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:

```
=============== RESTART: C:/Users/itisha mishra/Documents/ins.py ============
Enter String : itisha
enter Shift Number : 2
1. For Encrypt
2. For Decrypt
 Enter Your choice : 1
Original String :  itisha
After Encryption :  kvkujc

=============== RESTART: C:/Users/itisha mishra/Documents/ins.py ============
Enter String : kvkujc
enter Shift Number : 2
1. For Encrypt
2. For Decrypt
 Enter Your choice : 2
Original String :  kvkujc
After Decryption :  itisha
```

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:

```
========== RESTART: C:/Users/itisha mishra/Documents/ins.py ==========
Enter Value:
1) Encrypt Text
2) Decrypt Text
3) See Key
4) Exit
Choice: 1
Enter Text to Encrypt: hello what's up!
Encrypted Text: SVOOL DSZG'H FK!
Enter Value:
1) Encrypt Text
2) Decrypt Text
3) See Key
4) Exit
Choice: 2
Enter Text to Decrypt: hisgs!
Decrypted Text: SRHTH!
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:

```
= RESTART: C:/Users/itisha mishra/Documents/ins.py
Enter key : MONARCHY

 1.Encryption
 2.Decryption:
 3.EXIT
 Enter Your Choice:
 1
ENTER MSG : PIZZA
CIPHER TEXT: SF UZ XR
 1.Encryption
 2.Decryption:
 3.EXIT
 Enter Your Choice:
 2
ENTER CIPHER TEXT:sfuzxr
PLAIN TEXT: PI ZX ZA
```

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:

```
= RESTART: C:/Users/itisha mishra/Documents/ins.py
Enter Key: bcusge
Enter Plaintext: itisha
CipherText:  jvckne
PlainBack:  itisha
```

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

```
import math
key = "HACK"
def encryptMessage(msg):
    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):
    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:

```
=============== RESTART: C:/Users/itisha mishra/Documents/ins.py ===
Encrypted Message: oH owmoTr_C emoemor_
Decryped Message: Come Home Tomorrow
```

Program 6: Write a python program to implement Railfence 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:

```
=============== RESTART: C:/Users/itisha mishra/Documents/ins.py ===
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.

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:

```
============== RESTART: C:\Users\itisha mishra\Documents\ins.py ==============
Public key:  (n=0xda3a147395ef3f9de5382d386f9092be2c80cf0a3825adcd41ed6beba8a53584af395d183be73e1a83228311e3d6e77bccb9456f5473ed3403b9690ad704e3e10a869b6382db9e8ca4dbfd548bf78df11300ec9e20fa69fcbaa8bd66ca0a638198fbaf19226f85c3e7d27e
267c63b11555aadad5a77c1611395088fbf93a4ceb, e=0x10001)
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDaOhRz1e8/neU4LThvkJK+LIDP
Cjg1rc1B7WvrqKUihK85XRg75z4agyKDEePW53vMuUVvVHPtNAO5aQrXBOPhCoab
Y4Lbnoyk2/1Ui/eN8RMA7J4q+mn8uqi9ZsoKY4GY+68ZIm+Fw+fSfiZ8Y7EVVara
lad8FhE5UIj7+TpM6wIDAQAB
-----END PUBLIC KEY-----
Private key: (n=0xda3a147395ef3f9de5382d386f9092be2c80cf0a3825adcd41ed6beba8a53584af395d183be73e1a83228311e3d6e77bccb9456f5473ed3403b9690ad704e3e10a869b6382db9e8ca4dbfd548bf78df11300ec9e20fa69fcbaa8bd66ca0a638198fbaf19226f85c3e7d27e
267c63b11555aadad5a77c1611395088fbf93a4ceb, d=0x57324edab285c2f329d46d093856d9b3a05fa59aad3dacf92f871910b435b3313dbc9b8ca3deedab354bb4789103dd41d3d711bdfece4570fbadf9a169b21c05221952f833c3c2b2b45de5bd47ffc97e7d3a3ea443a97e79df1dd381
d8b6dfe1704c638146cdee9f453ac32c4aa03e5829589119d6e4abb8a29572b1b3918a59)
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQDaOhRz1e8/neU4LThvkJK+LIDPCjg1rc1B7WvrqKU1hK85XRg7
5z4agyKDEePW53vMuUVvVHPtNAO5aQrXBOPhCoabY4Lbnoyk2/1Ui/eN8RMA7J4g
+mn8uqi9ZsoKY4GY+68ZIm+Fw+fSfiZ8Y7EVVaraladFhE5UIj7+TpM6wIDAQAB
AoGAVsJO2rKFwvMplGOJOFbZs6BfpZqtPazSL4cZELQ1szE9vJuMo9TtqrVLtHiR
A9iB09c8vf7ORXD7zfmhabIcB5IZUvgzw8KyrF31vUf/yX590j6KQ4l+ed5d04HY
tt/hcExjgUbN7p9FGsMsSqA+WC1YkRnW5Ku4opVysbOR1lkCQQDiTZ7v5FMRz6Vd
1QCY+xJGZtyD1Zt+TgVa23d4PxHCQ4e2Yk0U+iLZOrYdWnkVdcmuYvXh5faOTO6
SGVFOgL5AkEA9t01AdnTd9ZEAAlhoh5HZD11T+WKA24FBMlaQAIOMDQmmNNiS1t
Qe/X/yvRMXnZAmaaFVPpt81GzpB8mTtkAwJBAOH2GOLx2nroeOuOQgB+gzrmFJ0s
5ovhH5uvE8VvRC4y+61yxz4qrkGxTIVCB6/EvoWeMzHJdReCklHow6Wx3RECQH7r
/13wP5BqgN#+XQs8JBp+cdgtH4cLXqw7181fbdq4D9TKx4KS74yqIt5YE6zraY6Z
b/RtHon+mOk8+2rRIz8CQQCYGmAQmRa8f10swgmblFJz4XFK7fDwmc3ZaWBFW
9y2vJFCOUTJD1QbobqbUul7kPphnkmSrCSKtO5mMXly4
-----END RSA PRIVATE KEY-----
Encrypted: 89afdc9b51830a53341f18ce140a22502fa2a1be1dc3765c9beaa5adf400f3b6ea9b35c33bd7cb6b4548be1c17ba2598a014693f3a5c4ec273c733b7a8103d89c221a9f54bc6637d6e48d8fab4093b4b49d226c1cd3f25e3e0425e5bcb95deb4c3cdafc4a55f4134d95782345d2b5
a6e1dcd2162e488a25e49abaa6864df4749
```

PRACTICAL 3

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

Program 1:

```
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:

```
=============== RESTART: C:\Users\itisha mishra\Documents\ins.py ===============
The byte equivalent of hash is : b"\x8b\xla\x99S\xc4a\xl2\x96\xa8'\xab\xf8\xc4x\
x04\xd7"
The byte equivalent of hash is : b'\xce=.Zp\x88\xl7\x84q\x8c\xaa\x92#Z\xc9\xa7'
```

Program 2:

```
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:

```
=============== RESTART: C:\Users\itisha mishra\Documents\ins.py =============
Enter the value to encode what's up?
The hexadecimal equivalent of SHA1 is :
ad979180a27c6cf6dc368c9928e53438774c5277
```

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:

```
=============== RESTART: C:\Users\itisha mishra\Documents\ins.py ==============
Signature is valid.

=============== RESTART: C:\Users\itisha mishra\Documents\ins.py ==============
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:

```
=============== RESTART: C:\Users\itisha mishra\Documents\ins.py ============
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.

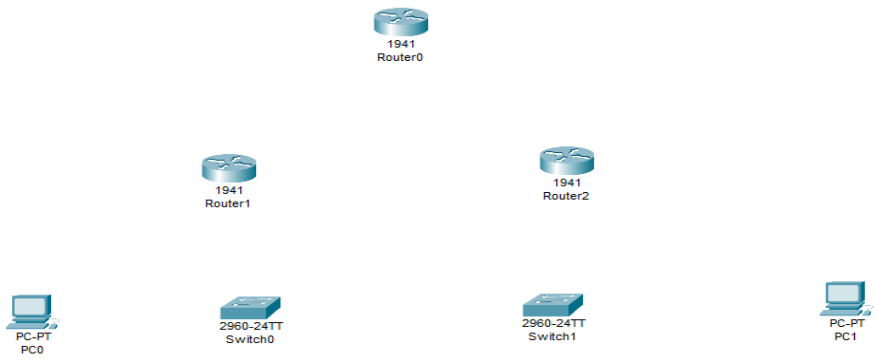
Requirements:

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

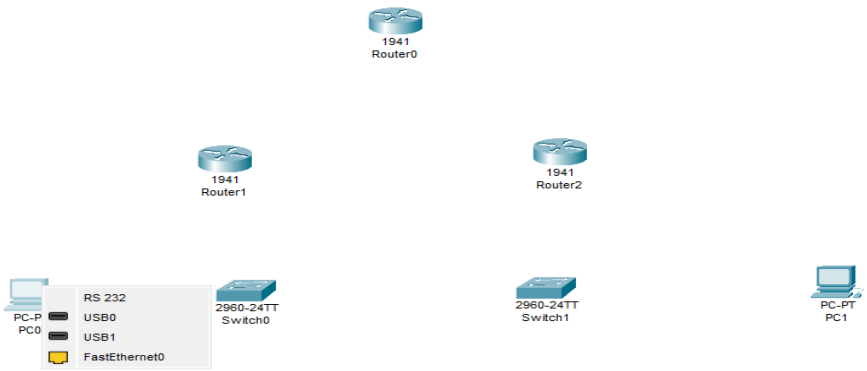
  PC 0 – 192.168.1.2
  PC 1 – 192.168.2.2
  Router 1 (G 0/0) - 20.0.0.1
  Router 1 (G 0/1) - 192.168.1.1
  Router 0 (G 0/0) - 30.0.0.2
  Router 0 (G 0/1) - 20.0.0.2
  Router 2 (G 0/0) - 30.0.0.1
  Router 2 (G 0/1) - 192.168.2.1

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

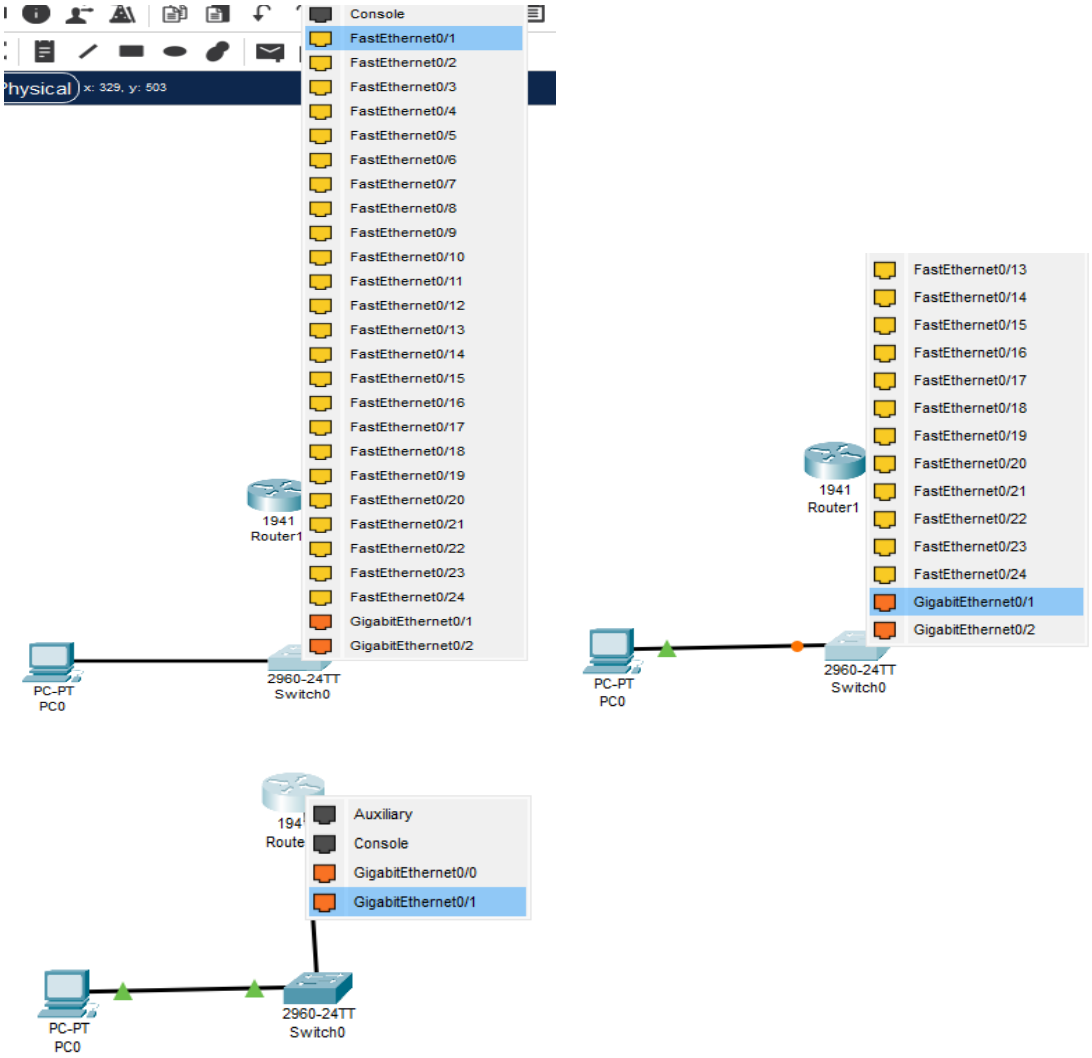
Connect the PC with Switches and also connect switches with Router using copper straight wire as given below.



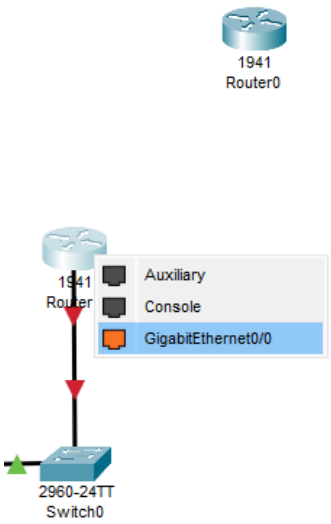
Connect with copper straight wire

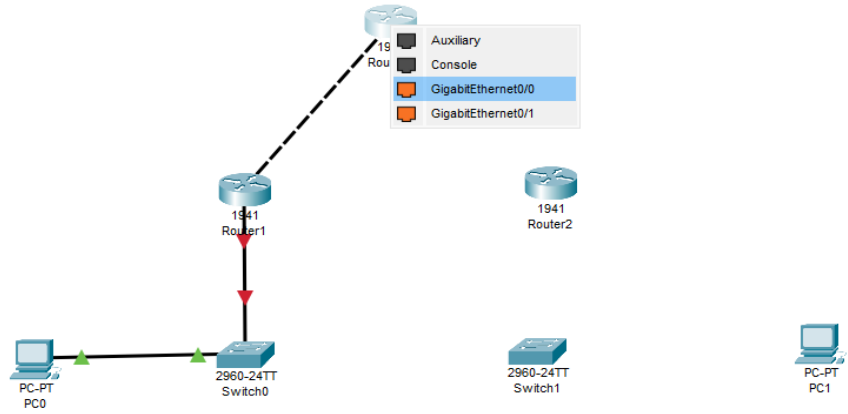


Connect the PC to the ethernet (FastEthernet0/24) and connect it with switch with GigabitEthernet0/1

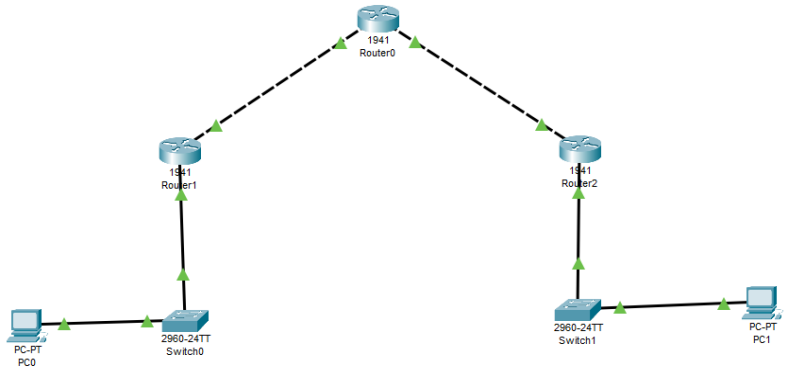


Connect the Router 1 with Router 0 and Router 0 with Router 2 using copper cross wire as given below.





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
```

