PRACTICAL 1

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

Program 1: Write a python program to implement Ceaser 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:

```
Enter String : information
enter Shift Number : 4
1. For Encrypt
2. For Decrypt
 Enter Your choice : 1
Original String :  information
After Encryption :  mrjsvqexmsr
```

```
Enter String : mrjsvqexmsr
enter Shift Number : 4
1. For Encrypt
2. For Decrypt
 Enter Your choice : 2
Original String :  mrjsvqexmsr
After Decryption :  information
```

Program 2: Write a python program to implement Mono-alphabetic Cipher.

```
alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
keyword = "ZYXWVUTSRQPONMLKJIHGFEDCBA"
def encrypt(Plaintext):
    result = ""
    for char in Plaintext:
        if char in alphabet:
            num = alphabet.find(char)
            result += keyword[num]
        else:
            result += char
    print("Encrypted Text:", result)
def decrypt(Ciphertext):
    result = ""
    for char in Ciphertext:
        if char in keyword:
            num = keyword.find(char)
            result += alphabet[num]
        else:
            result += char
    print("Decrypted Text:", result)
while True:
    try:
        n = int(input("Enter Value:\n1) Encrypt Text\n2) Decrypt Text\n3) See Key\n4) Exit\nChoice: "))
    except ValueError:
        print("Invalid input; please enter a number between 1 and 4.")
        continue
    if n == 1:
        Plaintext = input("Enter Text to Encrypt: ")
        encrypt(Plaintext.upper())
    elif n == 2:
        Ciphertext = input("Enter Text to Decrypt: ")
        decrypt(Ciphertext.upper())
    elif n == 3:
        print("Substitution Key (Keyword):", keyword)
    elif n == 4:
        print("Exiting the program.")
        break
    else:
        print("Invalid Input; Enter Again!!")
```

OUTPUT:

```
Enter Value:
1) Encrypt Text
2) Decrypt Text
3) See Key
4) Exit
Choice: 1
Enter Text to Encrypt: information
Encrypted Text: RMULINZGRLM
Enter Value:
1) Encrypt Text
2) Decrypt Text
3) See Key
4) Exit
Choice: 2
Enter Text to Decrypt: rmulinzgrlm
Decrypted Text: INFORMATION
Enter Value:
1) Encrypt Text
2) Decrypt Text
3) See Key
4) Exit
Choice: 3
Substitution Key (Keyword): ZYXWVUTSRQPONMLKJIHGFEDCBA
Enter Value:
1) Encrypt Text
2) Decrypt Text
3) See Key
4) Exit
Choice: 4
Exiting the program.
```

Program 3: Write a python program to implement 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:

```
Enter key : 2

 1.Encryption
 2.Decryption:
 3.EXIT
```

```
Enter Your Choice:
1
ENTER MSG : hello
CIPHER TEXT: IF NV MK
 1.Encryption
 2.Decryption:
 3.EXIT
```

```
Enter Your Choice:
2
ENTER CIPHER TEXT:ifnvmk
PLAIN TEXT: HE LX LO
 1.Encryption
 2.Decryption:
 3.EXIT
```

Program 4: Write a python program to implement 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

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:

```
Enter Key: word
Enter Plaintext: open
CipherText:  kdvq
PlainBack:  open
```

Program 5: Write a python program to implement Simple Columnar Transposition Cipher.

```
import math
key = "HACK"

def encryptMessage(msg):    #Encryption
    cipher = ""
    k_indx = 0
    msg_len = float(len(msg))
    msg_lst = list(msg)
    key_lst = sorted(list(key))
    col = len(key)
    row = int(math.ceil(msg_len / col))
    fill_null = int((row * col) - msg_len)
    msg_lst.extend('_' * fill_null)
    matrix = [msg_lst[i: i + col]
              for i in range(0, len(msg_lst), col)]
```

```
    for _ in range(col):
        curr_idx = key.index(key_lst[k_indx])
        cipher += ''.join([row[curr_idx]
                           for row in matrix])
        k_indx += 1
    return cipher

def decryptMessage(cipher):     # Decryption
    msg = ""
    k_indx = 0
    msg_indx = 0
    msg_len = float(len(cipher))
    msg_lst = list(cipher)
    col = len(key)
    row = int(math.ceil(msg_len / col))
    key_lst = sorted(list(key))
    dec_cipher = []
    for _ in range(row):
        dec_cipher += [[None] * col]
    for _ in range(col):
        curr_idx = key.index(key_lst[k_indx])
        for j in range(row):
            dec_cipher[j][curr_idx] = msg_lst[msg_indx]
            msg_indx += 1
        k_indx += 1
    try:
        msg = ''.join(sum(dec_cipher, []))
    except TypeError:
        raise TypeError("This program cannot",
                        "handle repeating words.")
    null_count = msg.count('_')
    if null_count > 0:
        return msg[: -null_count]
    return msg
msg = "Come Home Tomorrow"
cipher = encryptMessage(msg)
print("Encrypted Message: {}".
      format(cipher))
print("Decryped Message: {}".
      format(decryptMessage(cipher)))
```

OUTPUT:

```
Encrypted Message: oH owmoTr_C emoemor_
Decryped Message: Come Home Tomorrow
```

Program 6: Write a python program to implement Rail fence Cipher.

```
def encryptRailFence(text, key):
    rail = [['\n' for i in range(len(text))]
                for j in range(key)]
    dir_down = False
    row, col = 0, 0

    for i in range(len(text)):
        if (row == 0) or (row == key - 1):
            dir_down = not dir_down
        rail[row][col] = text[i]
        col += 1
        if dir_down:
            row += 1
        else:
            row -= 1
    result = []
    for i in range(key):
        for j in range(len(text)):
            if rail[i][j] != '\n':
                result.append(rail[i][j])
    return("" . join(result))

def decryptRailFence(cipher, key):
    rail = [['\n' for i in range(len(cipher))]
                for j in range(key)]
    dir_down = None
    row, col = 0, 0
    for i in range(len(cipher)):
        if row == 0:
            dir_down = True
        if row == key - 1:
            dir_down = False
        rail[row][col] = '*'
        col += 1
        if dir_down:
            row += 1
        else:
            row -= 1
    index = 0
```

```
    for i in range(key):
        for j in range(len(cipher)):
            if ((rail[i][j] == '*') and
            (index < len(cipher))):
                rail[i][j] = cipher[index]
                index += 1
    result = []
    row, col = 0, 0
    for i in range(len(cipher)):
        if row == 0:
            dir_down = True
        if row == key-1:
            dir_down = False
        if (rail[row][col] != '*'):
            result.append(rail[row][col])
            col += 1
        if dir_down:
            row += 1
        else:
            row -= 1
    return("".join(result))

if __name__ == "__main__":
    print(encryptRailFence("attack at once", 2))
    print(encryptRailFence("defend the east wall", 3))

    print(decryptRailFence("atc toctaka ne", 2))
    print(decryptRailFence("dnhaweedtees alf  tl", 3))
```

OUTPUT:

```
atc toctaka ne
dnhaweedtees alf  tl
attack at once
defend the east wall
```

PRACTICAL 2

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

pip install pycryptodome

```
C:\Users\Tasmiya Shaikh>pip install pycryptodome
Collecting pycryptodome
  Downloading pycryptodome-3.23.0-cp37-abi3-win_amd64.whl.metadata (3.5 kB)
Downloading pycryptodome-3.23.0-cp37-abi3-win_amd64.whl (1.8 MB)
   ---------------------------------------- 1.8/1.8 MB 3.5 MB/s eta 0:00:00
Installing collected packages: pycryptodome
Successfully installed pycryptodome-3.23.0

[notice] A new release of pip is available: 24.2 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Program:

```
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(keyPair.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:

```
Public key: (n=0xa2f45259ea6584ba30e19931b7e7c07850a03db755999de26b3831b584ba7997892c9b16c281ce8f6c26d3a4aa3b059b8e07b600b7485510b3f266a24a0fa541cf281b8b8dc9a8e5d7885f999afa00881f66
ed56b107be59a5bf38b441196c8d0d08e8bd46603b5e6ba007eb125e6a9b64c99f04af55a0a05e4e7f4ac03ea6df, e=0x10001)
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCi9FJZ6mWEujDhmTG358B4UKA9
t1WZneJrODG1hLp5l4ksmxbCgc6PbCbTpKo7BZuOB7YAt0hVELPyZqJKD6VBzygb
i43JqOXXiF+ZmvoAiB9m7VaxB75Zpb84tEEZbI0NCOi9RmA7XmugB+sSXmqbZMmf
BK9VoKBeTn9KwD6m3wIDAQAB
-----END PUBLIC KEY-----
Private key: (n=0xa2f45259ea6584ba30e19931b7e7c07850a03db755999de26b3831b584ba7997892c9b16c281ce8f6c26d3a4aa3b059b8e07b600b7485510b3f266a24a0fa541cf281b8b8dc9a8e5d7885f999afa00881f6
6ed56b107be59a5bf38b441196c8d0d08e8bd46603b5e6ba007eb125e6a9b64c99f04af55a0a05e4e7f4ac03ea6df, d=0x29f42f53990684128fcd356cd885614758b2b8d0ffc7ea1f462d9496c0526645be7a6e7f809a7cfcee
025d111306d91dc08679b45b7bd2f2d6f17cbd21c63f6c75a935f48e269fb1e60f864d957ce7359ed1534deba1fb8acc44eb76da701fe0fca021aecf65938a496822c3bd701317c1378fa52e3fa65ea827b80cd4210531)
-----BEGIN RSA PRIVATE KEY-----
MIICWwIBAAKBgQCi9FJZ6mWEujDhmTG358B4UKA9t1WZneJrODG1hLp5l4ksmxbC
gc6PbCbTpKo7BZuOB7YAt0hVELPyZqJKD6VBzygbi43JqOXXiF+ZmvoAiB9m7Vax
B75Zpb84tEEZbI0NCOi9RmA7XmugB+sSXmqbZMmfBK9VoKBeTn9KwD6m3wIDAQAB
AoGAKfQvU5kGhBKPzTVs2IVhR1iyuND/x+ofRi2UlsBSZkW+em5/gJp8/O4CXRET
BtkdwlZ5tFt70vLW8Xy9IcY/bHWpNfSOJp+x5g+GTZV85zWe0VNN66H7isxE63ba
cB/g/KAhrs9Ik4pJaCLDvXATF8E3j6UuP6ZeqCe4DNQhBTECQQC+6wqJ6ZQU5D8G
60znN9eqZwAvU6sSEVl/QsNikPGrEb8UzGuviPTgi4ulUnnbDfieAsLwsKPHUqCT
bFUiANdnAkEA2oDz2VuT0JiQALpktS1e6qWHKc5sHYfQndLG25bk210ar1xgYYFA
B8qHW3OtxaveFGxUVi1xV7LHoepdlxzhyQJAfYYpwq0paGb9LX5tbKRhjq70Byna
kyGCUweBUP/9dL1RRcp1js7cOfpl0UedyLGx4antl7TvqbCg1OzoOFgXRwJADScg
BqB9mFvmpzSRJVrS0Nsf9MNJPMnVl/rxH3zlz++d0JqF7riiCxYE6dtfxXtgBeiz
dvCLVxUa8UMzWQpTCQJAEK9H6l/yPsUedhl3l1sTZlu06FpAeijaTymf8i1p8fyv
RvVqrByUhqO1Y1RClmfmYHHZiY+t5UCpN+T7+kUAmQ==
-----END RSA PRIVATE KEY-----
Encrypted: 2753954e8d4f6ee2987cc6a278cb8f94a4215e1956a204cc6f70abd07fe6dac534a5bdf65c6d113a5110bc12e03f787b1ac9c363121e0e264aa62ace5ac460af1e8cf0fea3b1ed180553ec201768d9381f2868cec571
8e11f273ff879fdf751ec4ff3acb6e95fd564a3e13a39c17ce4df4e05438ee1581244623a064286d9df5
```

PRACTICAL 3

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

Program 1: Write a python program to implement MD4 (Message Digest Algorithm 4).

```
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:

```
The byte equivalent of hash is : b"\x8b\x1a\x99S\xc4a\x12\x96\xa8'\xab\xf8\xc4x\x04\xd7"
The byte equivalent of hash is : b'\xce=.Zp\x88\x17\x84q\x8c\xaa\x92#Z\xc9\xa7'
```

Program 2: Write a python program to implement SHA (Secure Hash Algorithm).

