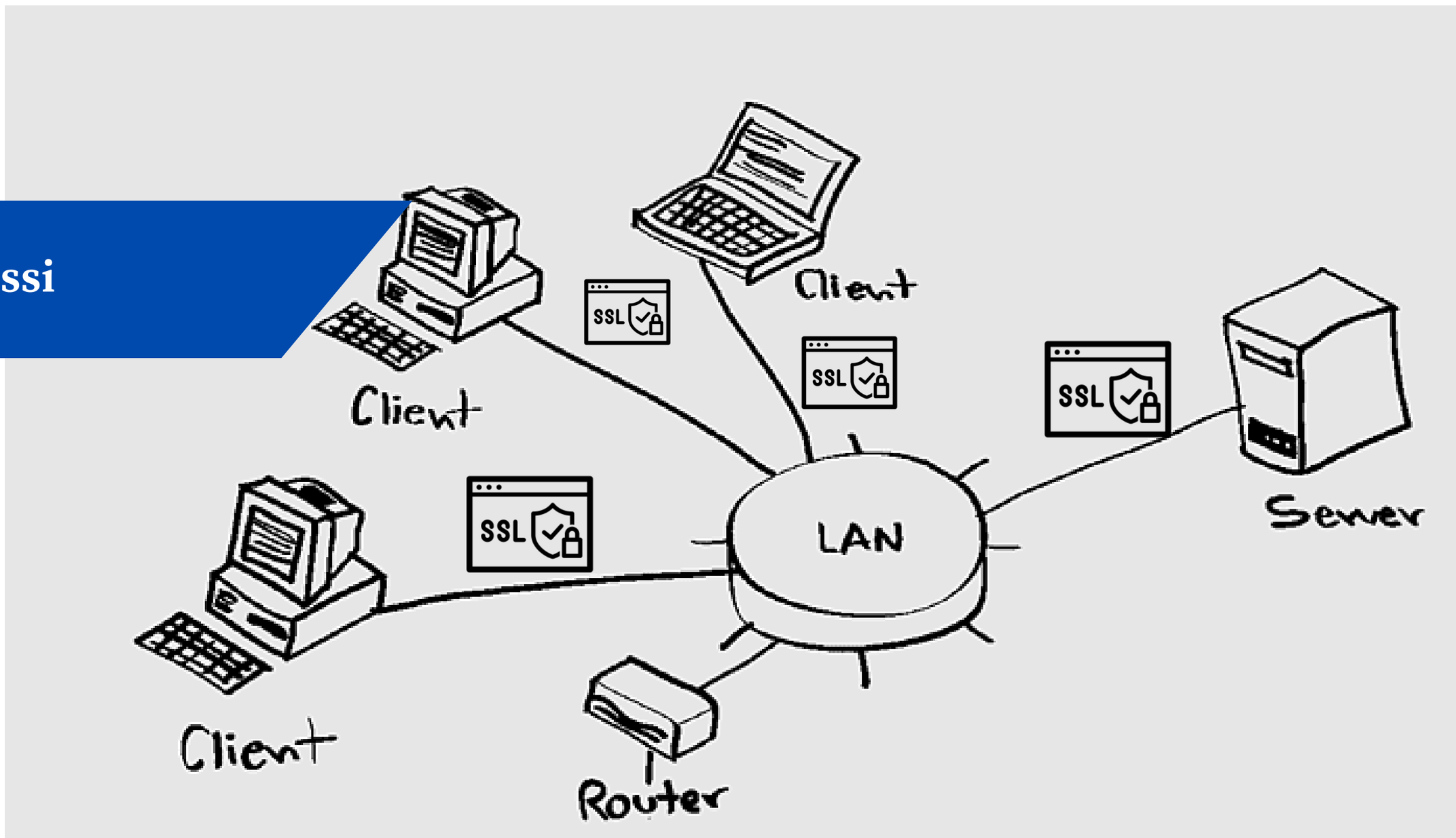
Implementation of client & server network in localhost with SSL in python

Raed Hermessi



Index

Introduction

SSL, or Secure Sockets Layer, is an encryption-based Internet security protocol. It was first developed by Netscape in 1995 for the purpose of ensuring privacy, authentication, and data integrity in Internet communications. SSL is the predecessor to the modern TLS encryption used today.
A website that implements SSL/TLS has "HTTPS" in its URL instead of "HTTP."

In this playbook, we're going to dive into the essentials of setting up and using SSL/TLS certificates on Windows, a critical skill for anyone working with secure communications. We'll start with installing OpenSSL, an indispensable tool for generating and managing certificates, and walk through creating essential file types such as .key, .crt, and .pem files. Next, we'll add our certificate to the Windows Trusted Certificates store, ensuring it's recognized by your system. Finally, we'll explore how to use Wireshark to inspect packet flows, allowing you to observe the secure data exchange in real-time and verify that your setup works as intended. Whether you're securing a web server, creating secure client-server applications, or just learning, this guide will provide a hands-on approach to mastering SSL/TLS basics .

Installing OpenSSL

We'll be using OpenSSL Light, a lightweight version of OpenSSL for Windows .
Start by downloading it from https://slproweb.com/products/Win32OpenSSL.html

Download Win32/Win64 OpenSSL

Download Win32/Win64 OpenSSL today using the links below!

File	Type	Description
Win64 OpenSSL v3.4.0 Light EXE | MSI	5MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.4.0 (Recommended for users by the creators of OpenSSL). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.4.0 EXE | MSI	221MB Installer	Installs Win64 OpenSSL v3.4.0 (Recommended for software developers by the creators of OpenSSL). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v3.4.0 Light EXE | MSI	4MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v3.4.0 (Only install this if you need 32-bit OpenSSL for Windows). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v3.4.0 EXE | MSI	180MB Installer	Installs Win32 OpenSSL v3.4.0 (Only install this if you need 32-bit OpenSSL for Windows). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.4.0 Light for ARM (EXPERIMENTAL) EXE | MSI	6MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.4.0 for ARM64 devices (Only install this VERY EXPERIMENTAL build if you want to try 64-bit OpenSSL for Windows on ARM processors). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.4.0 for ARM (EXPERIMENTAL) EXE | MSI	176MB Installer	Installs Win64 OpenSSL v3.4.0 for ARM64 devices (Only install this VERY EXPERIMENTAL build if you want to try 64-bit OpenSSL for Windows on ARM processors). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.

