

tKC

.NET reversing Tips

2



EntryPoint Method

Attention

Published for educational Purposes only

Finding Entrypoint Method [EPM]

Entrypoint method is first method called on your .net application startup, and it's very Important to be able to locate it in reflector or Ildasm, in a simple .net application it may Look like this:-

```
Public Shared Sub Main()  
  
    Application.Run(New MainForm)  
  
End Sub
```

The importance of this method is that you can start tracing the program actions from the beginning until you reach the protection scheme which is usually checked on Startup!

The second thing you can benefit from this Method is identifying the MainForm Class which will be used as a main form for the targeted application, If you see the **Application.run** method wandering around then you must keep an eye on the passed Arguments to this function.

To find the entrypoint RawData offset you must do this :

1. First open the targeted application in CFF explorer.
2. Go to the **".NET directory"** node
3. You will see a grid listing many things now, Just find the **"EntrypointToken"** row now.
4. now type the Last column value for this row and it's a Dword value that will guide us to the entrypoint method

The value that you have just typed takes this format **00000000** and suppose it's **06000015** here, you might be wondering what a token is ? It's a dword value that represents a table and an index into that table, i.e. It points to a table and to a row in that table Too, for example our **06000015** token here can be read like this.

06	000015
Table index	Row index in that table

The table we are talking about here is the **Methods** table, and you can view it in CFF if You go to **"MetaData Tables"** node, you will see a tree with gray nodes, find the **Methods** table and expand it, and you will see a list of nodes indexed from 1 and up.

The nodes in that table represent every method used in the targeted .net application, now back To **000015** value which equals **21** in decimal, find the node indexed **21** and click it.

A small table on the right listing some info about this method will show, what we are interested in here is the first row which represents the RVA for that Method, read its Value column [last column] and type it. say it's **0xC9A4**

Finding file offset for the EPM with CFF

A .net PE file contains three sections: **.text .reloc .rsrc**
 The .text section contains the Import Table, the Import Address Table and the **.NET Section.**

Now let's assume that

ImageBase for a .net PE file is **0x400000**
 .text section virtual address is **0x2000**
 .text section Raw Address is **0x200**
 EntryPoint Method virtual address is **0xC9A4**

When file is mapped to memory it will look like this

0x400000	0x402000	0x40C9A4	← RVA

ImageBase >>>	.text >>>	EP_Method	

0x0	0x2000	0xC9A4	← VA

According to this :

We will find the **.text section** data **0x2000 bytes** from Imagebase when file Is mapped to memory.

We will find the the Method data **0xC9A4 bytes** from Imagebase when file is mapped to memory.

Ok, now we will calculate the offset at which we will find the ep_method in the .text section.

$$\begin{aligned}
 &= [\text{EP_Method VA}] - [\text{.text section VA}] \\
 &= 0xC9A4 - 0x2000 \\
 &= 0xA9A4
 \end{aligned}$$

So the method is **0xA9A4** bytes into this .text section
 i.e. The method data begins at **0xA9A4** in the .text section data.

If we use the .text section RawData Offset we can calculate the RawData Offset for the method too.

$$\begin{aligned}
 \text{Method RawData Offset} &= \text{.text section RawData Offset} + 0xA9A4 \\
 &= 0x200 + 0xA9A4 \\
 &= 0xABA4
 \end{aligned}$$

So the method offset in the file is **0xABA4**

We will come to this now :

EntryPoint Method File Offset = [EntryPoint VirtualAddress] -
[Section.txt VirtualAddress] + [Section.txt RawAddress]

These 3 variables can be obtained from CFF explorer.

I hope this was easy, if you don't get it then read it again.

You can use the Address converter in CFF to calculate the file offset for Any method if you have the RVA.

Finding the file offset for the EPM with Ildasm

This is an easier job, all you need to know is a series of actual bytes
From the Entrypoint Method disassembly, this method can be used for any Method not only EPM.

This is an example:

```
-----  
.method public hidebysig static void Main() cil managed  
// SIG: 00 00 01  
{  
  .entrypoint  
  .custom instance void  
[mscorlib]System.STAThreadAttribute::.ctor() = ( 01 00 00 00 )  
  // Method begins at RVA 0xc538  
  // Code size 11 (0xb)  
  .maxstack 8  
  IL_0000: /* 73 | (06)000012 */ newobj instance  
void Scroller.Form1::.ctor()  
  IL_0005: /* 28 | (0A)000001 */ call void  
[System.Windows.Forms]System.Windows.Forms.Application::Run(class [System.Windows.Forms]System.Windows.Forms.Form)  
  IL_000a: /* 2A | */ ret  
} // end of method Form1::Main  
-----
```

Ok this a disassembly for the entrypoint method from a simple .net application,
type the sequence of IL instructions in this method and
Then use a hex editor to find the series of these bytes, here you will be Searching
for this series

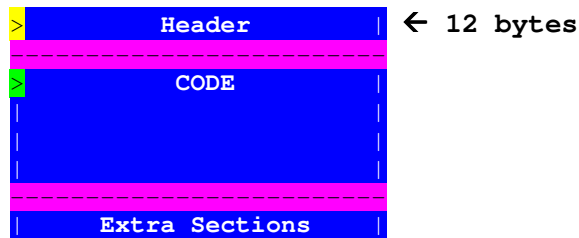
From IL_0000	7312000006
From IL_0005	280100000A
From IL_000a	2A

So the entire series is **73,12,00,00,06,28,01,00,00,0A,2A**
Usually 10 bytes are more than enough to find the right offset.