```
import hashlib
str = input("Enter the value to encode ")
result = hashlib.sha1(str.encode())
print("The hexadecimal equivalent of SHA1 is : ")
print(result.hexdigest())
```

OUTPUT:

```
Enter the value to encode hello
The hexadecimal equivalent of SHA1 is :
aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d
```

PRACTICAL 4

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

Program:

```
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(original_document)  #If put "modified_document" it will be invalid
signature = pkcs1_15.new(RSA.import_key(private_key)).sign(original_hash)
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:

If original and modified document is same

```
Signature is valid.
```

If original and modified document is not same

```
Signature is invalid.
```

PRACTICAL 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.

Program:

```
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:

```
The Value of q is :23
The Value of alpha is :9
Secret Number for Alice is :4
Secret Number for Bob is :6
Secret key for the Alice is : 12
Secret Key for the Bob is : 12
```

PRACTICAL 6

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

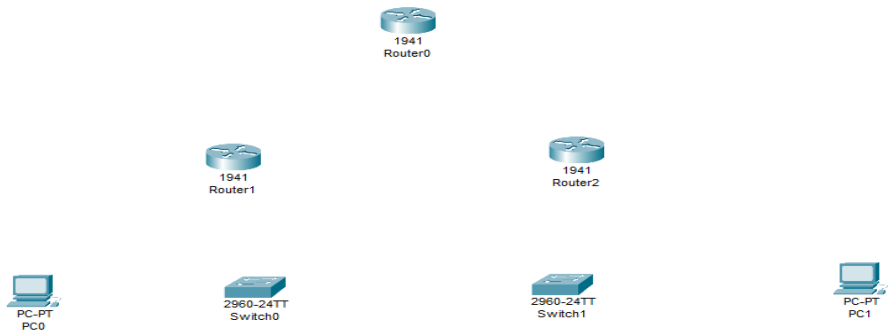
Requirements:

- Take 3 Routers (1941)
- Take 2 Switches (2960)
- Take 2 PC
- Configuration:

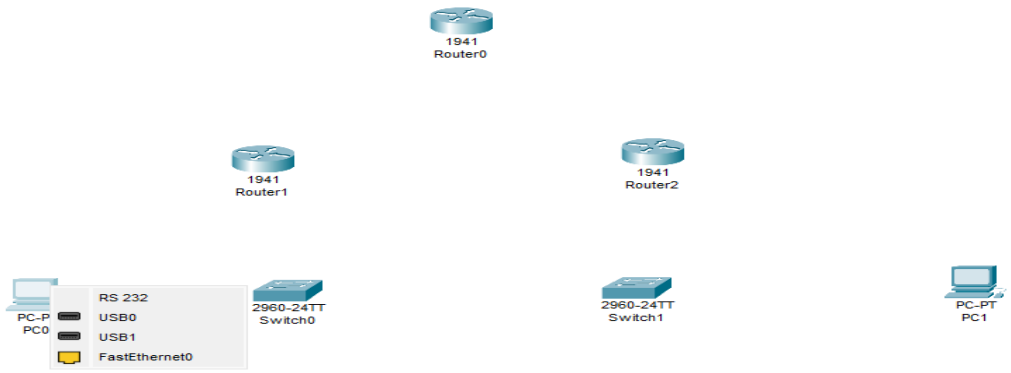
  PC 0 – 192.168.1.2
  PC 1 – 192.168.2.2
  Router 1 (G 0/0) - 20.0.0.1
  Router 1 (G 0/1) - 192.168.1.1
  Router 0 (G 0/0) - 30.0.0.2
  Router 0 (G 0/1) - 20.0.0.2
  Router 2 (G 0/0) - 30.0.0.1
  Router 2 (G 0/1) - 192.168.2.1

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

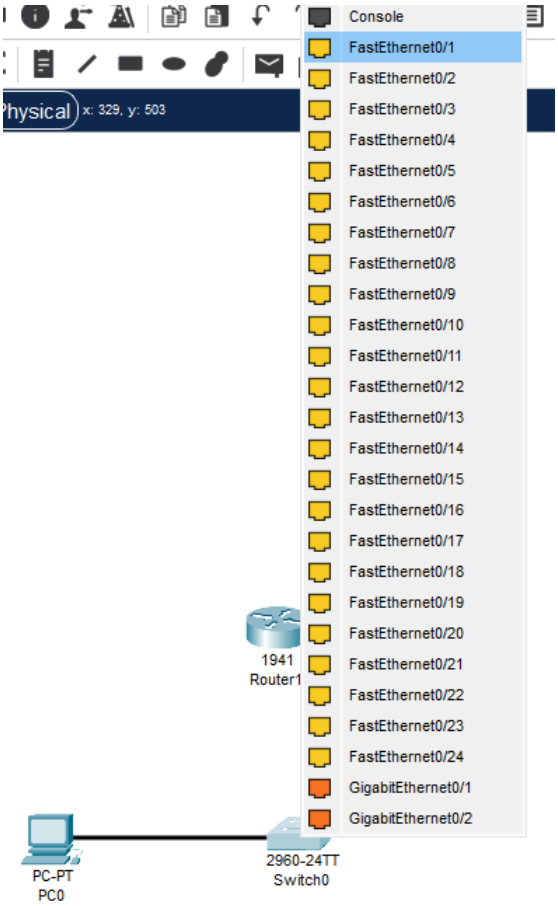
Connect the PC with Switches and also connect switches with Router using copper straight wire as given below.

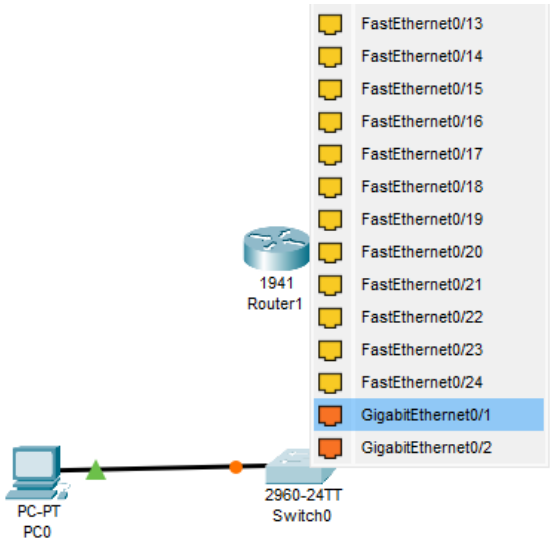


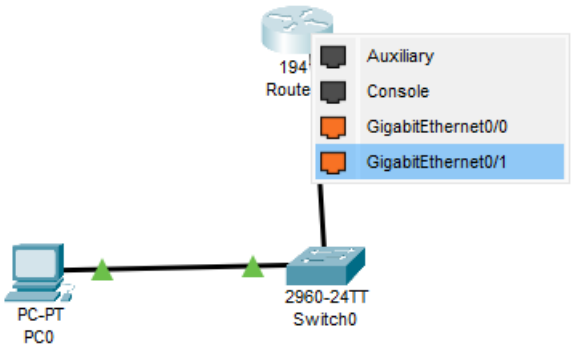
Connect with copper straight wire



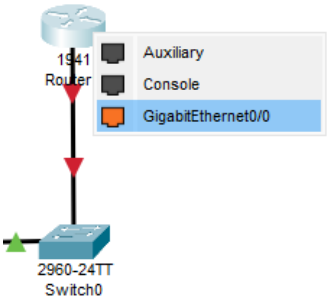
Connect the PC to the ethernet (FastEthernet0/24) and connect it with switch with GigabitEthernet0/1

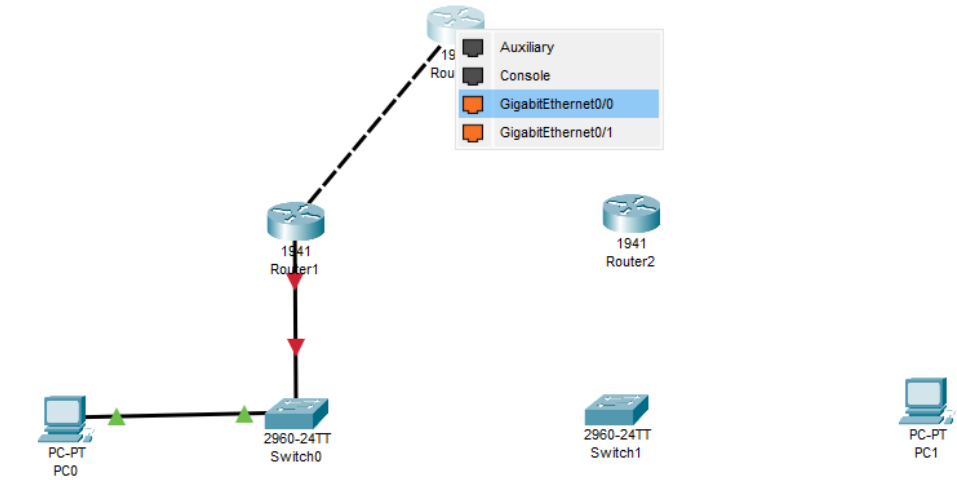




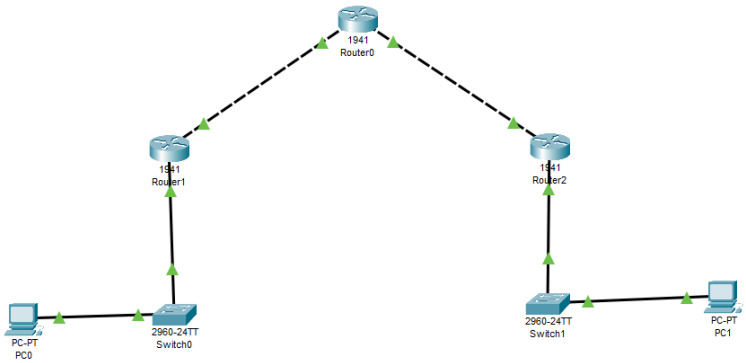


Connect the Router 1 with Router 0 and Router 0 with Router 2 using copper cross wire as given below.





Valid Connection:



Step 2: CLI Commands-

Router1-

```
Router#enable
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# ip route 0.0.0.0 0.0.0.0 20.0.0.2
Router(config)#
```

Router2-

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# ip route 0.0.0.0 0.0.0.0 30.0.0.2
Router(config)#
```

Providing Hostname:

