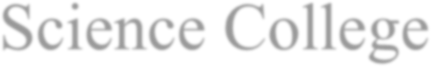


R.D & S.H National College & S.W. A



Science College

PRACTICAL JOURNAL OF

INFORMATION & NETWORK SECURITY

**PRAVALITH RAO**

**ROLL NO: CS21013**

**SEAT NO: 2222254**

**TYBSC COMPUTER SCIENCE**



**R.D. & S. H. National College & S. W.A. Science College Bandra, Mumbai – 400050.**

**Department of Computer Science**

**CERTIFICATE**

This is to certify that **MR. PRAVALITH RAO** of **T.Y.B.Sc(Computer Science)** class **(V Semester)** bearing University Seat No. **2222254** has satisfactorily completed Practical’s, in the subject of **Cyber Forensics** a part of B.Sc. in Computer Science Program during the academic year 2023 – 2024.

**Date of Certification:**

**Subject Incharge Co-ordinator,**

**Department Computer Science**

**Signature of Examiner**

# INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| **SR.NO.** | **TITLE** | **DATE** | **SIGN** |
| 1 | Implementing Substitution and Transposition Ciphers | 08/08/2023 |  |
| 2 | RSA Encryption and Decryption | 24/08/2023 |  |
| 3 | Message Authentication Codes | 31/08/2023 |  |
| 4 | Digital Signatures | 05/09/2023 |  |
| 5 | Key Exchange using Diffie-Hellman | 05/09/2023 |  |
| 6 | IP Security (IPsec) Configuration | 11/09/2023 |  |
| 7 | Web Security with SSL/TLS | 26/09/2023 |  |
| 8 | Malware Analysis and Detection | 26/09/2023 |  |
| 9 | Firewall Configuration and  Rule-based Filtering | 03/10/2023 |  |

**Practical No: 1**

**Aim**: Implementing Substitution and Transposition Ciphers:

Design and implement algorithms to encrypt and decrypt messages using classical substitution and transposition techniques.

**Code:**

1. **Caesar Cipher:**

def encrypt(string, shift): cipher = ''

for char in string: if char == ' ':

cipher = cipher + char elif char.isupper():

cipher = cipher + chr((ord(char) + shift - 65) % 26 + 65) else:

cipher = cipher + chr((ord(char) + shift - 97) % 26 + 97) return cipher

def decrypt(string, shift): cipher = ''

for char in string: if char == ' ':

cipher = cipher + char elif char.isupper():

cipher = cipher + chr((ord(char) + (26-shift) - 65) % 26 + 65) else:

cipher = cipher + chr((ord(char) + (26-shift) - 97) % 26 + 97) return cipher

text = input("Enter String : ")

s = int(input("enter Shift Number : "))

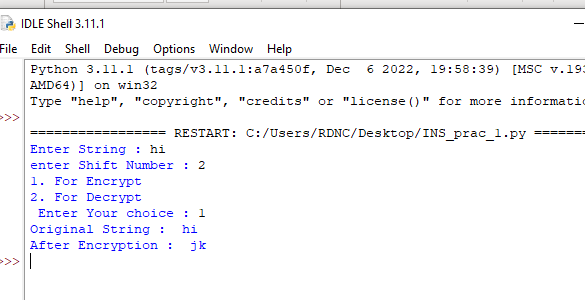
option = int(input("1. For Encrypt \n2. For Decrypt\n Enter Your choice : ")) print("Original String : ", text)

if( option == 1):

print("After Encryption : ", encrypt(text, s)) else:

print("After Decryption : ", decrypt(text, s))

**Output:**



1. **Mono alphabetic cipher:**

import random

alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ" keyword = "ZYXWVUTSRQPONMLKJIHGFEDCBA"

def encrypt(Plaintext): result = ""

for char in Plaintext:

num = alphabet.find(char) result = result+keyword[num]

print(result)

def decrypt(Ciphertext): result=""

for char in Ciphertext:

num = keyword.find(char)

result = result + alphabet[num] print(result)

while True:

n = int(input("Enter Value:: \n1) To Encrypt Text:: \n2) To Decrypt Text:: \n3) See Key::

\n4) To Exit \n")) if(n == 1):

Plaintext = str(input("Enter Text::")) encrypt(Plaintext.upper())

elif(n == 2):

Ciphertext = str(input("Enter Crypt Text::")) decrypt(Ciphertext.upper()

elif(n == 3): print(keyword)

elif(n == 4): break else:

Print("Invalid Input; Enter Again!!")

**Output:**



1. **Playfair Cipher:** key = input("Enter key : ") key = key.replace(" ", "") key = key.upper()

def matrix(x, y, initial):

return [[initial for i in range(x)] for j in range(y)]

result = list() for c in key:

if c not in result: if c == 'J':

result.append('I') else:

result.append(c) flag = 0

for i in range(65, 91): if chr(i) not in result:

if i == 73 and chr(74) not in result: result.append("I")

flag = 1

elif flag == 0 and i == 73 or i == 74: pass

else:

result.append(chr(i))

k = 0

my\_matrix = matrix(5, 5, 0) for i in range(0, 5):

for j in range(0, 5): my\_matrix[i][j] = result[k] k += 1

def locindex(c): loc = list()

if c == 'J': c = 'I'

for i, j in enumerate(my\_matrix): for k, l in enumerate(j):

if c == l: loc.append(i) loc.append(k) return loc

def encrypt():

msg = str(input("ENTER MSG : ")) msg = msg.upper()

msg = msg.replace(" ", "") i = 0

for s in range(0, len(msg) + 1, 2): if s < len(msg) - 1:

if msg[s] == msg[s + 1]:

msg = msg[:s + 1] + 'X' + msg[s + 1:] if len(msg) % 2 != 0:

msg = msg[:] + 'X' print("CIPHER TEXT:", end=' ')

while i < len(msg): loc = list()

loc = locindex(msg[i]) loc1 = list()

loc1 = locindex(msg[i + 1]) if loc[1] == loc1[1]:

print("{}{}".format(my\_matrix[(loc[0] + 1) % 5][loc[1]], my\_matrix[(loc1[0] + 1) % 5][loc1[1]]), end=' ')

elif loc[0] == loc1[0]:

print("{}{}".format(my\_matrix[loc[0]][(loc[1] + 1) % 5],

my\_matrix[loc1[0]][(loc1[1] + 1) % 5]), end=' ') else:

print("{}{}".format(my\_matrix[loc[0]][loc1[1]], my\_matrix[loc1[0]][loc[1]]), end=' ') i = i + 2

def decrypt():

msg = str(input("ENTER CIPHER TEXT:")) msg = msg.upper()

msg = msg.replace(" ", "")

print("PLAIN TEXT:", end=' ') i = 0

while i < len(msg): loc = list()

loc = locindex(msg[i]) loc1 = list()

loc1 = locindex(msg[i + 1]) if loc[1] == loc1[1]:

print("{}{}".format(my\_matrix[(loc[0] - 1) % 5][loc[1]], my\_matrix[(loc1[0] - 1) % 5][loc1[1]]), end=' ')

elif loc[0] == loc1[0]:

print("{}{}".format(my\_matrix[loc[0]][(loc[1] - 1) % 5], my\_matrix[loc1[0]][(loc1[1]

- 1) % 5]), end=' ')

else:

print("{}{}".format(my\_matrix[loc[0]][loc1[1]], my\_matrix[loc1[0]][loc[1]]), end=' ') i = i + 2

while (1):

choice = int(input("\n 1.Encryption \n 2.Decryption: \n 3.EXIT \n Enter Your Choice: \n ")) if choice == 1:

encrypt()

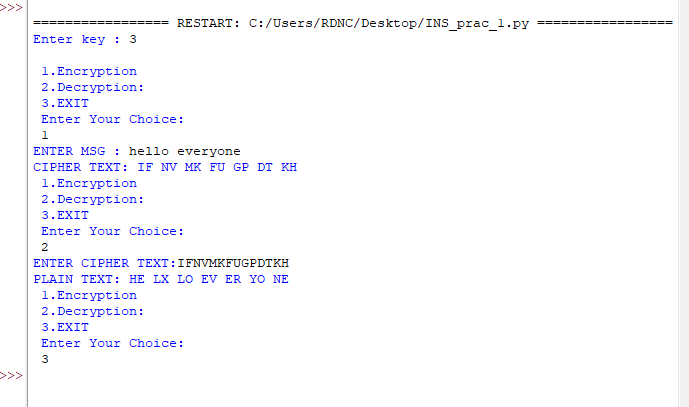
elif choice == 2: decrypt()

elif choice == 3: exit()

else:

print("Choose correct choice")

**Output:**



1. **Vernam Cipher:**

def Vernam(Plain,Key,Flag): result=""

for i in range(len(Plain)): char=Plain[i]

if(Flag):

result+=chr((ord(char)-97 +ord(Key[i])-97)%26 +97) else:

result += chr((ord(char) - ord(Key[i])+26) % 26 + 97) return result

# assumption:- for simplicity we are only considering lowercase-values and without spaces

if name ==" main\_\_": Key=''.join(input("Enter Key: ").lower().split())

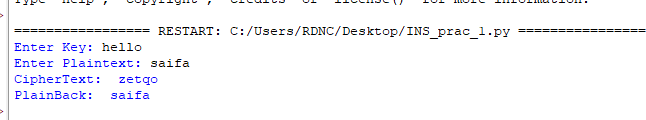
Plain=''.join(input("Enter Plaintext: ").lower().split()) if(len(Key)!=len(Plain)):

print("Invalid Key!") exit(None)

CipherText=Vernam(Plain,Key,True) print("CipherText: ",CipherText)

print("PlainBack: ",Vernam(CipherText,Key,False))

Output:



1. **Playfair Cipher:** key = input("Enter key : ") key = key.replace(" ", "") key = key.upper()

def matrix(x, y, initial):

return [[initial for i in range(x)] for j in range(y)]

result = list() for c in key:

if c not in result: if c == 'J':

result.append('I') else:

result.append(c) flag = 0

for i in range(65, 91): if chr(i) not in result:

if i == 73 and chr(74) not in result: result.append("I")

flag = 1

elif flag == 0 and i == 73 or i == 74: pass

else:

result.append(chr(i))

k = 0

my\_matrix = matrix(5, 5, 0) for i in range(0, 5):

for j in range(0, 5): my\_matrix[i][j] = result[k] k += 1

def locindex(c): loc = list()

if c == 'J': c = 'I'

for i, j in enumerate(my\_matrix): for k, l in enumerate(j):

if c == l: loc.append(i) loc.append(k) return loc

def encrypt():

msg = str(input("ENTER MSG : ")) msg = msg.upper()

msg = msg.replace(" ", "") i = 0

for s in range(0, len(msg) + 1, 2): if s < len(msg) - 1:

if msg[s] == msg[s + 1]:

msg = msg[:s + 1] + 'X' + msg[s + 1:] if len(msg) % 2 != 0:

msg = msg[:] + 'X' print("CIPHER TEXT:", end=' ')

while i < len(msg): loc = list()

loc = locindex(msg[i]) loc1 = list()

loc1 = locindex(msg[i + 1]) if loc[1] == loc1[1]:

print("{}{}".format(my\_matrix[(loc[0] + 1) % 5][loc[1]], my\_matrix[(loc1[0] + 1) % 5][loc1[1]]), end=' ')

elif loc[0] == loc1[0]: print("{}{}".format(my\_matrix[loc[0]][(loc[1] + 1) % 5],

my\_matrix[loc1[0]][(loc1[1] + 1) % 5]), end=' ')

else:

print("{}{}".format(my\_matrix[loc[0]][loc1[1]], my\_matrix[loc1[0]][loc[1]]), end=' ') i = i + 2

def decrypt():

msg = str(input("ENTER CIPHER TEXT:")) msg = msg.upper()

msg = msg.replace(" ", "")

print("PLAIN TEXT:", end=' ') i = 0

while i < len(msg): loc = list()

loc = locindex(msg[i]) loc1 = list()

loc1 = locindex(msg[i + 1]) if loc[1] == loc1[1]:

print("{}{}".format(my\_matrix[(loc[0] - 1) % 5][loc[1]], my\_matrix[(loc1[0] - 1) % 5][loc1[1]]), end=' ')

elif loc[0] == loc1[0]:

print("{}{}".format(my\_matrix[loc[0]][(loc[1] - 1) % 5], my\_matrix[loc1[0]][(loc1[1]

- 1) % 5]), end=' ')

else:

print("{}{}".format(my\_matrix[loc[0]][loc1[1]], my\_matrix[loc1[0]][loc[1]]), end=' ') i = i + 2

while (1):

choice = int(input("\n 1.Encryption \n 2.Decryption: \n 3.EXIT \n Enter Your Choice: \n ")) if choice == 1:

encrypt()

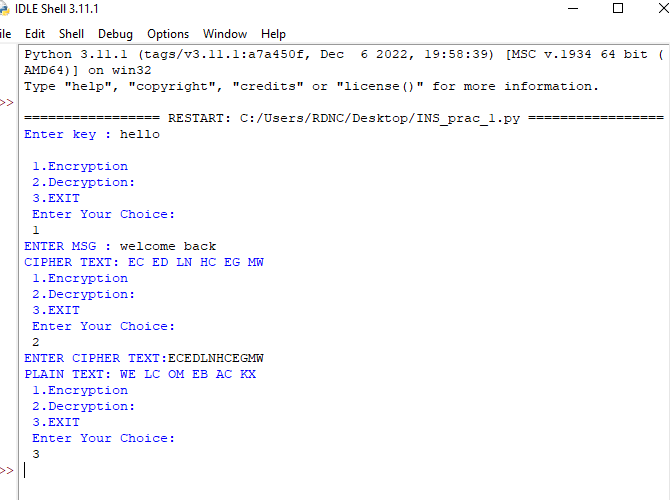
elif choice == 2: decrypt()

elif choice == 3: exit()

else:

print("Choose correct choice")

**Output:**



1. **Rail fence Cipher:**

def sequence(n): arr=[]

i=0 while(i<n-1):

arr.append(i) i+=1

while(i>0): arr.append(i) i-=1

return(arr)

def railfence(s,n): s = s.lower()

L = sequence(n)

print("The raw sequence of indices: ",L) temp=L

while(len(s)>len(L)):

L=L+temp

for i in range(len(L)-len(s)): L.pop()

print("The row indices of the characters in the given string: ",L) print("Transformed message for encryption: ",s)

num=0 cipher\_text="" while(num<n):

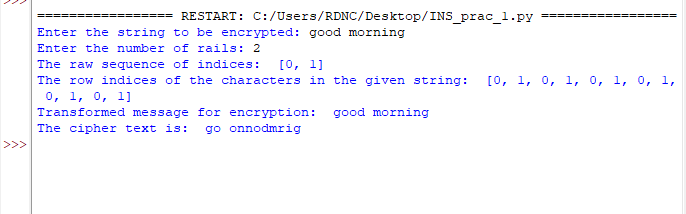
for i in range(L.count(num)): cipher\_text=cipher\_text+s[L.index(num)]

L[L.index(num)]=n num+=1

print("The cipher text is: ",cipher\_text)

plain\_text=input("Enter the string to be encrypted: ") n=int(input("Enter the number of rails: ")) railfence(plain\_text,n)

**Output:**



# Practical No: 2

**Aim:**

RSA Encryption and Decryption:

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

**Code:**

from Crypto.PublicKey import RSA

from Crypto.Cipher import PKCS1\_OAEP import binascii

keyPair = RSA.generate(1024)

pubKey = keyPair.publickey()

print(f"Public key: (n={hex(pubKey.n)}, e={hex(pubKey.e)})") pubKeyPEM = pubKey.export\_key() print(pubKeyPEM.decode('utf-8'))

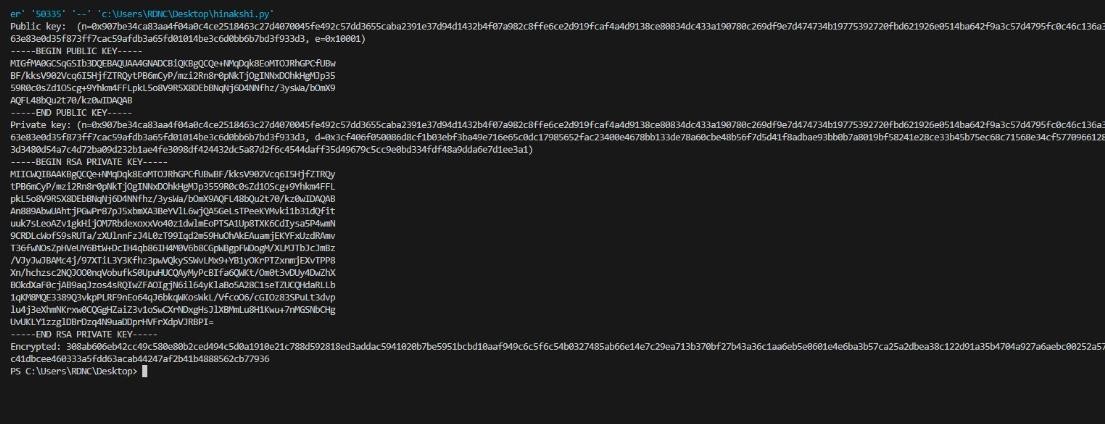
print(f"Private key: (n={hex(pubKey.n)}, d={hex(keyPair.d)})") privKeyPEM = keyPair.export\_key() print(privKeyPEM.decode('utf-8'))

msg = b'Hello Class'

encryptor = PKCS1\_OAEP.new(pubKey) encrypted = encryptor.encrypt(msg)

print("Encrypted:", binascii.hexlify(encrypted).decode('utf-8'))

**Output:**



# Practical No: 3

**Aim:** Message Authentication Codes: Implement algorithms to generate and verify message authentication codes (MACs) for ensuring data integrity and authenticity.

**Code:**

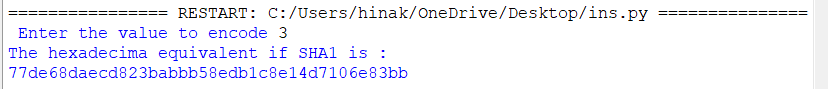
**SHA:**

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:**



## MD5:

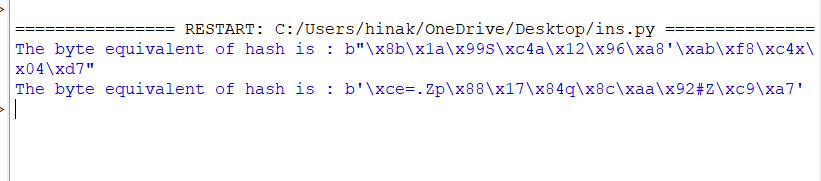
import hashlib

result = hashlib.md5(b'Hello') result1 = hashlib.md5(b'Fello')

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

print("The byte equivalent of hash is : ", end ="") print(result1.digest()

**Output:**



# Practical No. 4

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

**Code:**

from Crypto.PublicKey import RSA from Crypto.Signature import pkcs1\_15 from Crypto.Hash import SHA256

key = RSA.generate(2048) private\_key = key.export\_key()

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

original\_document = b"This is the original document content." modified\_document = b"This is the modified document content." original\_hash = SHA256.new(original\_document) modified\_hash = SHA256.new(modified\_document)

signature = pkcs1\_15.new(RSA.import\_key(private\_key)).sign(original\_hash) 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 No: 5

**Aim:** Key Exchange using Diffie-Hellman: Implement the Diffie-Hellman key exchange algorithm to securely exchange keys between two entities over an insecure network.

**Code:**

from random import randint if name == ' main ':

q = 23

alpha = 9

print('The Value of q is :%d'%(q)) print('The Value of alpha is :%d'%(alpha))

XA = 4

print('Secret Number for Alice is :%d'%(XA)) YA = int(pow(alpha,XA,q))

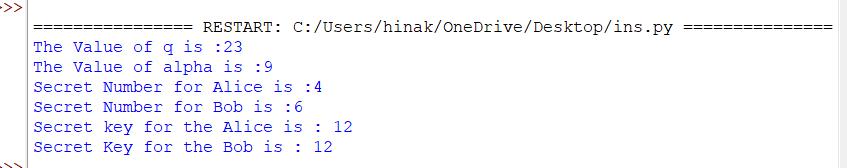
XB = 6

print('Secret Number for Bob is :%d'%(XB)) YB = int(pow(alpha,XB,q))

k1 = int(pow(YB,XA,q)) k2 = int(pow(YA,XB,q))

print('Secret key for the Alice is : %d'%(k1)) print('Secret Key for the Bob is : %d'%(k2))

**Output:**



# Practical No: 6

**Aim:-** IP Security (IPsec) Configuration: Configure IPsec on network devices to provide secure communication and protect against unauthorized access and attacks.

