# Thick Client Pentest Lab Setup: DVTA (Part 2)

January 13, 2021 By Raj Chandel

In the previous article, we have discussed the Lab setup of Thick Client: DVTA

You can simply take a walkthrough by visiting here: – **Thick Client Pentest Lab Setup: DVTA**

In this article, we are going to discuss how can we configure the DVTA application to connect to our server For this, I'm going to use one single window 10 instances for the entire setup. That means both the DVTA client as well as the SQL server are going to run on the same machine. You can configure the DVTA client and SQL server on Different Window 10 instances.

## Table of Content

## Prerequisites

- Good knowledge of C# (C Sharp) programming knowledge.
- A bit of Reverse Engineering knowledge.

## Download Links & Credentials

DVTA application: –
**https://mega.nz/file/rPJQ0LTL#CuhQ9HBeEIKHa67jbITFQbt49FZalugWlRe5A33O6y4**

DnSpy application: – **https://github.com/dnSpy/dnSpy/releases**

Database users and password: –

**Credentials:**

```
vijay:vijay
raj:raj
admin:admin123
```

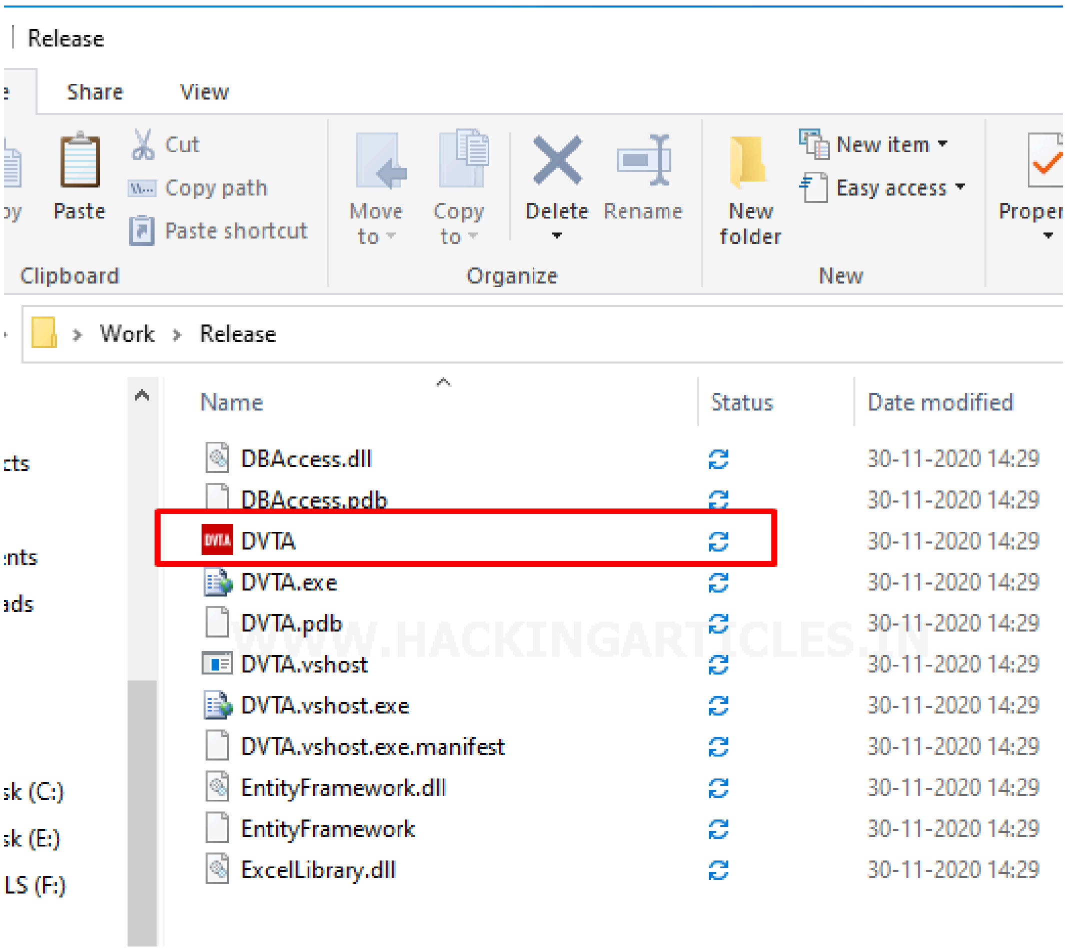
## Download and install the DVTA application

You can directly download the DVTA application by using this link

```
https://mega.nz/file/rPJQ0LTL#CuhQ9HBeEIKHa67jbITFQbt49FZalugWlRe5A33O6y4
```

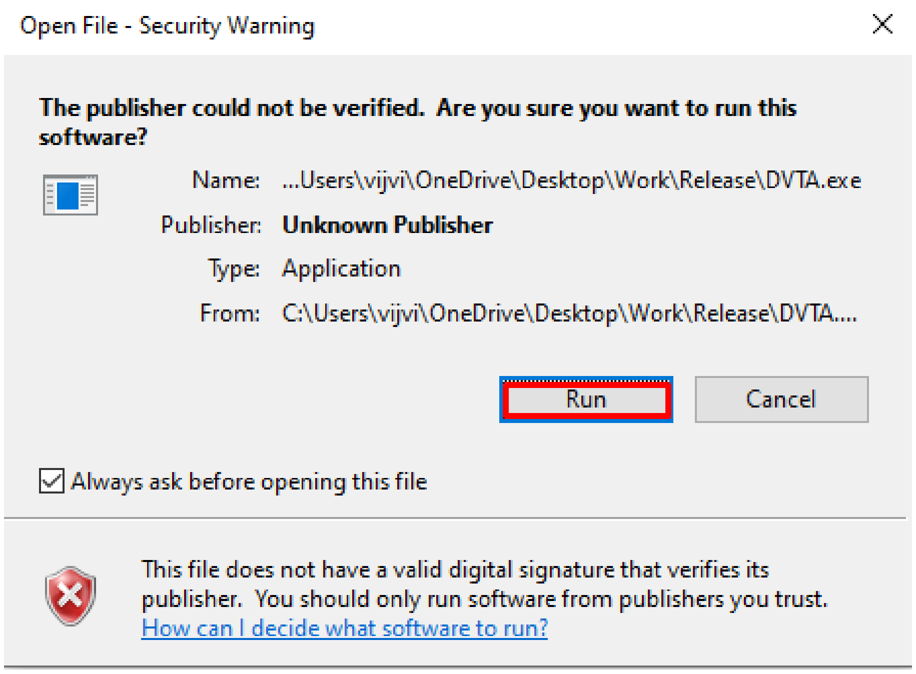
when the download completes extract it into a new folder which similarly looks like this as shown in the image below.

Alright, this is the Release File that we have gotten for the testing. If you see there are a couple of file inside of the Release folder and the main application here is the one with the Red icon named **"DVTA.exe"**



| Name | Status | Date modified |
|---|---|---|
| DBAccess.dll | | 30-11-2020 14:29 |
| DBAccess.pdb | | 30-11-2020 14:29 |
| DVTA | | 30-11-2020 14:29 |
| DVTA.exe | | 30-11-2020 14:29 |
| DVTA.pdb | | 30-11-2020 14:29 |
| DVTA.vshost | | 30-11-2020 14:29 |
| DVTA.vshost.exe | | 30-11-2020 14:29 |
| DVTA.vshost.exe.manifest | | 30-11-2020 14:29 |
| EntityFramework.dll | | 30-11-2020 14:29 |
| EntityFramework | | 30-11-2020 14:29 |
| ExcelLibrary.dll | | 30-11-2020 14:29 |

So, let's give a double click to the DVTA.exe and run it.


Open File - Security Warning

**The publisher could not be verified. Are you sure you want to run this software?**

Name: ...Users\vijvi\OneDrive\Desktop\Work\Release\DVTA.exe

Publisher: **Unknown Publisher**

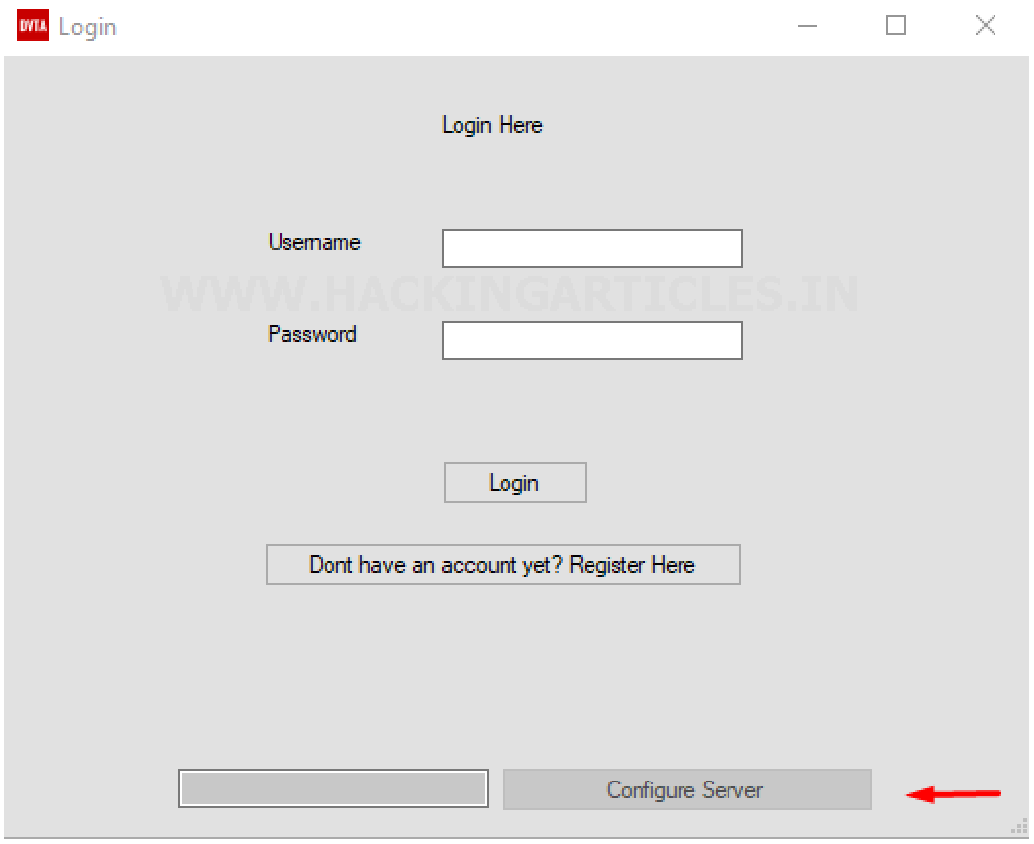
Type: Application

From: C:\Users\vijvi\OneDrive\Desktop\Work\Release\DVTA....

☑ Always ask before opening this file

This file does not have a valid digital signature that verifies its publisher. You should only run software from publishers you trust.
How can I decide what software to run?

Once that done you should see a prompt window something like this. If you notice the bottom part of the prompt window, we can configure the server here.