Once downloaded, run the installer and follow the on-screen instructions.
Be sure to select the option to add OpenSSL to your system's PATH during installation, which allows you to run OpenSSL commands from any command prompt window.
After installation, open a command prompt and type openssl version **to confirm it's installed correctly.**

```
Win64 OpenSSL Command Pr

Win64 OpenSSL Command Prompt

OpenSSL 3.4.0 22 Oct 2024 (Library: OpenSSL 3.4.0 22 Oct 2024)
built on: Tue Oct 22 23:27:41 2024 UTC
platform: VC-WIN64A
options:  bn(64,64)
compiler: cl  /Z7 /Fdossl_static.pdb /Gs0 /GF /Gy /MD /W3 /wd4090 /nologo /O2 -DL_ENDIAN -DOPENSSL_PIC -D"OPENSSL_BUILDI
NG_OPENSSL" -D"OPENSSL_SYS_WIN32" -D"WIN32_LEAN_AND_MEAN" -D"UNICODE" -D"_UNICODE" -D"_CRT_SECURE_NO_DEPRECATE" -D"_WINS
OCK_DEPRECATED_NO_WARNINGS" -D"NDEBUG" -D_WINSOCK_DEPRECATED_NO_WARNINGS -D_WIN32_WINNT=0x0502
OPENSSLDIR: "C:\Program Files\Common Files\SSL"
ENGINESDIR: "C:\Program Files\OpenSSL\lib\engines-3"
MODULESDIR: "C:\Program Files\OpenSSL\lib\ossl-modules"
Seeding source: os-specific
CPUINFO: OPENSSL_ia32cap=0xfffaf38fffebffff:0x18405fc6f3bfa7a9

C:\Users\THINKPAD>openssl version
OpenSSL 3.4.0 22 Oct 2024 (Library: OpenSSL 3.4.0 22 Oct 2024)

C:\Users\THINKPAD>
```

Create .key , .crt , .pem files

In this section, we will be creating the required .key, .crt, and .pem files using a few simple OpenSSL commands. Here's a breakdown:

note : i have create a folder Keys in my Desktop , to store the whole porject

Generate a Private Key (.key file):

```
C:\WINDOWS\system32\cmd.

Microsoft Windows [Version 10.0.22631.4391]
(c) Microsoft Corporation. All rights reserved.

C:\Users\THINKPAD>cd Desktop

C:\Users\THINKPAD\Desktop>mkdir KEYS

C:\Users\THINKPAD\Desktop>cd KEYS

C:\Users\THINKPAD\Desktop\KEYS>openssl genrsa -aes256 -out private.key 2048
Enter PEM pass phrase:

Verifying - Enter PEM pass phrase:


C:\Users\THINKPAD\Desktop\KEYS>
```

This command creates a 2048-bit encrypted private key file.
we can check that our file has been created in our Directory .

```
private.key
C: > Users > THINKPAD > Desktop > KEYS > private.key
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFNTBfBgkqhkiG9w0BBQ0wUjAxBgkqhkiG9w0BBQwwJAQQlvReVuLFZx8dCqAk
uAxfKgICCAAwDAYIKoZIhvcNAgkFADAdBglghkgBZQMEASoEENUGi+clY1BccIAd
6j6xYV8EggTQIbWn3FrUiPtA622b953E/RbAoc8IQCBtpP/QCXcMqN+a/c0EUrXJ
+zAkQcedazjRs5wIY5riKvslmXr6v+eXTX3vlzyNJNpGKMnVa52ULLHLbKbZ293B
Mv4zXtY4roMbM3Obbfu2iUsZ48puI5emAezDGf9Hy2GGWhebMpZAgmGga7pXNCX5
oOZdJzVsrcF0ImD120HMZABK6TVzq7odqeD0yIVWWuxpMdU4c1e5LLmx8Fl80xGF
Iib0QPU2867CEPqqrY679ybvNBje1WCRzN/MlmtUayczcb8OVbOBdqMGIyeWQYuF
dqO5dCvDAh3M3FkxcWLhdxrLLrhrYpGyfpEnyph8gcRj6aWZ++3k8rF8C7VNYmYm
KfEICZclFGun3BHom/g+kygO3Tqb3crRHXEbz6fP+xQ/5sBFITkoCWCIuwU39RND
8aSAgwH2FYOVsKvx2Iedj7xgtrfk/gEPSElhbE3Iu+ke2JiPLr/jgRzRPzzknp0f
SQ9VAv94FrEu3vmnFrLAeJYWymZWMX9u5yJedRAGe96G+RVrJ9EigChwFT3+bnkN
VEytKynlw02Vn3FknvnMSXyPPEDm6bgA7+k5FcQ1VaR0BFGD5eJ/Cv64BPffVvTV
7cMk0l1EW7G6Tfk64L2DKorRF38U3x9WyIaPbtNLAQpQZ7JWPJgS43eeLFb37MLE
YkV75xo5E2Pmu4kpgpOi2przPfWZUbZRnDljoK8STmRmxFUwmG9w3JqVXyJWtrMt
vymDcv8EOtR8ExnvyYsmiAhd0OSCNxffcAa9ajirR3WstMmuUeKAlhb3R9fBCOwV
9UscoAsPBJOrTvCkOsMsgv8B9j5x6/2HeLyzBMGic0kshDE1JUp/5d3FPNKDTMxF
nYRMv+AYuMicdQgdvUDTIIPaaa7WLucZOBACAMlqLPbZOjqA2uiJfOgMhzCZmoiZ
50s9o3a3QGHaJBSQhX6Myz+WLSFi5osjrktyPdlKN19qEMbqEfDq3BbpTQ0JOLm0
vIKwXLny6stVndYAUtkWWOjE413GIZTo3us0Y7NeWtYpxEJ8aLwVKiHPaEc4U/qL
iO7KKB9DnLkdOPI2s9QLDn6ZBrE65Jr7Czv88xAf9gDIzowZTxhww1zw6AAWd4sI
FzVPHvs4rlprPG0aZ5jrepFkG2WMk/aErIWpvhLiGgirlE5s0kkmacw1xXfy8Sov
i99amyhkLXeB2uUZa5q8c6iI8leE8KiokkQ0aXQijLBZEFEybPAVdJaYmuIWpWpk
qUrfdIIgEZdnnx6dS8YfVZLvK7Vvk2zvViNjppp+jCAUAkwIkDAd06Q11i7nD/Mj
Sq37heMXg4/L2XoEaUnCpsvxbear2qvCacrNMM3/fogoW+Gx9vV9DcDFq6Tm/H5k
FxvWDP19jktYpnM79o/ziWYWhbOp10ZSidUdVD8YG+wKY+Uz18g7IvspCv+hRNGh
7qQe7x+7GmuYOalOSfz0hcduCAdiowYyeRGZcpkGXoZJMVmATOINqHtNKnb5l3IS
cBVn042JPtVD2BcsMRfleB3RpTrI1Q0f5XmkKPefsaD9H2c7w8VaCjJtyCFG6Ffg
WQBeGkND7vUfW30woh5wrKRY5gsf7uOBKQQGlECmo9yutg34nlT0WLM=
-----END ENCRYPTED PRIVATE KEY-----
```

Remove Passphrase from Private Key:

```
C:\WINDOWS\system32\cmd.

C:\Users\THINKPAD\Desktop\KEYS>openssl rsa -in private.key -out private.key
Enter pass phrase for private.key:

writing RSA key

C:\Users\THINKPAD\Desktop\KEYS>
```

note : use the same passphrase used in the previous step , you can use any password (ex : 1234 , abcd)

Generate a Certificate (.crt file):

```
C:\WINDOWS\system32\cmd.

C:\Users\THINKPAD\Desktop\KEYS>openssl req -new -x509 -nodes -sha1 -key private.key -out certificate.crt -days 36500
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:localhost
Email Address []:

C:\Users\THINKPAD\Desktop\KEYS>
```