**Steps to be followed**:

Download Cisco Packet Tracer Requirements:

Take 3 Routers (1941)

Take 2 Switches (2960)

Take 2 PC Configuration:

PC 0 – 192.168.1.2

PC 1 – 192.168.2.2

Router 1 (G 0/0) - 20.0.0.1

Router 1 (G 0/1) - 192.168.1.1

Router 0 (G 0/0) - 30.0.0.2

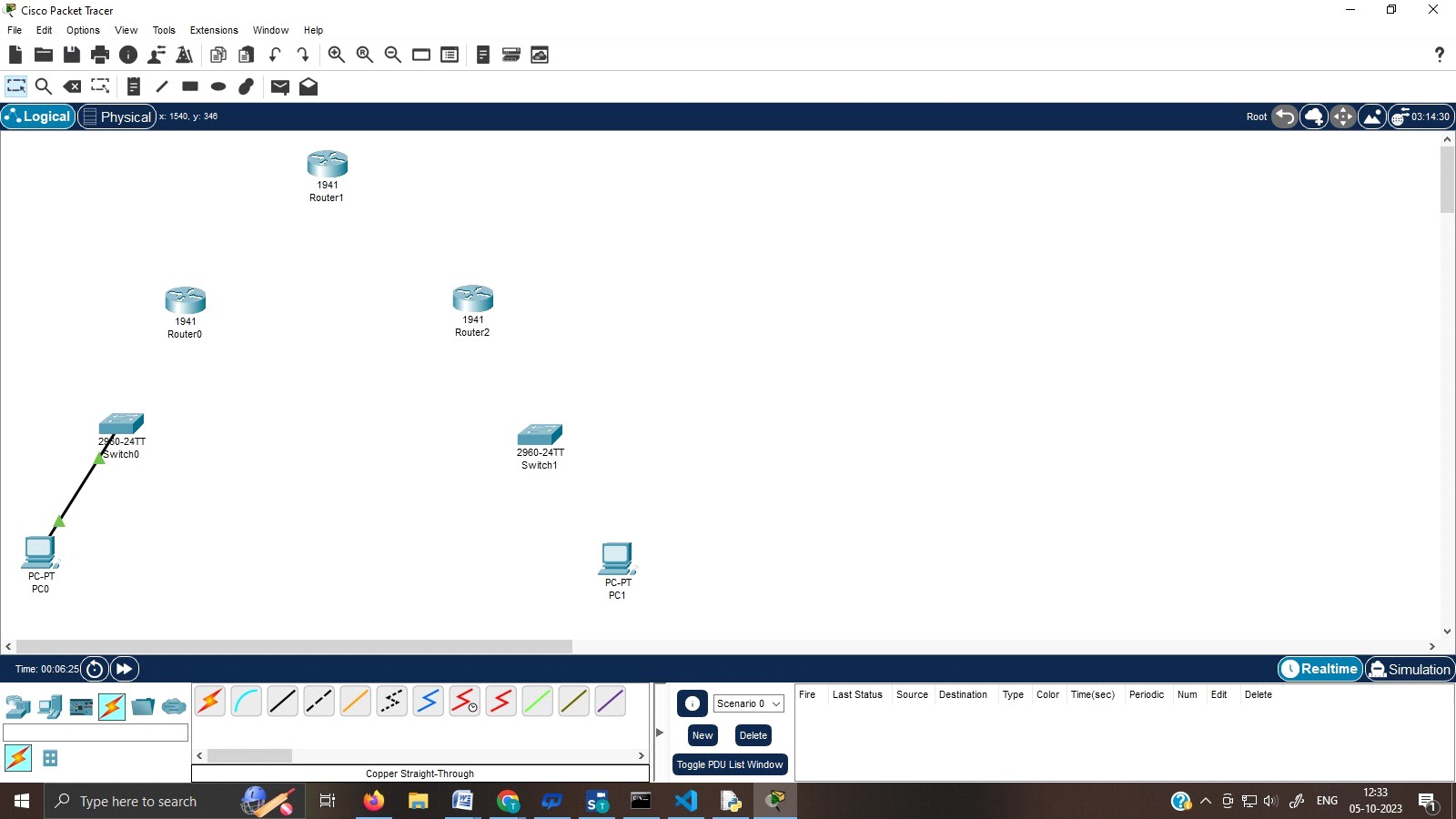
Router 0 (G 0/1) - 20.0.0.2

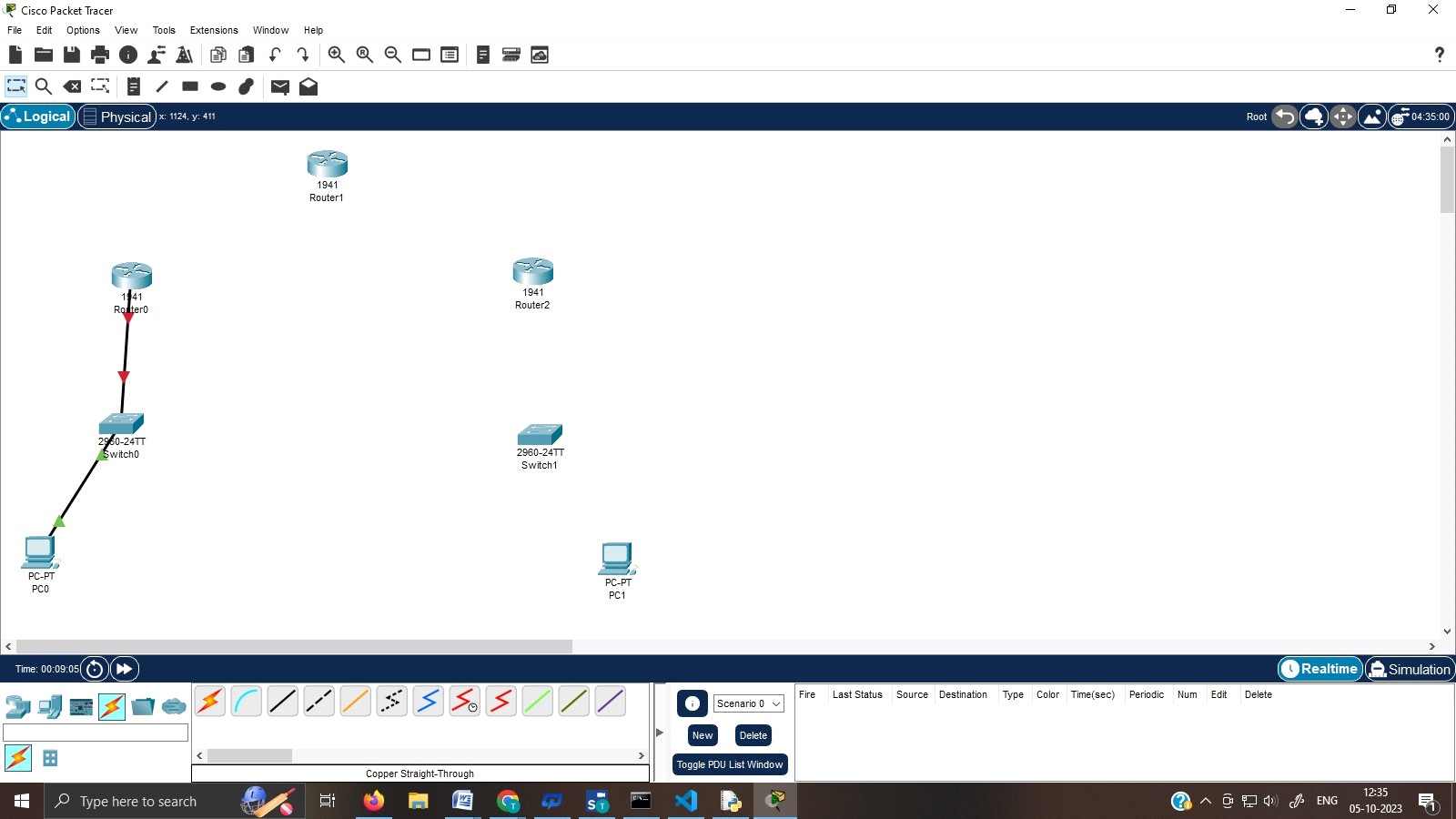
Router 2 (G 0/0) - 30.0.0.1

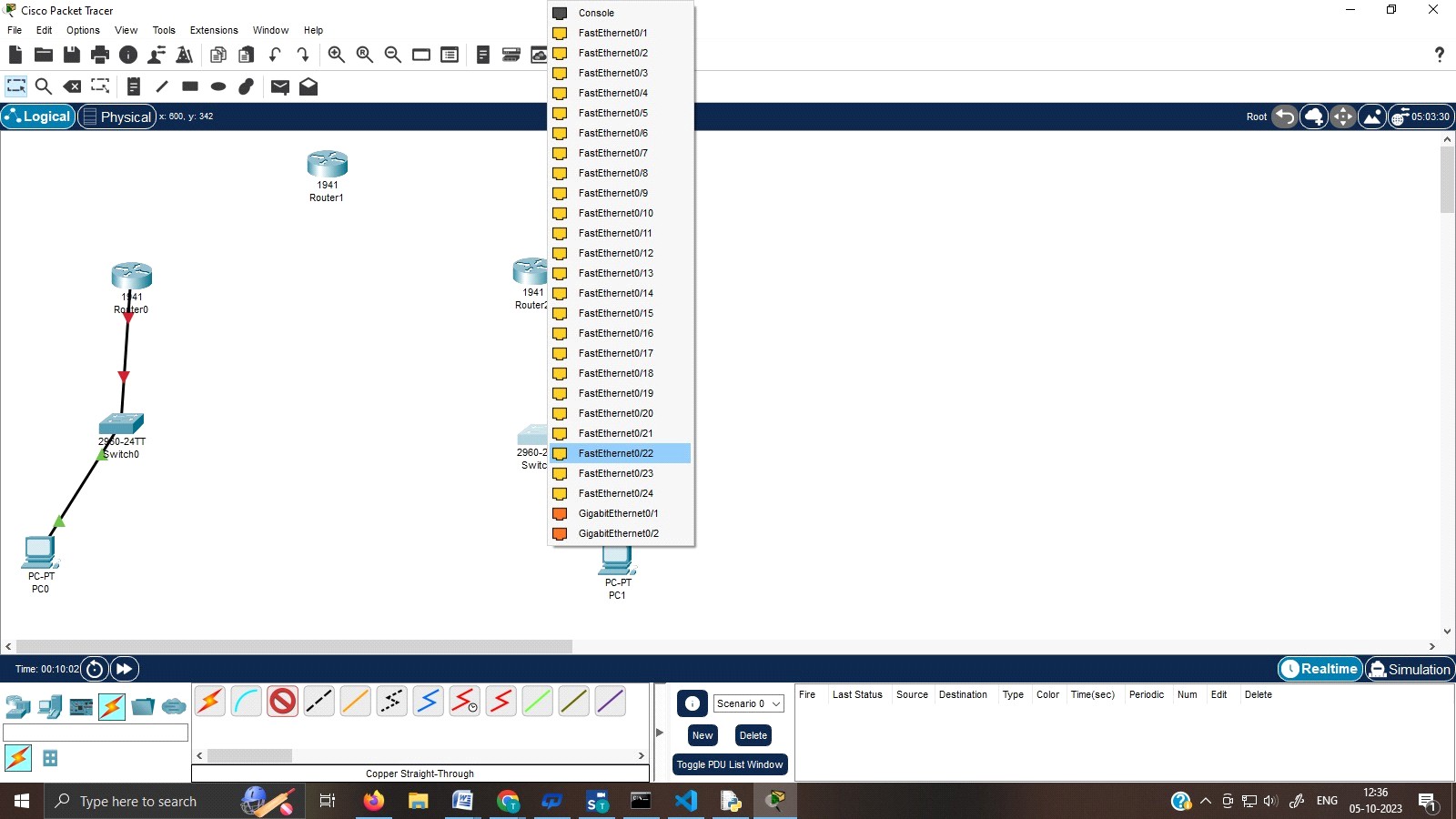
Router 2 (G 0/1) - 192.168.2.1

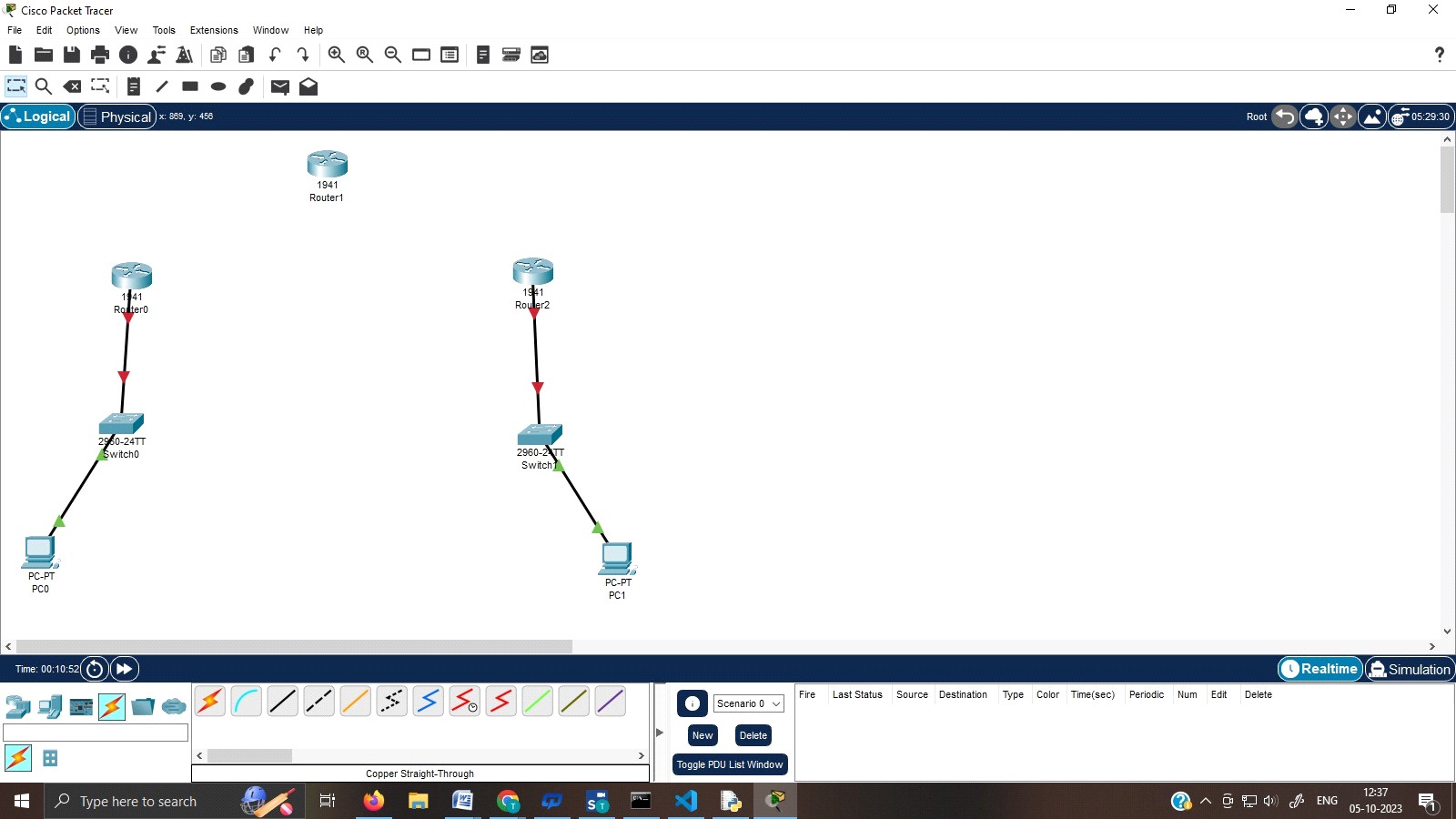
Step 1: Implementing the Topology using Cisco Packet Tracer, configure the IP address and set the IP route.

Connect the PC with Switches and also connect switches with Router using copper straight wire as given below