There is one last thing I should mention here, this method will lead you

To the offset of first actual byte, but the previous one will take you to
The Method header bytes which comes before the Code bytes.

A .net method structure



The first method will take you to > but the this will take you to >, the
Header size is not that large so if you use the first method then you will have to
add **12** bytes [[Still checking this number](#)] to find the first actual byte of code.

Finding the EntryPoint Method node in Ildasm Tree

So you wanna see the decompilation of entrypoint method ?
After you knew the EntryPoint method RVA from CFF explorer, it's time
To have look at its code.

You can use both ILDasm and Reflector for this job, but remember that
ILDasm will only show decompilation for .net methods in IL format, If you
Are lucky then Reflector will decompile the Entrypoint method code to your
Favorite .NET language or you will have to depend on Ildasm to analyze the
Code.

Both Ildasm and Reflector show the assembly classes in a tree view style, but
only Ildasm will tell you the RVA value for every method you decompile.

Look at this code from Ildasm

```
=====
.method public hidebysig static void  Main() cil managed
// SIG: 00 00 01
{
    .entrypoint
    .custom instance void
[mscorlib]System.STAThreadAttribute::.ctor() = ( 01 00 00 00 )
    // Method begins at RVA 0xc538
    // Code size      11 (0xb)
    .maxstack 8
    IL_0000: /* 73 | (06)000012 */ newobj      instance
void Scroller.Form1::.ctor()
    IL_0005: /* 28 | (0A)000001 */ call      void
[System.Windows.Forms]System.Windows.Forms.Application::Run(cl
ass [System.Windows.Forms]System.Windows.Forms.Form)
    IL_000a: /* 2A | */ ret
} // end of method Form1::Main
```

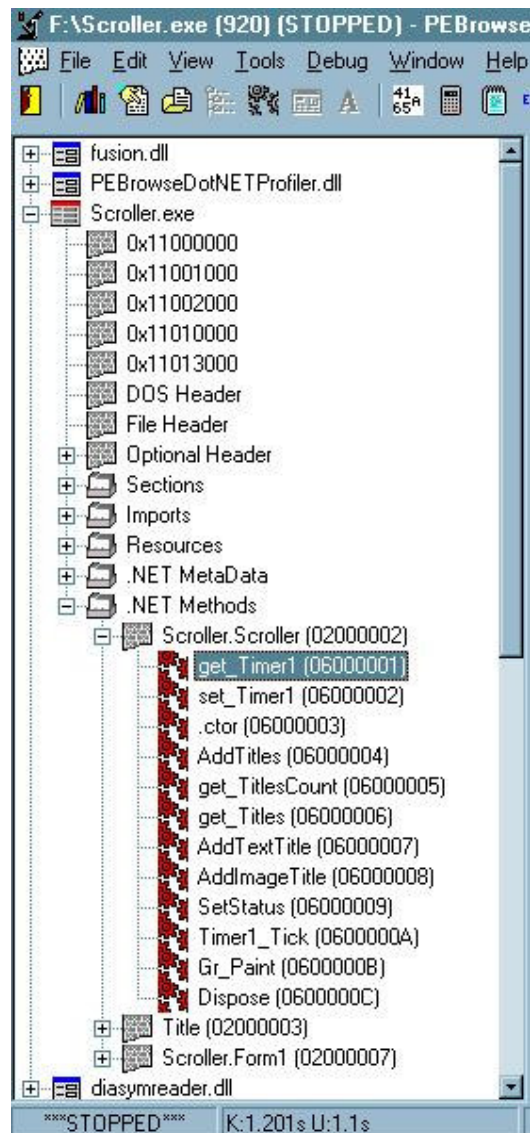
Most of the time you will face obfuscated code, and this means
You can't know which node in Ildasm is the Entrypoint method, and it
Will become harder when there are hundreds or thousands of nodes !

You already know the Entrypoint Method RVA from CFF explorer, Right ?
Now we will fish for the entrypoint method node, in ILDasm you have to
Decompile some method in any class you pick randomly and see its RVA value In
the disassembly text just like above, If it's higher than your EPM RVA Then you
should pick a higher level node, the higher the node the less RVA For its methods,
and as you go down the RVA value will increase, so this Should take a minute or
2 to find the entrypoint method node in ILDasm.

Using EntryPoint Method with PEBrowse Debugger

After you knew the Entrypoint token from CFF explorer for the targeted
application, you can use that token to find the entrypoint method node in
PEBrowse and add a
Breakpoint to make the .net application break as soon as the EPM is JIT-
Compiled, so
This is what you have to do:

1. Open the targeted application in PEBrowse and wait until all libraries and modules are loaded
2. PEBrowse will stop just before EPM is called, so it's time to find the node and add a Breakpoint there.
3. The .net modules will have a red icon in the modules tree, our .net application loaded here is "**Scroller.exe**", go to the "**.NET Methods**" node and you will see that every class is listed there with its methods.
4. You will see the token for every method beside its name, the "get_Timer1" token is displayed and it's **06000001**
5. Because you already know the EPM token from CFF, you can search for the right node here, as you go down the token value will increase just like RVA in ILDasm and that's helpful.
6. As soon as you find the right node, right-click it and then click on the "Add Breakpoint" menu



THE END

That's all for now :-)

I hope you enjoy this tutorial, See you soon with next chapter [If I ever write it !]

tKC

Tuesday, June 27, 2006

www.reteam.org