The original DVTA application doesn't have this feature and the IP address is hardcoded. The problem with that is we need to have a server with the exact same IP that is hardcoded in the application. For this, all we need to compile the source code to modify the application somehow so that it points or redirects to our server.

Now, our main Task before configuring the server is to enable the **“configure Server”** button so that it can talk to our server.

*This requires a bit of Reversing knowledge…. And this going to be fun*.

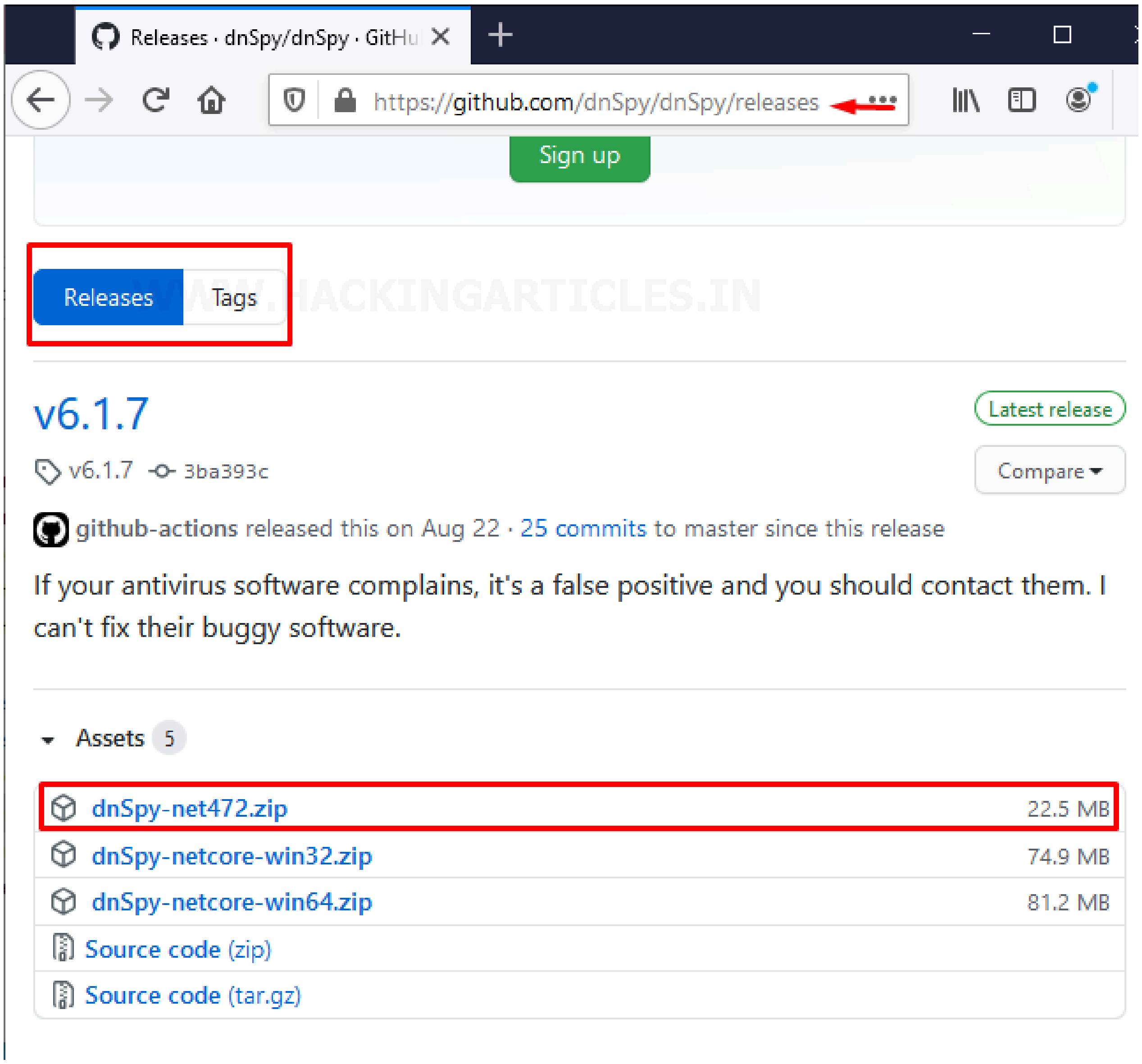
## Reversing and Modifying the original DVTA application

Now let’s see how we can enable the **“configure server button”**.

*Before we enable this button let’s think a while what are the situations where we came across this kind of disabled buttons in the client applications in the real world when we do pentesting. There are cases where a user logs In and some menus are disabled for him. For instance, he is the maker and he can’t have checker menus so some developers disable them in the client assuming that the user cannot enable them and these disabled menus can be easily enabled in C sharp.net applications.*

Now let’s see how we can do that. First of all, simply take a copy of the Extracted DVTA Release folder for the case if we did something wrong or messed up with codes so simply, we can start from fresh without downloading it again. To enable the “configure server” button we’re going to use a tool called dnSpy. You can download it directly by following this link: – **https://github.com/dnSpy/dnSpy/releases**

Download the latest version from the release section.



github-actions released this on Aug 22 · 25 commits to master since this release

If your antivirus software complains, it's a false positive and you should contact them. I can't fix their buggy software.

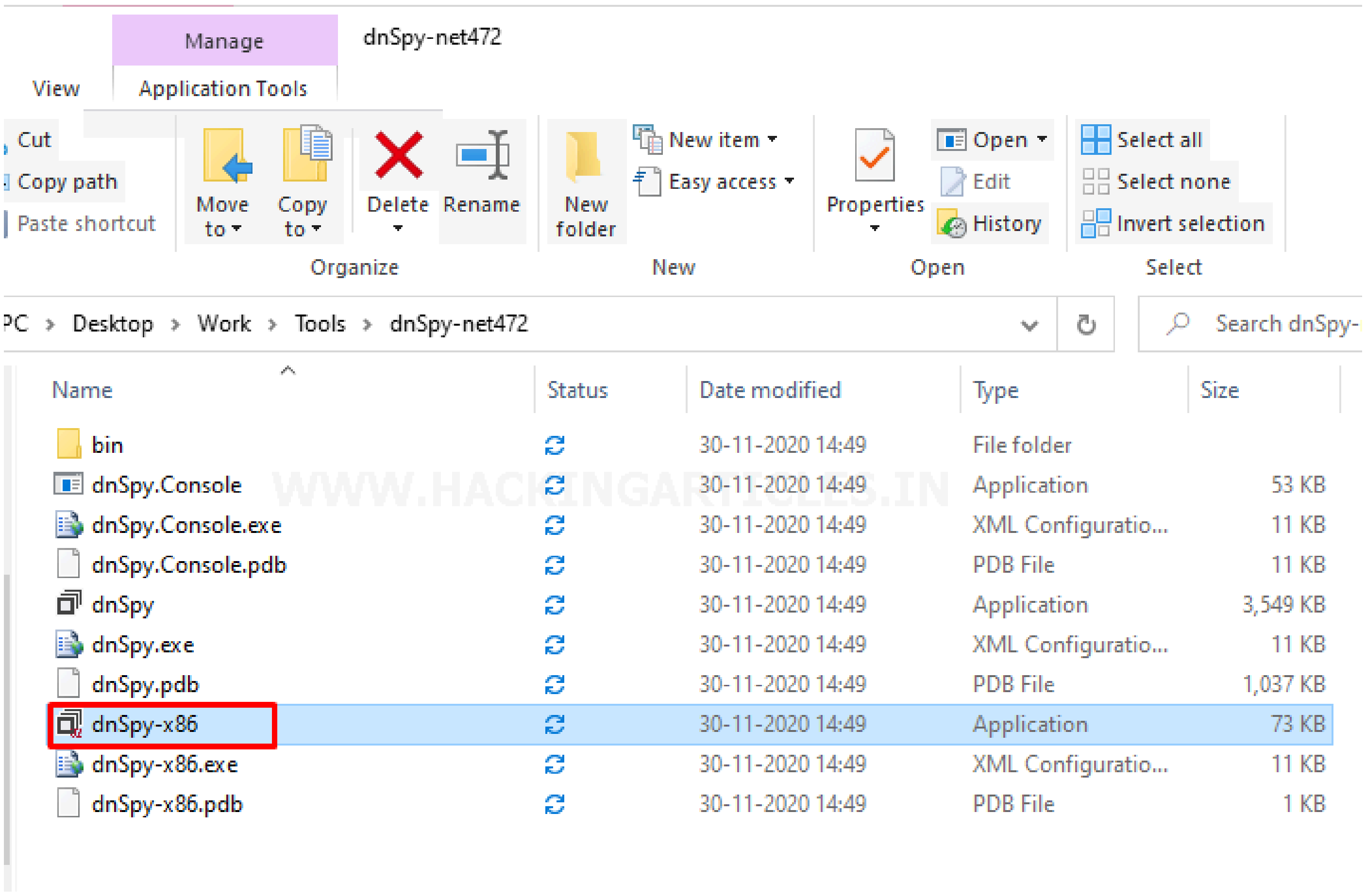
Assets 5

| | |
|---|---|
| dnSpy-net472.zip | 22.5 MB |
| dnSpy-netcore-win32.zip | 74.9 MB |
| dnSpy-netcore-win64.zip | 81.2 MB |
| Source code (zip) | |
| Source code (tar.gz) | |

When the download is complete extract it into a new folder.

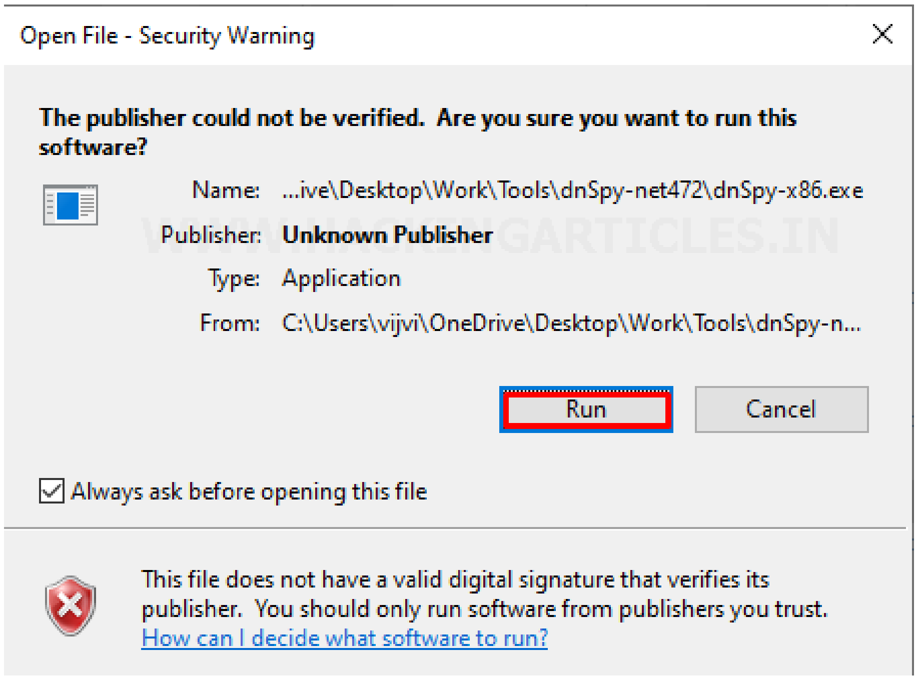
Once the extraction is complete you should see two versions of dnSpy one is 32-bit and the other one 64-bit. We are going to use the 32-bit version because our target application is 32-bit.