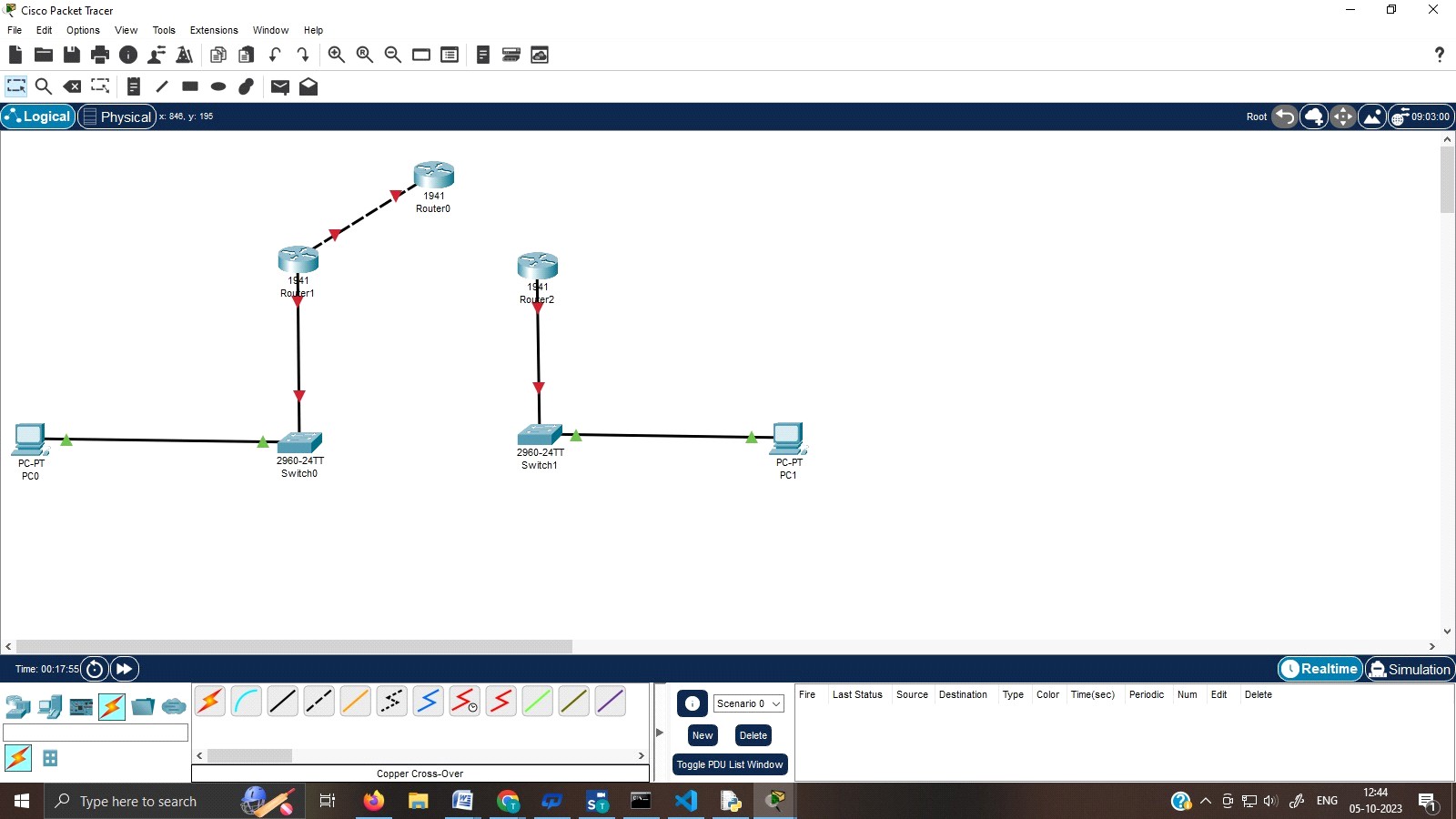


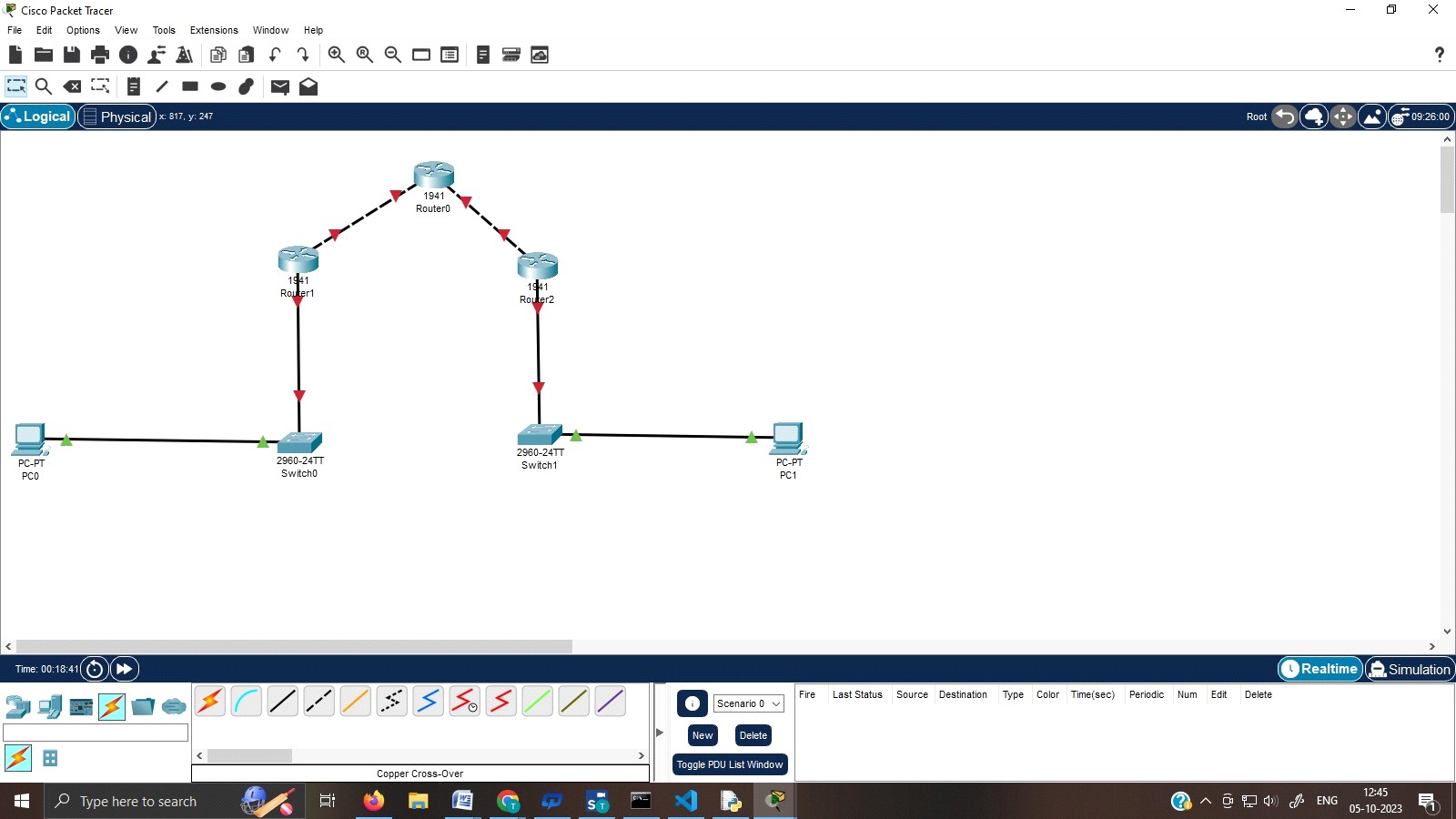




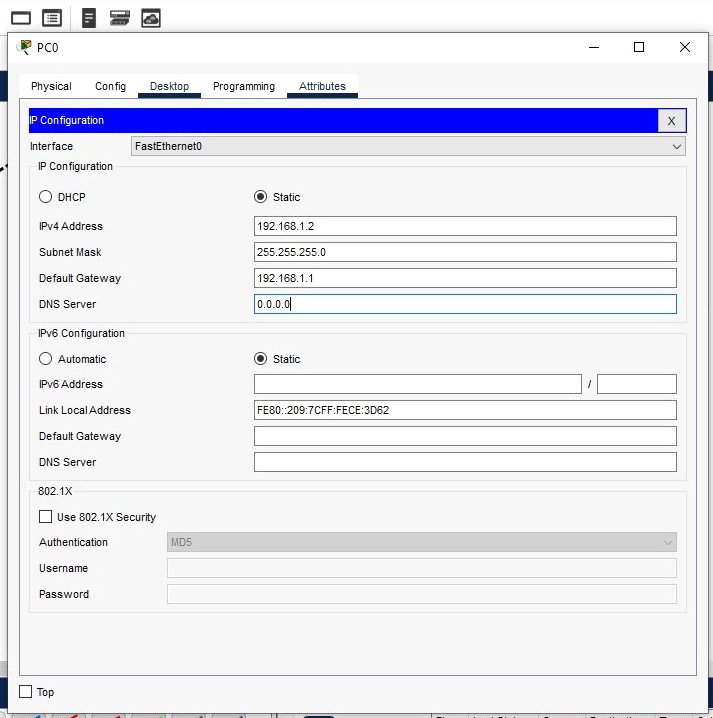


Connect the Router 1 with Router 0 and Router 0 with Router 2 using copper cross wire as given below.

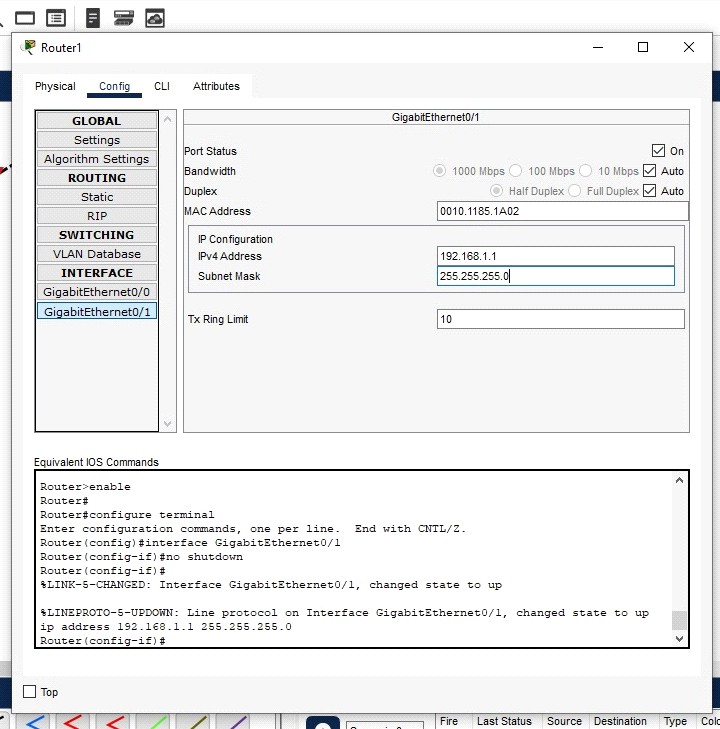


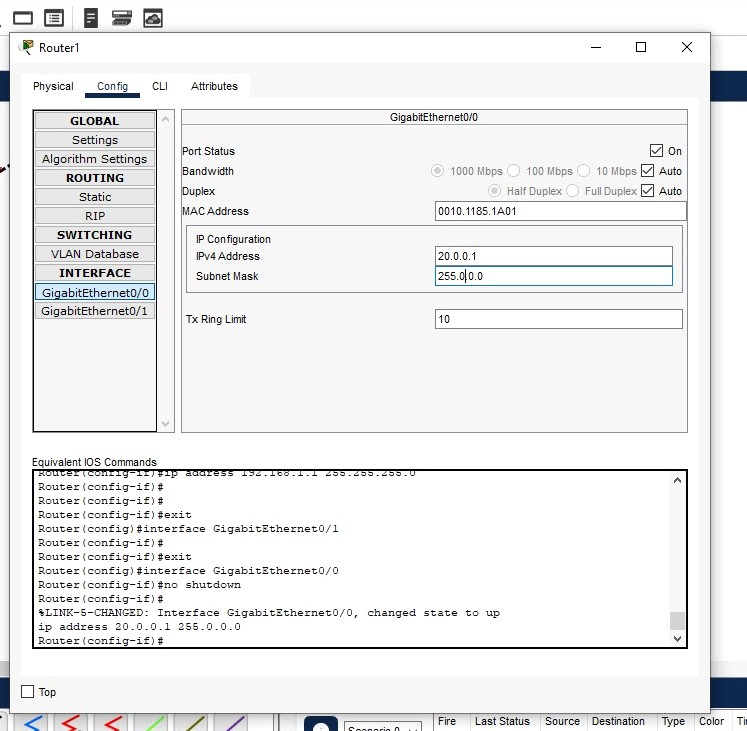


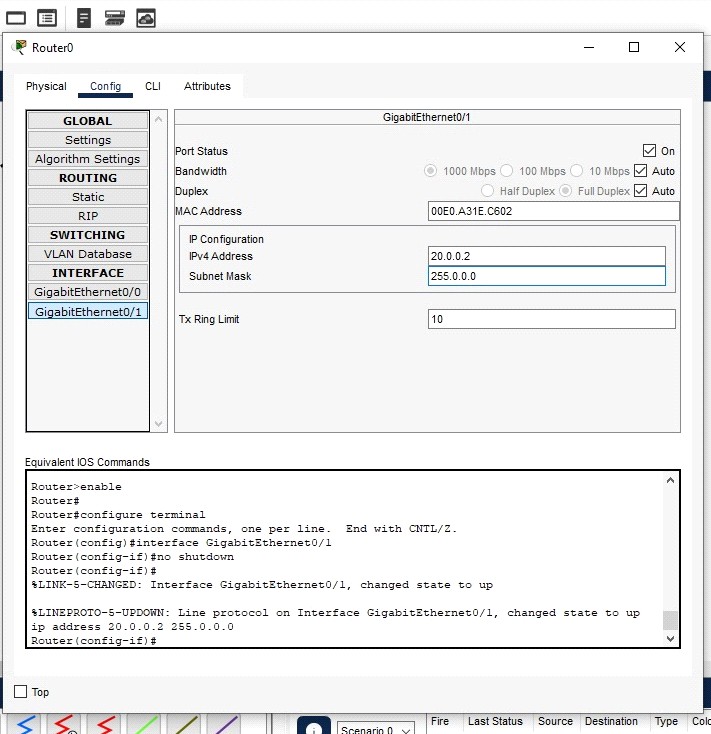
Provide the IP address to PC 0 and PC 1

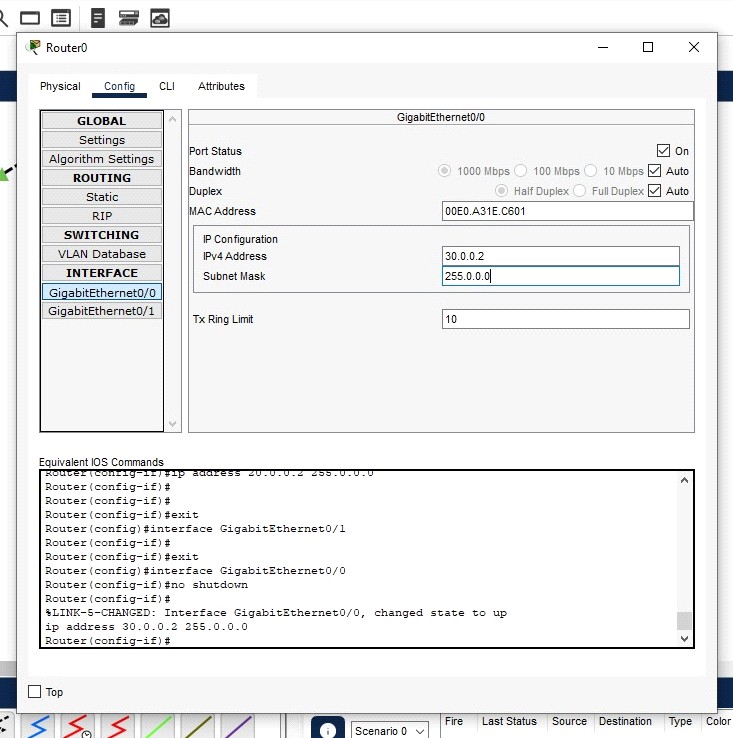


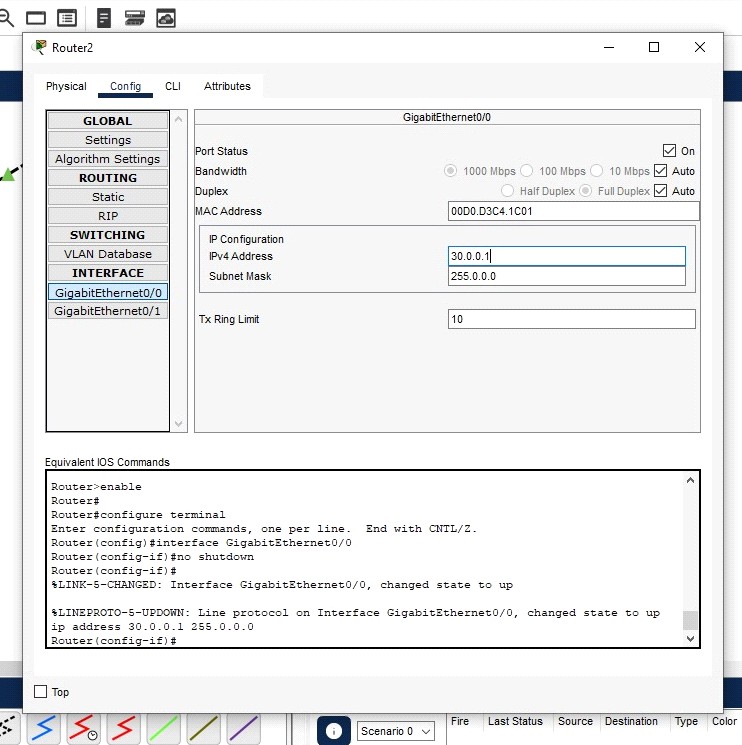
Provide the IP Configuration to the Routers

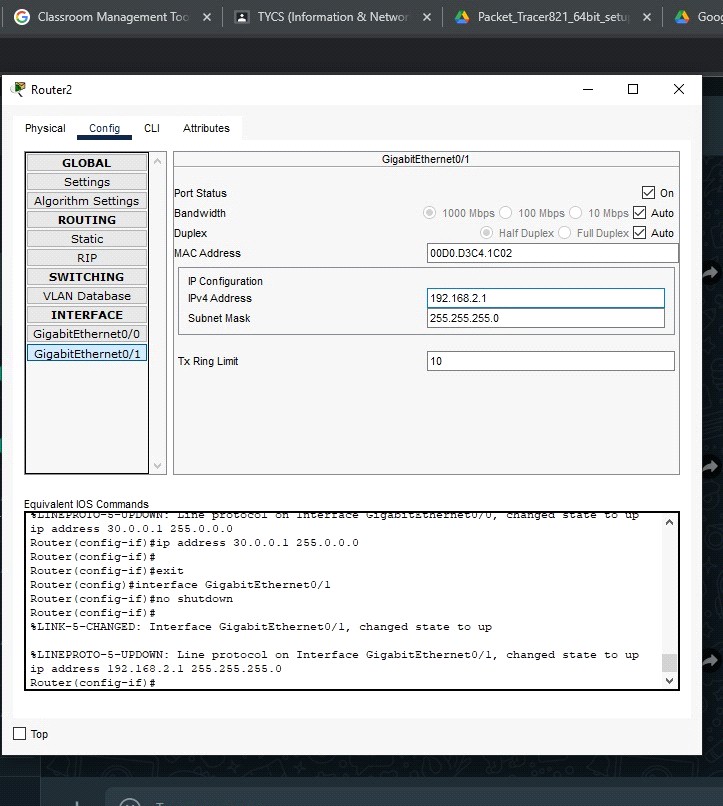




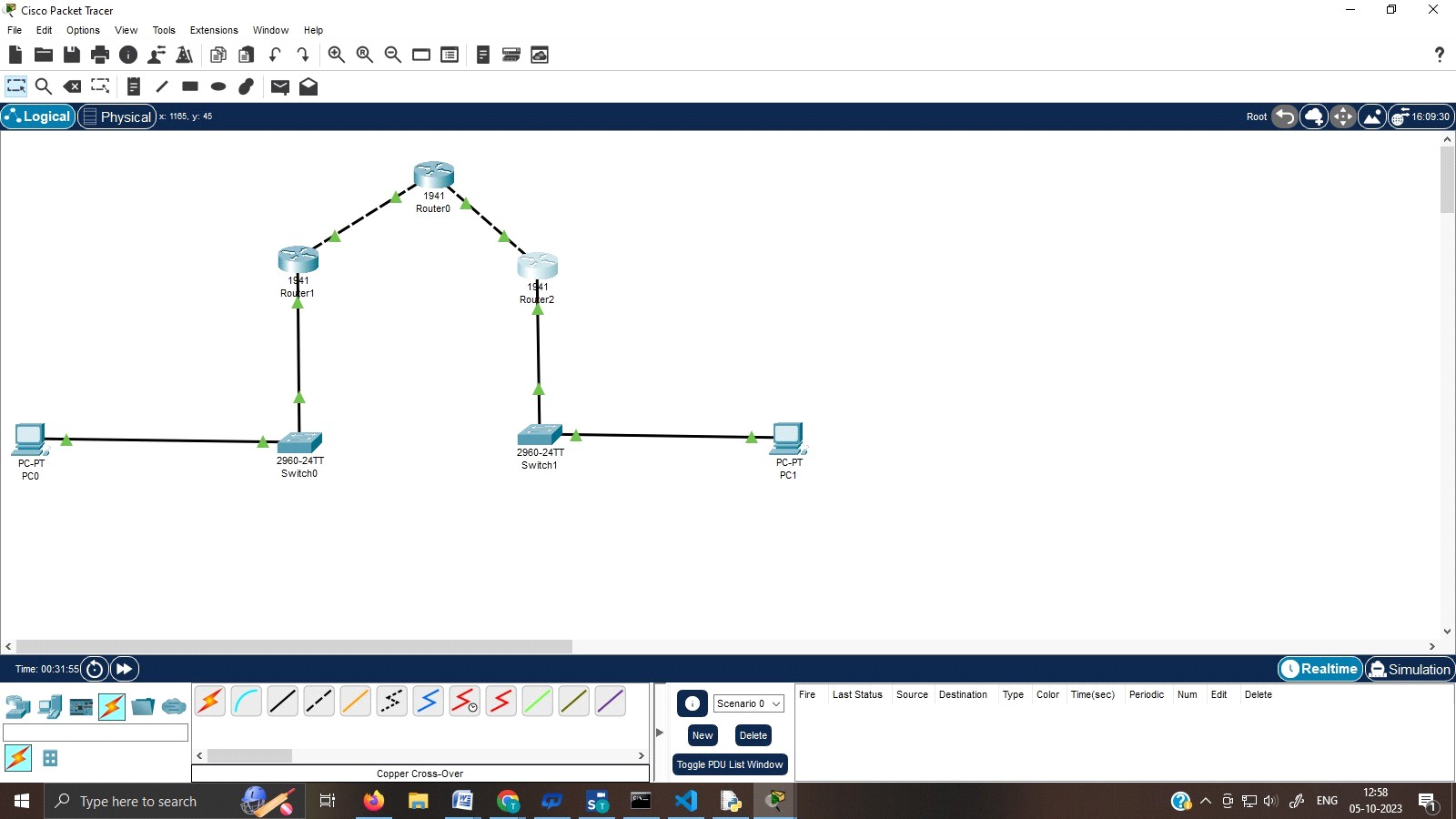




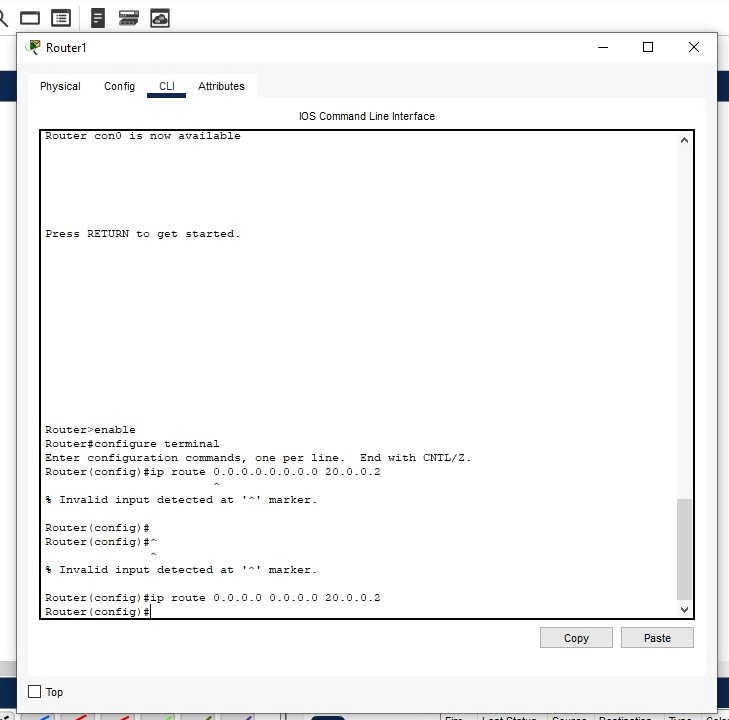




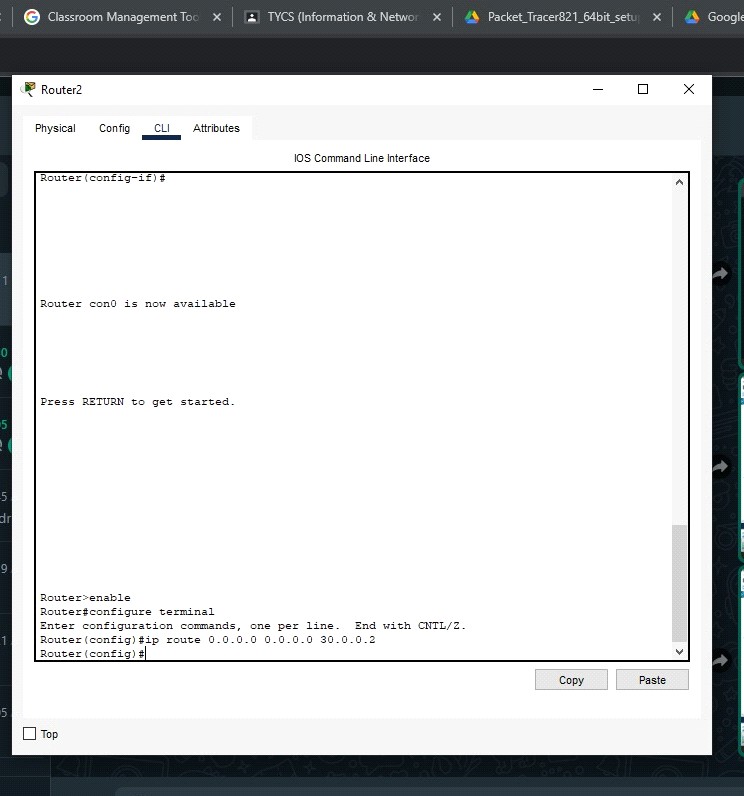
Now we can see all the connections are properly configured as given below (Green Arrow)



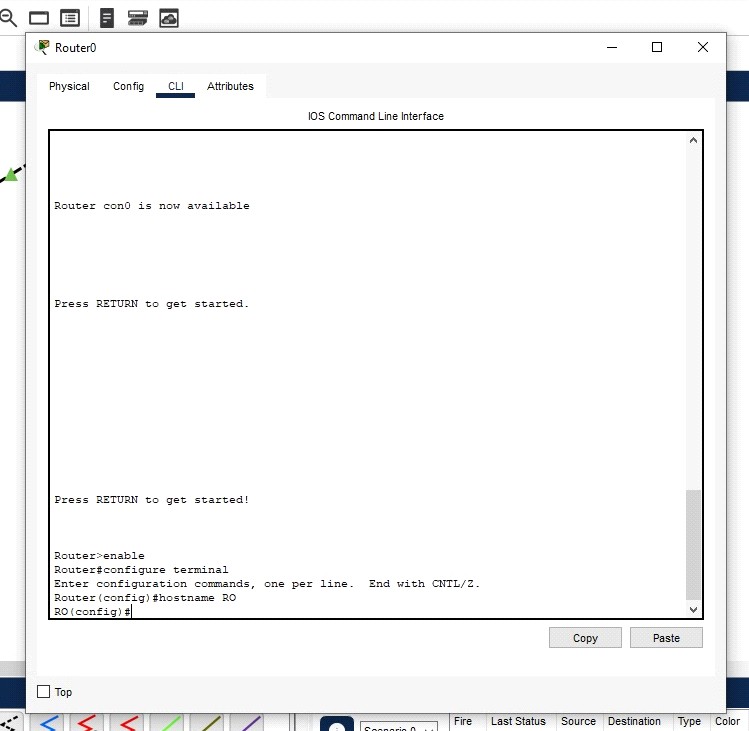
* We have to set the path from Router 0 - 20.0.0.2 to Router 1 and Router 0 – 30.0.0.2 to Router 2 as shown below



Click on the Router 1 and go to CLI mode follow the below steps.

