readme

packed (rev)

> This writeup is based on the challenge author explanation himself, Fareed Fauzi

TL;DR

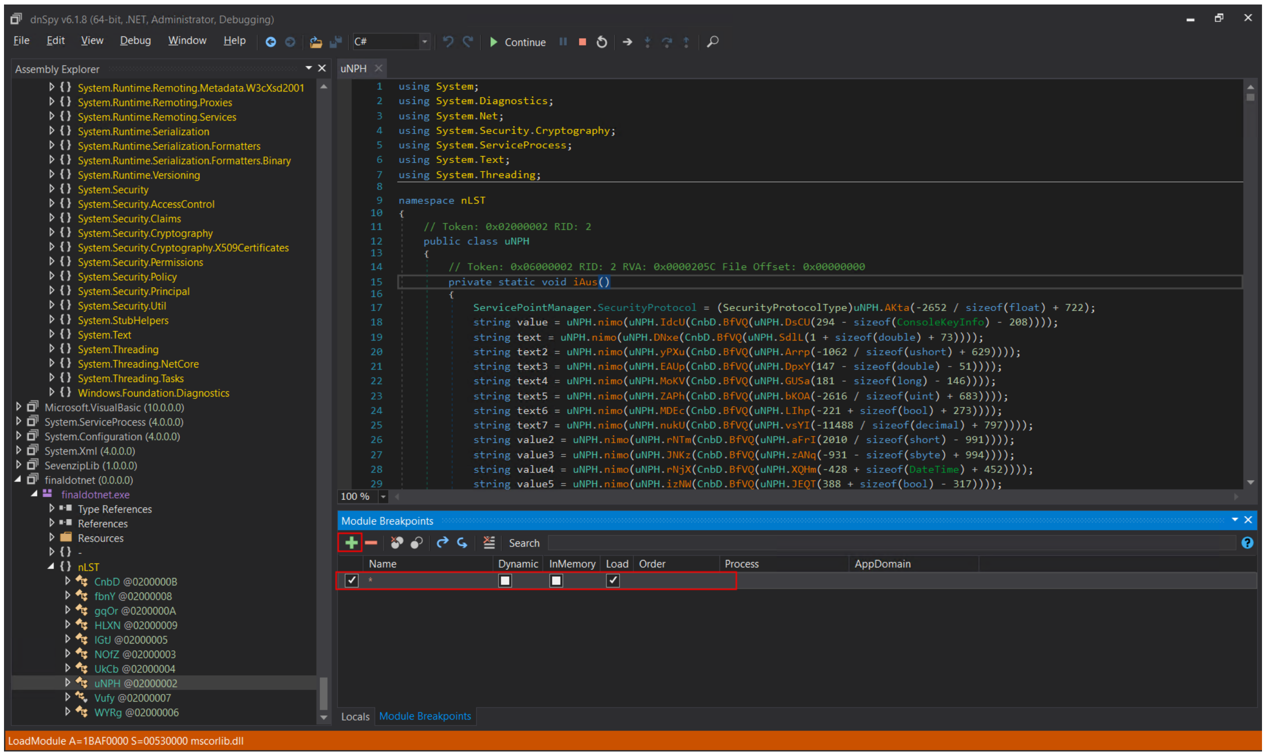
The author mentioned that this challenge is about `.NET` manual unpacking.

> Self assumption: It is packed because there are `Assembly` stuff and under `nLST.KMZdl.FXwnR`, there is `Assembly.Load` and under `nLST.sLcn.dEAB` function there is a variable named `entryPoint`

Solution

Enable breakpoints on all modules:

- `Debug -> Windows -> Module Breakpoints`
- Add breakpoint, name = `*` (for all modules), enable `Dynamic`, `Inmemory`, and `Load`