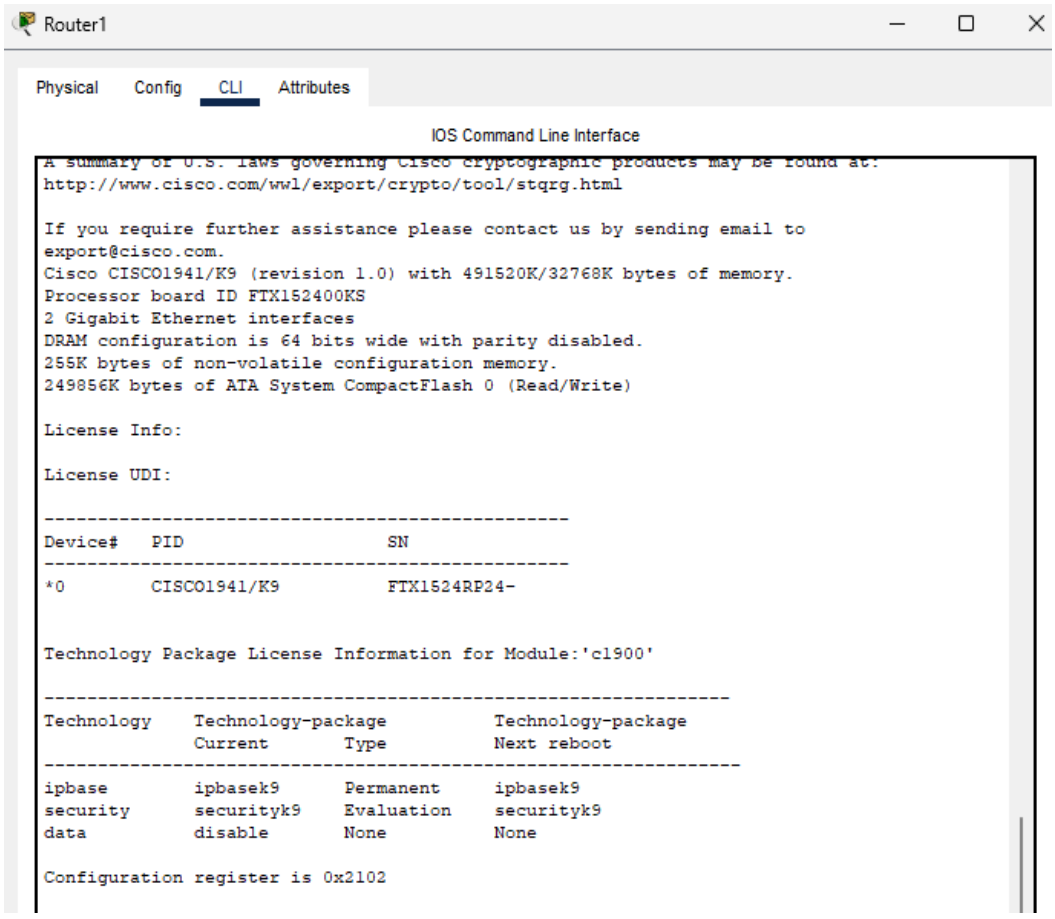
Router2

```
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
```

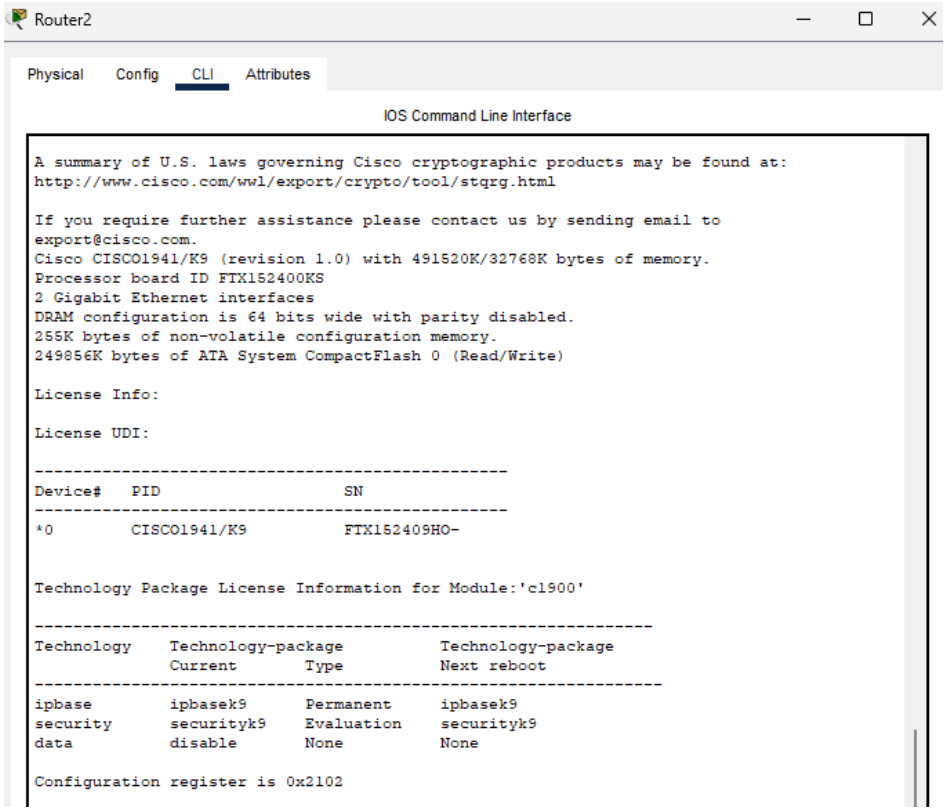
Step 3: Security package:

