

K P Uday Krishna

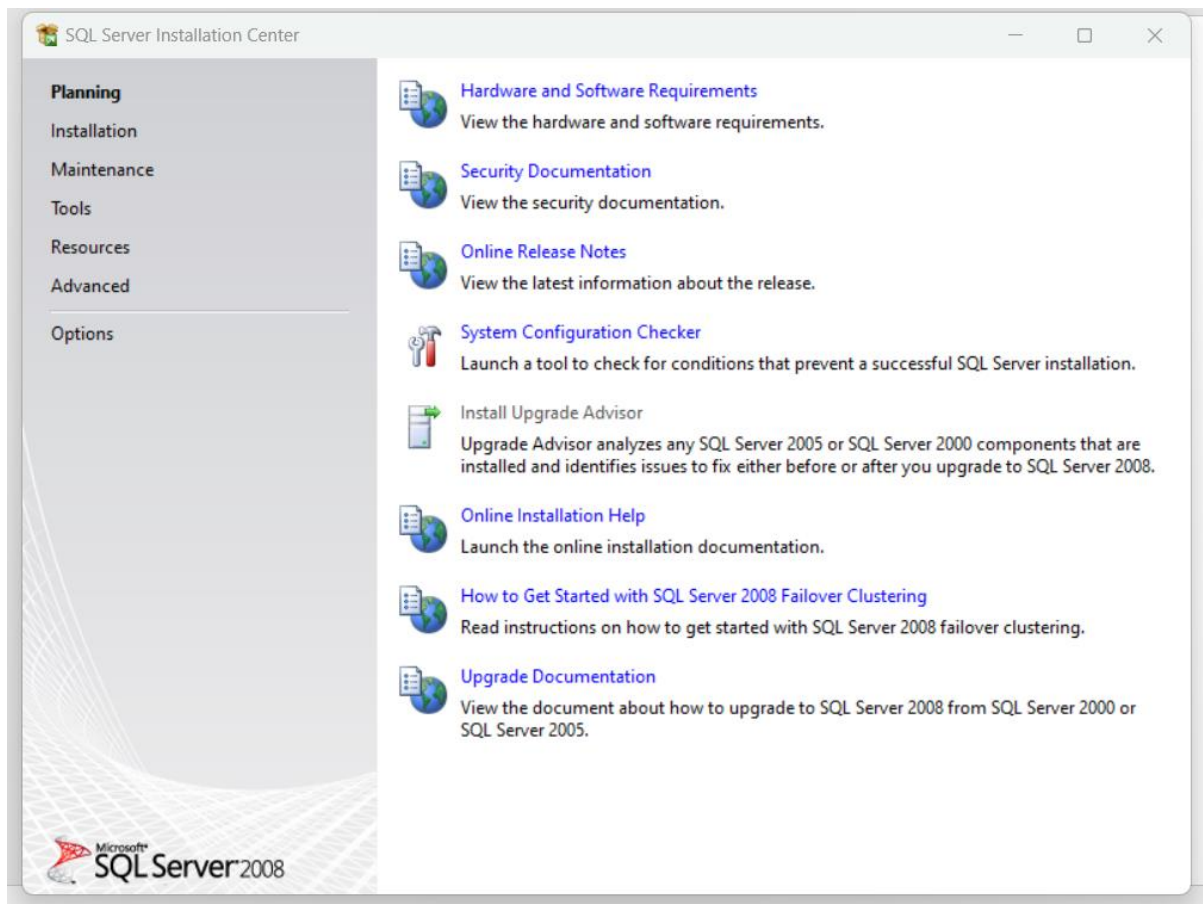
CB.SC.P2CYS23012

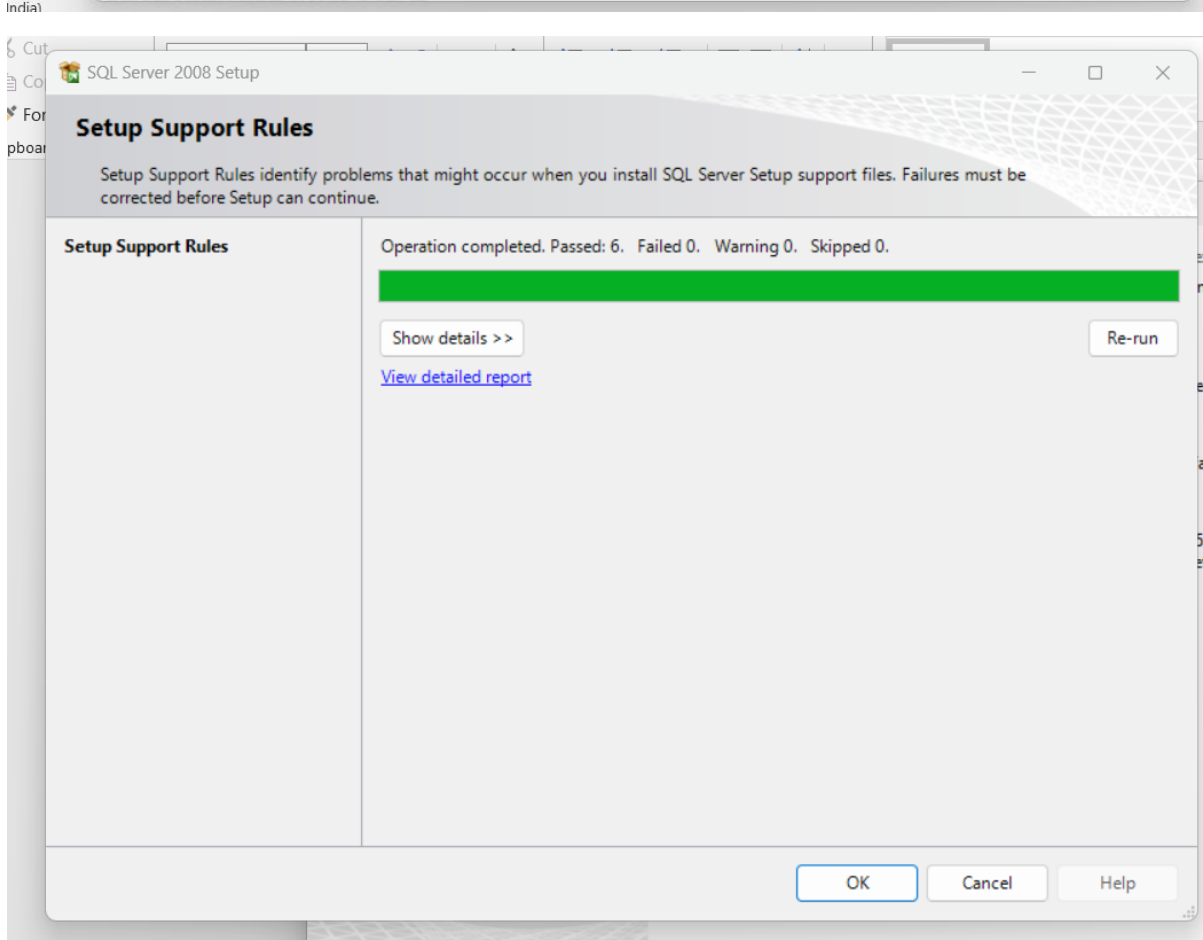
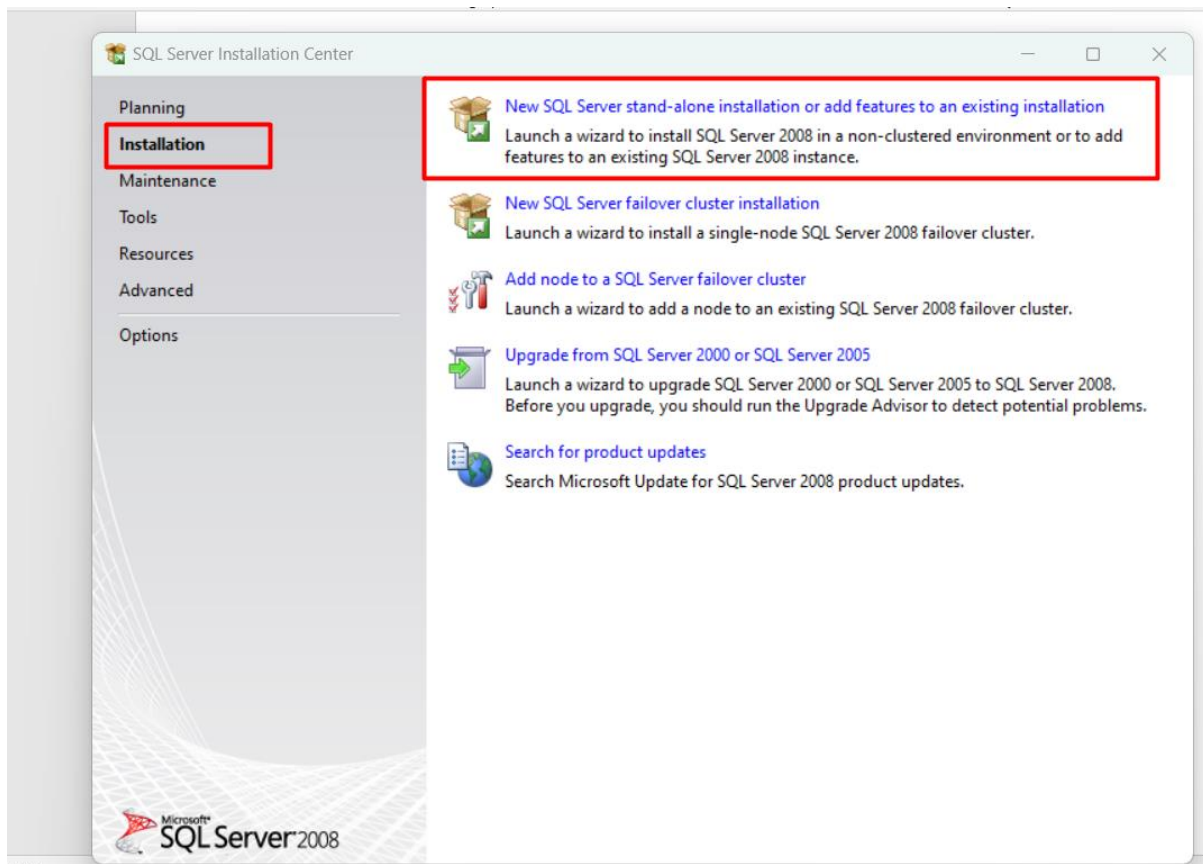
SSE Assignment 2