dnSpy v6.1.8 (64-bit, .NET, Administrator, Debugging)
File Edit View Debug Window Help
C#
Continue
Assembly Explorer
System.Runtime.Remoting.Metadata.W3cXsd2001
System.Runtime.Remoting.Proxies
System.Runtime.Remoting.Services
System.Runtime.Serialization
System.Runtime.Serialization.Formatters
System.Runtime.Serialization.Formatters.Binary
System.Runtime.Versioning
System.Security
System.Security.AccessControl
System.Security.Claims
System.Security.Cryptography
System.Security.Cryptography.X509Certificates
System.Security.Permissions
System.Security.Policy
System.Security.Principal
System.Security.Util
System.StubHelpers
System.Text
System.Threading
System.Threading.NetCore
System.Threading.Tasks
Windows.Foundation.Diagnostics
Microsoft.VisualBasic (10.0.0.0)
System.ServiceProcess (4.0.0.0)
System.Configuration (4.0.0.0)
System.Xml (4.0.0.0)
SevenzipLib (1.0.0.0)
finaldotnet (0.0.0.0)
finaldotnet.exe
Type References
References
Resources
-
nLST
CnbD @0200000B
fbnY @02000008
gqOr @0200000A
HLXN @02000009
IGtJ @02000005
NOfZ @02000003
UkCb @02000004
uNPH @02000002
Vufy @02000007
WYRg @02000006
uNPH
100 %
Module Breakpoints
Search
Locals
Module Breakpoints

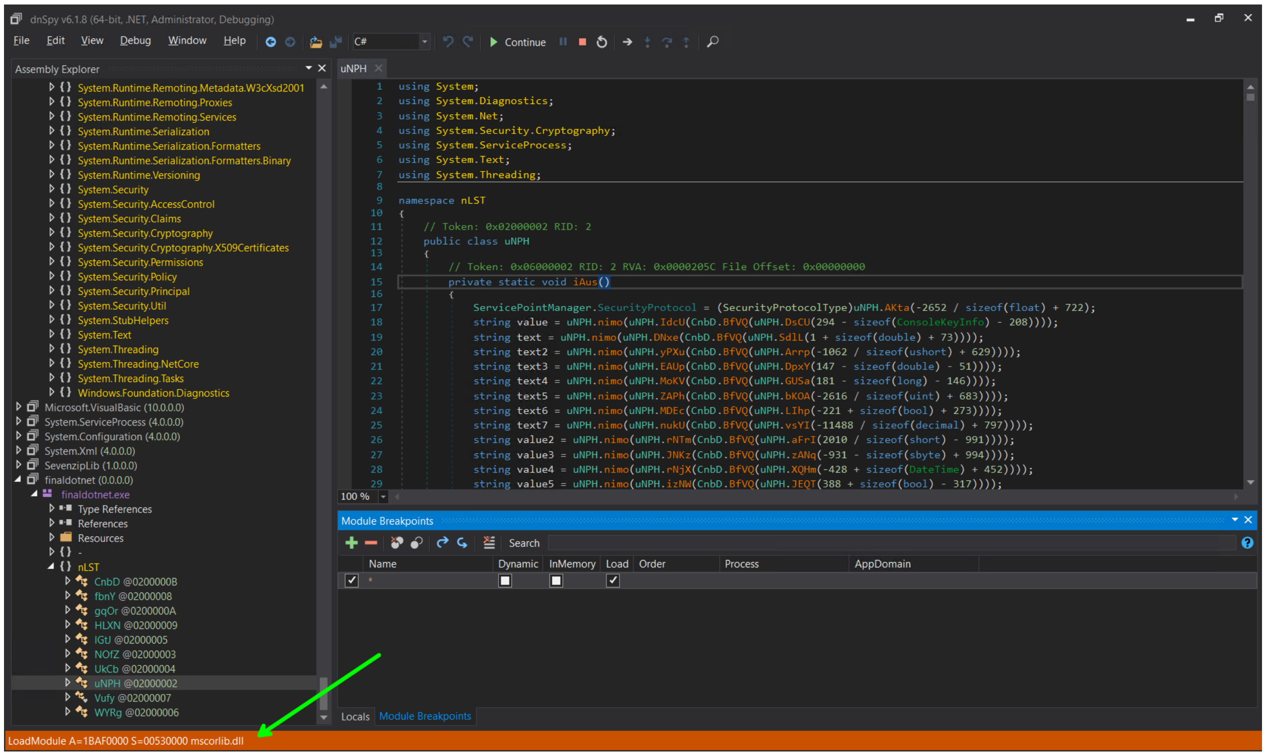
```
using System;
using System.Diagnostics;
using System.Net;
using System.Security.Cryptography;
using System.ServiceProcess;
using System.Text;
using System.Threading;

namespace nLST
{
    // Token: 0x02000002 RID: 2
    public class uNPH
    {
        // Token: 0x06000002 RID: 2 RVA: 0x0000205C File Offset: 0x00000000
        private static void iAus()
        {
            ServicePointManager.SecurityProtocol = (SecurityProtocolType)uNPH.AKta(-2652 / sizeof(float) + 722);
            string value = uNPH.nimo(uNPH.IdcU(CnbD.BfVQ(uNPH.DsCU(294 - sizeof(ConsoleKeyInfo) - 208))));
            string text = uNPH.nimo(uNPH.DNxe(CnbD.BfVQ(uNPH.SdlL(1 + sizeof(double) + 73))));
            string text2 = uNPH.nimo(uNPH.yPXu(CnbD.BfVQ(uNPH.Arrp(-1062 / sizeof(ushort) + 629))));
            string text3 = uNPH.nimo(uNPH.EAUp(CnbD.BfVQ(uNPH.DpxY(147 - sizeof(double) - 51))));
            string text4 = uNPH.nimo(uNPH.MoKV(CnbD.BfVQ(uNPH.GUSa(181 - sizeof(long) - 146))));
            string text5 = uNPH.nimo(uNPH.ZAPh(CnbD.BfVQ(uNPH.bKOA(-2616 / sizeof(uint) + 683))));
            string text6 = uNPH.nimo(uNPH.MDEc(CnbD.BfVQ(uNPH.LIhp(-221 + sizeof(bool) + 273))));
            string text7 = uNPH.nimo(uNPH.nukU(CnbD.BfVQ(uNPH.vsYI(-11488 / sizeof(decimal) + 797))));
            string value2 = uNPH.nimo(uNPH.rNTm(CnbD.BfVQ(uNPH.aFrI(2010 / sizeof(short) - 991))));
            string value3 = uNPH.nimo(uNPH.JNKz(CnbD.BfVQ(uNPH.zANq(-931 - sizeof(sbyte) + 994))));
            string value4 = uNPH.nimo(uNPH.rNjX(CnbD.BfVQ(uNPH.XQHm(-428 + sizeof(DateTime) + 452))));
            string value5 = uNPH.nimo(uNPH.izNW(CnbD.BfVQ(uNPH.JEQT(388 + sizeof(bool) - 317))));
```