Enable security package for Router1 and Router2

Router1-

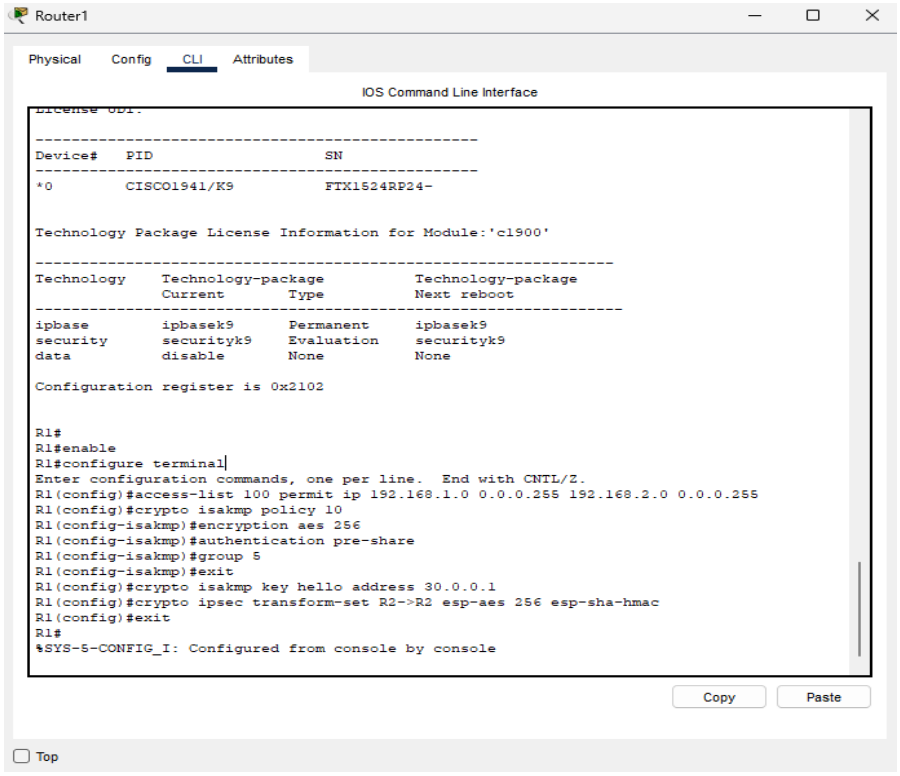


Router2-

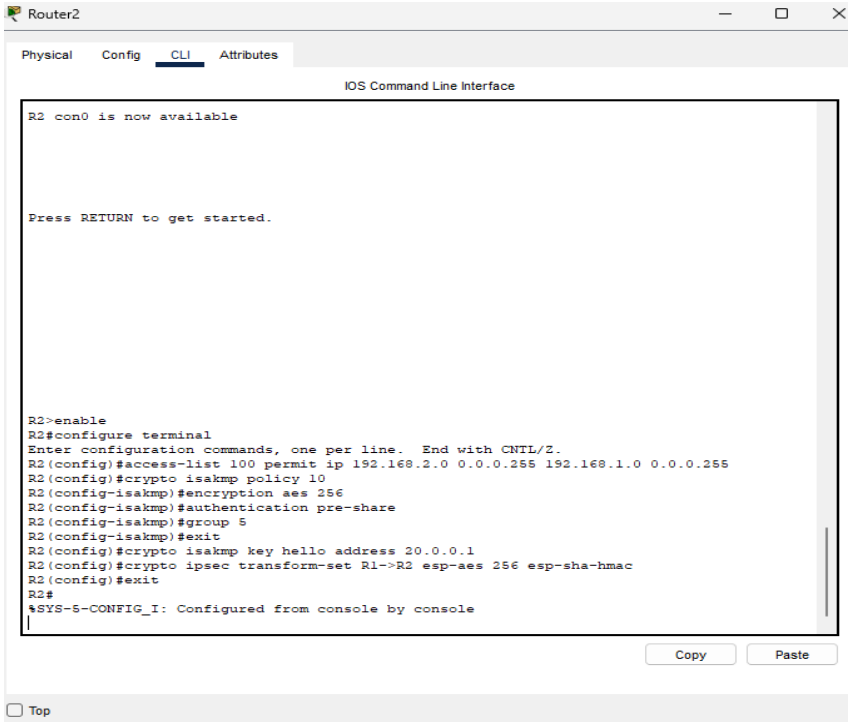


Step 4: Apply the Access Control List (ACL)-

Router1-



Router2-

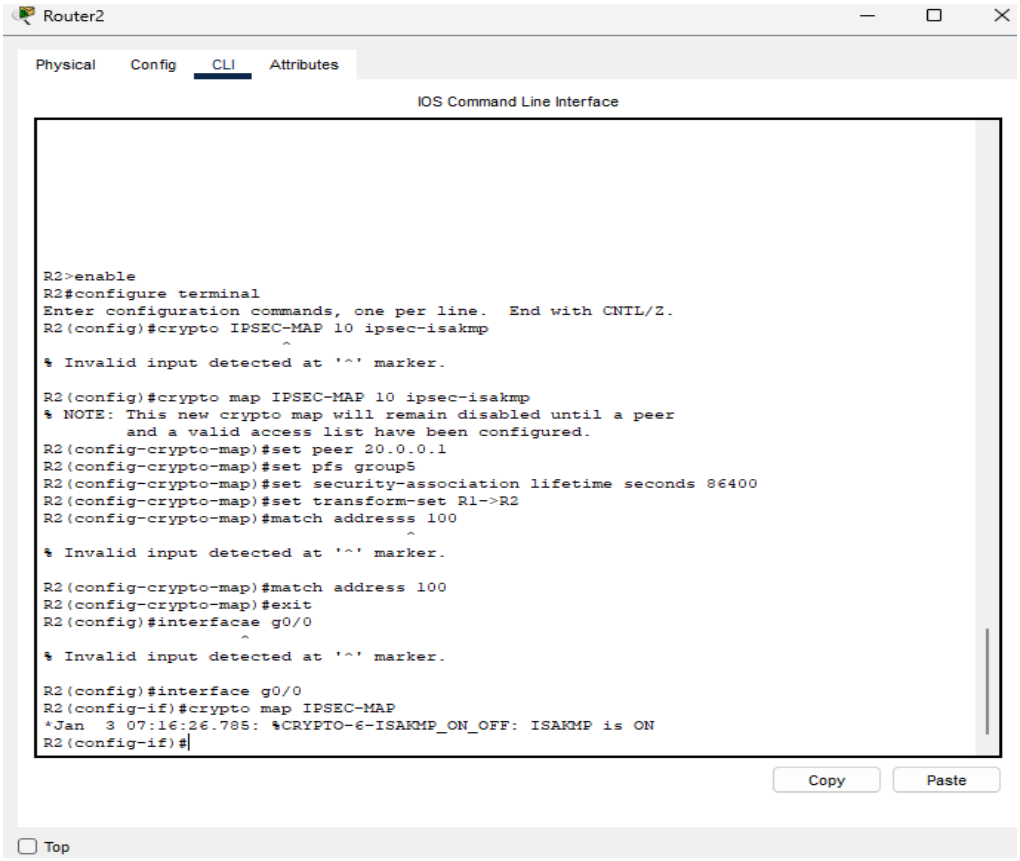


Step 5: Create a Crypto Map-

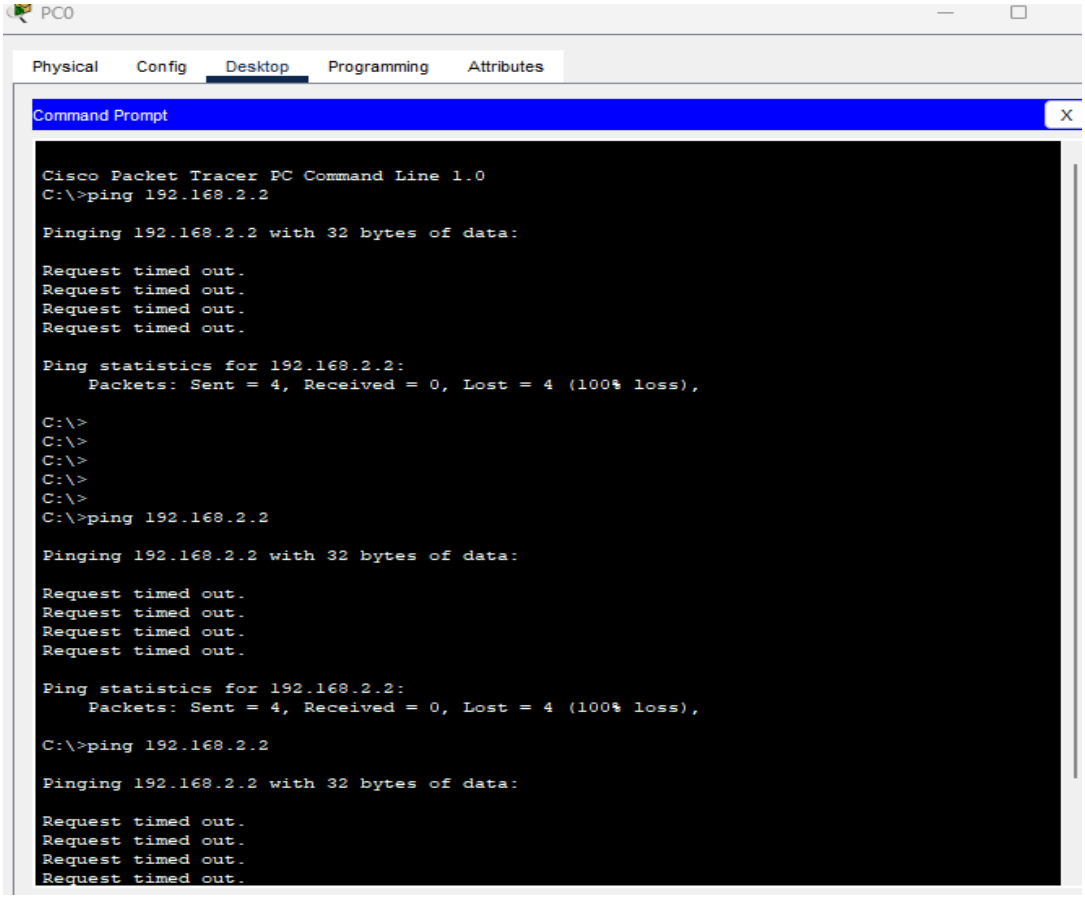
Router1-

```
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-

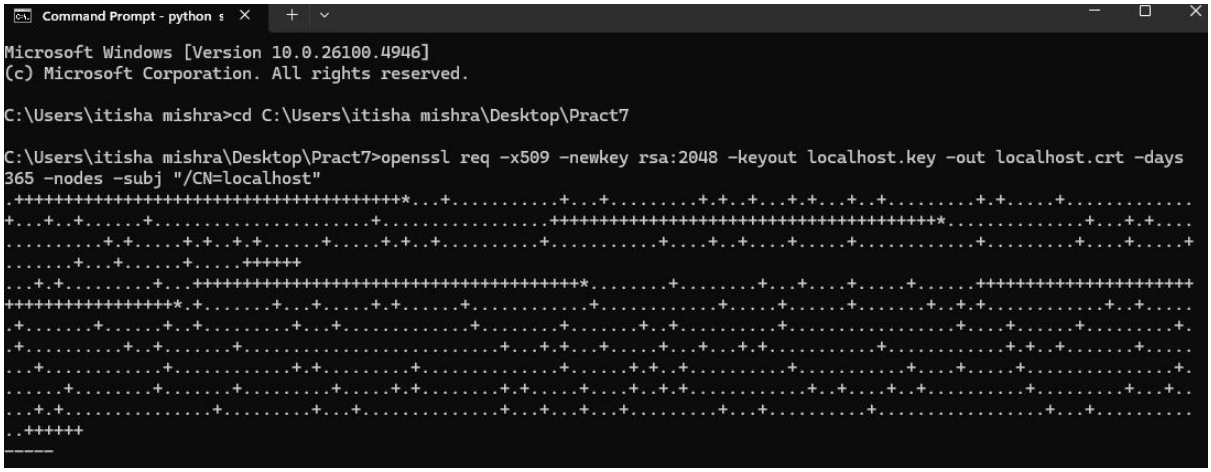


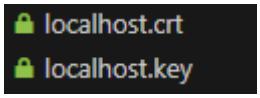
PRACTICAL 7

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: 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(("", 8443))
    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: Client Side Program

```
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('localhost.crt')
    ssl_sock = context.wrap_socket(sock, server_hostname="localhost")
    # Connect to the server
    ssl_sock.connect(("localhost", 8443))
    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'))
```

Output:

- **Run the server.py file -**

```
TERMINAL
PS C:\Users\itisha mishra\Desktop\Pract7> & "C:/Program Files/Python312/python.exe" "c:/Users/itisha mishra/Desktop/Pract7/server
.py"
Server ready and listening for connections
```

- **Keep the server running, and run the client.py file –**

```
PS C:\Users\itisha mishra\Desktop\Pract7> python client.py
Connected to server
>
```

- **Server output -**

```
PS C:\Users\itisha mishra\Desktop\Pract7> & "C:/Program Files/Python312/python.exe" "c:/Users/itisha mishra/Desktop/Pract7/
server.py"
Server ready and listening for connections
New connection from 127.0.0.1:50500
```

- **Test the connection by sending a message from the client by typing in the console.**

```
PS C:\Users\itisha mishra\Desktop\Pract7> python client.py
Connected to server
>SSL Secured!
Server responded: SSL Secured!
>
```

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