So, if you see in the below image the one which is having dnSpy-x86 is the 32-bit version and the other one that just says the dnSpy is the 64-bit version.



| Name | Status | Date modified | Type | Size |
|---|---|---|---|---|
| bin | | 30-11-2020 14:49 | File folder | |
| dnSpy.Console | | 30-11-2020 14:49 | Application | 53 KB |
| dnSpy.Console.exe | | 30-11-2020 14:49 | XML Configuratio... | 11 KB |
| dnSpy.Console.pdb | | 30-11-2020 14:49 | PDB File | 11 KB |
| dnSpy | | 30-11-2020 14:49 | Application | 3,549 KB |
| dnSpy.exe | | 30-11-2020 14:49 | XML Configuratio... | 11 KB |
| dnSpy.pdb | | 30-11-2020 14:49 | PDB File | 1,037 KB |
| dnSpy-x86 | | 30-11-2020 14:49 | Application | 73 KB |
| dnSpy-x86.exe | | 30-11-2020 14:49 | XML Configuratio... | 11 KB |
| dnSpy-x86.pdb | | 30-11-2020 14:49 | PDB File | 1 KB |

So, without wasting time Quickly open up **“dnSpy-x86”** by giving it a double click and simply run it


Open File - Security Warning

**The publisher could not be verified. Are you sure you want to run this software?**

Name: ...ive\Desktop\Work\Tools\dnSpy-net472\dnSpy-x86.exe

Publisher: **Unknown Publisher**

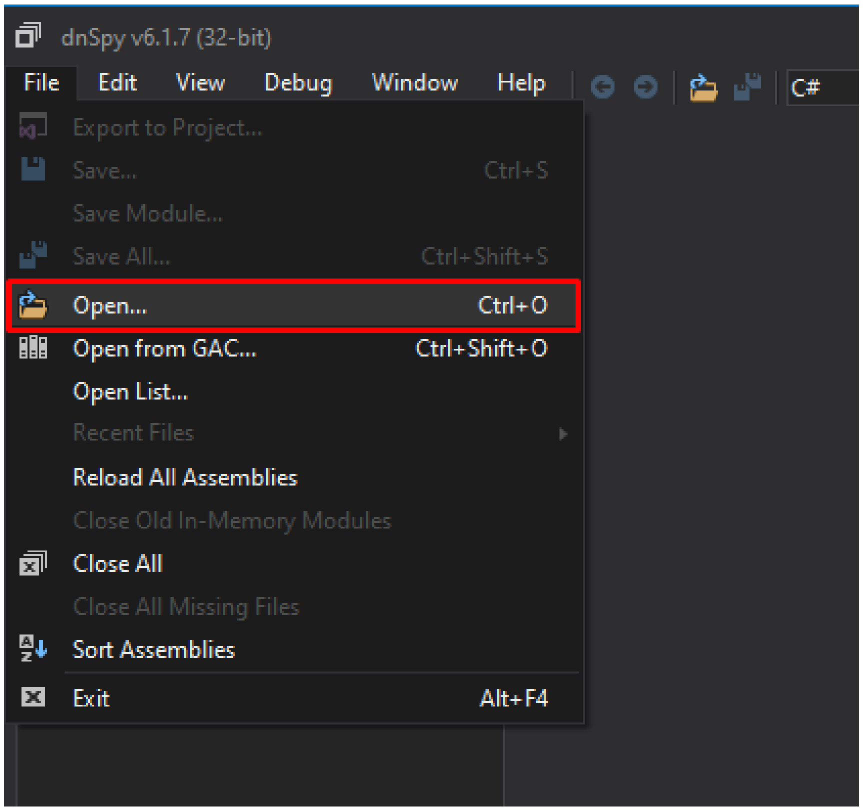
Type: Application

From: C:\Users\vijvi\OneDrive\Desktop\Work\Tools\dnSpy-n...

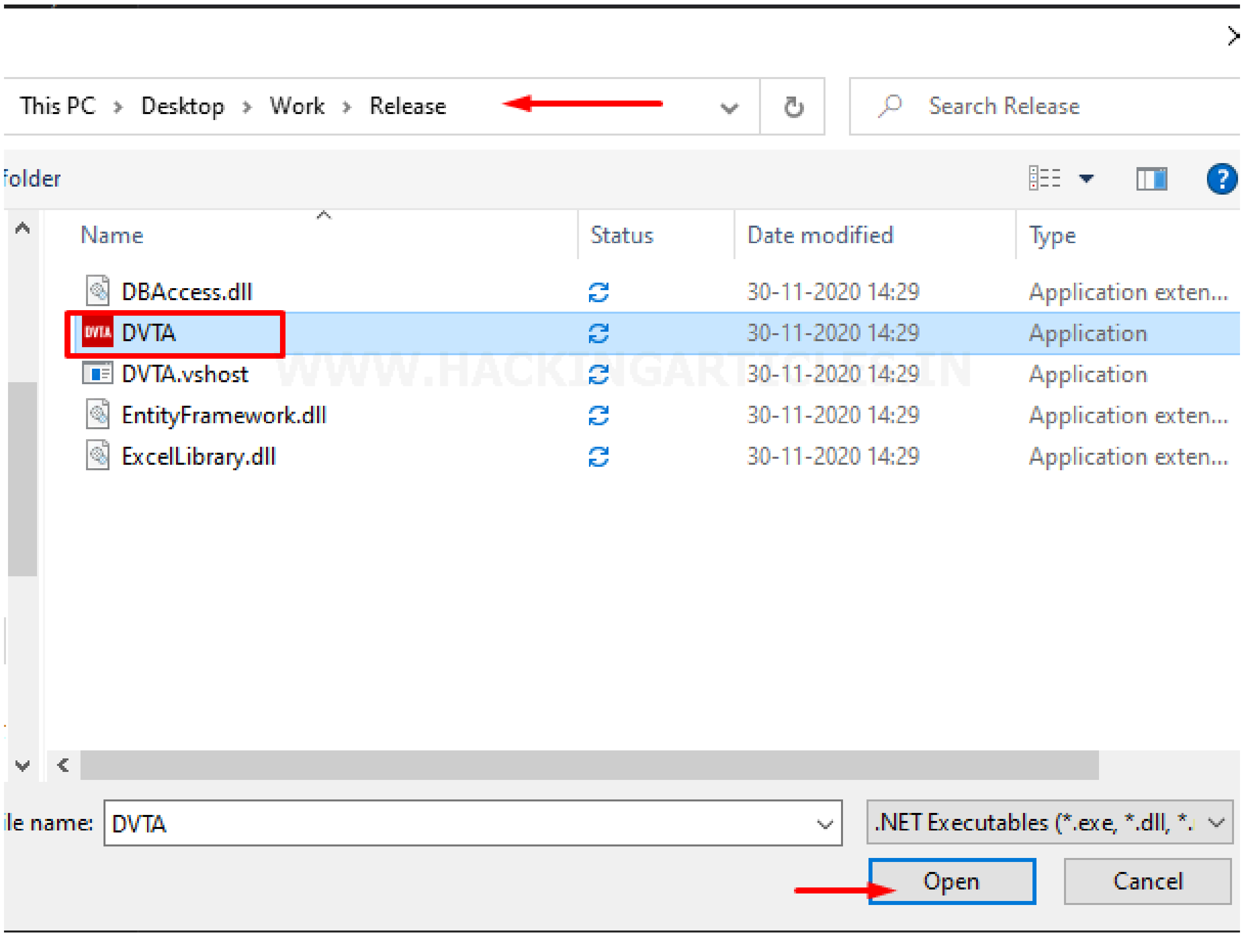
☑ Always ask before opening this file

This file does not have a valid digital signature that verifies its publisher. You should only run software from publishers you trust.
How can I decide what software to run?

When the dnSpy application started open up the DVTA application by navigating to **File > open**



And go to the Release folder of the DVTA application and choose DVTA.exe. So, this is the main application that we're going to modify.



| Name | Status | Date modified | Type |
|---|---|---|---|
| DBAccess.dll | | 30-11-2020 14:29 | Application exten... |
| DVTA | | 30-11-2020 14:29 | Application |
| DVTA.vshost | | 30-11-2020 14:29 | Application |
| EntityFramework.dll | | 30-11-2020 14:29 | Application exten... |
| ExcelLibrary.dll | | 30-11-2020 14:29 | Application exten... |

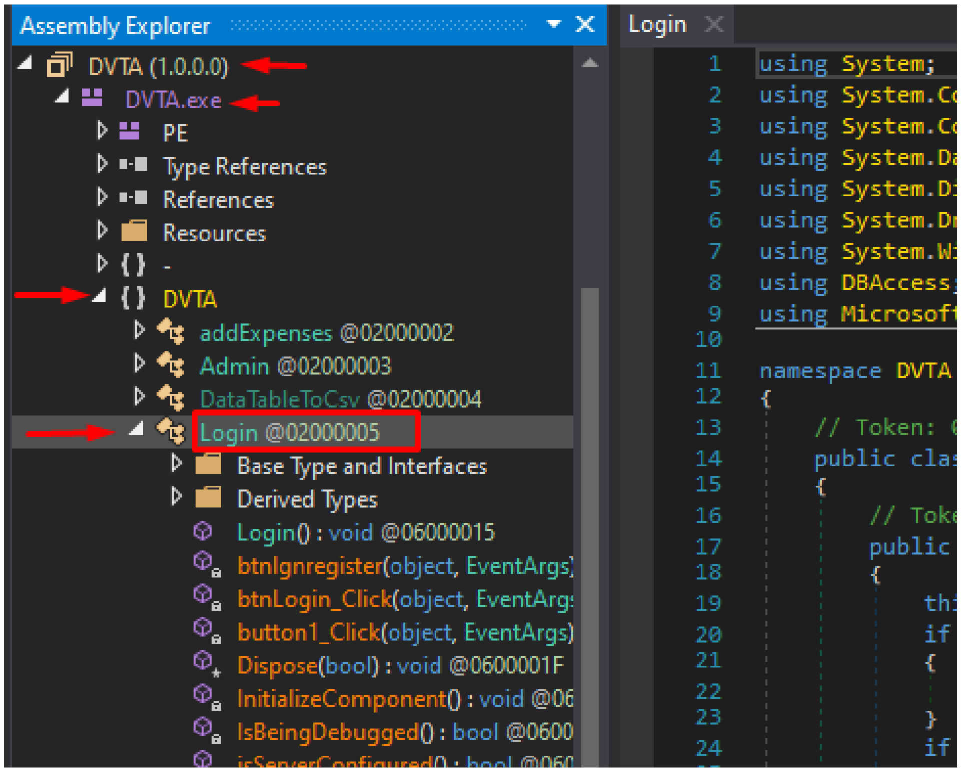
ile name: DVTA

.NET Executables (*.exe, *.dll, *.