	Name	Dynamic	InMemory	Load	Order	Process	AppDomain
✓	*	☐	☐	✓			

```
LoadModule A=1BAF0000 S=00530000 mscorlib.dll
```

Now start debugging the program. Notice on the bottom left hand corner, it logs what modules are loaded. We can ignore system related stuff `dll` like:

- `mscorlib.dll`
- `Microsoft.VisualBasic.dll`
- `System.Core.dll`
- `System.dll`



```
System.Runtime.Remoting.Metadata.W3cXsd2001
System.Runtime.Remoting.Proxies
System.Runtime.Remoting.Services
System.Runtime.Serialization
System.Runtime.Serialization.Formatters
System.Runtime.Serialization.Formatters.Binary
System.Runtime.Versioning
System.Security
System.Security.AccessControl
System.Security.Claims
System.Security.Cryptography
System.Security.Cryptography.X509Certificates
System.Security.Permissions
System.Security.Policy
System.Security.Principal
System.Security.Util
System.StubHelpers
System.Text
System.Threading
System.Threading.NetCore
System.Threading.Tasks
Windows.Foundation.Diagnostics
Microsoft.VisualBasic (10.0.0.0)
System.ServiceProcess (4.0.0.0)
System.Configuration (4.0.0.0)
System.Xml (4.0.0.0)
SevenzipLib (1.0.0.0)
finaldotnet (0.0.0.0)
  finaldotnet.exe
    Type References
    References
    Resources
    -
    nLST
      CnbD @0200000B
      fbnY @02000008
      gqOr @0200000A
      HLXN @02000009
      IGtJ @02000005
      NOfZ @02000003
      UkCb @02000004
      uNPH @02000002
      Vufy @02000007
      WYRg @02000006
```

```
using System;
using System.Diagnostics;
using System.Net;
using System.Security.Cryptography;
using System.ServiceProcess;
using System.Text;
using System.Threading;

namespace nLST
{
    // Token: 0x02000002 RID: 2
    public class uNPH
    {
        // Token: 0x06000002 RID: 2 RVA: 0x0000205C File Offset: 0x00000000
        private static void iAus()
        {
            ServicePointManager.SecurityProtocol = (SecurityProtocolType)uNPH.AKta(-2652 / sizeof(float) + 722);
            string value = uNPH.nimo(uNPH.IdcU(CnbD.BfVQ(uNPH.DsCU(294 - sizeof(ConsoleKeyInfo) - 208))));
            string text = uNPH.nimo(uNPH.DNxe(CnbD.BfVQ(uNPH.SdlL(1 + sizeof(double) + 73))));
            string text2 = uNPH.nimo(uNPH.yPXu(CnbD.BfVQ(uNPH.Arrp(-1062 / sizeof(ushort) + 629))));
            string text3 = uNPH.nimo(uNPH.EAUp(CnbD.BfVQ(uNPH.DpxY(147 - sizeof(double) - 51))));
            string text4 = uNPH.nimo(uNPH.MoKV(CnbD.BfVQ(uNPH.GUSa(181 - sizeof(long) - 146))));
            string text5 = uNPH.nimo(uNPH.ZAPh(CnbD.BfVQ(uNPH.bKOA(-2616 / sizeof(uint) + 683))));
            string text6 = uNPH.nimo(uNPH.MDEc(CnbD.BfVQ(uNPH.LIhp(-221 + sizeof(bool) + 273))));
            string text7 = uNPH.nimo(uNPH.nukU(CnbD.BfVQ(uNPH.vsYI(-11488 / sizeof(decimal) + 797))));
            string value2 = uNPH.nimo(uNPH.rNTm(CnbD.BfVQ(uNPH.aFrI(2010 / sizeof(short) - 991))));
            string value3 = uNPH.nimo(uNPH.JNKz(CnbD.BfVQ(uNPH.zANq(-931 - sizeof(sbyte) + 994))));
            string value4 = uNPH.nimo(uNPH.rNjX(CnbD.BfVQ(uNPH.XQHm(-428 + sizeof(DateTime) + 452))));
            string value5 = uNPH.nimo(uNPH.izNW(CnbD.BfVQ(uNPH.JEQT(388 + sizeof(bool) - 317))));
```

	Name	Dynamic	InMemory	Load	Order	Process	AppDomain
✓	*	☐	☐	✓			

```
LoadModule A=1BAF0000 S=00530000 mscorlib.dll
```

Aside from the `dll` stuffs, initially, there are `finaldotnet.exe` and `SevenzipLib` modules being loaded. The first one is whatever as it is the executable itself (I suppose) and the second one is kind of interesting (this may be another sign of the executable being packed). However, we will just continue to observe what other modules are being loaded.