```
PS C:\Users\itisha mishra\Desktop\Pract7> & "C:/Program Files/Python312/python.exe" "c:/Users/itisha mishra/Desktop/Pract7/
server.py"
Server ready and listening for connections
New connection from 127.0.0.1:50500
[127.0.0.1:50500] SSL Secured!
```

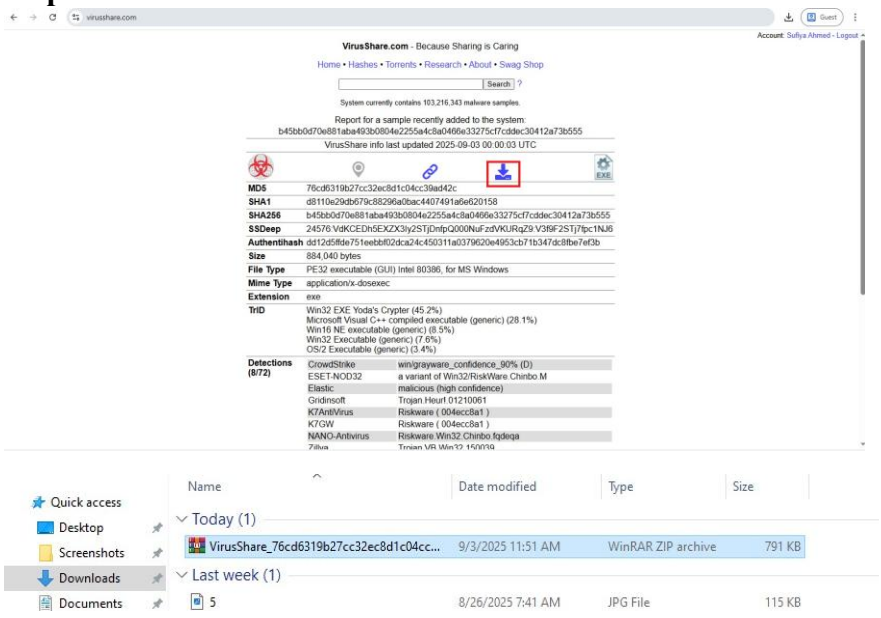
PRACTICAL 8

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

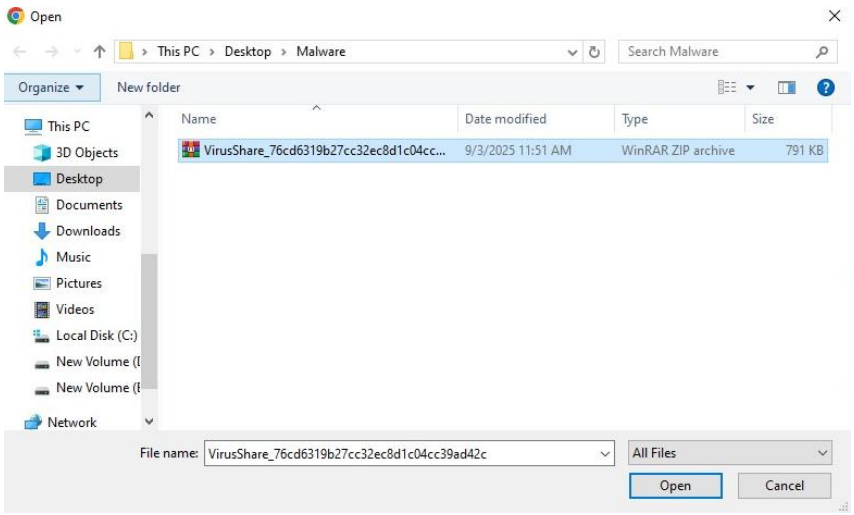
Step 1: Visit the website VirusShare.com for downloading the virus samples.

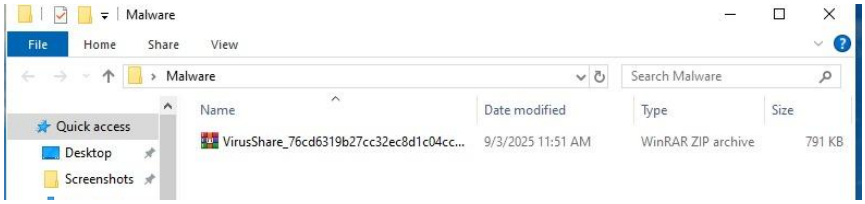
Step 2: Login:
Username: Sufiya111
Password: INSPrac9

Step 3: Click on download.

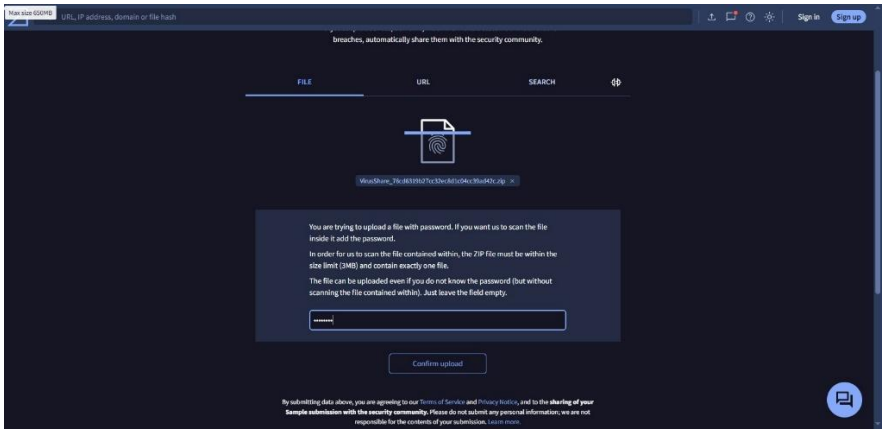


Step 4: Create a folder on the desktop name as Malware.

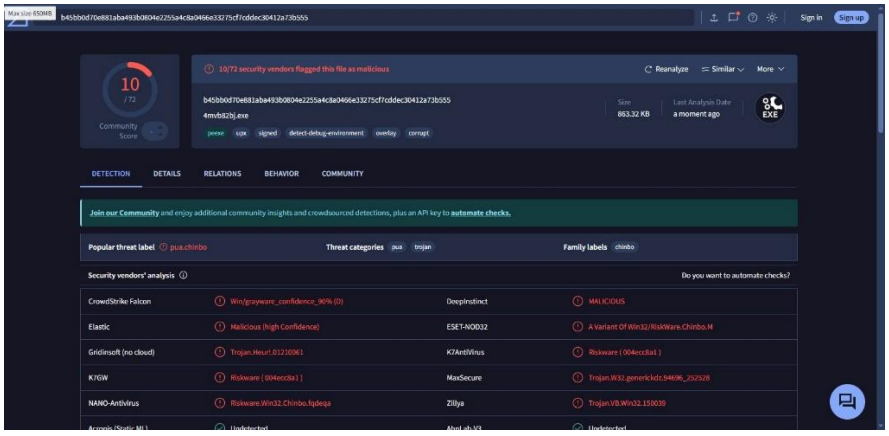




Step 5: Now scan the zip file on the website www.virustotal.com. Choose a file and scan it.



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.**

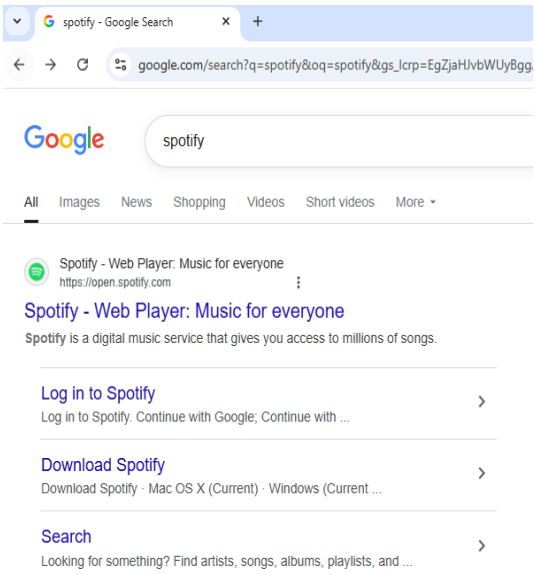
We would use firewall to block
1) A Port
2) A Program
3) A Website

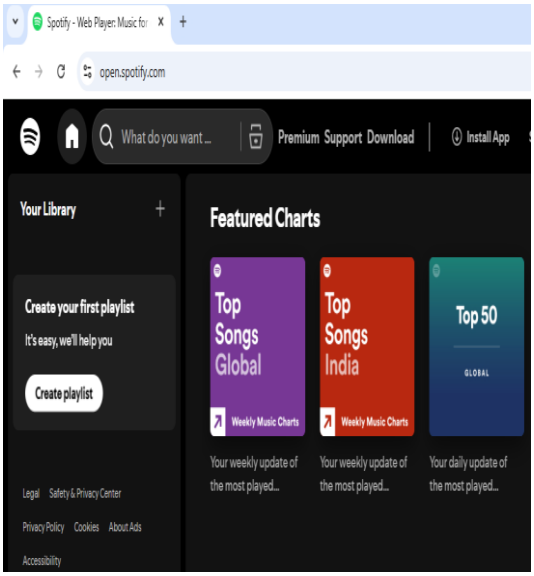