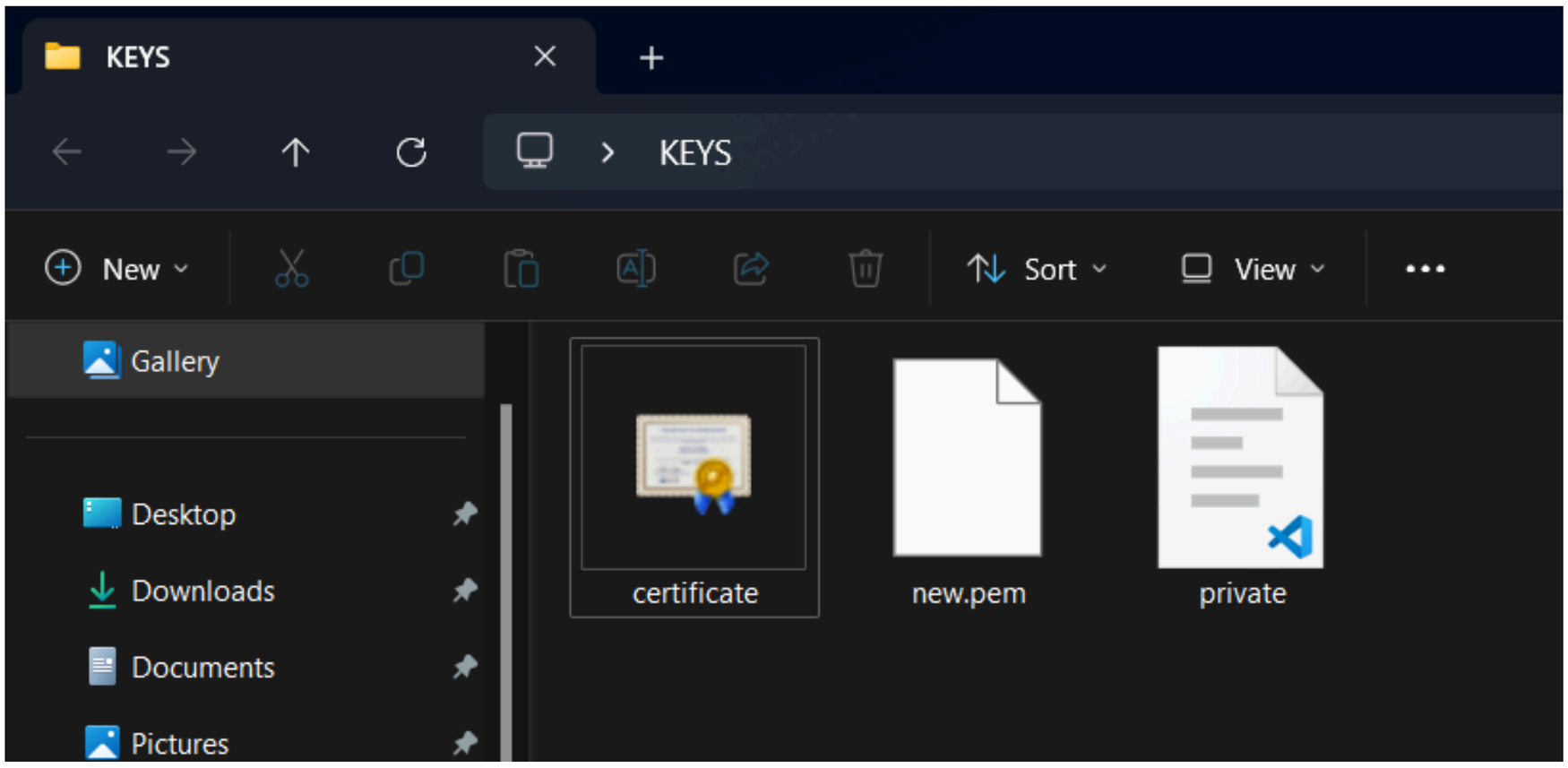
This configuration keeps all settings at their default values, except for the Common Name, which is set to localhost. This is used for local development, where the SSL certificate is valid for your local server (localhost). Other fields like Country, State, and Email are left as default or blank.

Create a .pem file:

```
C:\Users\THINKPAD\Desktop\KEYS>openssl req -x509 -new -nodes -key private.key -sha1 -days 36500 -out new.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:localhost
Email Address []:

C:\Users\THINKPAD\Desktop\KEYS>
```

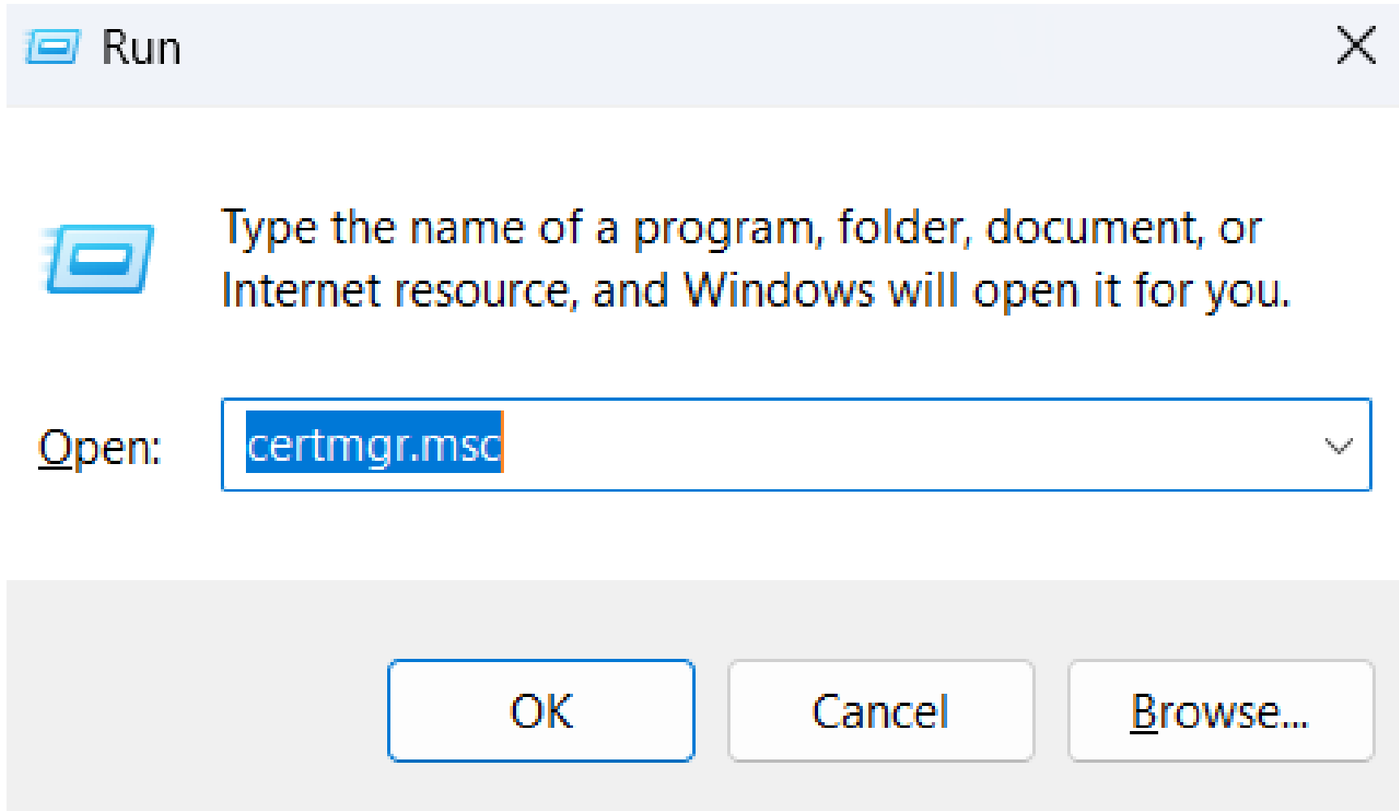
This command generates a .pem file, commonly used for combined certificate and key storage. use the same configuration used in the previous step.