Router0-

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# hostname R0
R0(config)#exit
R0#
%SYS-5-CONFIG_I: Configured from console by console
```

Router1-

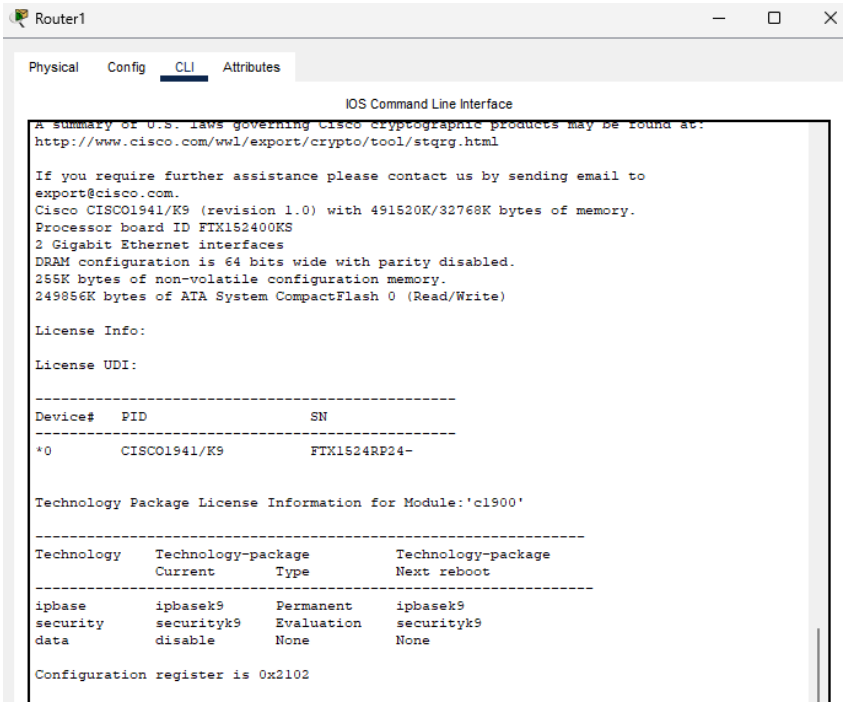
```
Router#enable
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# ip route 0.0.0.0 0.0.0.0 20.0.0.2
Router(config)#hostname R1
R1(config)#
R1(config)#exit
R1#
%SYS-5-CONFIG_I: Configured from console by console
```

Router2

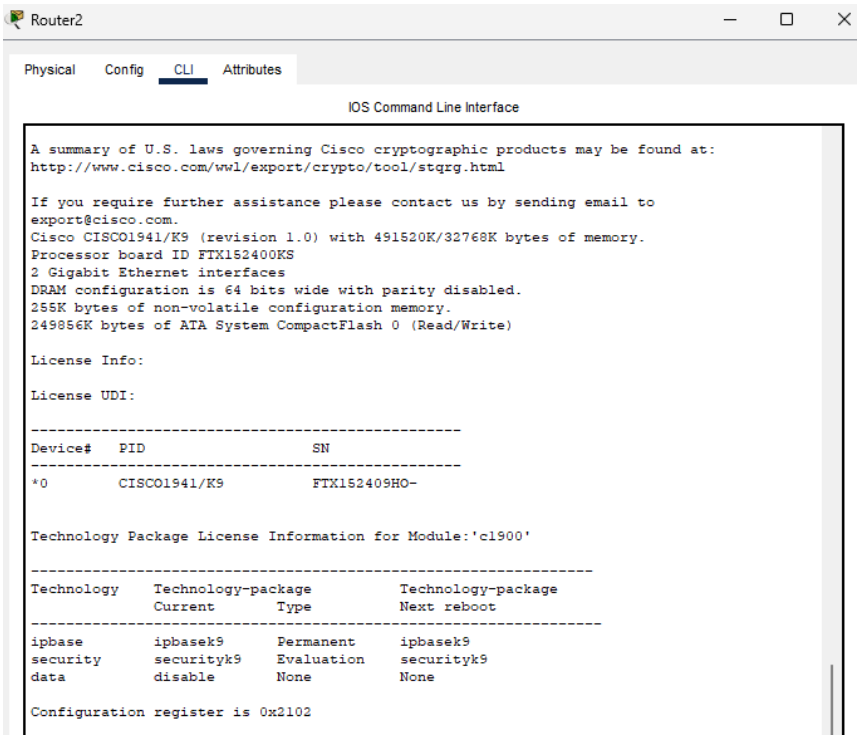
```
Router#enable
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# hostname R2
R2(config)#exit
R2#
%SYS-5-CONFIG_I: Configured from console by console
```

Step 3: Security package:
Enable security package for Router1 and Router2

Router1-

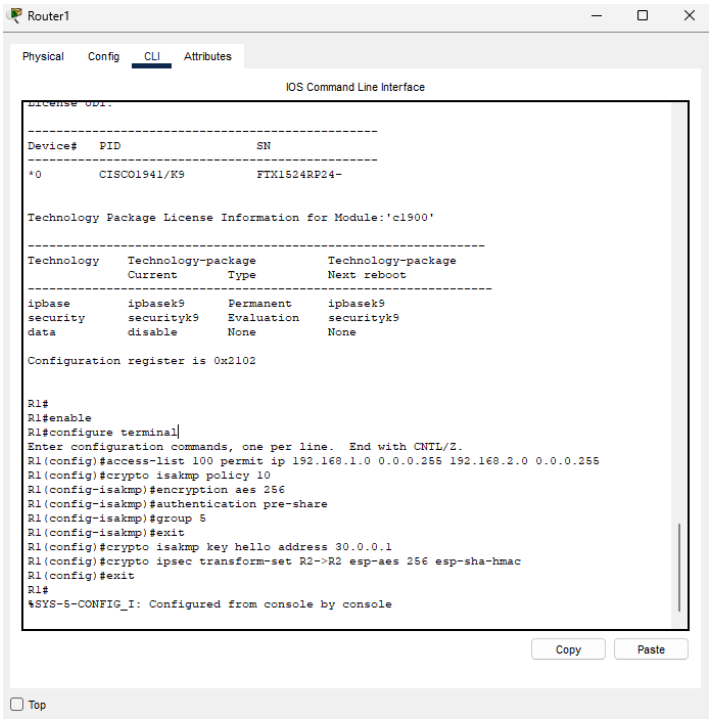


Router2-

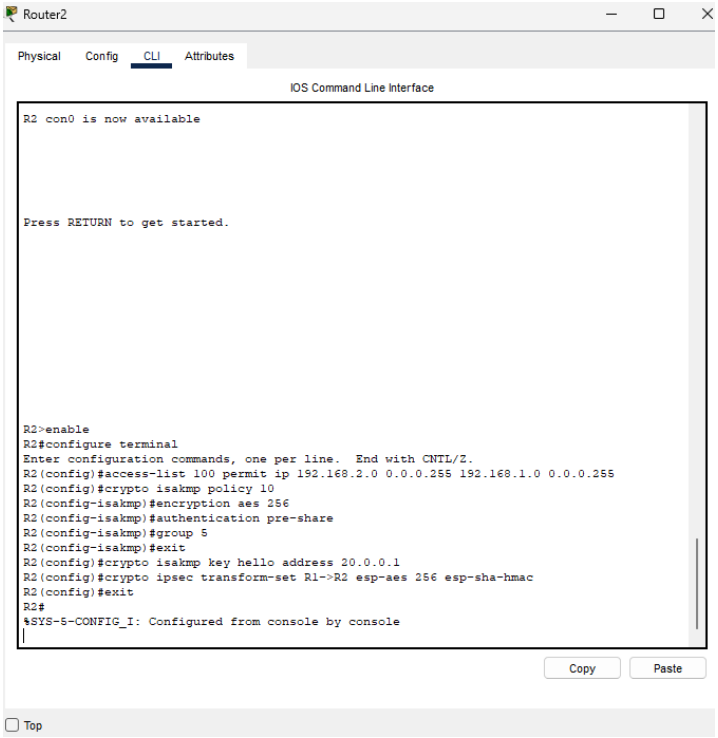


Step 4: Apply the Access Control List (ACL)-

Router1-



Router2-

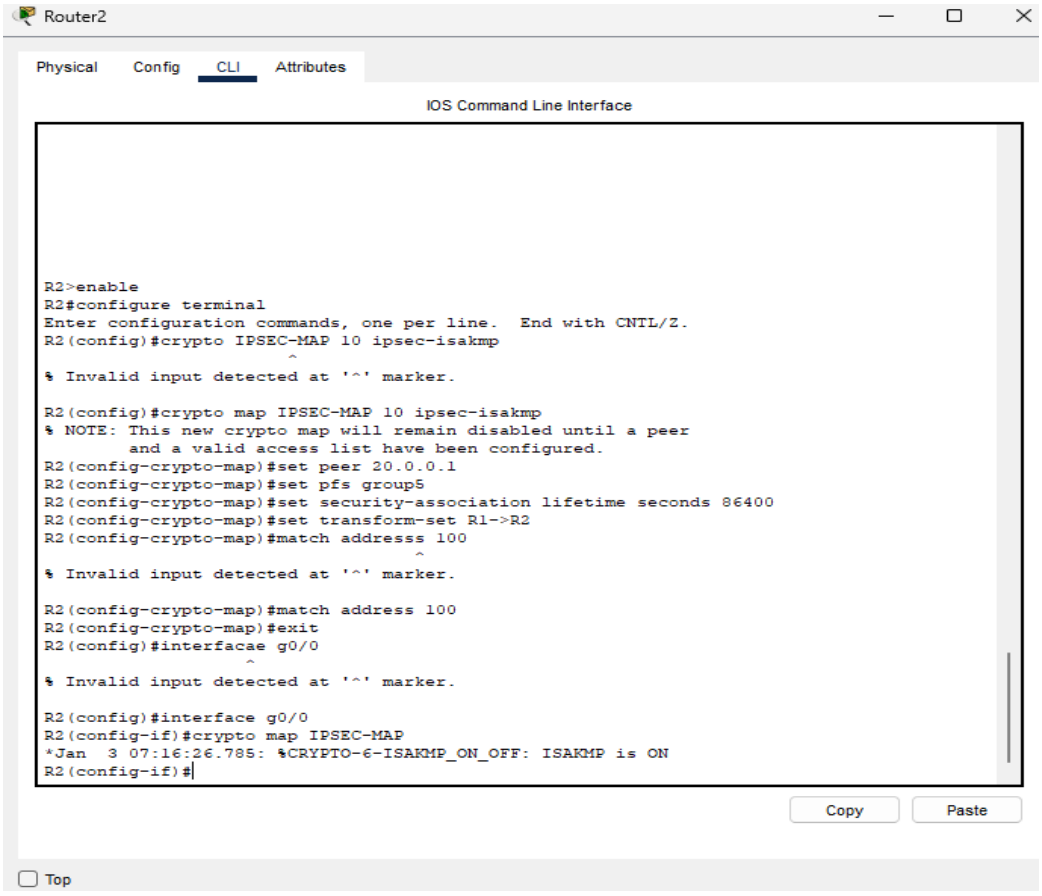


Step 5: Create a Crypto Map-

Router1-

```
R1#enable
R1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#crypto map IPSEC-MAP 10 ipsec-isakmp
% NOTE: This new crypto map will remain disabled until a peer
        and a valid access list have been configured.
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
ERROR: transform set with tag R1->R2 does not exist.
R1(config-crypto-map)#match address 100
R1(config-crypto-map)#exit
R1(config)#exit
R1#
%SYS-5-CONFIG_I: Configured from console by console
enable
R1#enable
R1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface g0/0
R1(config-if)#crypto map IPSEC-MAP
*Jan  3 07:16:26.785: %CRYPTO-6-ISAKMP_ON_OFF: ISAKMP is ON
R1(config-if)#
```

Router2-



Step 6: Verify the output by pinging the PC-

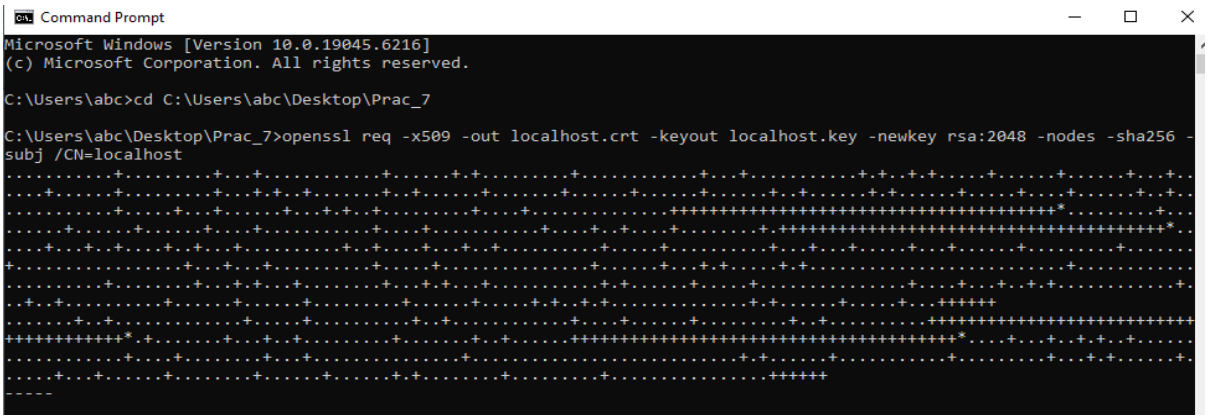
```
Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Reply from 192.168.2.2: bytes=32 time<1ms TTL=126

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 1, Lost = 3 (75% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:

Reply from 192.168.2.2: bytes=32 time<1ms TTL=126
Reply from 192.168.2.2: bytes=32 time<1ms TTL=126
Reply from 192.168.2.2: bytes=32 time<1ms TTL=126
Reply from 192.168.2.2: bytes=32 time<1ms TTL=126

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

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.

Step 1: Download FireDeamon.

