

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 Decrvotion : 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_idx])
        cipher += ".join([row[curr_idx]
                           for row in matrix])

    k_idx += 1

    return cipher

def decryptMessage(cipher):
    msg = ""
    k_idx = 0
    msg_idx = 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_idx])
        for j in range(row):
            dec_cipher[j][curr_idx] = msg_lst[msg_idx]
            msg_idx += 1
        k_idx += 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'))
```

[illegible]

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\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:

```
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

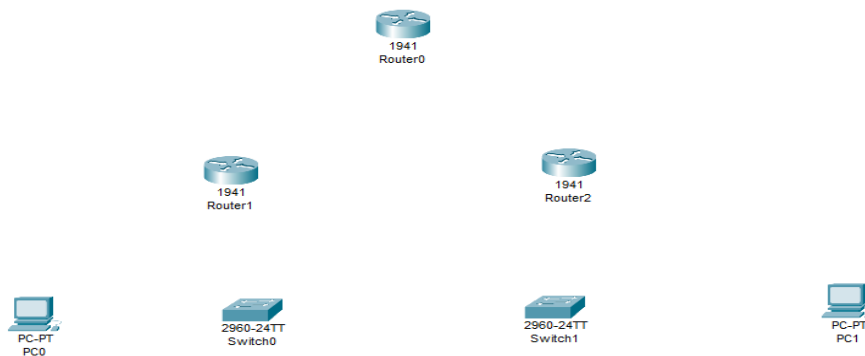
AIM: IP Security (IPsec) Configuration: Configure IPsec on network devices to provide secure communication and protect against unauthorized access and attacks.

Requirements:

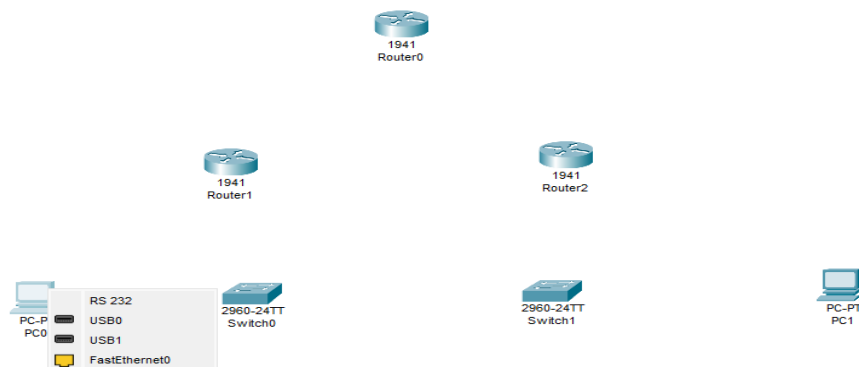
- Take 3 Routers (1941)
- Take 2 Switches (2960)
- Take 2 PC
- Configuration:
 - PC 0 – 192.168.1.2
 - PC 1 – 192.168.2.2
 - Router 1 (G 0/0) - 20.0.0.1
 - Router 1 (G 0/1) - 192.168.1.1
 - Router 0 (G 0/0) - 30.0.0.2
 - Router 0 (G 0/1) - 20.0.0.2
 - Router 2 (G 0/0) - 30.0.0.1
 - Router 2 (G 0/1) - 192.168.2.1