Now, our main task is to find out the exact location of the button which is disabled and then patch the binary in a way that it's going to be enabled

On the left side menu, click on the DVTA and expand the DVTA.exe. Once it expanded select DVTA. Now, If you see, there seems to be a couple of here within the DVTA application. Basically, these are C sharp files in the source code. If you notice above when you open the DVTA application, it open's up the **login page of DVTA** and on that page the bottom part has the **"configure server"** button. So, this is what we're interested in.

Now, without wasting time click on Login and it will start automatically decompiling the source code for us.

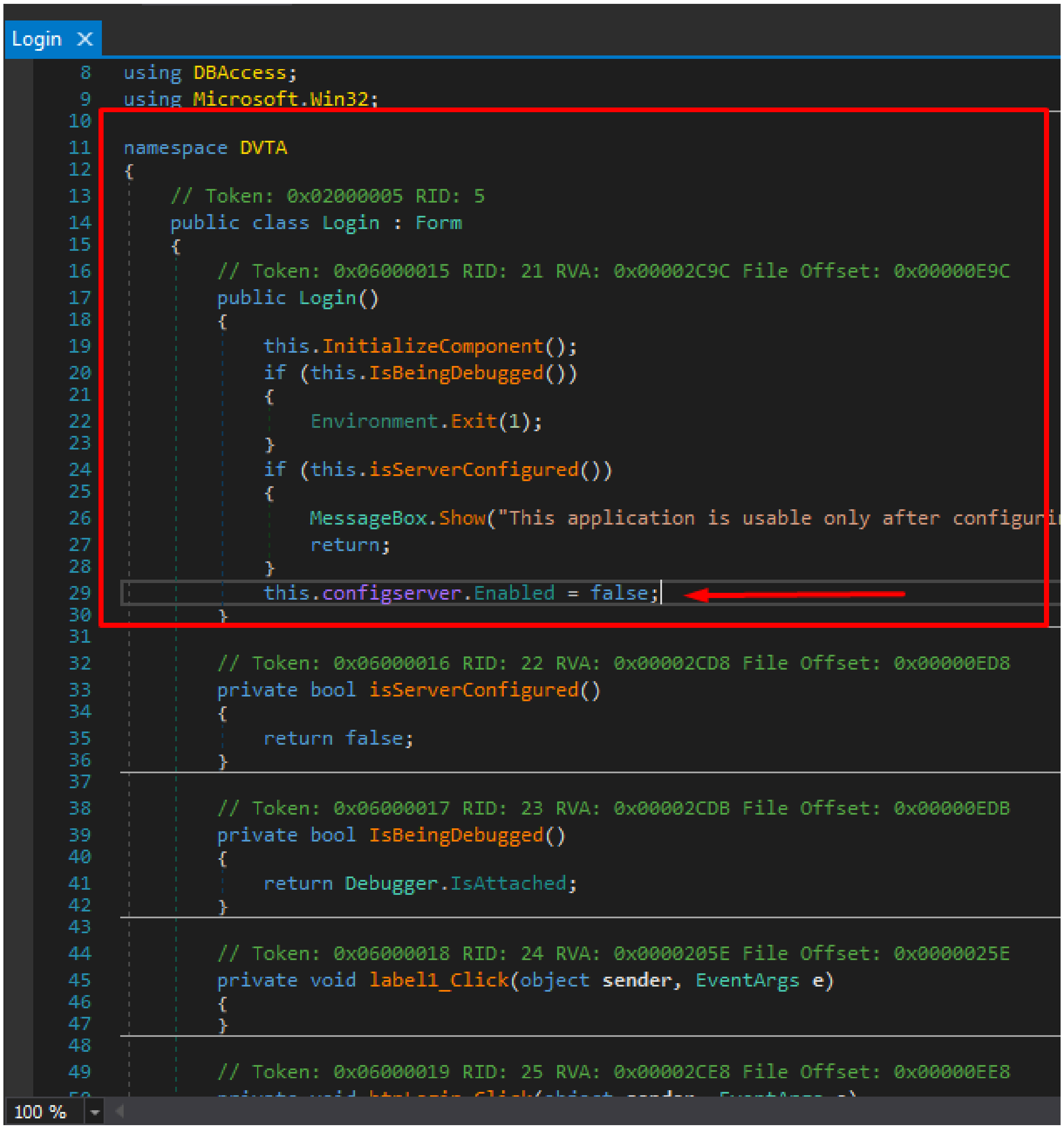


```
using System;
using System.Co
using System.Co
using System.Da
using System.Di
using System.Dr
using System.Wi
using DBAccess;
using Microsoft

namespace DVTA
{
    // Token: 0
    public clas
    {
        // Toke
        public
        {
            thi
            if
            {

            }
            if
```

Now, it seems very interesting and it's pretty easy for us to understand the entire source code of what this application is doing here. When you scroll down to the source code you can find there is a condition that is looking for a method called **"is a server configured"** when you click on this it will see where the definition is and when you see the definition it is always returning **"false"** and what happens if its return the value True.

Let's see…

*It is going to show the particular message **"MessageBox.Show("This application is usable only after configuring the server")*** That means this block is never going to execute because it is always **"returning false"**. So what that means is "**this.configserver.Enabled = False;**" is the part which is going to get executed always as highlighted by the right arrow  In the below image

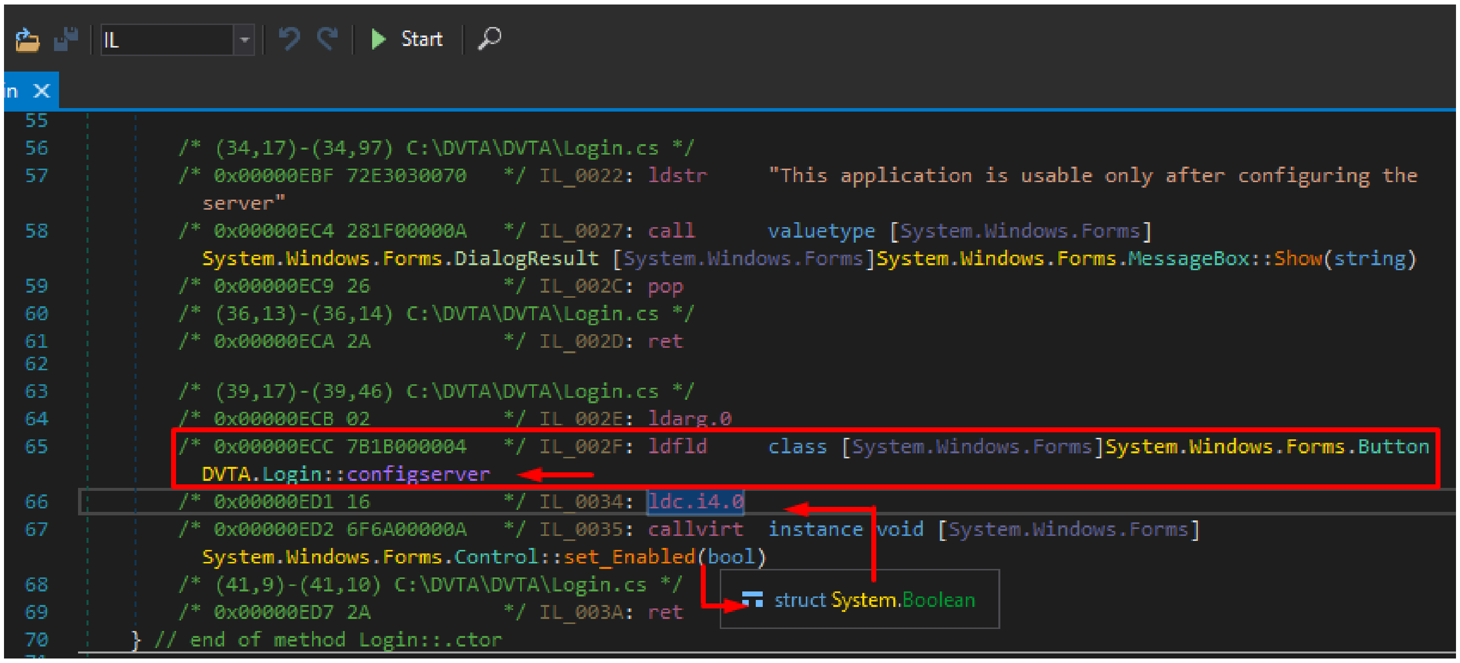


```
using DBAccess;
using Microsoft.Win32;

namespace DVTA
{
    // Token: 0x02000005 RID: 5
    public class Login : Form
    {
        // Token: 0x06000015 RID: 21 RVA: 0x00002C9C File Offset: 0x00000E9C
        public Login()
        {
            this.InitializeComponent();
            if (this.IsBeingDebugged())
            {
                Environment.Exit(1);
            }
            if (this.isServerConfigured())
            {
                MessageBox.Show("This application is usable only after configuri
                return;
            }
            this.configserver.Enabled = false;
        }

        // Token: 0x06000016 RID: 22 RVA: 0x00002CD8 File Offset: 0x00000ED8
        private bool isServerConfigured()
        {
            return false;
        }

        // Token: 0x06000017 RID: 23 RVA: 0x00002CDB File Offset: 0x00000EDB
        private bool IsBeingDebugged()
        {
            return Debugger.IsAttached;
        }

        // Token: 0x06000018 RID: 24 RVA: 0x0000205E File Offset: 0x0000025E
        private void label1_Click(object sender, EventArgs e)
        {
        }

        // Token: 0x06000019 RID: 25 RVA: 0x00002CE8 File Offset: 0x00000EE8
```

If you noticed that is all the source code is are in C sharp.