Step 2: Generate a new self-signed SSL certificate and key for localhost:



Step 2: Server side Program

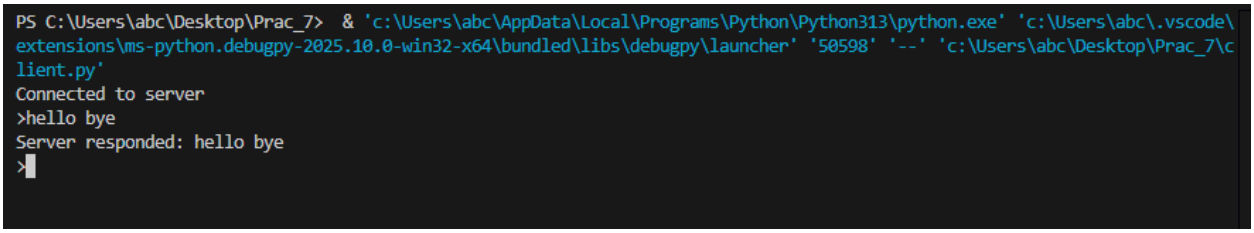
```
import socket
import ssl

context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
context.load_cert_chain(certfile="localhost.crt", keyfile="localhost.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")
    while True:
        sock, address = server.accept()
        print("New connection from", f"{address[0]}:{address[1]}")

        ssl_sock = context.wrap_socket(sock, server_side=True)

        while True:
            data = ssl_sock.recv(1024)
            decoded = data.decode('utf-8')

            if decoded == "":
                break
            print(f"[{address[0]}:{address[1]}] {decoded}")

            ssl_sock.sendall(data)
```

```
        print("Closing connection with", f"{address[0]}:{address[1]}")
        ssl_sock.close()
```

Step 3: Client side Program

```
import socket
import ssl

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
    sock.settimeout(10)

    context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
    context.load_verify_locations('localhost.crt')

    ssl_sock = context.wrap_socket(sock, server_hostname="localhost")

    ssl_sock.connect(("localhost", 4434))
    print("Connected to server")

    while True:
        ssl_sock.sendall(bytes(input(">"), "utf-8"))
        data = ssl_sock.recv(1024)
        print("Server responded:", data.decode('utf-8'))
```

OUTPUT:



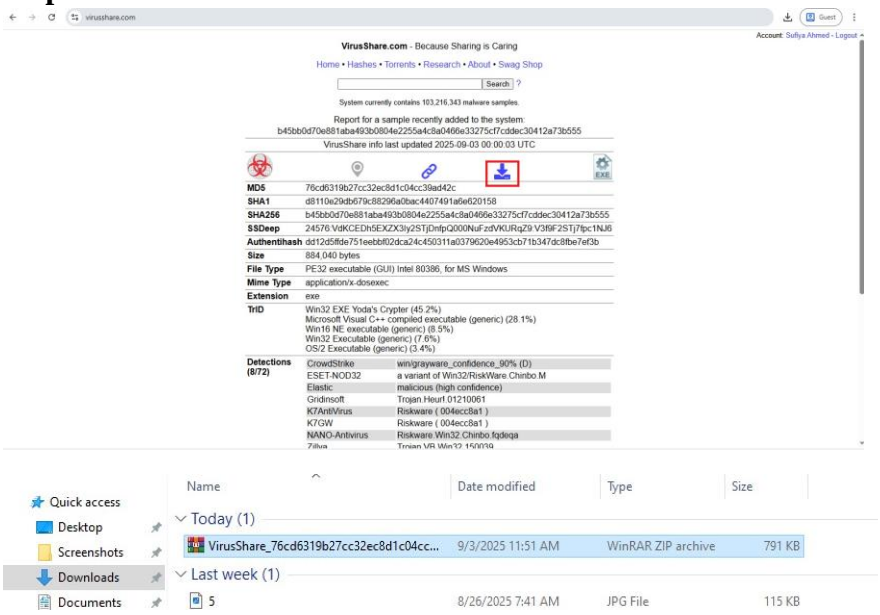
PRACTICAL 8

AIM: Malware Analysis and Detection: Analyze and identify malware samples using antivirus tools, analyze their behavior, and develop countermeasures to mitigate their impact.

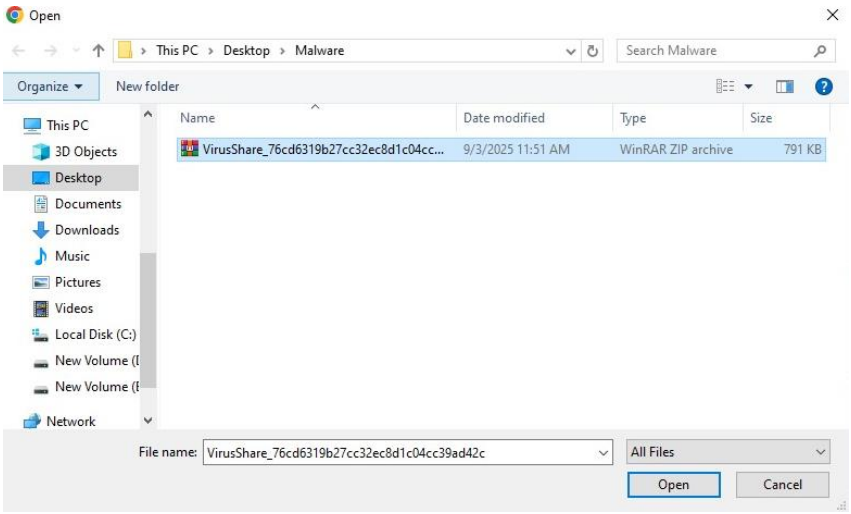
Step 1: Visit the website VirusShare.com for downloading the virus samples.

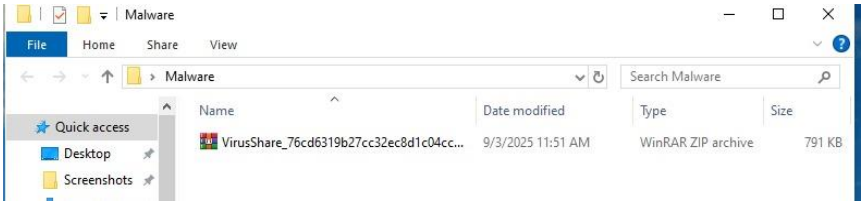
Step 2: Login:
Username: Sufiya111
Password: INSPrac9

Step 3: Click on download.

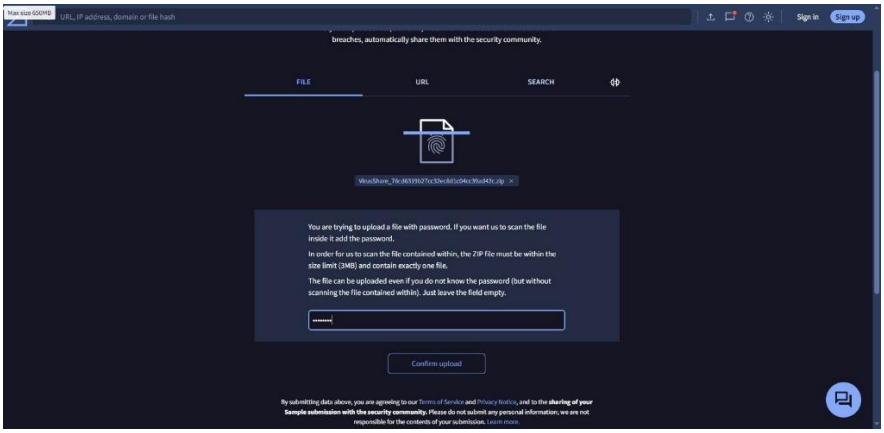


Step 4: Create a folder on the desktop name as Malware.

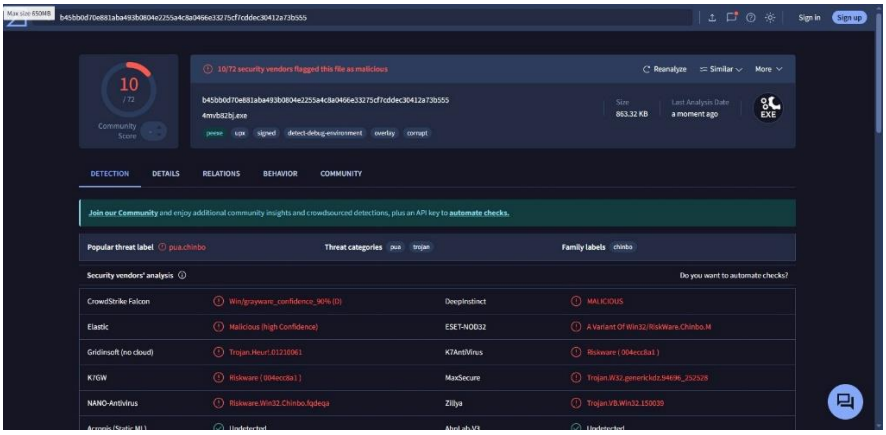




Step 5: Now scan the zip file on the website www.virustotal.com. Choose a file and scan it.



Password – infected



PRACTICAL 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.

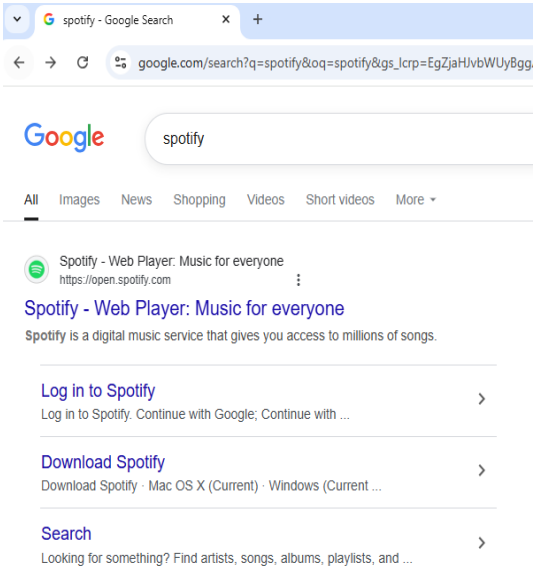
We would use firewall to block
1) A Port
2) A Program
3) A Website

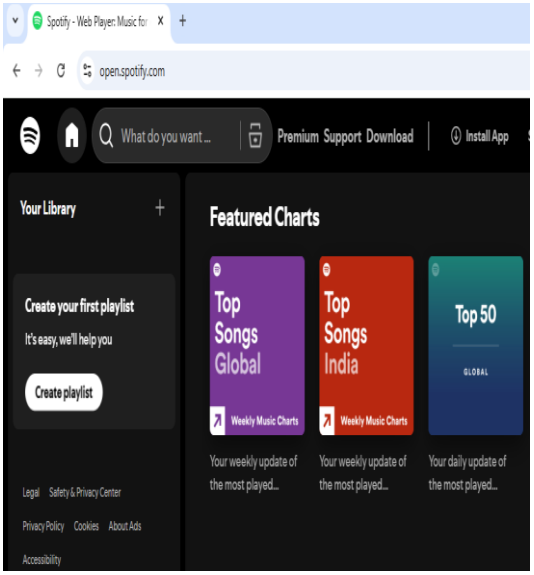
Part 1: Blocking the HTTP and HTTPS (Port 80 and Port 443) using the Firewall. Before starting with the blocking port process, we note that the applications running at the server end are identified with the well-known Port numbers, some of the commonly used are as follows.

Port Number	Protocol	Application
20	TCP	FTP data
21	TCP	FTP control
22	TCP	SSH
25	TCP	SMTP
53	UDP, TCP	DNS
80	TCP	HTTP (WWW)
110	TCP	POP3
443	TCP	SSL

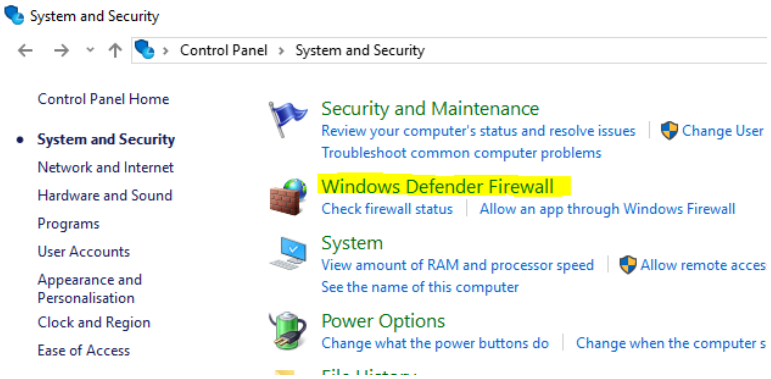
For Inbound-

Step 1: We access any website through the browser and confirm that the HTTP/HTTPS protocols are working. Go to Browser > Search Anything > Open > Working.