Part 1: Blocking the HTTP and HTTPS (Port 80 and Port 443) using the Firewall. Before starting with the blocking port process, we note that the applications running at the server end are identified with the well-known Port numbers, some of the commonly used are as follows.

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

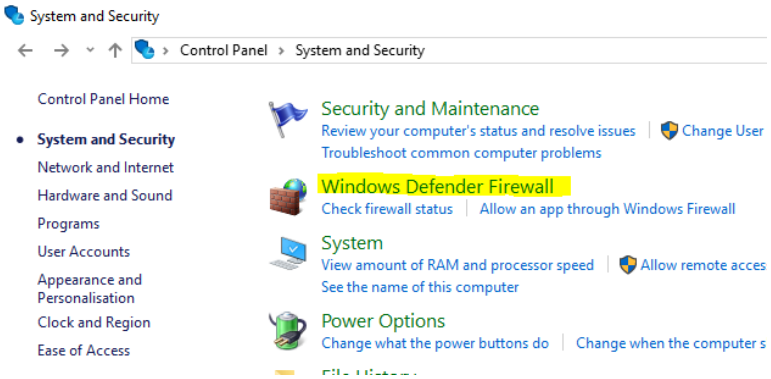
For Inbound-

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.

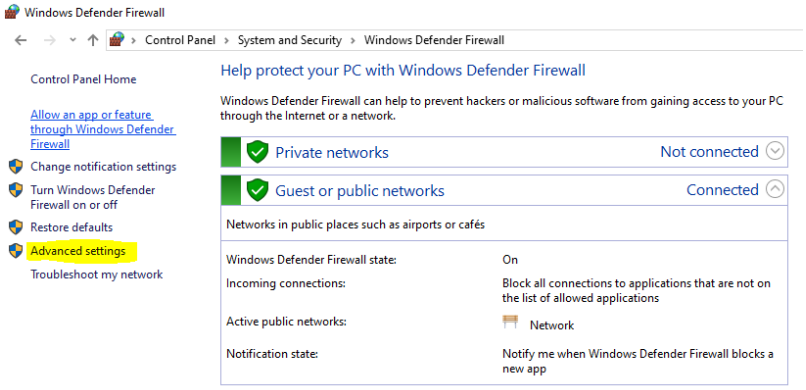




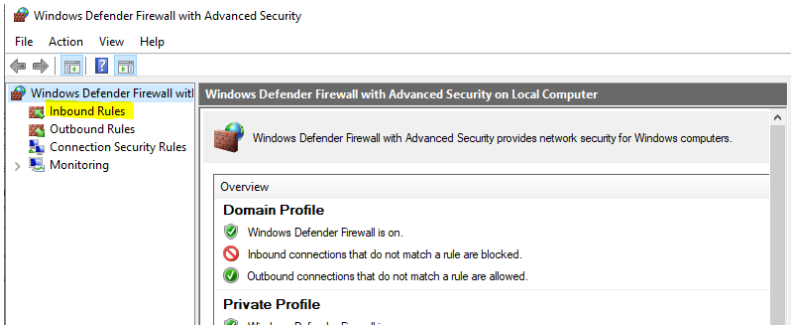
Step 2: We open ‘Windows Defender Firewall’



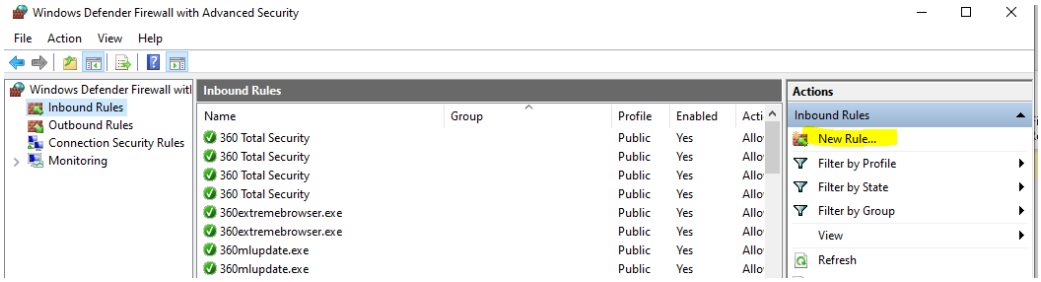
Step 3: Next, we click on ‘Advanced settings’



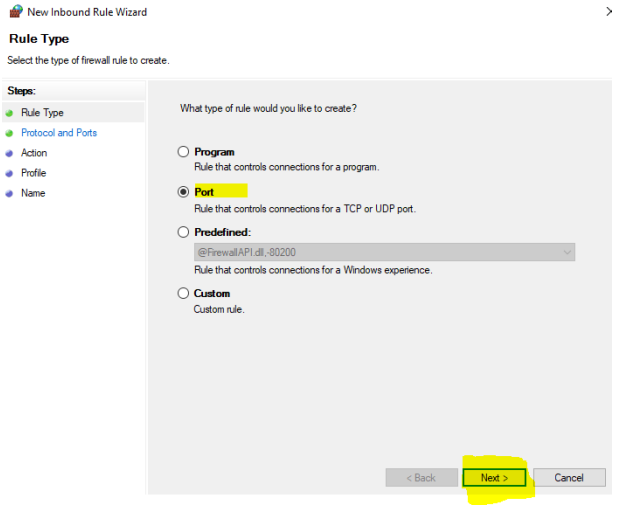
Step 4: Next go to ‘Inbound rules’



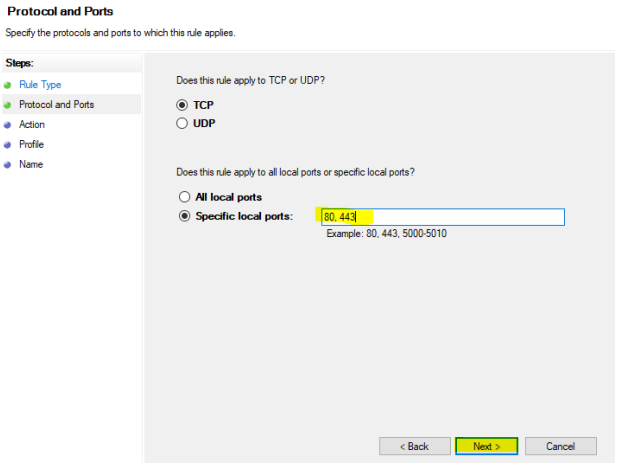
Step 5: Now create a ‘new rule’ for Inbound



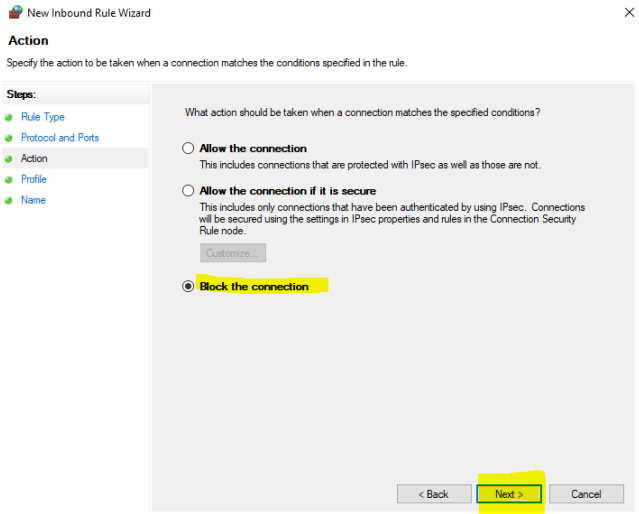
Step 6: Select ‘Port’ and click ‘Next’



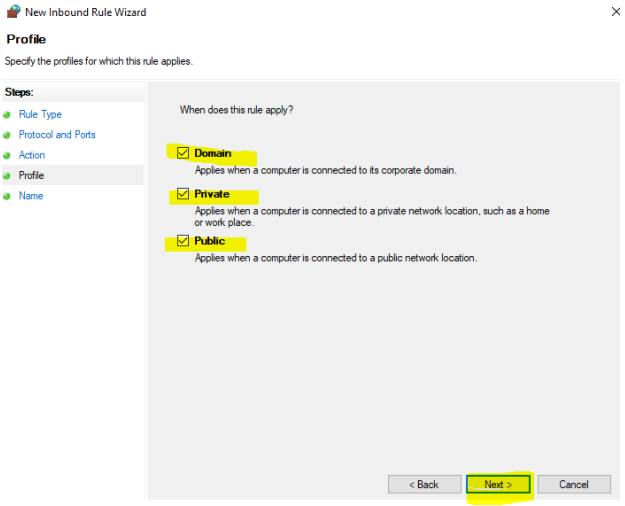
Step 7: Give Port ‘80, 443’ and click ‘Next’



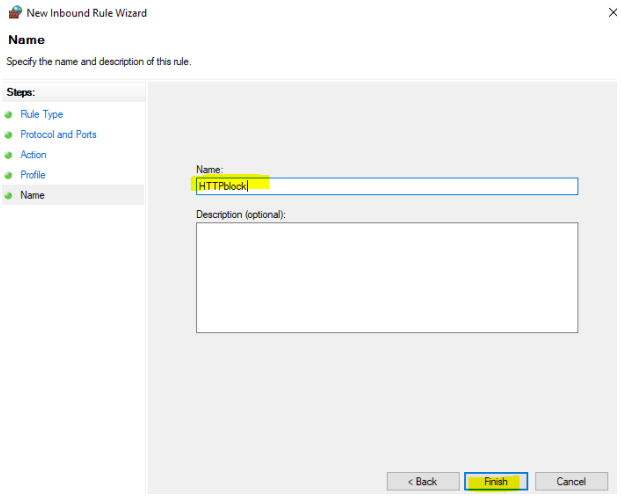
Step 8: Block the connection



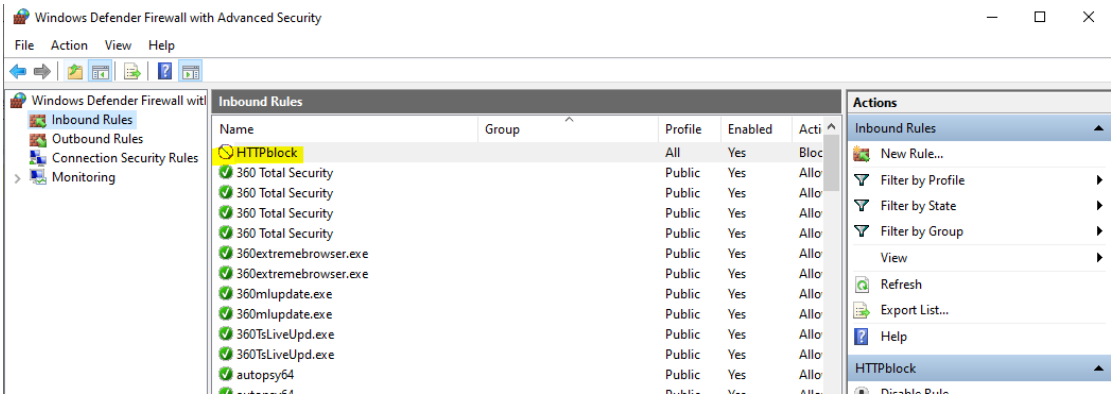