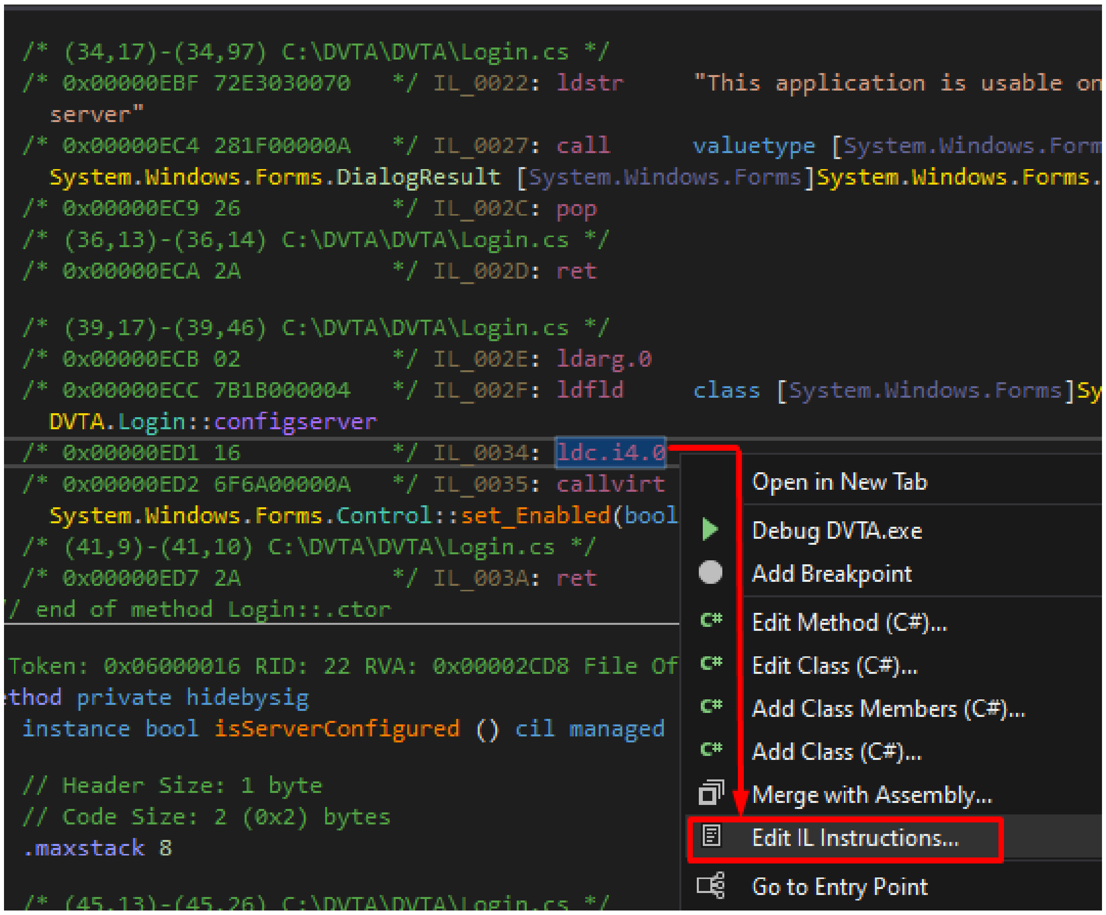
Let’s take a look at the IL version of this source code. IL is an intermediate language. This is similar to the old assembly language in the C programming world. You can convert the source code into the IL by navigating to a drop-down box nearby the start icon. If you compare the C sharp code or IL code basically, they are doing the same thing.

As we have seen earlier “config server” is a button that you can easily find here. It’s a button of **“class [system.windows.Forms]System.Windows.Forms.Button “** after this particular line there seems to be a call to “**System.Windows.Forms.Control**” and set Enabled method is being called and this Set_Enabled method is taking a Boolean value which can be true or false. If you see there seems to be a value which is ldc.i4.0 and then if you hover over your mouse on Set_Enabled(bool) it pushes the integer value of 0 onto the evaluation stack as an int 32 and probably that's the reason why the button is always disabled because this value is being pushed under the stack as an argument and set enable is always set to false.



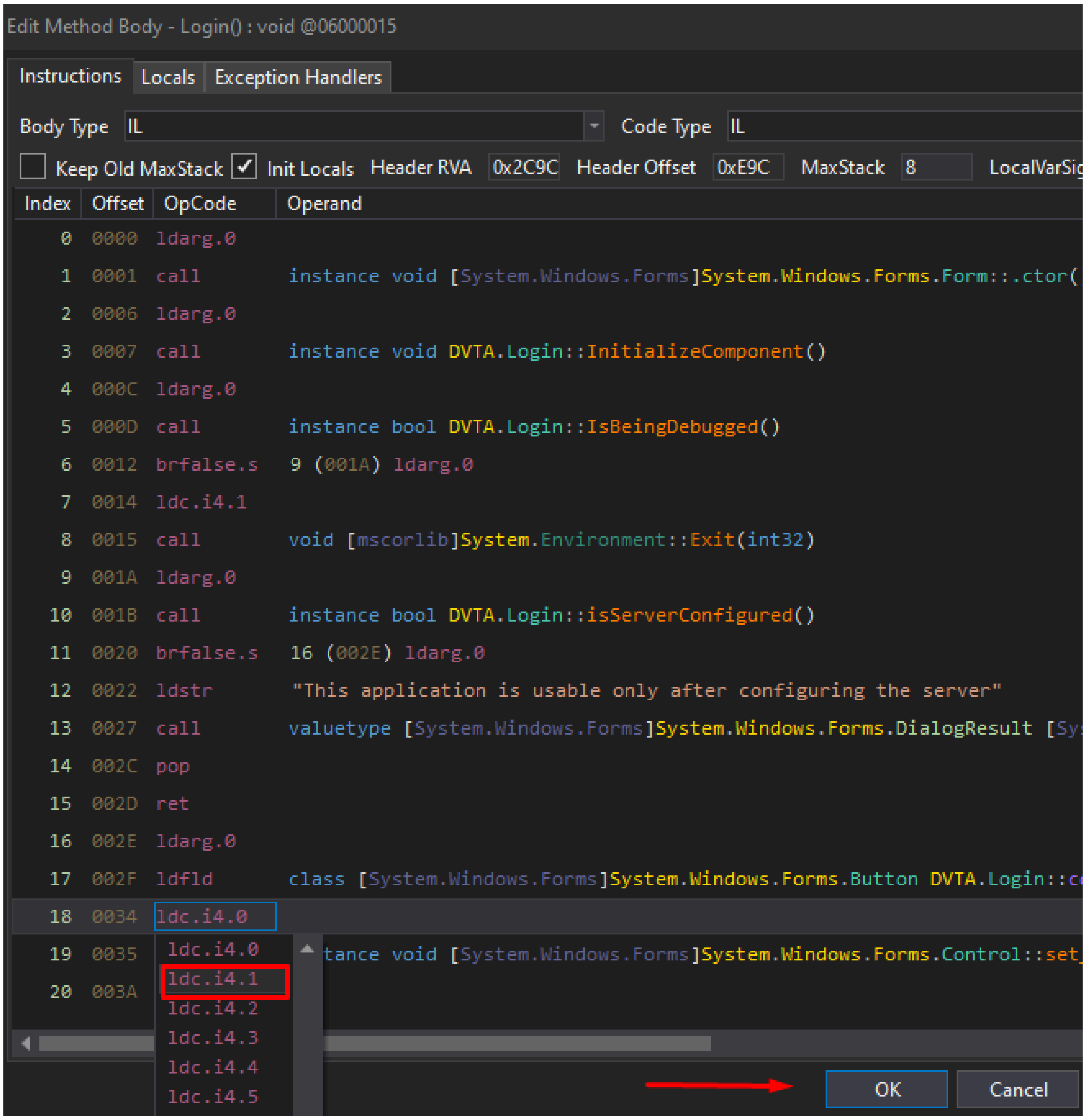
```
        /* (34,17)-(34,97) C:\DVTA\DVTA\Login.cs */
        /* 0x00000EBF 72E3030070   */ IL_0022: ldstr      "This application is usable only after configuring the
          server"
        /* 0x00000EC4 281F00000A   */ IL_0027: call       valuetype [System.Windows.Forms]
          System.Windows.Forms.DialogResult [System.Windows.Forms]System.Windows.Forms.MessageBox::Show(string)
        /* 0x00000EC9 26           */ IL_002C: pop
        /* (36,13)-(36,14) C:\DVTA\DVTA\Login.cs */
        /* 0x00000ECA 2A           */ IL_002D: ret

        /* (39,17)-(39,46) C:\DVTA\DVTA\Login.cs */
        /* 0x00000ECB 02           */ IL_002E: ldarg.0
        /* 0x00000ECC 7B1B000004   */ IL_002F: ldfld      class [System.Windows.Forms]System.Windows.Forms.Button
          DVTA.Login::configserver
        /* 0x00000ED1 16           */ IL_0034: ldc.i4.0
        /* 0x00000ED2 6F6A00000A   */ IL_0035: callvirt   instance void [System.Windows.Forms]
          System.Windows.Forms.Control::set_Enabled(bool)
        /* (41,9)-(41,10) C:\DVTA\DVTA\Login.cs */
        /* 0x00000ED7 2A           */ IL_003A: ret
    } // end of method Login::.ctor
```

Now, give a right-click to the value of ldc.i4.0 and **"Edit the IL instructions"** as shown below



```
/* (34,17)-(34,97) C:\DVTA\DVTA\Login.cs */
/* 0x00000EBF 72E3030070   */ IL_0022: ldstr      "This application is usable on
  server"
/* 0x00000EC4 281F00000A   */ IL_0027: call       valuetype [System.Windows.Form
  System.Windows.Forms.DialogResult [System.Windows.Forms]System.Windows.Forms.
/* 0x00000EC9 26           */ IL_002C: pop
/* (36,13)-(36,14) C:\DVTA\DVTA\Login.cs */
/* 0x00000ECA 2A           */ IL_002D: ret

/* (39,17)-(39,46) C:\DVTA\DVTA\Login.cs */
/* 0x00000ECB 02           */ IL_002E: ldarg.0
/* 0x00000ECC 7B1B000004   */ IL_002F: ldfld      class [System.Windows.Forms]Sy
  DVTA.Login::configserver
/* 0x00000ED1 16           */ IL_0034: ldc.i4.0
/* 0x00000ED2 6F6A00000A   */ IL_0035: callvirt
  System.Windows.Forms.Control::set_Enabled(bool
/* (41,9)-(41,10) C:\DVTA\DVTA\Login.cs */
/* 0x00000ED7 2A           */ IL_003A: ret
/ end of method Login::.ctor

Token: 0x06000016 RID: 22 RVA: 0x00002CD8 File Of
thod private hidebysig
 instance bool isServerConfigured () cil managed

 // Header Size: 1 byte
 // Code Size: 2 (0x2) bytes
 .maxstack 8

 /* (45,13)-(45,26) C:\DVTA\DVTA\Login.cs */
```

Now, it will open up another prompt and give a right-click to the value ldc.14.0 and further then select ldc.i4.1 instead of 0 and hit ok as shown below