DVTA Setup - DVTA - Part 1 - Setup (parsiya.net)

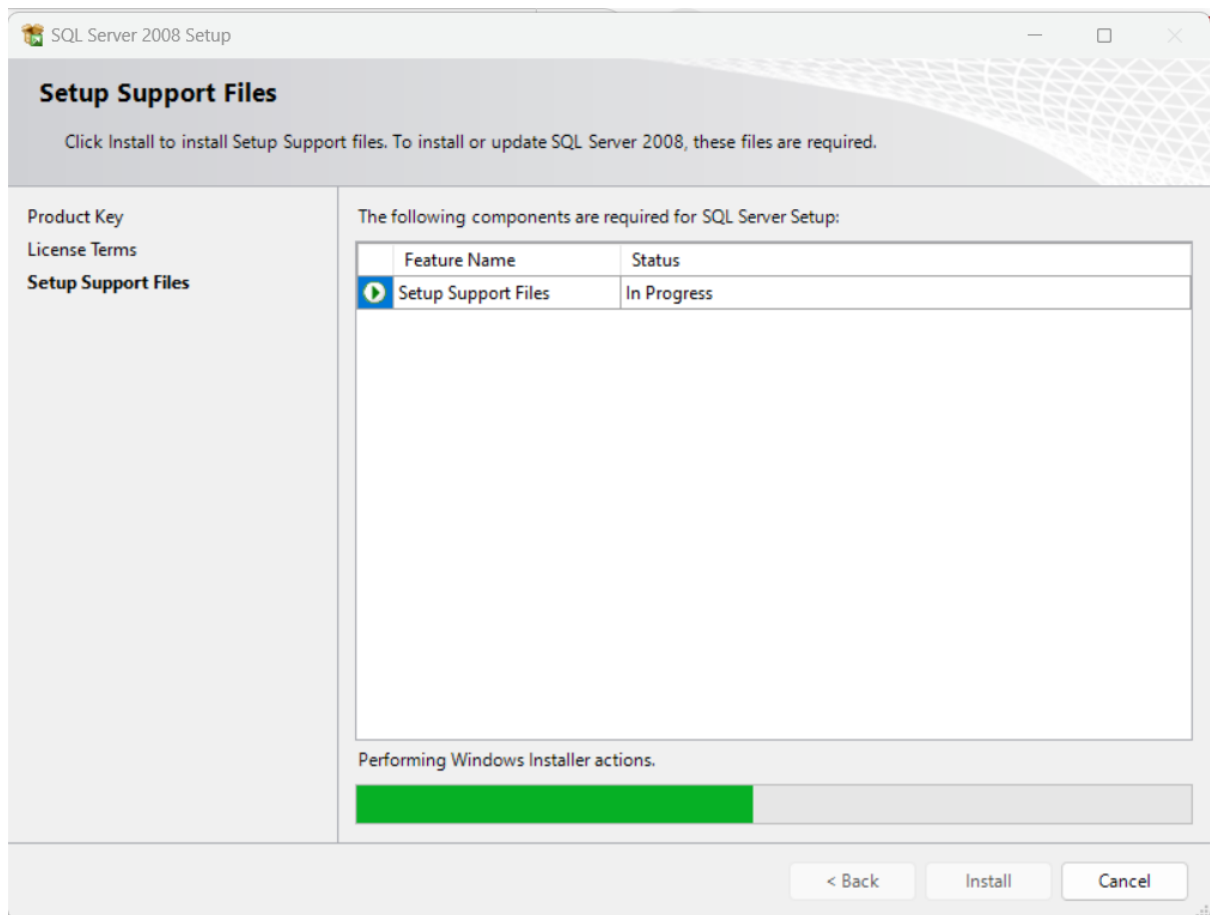
Use CFF Framework to analyse custom DVTA.exe created above.