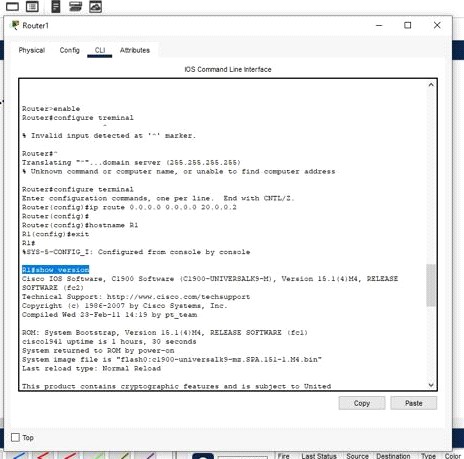


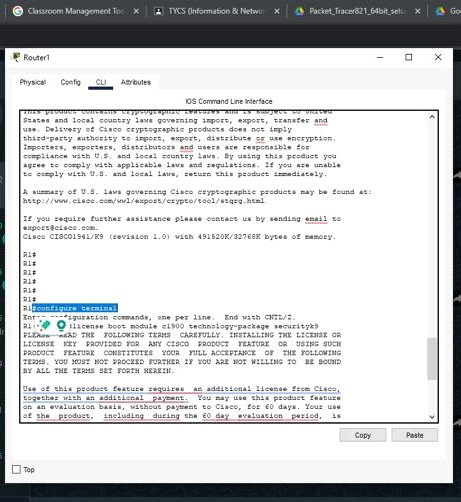
Step 2: Configure the Hostname on all Routers and enable the security package on R1 and R2, ping one PC from the other (all packets are lost)

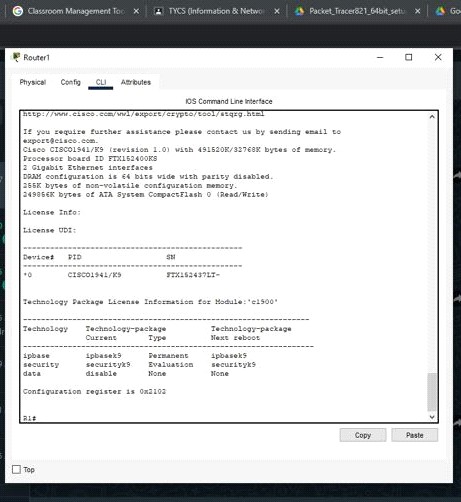
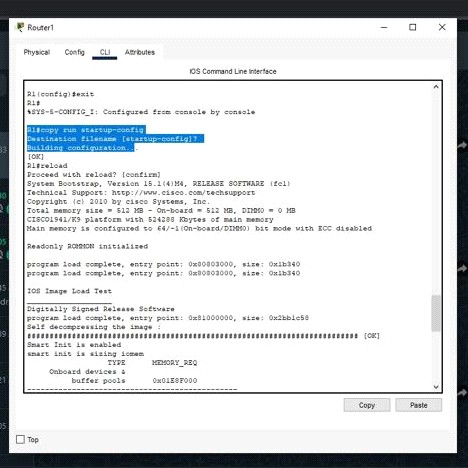


Enable the security packages on the Router 1 and Router 2.

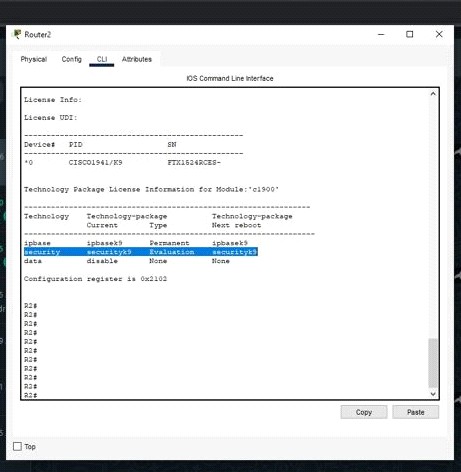
First of all, let us check whether the security version is enabled or not

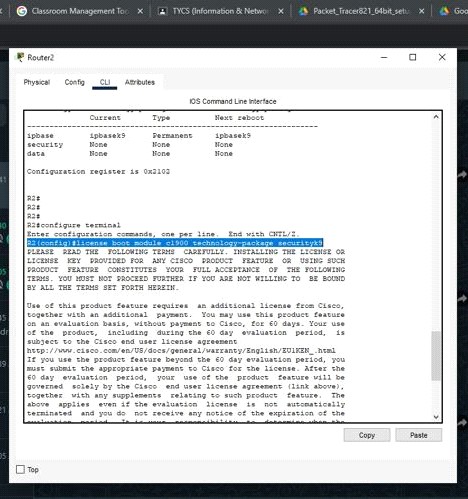


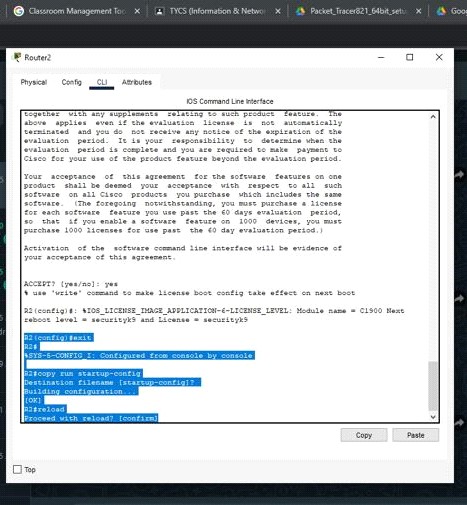


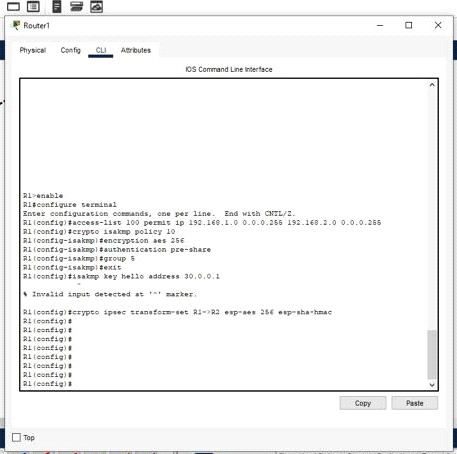


To enable the security package, follow the below commands.

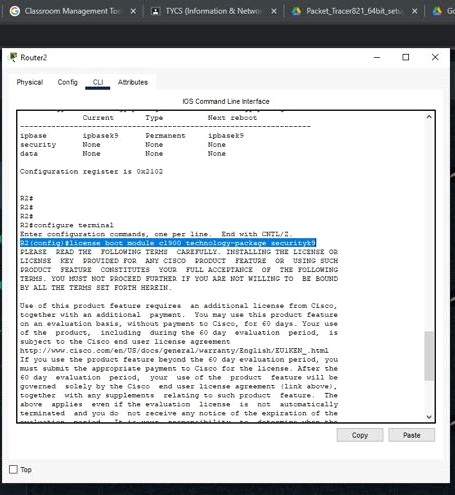
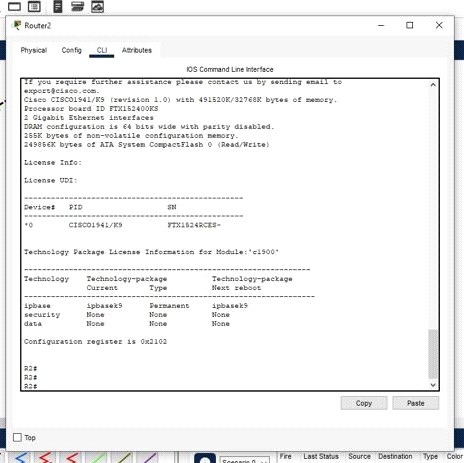


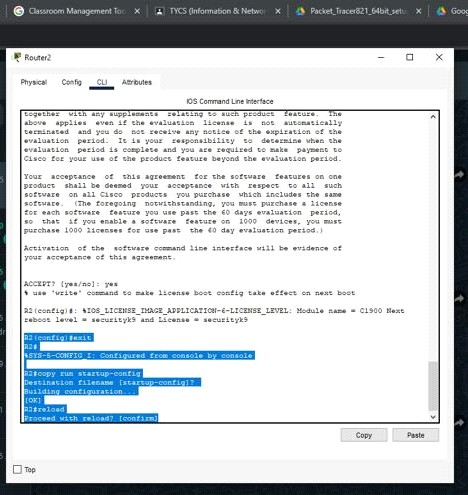


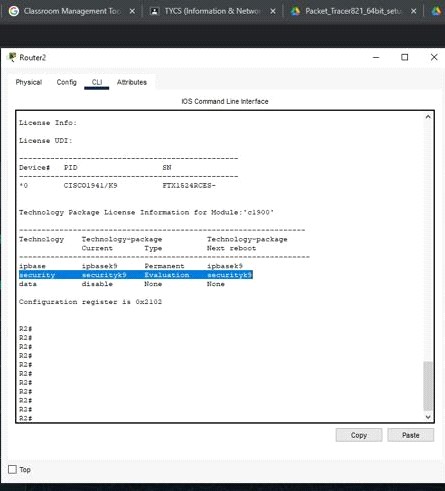


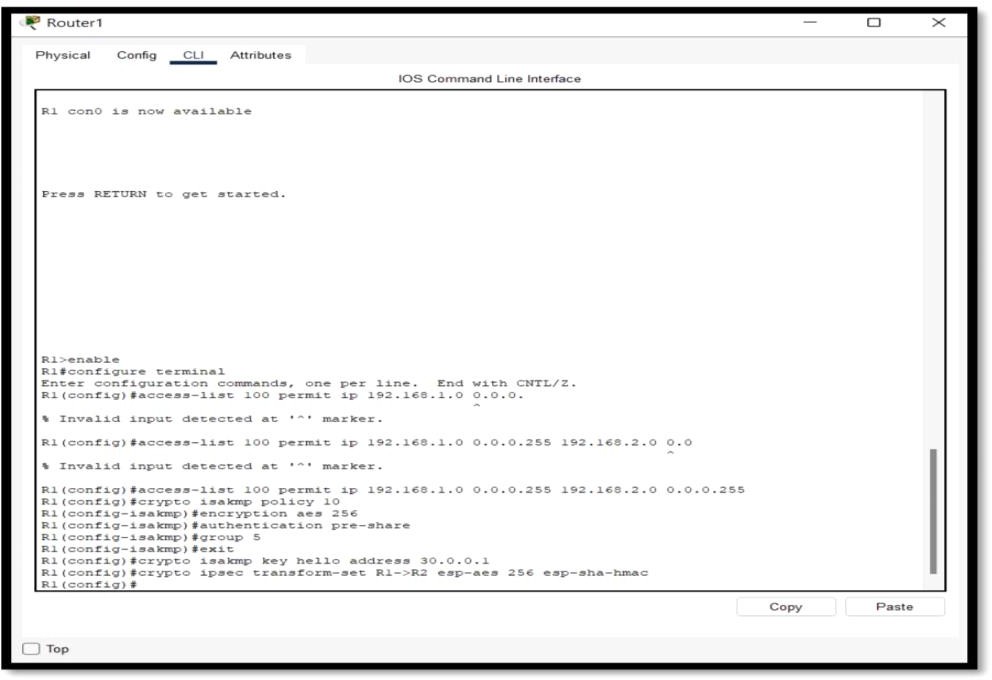


Follow the same steps for Router 2.







Step 3: Apply the Access Control List (ACL) at Router 1 and Router 2, Set the ISAKMP policy and ISAKMP key, Set IPsec transform set.

Step 4: Create the Crypto map. Apply the Crypto map to the required interface.

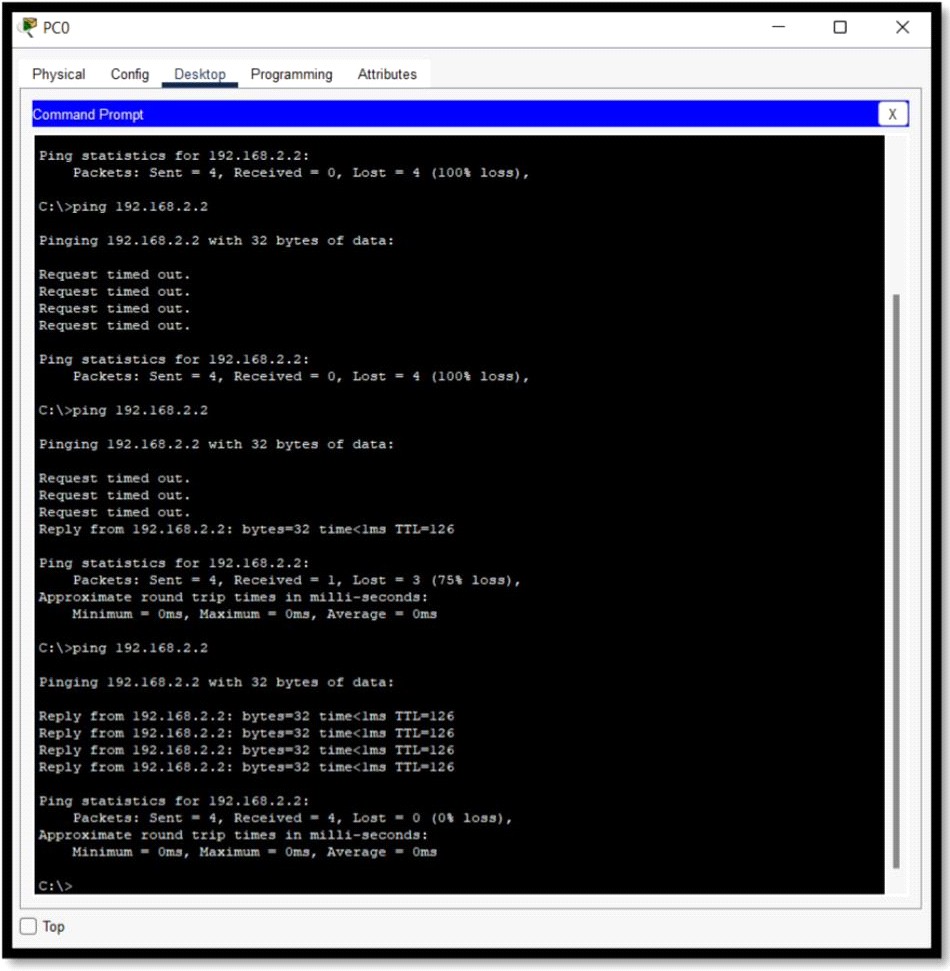




Step 5: Verify the output by pinging one PC from other (ping is successful) Now go to command prompt of PC0 and ping 192.168.2.2

It will successfully generate the output now

**OUTPUT:**



# Practical No: 7

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

We implement this using a simple echo socket server in python.

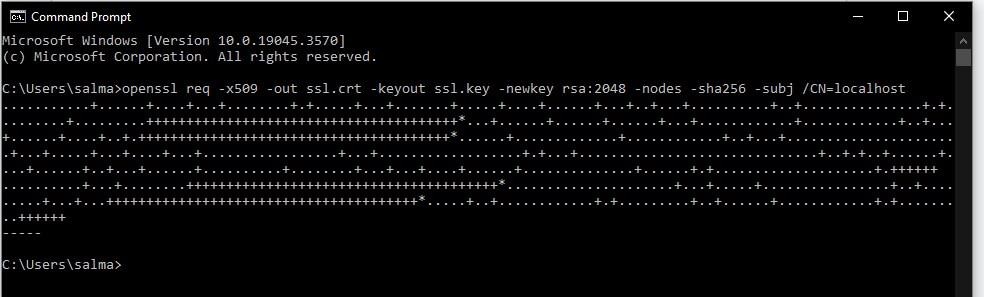
**Pre requisites:**

* Python 3
* OpenSSL 3.1:

https://kb.firedaemon.com/support/solutions/articles/4000121705#Download-OpenS SL (Make sure to add the bin folder, usually **C:\Program Files\FireDaemon**

**OpenSSL 3\bin** to your environment variable path)

## Step 1: Use the following command to generate a new self-signed SSL certificate and key for localhost:

 openssl req -x509 -out localhost.crt -keyout localhost.key -newkey rsa:2048 -nodes - sha256 -subj /CN=localhost



This command creates a certificate (ssl.crt) and key (ssl.key) file to be used for the secure session. Keep these files together with the below-created python files

## Step 2 : Create the SSL socket server (server.py) with the following code:

import socket import ssl context = ssl.SSLContext(ssl.PROTOCOL\_TLS\_SERVER) context.load\_cert\_chain(certfile="ssl.crt", keyfile="ssl.key")

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as server: server.bind(("", 4434)) server.listen(5)

print("Server ready and listening for connections")

# Wait for new connections in a loop while True:

sock, address = server.accept() print("New connection from", f"{address[0]}:{address[1]}")

# Wrap socket with ssl ssl\_sock = context.wrap\_socket(sock, server\_side=True)

while True:

data = ssl\_sock.recv(1024) # Decode byte array to utf-8 string decoded = data.decode('utf-8')

# Close the socket if the sock sends empty bytes if decoded == "":

break

# Log what the sock sends print(f"[{address[0]}:{address[1]}] {decoded}")

# Echo the data back to the sock ssl\_sock.sendall(data)

# Gracefully close the connection and wait for next one print("Closing connection with", f"{address[0]}:{address[1]}") ssl\_sock.close()

## Step 3 : Create the SSL socket client (client.py) with the following code

import socket import ssl

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as sock: sock.settimeout(10) # Wrap socket with ssl

context = ssl.SSLContext(ssl.PROTOCOL\_TLS\_CLIENT) context.load\_verify\_locations('ssl.crt') ssl\_sock =

context.wrap\_socket(sock, server\_hostname="localhost")

# Connect to the server ssl\_sock.connect(("localhost", 4434)) print("Connected to server")

# Send input data to server and wait for response in a loop while True:

ssl\_sock.sendall(bytes(input(">"), "utf-8")) data

= ssl\_sock.recv(1024)

print("Server responded:", data.decode('utf-8'))