Edit Method Body - Login() : void @06000015

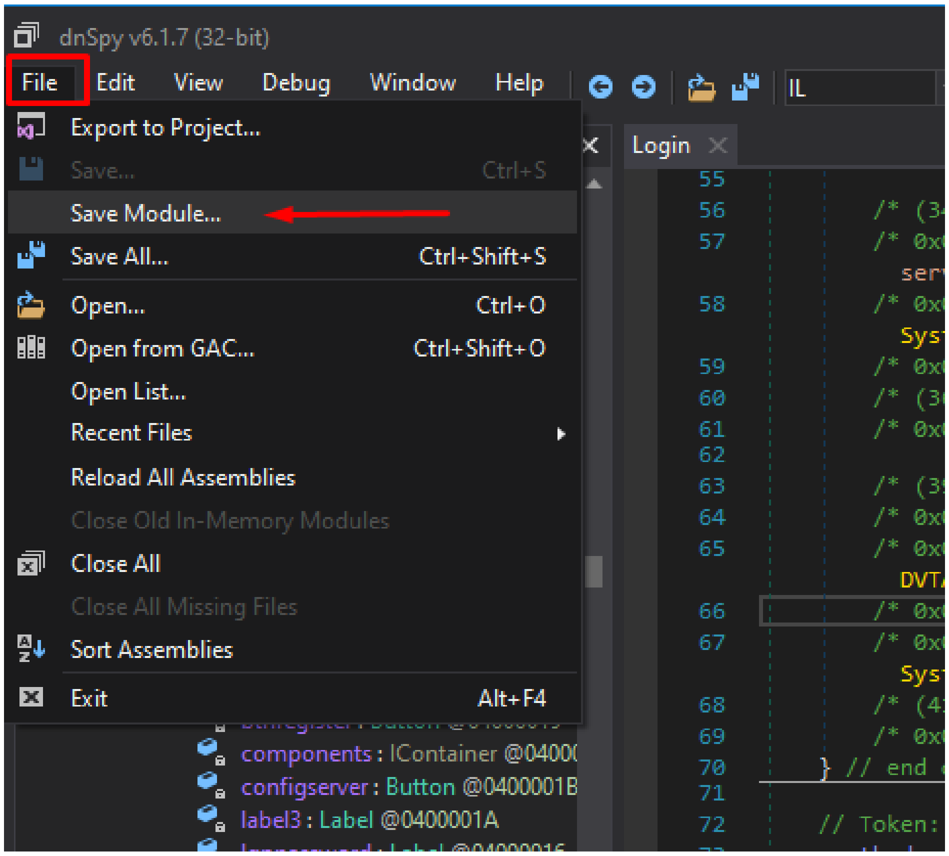
Instructions | Locals | Exception Handlers

Body Type: IL — Code Type: IL

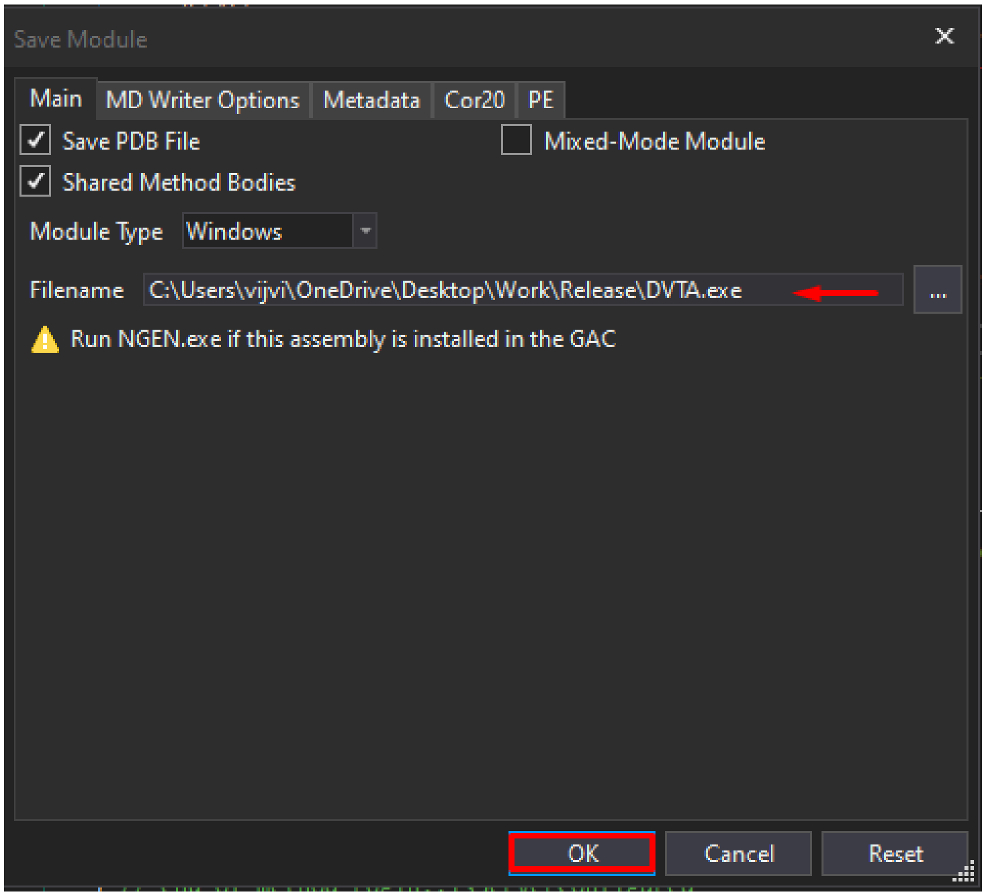
[ ] Keep Old MaxStack [✓] Init Locals — Header RVA: 0x2C9C — Header Offset: 0xE9C — MaxStack: 8 — LocalVarSig

```
Index  Offset  OpCode      Operand
0      0000    ldarg.0
1      0001    call        instance void [System.Windows.Forms]System.Windows.Forms.Form::.ctor(
2      0006    ldarg.0
3      0007    call        instance void DVTA.Login::InitializeComponent()
4      000C    ldarg.0
5      000D    call        instance bool DVTA.Login::IsBeingDebugged()
6      0012    brfalse.s   9 (001A) ldarg.0
7      0014    ldc.i4.1
8      0015    call        void [mscorlib]System.Environment::Exit(int32)
9      001A    ldarg.0
10     001B    call        instance bool DVTA.Login::isServerConfigured()
11     0020    brfalse.s   16 (002E) ldarg.0
12     0022    ldstr       "This application is usable only after configuring the server"
13     0027    call        valuetype [System.Windows.Forms]System.Windows.Forms.DialogResult [Sy
14     002C    pop
15     002D    ret
16     002E    ldarg.0
17     002F    ldfld       class [System.Windows.Forms]System.Windows.Forms.Button DVTA.Login::c
18     0034    ldc.i4.0
19     0035                tance void [System.Windows.Forms]System.Windows.Forms.Control::set_
20     003A
```

Now save the configuration by navigating to the **File > Save Module**



Now save this file in the same place where your original DVTA exist so that it can replace the original DVTA.exe as shown below



[✓] Save PDB File
[ ] Mixed-Mode Module
[✓] Shared Method Bodies
Module Type: Windows
Filename: C:\Users\vijvi\OneDrive\Desktop\Work\Release\DVTA.exe
Run NGEN.exe if this assembly is installed in the GAC

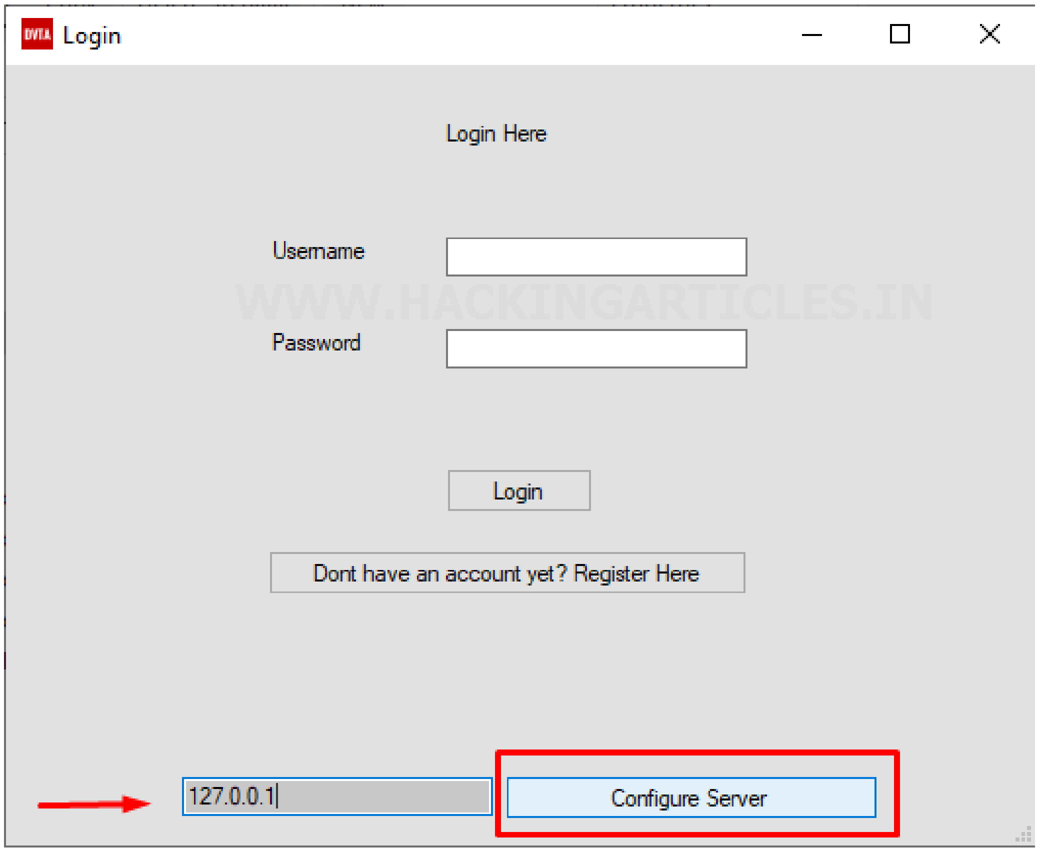
Now we can close the dnSpy application and open up the Modified application DVTA.exe by going to the same directory.

## Configure & log in to the Modified DVTA Application

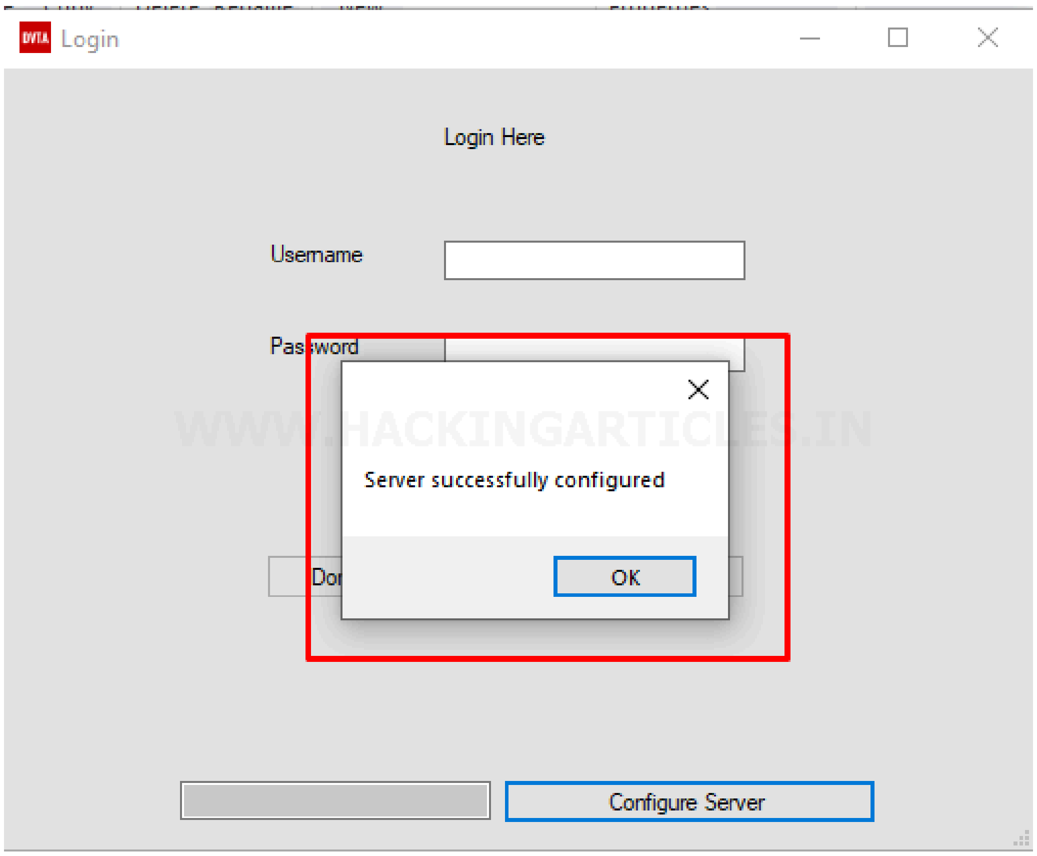
Hurray…as you can see the "configure server button is enabled

So, we're all set to configure the server. As I told you earlier, we are going to use the same server as the database.