Step 9: Check all the rules application are enable



Step 10: Name the rule



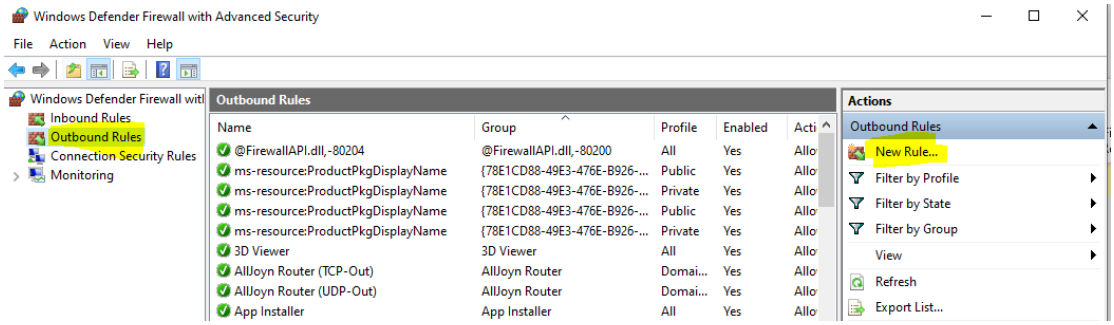
New rule for inbound is added.



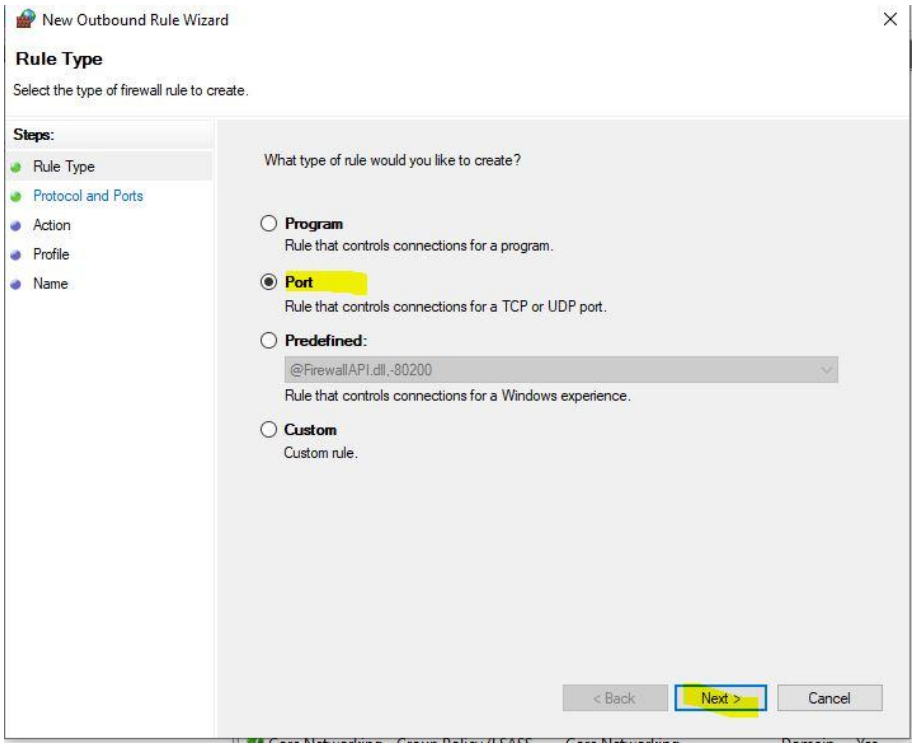
For outbound-

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

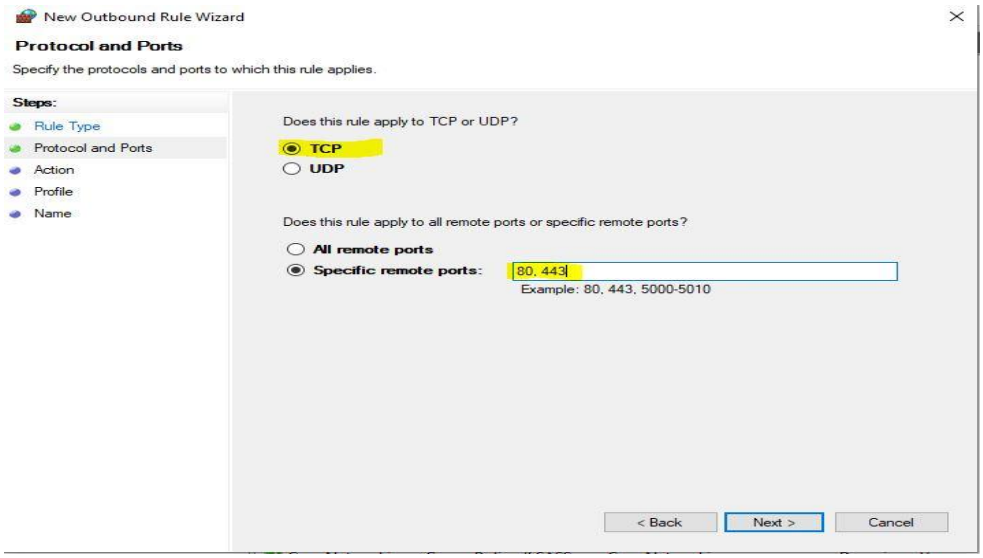
Step 1: Create a ‘new rule’ for Outbound



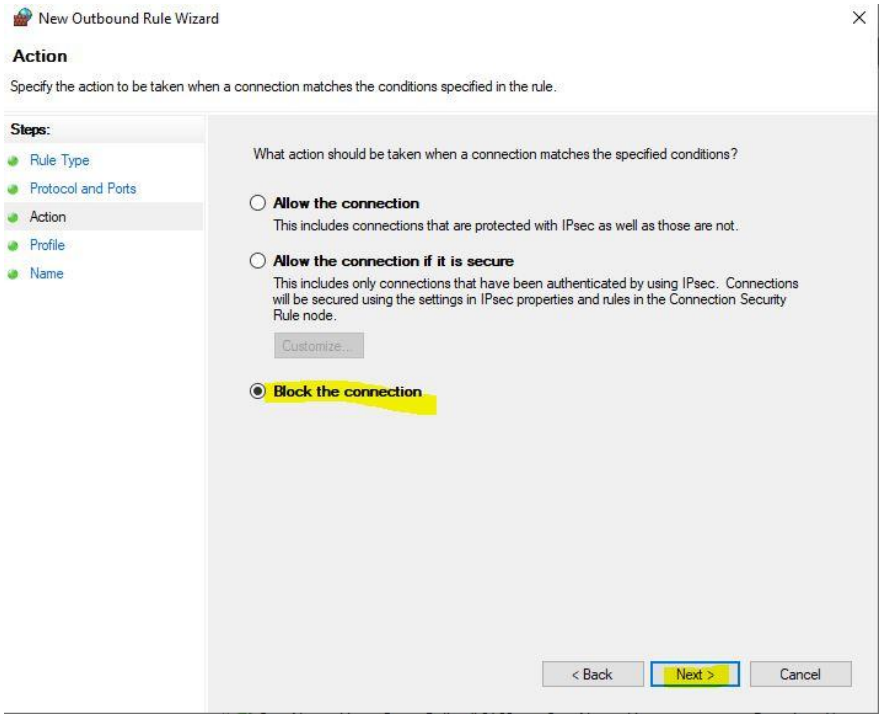
Step 2: Select ‘Port’ and click ‘Next’



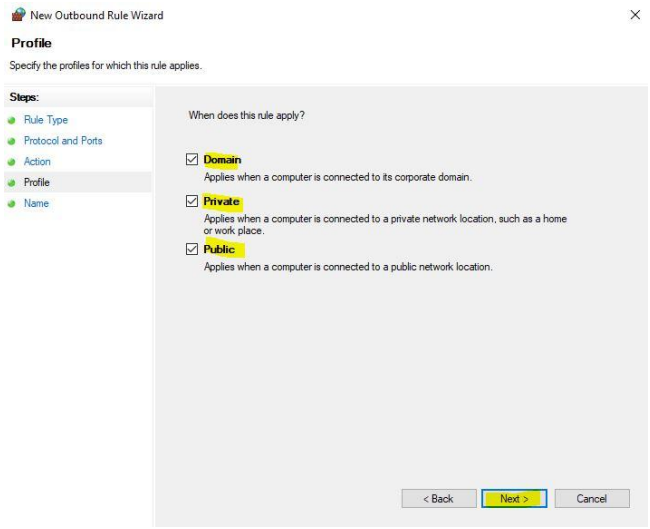
Step 3: Give Port ‘80, 443’ and click ‘Next’



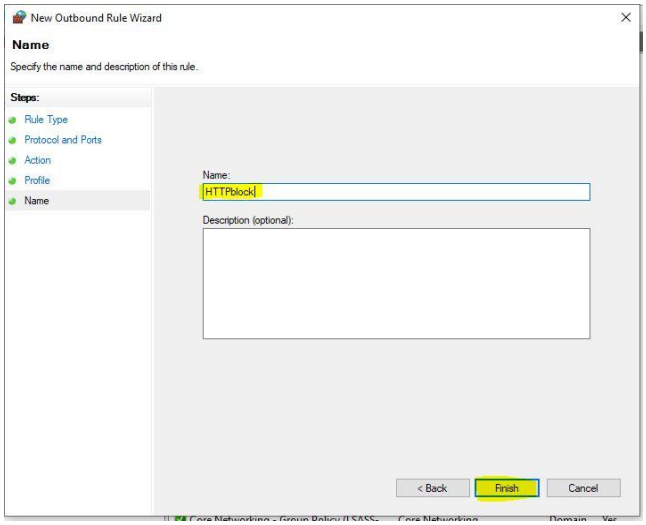
Step 4: Block the connection



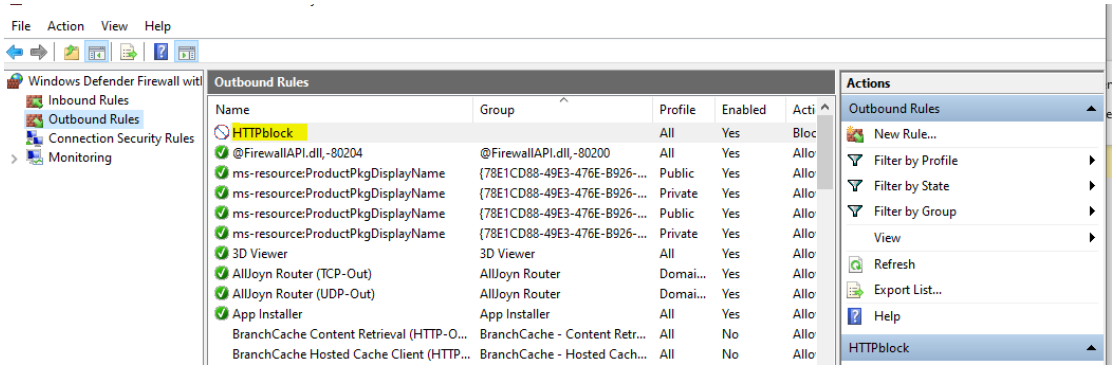
Step 5: Check all the rules application are enable



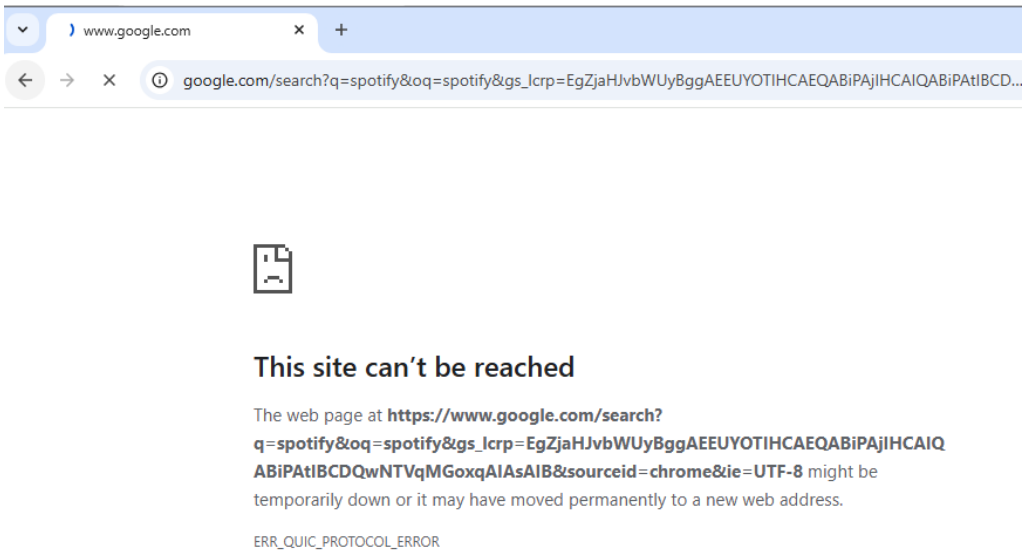
Step 6: Name the rule



New rule for outbound is added.

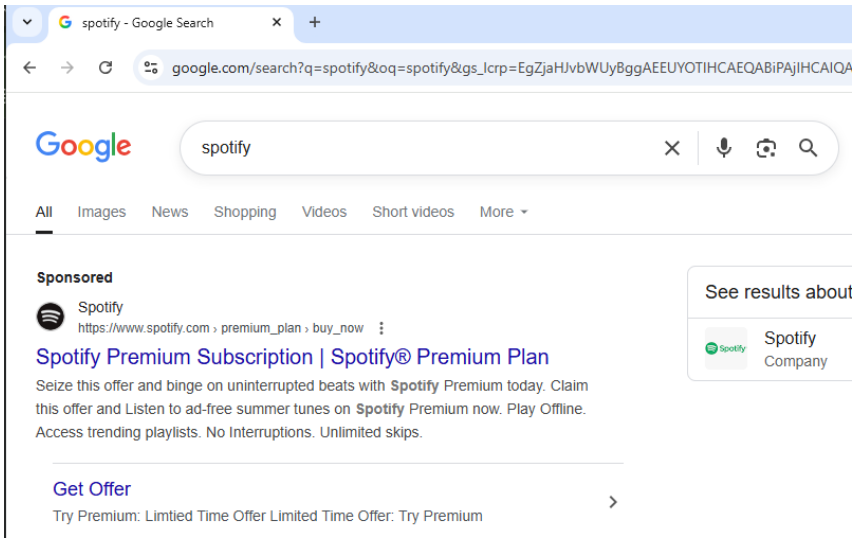


Now the access is blocked-



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.

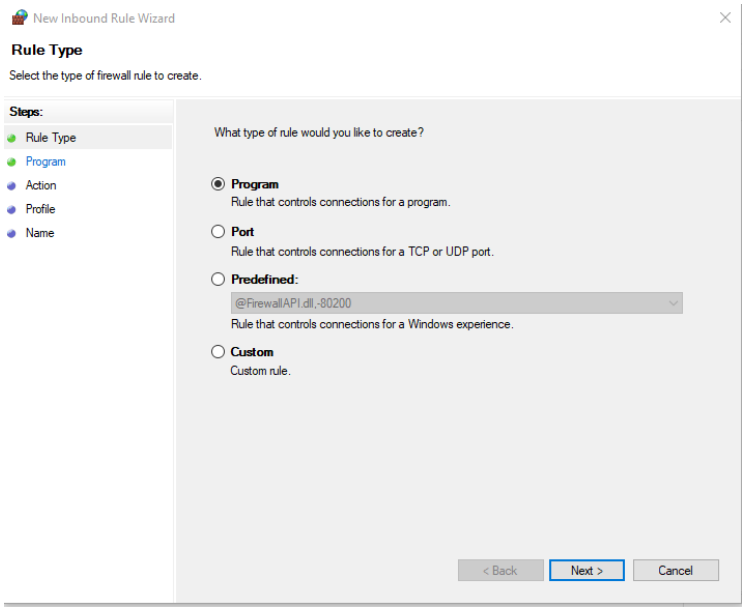
Step 1: Open Windows Defender Firewall.

Step 2: Go to Advance Settings

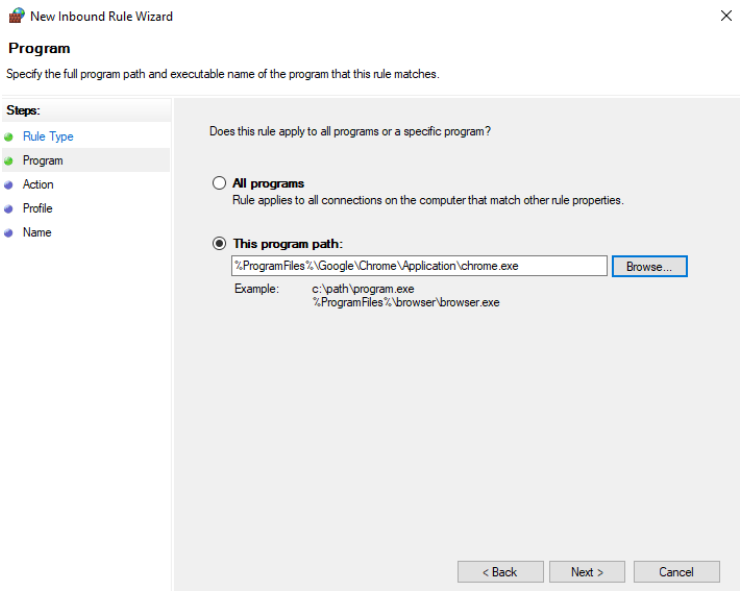