To Start with the DVTA, first we need to setup SQL Server 2008

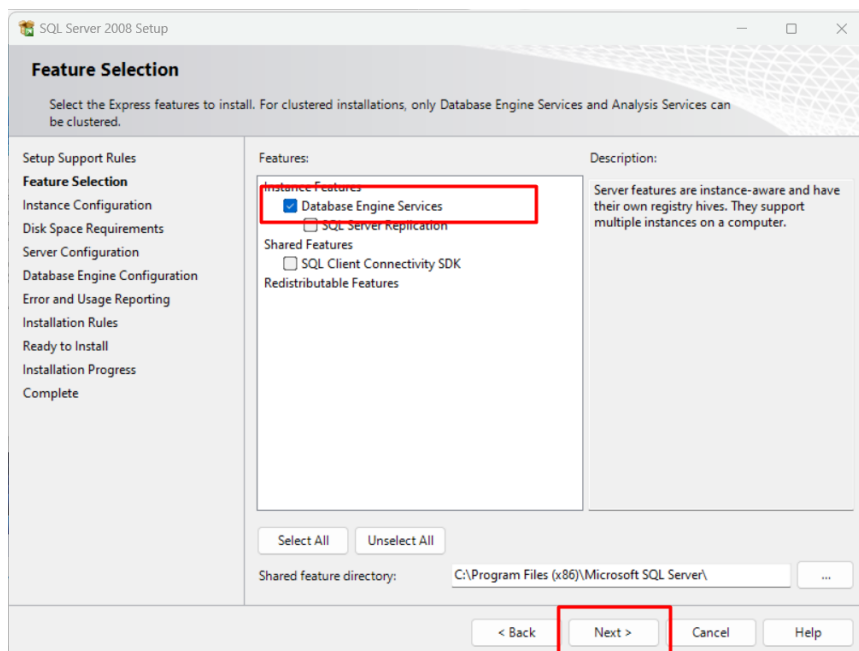




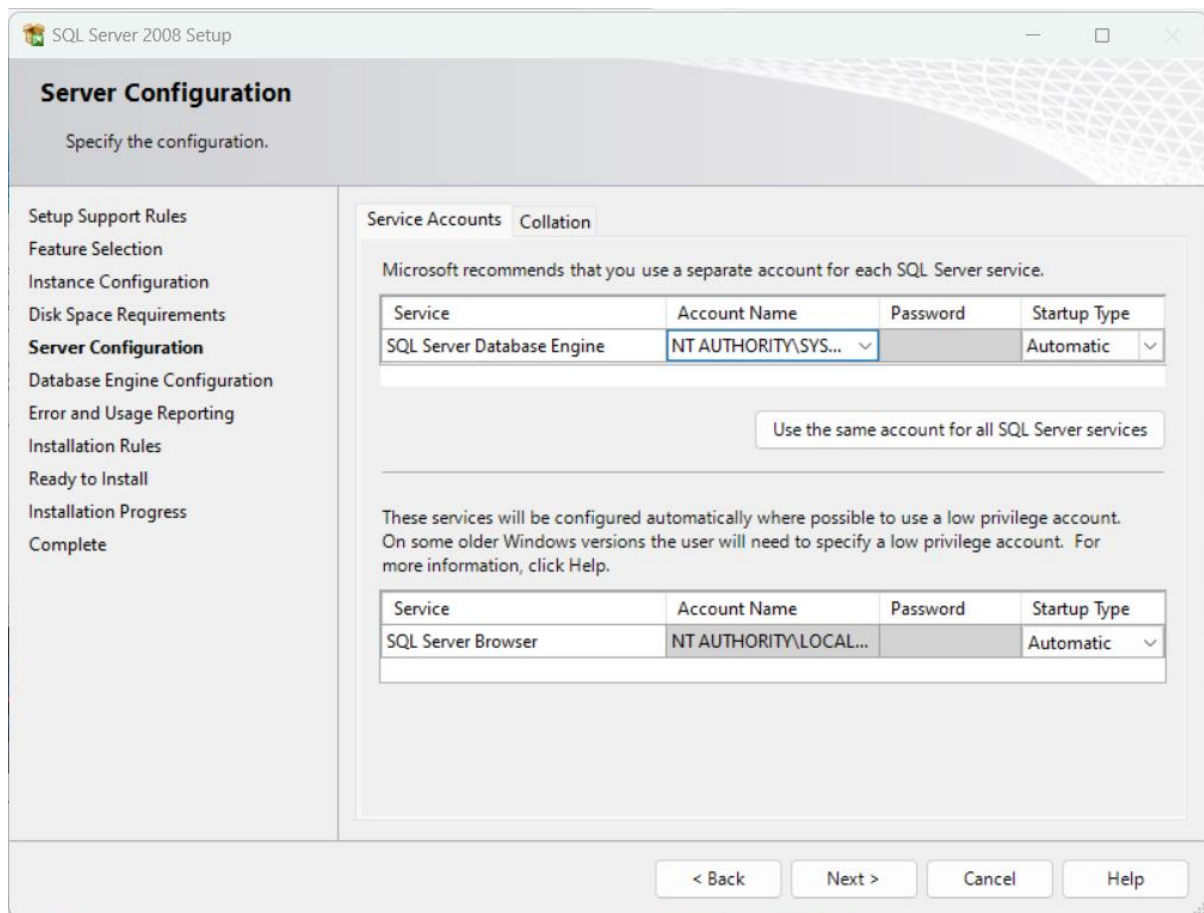
We have to setup support files



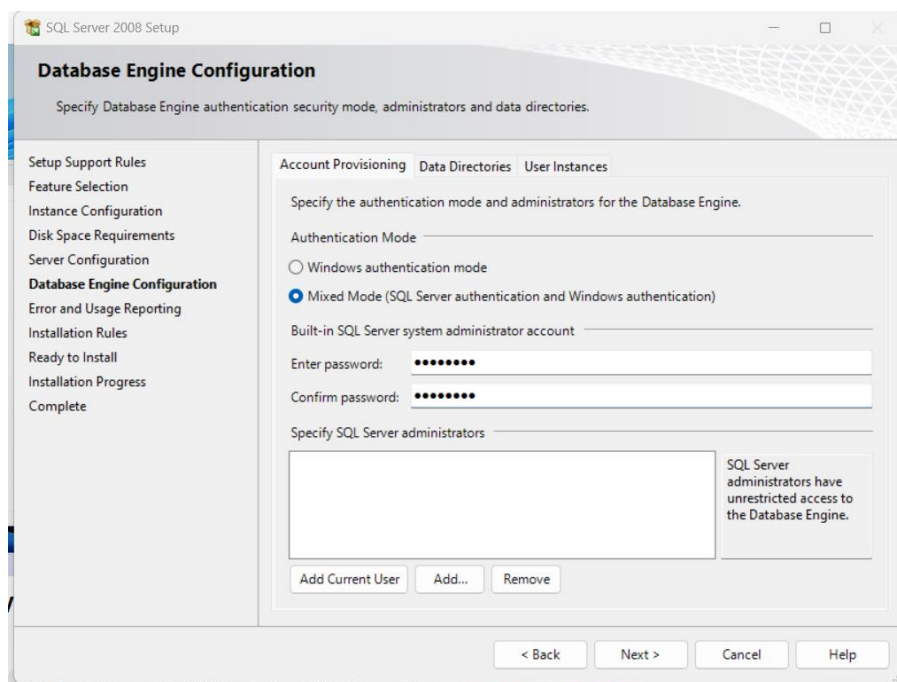
Add Database Engine



Now configure server

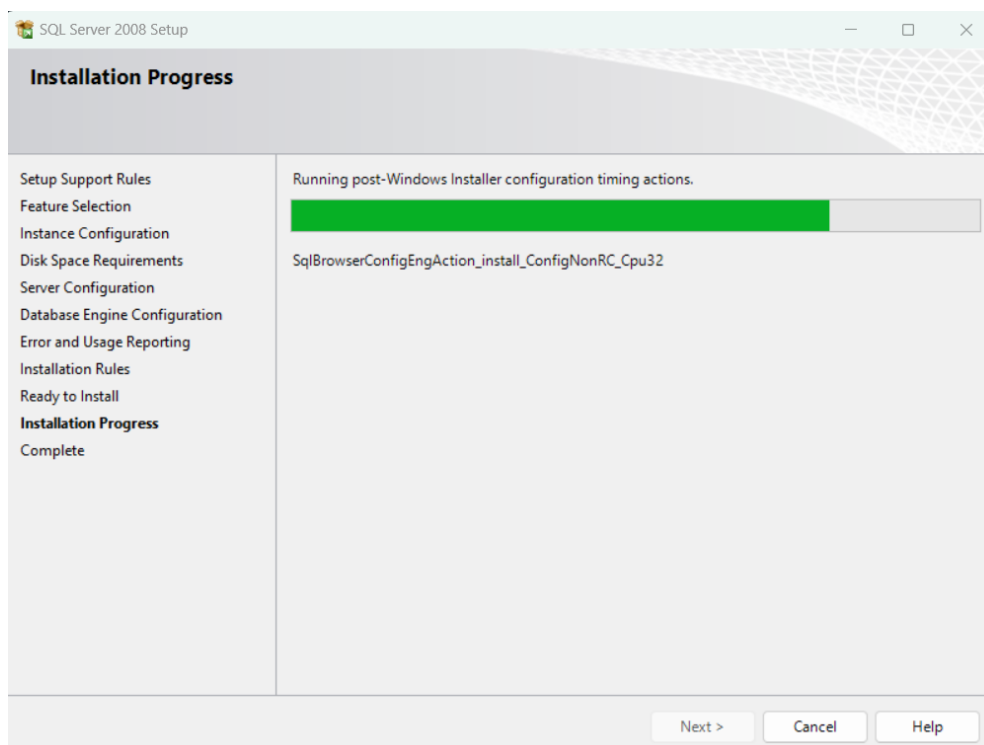
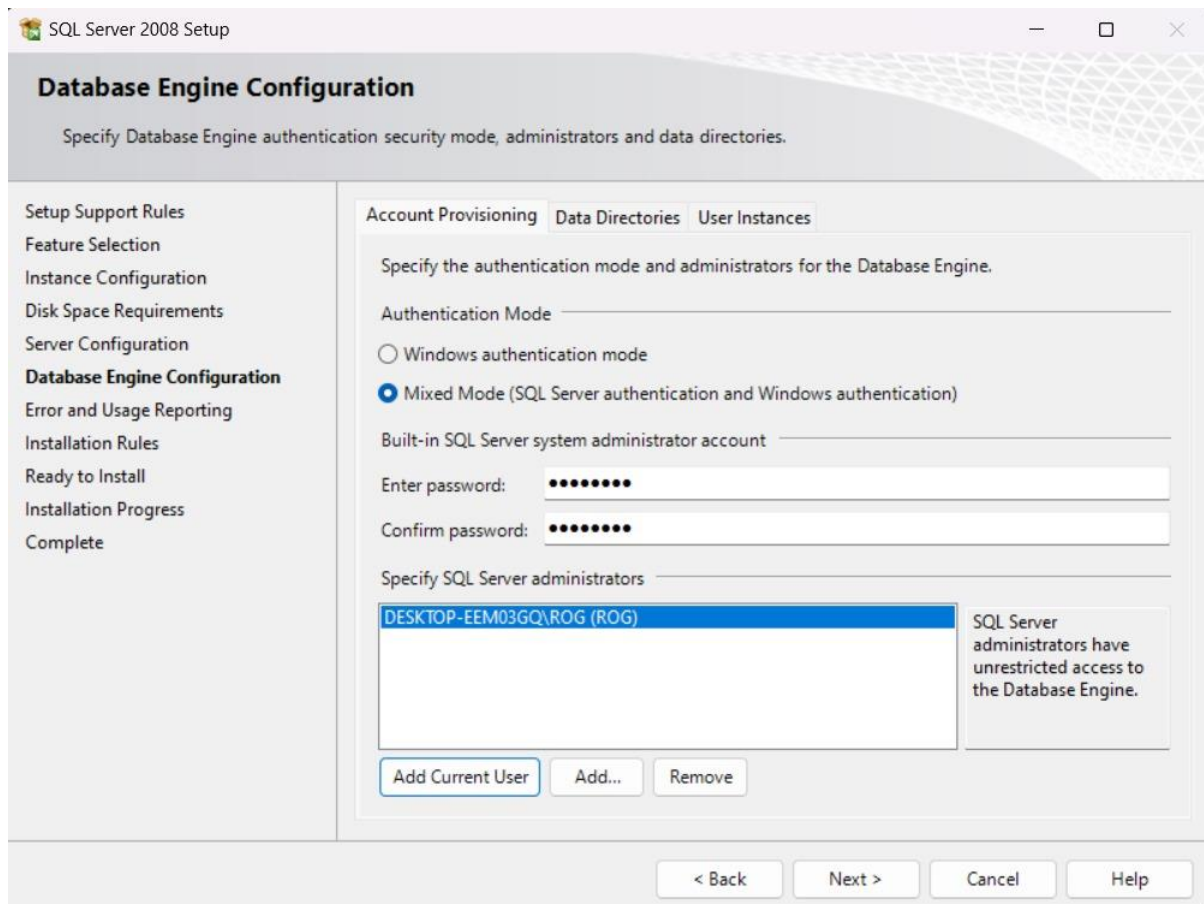


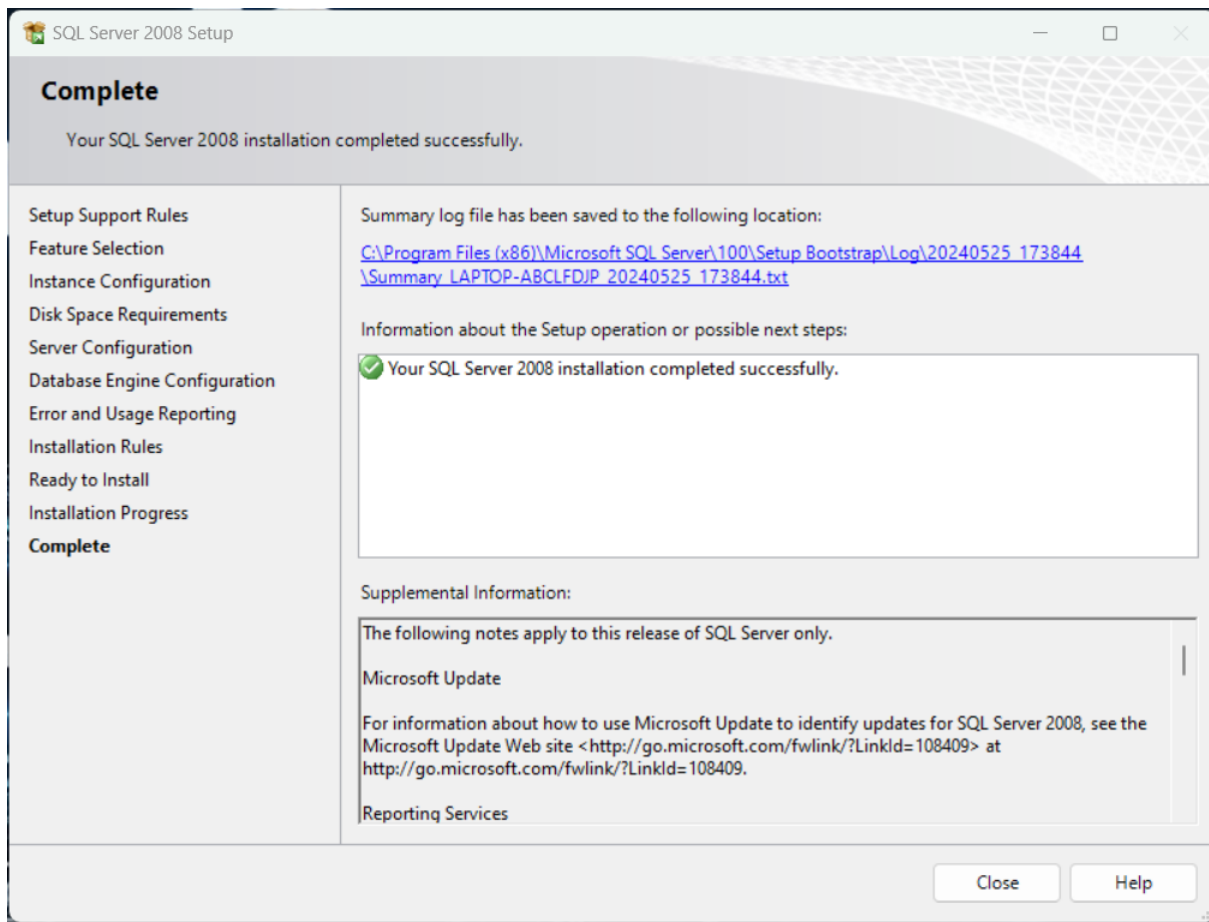
Add a password as a configuration of Server