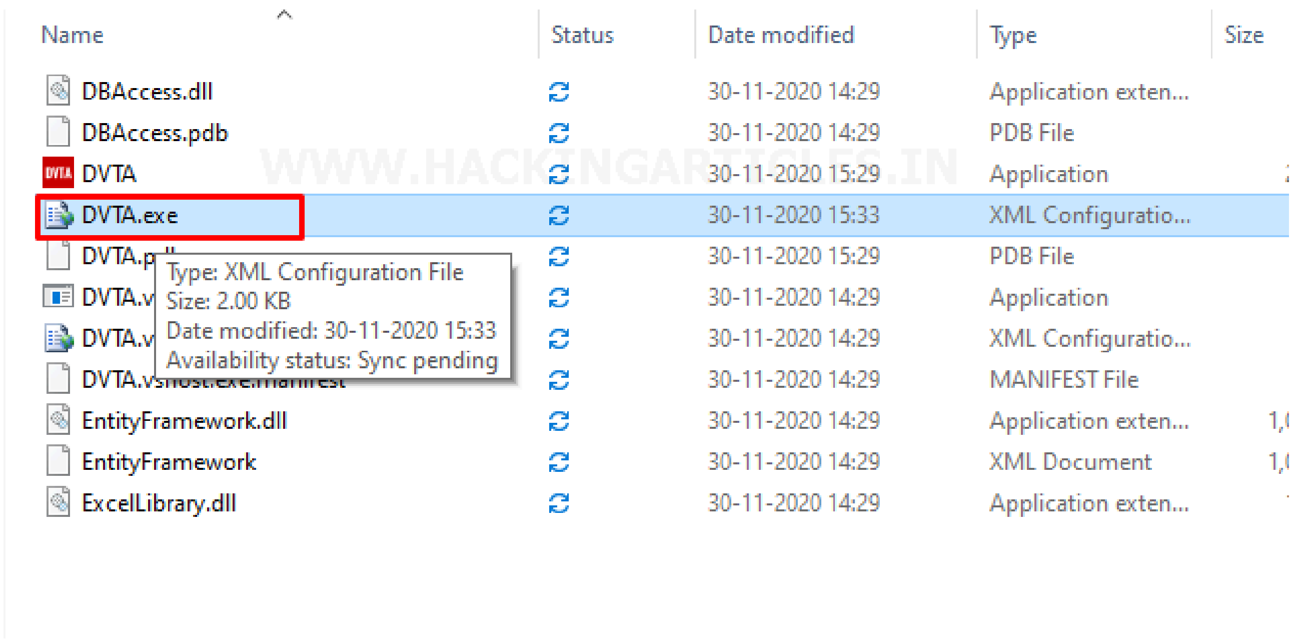
So, let's enter **"127.0.0.1"** as our server and click on **"configure server".**



As you can see our server is successfully configured.



Now, just to make sure everything is fine to come back to the directory of release folder of DVTA and give a double-tap to the DVTA.exe XML configuration file which is just below the DVTA application as shown below



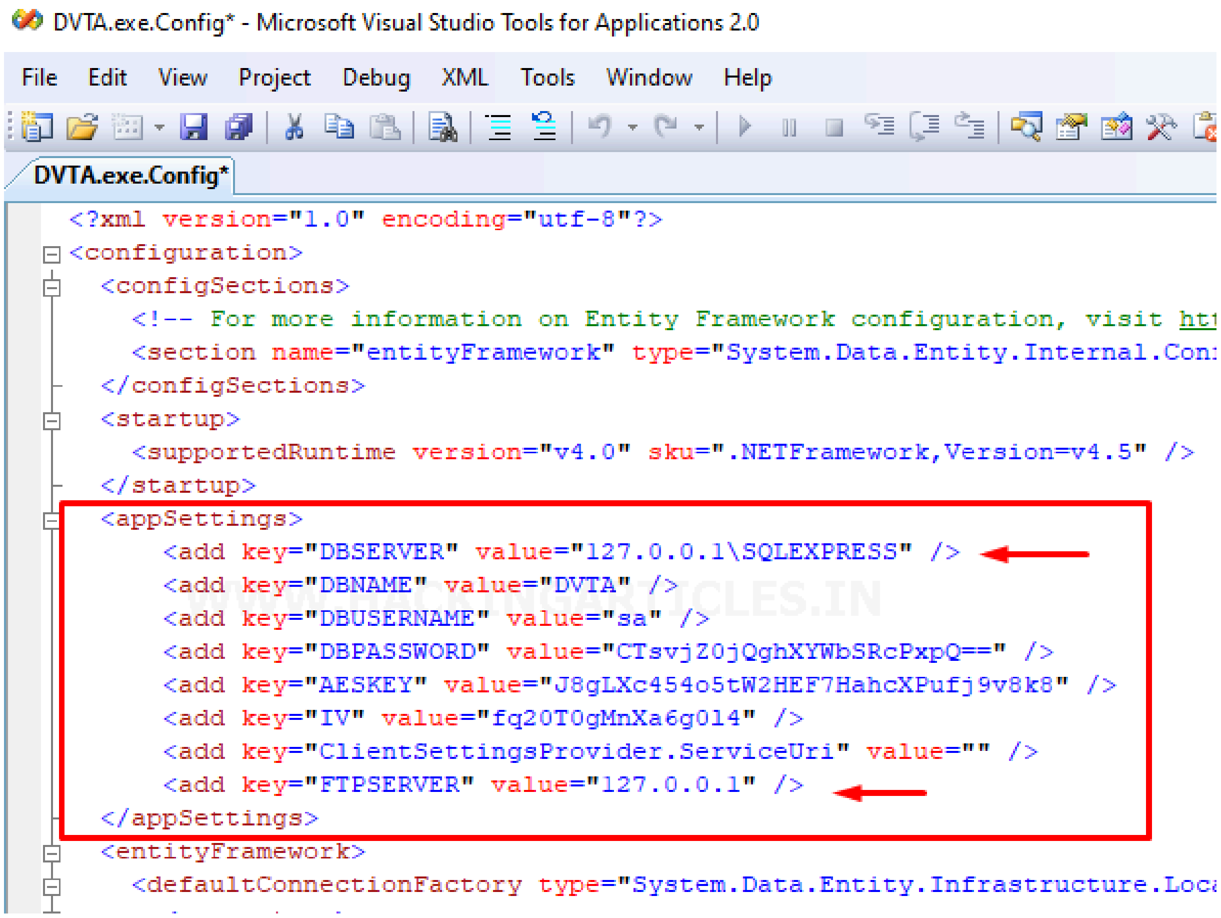
| Name | Status | Date modified | Type | Size |
|---|---|---|---|---|
| DBAccess.dll | | 30-11-2020 14:29 | Application exten... | |
| DBAccess.pdb | | 30-11-2020 14:29 | PDB File | |
| DVTA | | 30-11-2020 15:29 | Application | |
| DVTA.exe | | 30-11-2020 15:33 | XML Configuratio... | |
| DVTA.p | | 30-11-2020 15:29 | PDB File | |
| DVTA.v | | 30-11-2020 14:29 | Application | |
| DVTA.v | | 30-11-2020 14:29 | XML Configuratio... | |
| DVTA.vshost.exe.manifest | | 30-11-2020 14:29 | MANIFEST File | |
| EntityFramework.dll | | 30-11-2020 14:29 | Application exten... | 1,( |
| EntityFramework | | 30-11-2020 14:29 | XML Document | 1,( |
| ExcelLibrary.dll | | 30-11-2020 14:29 | Application exten... | |

Type: XML Configuration File
Size: 2.00 KB
Date modified: 30-11-2020 15:33
Availability status: Sync pending

This is basically a configuration file. Give it a double tap and open the configuration file.

When you see the application settings you can see that the **Database server is "127.0.0.1/SQLExpress"** and the "**FTP server is also configured to 127.0.0.1**".

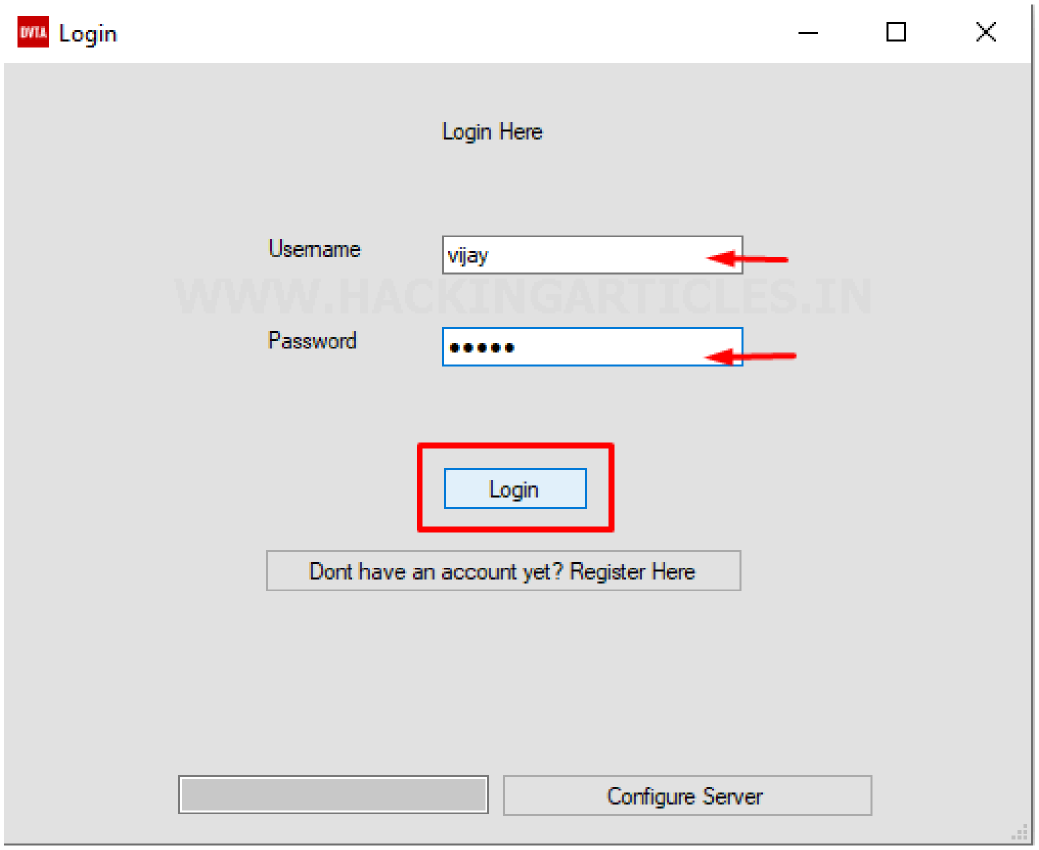
Awesome… now we're good with the configuration.