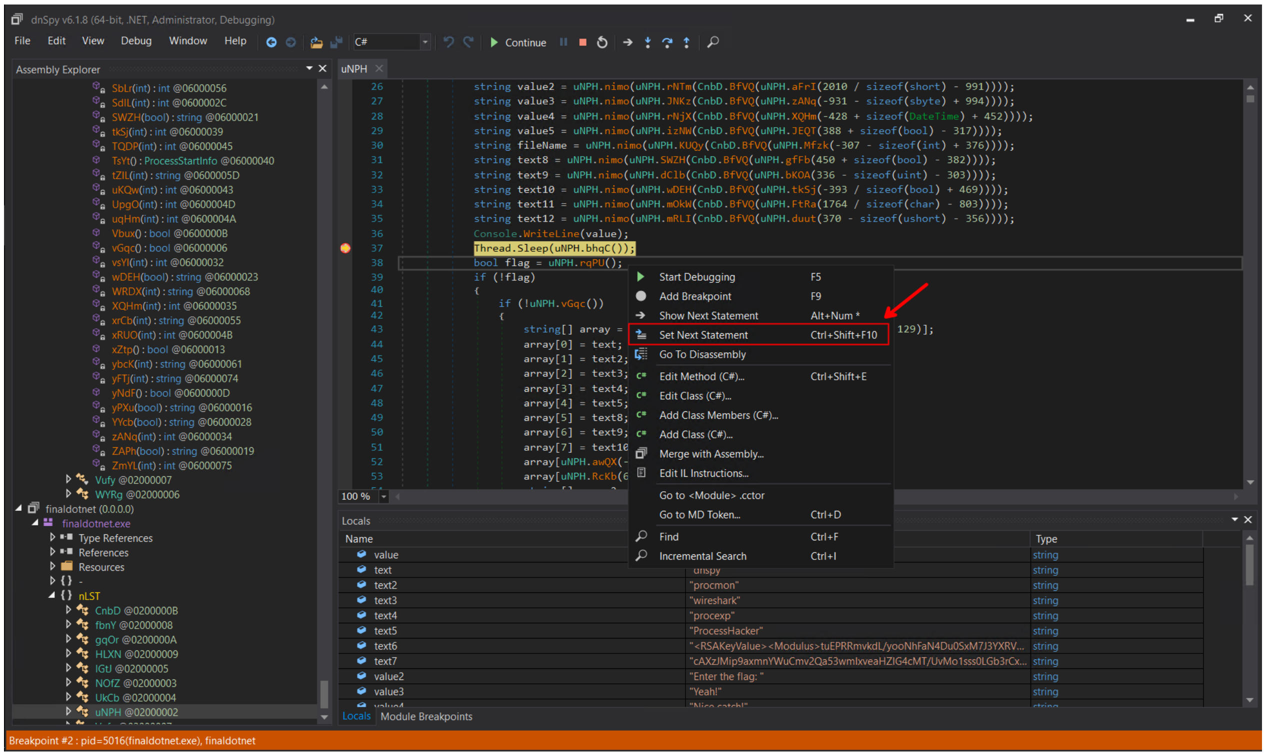
Immediately after `SevenzipLib` is loaded and we `continue` the execution, a module called `finaldotnet` is loaded (note the missing `.exe`). This is the unpacked executable. Next, we `step out` and notice that we are now at `nLST.KMZdl.FXwnR` function, specifically this line of code:

```
result = Assembly.Load(sLcn.Ptpi(array));
```

Then, we `step out` again to reach the `nLST.sLcn.dEAB` function. Inside this function, we could see that there is a variable called `entryPoint`, which is most probably belongs to the unpacked one. On line 99, we can see `entryPoint.Invoke(null, parameters)` and set a breakpoint here. This is probably where we start executing the unpacked version. After we `continue` and stop at the previous set breakpoint, we just do `step over` until there is a dialog box which tries to load something and we finally reach `nLST.uNPH.iAus` which is the `main` function of the unpacked executable.

There are lots of variable initializations and then `Thread.Sleep` is called. To bypass the

