**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No** | **Name of the practical** | **Date** | **Signature** |
| **1** | **Implementing Substitution and Transposition Ciphers:** Design and implement algorithms to encrypt and decrypt messages using classical substitution and transposition techniques. |  |  |
| **2** | **RSA Encryption and Decryption:** Implement the RSA algorithm for public-key encryption and decryption, and explore its properties and security considerations. |  |  |
| **3** | **Message Authentication Codes:** Implement algorithm to generate and verify message authentication codes (MACs) for ensuring data integrity and authenticity |  |  |
| **4** | **Digital Signatures:** Implement digital signature algorithms such as RSA-based signatures, and verify the integrity and authenticity of digitally signed messages. |  |  |
| **5** | **Key Exchange using Diffie-Hellman:** Implement the Diffie-Hellman key exchange algorithm to securely exchange keys between two entities over an insecure network |  |  |
| **6** | **IP Security (IPsec) Configuration:** Configure IPsec on network devices to provide secure communication and protect against unauthorized access and attack |  |  |
| **7** | **Web Security with SSL/TLS:** Configure and implement secure web communication using SSL/TLS protocols, including certificate management and secure session management |  |  |
| **8** | **Intrusion Detection system**: Set up and configure an Intrusion detection system (IDS) to monitor network traffic and detect potential security breaches or malicious activities |  |  |
| **9** | **Malware Analysis and Detection:**  Analyse and identify malware samples using antivirus tools, analyse their behaviour, and develop countermeasures to mitigate their impact. |  |  |
| **10** | **Firewall configuration and Rule-based Filtering:** Configure and test firewall rules to control network traffic, filter packets based on specified criteria, and protect network resources from unauthorized access. |  |  |

**PRACTICAL : 1**

**Aim :** Implementing Substitution and Transposition Ciphers: Design and implement algorithms to encrypt and decrypt messages using classical substitution and transposition techniques.

**Solution:**

Substitution Technique

**A) Caesar Cipher**

**Source Code:-**

def encrypt(text, s):

cipher\_t = ""

for char in text:

if char.isupper():

cipher\_t += chr((ord(char) + s - 65) % 26 + 65)

elif char.islower():

cipher\_t += chr((ord(char) + s - 97) % 26 + 97)

else:

cipher\_t += char # leave non-alphabetic characters unchanged

return cipher\_t

def decrypt(cipher\_t, s):

plain\_t = ""

for char in cipher\_t:

if char.isupper():

plain\_t += chr((ord(char) - s - 65) % 26 + 65)

elif char.islower():

plain\_t += chr((ord(char) - s - 97) % 26 + 97)

else:

plain\_t += char # leave non-alphabetic characters unchanged

return plain\_t

text = input("Enter the text to encrypt: ")

s = 3

print("Text: " + text)

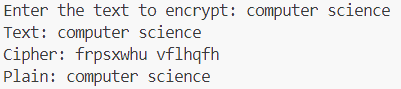
encrypted\_text = encrypt(text, s)

print("Cipher: " + encrypted\_text)

decrypted\_text = decrypt(encrypted\_text, s)

print("Plain: " + decrypted\_text)

**OUTPUT:**

****

**B) RailFence**

**Source Code:-**

def rail\_fence\_cipher(plain\_text, num\_rails):

rails = ['' for \_ in range(num\_rails)]

direction = 1 # Direction: 1 for downwards, -1 for upwards

rail\_index = 0

for char in plain\_text:

rails[rail\_index] += char

rail\_index += direction

# Change direction when reaching top or bottom rail

if rail\_index == 0 or rail\_index == num\_rails - 1:

direction \*= -1

cipher\_text = ''.join(rails)

return cipher\_text

def main():

print("\*\*\*\*\*\*RAILFENCE CIPHER\*\*\*\*\*\*")

plain\_text = input("Enter the plaintext: ")

num\_rails = int(input("Enter the number of Rails: "))

cipher\_text = rail\_fence\_cipher(plain\_text, num\_rails)

print("The Cipher text is:")

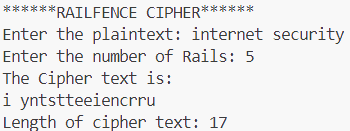
print(cipher\_text)

print("Length of cipher text:", len(cipher\_text))

if \_\_name\_\_ == "\_\_main\_\_":

main()

**OUTPUT:**

****

**PRACTICAL NO :- 2**

**Aim:-** Implement the RSA algorithm for public-key encryption and decryption,and explore its properties and security considerations.

**Source Code:-**

import math

#step 1

p =3

q = 7

#step 2

n = p\*q

print("n= ", n)

#step 3

phi = (p-1)\*(q-1)

#step 4

e = 2

while(e<phi):

if(math.gcd(e, phi)==1):

break

else:

e+=1

print("e = ", e)

#step 5

k = 2

d = ((k\*phi)+1)/e

print("d = ", d)

print(f'Public key: {e, n}')

print(f'Private key: {d, n}')

#Get the original msg from the user

msg = int(input("Enter the original msg (an integer): "))

print(f'Original message:{msg}')

#encryption

C = pow(msg, e)

C = math.fmod(C, n)

print(f'Encrypted message: {C}')

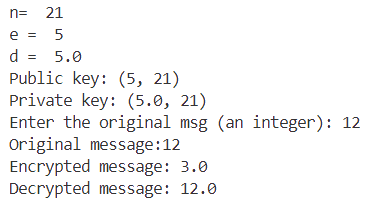
#decryption

M = pow(C, d)

M =math.fmod(M, n)

print(f'Decrypted message: {M}')

**OUTPUT:**

****

**PRACTICAL : 3**

**Aim :** To study and implement the Message Authentication Code for ensuring the message integrity and authenticity.

**Code: Python code for implementing MD5 Algorithm:**

import hashlib

result = hashlib.md5(b'Computer')

result1 = hashlib.md5(b'Science')

# printing the equivalent byte value.

print("The byte equivalent of hash is : ", end ="")

print(result.digest())

print("The byte equivalent of hash is : ", end ="")

print(result1.digest())

**OUTPUT:**

****

**Code: Python code for implementing SHA Algorithm**

import hashlib

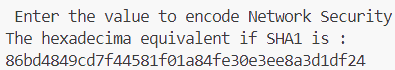
str = input(" Enter the value to encode ")

result = hashlib.sha1(str.encode())

print("The hexadecima equivalent if SHA1 is : ")

print(result.hexdigest())

**OUTPUT:**

****

**PRACTICAL : 4**

**Aim :** Digital Signatures: Implement digital signature algorithms such as RSA-based signatures, and verify the integrity and authenticity of digitally signed messages.

**Source Code :**

from Crypto.PublicKey import RSA

from Crypto.Signature import pkcs1\_15

from Crypto.Hash import SHA256

#Generate RSA Key pair

key = RSA.generate(2048)

private\_key=key.export\_key()

public\_key=key.publickey().export\_key()

#Simulated document content

original\_document = b"This is the original document content."

modified\_document=b"This is the modified document content."

#Hash the document content

original\_hash=SHA256.new(original\_document)

modified\_hash=SHA256.new(modified\_document)

#Create a signature using the private key

signature = pkcs1\_15.new(RSA.import\_key(private\_key)).sign(original\_hash)

#Verify the signature using the public key with the modified content

try:

pkcs1\_15.new(RSA.import\_key(public\_key)).verify(modified\_hash,signature)

print("Signature is valid.")

except(ValueError,TypeError):

print("Signature is invalid.")

**OUTPUT:**

****



**PRACTICAL : 5**

**Aim:** To study and implement the Diffe-Hellman key exchange algorithm for secure exchange of keys between two entities.

**Source Code A :**

from random import randint

if \_\_name\_\_ == '\_\_main\_\_':

P = 23

G = 9

print('The Value of P is :%d'%(P))

print('The Value of G is :%d'%(G))

a = 4

print('Secret Number for Alice is :%d'%(a))

x = int(pow(G,a,P))

b = 6

print('Secret Number for Bob is :%d'%(b))

y = int(pow(G,b,P))

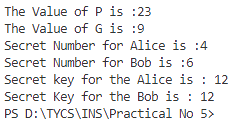
ka = int(pow(y,a,P))

kb = int(pow(x,b,P))

print('Secret key for the Alice is : %d'%(ka))

print('Secret Key for the Bob is : %d'%(kb))

**OUTPUT:**

****

**Code B:**

def calculate\_power(x, y, P):

return pow(x, y, P)

def main():

# Take public keys input

print("Both users should agree upon the public keys G and P")

G = int(input("Enter value for public key G: "))

P = int(input("Enter value for public key P: "))

# Take private keys input

a = int(input("Enter value for private key selected by user 1: "))

b = int(input("Enter value for private key selected by user 2: "))

# Calculate x and y keys

x = calculate\_power(G, a, P)

y = calculate\_power(G, b, P)

# Calculate secret keys

ka = calculate\_power(y, a, P)

kb = calculate\_power(x, b, P)

# Print secret keys

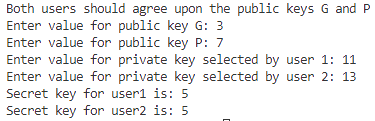
print(f"Secret key for user1 is: {ka}")

print(f"Secret key for user2 is: {kb}")

if \_\_name\_\_ == "\_\_main\_\_":

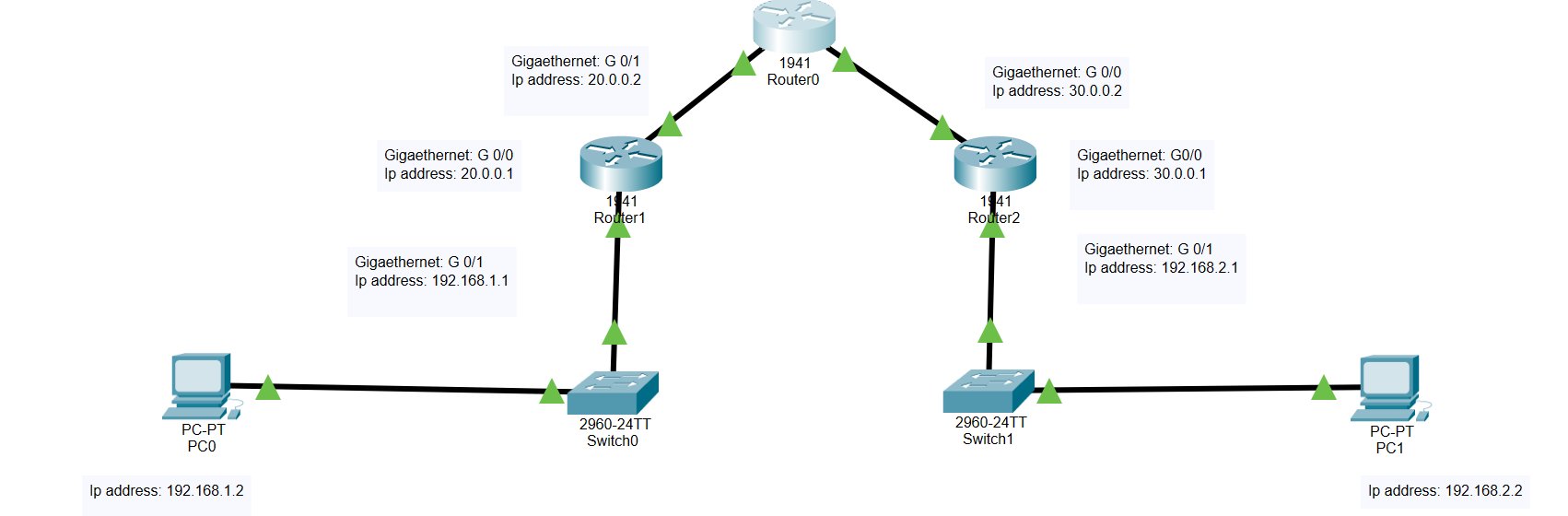
main()