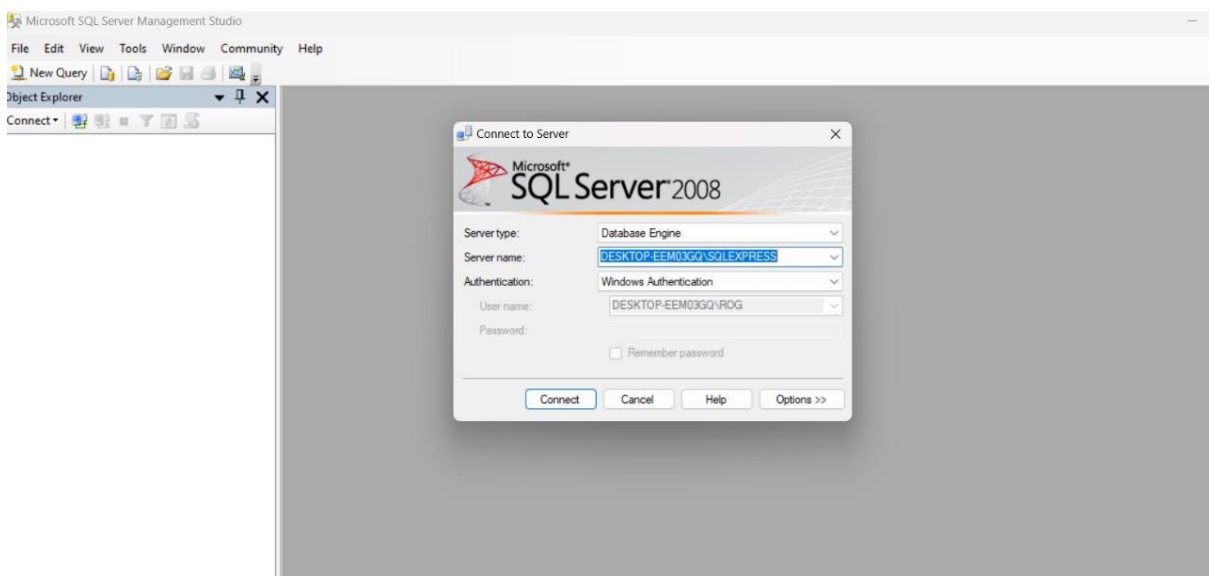
Password Should be – 12345678

- We need to add users SQL Server Administrator





- start SQL Server 2008



Create New Database

ScriptHelp

Database name:DVTA

Owner:<default>

☒ Use full-text indexing

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth
DVTA	Rows ...	PRIMARY	2	By 1 MB, unrestricted growth
DVTA_log	Log	Not Applicable	1	By 10 percent, unrestricted growth

Add

Remove

OK

Cancel

Sql Server Configuration Manager

FileActionViewHelp

SQL Server Configuration Manager (Local)

SQL Server Services

SQL Server Network Configuration (32bit)

Protocols for SQLEXPRESS

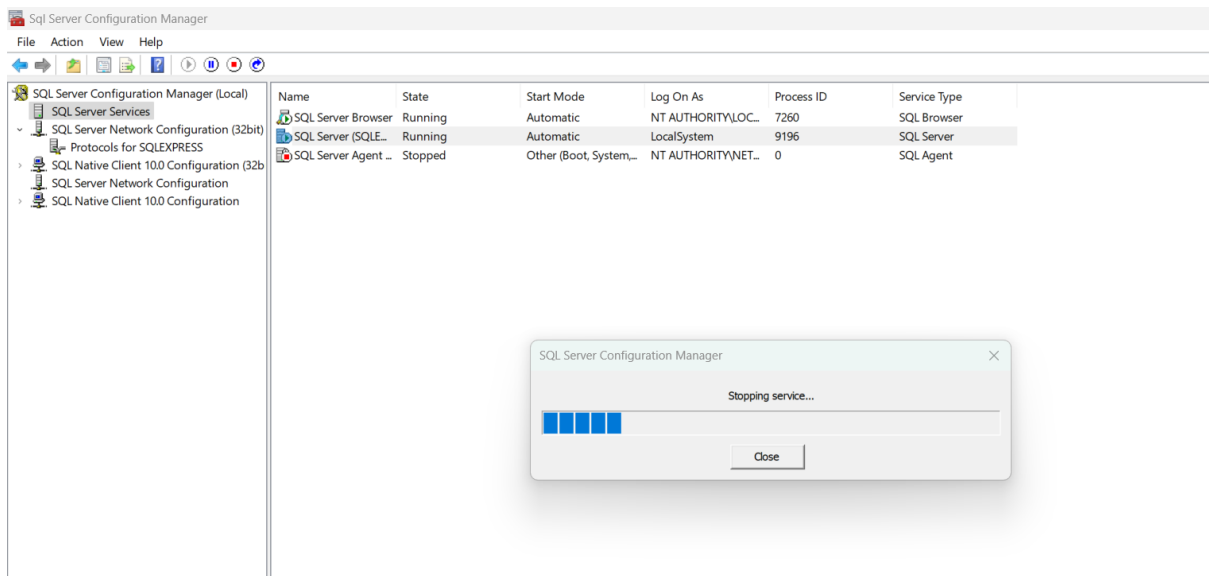
SQL Native Client 10.0 Configuration (32b