Step 3: Click on Inbound Rule

Step 4: Create new rule

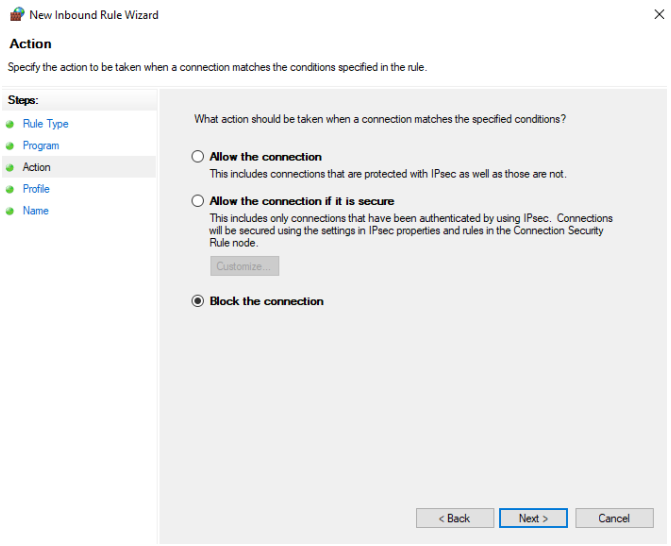
Step 5: Select Program and click on Next button



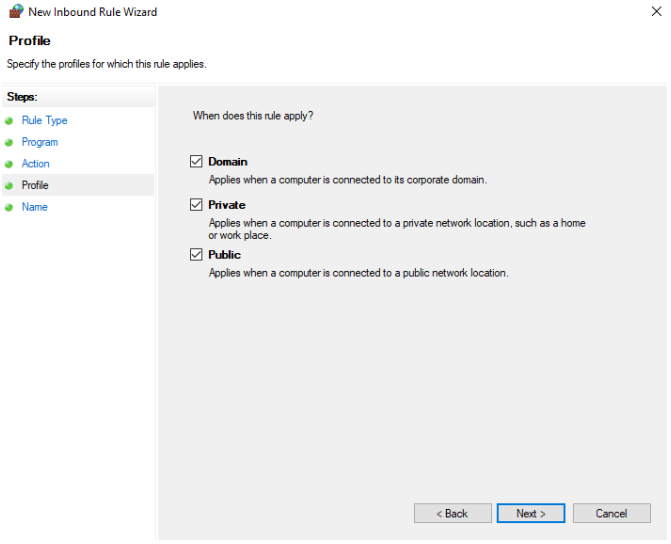
Step 6: We are blocking the chrome program so browse the path and click on next.



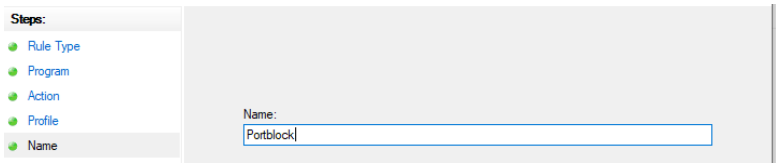
Step 7: Now click on block the connection.



Step 8: Tick all the checkboxes.

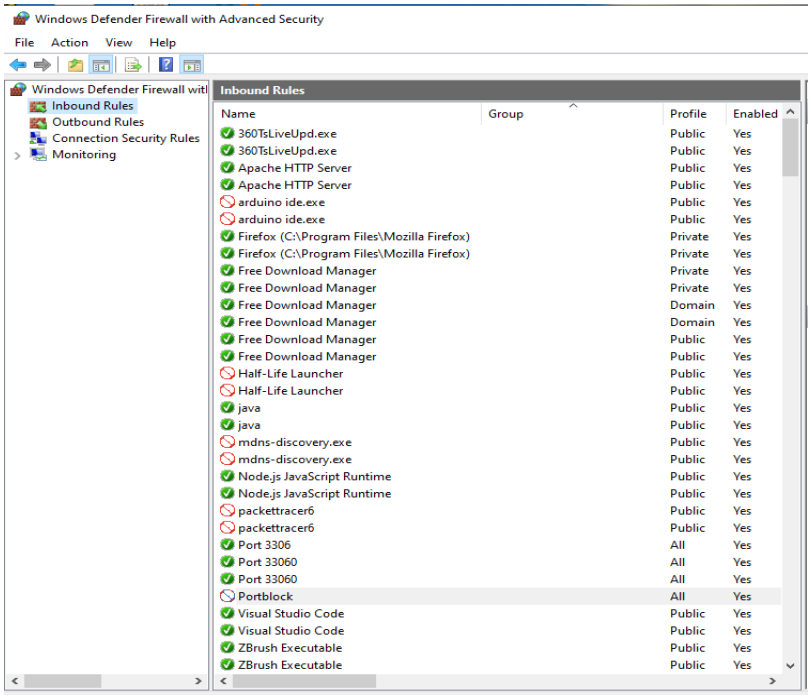


Step 9: After clicking on Next button give the name to that rule.

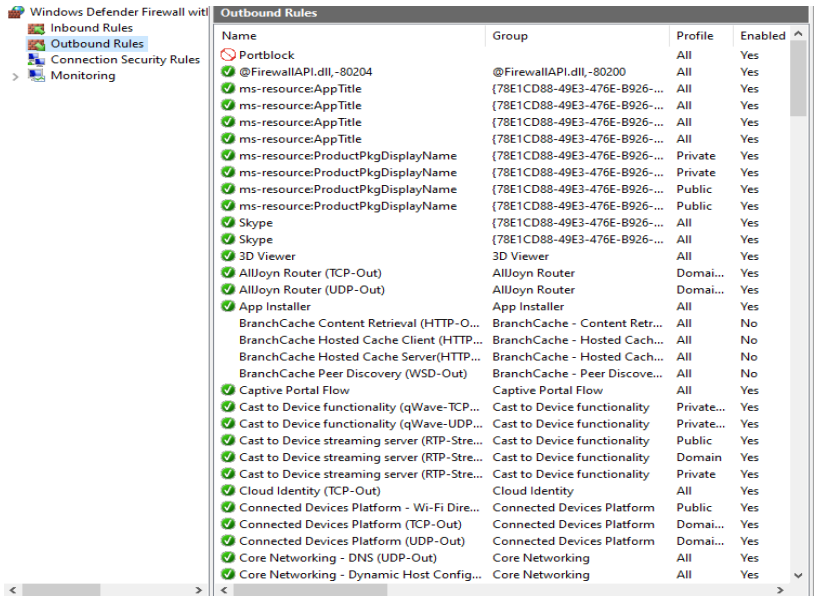


Then click on finish:

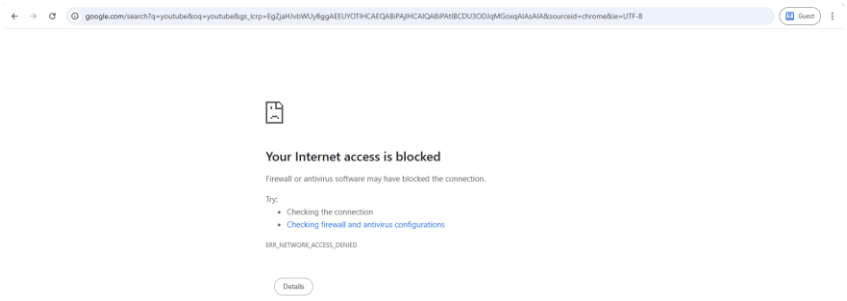
The Inbound rule is created:



Step 10: Follow the same steps for Outbound Rule.

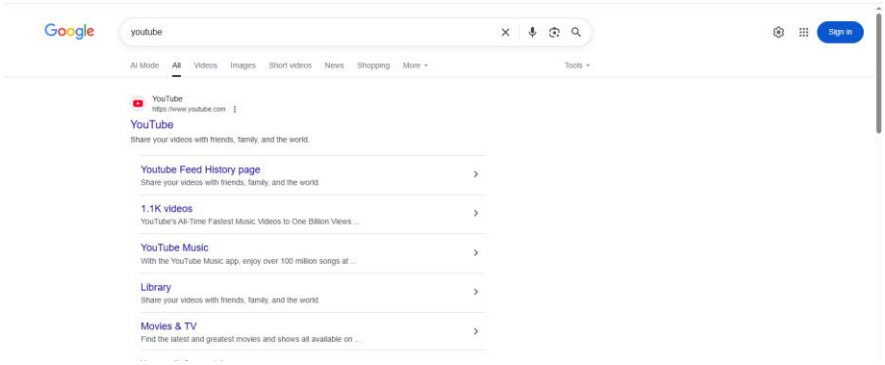


Step 11: Now open chrome and check it will show connect to internet

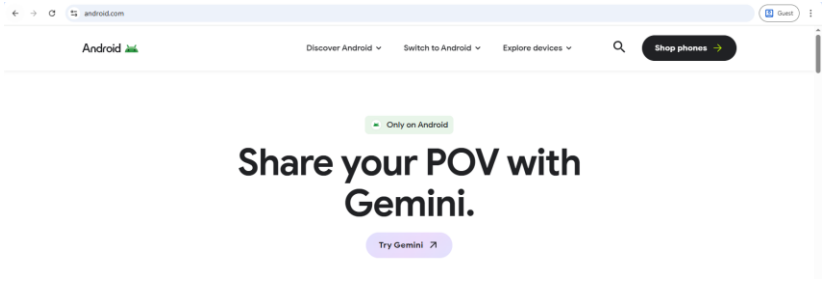


Now to unblock the Chrome:
Go to firewall > Go to inbound > Right click and delete the rule
Go to firewall > Go to outbound > Right click and delete the rule

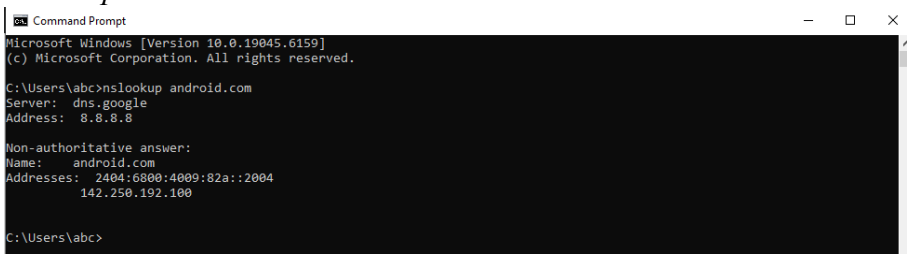
After deleting inbound and outbound rule-