**Output:**

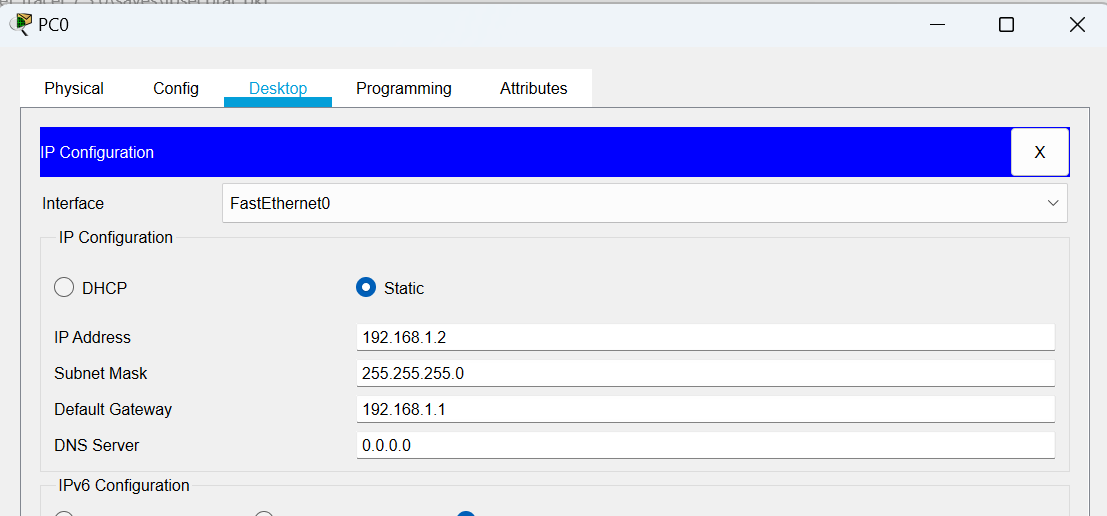


**PRACTICAL NO :- 6**

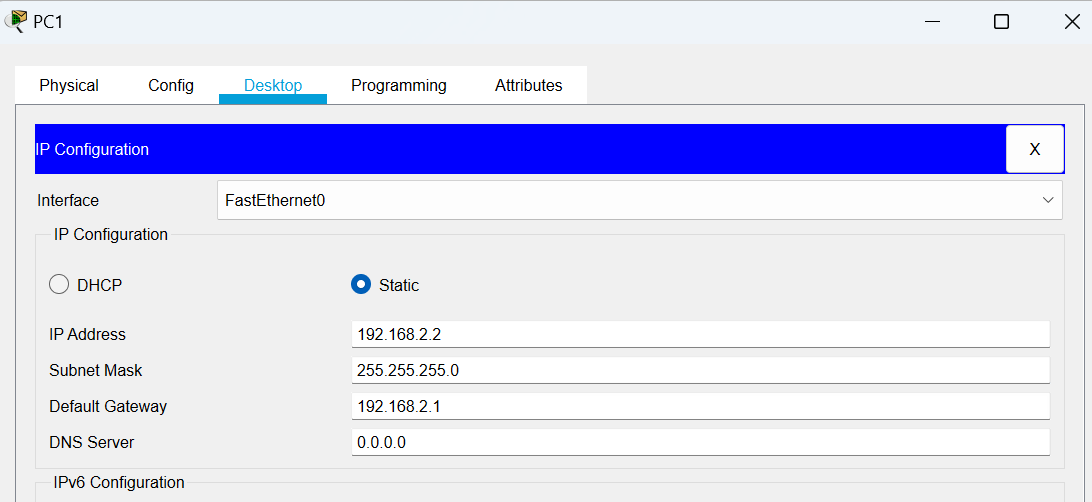
**Program Statement:-Ip Security (IPSEC) configuration.**