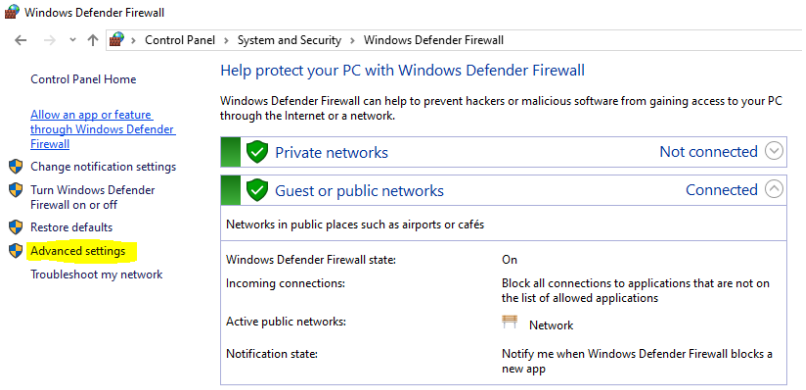




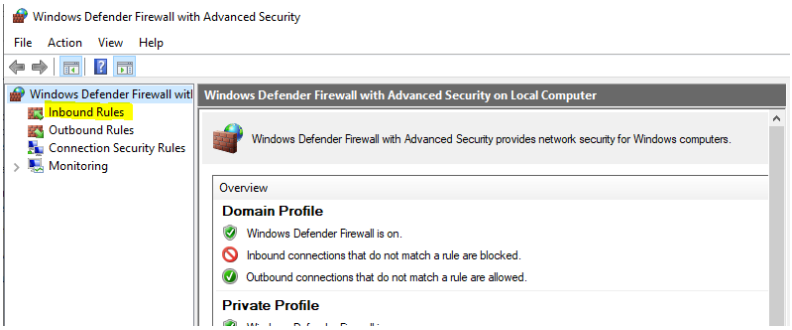
Step 2: We open ‘Windows Defender Firewall’



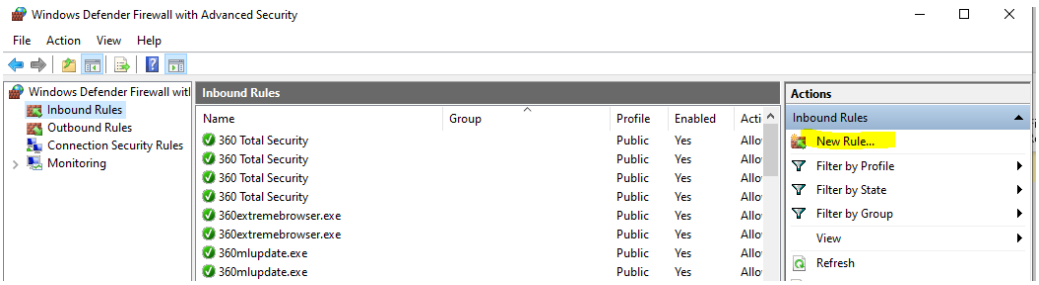
Step 3: Next, we click on ‘Advanced settings’



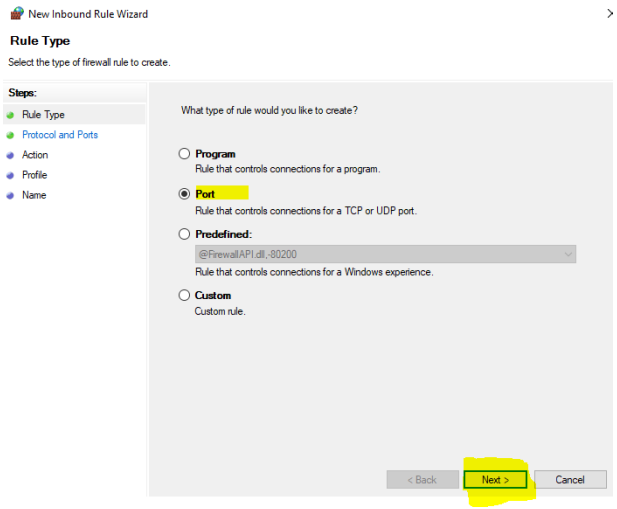
Step 4: Next go to ‘Inbound rules’



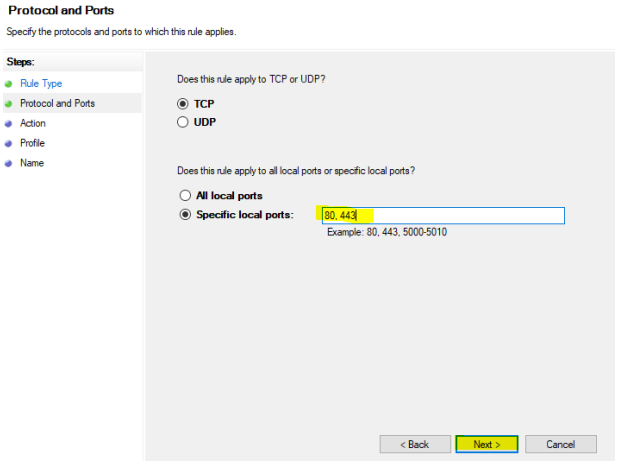
Step 5: Now create a ‘new rule’ for Inbound



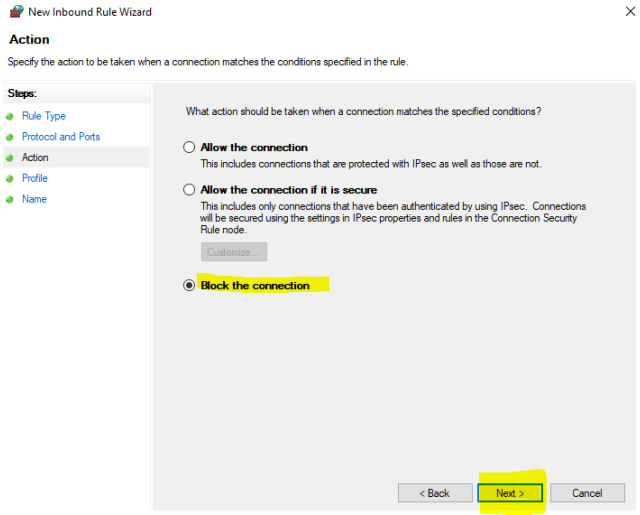
Step 6: Select ‘Port’ and click ‘Next’



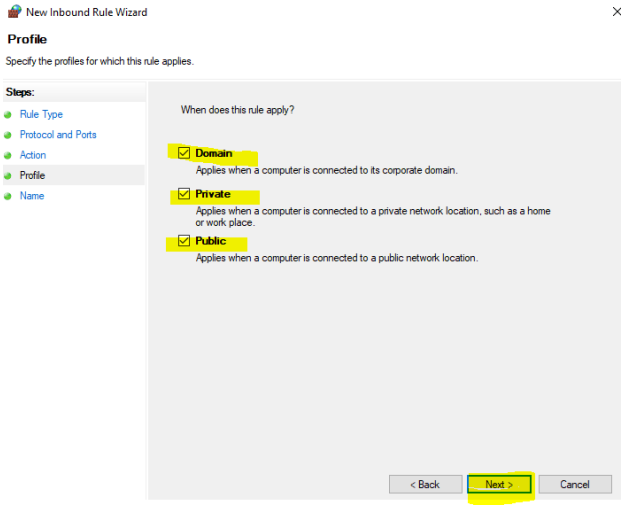
Step 7: Give Port ‘80, 443’ and click ‘Next’



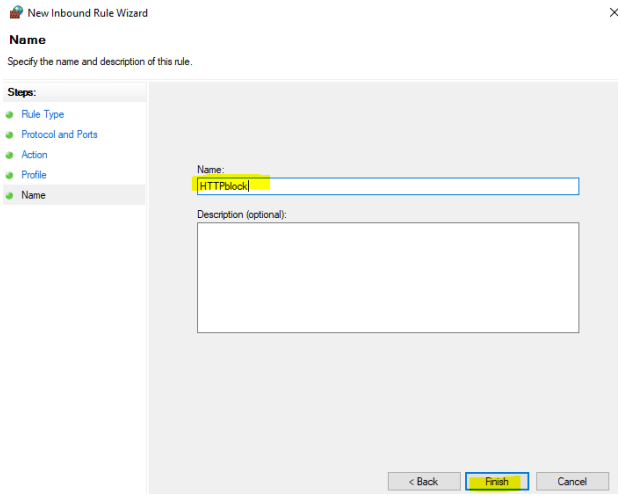
Step 8: Block the connection



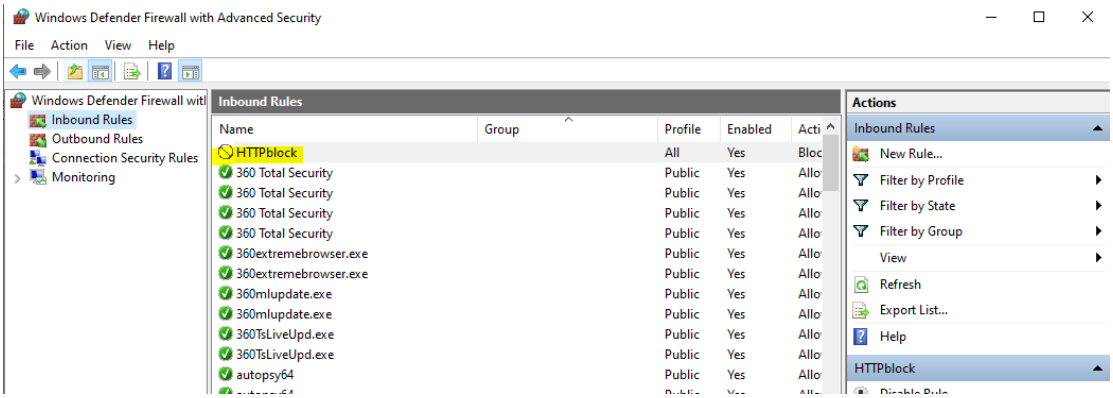
Step 9: Check all the rules application are enable



Step 10: Name the rule



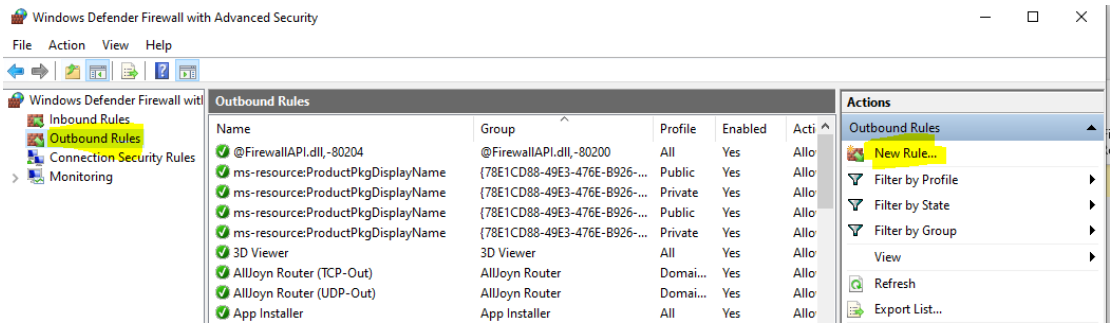
New rule for inbound is added.



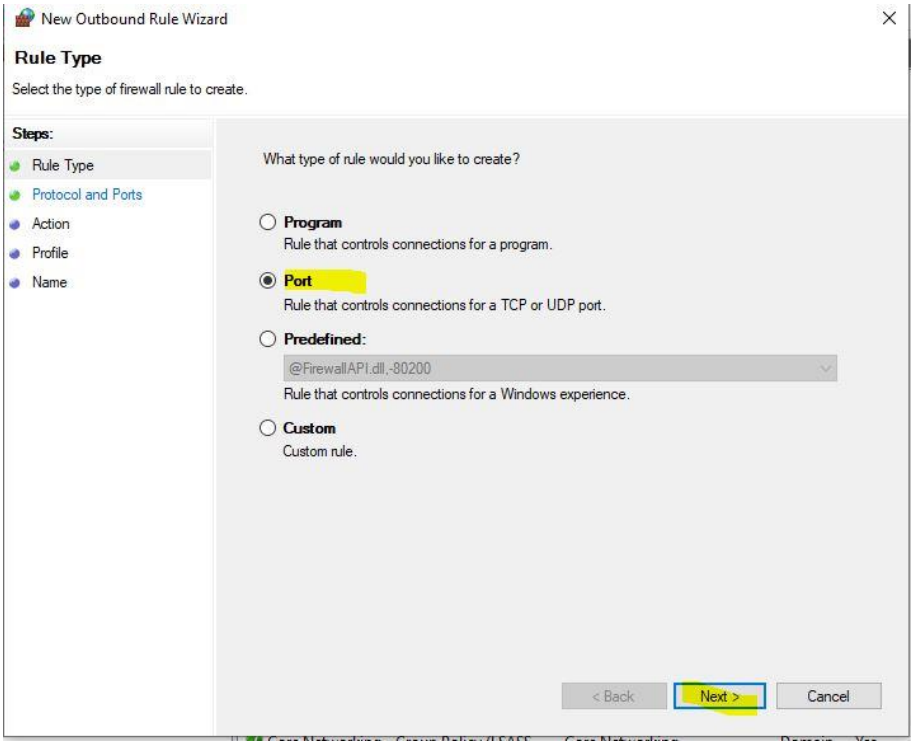
For outbound-

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

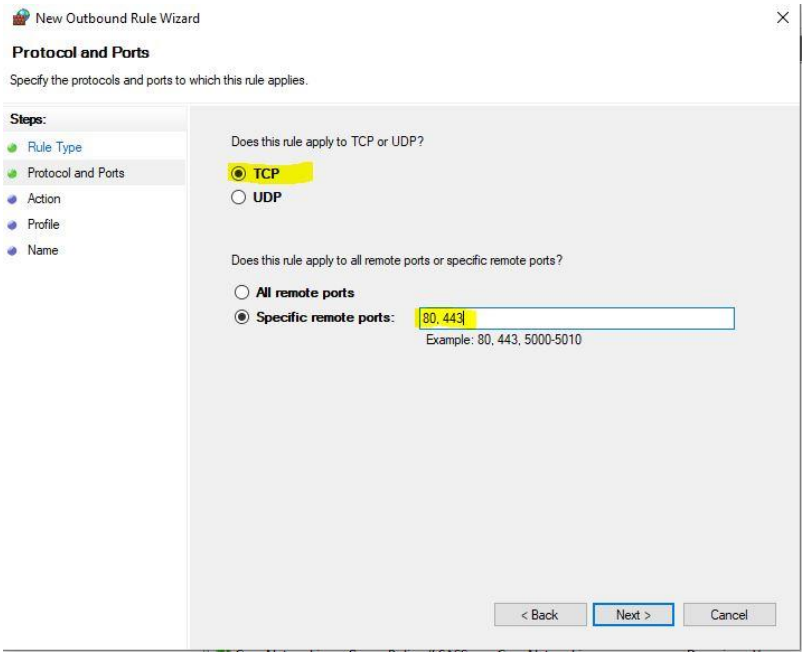
Step 1: Create a ‘new rule’ for Outbound