These files are essential for SSL/TLS certificate management and will be useful for secure communication setups.

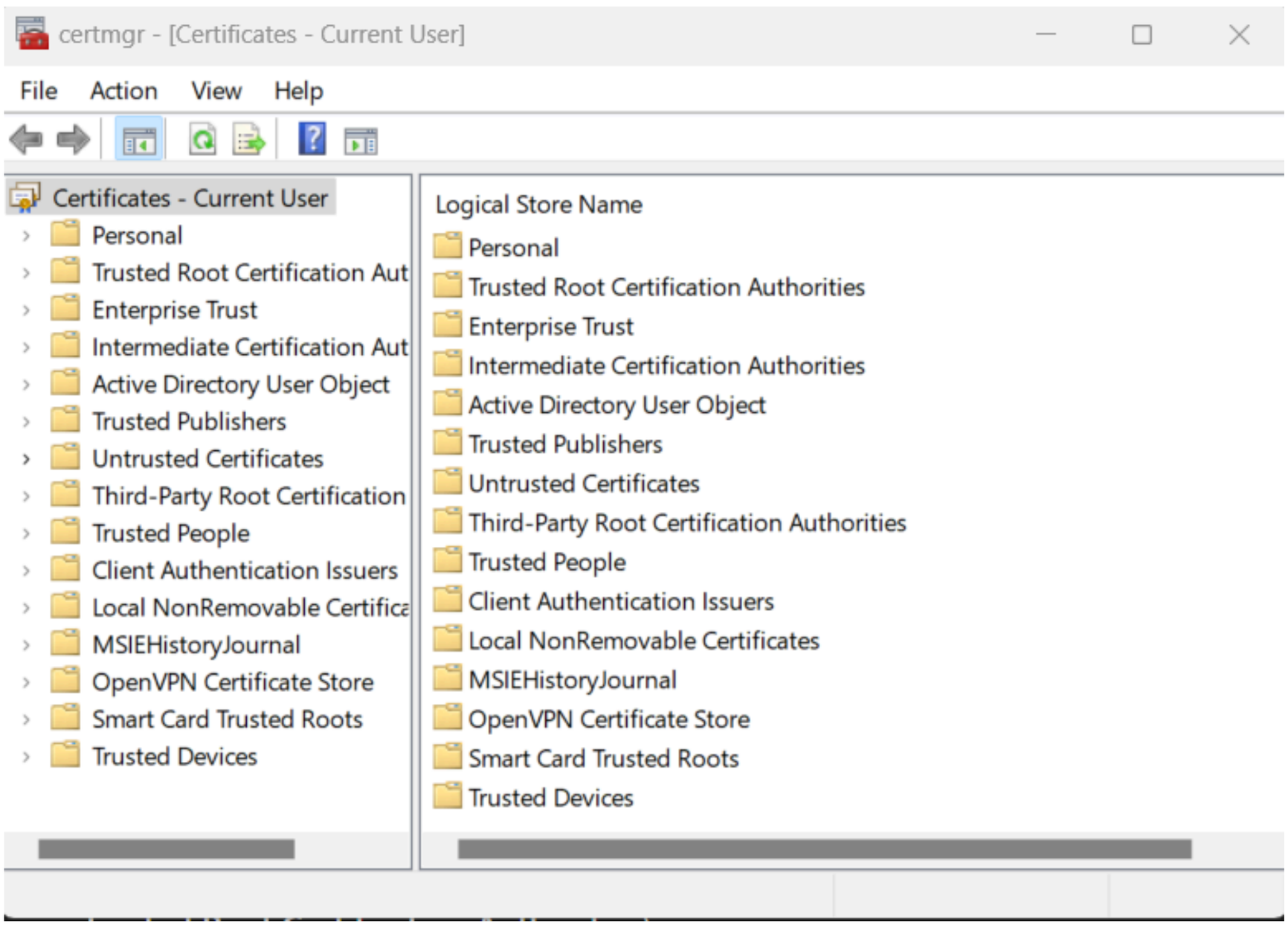


we can check that everything looks great to this point .

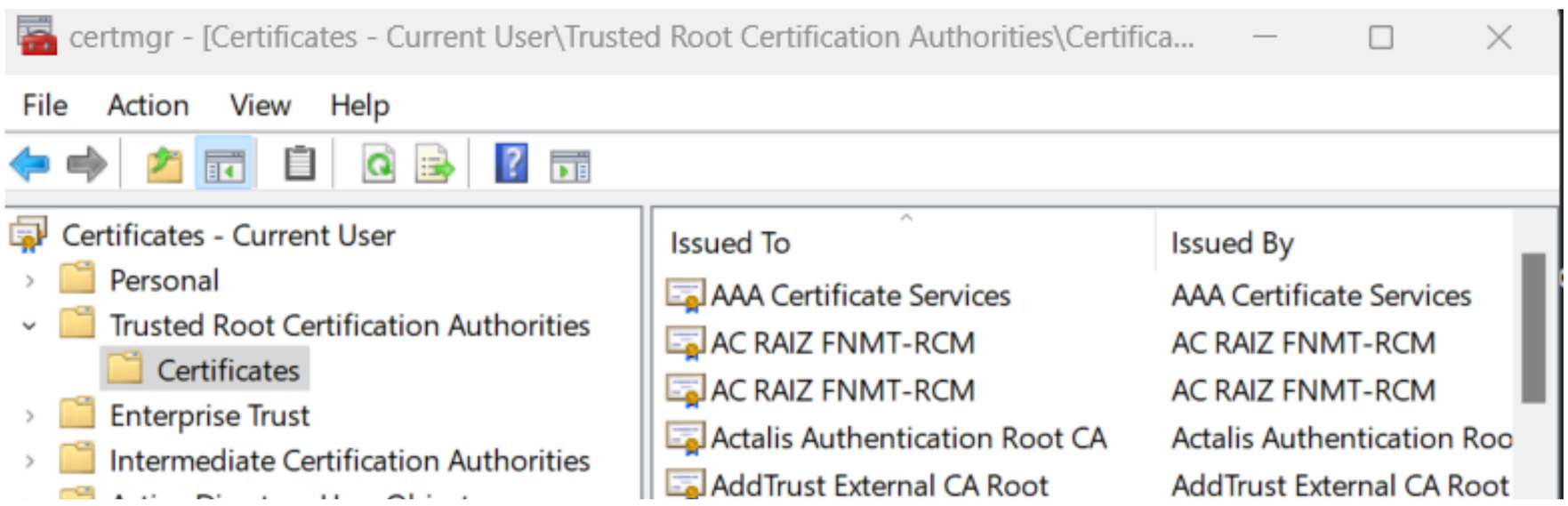
Add our Certificate file to our windows Cluster circuit :