****

**Configuring PC0:**

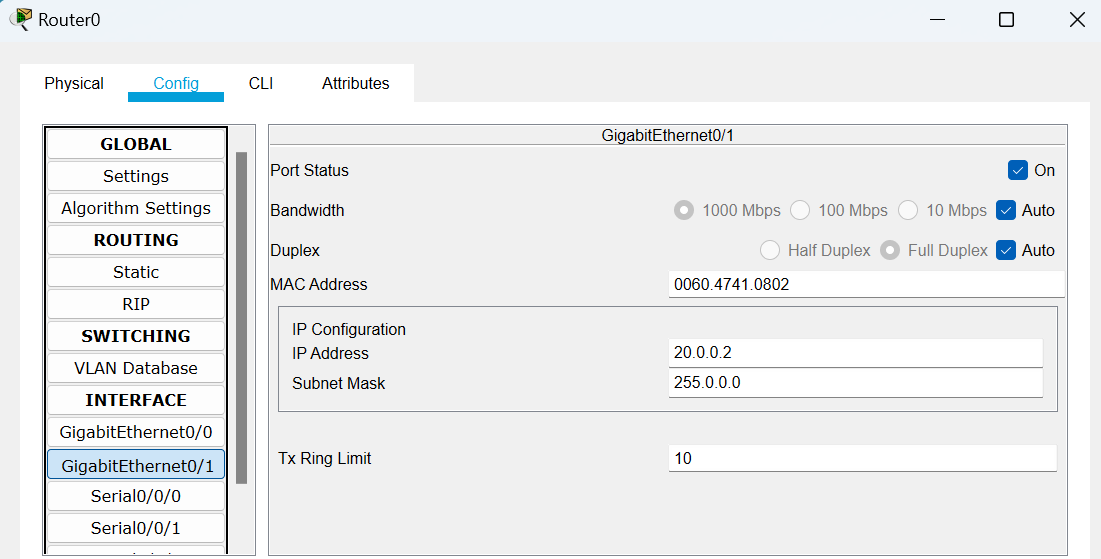
****

**Configuring PC1:**

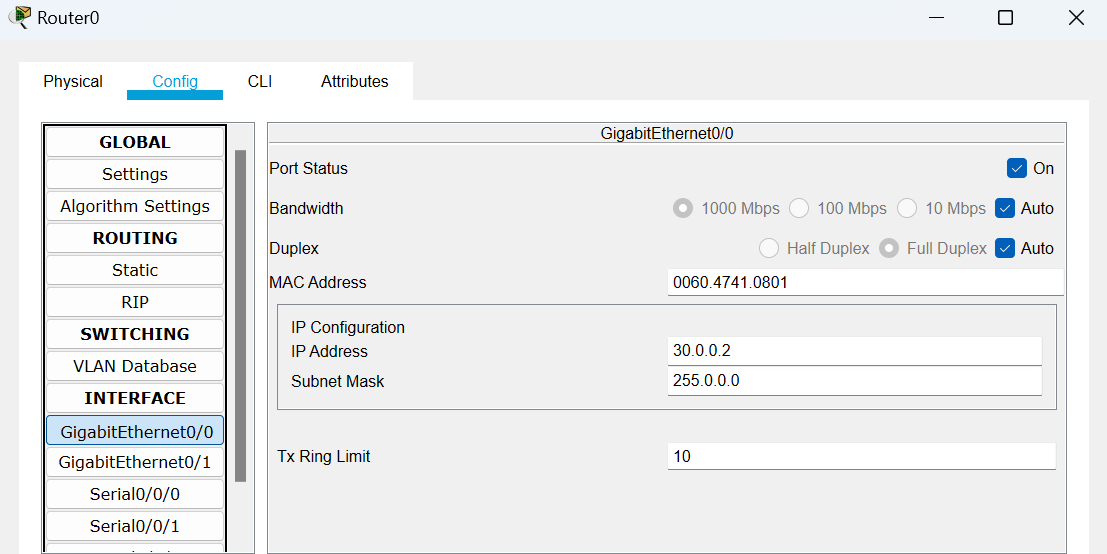
****

**Configuring Router0:**

**Interface GigabitEthernet0/1:**

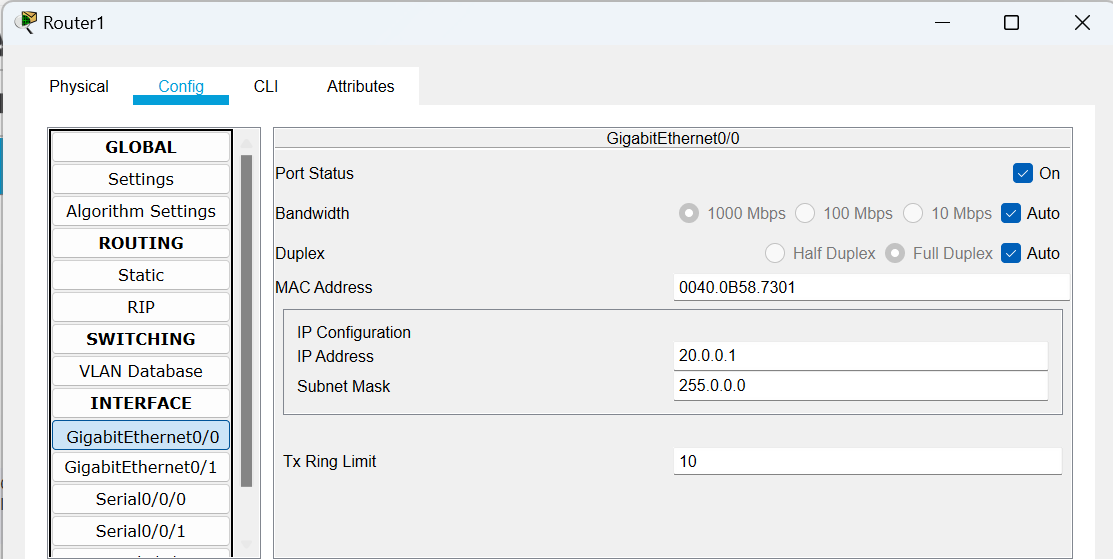
****

**Interface GigabitEthernet0/0:**

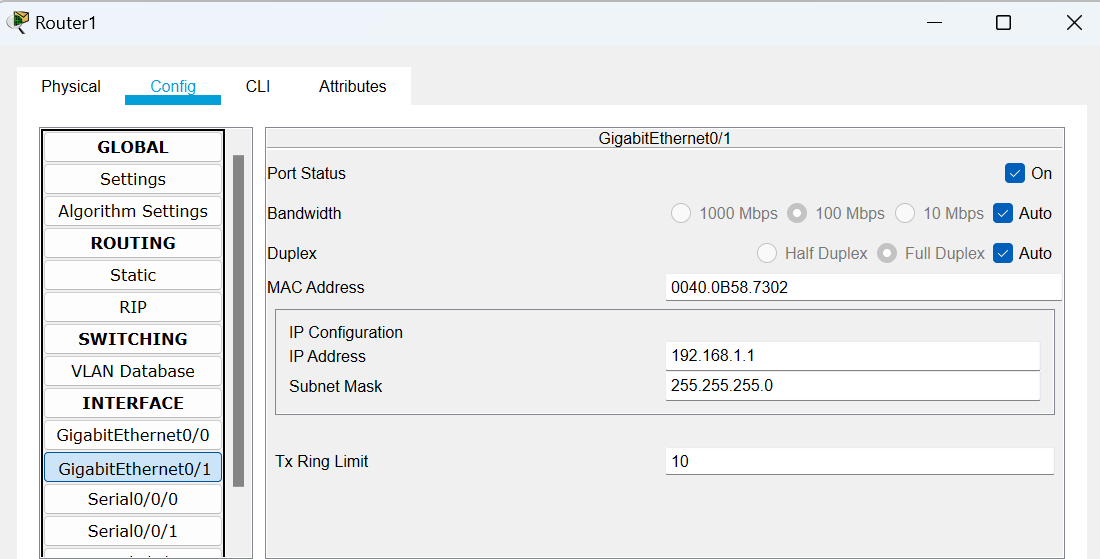
****

**Configuring Router1:**

**Interface GigabitEthernet0/0:**

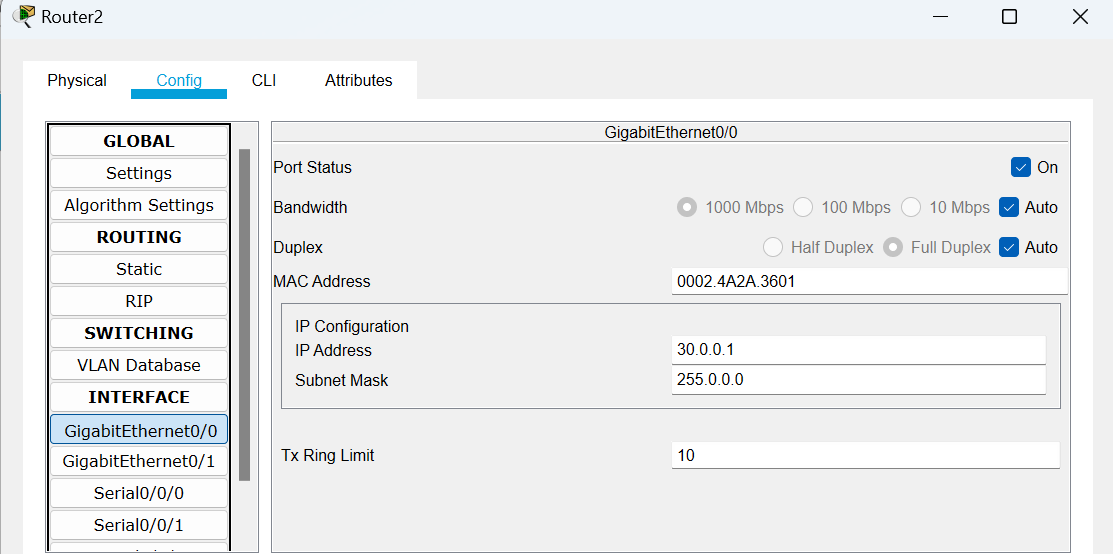
****

**Interface GigabitEthernet0/1:**

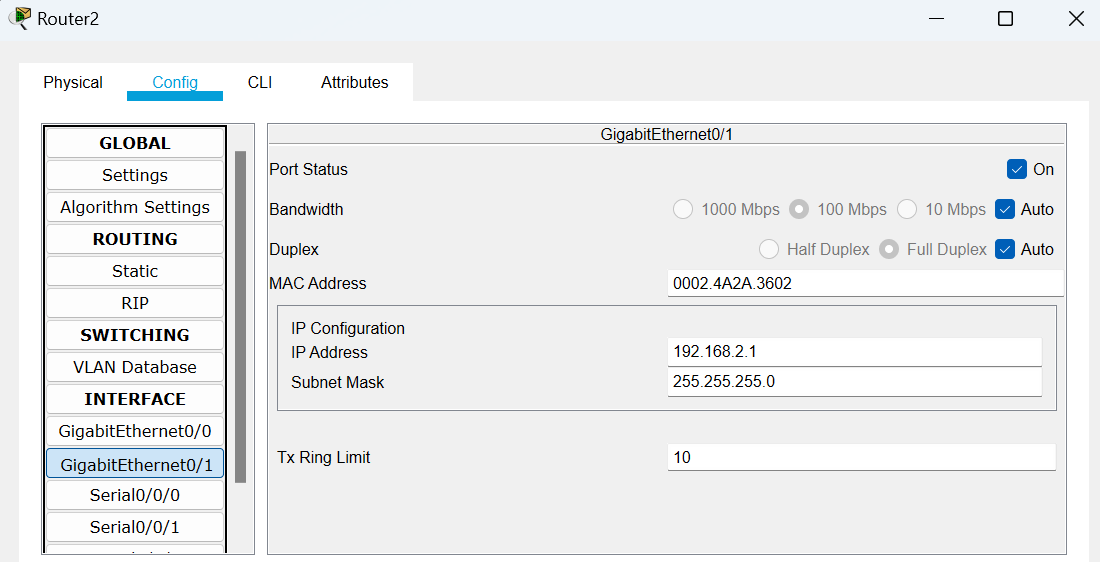
****

**Configuring Router2:**

**Interface GigabitEthernet0/0:**

****

**Interface GigabitEthernet0/1:**

****

**Checking and Enabling the Security features in Router R1 and R2:**

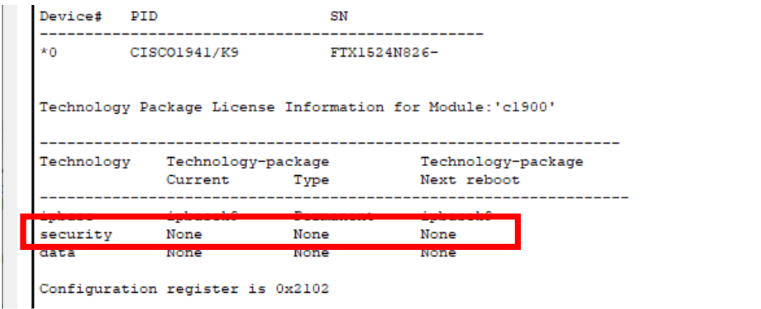
**Enter the following command in the CLI mode of Router1**

Router(config)#ip route 0.0.0.0 0.0.0.0 20.0.0.2

Router(config)#hostname R1

R1(config)#exit

R1#show version

****

(We see that the security feature is not enabled, hence we need to enable the security package

R1#

R1#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

R1(config)#

R1(config)#license boot module c1900 technology-package securityk9

R1(config)#exit

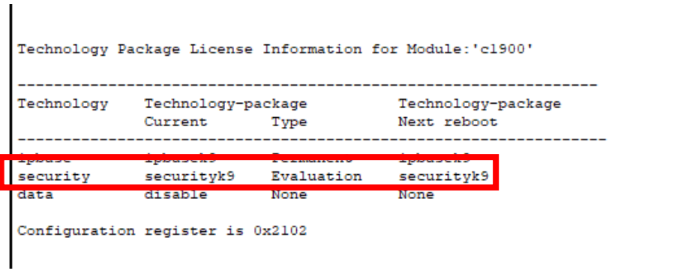
R1#

R1#copy run startup-config

R1#reload

R1>enable

R1#show version

****

(The security package is enabled)

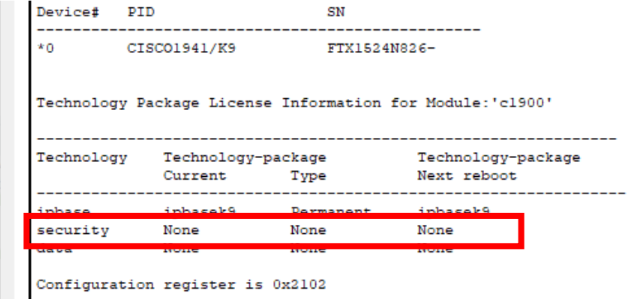
Enter the following command in the CLI mode of Router2

Router(config)#ip route 0.0.0.0 0.0.0.0 30.0.0.2

Router(config)#hostname R2

R2(config)#exit

R2#show version

****

(We see that the security feature is not enabled, hence we need to enable the security package

R2#

R2#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

R2(config)#

R2(config)#license boot module c1900 technology-package securityk9

R2(config)#exit

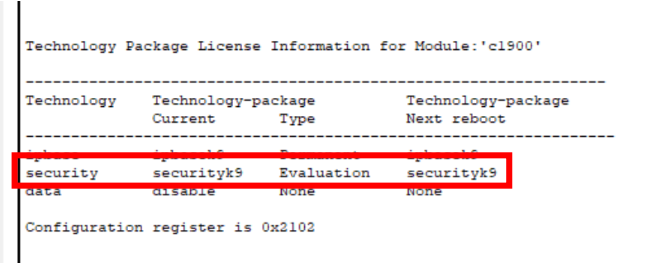
R2#

R2#copy run startup-config

R2#reload

R2>enable

R2#show version

****

(The security package is enabled)

Enter the following command in the CLI mode of Router0

Router>enable

Router#configure terminal

Router(config)#hostname R0

R0(config)#

**Defining the Hostname for all Routers and Configuring the Routers R1 and R2 for IPSec VPN tunnel**

R1#configure terminal

R1(config)#access-list 100 permit ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255

