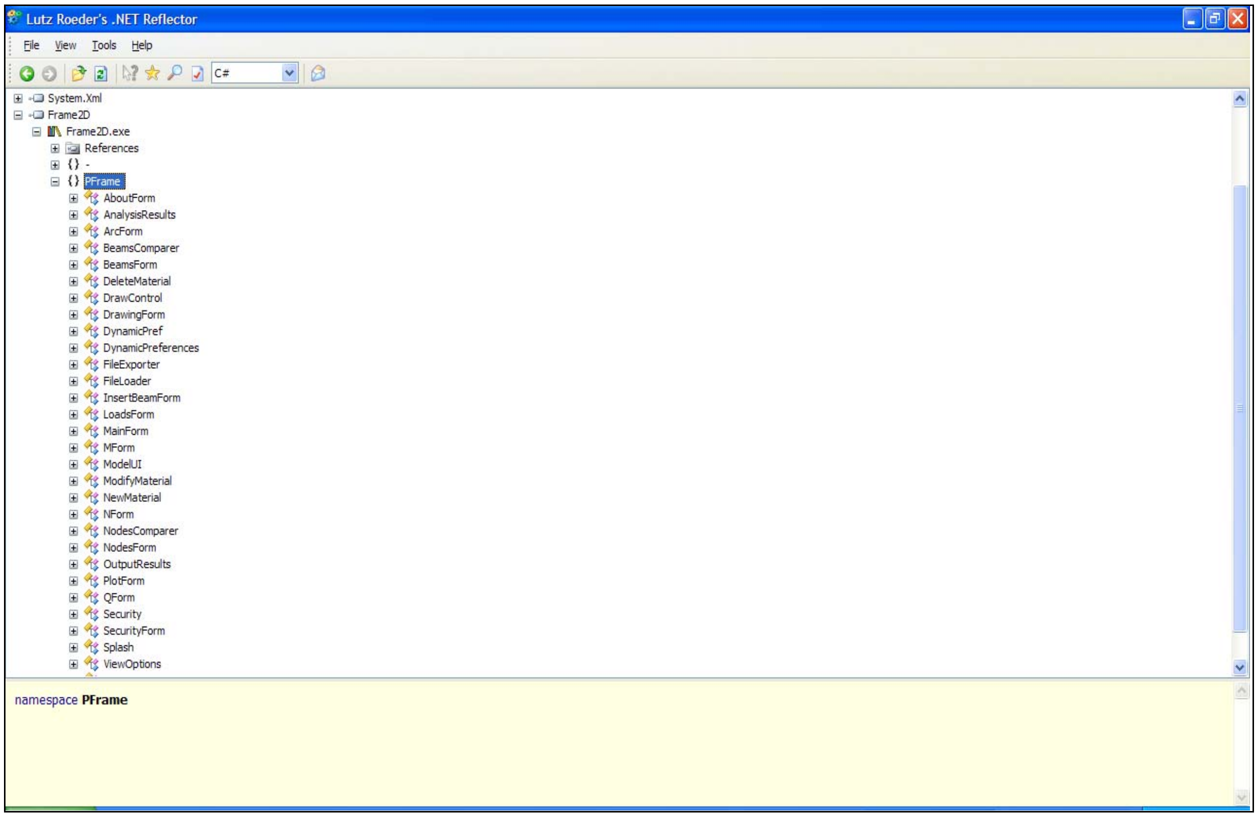
License file not found. Program will run in Truss Analysis DEMO mode.

Figure 1 - Oh no! We've got no license!

Hmm...see the title? "2D Frame Analysis [Dynamic Edition]"? And now it will switch to Truss Analysis DEMO? Seems like it's a multiple-version application...

1.2.3 Opening the target in .NET Reflector

Let's open our target in .NET Reflector – Reflector might ask about a default list but just pick the newest you've got. Go to the tree "Frame2D" → "Frame2D.exe" → "Pframe" (see Figure 2):