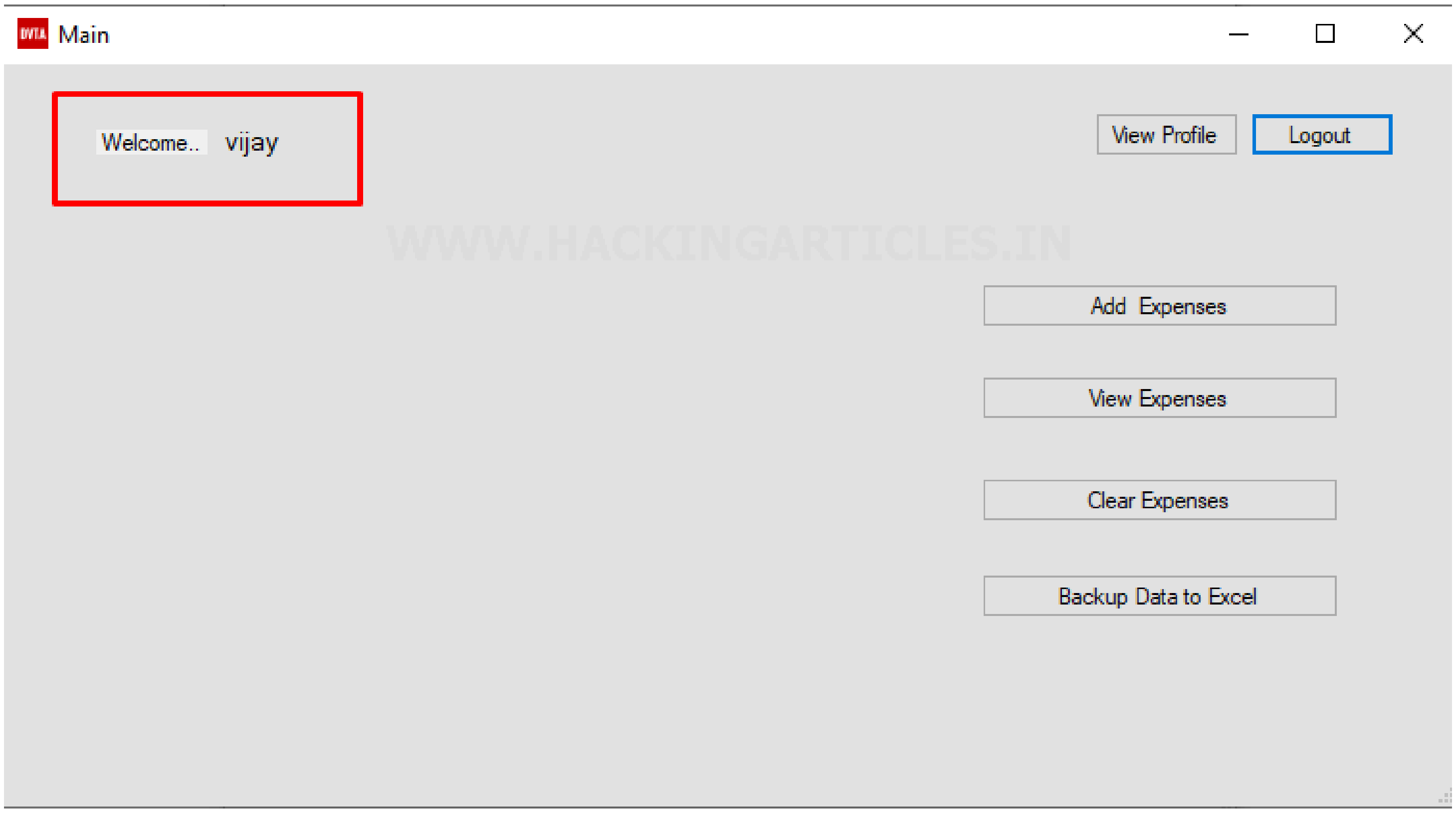
DVTA.exe.Config* - Microsoft Visual Studio Tools for Applications 2.0

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit htt
    <section name="entityFramework" type="System.Data.Entity.Internal.Con
  </configSections>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
  </startup>
  <appSettings>
      <add key="DBSERVER" value="127.0.0.1\SQLEXPRESS" />
      <add key="DBNAME" value="DVTA" />
      <add key="DBUSERNAME" value="sa" />
      <add key="DBPASSWORD" value="CTsvjZ0jQghXYWbSRcPxpQ==" />
      <add key="AESKEY" value="J8gLXc454o5tW2HEF7HahcXPufj9v8k8" />
      <add key="IV" value="fq20T0gMnXa6g0l4" />
      <add key="ClientSettingsProvider.ServiceUri" value="" />
      <add key="FTPSERVER" value="127.0.0.1" />
  </appSettings>
  <entityFramework>
    <defaultConnectionFactory type="System.Data.Entity.Infrastructure.Loca
```

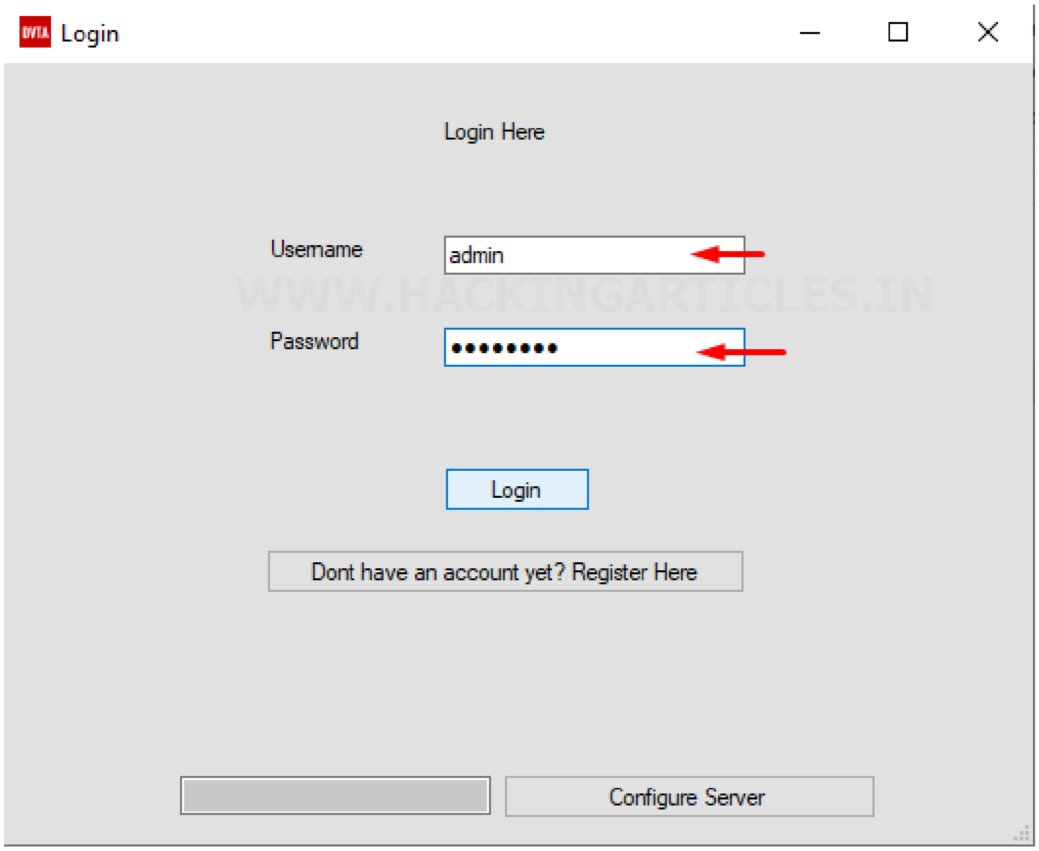
Now open the modified DVTA application and log in as the user **"vijay"** that we created previously on the SQL Server setup.



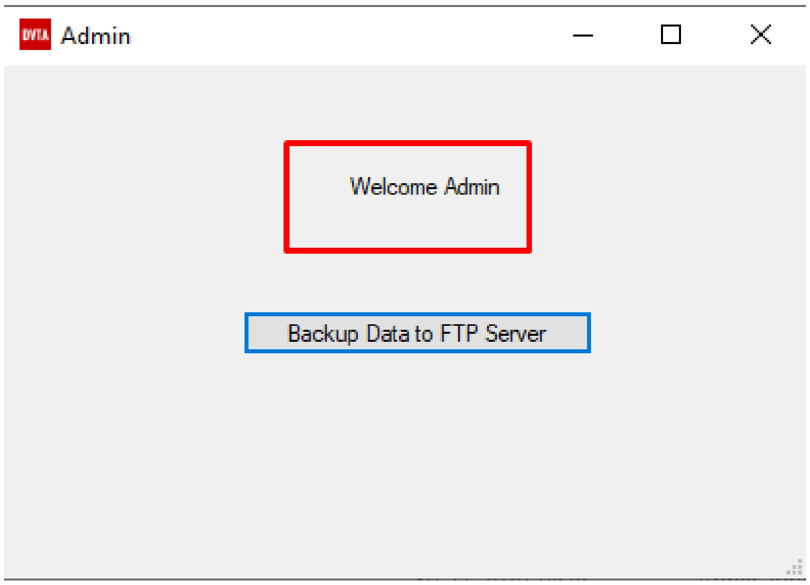
Great, as we can see we have successfully logged in to the application



Let’s also verify another user, for example, admin and check which rights admin have



Great, as we can see everything works fine and we’re successfully logged in as user admin.



Congratulations.

Now if you have been done until here then you are all set with your Damn Vulnerable Thick Client Application setup.

From the next part, we are going to systematically pentesting this application for various issues.

One most important thing **"you can use all these steps from here as a checklist for your real-world Thick Client Penetration Testing Engagements".**