R1(config)#crypto isakmp policy 10

R1(config-isakmp)#encryption aes 256

R1(config-isakmp)#authentication pre-share

R1(config-isakmp)#group 5

R1(config-isakmp)#exit

R1(config)#crypto isakmp key ismile address 30.0.0.1

R1(config)#crypto ipsec transform-set R1->R2 esp-aes 256 esp-sha-hmac

R1(config)#

R2#

R2#configure terminal

R2(config)#access-list 100 permit ip 192.168.2.0 0.0.0.255 192.168.1.0 0.0.0.255

R2(config)#crypto isakmp policy 10

R2(config-isakmp)#encryption aes 256

R2(config-isakmp)#authentication pre-share

R2(config-isakmp)#group 5

R2(config-isakmp)#exit

R2(config)#crypto isakmp key ismile address 20.0.0.1

R2(config)#crypto ipsec transform-set R2->R1 esp-aes 256 esp-sha-hmac

R2(config)#

R1>enable

R1#configure terminal

R1(config)#crypto map IPSEC-MAP 10 ipsec-isakmp

R1(config-crypto-map)#set peer 30.0.0.1

R1(config-crypto-map)#set pfs group5

R1(config-crypto-map)#set security-association lifetime seconds 86400

R1(config-crypto-map)#set transform-set R1->R2

R1(config-crypto-map)#match address 100

R1(config-crypto-map)#exit

R1(config)#interface g0/0

R1(config-if)#crypto map IPSEC-MAP

R2>enable

R2#configure terminal

R2(config)#crypto map IPSEC-MAP 10 ipsec-isakmp

R2(config-crypto-map)#set peer 20.0.0.1

R2(config-crypto-map)#set pfs group5

R2(config-crypto-map)#set security-association lifetime seconds 86400

R2(config-crypto-map)#set transform-set R2->R1

R2(config-crypto-map)#match address 100

R2(config-crypto-map)#exit

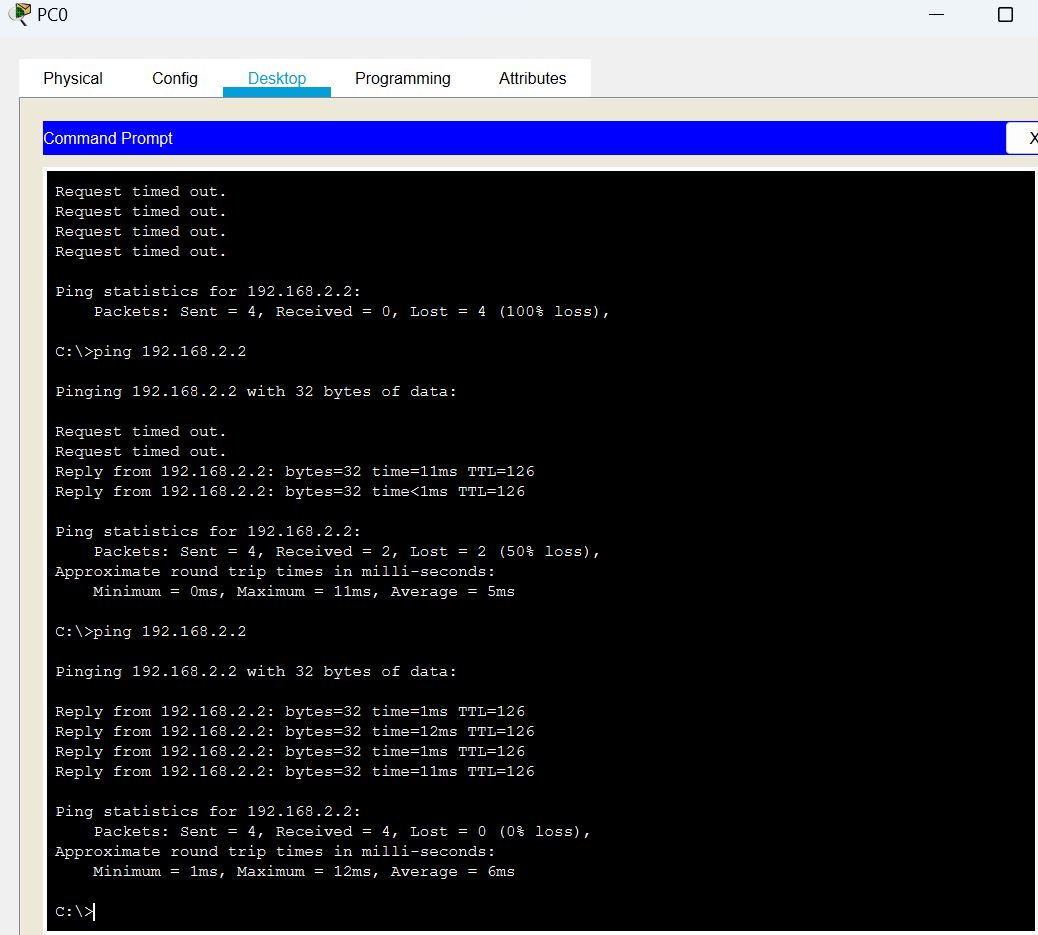
R2(config)#interface g0/0

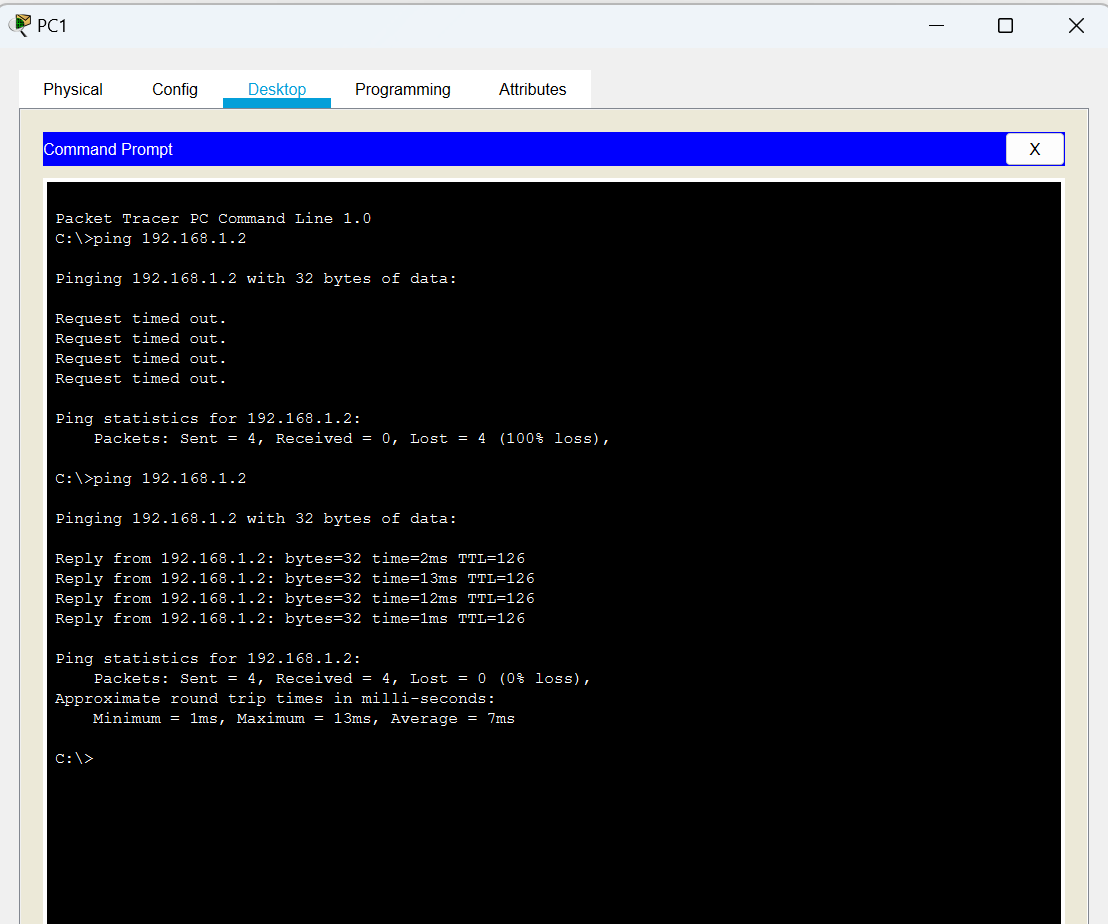
R2(config-if)#crypto map IPSEC-MAP

We verify the working of the IPSec VPN tunnel using the ping command as follows

**Output:-**

Pinging PC0(192.168.2.2) from PC1 and then PC1(192.168.1.2) from PC0

****

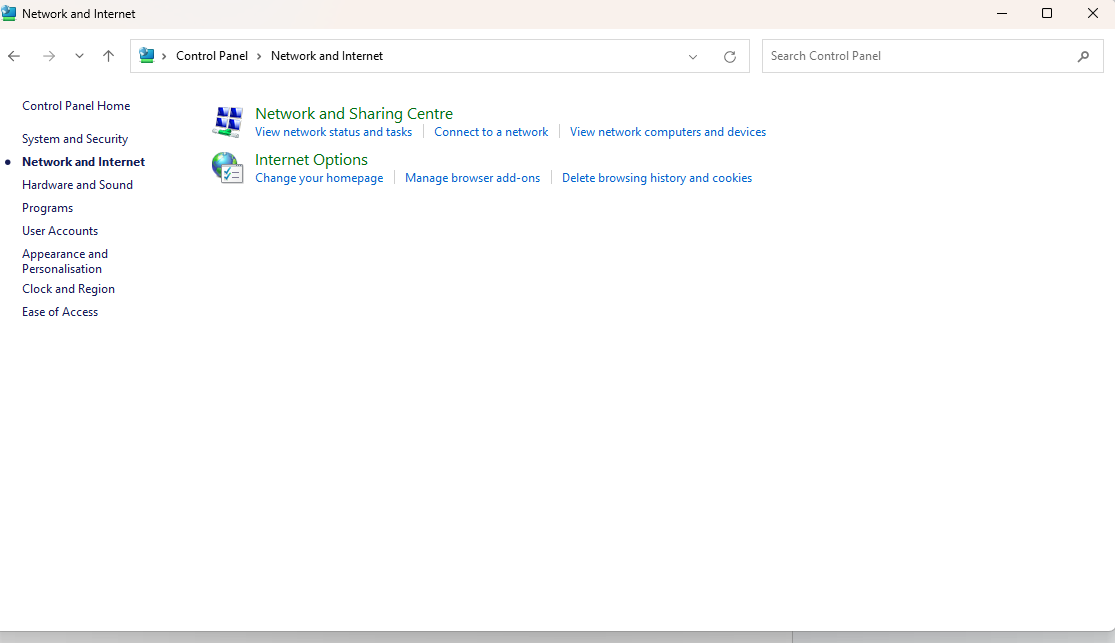
****

**PRACTICAL : 7**

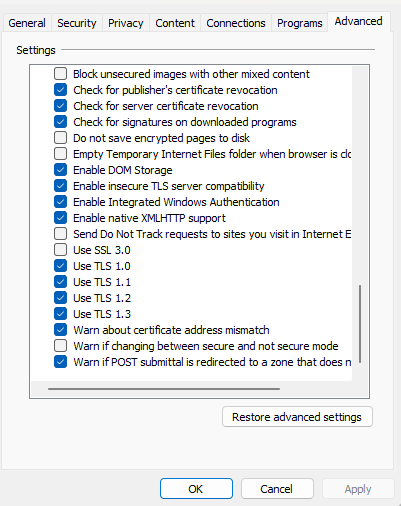
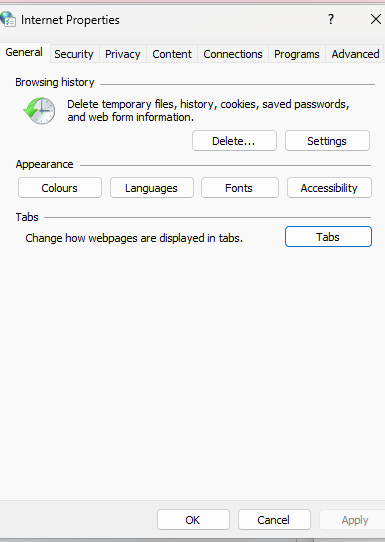
**Aim :** Web Security with SSL/TLS.Configure and implement secure web communication using SSL/TLS protocols, including certificate management and secure session establishment.