SQL Server Network Configuration

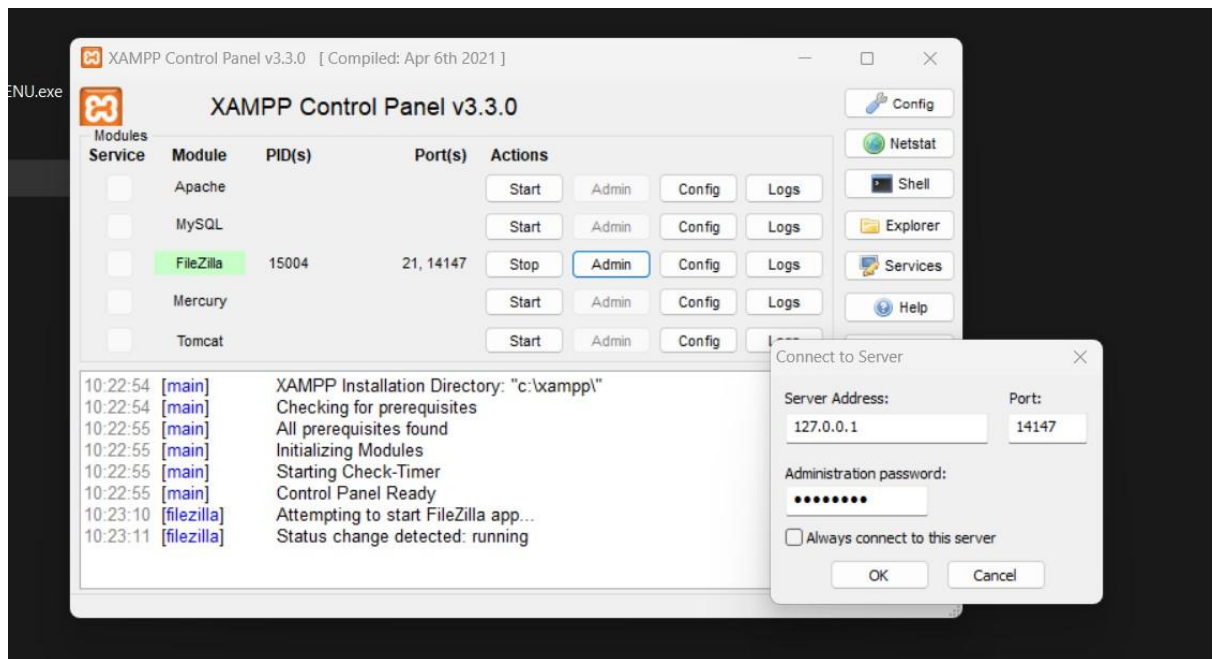
SQL Native Client 10.0 Configuration

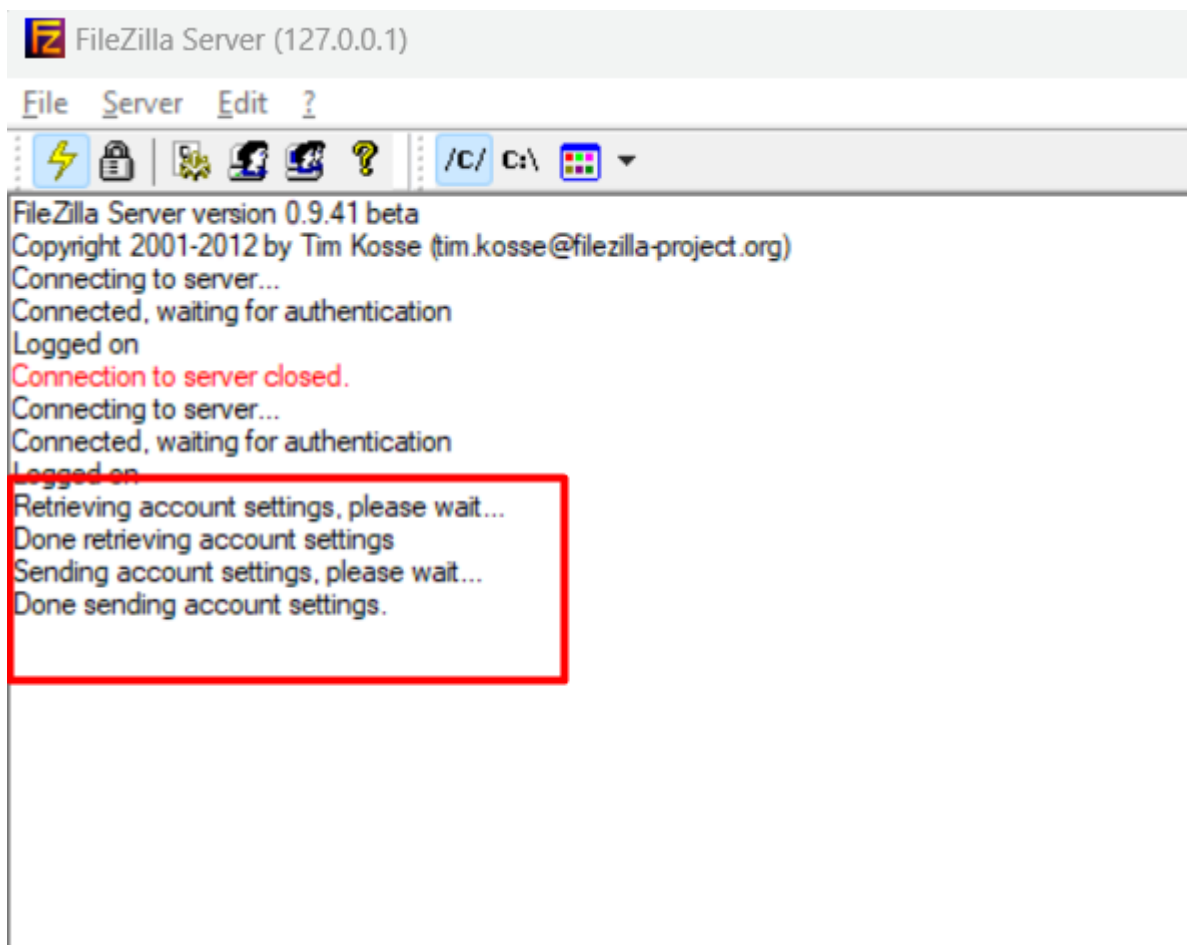
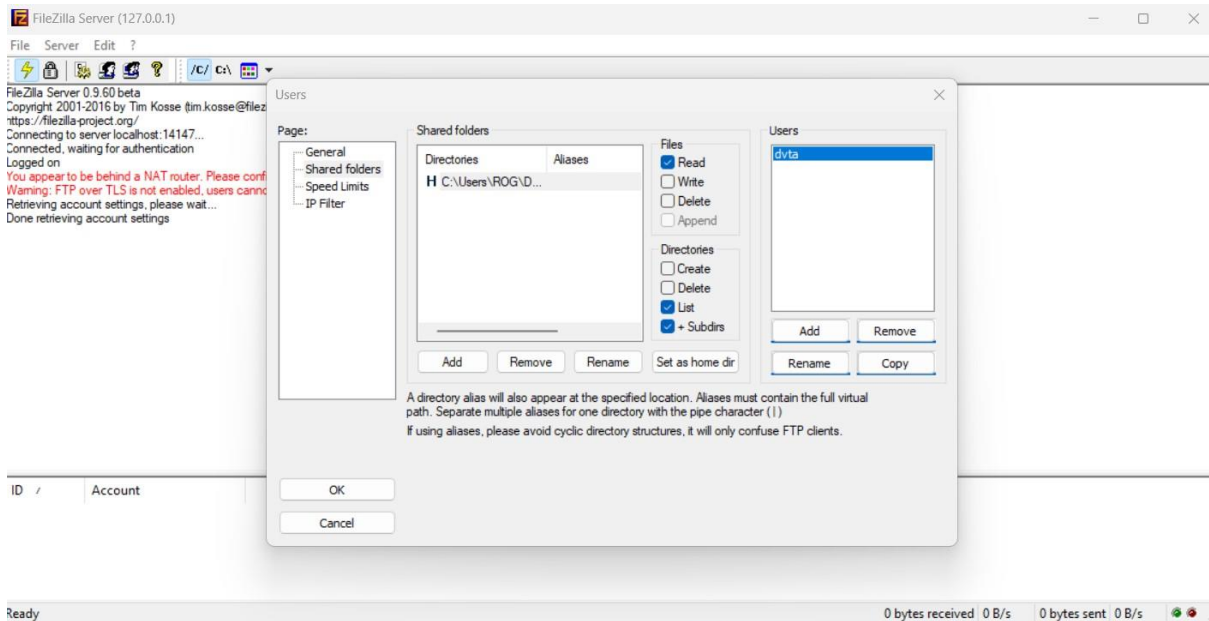
Protocol Name	Status
Shared Memory	Enabled
Named Pipes	Disabled
TCP/IP	Enabled
VIA	Disabled

- Restart SQL Server



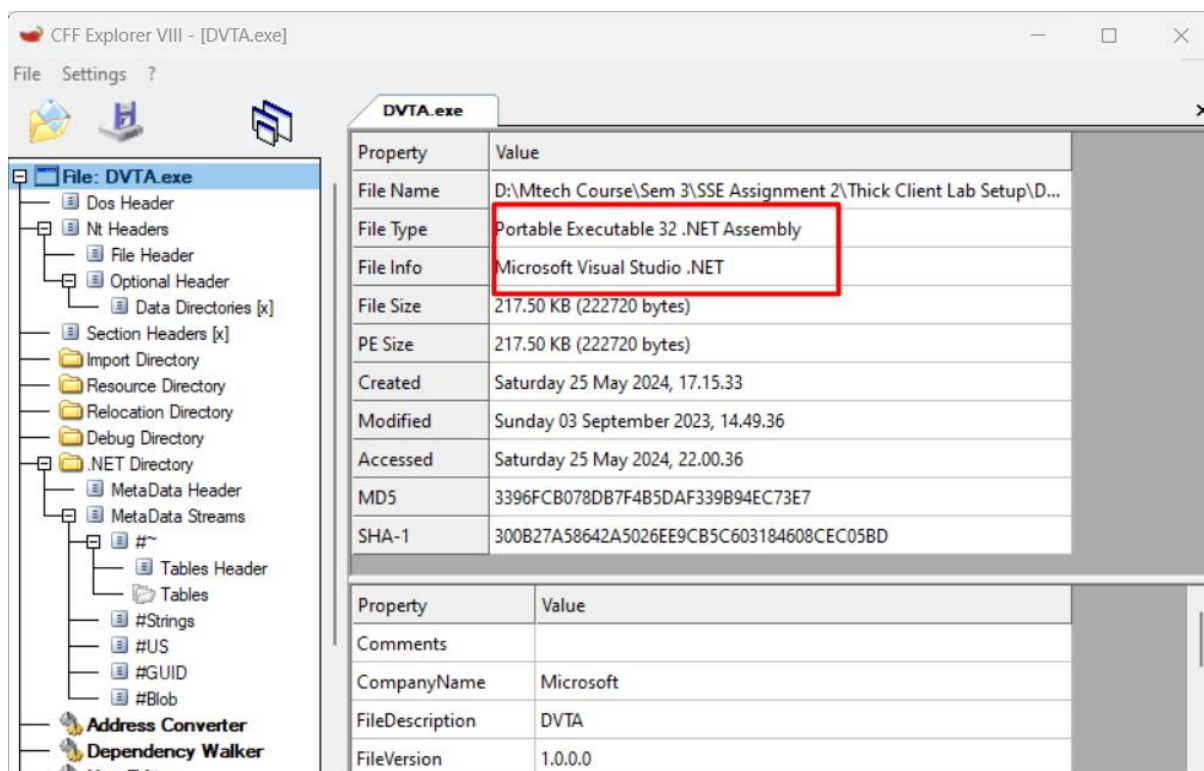
- Start a Filezilla Server in your computer
- Going to Admin Panel