## Step 4 : Run the server.py file

**Step 5 : Keep the server running, and run the client.py file Client Output :**

## Server Output :

**Step 6 : Test the connection by sending a message from the client by typing in the console.**



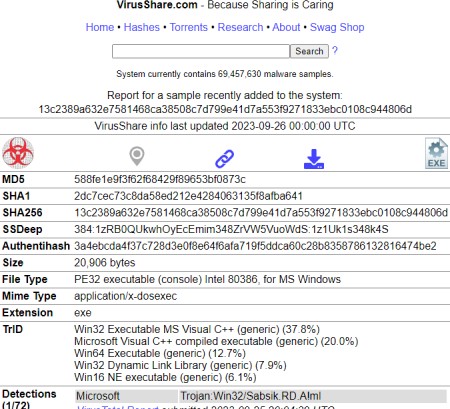


**The server will echo the same content of the message back to the client.**

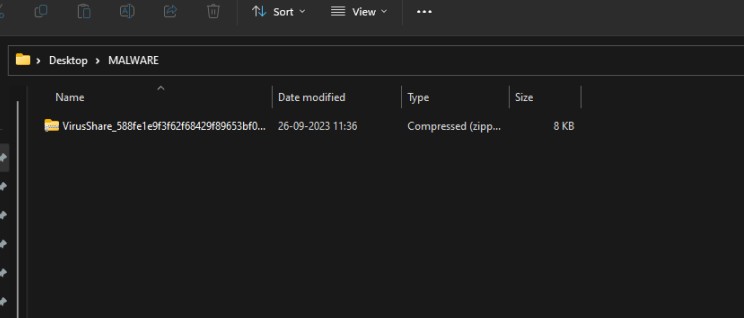
# Practical No: 8

**Aim:** Malware Detection and Analysis

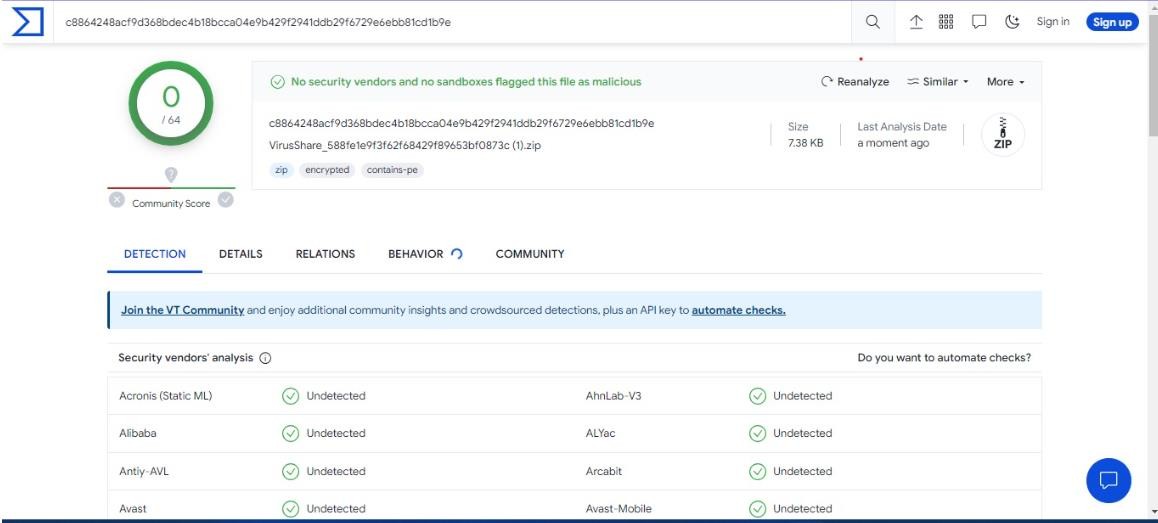
**Output:**



1. Create a folder “MALWARE” on desktop and paste the zip file in that folder.



1. Now scan the zip file on the website: [www.virustotal.com](http://www.virustotal.com/)



It is Undetected

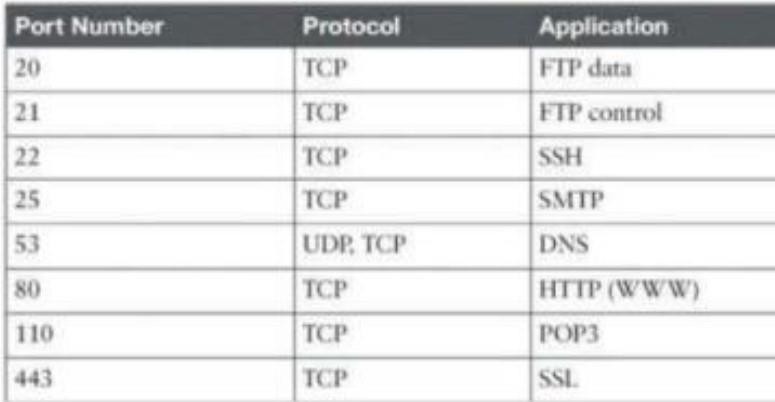
No Security vendors and no sandboxes flagged this file as malicious.

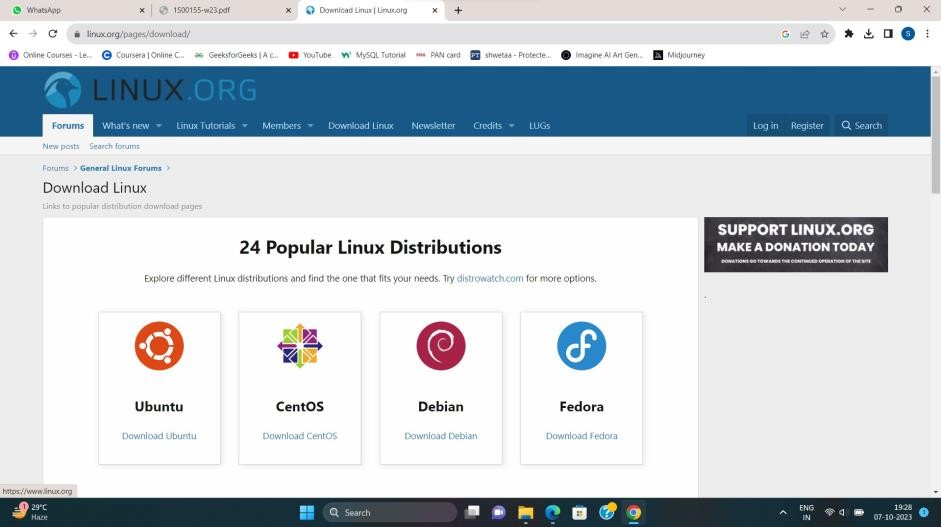
# Practical No: 9

**AIM:** 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.

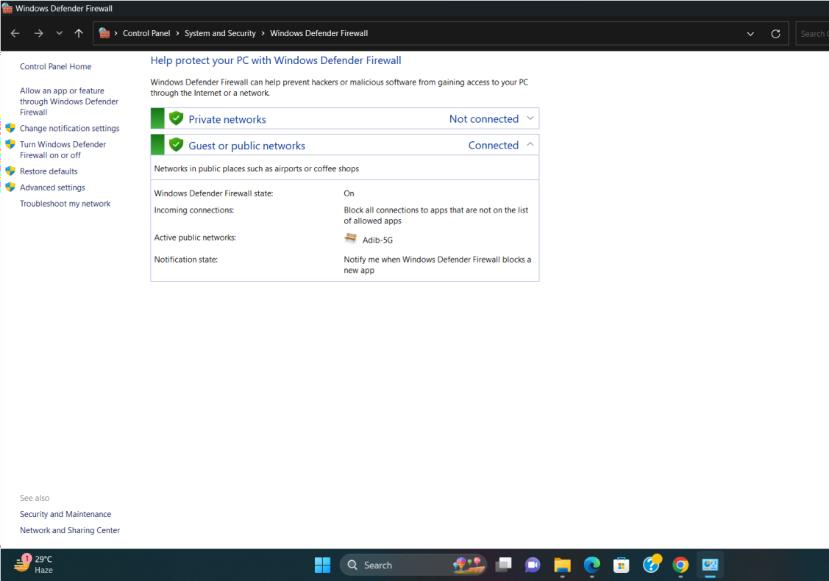
We would use a firewall to block

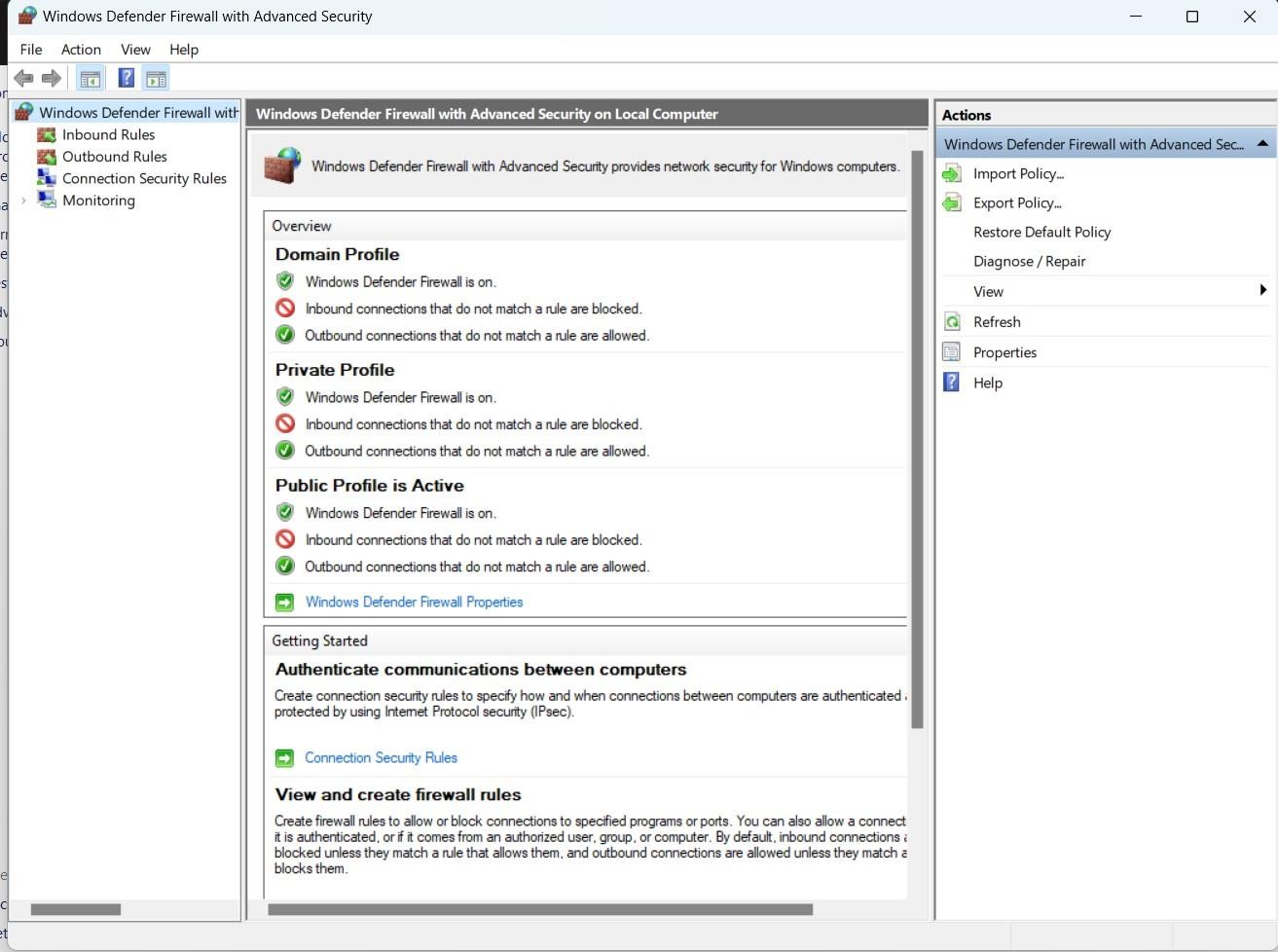
1. A Port
2. A Program
3. A Website

**Part 1:** Blocking the HTTP and HTTPS (Port 80 and Port 443) using the Firewall Before starting with the blocking port process, we note that the applications running at the server end are identified with the well-known Port numbers, some of the commonly used are as follows

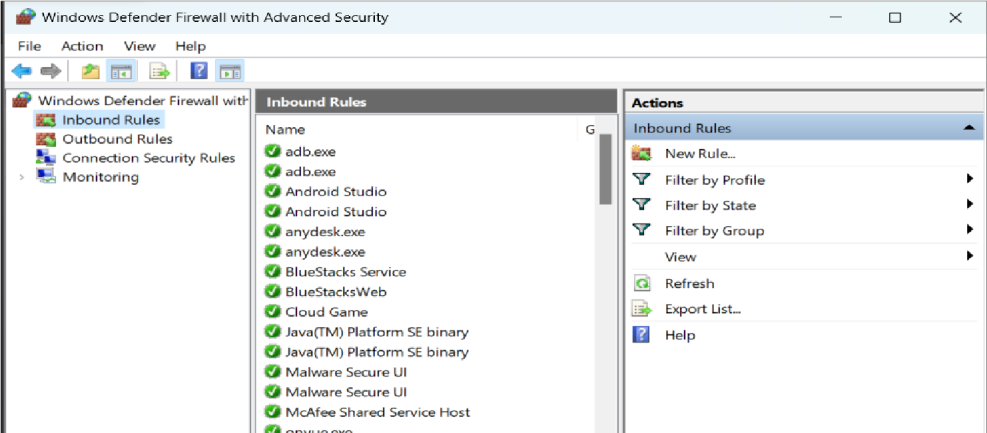
We access any website through the browser and confirm that the HTTP/HTTPS protocols are working. Go to Browser > Search Anything > Open > Working

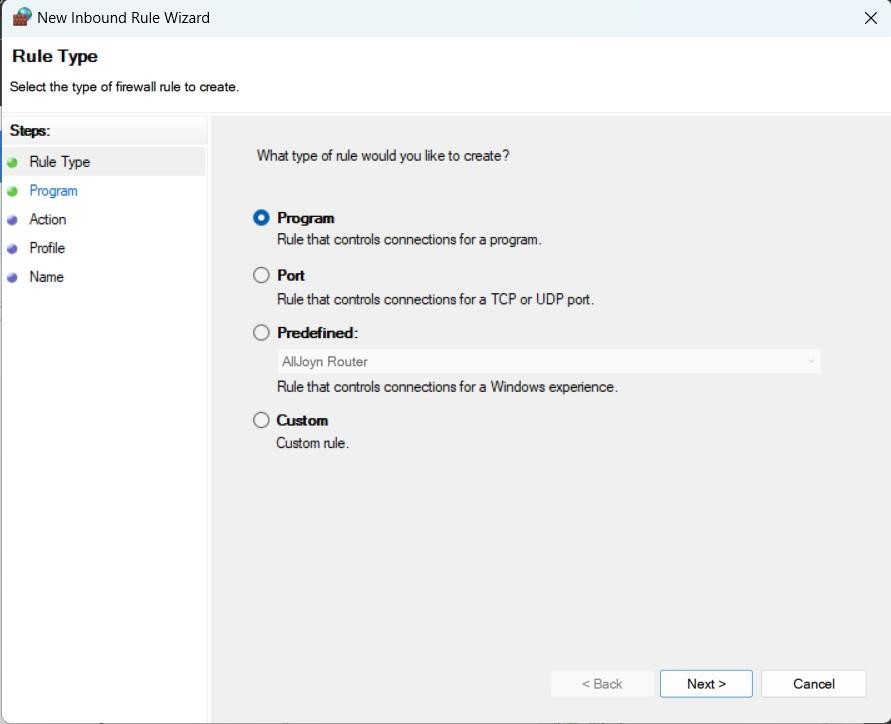
Step 2: We open ‘Windows Defender Firewall’

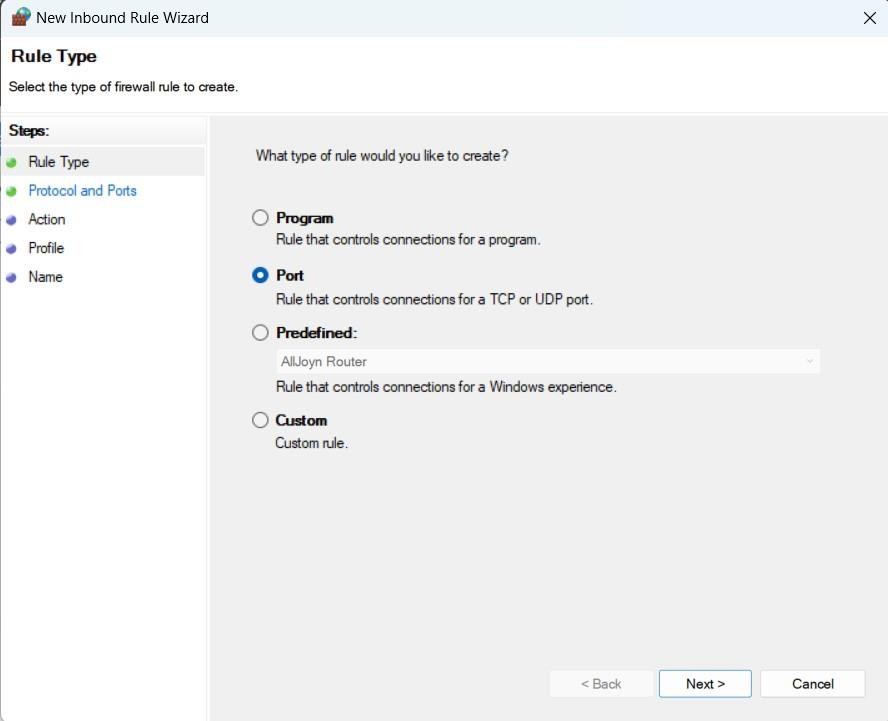


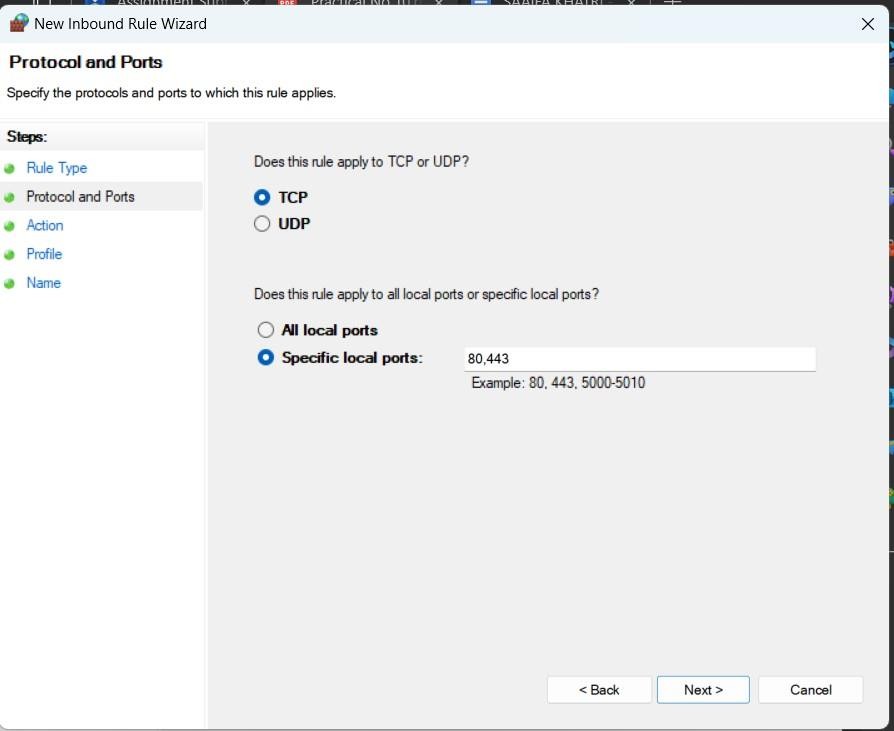
Next, we click on ‘Advanced settings’

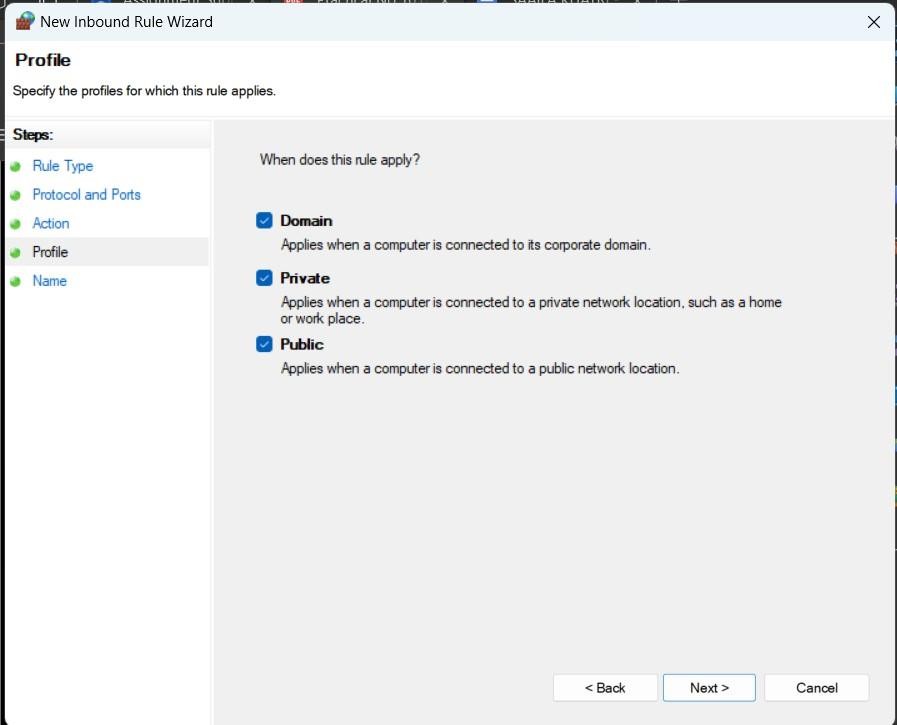
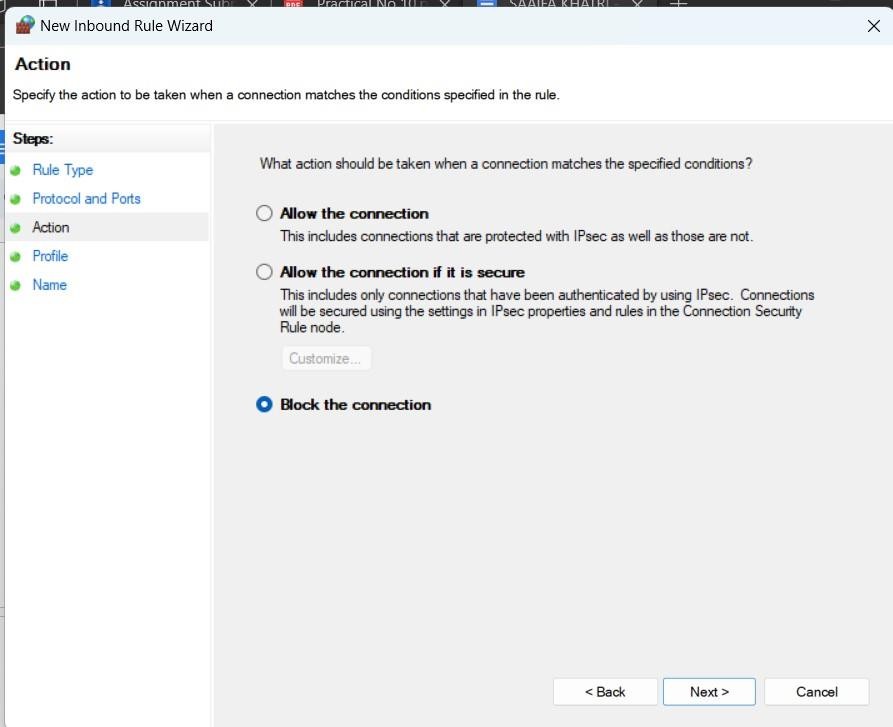
Next, we click on ‘Inbound Rules’ now right click on inbound rule and create new rule