**Solution:**

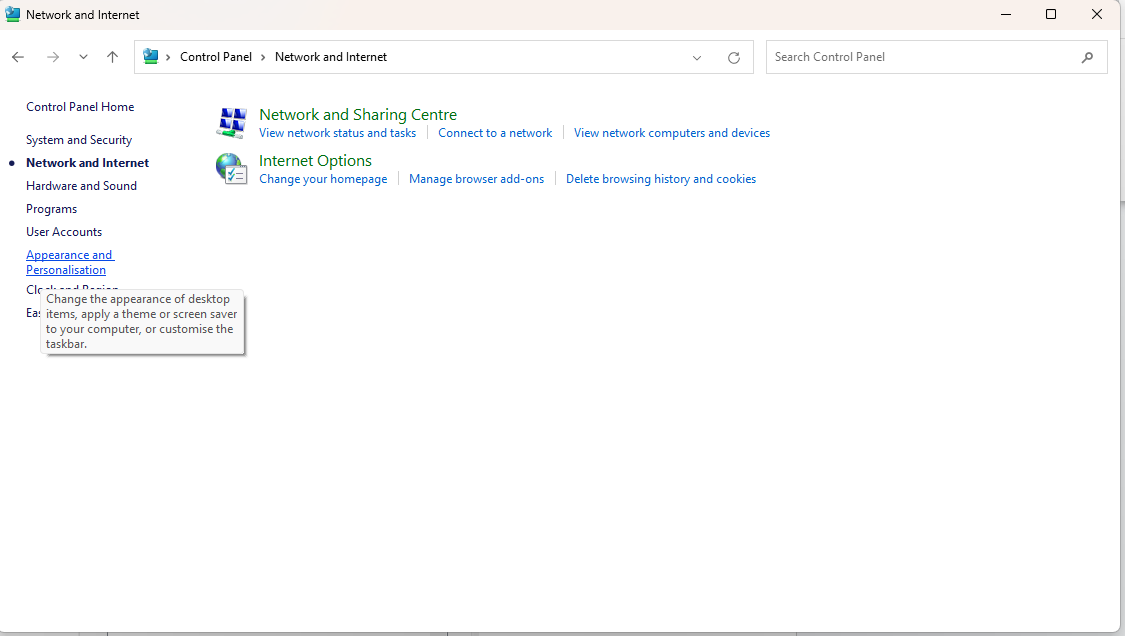
1)Open the control panel and select internet options.



2. Select advanced option

****

3.Now select Use 1.0 and press ok.



4. Done

**PRACTICAL : 8**

**Aim:** Intrusion Detection System:

Set up and configure an intrusion detection system (IDS) to monitor network traffic and

detect potential security breaches or malicious activities.

**Source Code:**

Server Side

import socket

import threading

class IDS:

def \_\_init\_\_(self):

self.attack\_signatures=['malware','exploit', 'attack']

self.detected\_attacks = []

def detect\_attack(self,packet):

for signature in self.attack\_signatures:

if signature in packet:

self.detected\_attacks.append(signature)

print(f"Detected{signature} attack in packet:{packet}")

def listen\_for\_traffic(ids):

host = '0.0.0.0'

port = 5501

with socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)as server\_socket:

server\_socket.bind((host,port))

server\_socket.listen(5)

print(f"Listening on {host}:{port} for incoming traffic...")

try:

while True:

client\_socket,client\_addr = server\_socket.accept()

client\_ip,client\_port = client\_addr

print(f"Accepted connection from {client\_ip},client\_port{client\_port}")

packet = client\_socket.recv(1024).decode('utf-8')

ids.detect\_attack(packet)

accept\_count = False

client\_socket.close()

except KeyboardInterrupt:

print("Server Stopping...")

def main():

ids = IDS()

traffic\_listener = threading.Thread(target = IDS.listen\_for\_traffic, args =(ids,))

traffic\_listener.start()

if \_\_name\_\_ == "\_\_main\_\_" :

main()

Clientside

import socket

def main():

server\_ip = '127.0.0.1' #Change This to the IP address Where the server is running

server\_port = 5501 # Change this to the port on which the server is listening

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client\_socket.connect((server\_ip, server\_port))

message = "This is a test packet with malware."

client\_socket.send(message.encode('utf-8'))

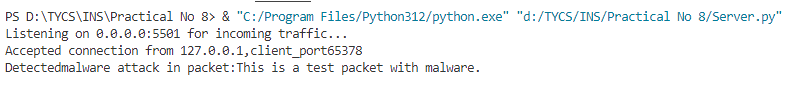
print(f"Send message to server:{message}")

client\_socket.close()

if \_\_name\_\_ == "\_\_main\_\_" :

main()

**Output:-**

****

****

**PRACTICAL NO :- 9**

**Program Statement:-**  Malware Analysis and Detection:

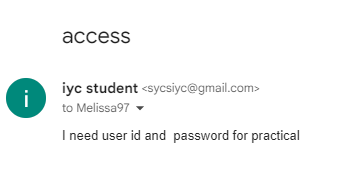
Analyze and identify malware samples using antivirus tools, analyze their behavior, and

develop countermeasures to mitigate their impact.

**Step 1:**

Go to virusshare.com and request for ID and password.

Email Melissa at Melissa97@virusshare.com with 'access' in the subject. She will review your request and hopefully send you an invitation link.

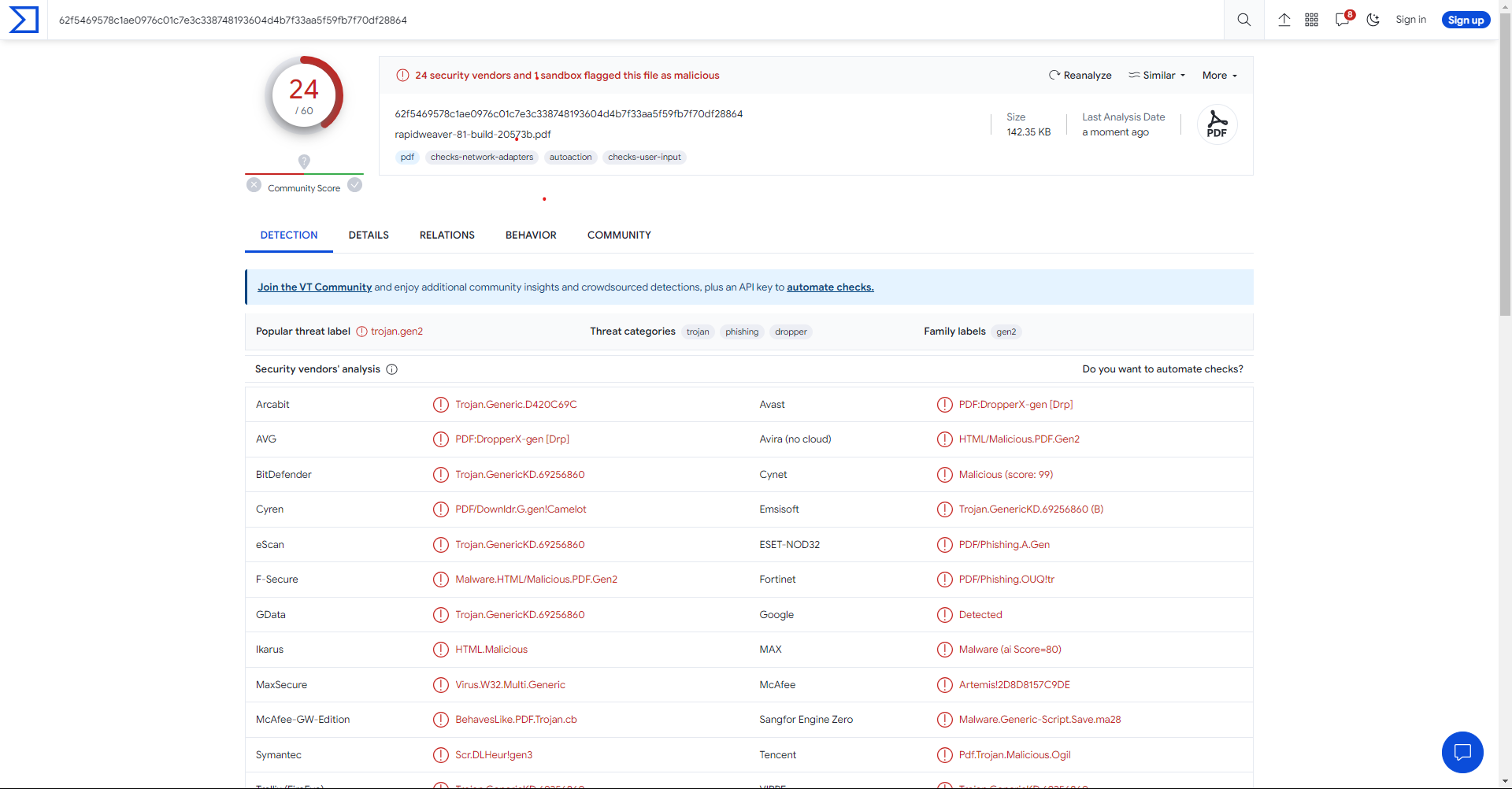


**Step 2:**After receiving email go to virusshare.com and login with your credentials .

Download zip file of virus

**Step 3:** Go to <https://www.virustotal.com> and upload a zip file and enter the password ‘infected’.

**Step 4:** It will show all the detected file

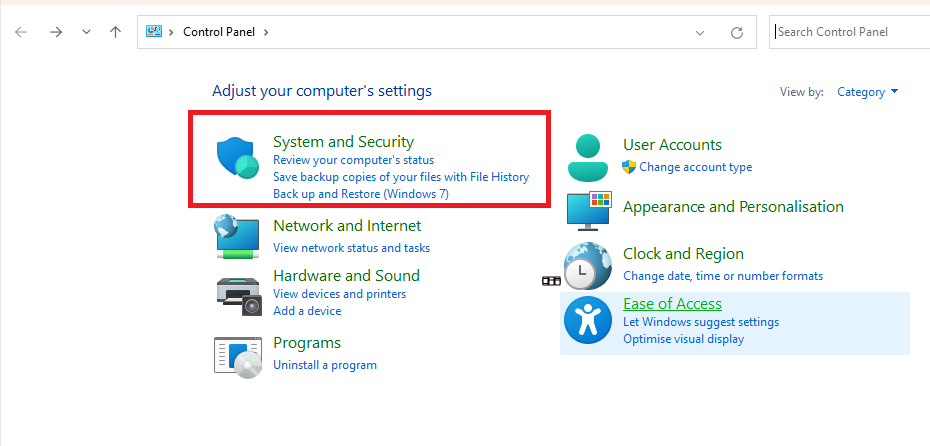


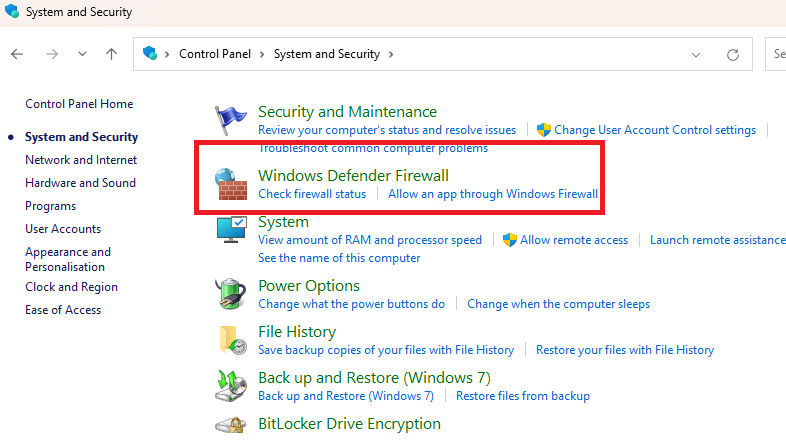
**PRACTICAL NO :- 10**

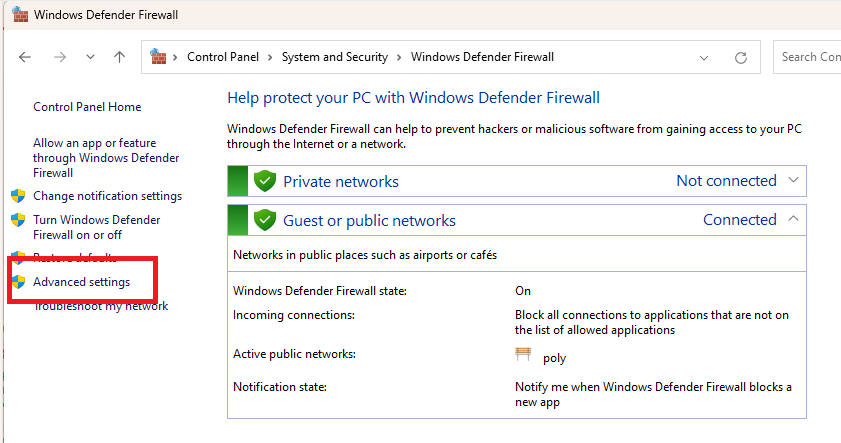
**Program Statement:-** Configure and test firewall rules to control network traffic,filter packets based on specified criteria, and protect network resources from unauthorized access.