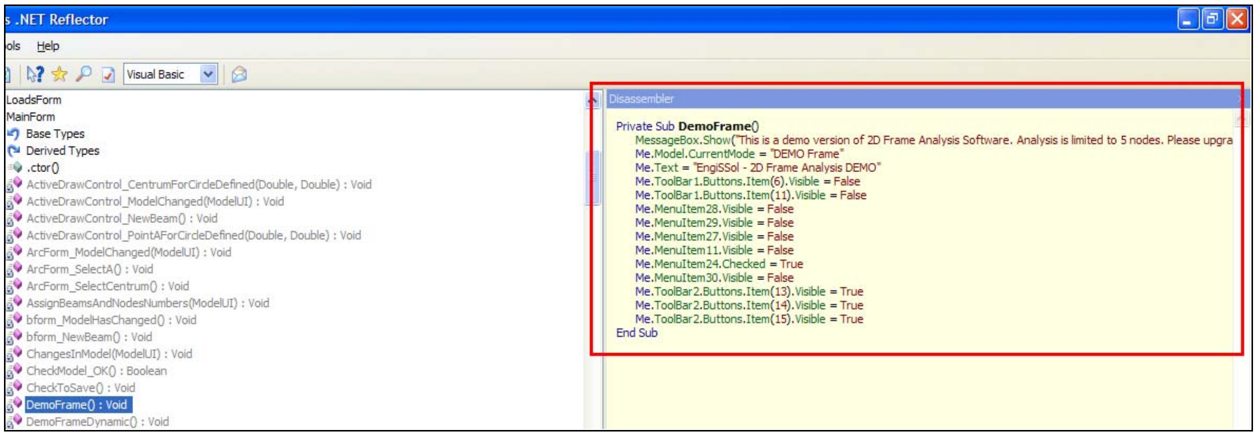
```
System.Xml
Frame2D
  Frame2D.exe
    References
    { } -
    { } PFrame
      AboutForm
      AnalysisResults
      ArcForm
      BeamsComparer
      BeamsForm
      DeleteMaterial
      DrawControl
      DrawingForm
      DynamicPref
      DynamicPreferences
      FileExporter
      FileLoader
      InsertBeamForm
      LoadsForm
      MainForm
      MForm
      ModelUI
      ModifyMaterial
      NewMaterial
      NForm
      NodesComparer
      NodesForm
      OutputResults
      PlotForm
      QForm
      Security
      SecurityForm
      Splash
      ViewOptions

namespace PFrame
```

Figure 2 - Our target in .NET Reflector

Lovely.

.NET is neat because the executable is never REALLY compiled – all the source code can be read. And yes, Reflector does the job ;).
You can see the source from IL through C# and Visual Basic to Delphi. Quite nice because you can easier analyze you through what the program does if you know just a little high-level language. Just too bad it can't dump it in high-level language ;(.

Now, since our license file check initialized in sync with the application, it is rather interesting to find the main form. Go to "MainForm" in Reflector and expand the tree.
It is stuffed with things that happen within our MainForm. Check out "DemoFrame() : void" - right click and select "Disassembler". If it asks you to manually resolve stuff, just skip it.
You'll see this (I've chosen Visual Basic as the high-level language, but you can alter that in the toolbar) (Figure 3):



```
LoadsForm
MainForm
  Base Types
  Derived Types
  .ctor()
  ActiveDrawControl_CentrumForCircleDefined(Double, Double) : Void
  ActiveDrawControl_ModelChanged(ModelUI) : Void
  ActiveDrawControl_NewBeam() : Void
  ActiveDrawControl_PointAForCircleDefined(Double, Double) : Void
  ArcForm_ModelChanged(ModelUI) : Void
  ArcForm_SelectA() : Void
  ArcForm_SelectCentrum() : Void
  AssignBeamsAndNodesNumbers(ModelUI) : Void
  bform_ModelHasChanged() : Void
  bform_NewBeam() : Void
  ChangesInModel(ModelUI) : Void
  CheckModel_OK() : Boolean
  CheckToSave() : Void
  DemoFrame() : Void
  DemoFrameDynamic() : Void
```

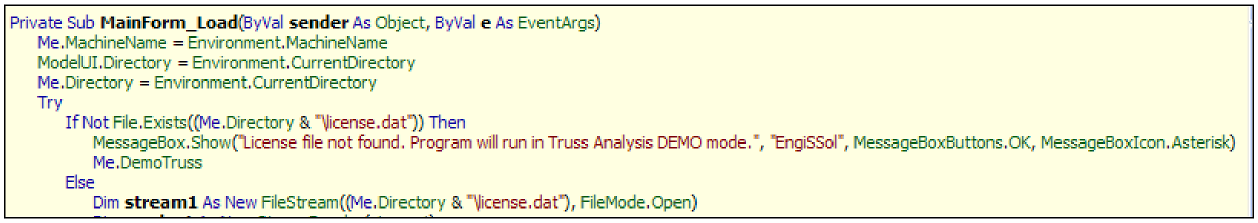
```vb
Private Sub DemoFrame()
    MessageBox.Show("This is a demo version of 2D Frame Analysis Software. Analysis is limited to 5 nodes. Please upgra
    Me.Model.CurrentMode = "DEMO Frame"
    Me.Text = "EngiSSol - 2D Frame Analysis DEMO"
    Me.ToolBar1.Buttons.Item(6).Visible = False
    Me.ToolBar1.Buttons.Item(11).Visible = False
    Me.MenuItem28.Visible = False
    Me.MenuItem29.Visible = False
    Me.MenuItem27.Visible = False
    Me.MenuItem11.Visible = False
    Me.MenuItem24.Checked = True
    Me.MenuItem30.Visible = False
    Me.ToolBar2.Buttons.Item(13).Visible = True
    Me.ToolBar2.Buttons.Item(14).Visible = True
    Me.ToolBar2.Buttons.Item(15).Visible = True
End Sub
```

Figure 3 - The code of the procedure DemoFrame()

If you investigate further, there are also procedures called like "DemoFrameDynamic" and "DemoTruss". As the illustration suggests, the procedures tell the program which features that are unlocked and which are not. This confirms my theory about a multi-version program ;).