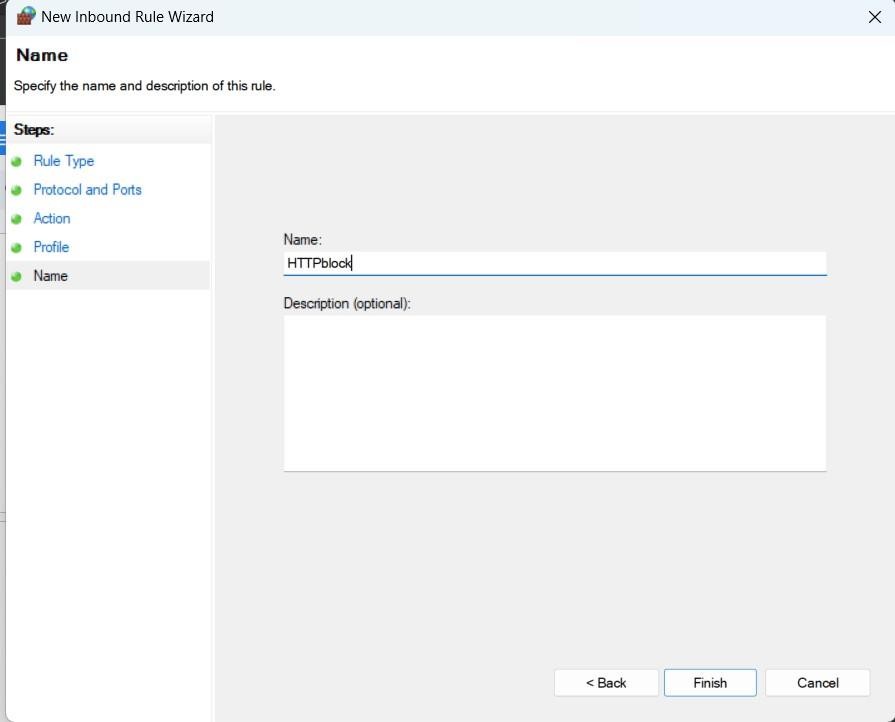


Then click on ‘New Rule’

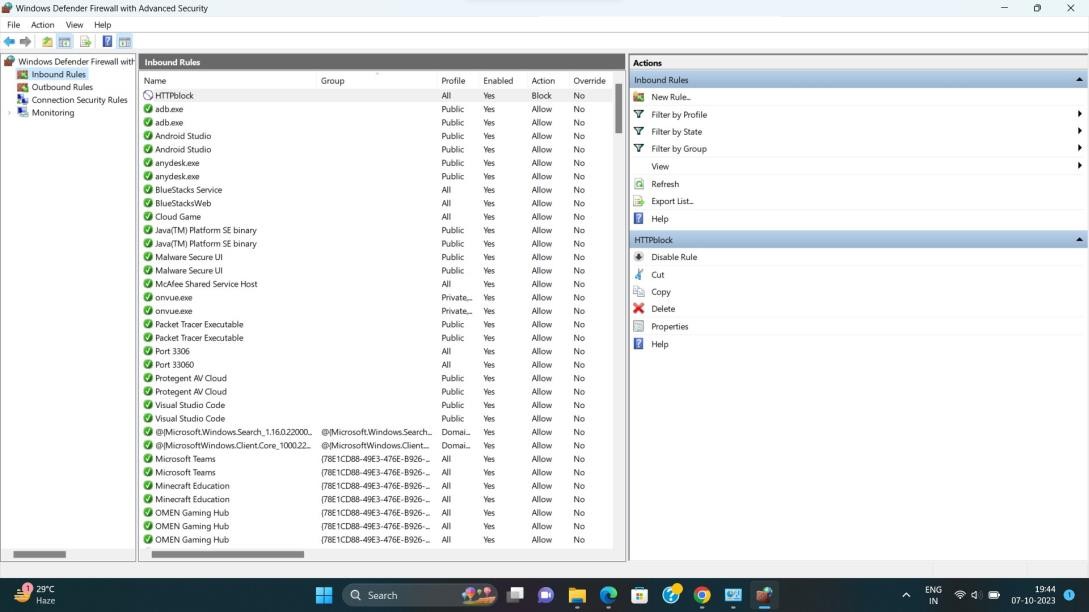
Select the radio button ‘Port’ and click ‘Next’ and enter the following



After next, we need to finalize the rule

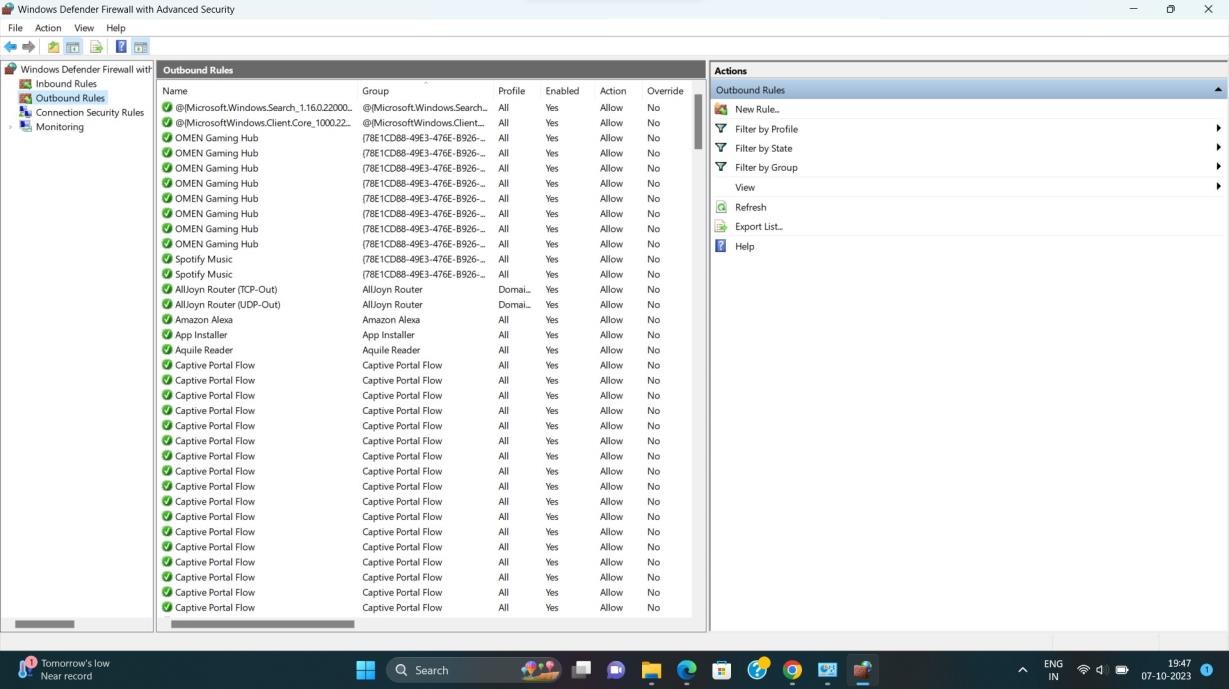
After clicking the ‘Next’ button we need to name the rule and click finish

The “new” inbound rule is added

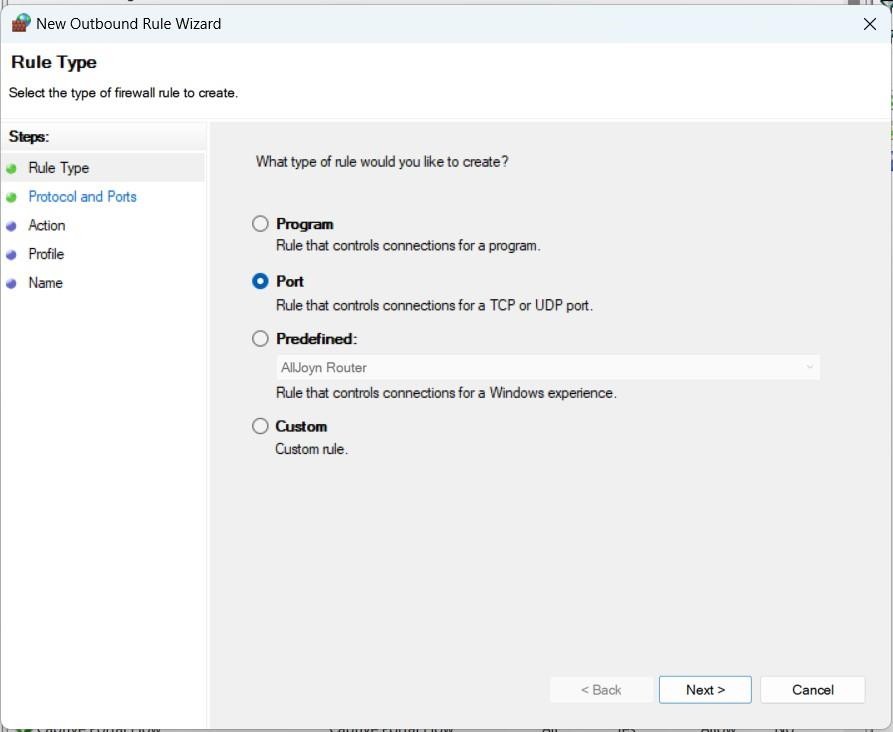


We repeat all the above steps for creating ‘Outbound Rules’, and then try to access the internet.

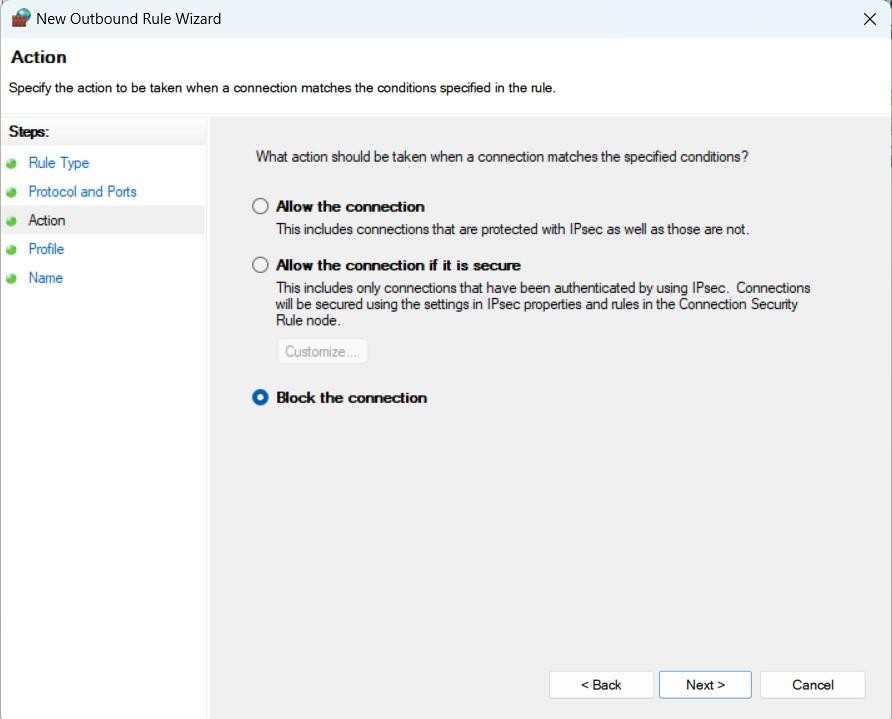
We see that the access is blocked For Outbound Rule:

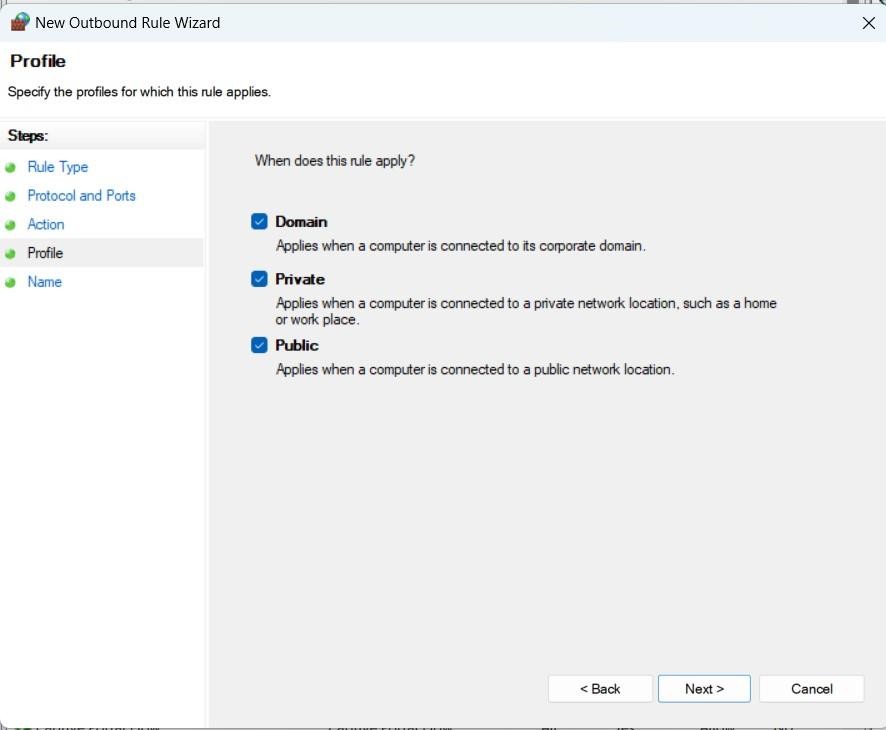


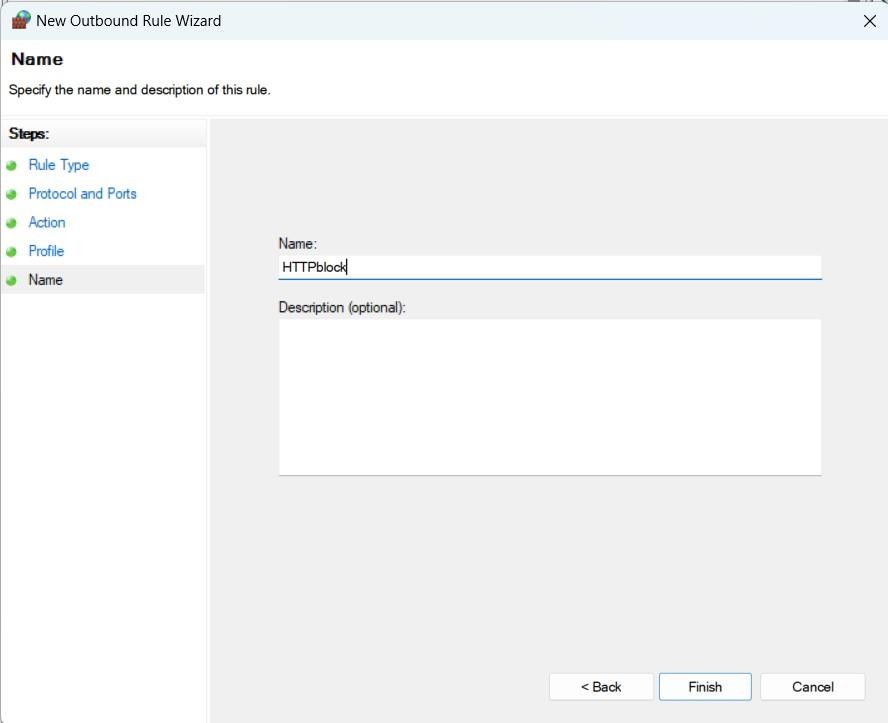
Click on “new rule” and select “port”, click next

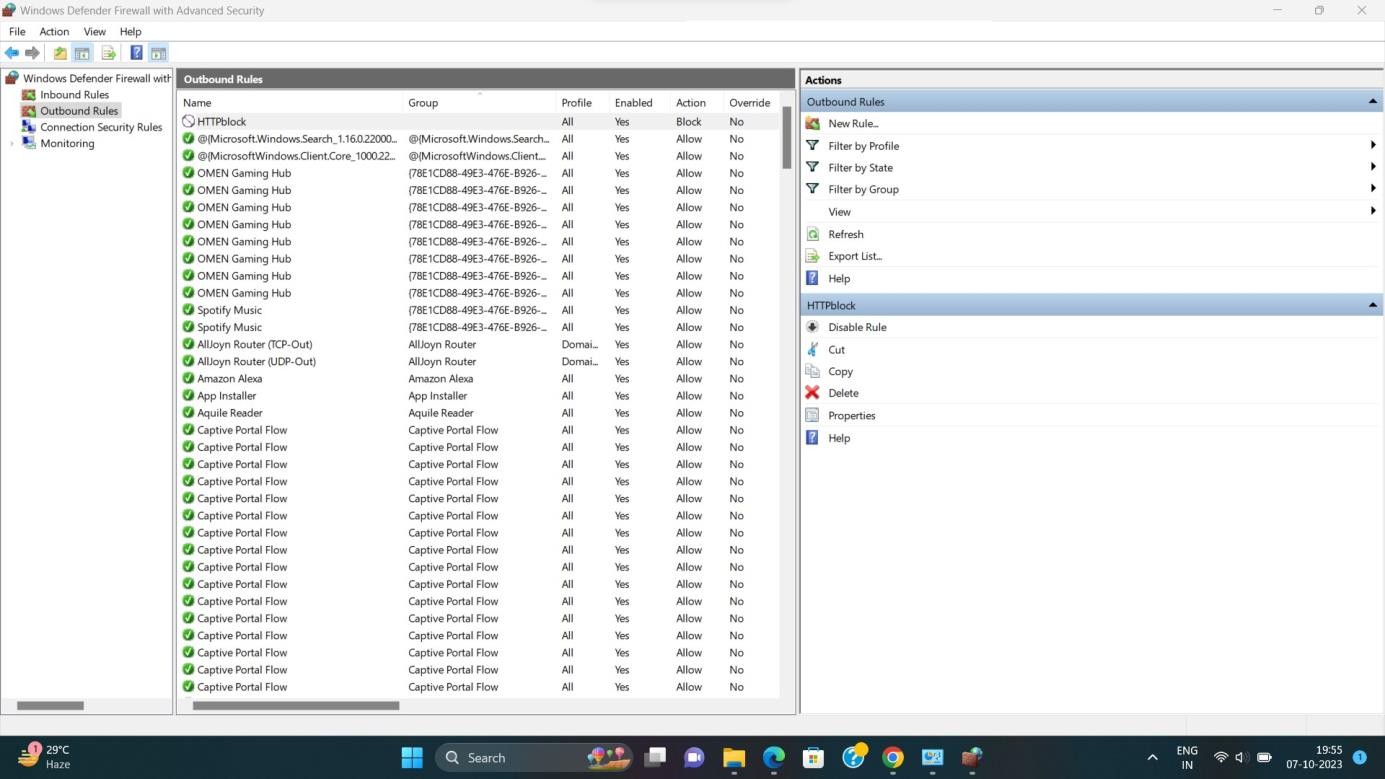


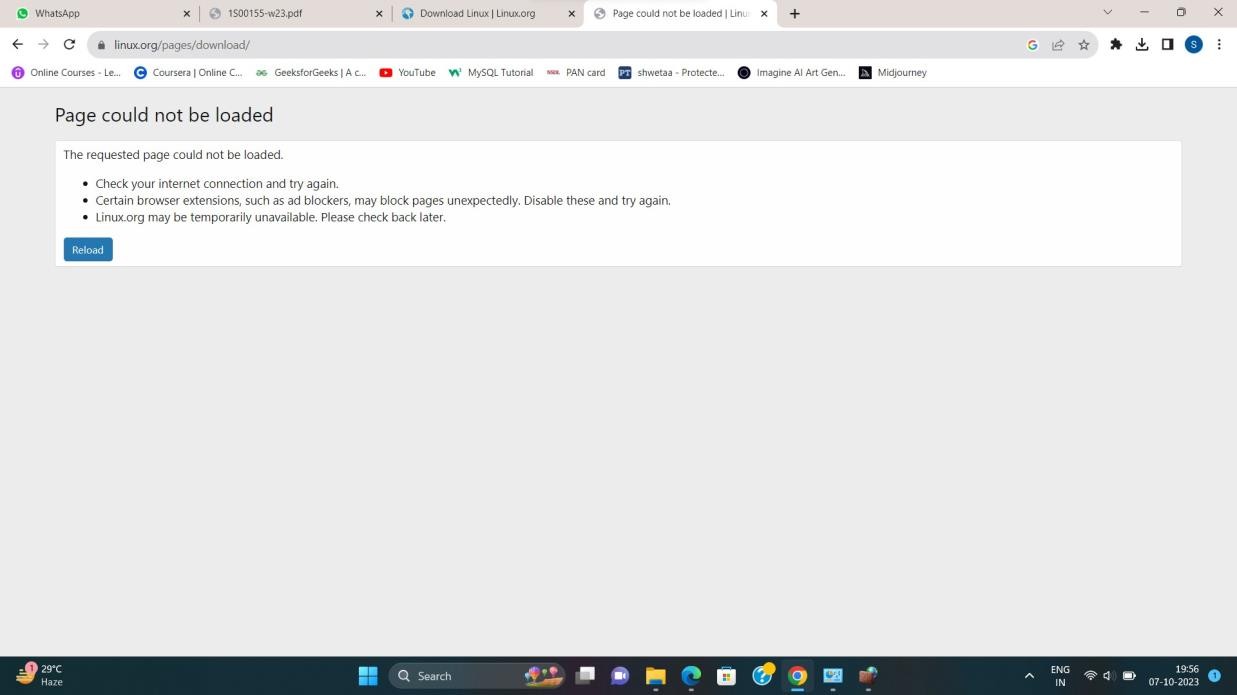
Add the following in the port wizard and click on next