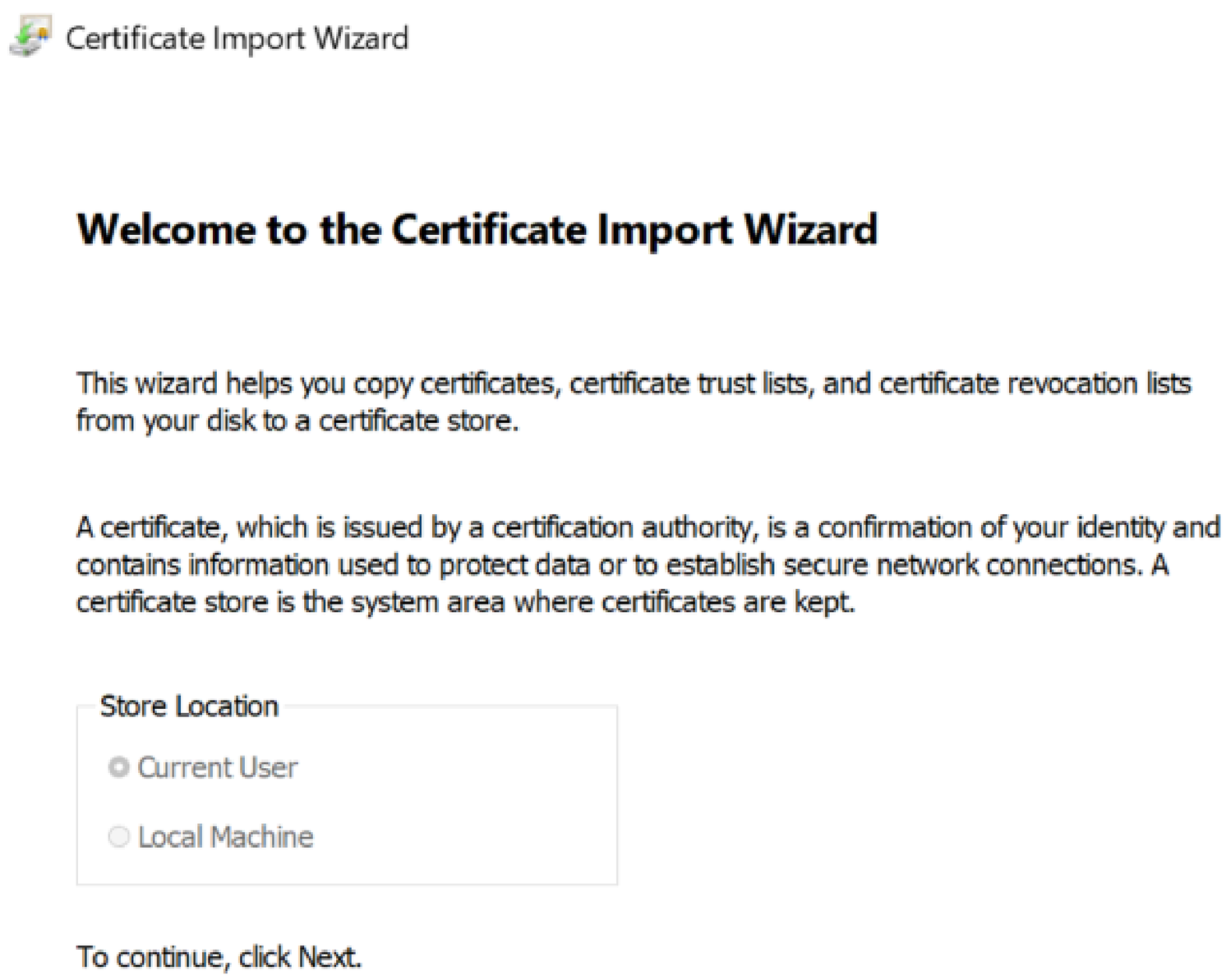
Press Windows + R, type certmgr.msc, and press Enter to open the Certificate Manager. Alternatively, press the Windows key, type Manage computer certificates, and select it.



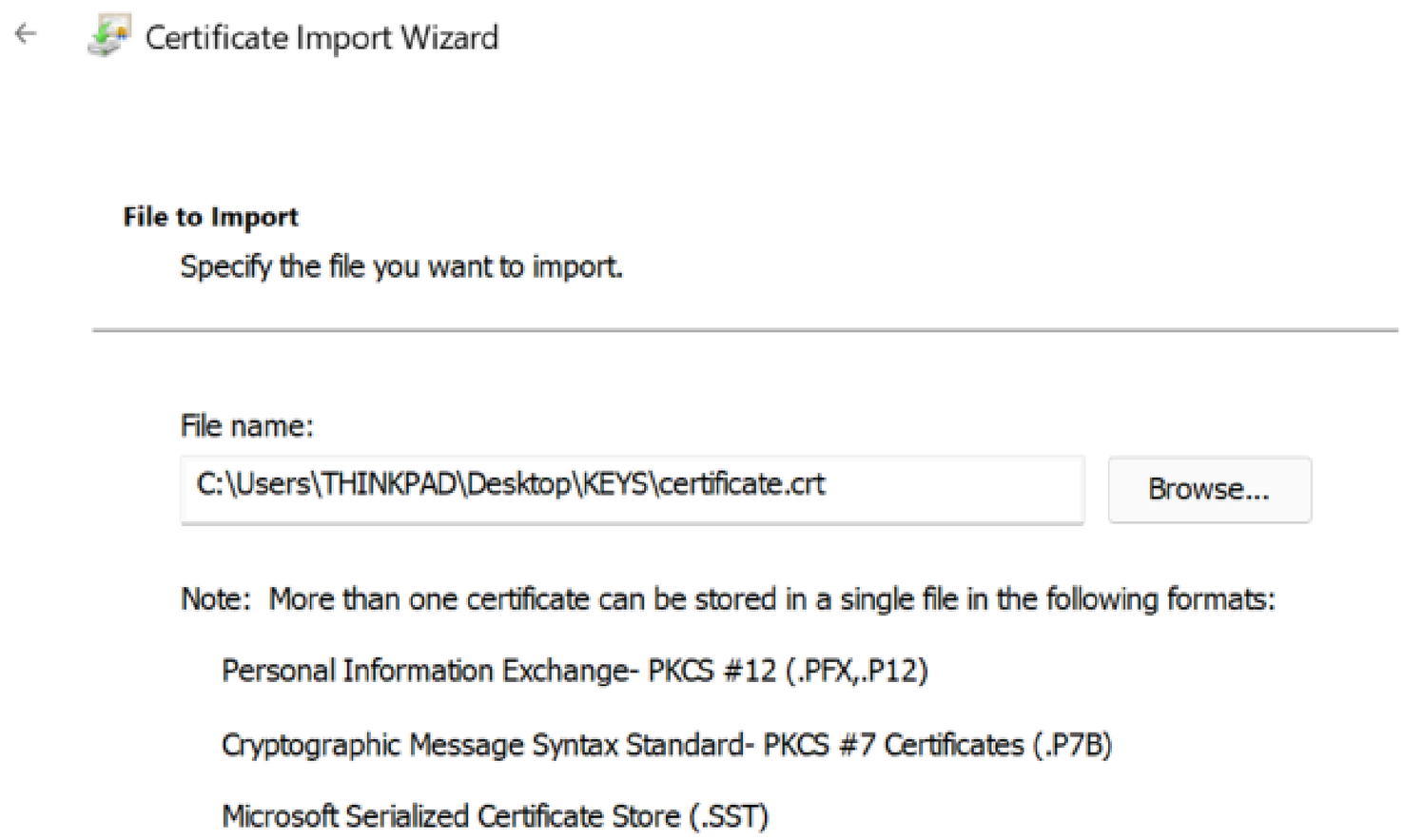
next head to Trusted Root Certification Authorities > Certificates



next step click on Action > All tasks > import and import our Certificate file that we created



we just click next ,



and we're done . now we need to set up our server and our client program .