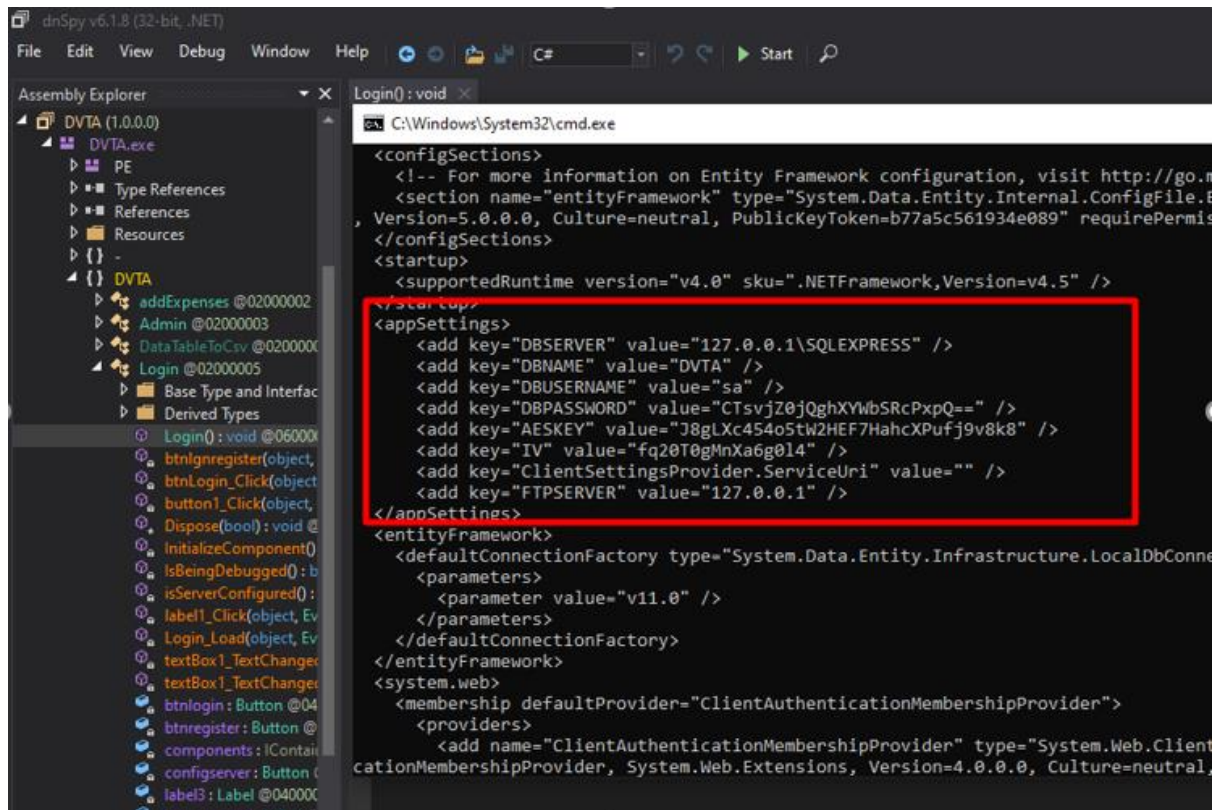
1) Identify the Application architecture, languages and frameworks used?

- Upon opening the dvta.exe in CFF-explorer, we can identify the following information
- Architecture – 32 bit & 2 tier [Since it communicates with the database.]
- Languages used - .NET Assembly
- Frameworks - .NET framework



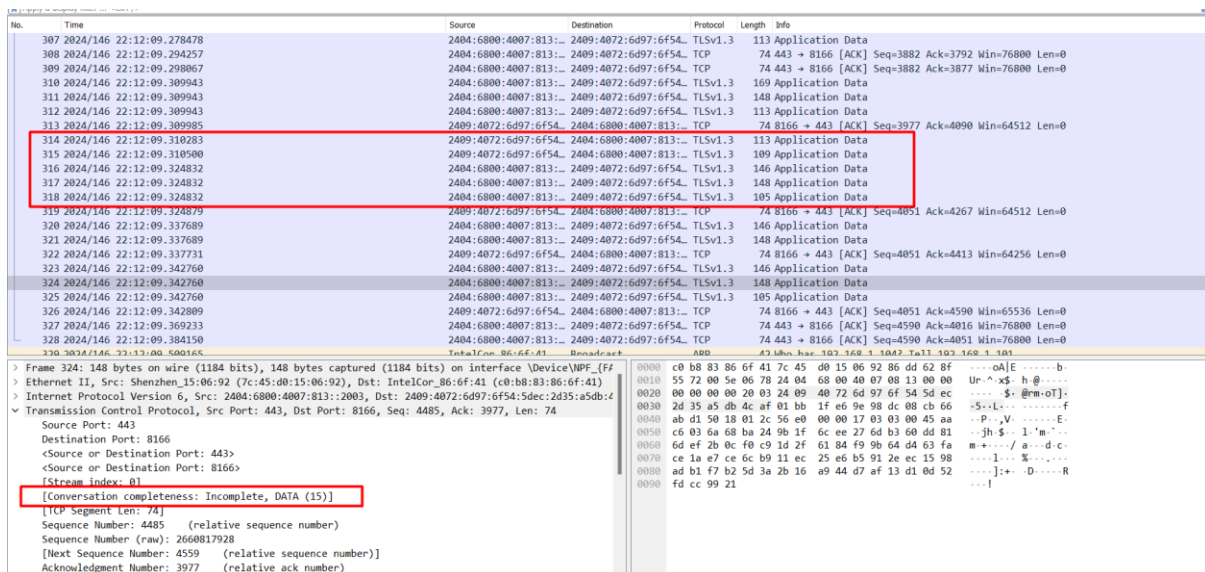
2. Decompile and try to retrieve the source code of the application? Also, check if any hardcoded sensitive information is found?

- By decompiling the application using DNSpy or MS Visual studio tools, we can see the source code of the application.



3) Sniff the traffic between client and server. Identify which protocol is being used for communication?

- With Wireshark we can sniff the client and server
- Next inspect the contents of the packets to determine whether the app is using TCP/UDP protocol for communication.
- In the packet inspection window, we can see that the protocol used by the dvta is TCP protocol.



4) Identify if unencrypted communication is happening between client and server?

- In this case we can use either ECHIMIRAGE / Wireshark. We have used Echomirage here.
- From the output we can see that when we login to DVTA, the data is sent as plaintext format to the database.

Traffic Log	Rules	Intercept
Outbound TCP data to 192.168.56.110:1433		
	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	0123456789ABCDEF
0x0000	01 09 00 B2 00 00 01 00 16 00 00 00 12 00 00 00	. . . *
0x0010	02 00 00 00 00 00 00 00 00 00 01 00 00 00 73 00 s.
0x0020	65 00 6C 00 65 00 63 00 74 00 20 00 69 00 74 00	e . l . e . c . t . i . t .
0x0030	65 00 6D 00 2C 00 20 00 70 00 72 00 69 00 63 00	e . m . , . . p . r . i . c .
0x0040	65 00 2C 00 20 00 64 00 61 00 74 00 65 00 2C 00	e . , . . d . a . t . e . , .
0x0050	74 00 69 00 6D 00 65 00 20 00 66 00 72 00 6F 00	t . i . m . e . . f . r . o .
0x0060	6D 00 20 00 65 00 78 00 70 00 65 00 6E 00 73 00	m . . e . x . p . e . n . s .
0x0070	65 00 73 00 20 00 77 00 68 00 65 00 72 00 65 00	e . s . . w . h . e . r . e .
0x0080	20 00 65 00 6D 00 61 00 69 00 6C 00 3D 00 27 00	. e . m . a . i . l . = . '
0x0090	72 00 61 00 79 00 6D 00 6F 00 6E 00 64 00 40 00	r . a . y . m . o . n . d . @ .
0x00A0	74 00 65 00 73 00 74 00 2E 00 63 00 6F 00 6D 00	t . e . s . t . . . c . o . m .
0x00B0	27 00	' .

5) Capture and analyse the communication using proxy tools (eg: Burpsuite, Echo mirage).