Seen the message from the illustration before? Yup, that's the second warning you get in the "demo" version of the application! We're close.

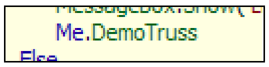
Scroll further down. What do we see? A procedure called "MainForm_Load(Object, EventArgs) : void". Nice! Disassemble that! See Figure 4:



```
Private Sub MainForm_Load(ByVal sender As Object, ByVal e As EventArgs)
    Me.MachineName = Environment.MachineName
    ModelUI.Directory = Environment.CurrentDirectory
    Me.Directory = Environment.CurrentDirectory
    Try
        If Not File.Exists((Me.Directory & "\license.dat")) Then
            MessageBox.Show("License file not found. Program will run in Truss Analysis DEMO mode.", "EngiSSol", MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
            Me.DemoTruss
        Else
            Dim stream1 As New FileStream((Me.Directory & "\license.dat"), FileMode.Open)
```

Figure 4 - Isn't that our first warning? ;)

Oh yes indeed. It checks if the license.dat file exists or not. If it doesn't exist it shows the warning and loads the demo version! See Figure 5:



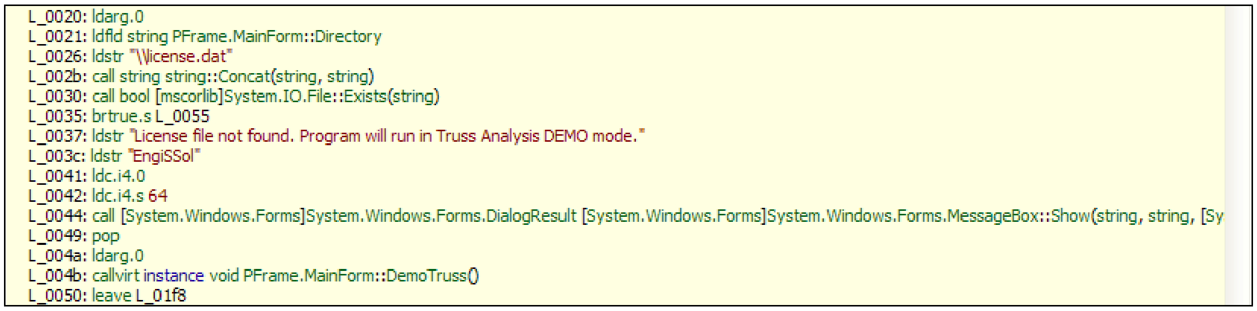
```
Me.DemoTruss
```

Figure 5 - The "Demo" is loaded!

If you dig deeper into the code, you'll find that the application decrypts the content of the license file to a string that starts with e.g. "Frame Dynamic" depending on the license. It then runs the appropriate procedure and therefore the version of the program.

We could low-level patch the application by altering all references to the license check procedure but it will take too long. We could also reverse the decryption procedure to make our own keygen (only KINGS make keygens – and we've got the code somewhere :D), but that will also take too long.
Instead, we will remove the demo-nag and make it jump immediately to the best program version. It requires no license file ;).

So, when still focused on our little nag, change language to IL. I suggest you print the source, because it's WAY longer than the high-level language.
Now we need to locate our patches.
This seems interesting (Figure 6):



```
L_0020: ldarg.0
L_0021: ldfld string PFrame.MainForm::Directory
L_0026: ldstr "\\license.dat"
L_002b: call string string::Concat(string, string)
L_0030: call bool [mscorlib]System.IO.File::Exists(string)
L_0035: brtrue.s L_0055
L_0037: ldstr "License file not found. Program will run in Truss Analysis DEMO mode."
L_003c: ldstr "EngiSSol"
L_0041: ldc.i4.0
L_0042: ldc.i4.s 64
L_0044: call [System.Windows.Forms]System.Windows.Forms.DialogResult [System.Windows.Forms]System.Windows.Forms.MessageBox::Show(string, string, [Sy
L_0049: pop
L_004a: ldarg.0
L_004b: callvirt instance void PFrame.MainForm::DemoTruss()
L_0050: leave L_01f8
```

Figure 6 - The source in IL

I've tried to "translate" the IL to some more understandable to know where to patch. These "translations" are probably not 100 % correct, since I have no experience in this language:

L_0021:	ldfld string PFrame.MainForm::Directory	Push the application's path
L_0026:	ldstr "\\license.dat"	Push the license file name
L_002b:	call string string::Concat(string, string)	Call the strings above
L_0030:	call bool [mscorlib]System.IO.File::Exists(string)	Check if the file exists via mscorlib.dll
L_0035:	brtrue.s L_0055	If true, go to line 0055, else...
L_0037:	ldstr "License file not found. Program will run in Truss Analysis DEMO mode."	...push the warning
L_003c:	ldstr "EngiSSol"	Push the warning's title