**A :- Port**

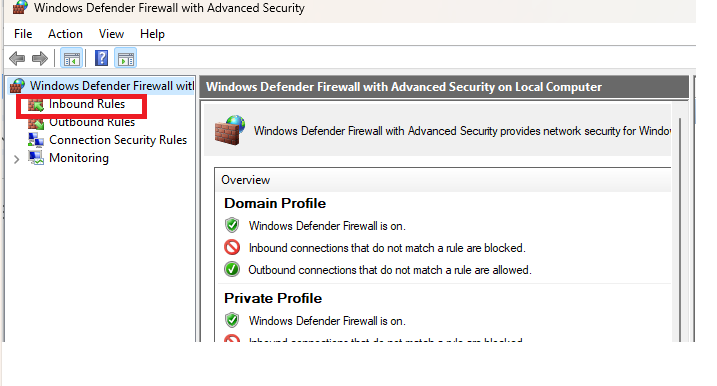
**Step 1:-** Start > Control Panel > Windows Firewall > Advanced Settings

****

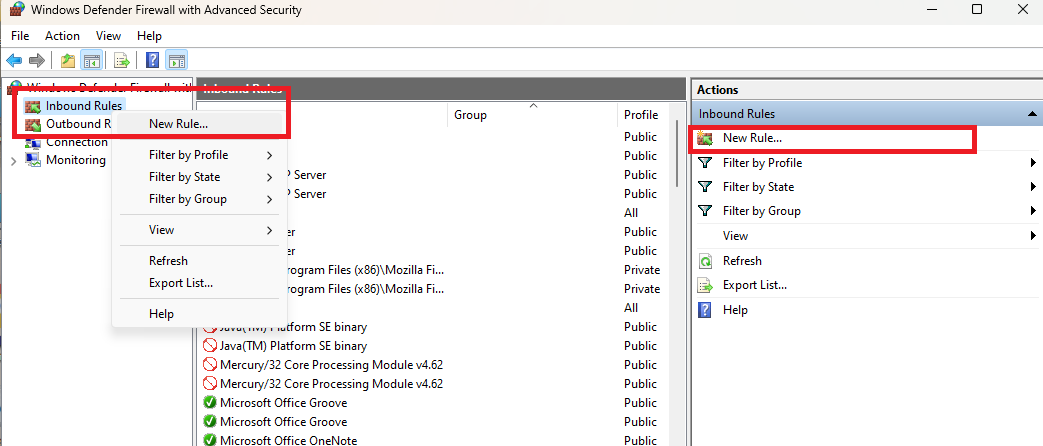
****

****

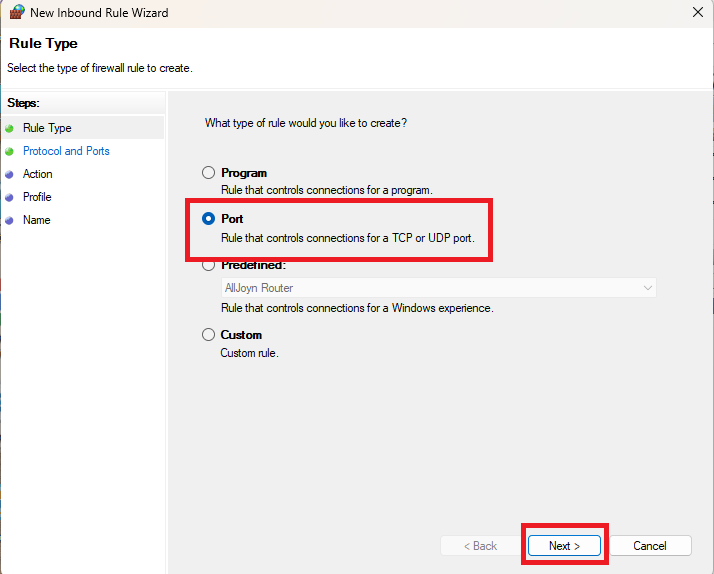
**Step 2:-** Left hand side of the window shows a list of rules.From the list, select Inbound Rules.

****

**Step 3:-** It will open ‘New Inbound Rule Wizard’ window.

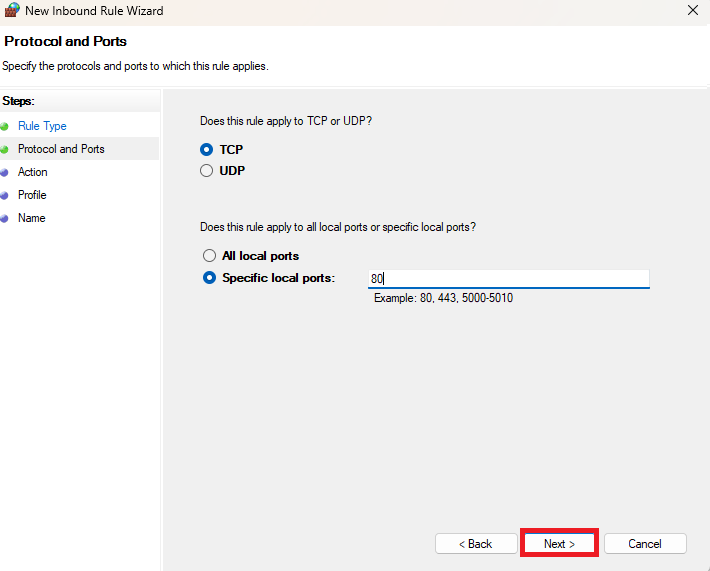
****

**Step 4:-** From it, select ‘port’ as the new Rule Type and click Next

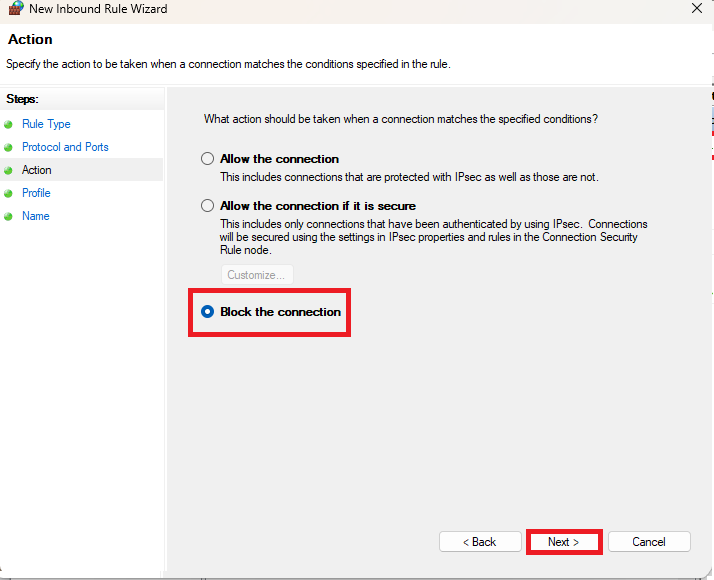
****

**Step 5:-** Select the option ‘TCP’ and in the field ‘Specific local ports’ enter ‘80’.

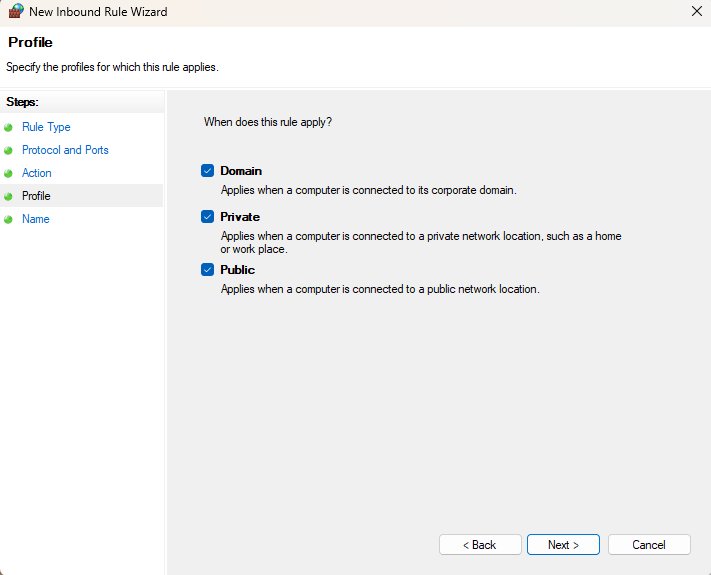
**Step 6:-** Click Next to Continue.

****

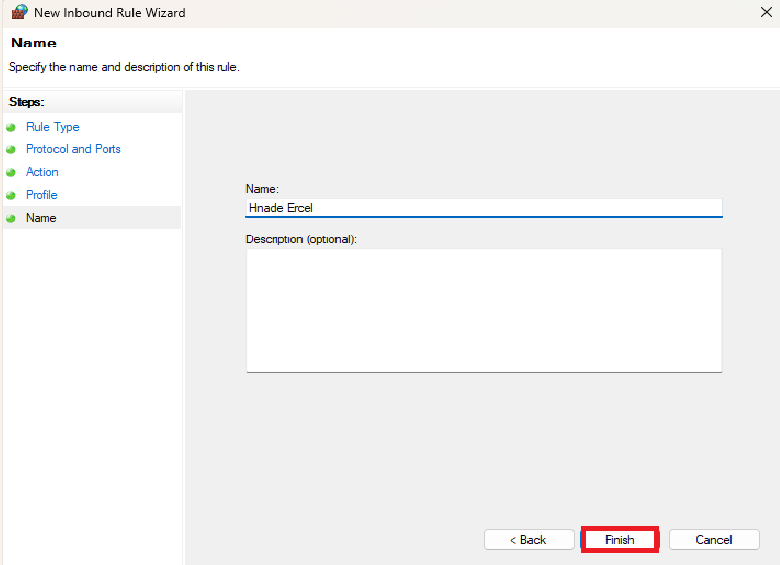
**Step 7:-** Next, select ‘Block the connection’ as the Action and click Next.