Server/Client program

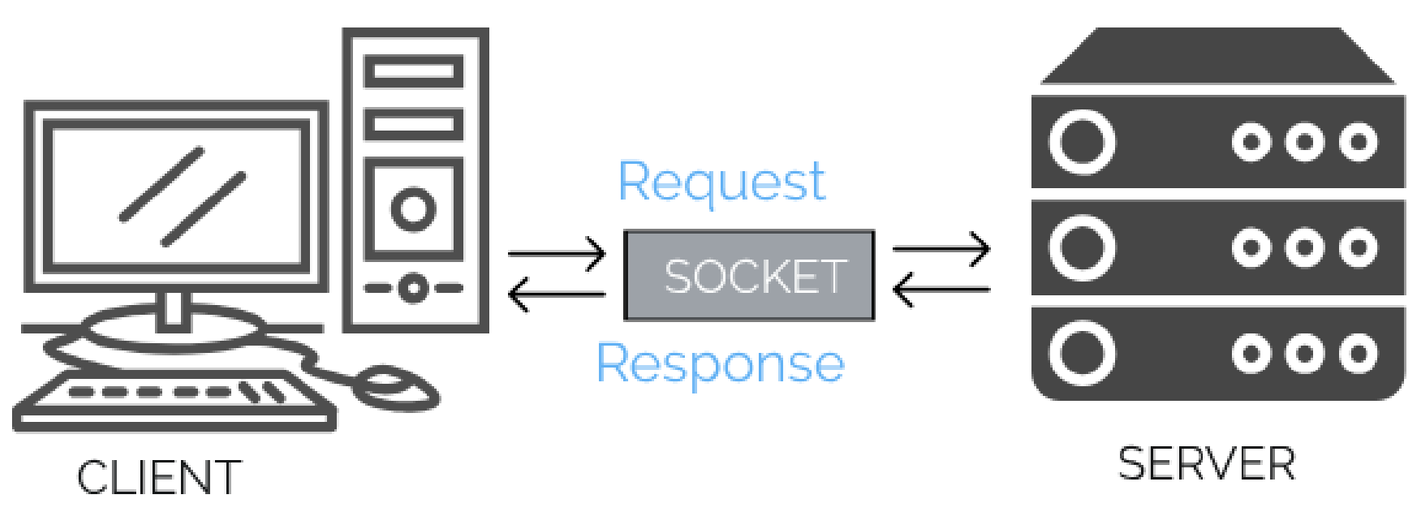
Python Libraries Required :

- **import socket**
- **import ssl**

Socket Code Explanation :

Socket Creation	→	socket.socket()
Bind the Socket	→	bind((<domain name> , <port number>)) # we need to pass tuple
Listen to the Socket	→	listen(<number of clients to accept>) “ we need to pass integer
Connecting to the Socket	→	connect((<domain name> , <port number>)) # we need to pass tuple

A socket is an endpoint for communication between two programs on a network, enabling inter-process communication (IPC). It is created using the ‘socket’ system call and provides bidirectional FIFO communication over the network. Each socket has a unique address, made up of an IP address and a port number. Sockets are commonly used in client-server applications, where the server creates a socket, binds it to a network address, and waits for client connections. The client creates a socket and connects to the server, allowing data transfer once the connection is established.



SSL Code Explanation :

SSL context creation	SSLContext(PROTOCOL_TLS_SERVER)
Loading key files in SSL obj.	load_cert_chain(<.pem> , <.key>) # we need to pass .pem , .key file path
Wrapping SSL obj with socket	wrap_socket(<socket obj> , server_side=True) # we need to pass socket obj.

Server.py & client.py :

server.py client.py

C: > Users > THINKPAD > Desktop > Keys > client.py > ...

```python
import socket
import ssl
import optparse

parser = optparse.OptionParser('usage%prog' + '-d <domain> '+'-p <port> ')
parser.add_option('-d' , dest='domain' , type='string' , help='specify the method')
parser.add_option('-p' , dest='port' , type='string' , help='specify the url')

options , args = parser.parse_args()
domain = str(options.domain)
port = int(options.port)


def get_secret_message():

    context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
    context.load_verify_locations('new.pem')

    soc = socket.socket()
    c_soc = context.wrap_socket(soc , server_hostname=domain)
    c_soc.connect((domain , port))
    print(f"[+] connection successful ...")
    msg = c_soc.recv(1024)
    print(msg.decode("utf-8"))
    c_soc.close()

if __name__ == '__main__':
    get_secret_message()
```