- From the below screenshot, we can understand that using wireshark we're able to capture & analyse the requests that are being sent to the database and to the server.

```

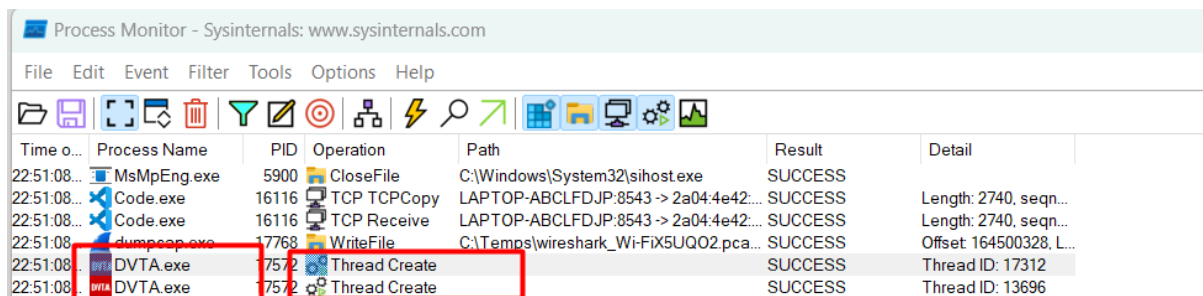
266 2024/146 22:11:58.929735 2404:6800:4007:826:: 2409:4072:6497:6f54:: TCP 74 443 + 8170 [ACK] Seq=3443 Ack=3145 Win=7680 Len=0
267 2024/146 22:11:58.929735 2404:6800:4007:826:: 2409:4072:6497:6f54:: TLSv1.3 113 Application Data
268 2024/146 22:11:58.95391 2404:6800:4007:826:: 2409:4072:6497:6f54:: TCP 74 443 + 8173 [ACK] Seq=2087 Ack=3318 Win=7680 Len=0
269 2024/146 22:11:58.95391 2404:6800:4007:826:: 2409:4072:6497:6f54:: TCP 74 443 + 8170 [ACK] Seq=3482 Ack=3318 Win=79616 Len=0
270 2024/146 22:11:58.971949 2409:4072:6497:6f54:: 2404:6800:4007:826:: TCP 74 8170 + 443 [ACK] Seq=3813 Ack=3482 Win=65536 Len=0
271 2024/146 22:11:58.980646 2404:6800:4007:826:: 2409:4072:6497:6f54:: TLSv1.3 139 Application Data
272 2024/146 22:11:58.980646 2404:6800:4007:826:: 2409:4072:6497:6f54:: TLSv1.3 105 Application Data
273 2024/146 22:11:58.980646 2404:6800:4007:826:: 2409:4072:6497:6f54:: TLSv1.3 113 Application Data
274 2024/146 22:11:58.980813 2409:4072:6497:6f54:: 2404:6800:4007:826:: TCP 74 8170 + 443 [ACK] Seq=3813 Ack=3617 Win=65536 Len=0
275 2024/146 22:11:58.982728 2409:4072:6497:6f54:: 2404:6800:4007:826:: TLSv1.3 113 Application Data
276 2024/146 22:11:58.982877 2409:4072:6497:6f54:: 2404:6800:4007:826:: TLSv1.3 109 Application Data
277 2024/146 22:11:58.938475 2404:6800:4007:826:: 2409:4072:6497:6f54:: TCP 74 443 + 8170 [ACK] Seq=3617 Ack=3887 Win=79616 Len=0
278 2024/146 22:11:59.141908 2404:6800:4007:826:: 2409:4072:6497:6f54:: TLSv1.3 143 Application Data
279 2024/146 22:11:59.141908 2404:6800:4007:826:: 2409:4072:6497:6f54:: TLSv1.3 105 Application Data
280 2024/146 22:11:59.142029 2409:4072:6497:6f54:: 2404:6800:4007:826:: TCP 74 8173 + 443 [ACK] Seq=3318 Ack=2987 Win=65536 Len=0
281 2024/146 22:11:59.146966 2409:4072:6497:6f54:: 2404:6800:4007:81f:: TCP 1444 8083 + 443 [ACK] Seq=1008 Ack=597 Win=254 Len=1370 [TCP segment of a reassemb.
282 2024/146 22:11:59.146966 2409:4072:6497:6f54:: 2404:6800:4007:81f:: TLSv1.2 553 Application Data
283 2024/146 22:11:59.147005 2404:6800:4007:826:: 2409:4072:6497:6f54:: TLSv1.3 113 Application Data
284 2024/146 22:11:59.147219 2409:4072:6497:6f54:: 2404:6800:4007:826:: TLSv1.3 113 Application Data
285 2024/146 22:11:59.147219 2409:4072:6497:6f54:: TCP 74 443 + 8083 [ACK] Seq=597 Ack=2378 Win=489 Len=0
286 2024/146 22:11:59.242911 2404:6800:4007:81f:: 2409:4072:6497:6f54:: TCP 74 443 + 8083 [ACK] Seq=597 Ack=2857 Win=489 Len=0
287 2024/146 22:11:59.249042 2404:6800:4007:826:: 2409:4072:6497:6f54:: TCP 74 443 + 8173 [ACK] Seq=3026 Ack=3357 Win=76800 Len=0
288 2024/146 22:11:59.352304 TotalLen: 86:6f:4c Broadcast: 480 47:4b:ba:102:168:3:1802:147:102:168:3:101

* Frame 282: 553 bytes on wire (4424 bits), 553 bytes captured (4424 bits) on interface \Device\NPF_{AEADBA8C-4580-B14E-AEAA-07BC}
* Ethernet II, Src: IntelCor_86:6f:4c (c0:8b:38:66:f1:4d), Dst: Shenzhen_15:06:92 (7c:45:d0:15:06:92)
* Internet Protocol Version 6, Src: 2409:4072:6497:6f54::826::d5c4:2d35::a5db::dc4f, Dst: 2404:6800:4007:81f::200a
* Transmission Control Protocol, Src Port: 8083, Dst Port: 443, Seq: 2378, Ack: 597, Len: 479
  Source Port: 8083
  Destination Port: 443
  <Source or Destination Port: 8083>
  <Source or Destination Port: 443>
  [Stream index: 7]
  [Conversation completeness: Incomplete (12)]
  [TCP Segment Len: 479]
  Sequence Number: 2378 (relative sequence number)
  Sequence Number (raw): 3323866800
  [Next Sequence Number: 2857 (relative sequence number)]
  [Acknowledgment Number: 507 (relative seq. number)]

```

6) Analyse the application workflow and observe which all files/folders are being used by the application using Process Monitor

- With the help of a tool called Process-Monitor can see that there are several files & folders being retrieved when running the DVTA.exe.

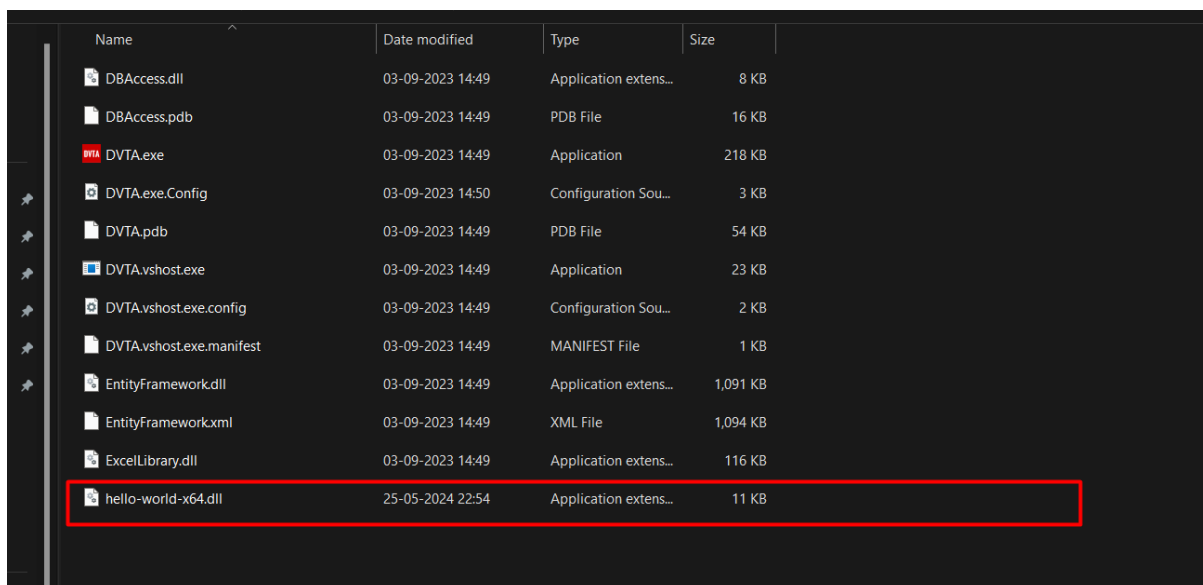


The screenshot shows the Process Monitor application window with the following data:

Time o...	Process Name	PID	Operation	Path	Result	Detail
22:51:08...	MsMpEng.exe	5900	CloseFile	C:\Windows\System32\sihost.exe	SUCCESS	
22:51:08...	Code.exe	16116	TCP TCPCopy	LAPTOP-ABCLFDJP:8543 -> 2a04:4e42...	SUCCESS	Length: 2740, seqn...
22:51:08...	Code.exe	16116	TCP Receive	LAPTOP-ABCLFDJP:8543 -> 2a04:4e42...	SUCCESS	Length: 2740, seqn...
22:51:08...	dumpcap.exe	17768	WriteFile	C:\Temp\wireshark_Wi-FiX5UQ02.pca...	SUCCESS	Offset 164500328, L...
22:51:08...	DVTA.exe	17572	Thread Create		SUCCESS	Thread ID: 17312
22:51:08...	DVTA.exe	17572	Thread Create		SUCCESS	Thread ID: 13696

7) Exploit DLL Hijacking vulnerability (You can use a simple legitimate “Hello World” printing dll.

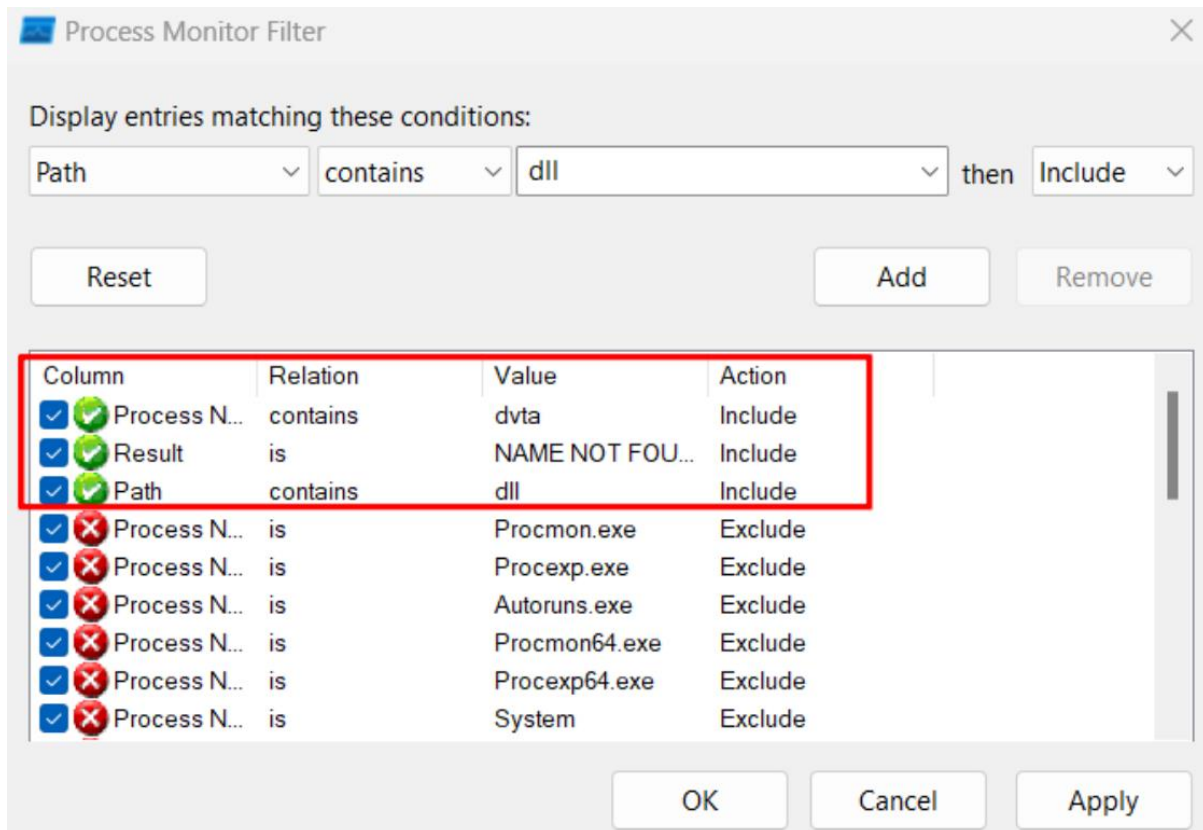
- In order to hijack a DLL, we need to find which DLL’s that are being loaded when DVTA.exe runs is not found.
- For this we need to open Procmon & set the following 3 filters .