`Thread.Sleep` function, we could just right click on the next line of code and select `Set Next Statement` .



```
SbLr(int) : int @06000056
SdIL(int) : int @0600002C
SWZH(bool) : string @06000021
tkSj(int) : int @06000039
TQDP(int) : int @06000045
TsYt() : ProcessStartInfo @06000040
tZIL(int) : string @0600005D
uKQw(int) : int @06000043
UpgO(int) : int @0600004D
uqHm(int) : int @0600004A
Vbux() : bool @0600000B
vGqc() : bool @06000006
vsYI(int) : int @06000032
wDEH(bool) : string @06000023
WRDX(int) : string @06000068
XQHm(int) : int @06000035
xrCb(int) : string @06000055
xRUO(int) : int @0600004B
xZtp() : bool @06000013
ybcK(int) : string @06000061
yFTj(int) : string @06000074
yNdF() : bool @0600000D
yPXu(bool) : string @06000016
YYcb(bool) : string @06000028
zANq(int) : int @06000034
ZAPh(bool) : string @06000019
ZmYL(int) : int @06000075
Vufy @02000007
WYRg @02000006
finaldotnet (0.0.0.0)
  finaldotnet.exe
    Type References
    References
    Resources
    { } -
    { } nLST
      CnbD @0200000B
      fbnY @02000008
      gqOr @0200000A
      HLXN @02000009
      IGtJ @02000005
      NOfZ @02000003
      UkCb @02000004
      uNPH @02000002
```

```
string value2 = uNPH.nimo(uNPH.rNTm(CnbD.BfVQ(uNPH.aFrI(2010 / sizeof(short) - 991))));
string value3 = uNPH.nimo(uNPH.JNKz(CnbD.BfVQ(uNPH.zANq(-931 - sizeof(sbyte) + 994))));
string value4 = uNPH.nimo(uNPH.rNjX(CnbD.BfVQ(uNPH.XQHm(-428 + sizeof(DateTime) + 452))));
string value5 = uNPH.nimo(uNPH.izNW(CnbD.BfVQ(uNPH.JEQT(388 + sizeof(bool) - 317))));