****

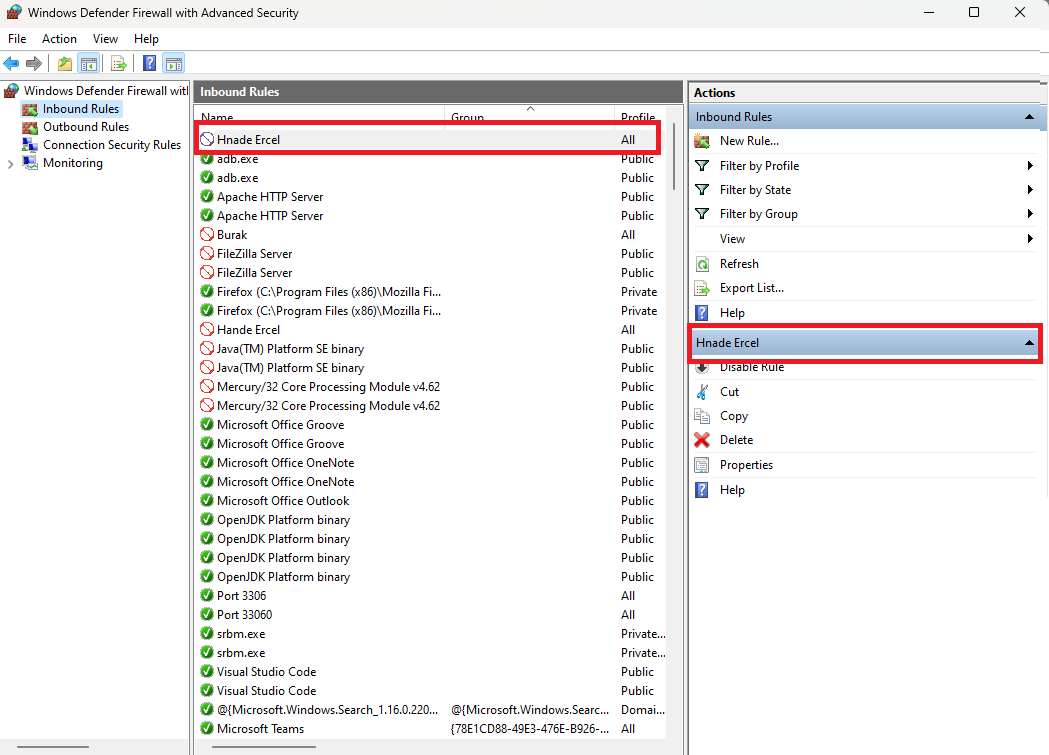
**Step 8:-** Later, select all the profiles available for different type of connections(Domain, Private and Public) and Click Next to continue.

****

**Step 9:-** Enter the name of your choice to the new rule ‘Hande Ercel’.Adding description for the same rule is optional.

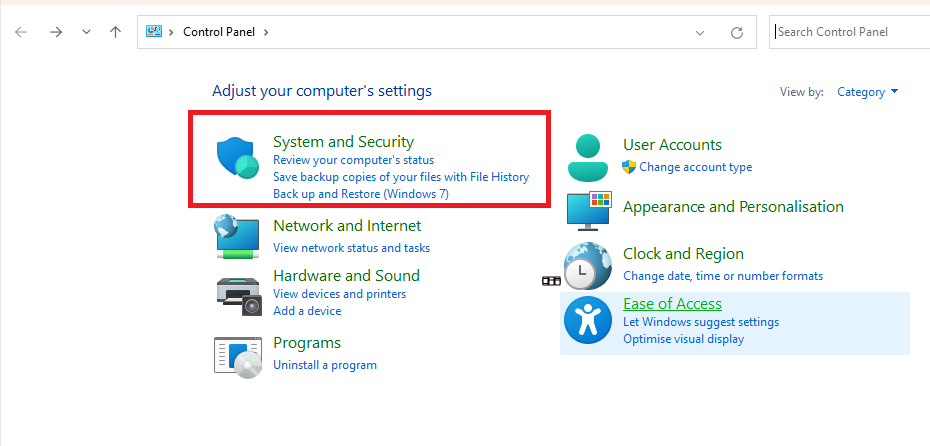
****

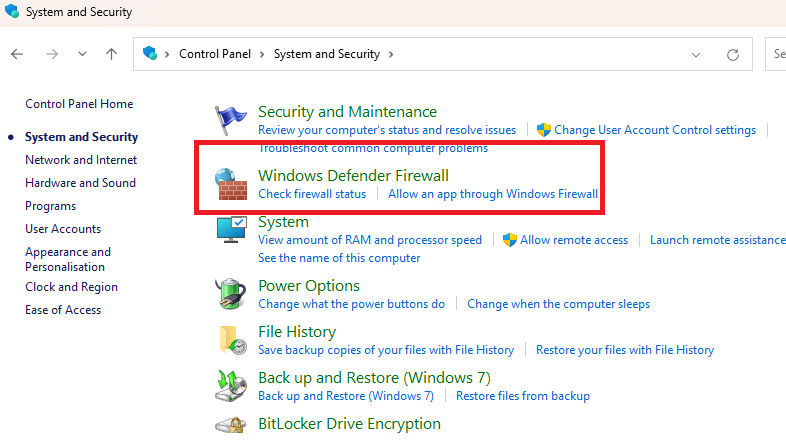
**Step 10:-** Finally, click the Finish button to configure the settings.

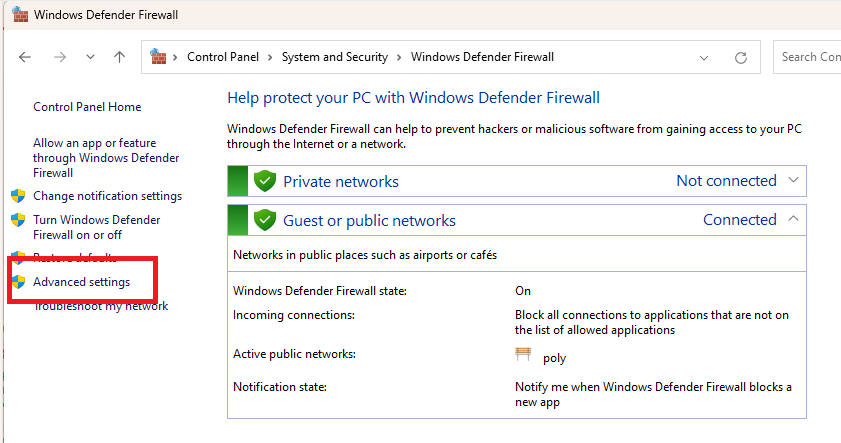
****

**B :- Program**

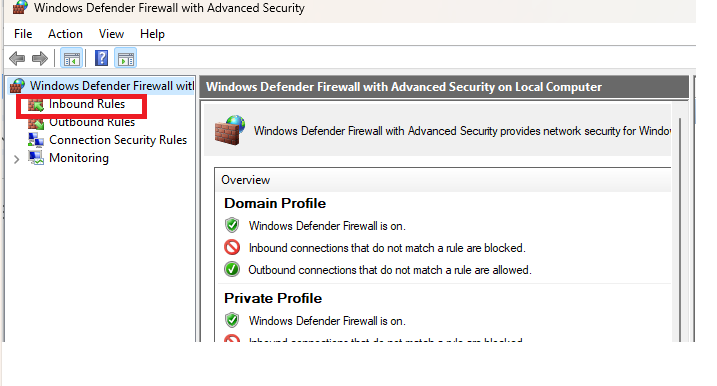
**Step 1:-** Start > Control Panel > Windows Firewall > Advanced Settings

****

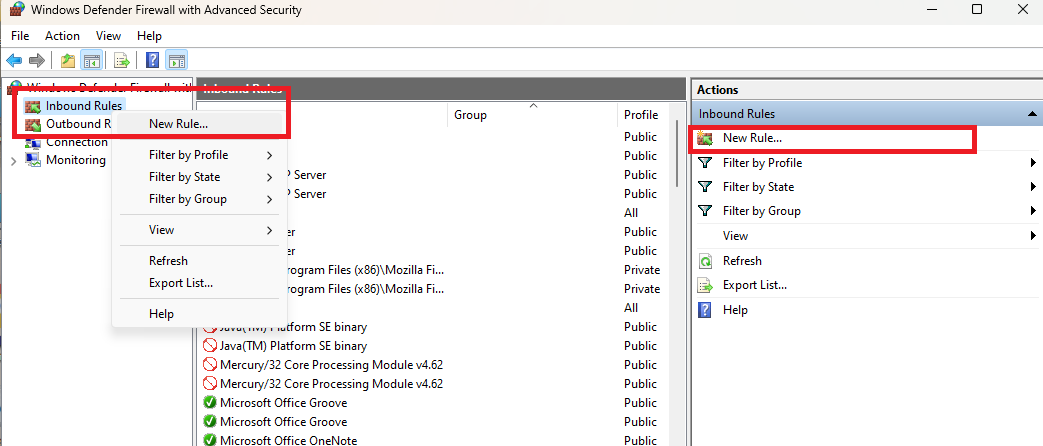
****

****

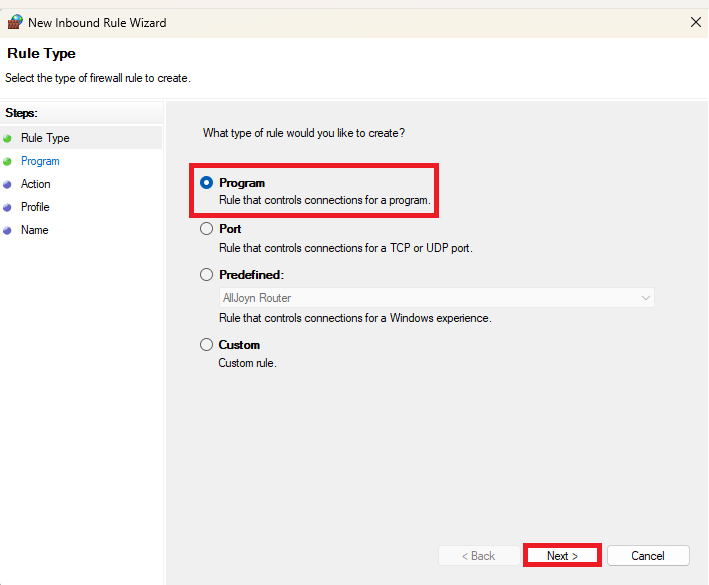
**Step 2:-** Left hand side of the window shows a list of rules.From the list, select Inbound Rules.