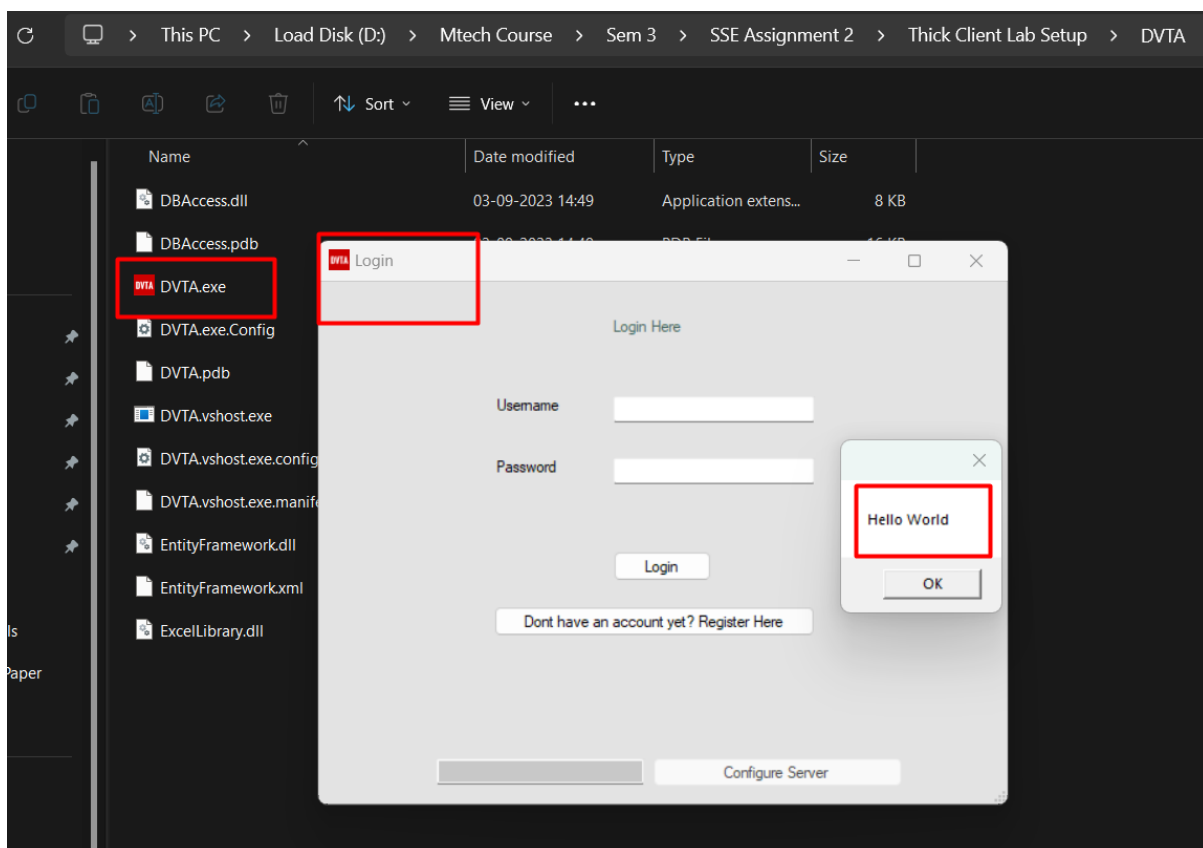
The screenshot shows a File Explorer window with the following data:

Name	Date modified	Type	Size
DBAccess.dll	03-09-2023 14:49	Application extens...	8 KB
DBAccess.pdb	03-09-2023 14:49	PDB File	16 KB
DVTA.exe	03-09-2023 14:49	Application	218 KB
DVTA.exe.Config	03-09-2023 14:50	Configuration Sou...	3 KB
DVTA.pdb	03-09-2023 14:49	PDB File	54 KB
DVTA.vshost.exe	03-09-2023 14:49	Application	23 KB
DVTA.vshost.exe.config	03-09-2023 14:49	Configuration Sou...	2 KB
DVTA.vshost.exe.manifest	03-09-2023 14:49	MANIFEST File	1 KB
EntityFramework.dll	03-09-2023 14:49	Application extens...	1,091 KB
EntityFramework.xml	03-09-2023 14:49	XML File	1,094 KB
ExcelLibrary.dll	03-09-2023 14:49	Application extens...	116 KB
hello-world-x64.dll	25-05-2024 22:54	Application extens...	11 KB

- We will Start Process Monitor Filter



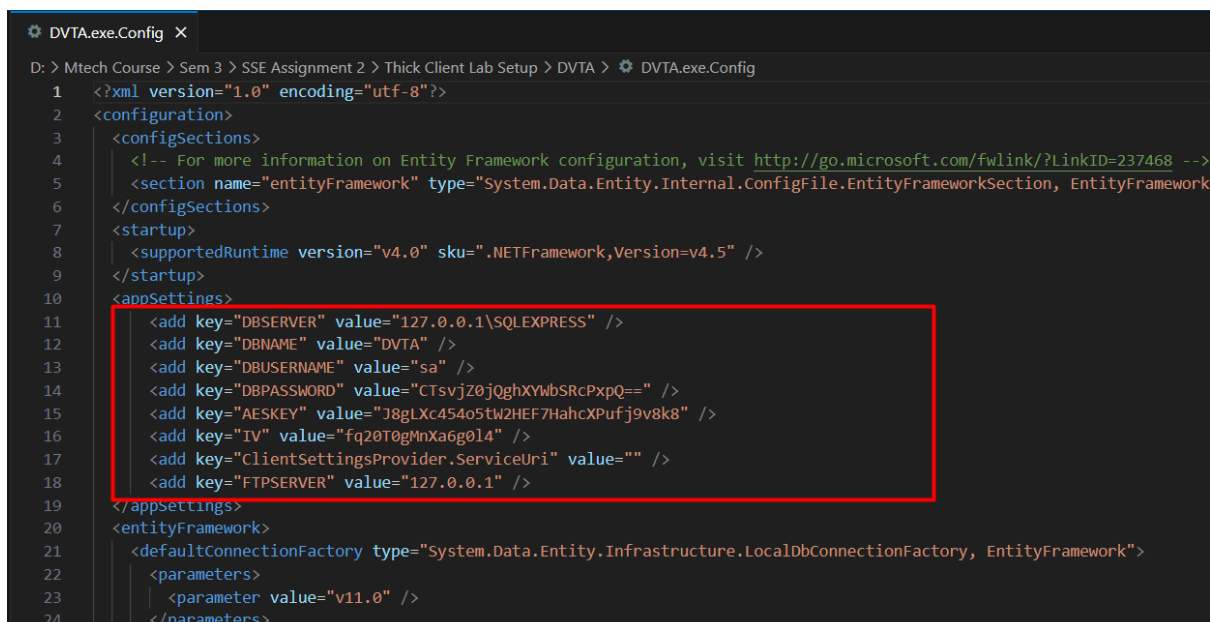
- When click DVTA.exe automatically Hello world pop up will appear with opening of DVTA Login Page



- As we can see now when the DVTA.exe runs, it loads our calc.dll along with the application. Thus we have hijacked the DLL.

8) Check for sensitive information in the configuration files of the thick client application?

- In the folder of DVTA, we have few files . One of the files is App.config. It contains the following sensitive information.
- We have to open Visual Studio and analyse DVTA.exe.config.



```

D:\> Mtech Course > Sem 3 > SSE Assignment 2 > Thick Client Lab Setup > DVTA > DVTA.exe.Config
1  <?xml version="1.0" encoding="utf-8"?>
2  <configuration>
3    <configSections>
4      <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237468 -->
5      <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework" />
6    </configSections>
7    <startup>
8      <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
9    </startup>
10   <appSettings>
11     <add key="DBSERVER" value="127.0.0.1\SQLEXPRESS" />
12     <add key="DBNAME" value="DVTA" />
13     <add key="DBUSERNAME" value="sa" />
14     <add key="DBPASSWORD" value="CTsvjZ0jQghXYWbSRcPxpQ==" />
15     <add key="AESKEY" value="J8gLXc454o5tW2HEF7HahcXPufj9v8k8" />
16     <add key="IV" value="fq20T0gMnXa6g014" />
17     <add key="ClientSettingsProvider.ServiceUri" value="" />
18     <add key="FTPSERVER" value="127.0.0.1" />
19   </appSettings>
20   <entityFramework>
21     <defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">
22       <parameters>
23         <parameter value="v11.0" />
24       </parameters>

```

9) Identify sensitive information found in memory?

- From the source code which we got from DNSpy, we got to know that it stores the username & password in HKCU/dvta registry file.
- We can visit the registry to find the sensitive information which is stored in the memory.
- We have to open registry editor to analyse dvta username and password.