string fileName = uNPH.nimo(uNPH.KUQy(CnbD.BfVQ(uNPH.Mfzk(-307 - sizeof(int) + 376))));
string text8 = uNPH.nimo(uNPH.SWZH(CnbD.BfVQ(uNPH.gfFb(450 + sizeof(bool) - 382))));
string text9 = uNPH.nimo(uNPH.dClb(CnbD.BfVQ(uNPH.bKOA(336 - sizeof(uint) - 303))));
string text10 = uNPH.nimo(uNPH.wDEH(CnbD.BfVQ(uNPH.tkSj(-393 / sizeof(bool) + 469))));
string text11 = uNPH.nimo(uNPH.mOkW(CnbD.BfVQ(uNPH.FtRa(1764 / sizeof(char) - 803))));
string text12 = uNPH.nimo(uNPH.mRLI(CnbD.BfVQ(uNPH.duut(370 - sizeof(ushort) - 356))));
Console.WriteLine(value);
Thread.Sleep(uNPH.bhqC());
bool flag = uNPH.rqPU();
if (!flag)
{
    if (!uNPH.vGqc())
    {
        string[] array =                                  129)];
        array[0] = text;
        array[1] = text2;
        array[2] = text3;
        array[3] = text4;
        array[4] = text5;
        array[5] = text8;
        array[6] = text9;
        array[7] = text10
        array[uNPH.awQX(-
        array[uNPH.RcKb(6
```

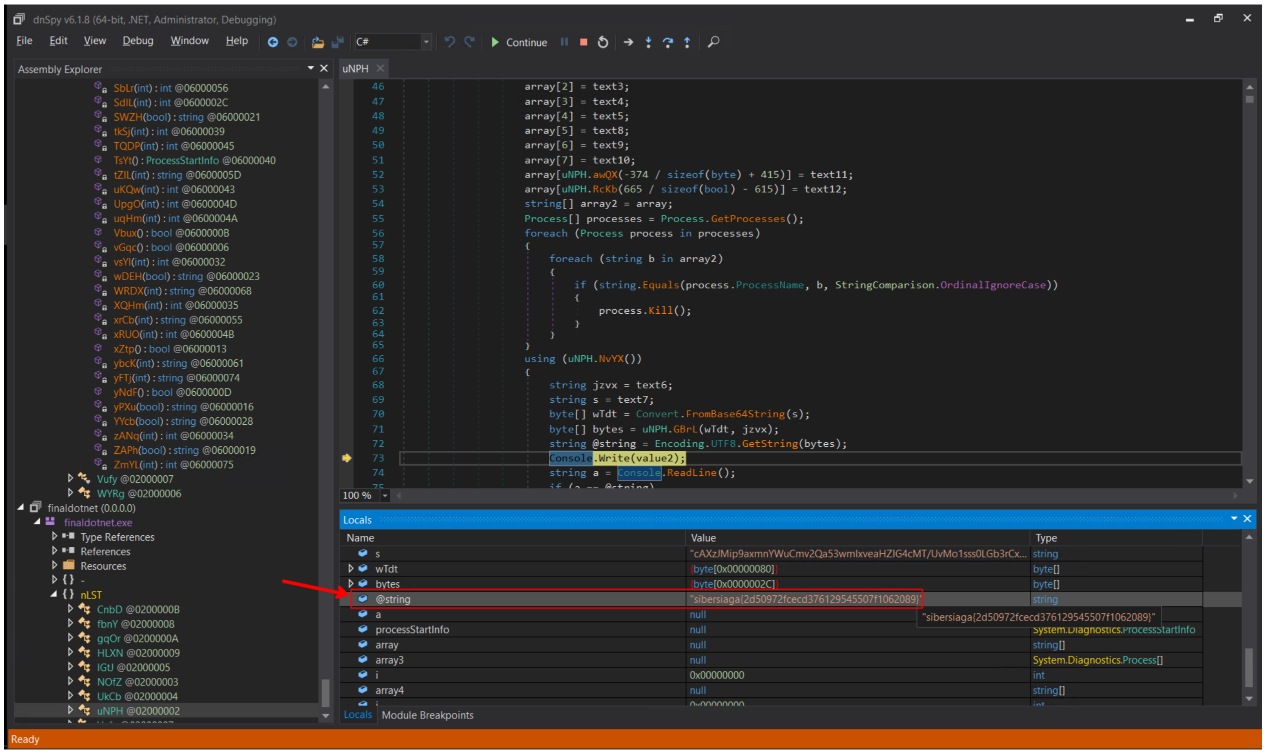
Name		Type
value		string
text	"dnspy"	string
text2	"procmon"	string
text3	"wireshark"	string
text4	"procexp"	string
text5	"ProcessHacker"	string
text6	"<RSAKeyValue><Modulus>tuEPRRmvkdL/yooNhFaN4Du0SxM7J3YXRV...	string
text7	"cAXzJMip9axmnYWuCmv2Qa53wmlxveaHZlG4cMT/UvMo1sss0LGb3rCx...	string
value2	"Enter the flag: "	string
value3	"Yeah!"	string
value4	"Nice catch!"	string

```
Breakpoint #2 : pid=5016(finaldotnet.exe), finaldotnet
```