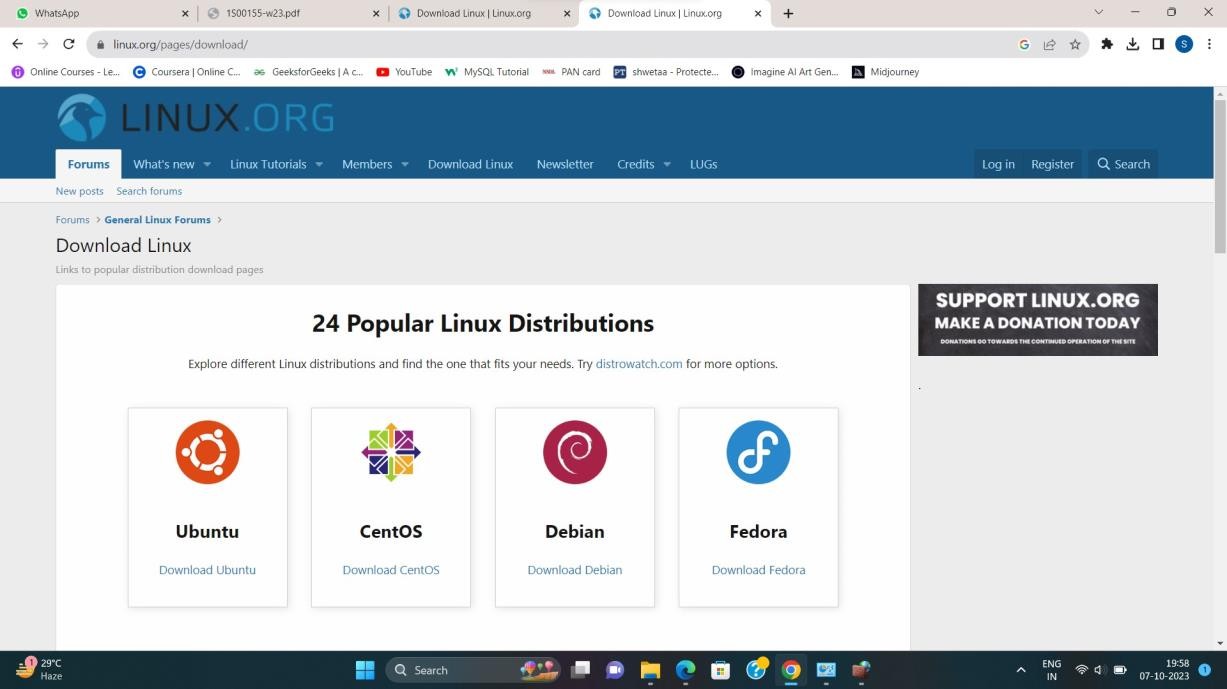






The new outbound rule is created

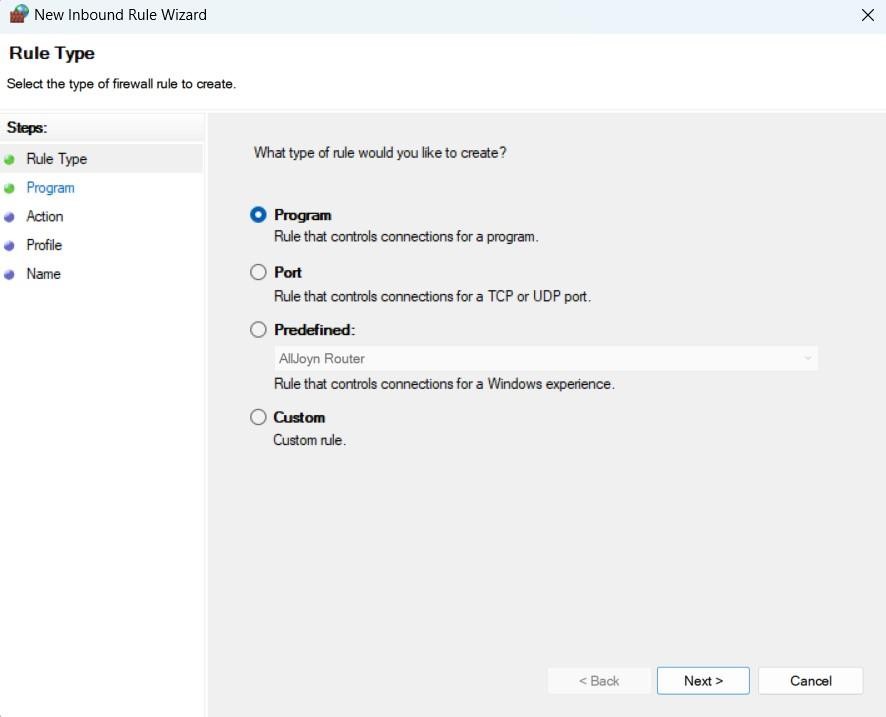
Trying to access the internet. We see that the accessed is blocked

Now to unblock the internet access. Go to the Windows Defender Firewall > Go to Inbound > Simply right click and delete it > Go to Outbound > Simply right click and delete it.

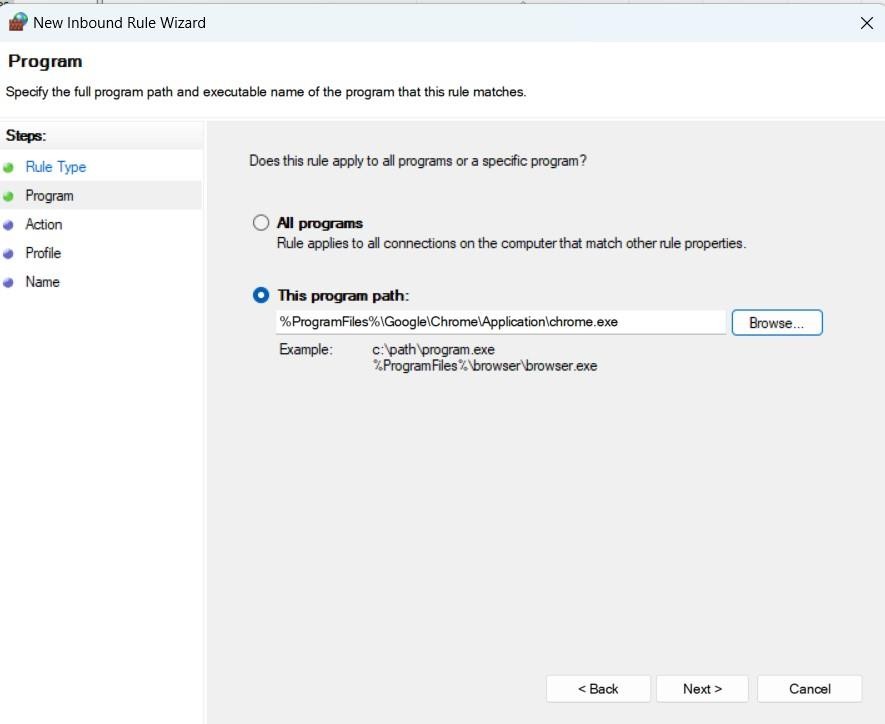
## PART 2: Blocking the Program

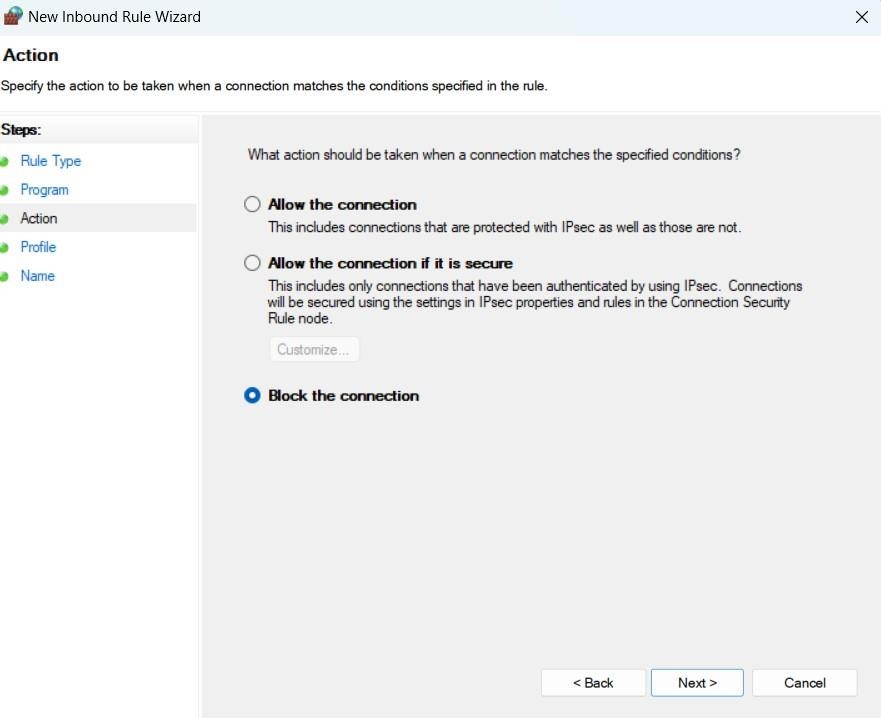
Step 1: Open Windows Defender Firewall Step 2: Go to Advance Settings

Step 3: Click on Inbound Rule Step 4: Create new rule

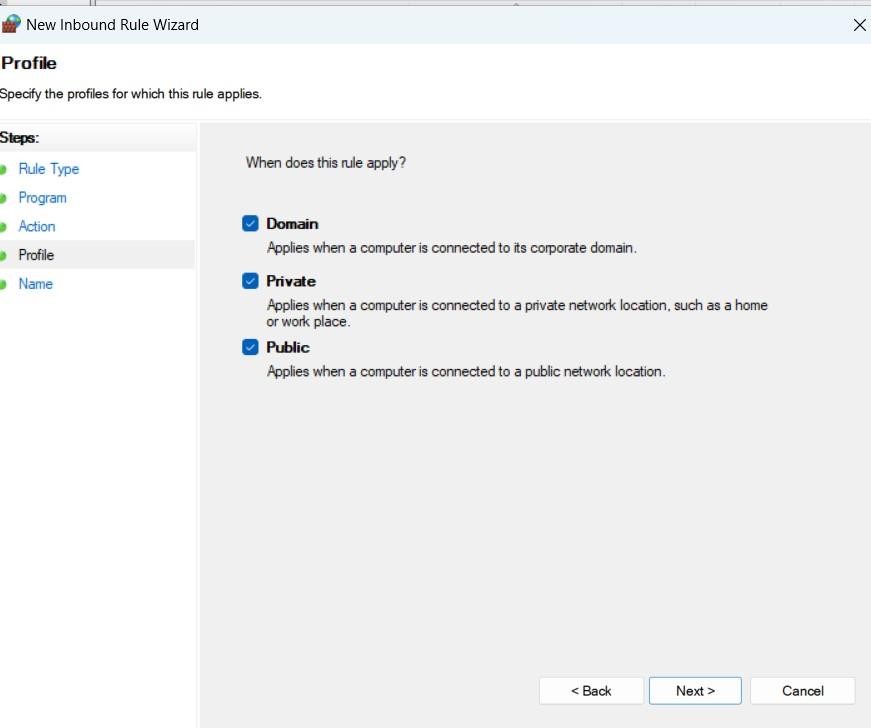
Step 5: Select Program and click on Next button

Step 6: We are blocking the chrome program so browse the path and click on next

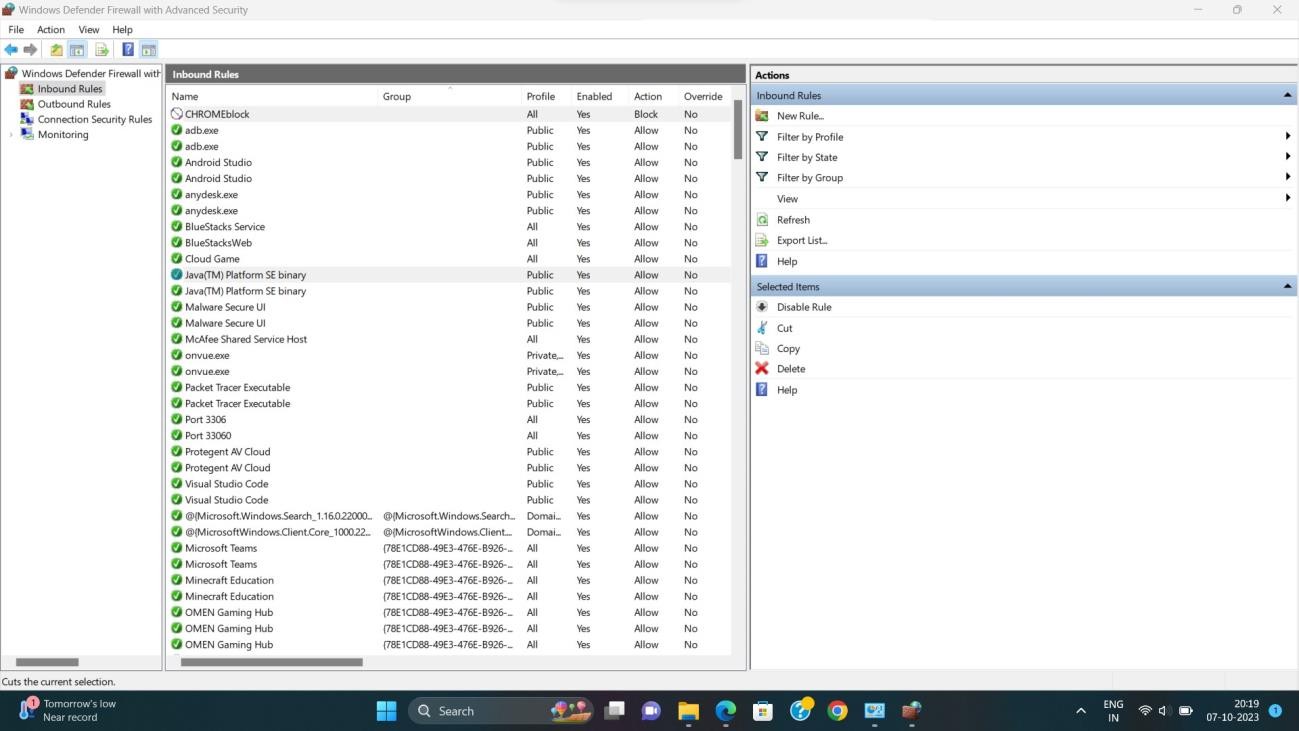


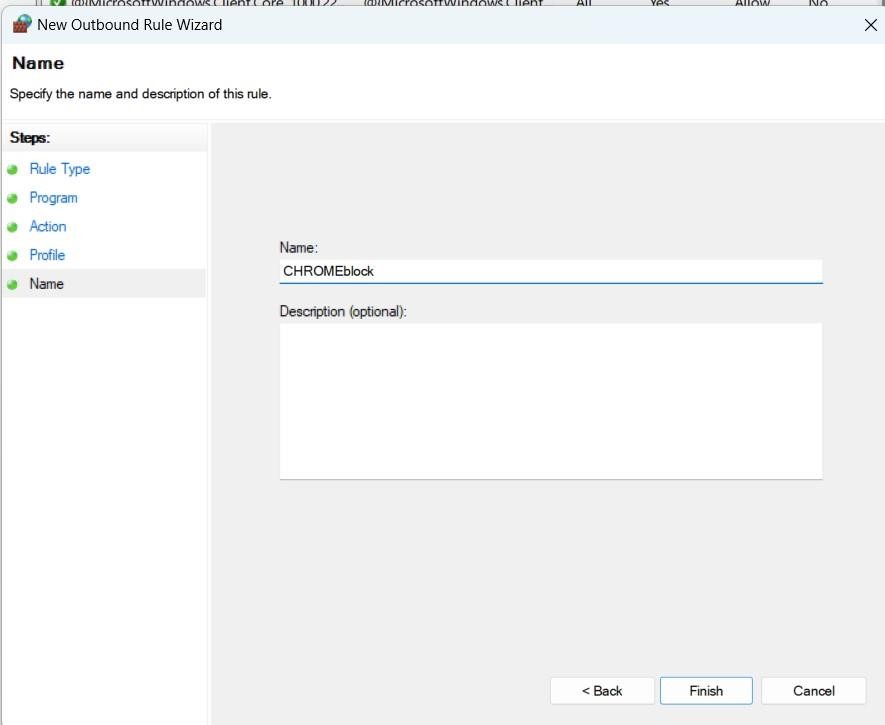
Step 7: Now click on block the connection