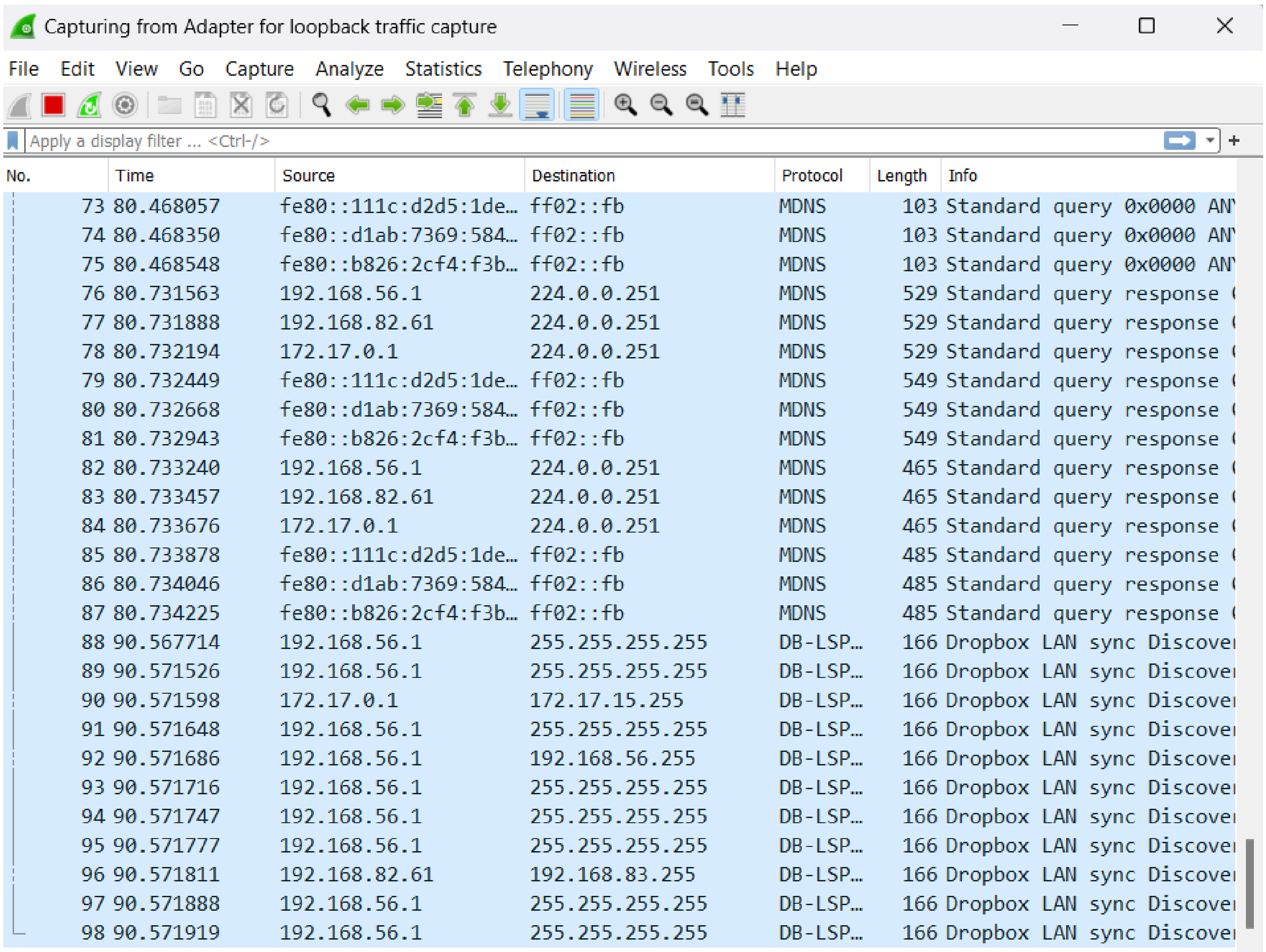
server.py

C: > Users > THINKPAD > Desktop > Keys > server.py > ...

```python
import socket
import ssl


host = "localhost"
port = 4444

def server_connection():

    context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
    context.load_cert_chain('new.pem', 'private.key')


    soc = socket.socket()
    soc.bind((host , port))
    print(f"[+] server is runnning at {host} with {port} .")
    print(f"[+] server is ready to accept requests ... ")
    soc.listen(3)

    s_soc = context.wrap_socket(soc , server_side=True)
    connection , address = s_soc.accept()

    print(f"[+] server is connected to {address}")
    connection.send(bytes(f"[+] welcome to server ({host} ,{port}) ...", 'utf-8'))
    s_soc.close()


if __name__ == '__main__':
    server_connection()
```

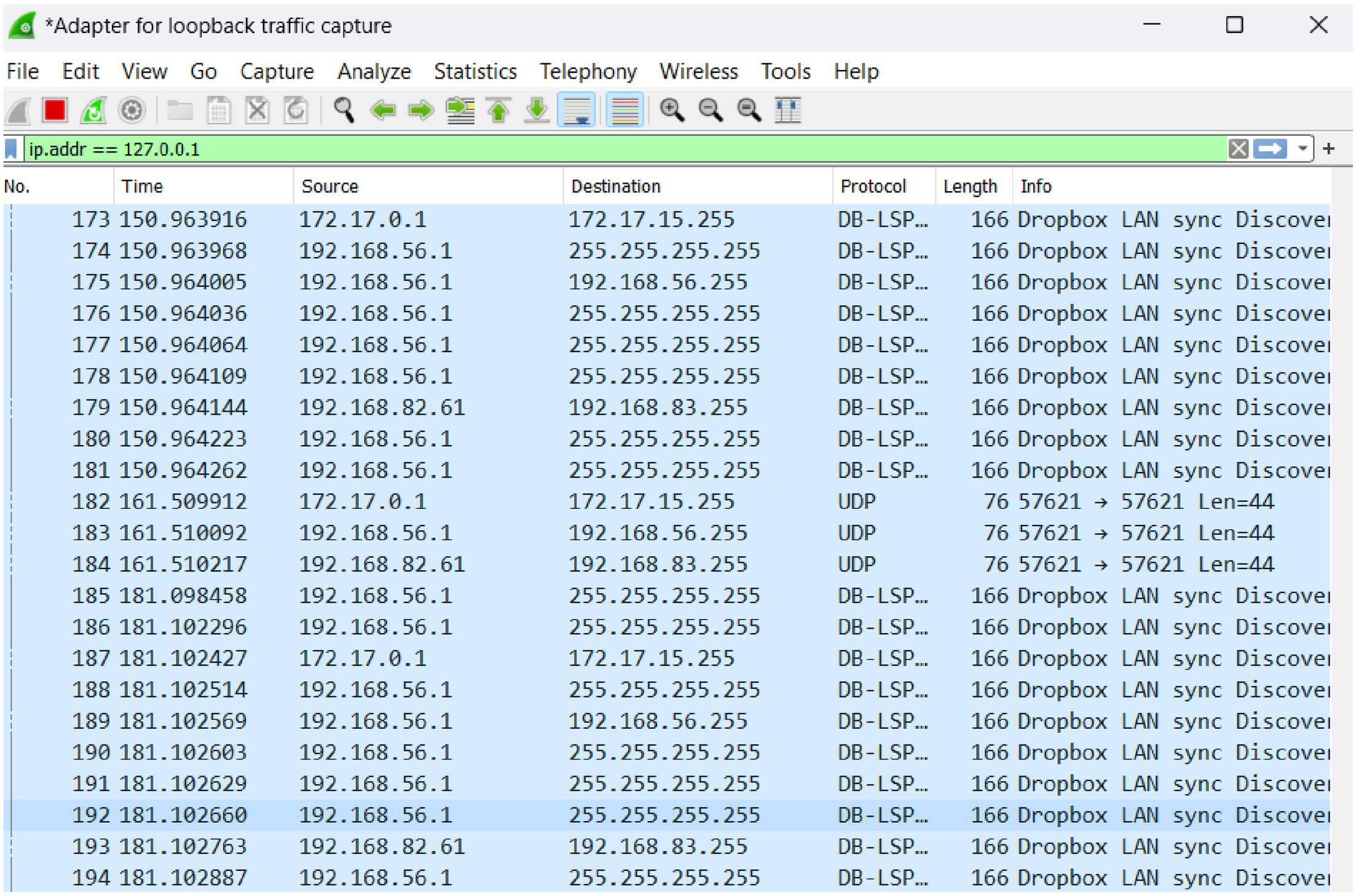
this is the full code of our Server and Client program , am not gonna go dive more into explaining each line of code , so i will keep it into a next Playbook about sockets .

now , we simply run our programs , but first let's open our Wireshark and head to the loopback packet capture . Loopback packets are sent to the IP address 127.0.0.1, which routes them back to the same computer. This is used for testing the computer's internal network functions without involving external networks.