There are lots of check (this could be skipped easily using `Set Next Statement`) and what we ultimately want to reach is this code block

```
using (uNPH.NvYX())
{
    string jzvx = text6;
    string s = text7;
    byte[] wTdt = Convert.FromBase64String(s);
    byte[] bytes = uNPH.GBrL(wTdt, jzvx);
    string @string = Encoding.UTF8.GetString(bytes);
    Console.Write(value2);
    string a = Console.ReadLine();
    if (a == @string)
    {
        Console.WriteLine(value3);
        Console.WriteLine(value4);
    }
    else
    {
        Console.WriteLine(value5);
    }
}
```

This is because we could see that the variable `value2` is `Enter the flag: ` and there is also `Console.Readline()`. Then our input is being compared with `@string` and if we take a look at `value3` and `value4`, both signifiy that our input is correct. If we inspect the value of `@string`, we could see our flag there.



dnSpy v6.1.8 (64-bit, .NET, Administrator, Debugging)
File Edit View Debug Window Help
C#
Continue
Assembly Explorer
uNPH
100 %
Locals
Module Breakpoints
Ready

```
array[2] = text3;
array[3] = text4;
array[4] = text5;
array[5] = text8;
array[6] = text9;
array[7] = text10;
array[uNPH.awQX(-374 / sizeof(byte) + 415)] = text11;
array[uNPH.RcKb(665 / sizeof(bool) - 615)] = text12;
string[] array2 = array;
Process[] processes = Process.GetProcesses();
foreach (Process process in processes)
{
    foreach (string b in array2)
    {
        if (string.Equals(process.ProcessName, b, StringComparison.OrdinalIgnoreCase))
        {
            process.Kill();
        }
    }
}
using (uNPH.NvYX())
{
    string jzvx = text6;
    string s = text7;
    byte[] wTdt = Convert.FromBase64String(s);
    byte[] bytes = uNPH.GBrL(wTdt, jzvx);
    string @string = Encoding.UTF8.GetString(bytes);
    Console.Write(value2);
    string a = Console.ReadLine();
    if (a == @string)
```

Name	Value	Type
s	"cAXzJMip9axmnYWuCmv2Qa53wmlxveaHZIG4cMT/UvMo1sss0LGb3rCx...	string
wTdt	{byte[0x00000080]}	byte[]
bytes	{byte[0x0000002C]}	byte[]
@string	"sibersiaga{2d50972fcecd376129545507f1062089}"	string
a	null	
processStartInfo	null	System.Diagnostics.ProcessStartInfo
array	null	string[]
array3	null	System.Diagnostics.Process[]
i	0x00000000	int
array4	null	string[]

"sibersiaga{2d50972fcecd376129545507f1062089}"

Flag: `sibersiaga{2d50972fcecd376129545507f1062089}`