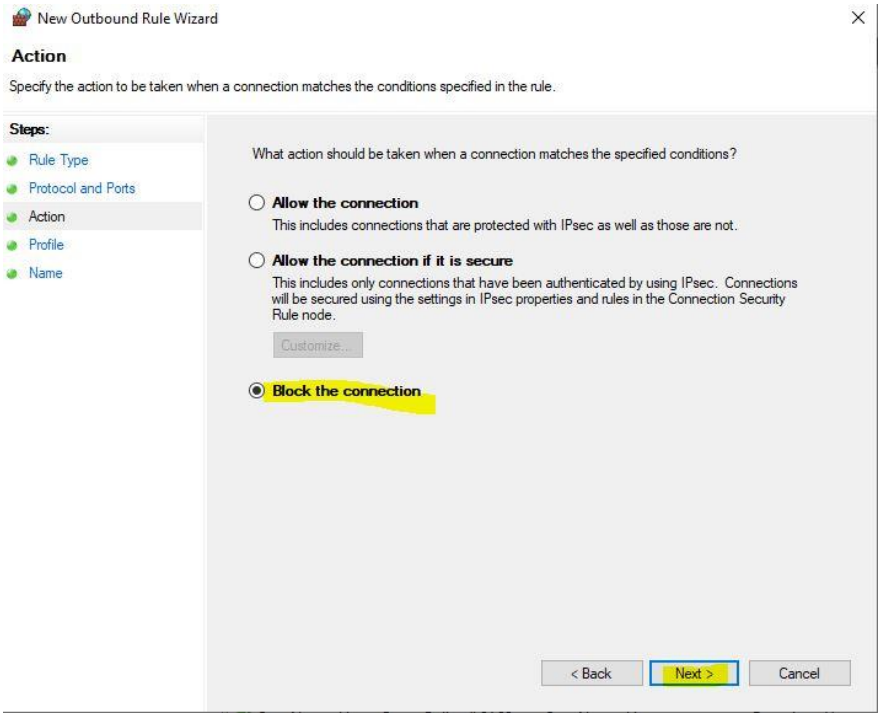
Step 2: Select ‘Port’ and click ‘Next’



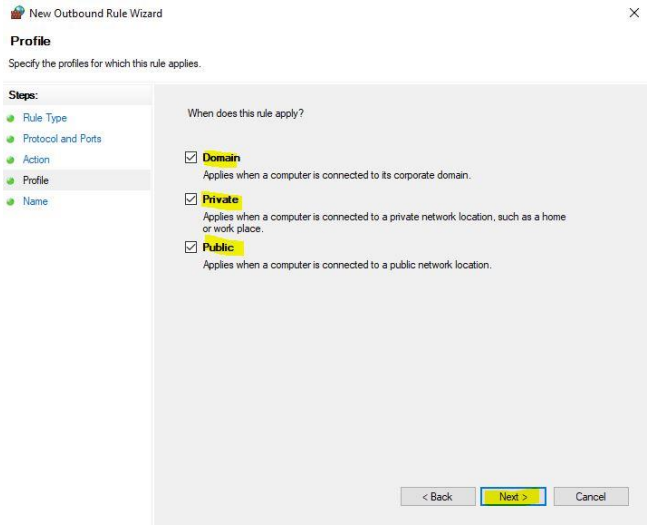
Step 3: Give Port ‘80, 443’ and click ‘Next’



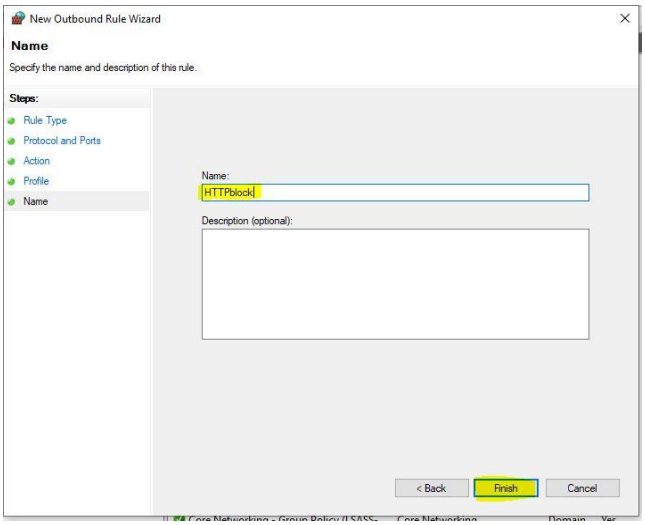
Step 4: Block the connection



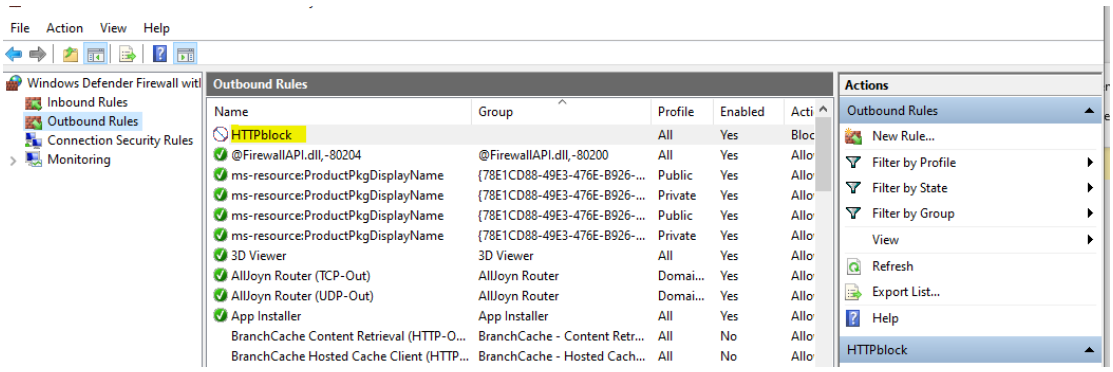
Step 5: Check all the rules application are enable



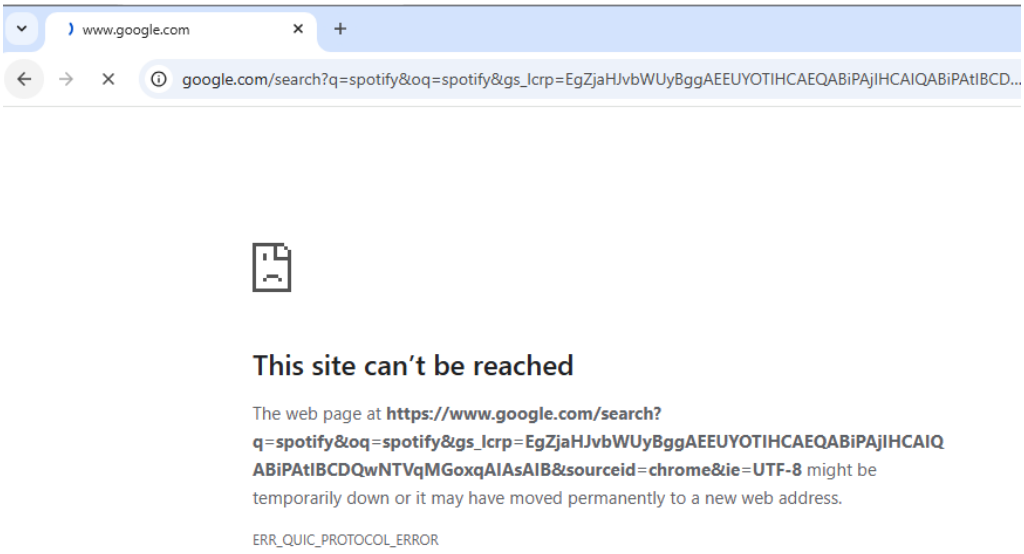
Step 6: Name the rule



New rule for outbound is added.

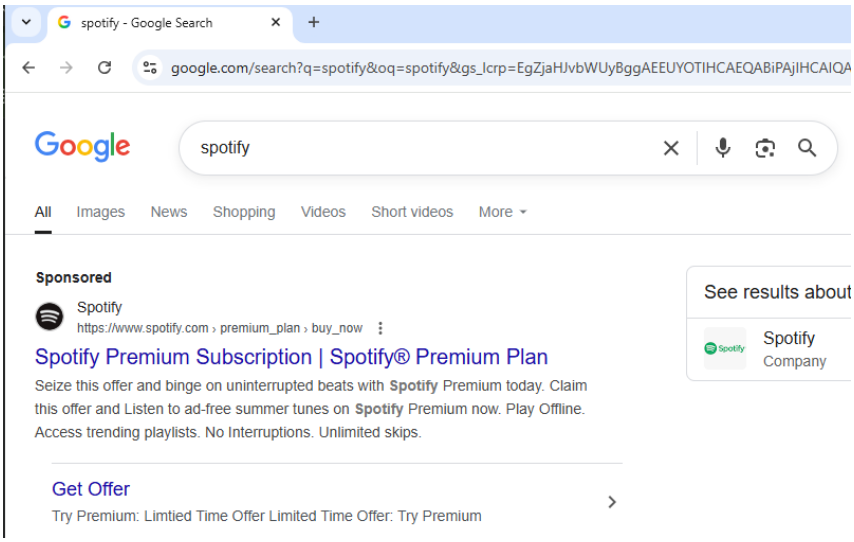


Now the access is blocked-



Now to unblock the internet:
Go to firewall > Go to inbound > Right click and delete the rule
Go to firewall > Go to outbound > Right click and delete the rule

After deleting inbound and outbound rule-



Part 2: Blocking the Program.

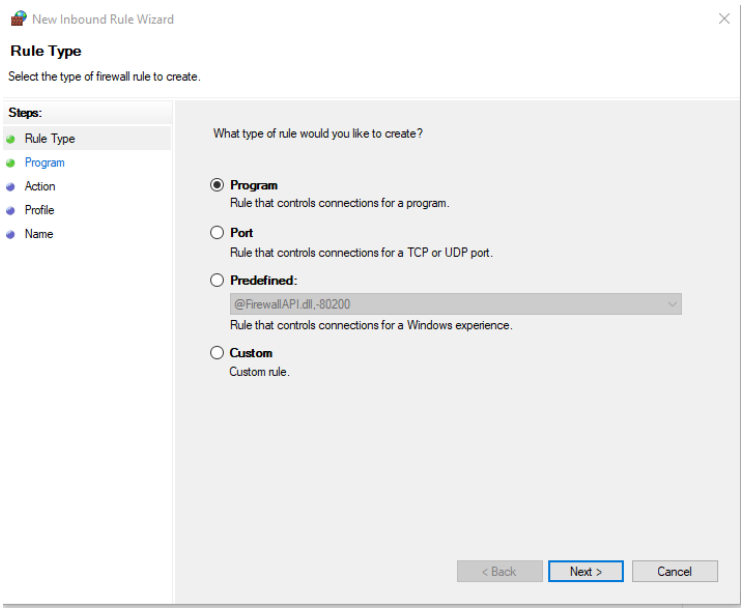
Step 1: Open Windows Defender Firewall.