****

**Step 3:-** It will open ‘New Inbound Rule Wizard’ window.

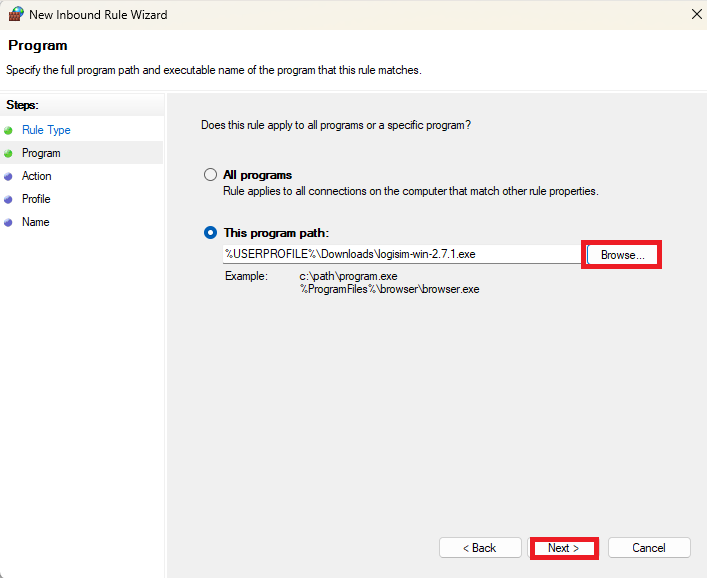
****

**Step 4:-** From it, select ‘port’ as the new Rule Type and click Next

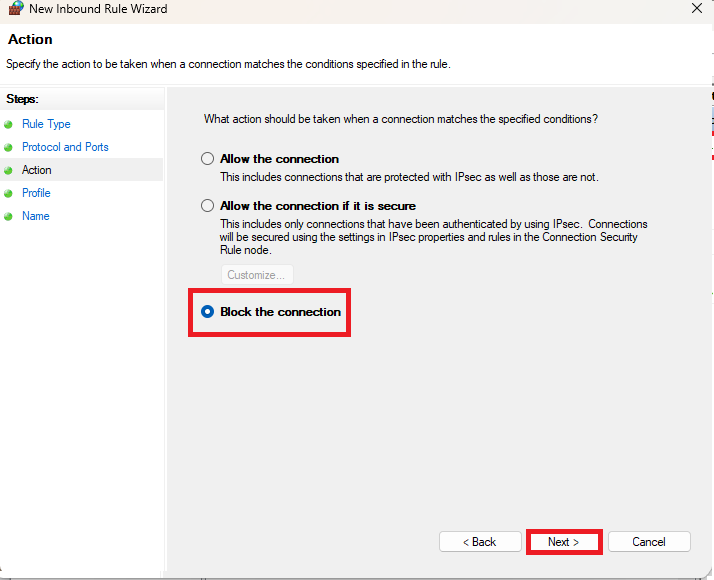


**Step 5:-** Browse **the exe file.**

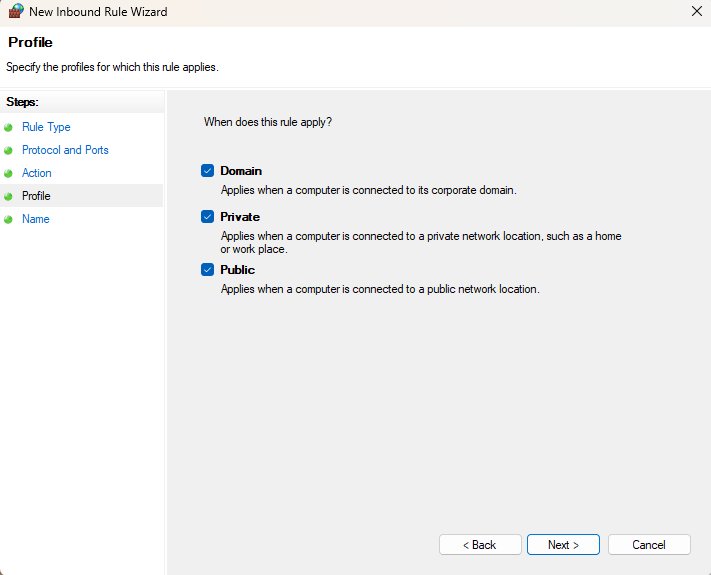
**Step 6:-** Click Next to Continue.

****

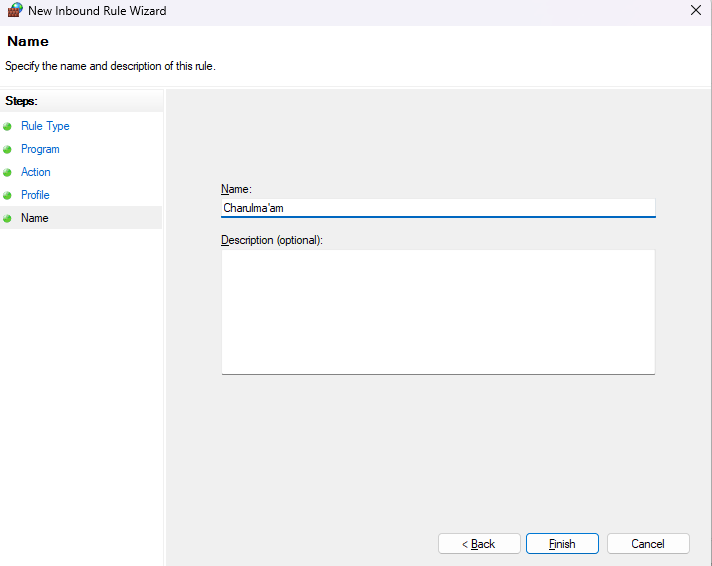
**Step 7:-** Next, select ‘Block the connection’ as the Action and click Next.

****

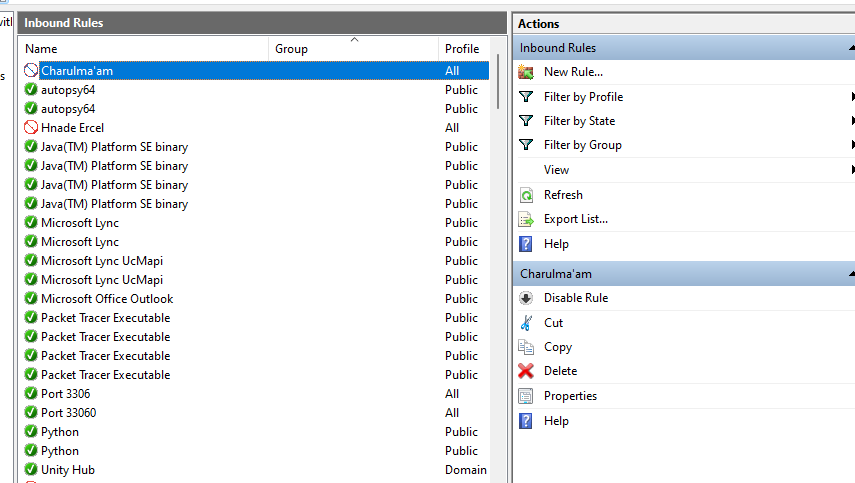
**Step 8:-** Later, select all the profiles available for different types of connections(Domain, Private and Public) and Click Next to continue.

****

**Step 9:-** Enter the name of your choice to the new rule ‘Hande Ercel’.Adding description for the same rule is optional.



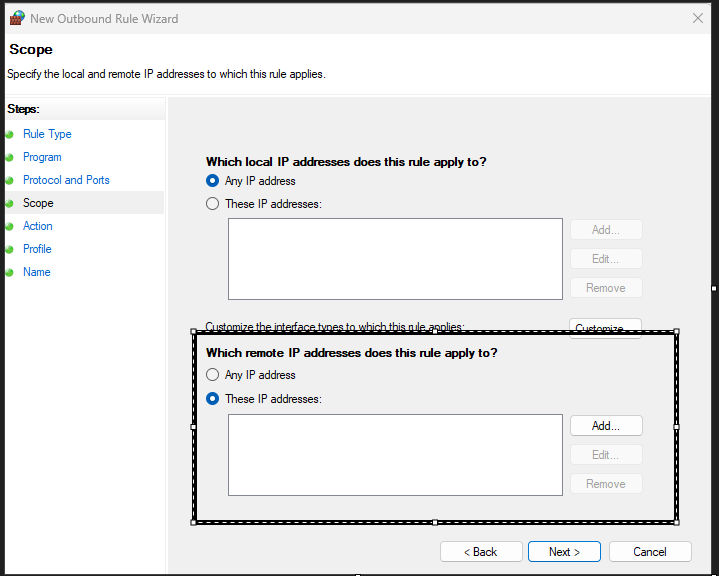
**Step 10:-** Finally, click the Finish button to configure the settings**.**



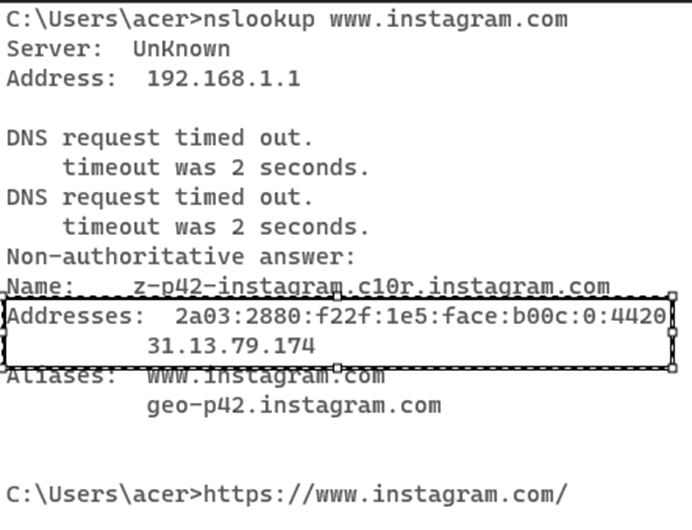
**C :- Website**

**Step 1:** Open control panel → Click ‘System and Security’ → Click ‘Windows Defender Firewall’ → Click ‘Advanced setting’ → Click ‘Outbound Rules’ → Click ‘New Rules ’ → Click ‘Custom’ and next → Check ‘All programs’ and next

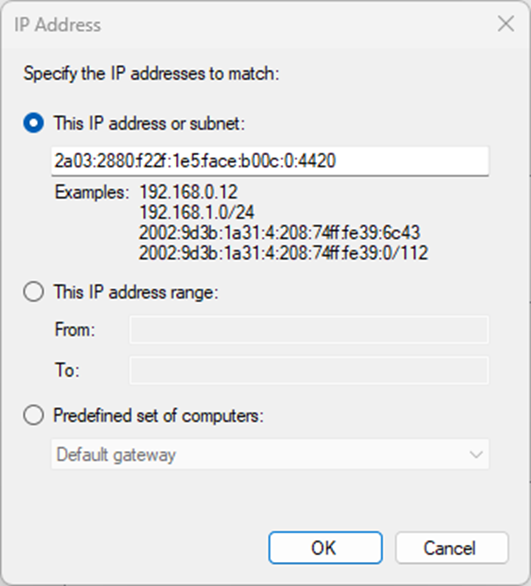
**Step 2:** Click on add in remote IP address

****

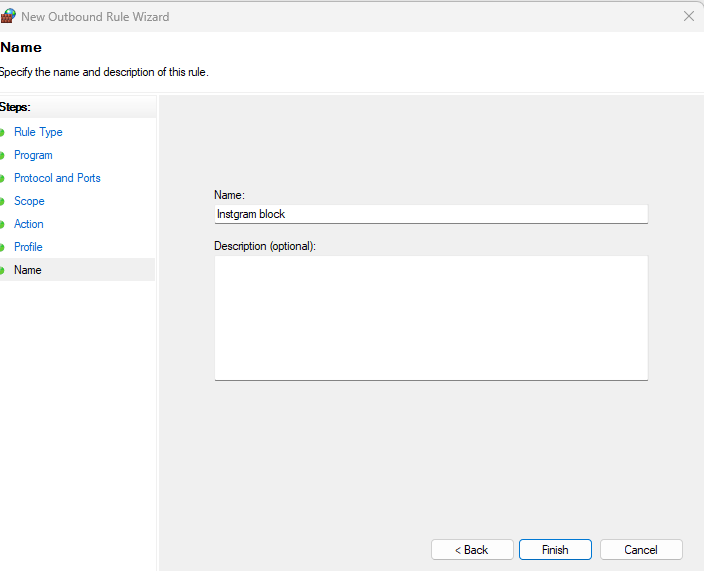
**Step 3 :** Open cmd as a administrator and type below command with website name

****

**Add these 2 IP address**

****

**Step 4:** After adding these IP address click next → and block the connection → click on next and block the connection

****

**Instagram block Successfully**

****