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

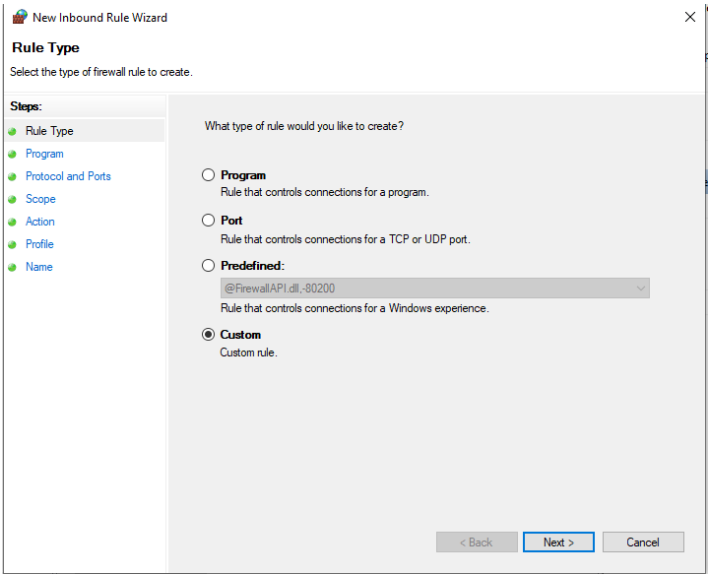
Step 1: Open Windows Defender Firewall.

Step 2: Go to Advance Settings

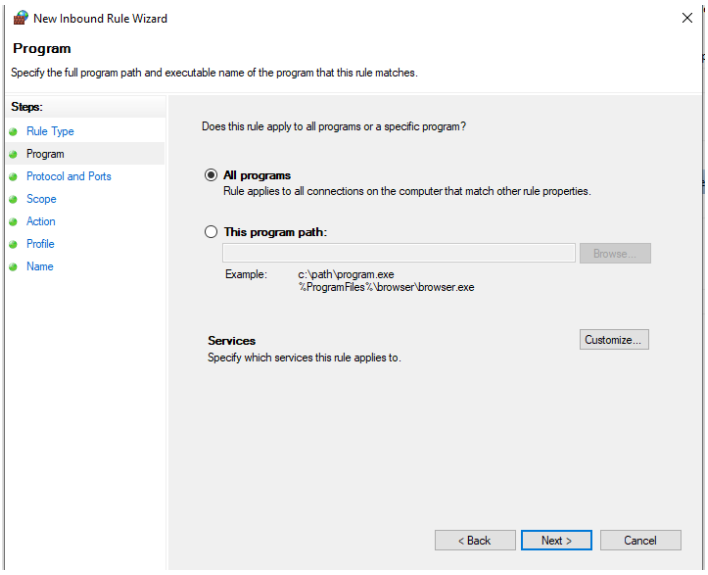
Step 3: Click on Inbound Rule

Step 4: Create new rule

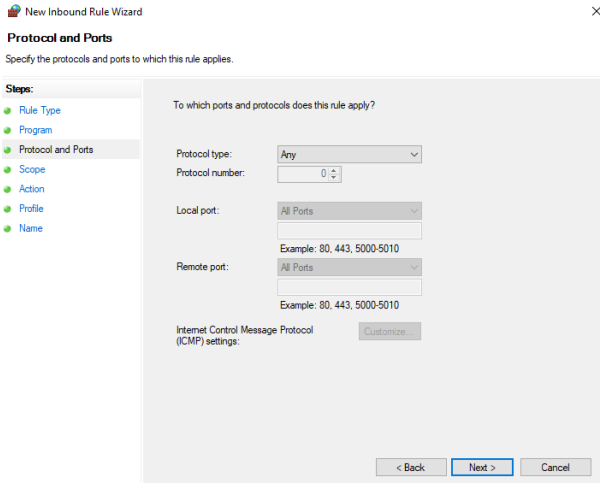
Step 5: Select Custom and click on Next button



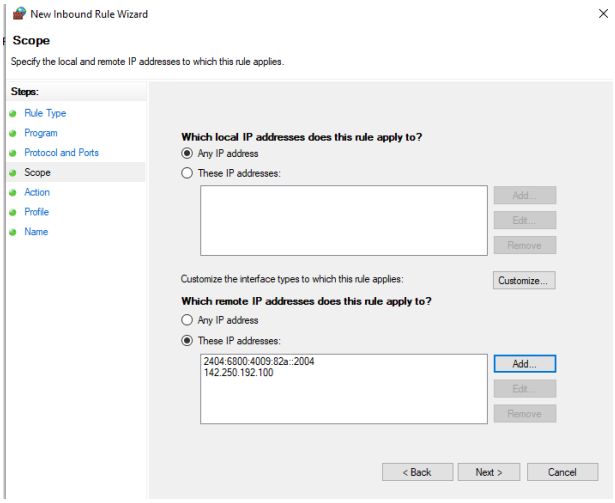
Step 6: Select “All programs” and click on Next button



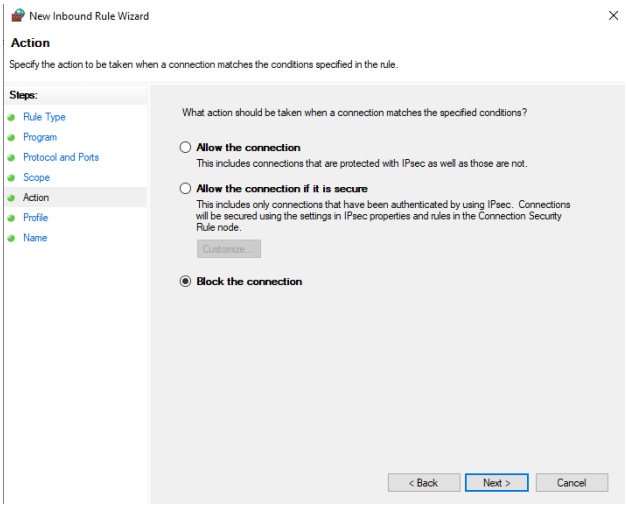
Step 7: Click on Next button



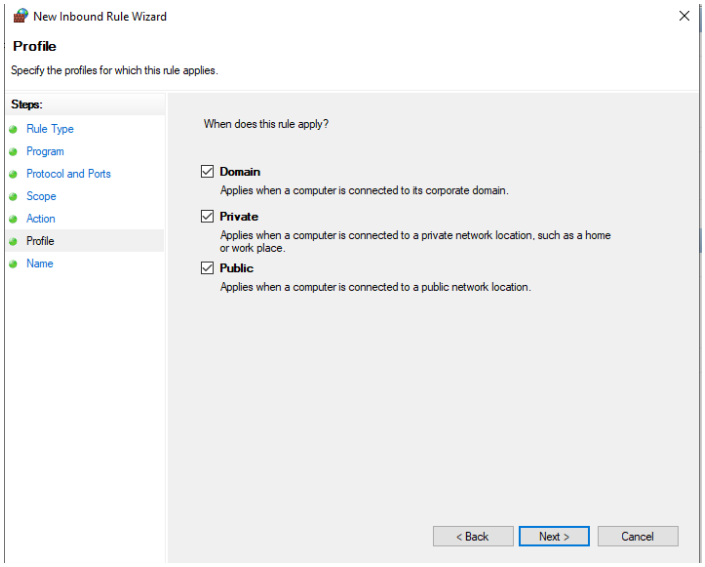
Step 8: Add both the addresses



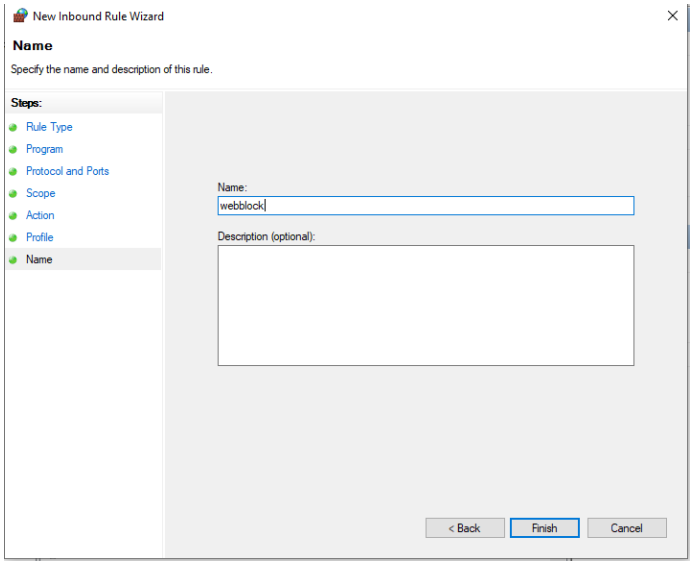
Step 9: Block the connection



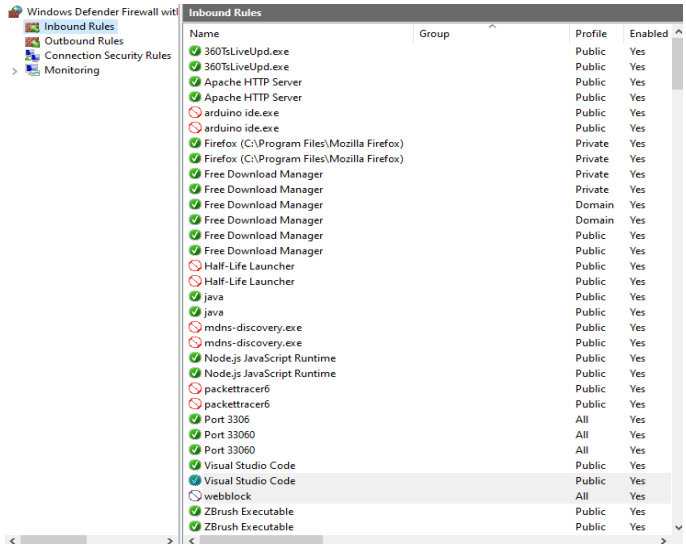
Step 10: Check all the boxes



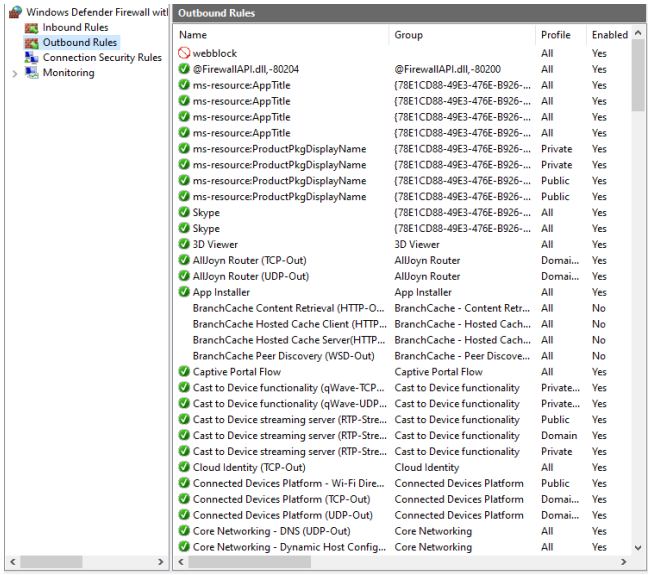
Step 11: Name the rule and click on finish.



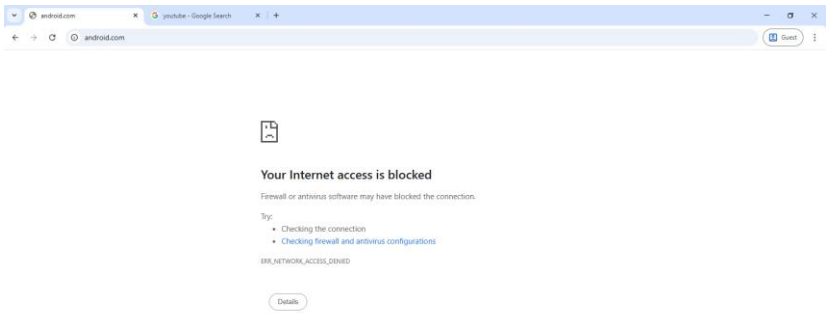
The Inbound Rule is created:



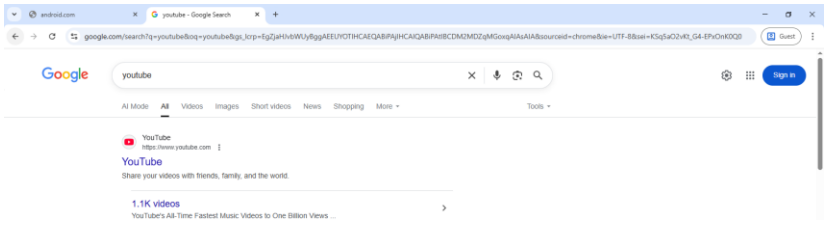
Now, follow the same steps for creating the outbound rule:



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-