Step 2: Go to Advance Settings

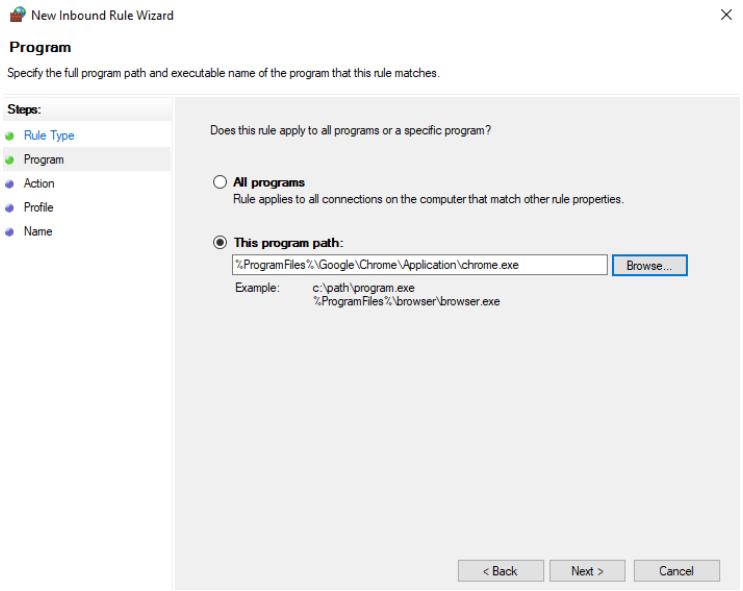
Step 3: Click on Inbound Rule

Step 4: Create new rule

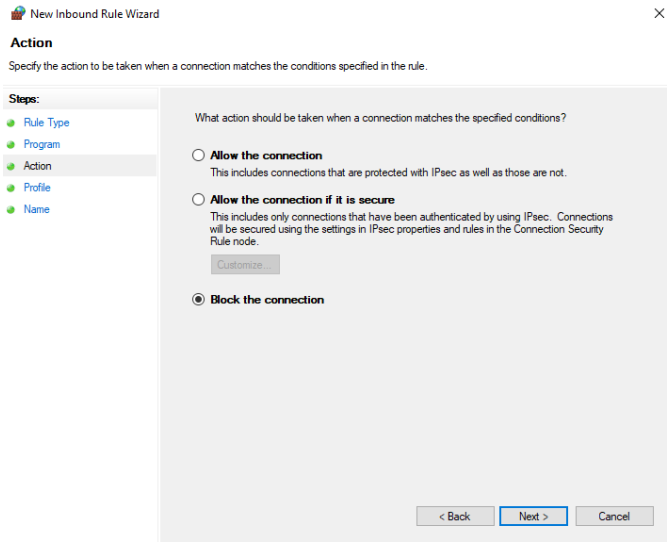
Step 5: Select Program and click on Next button



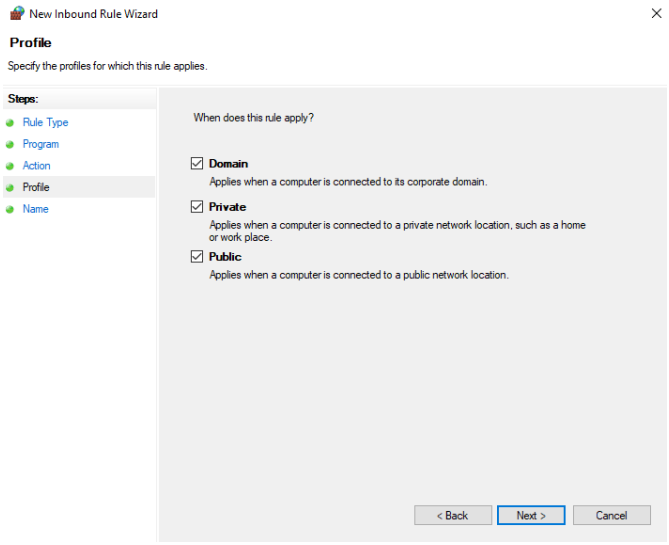
Step 6: We are blocking the chrome program so browse the path and click on next.



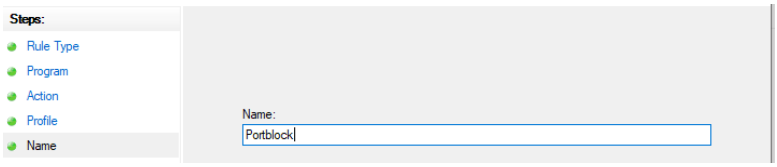
Step 7: Now click on block the connection.



Step 8: Tick all the checkboxes.

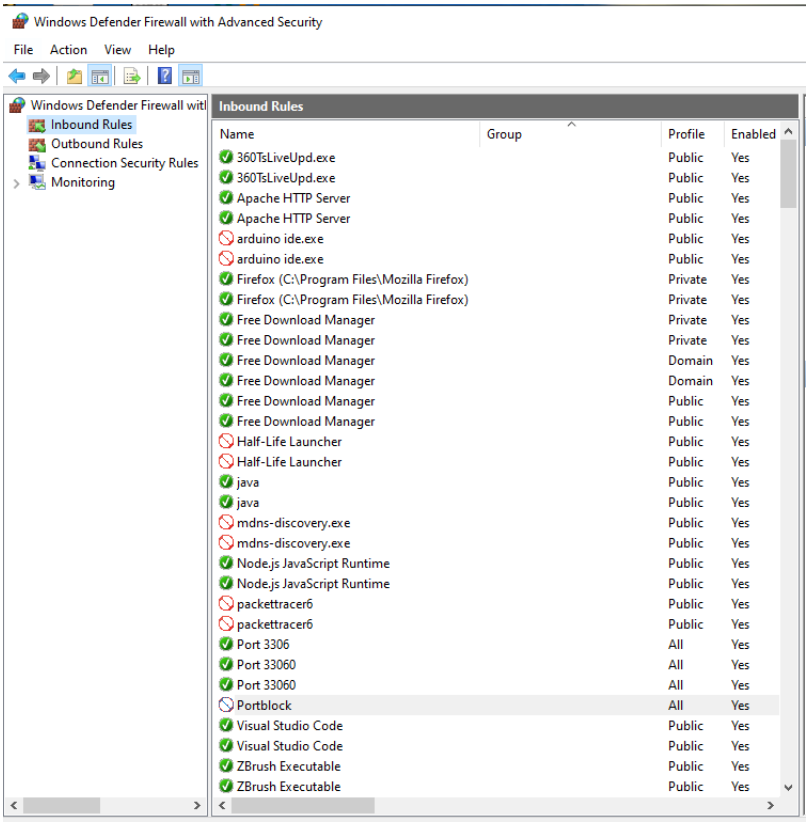


Step 9: After clicking on Next button give the name to that rule.

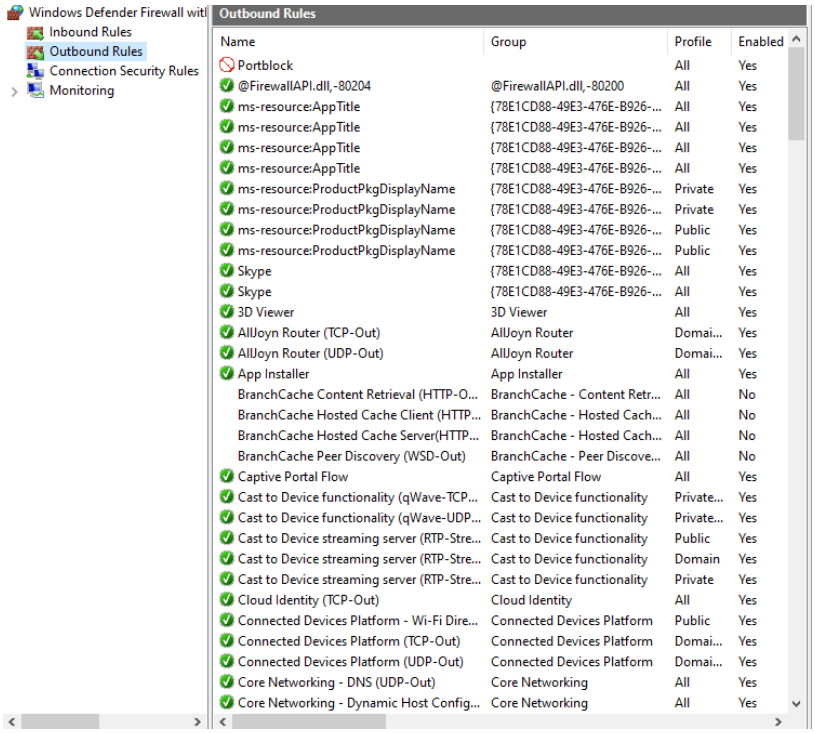


Then click on finish:

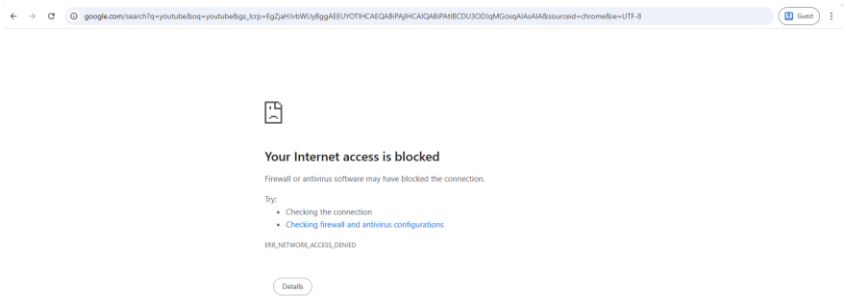
The Inbound rule is created:



Step 10: Follow the same steps for Outbound Rule.

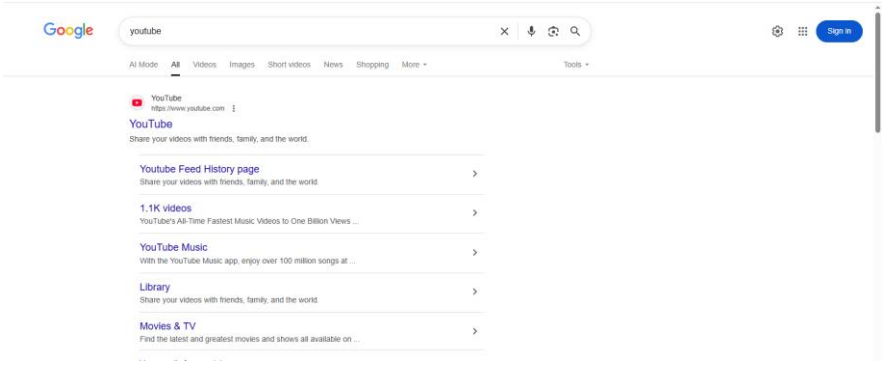


Step 11: Now open chrome and check it will show connect to internet

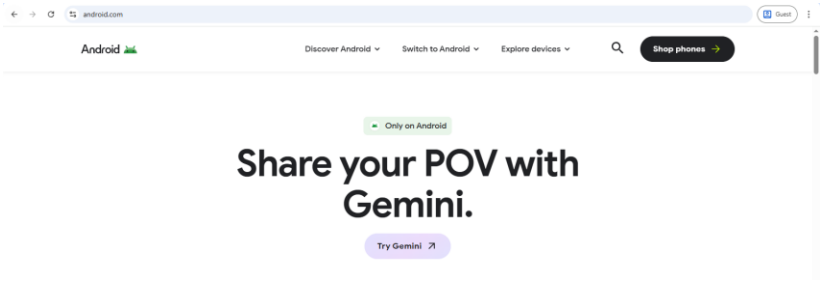


Now to unblock the Chrome:
Go to firewall > Go to inbound > Right click and delete the rule
Go to firewall > Go to outbound > Right click and delete the rule

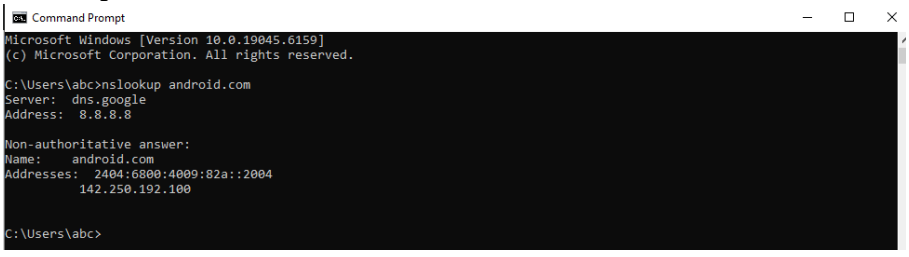
After deleting inbound and outbound rule-



Part 3: Blocking the website android.com
We open the browser and access the website, which is now accessible.