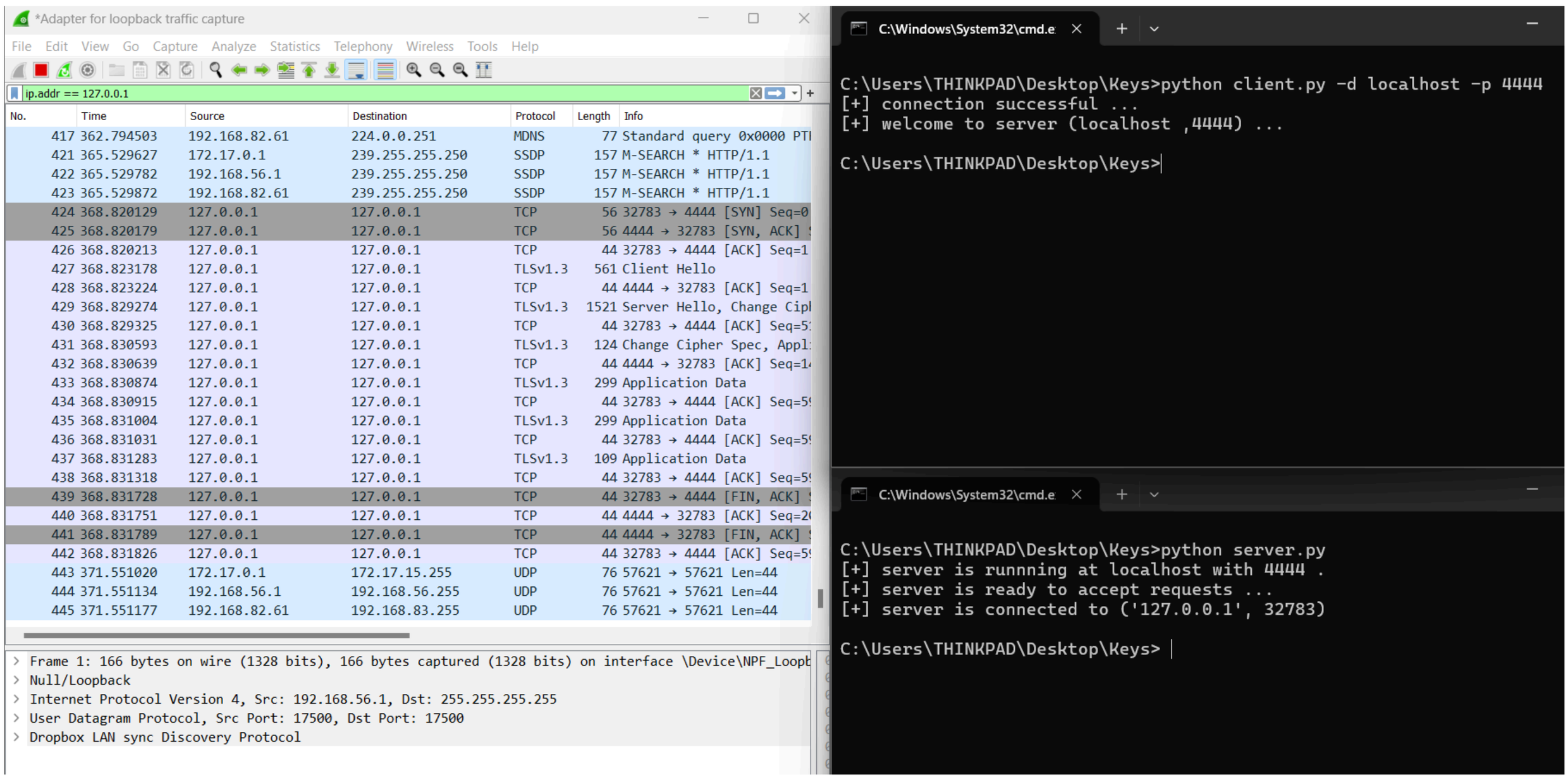


No.	Time	Source	Destination	Protocol	Length	Info
73	80.468057	fe80::111c:d2d5:1de…	ff02::fb	MDNS	103	Standard query 0x0000 AN
74	80.468350	fe80::d1ab:7369:584…	ff02::fb	MDNS	103	Standard query 0x0000 AN
75	80.468548	fe80::b826:2cf4:f3b…	ff02::fb	MDNS	103	Standard query 0x0000 AN
76	80.731563	192.168.56.1	224.0.0.251	MDNS	529	Standard query response
77	80.731888	192.168.82.61	224.0.0.251	MDNS	529	Standard query response
78	80.732194	172.17.0.1	224.0.0.251	MDNS	529	Standard query response
79	80.732449	fe80::111c:d2d5:1de…	ff02::fb	MDNS	549	Standard query response
80	80.732668	fe80::d1ab:7369:584…	ff02::fb	MDNS	549	Standard query response
81	80.732943	fe80::b826:2cf4:f3b…	ff02::fb	MDNS	549	Standard query response
82	80.733240	192.168.56.1	224.0.0.251	MDNS	465	Standard query response
83	80.733457	192.168.82.61	224.0.0.251	MDNS	465	Standard query response
84	80.733676	172.17.0.1	224.0.0.251	MDNS	465	Standard query response
85	80.733878	fe80::111c:d2d5:1de…	ff02::fb	MDNS	485	Standard query response
86	80.734046	fe80::d1ab:7369:584…	ff02::fb	MDNS	485	Standard query response
87	80.734225	fe80::b826:2cf4:f3b…	ff02::fb	MDNS	485	Standard query response
88	90.567714	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove
89	90.571526	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove
90	90.571598	172.17.0.1	172.17.15.255	DB-LSP…	166	Dropbox LAN sync Discove
91	90.571648	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove
92	90.571686	192.168.56.1	192.168.56.255	DB-LSP…	166	Dropbox LAN sync Discove
93	90.571716	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove
94	90.571747	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove
95	90.571777	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove
96	90.571811	192.168.82.61	192.168.83.255	DB-LSP…	166	Dropbox LAN sync Discove
97	90.571888	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove
98	90.571919	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove

since our server is configurated on the localhost , in the wireshark tab , tap ip.addr==127.0.0.1 , to see our traffic while we execute our server/client code .



No.	Time	Source	Destination	Protocol	Length	Info
173	150.963916	172.17.0.1	172.17.15.255	DB-LSP…	166	Dropbox LAN sync Discove
174	150.963968	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove
175	150.964005	192.168.56.1	192.168.56.255	DB-LSP…	166	Dropbox LAN sync Discove
176	150.964036	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove
177	150.964064	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove
178	150.964109	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove
179	150.964144	192.168.82.61	192.168.83.255	DB-LSP…	166	Dropbox LAN sync Discove
180	150.964223	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove
181	150.964262	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove
182	161.509912	172.17.0.1	172.17.15.255	UDP	76	57621 → 57621 Len=44
183	161.510092	192.168.56.1	192.168.56.255	UDP	76	57621 → 57621 Len=44
184	161.510217	192.168.82.61	192.168.83.255	UDP	76	57621 → 57621 Len=44
185	181.098458	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove
186	181.102296	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove
187	181.102427	172.17.0.1	172.17.15.255	DB-LSP…	166	Dropbox LAN sync Discove
188	181.102514	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove
189	181.102569	192.168.56.1	192.168.56.255	DB-LSP…	166	Dropbox LAN sync Discove
190	181.102603	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove
191	181.102629	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove
192	181.102660	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove
193	181.102763	192.168.82.61	192.168.83.255	DB-LSP…	166	Dropbox LAN sync Discove
194	181.102887	192.168.56.1	255.255.255.255	DB-LSP…	166	Dropbox LAN sync Discove



and done , we can see our captured packets , the server is working and our client is responding .