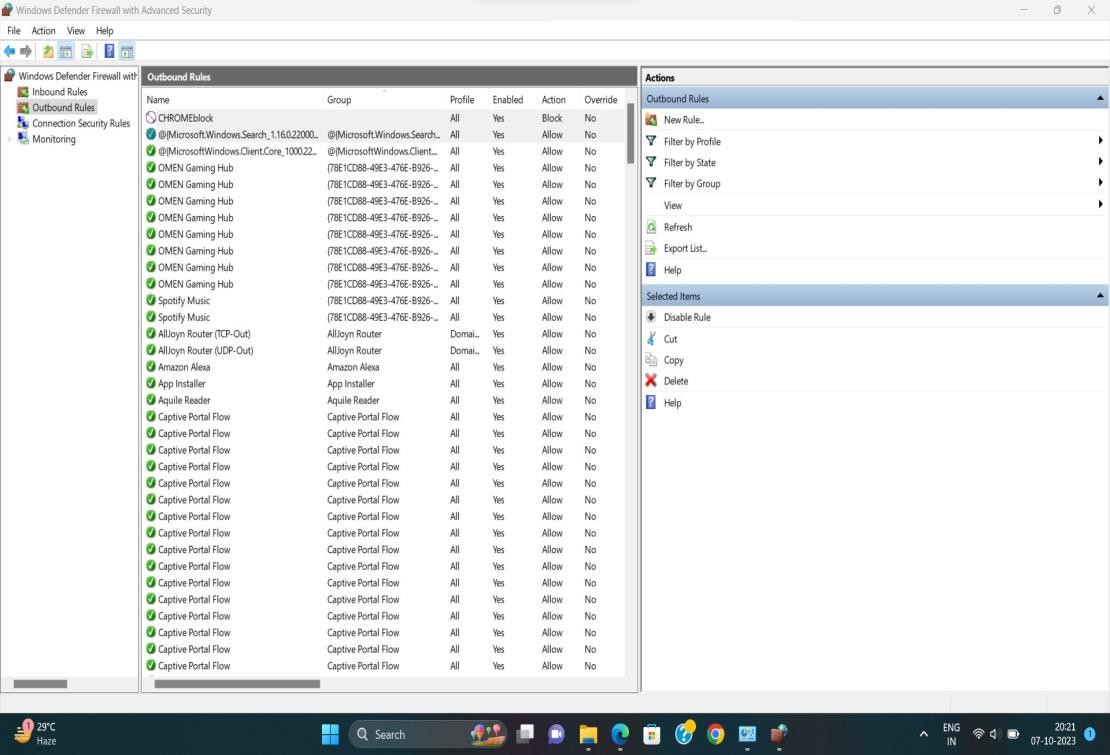
Step 8: Tick all the checkboxes

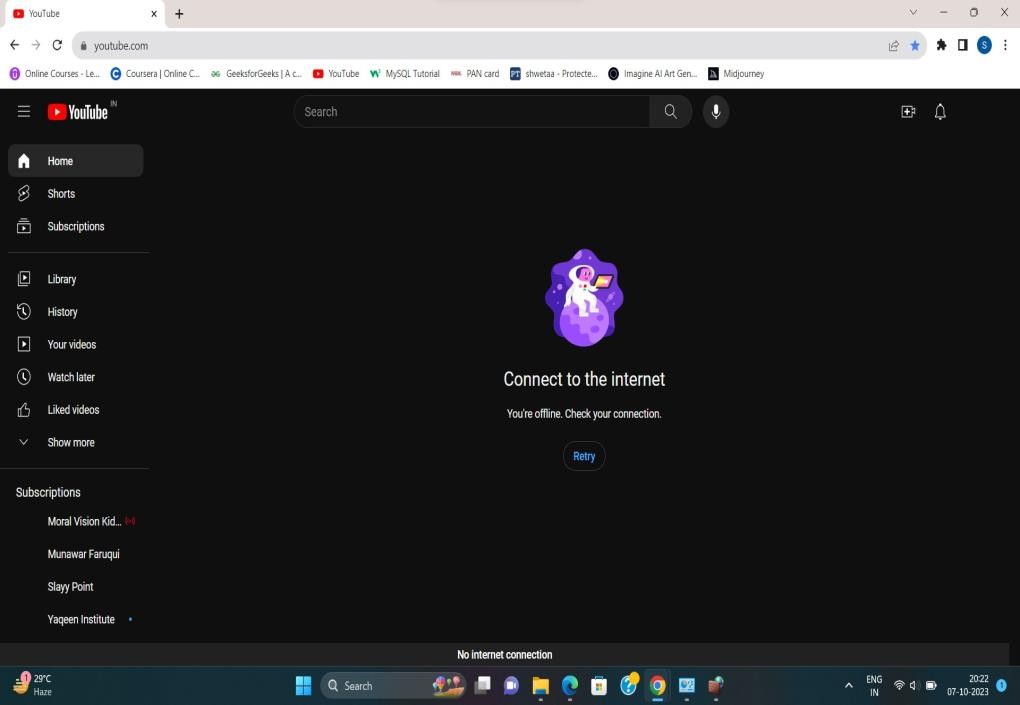


Step 9: After clicking on Next button give the name to that rule

Step 10: Click on Finish button and Inbound Rule is created

Step 11: Follow the same steps for Outbound Rule

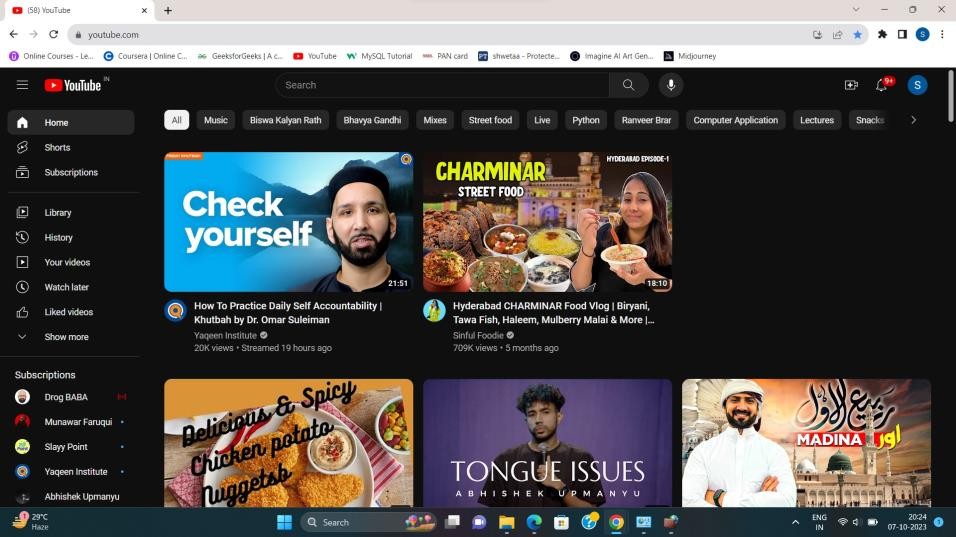


Step 12: Now open chrome and check it will show connect to internet

Step 13: To unblock the chrome > Go to Windows Defender Firewall > Advance Setting > Inbound Rule > Simply right click on that rule and delete it.

Step 14: Same process for Outbound Rule

`Step 15: Now open the chrome and check



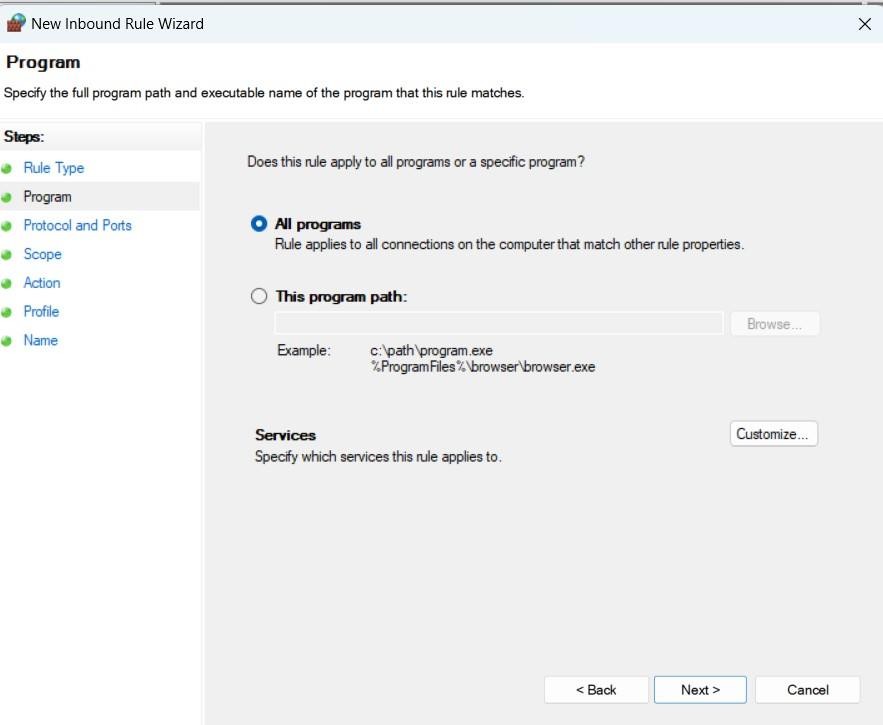
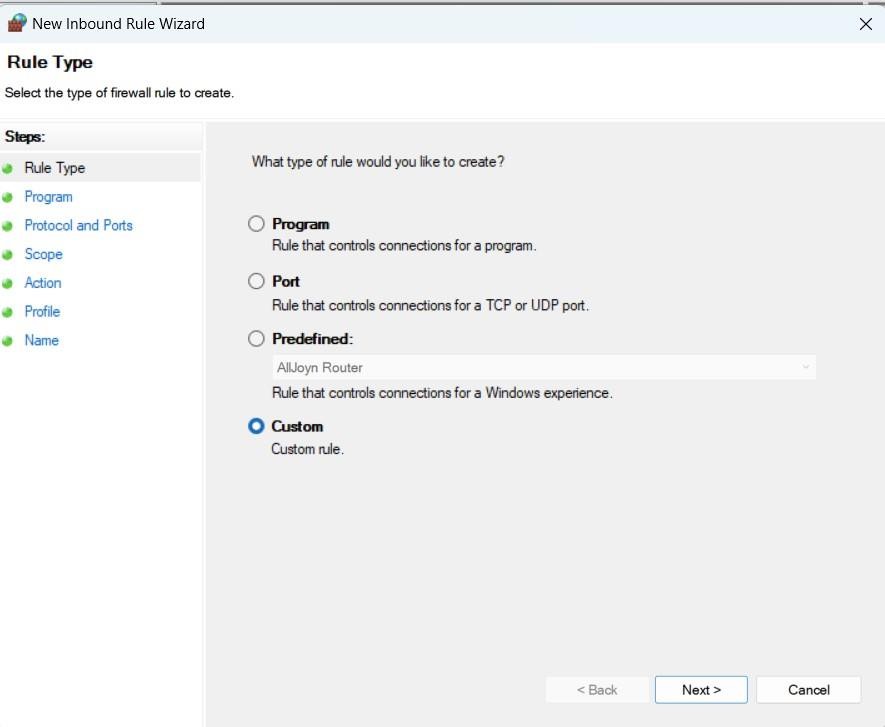
## Part 3: Blocking the website: wikipedia.com

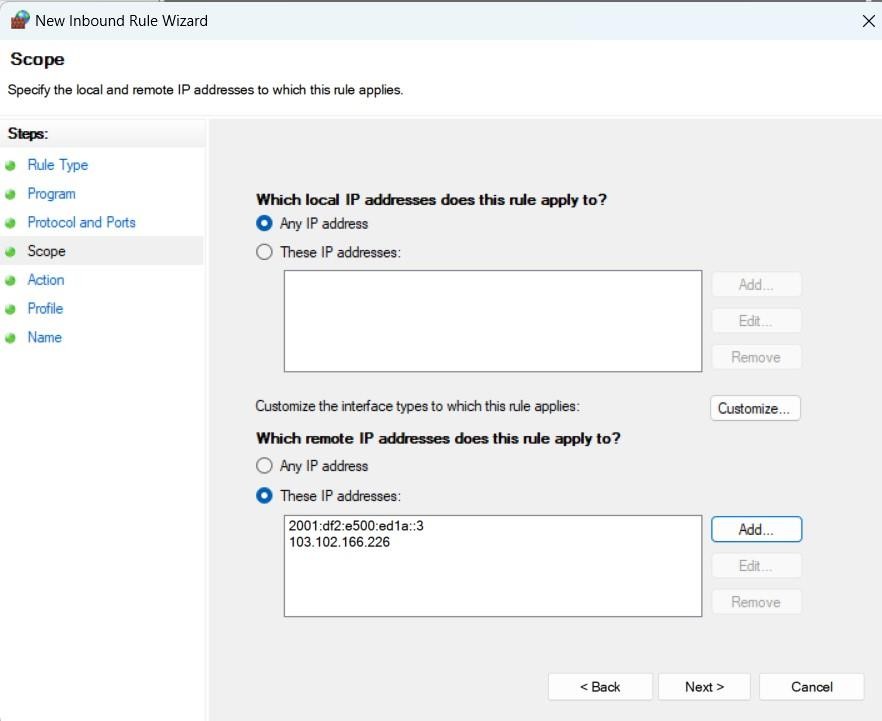
Server: UnKnown Address: 103.170.80.4

Non-authoritative answer: Name: wikipedia.com Addresses: 2001:df2:e500:ed1a::1103.102.166.224

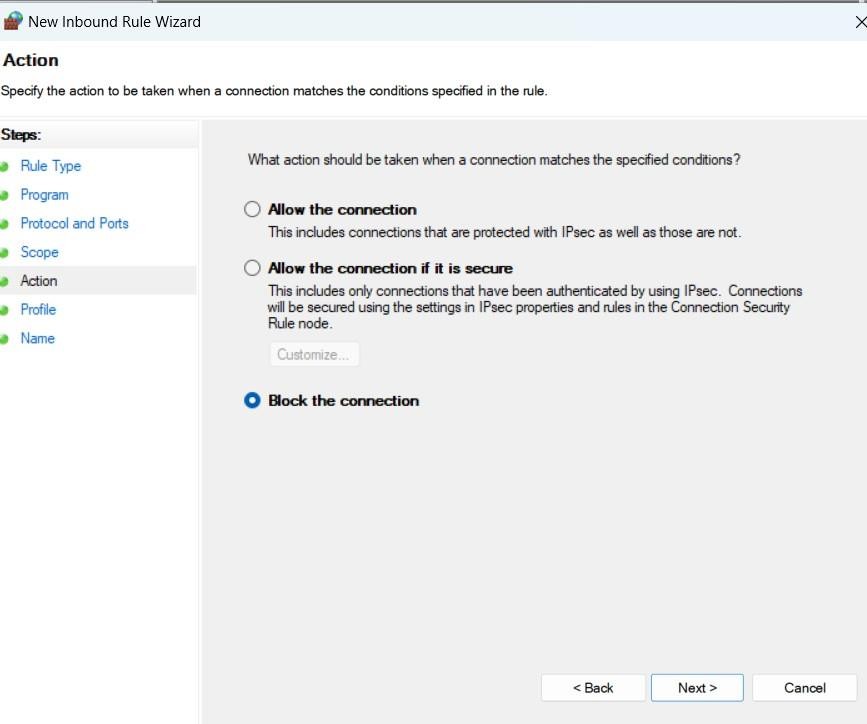
Create Inbound Rule for IPV6: We open the Windows firewall settings and apply the Inbound

Rule



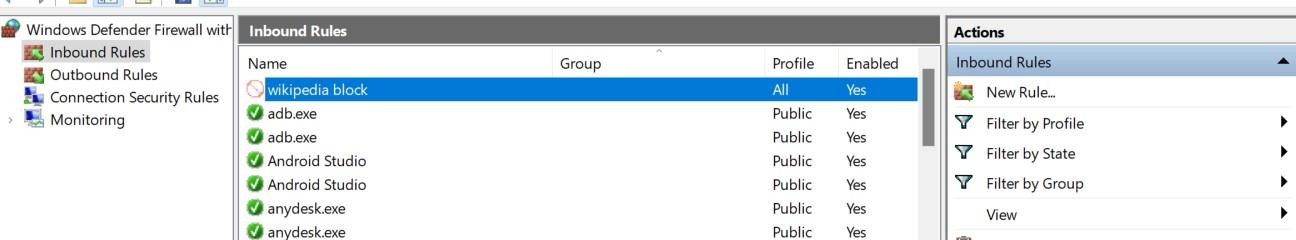
Insert the IP addresses of IPv6 and IPv4

Click on block connection



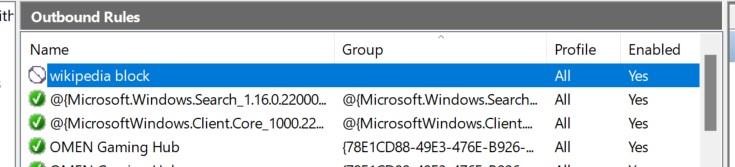
Click on Next simply

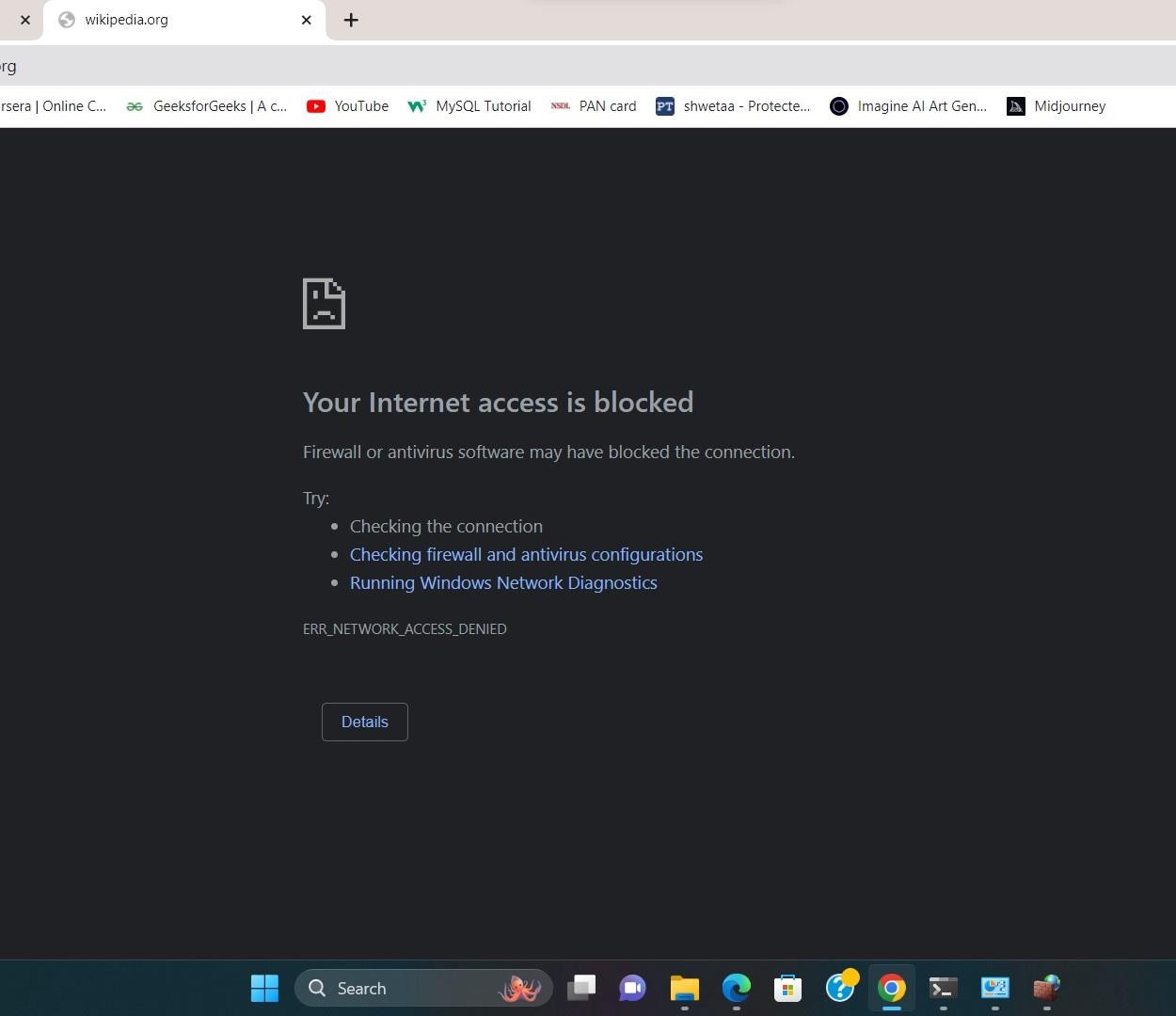
Now write the name for this Inbound rule



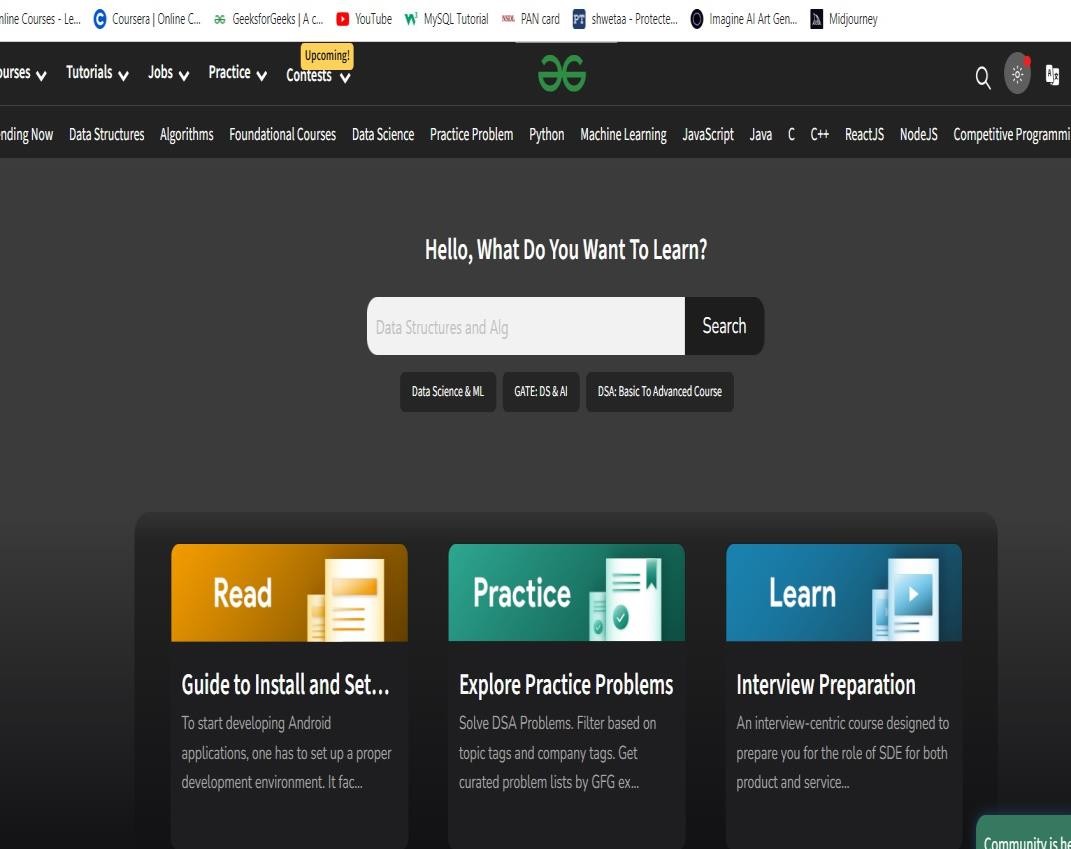
Click on the Finish button and the Inbound rule is added

Now, follow the same steps for creating the outbound rule



Go to the browser and search Wikipedia.org

Simultaneously we checked for other search its working properly



Now to unblock the website > Go to Windows Defender Firewall > Advance Settings > Inbound Rule > Simply right-click on that rule and delete it > Same process for Outbound > Now check below