We find the IP addresses of the website using the following command
nslookup android.com



We save the IP addresses
IPV4 - 142.250.192.100
IPV6 - 2404:6800:4009:822::2004

We will make Inbound and Outbound rule for IP addr

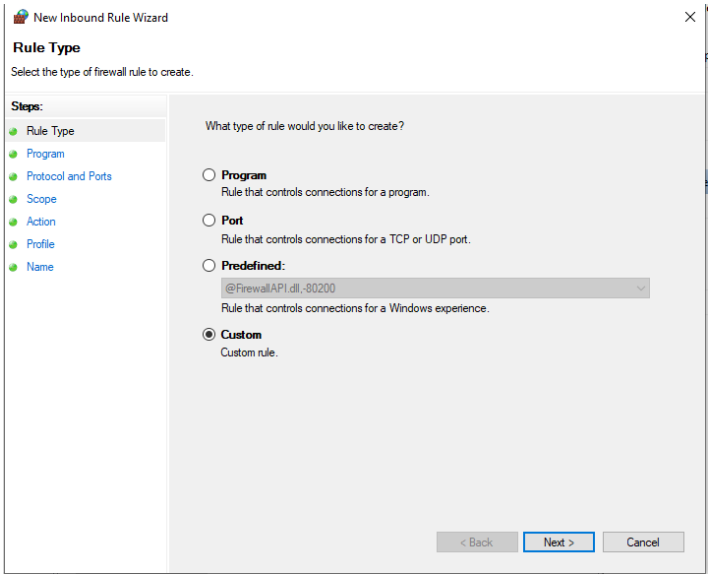
Step 1: Open Windows Defender Firewall.

Step 2: Go to Advance Settings

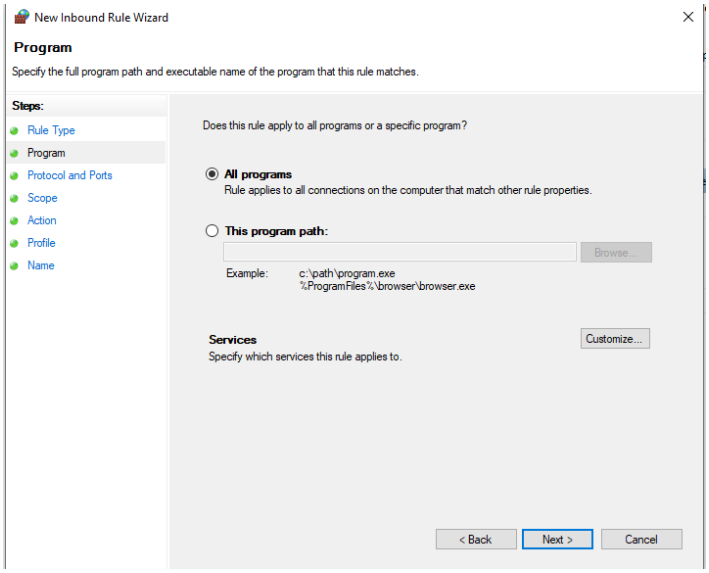
Step 3: Click on Inbound Rule

Step 4: Create new rule

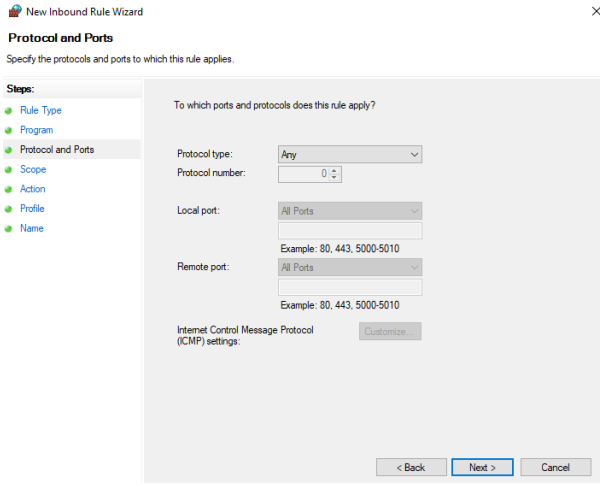
Step 5: Select Custom and click on Next button



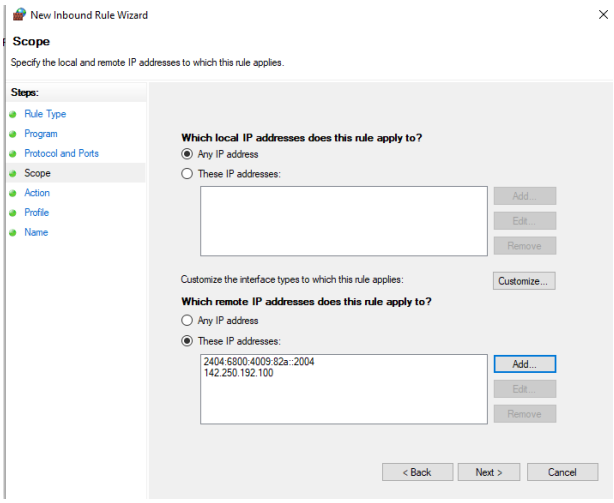
Step 6: Select “All programs” and click on Next button



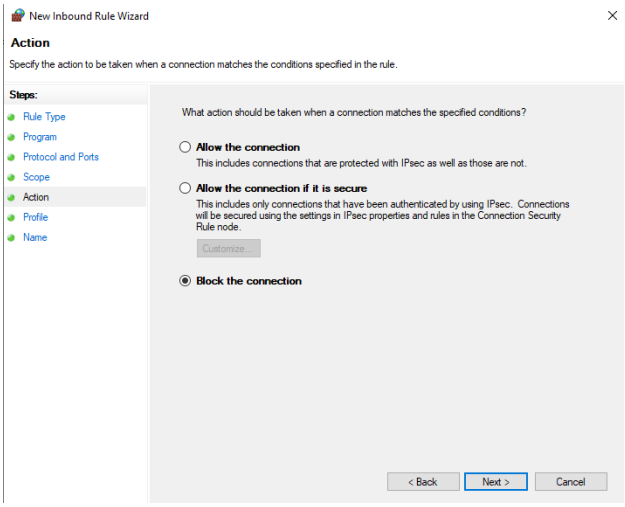
Step 7: Click on Next button



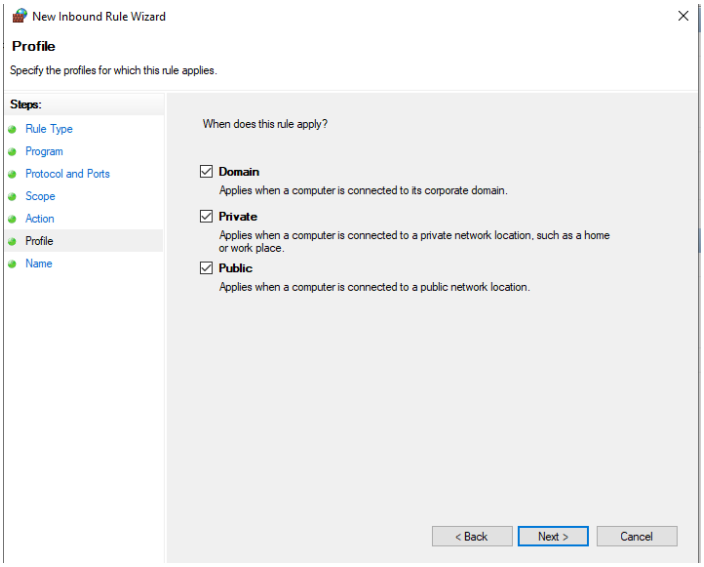
Step 8: Add both the addresses



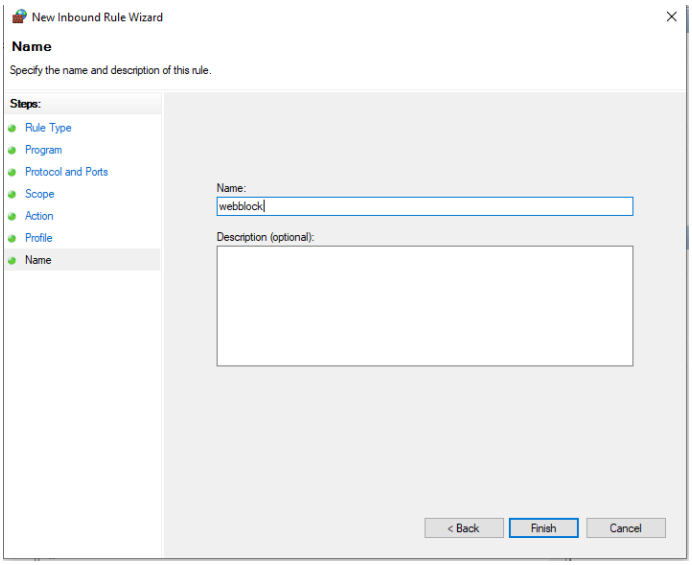
Step 9: Block the connection



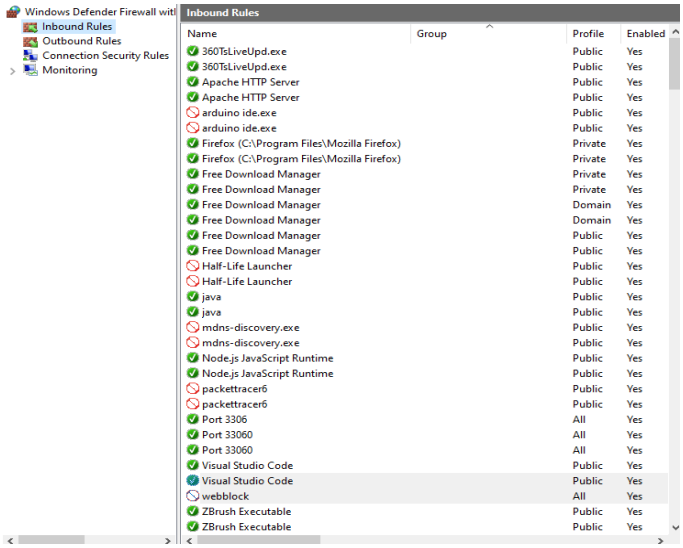
Step 10: Check all the boxes



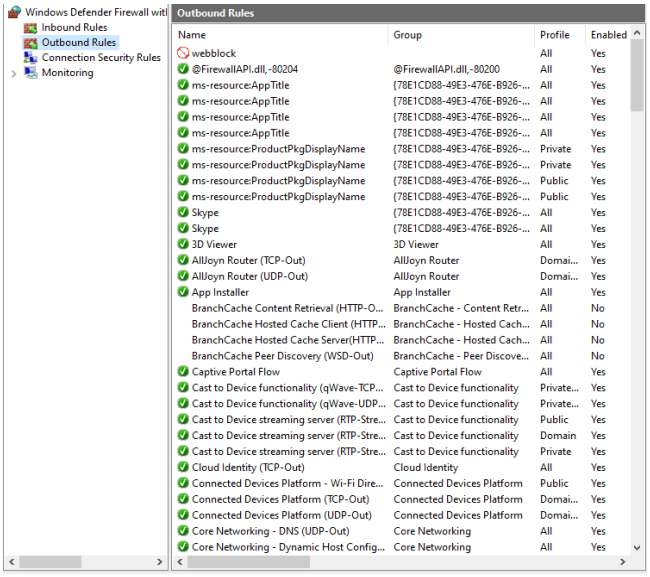
Step 11: Name the rule and click on finish.



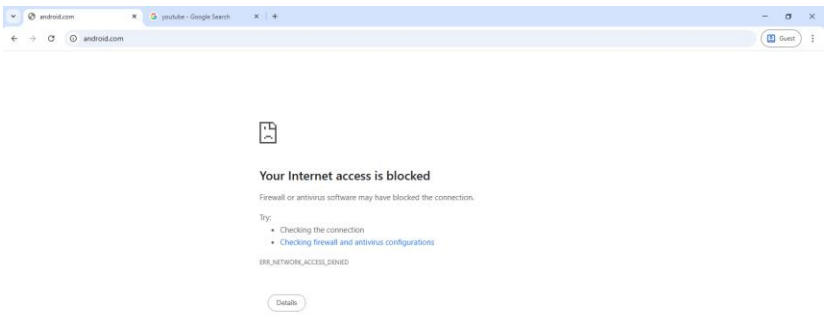
The Inbound Rule is created:



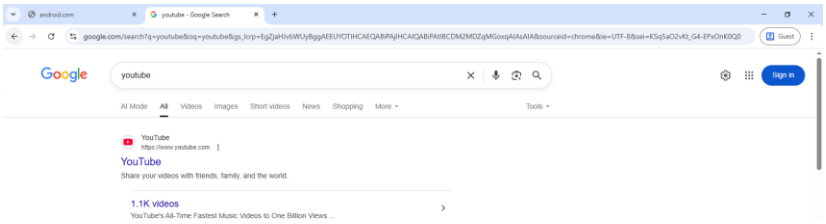
Now, follow the same steps for creating the outbound rule:



Go to browser and search android.com



Simultaneously we checked for other search its working properly.



Now to unblock the website:
Go to firewall > Go to inbound > Right click and delete the rule
Go to firewall > Go to outbound > Right click and delete the rule

After deleting inbound and outbound rule-