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

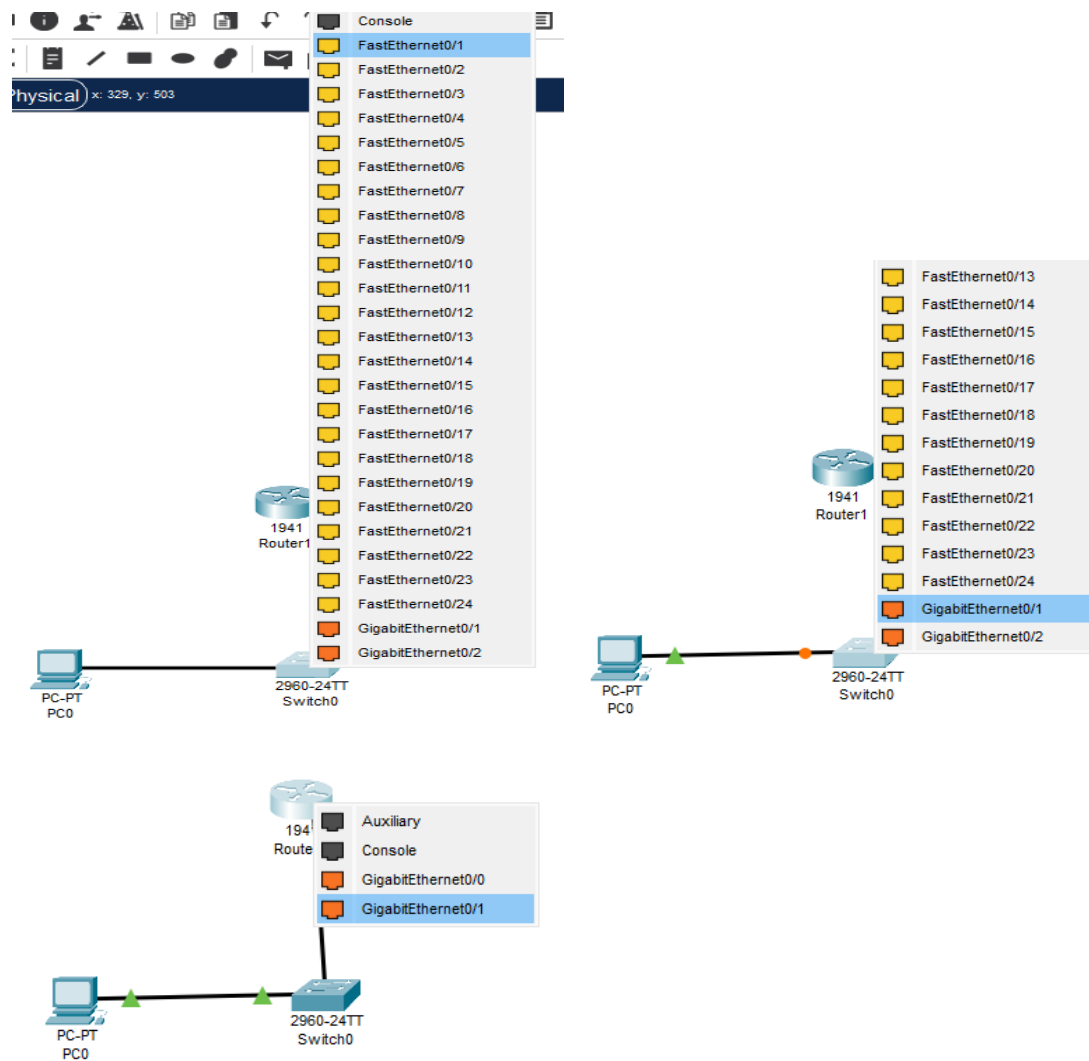
Connect the PC with Switches and also connect switches with Router using copper straight wire as given below.



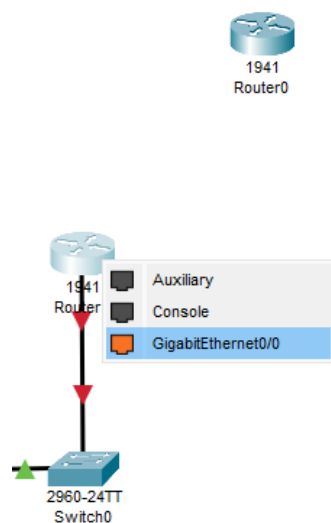
Connect with copper straight wire

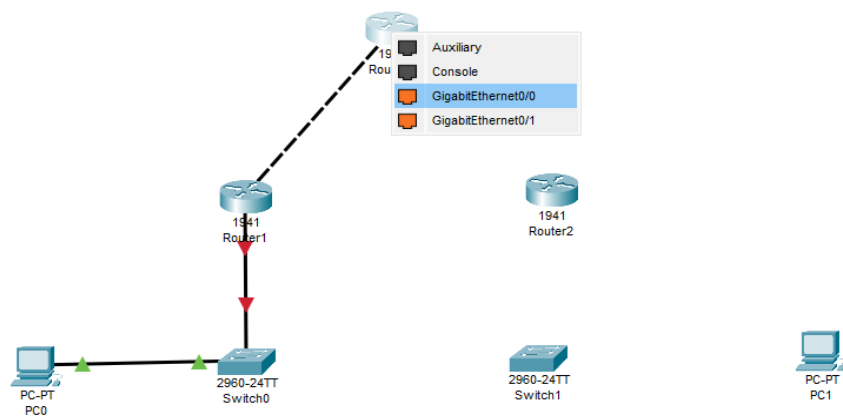


Connect the PC to the ethernet (FastEthernet0/24) and connect it with switch with GigabitEthernet0/1

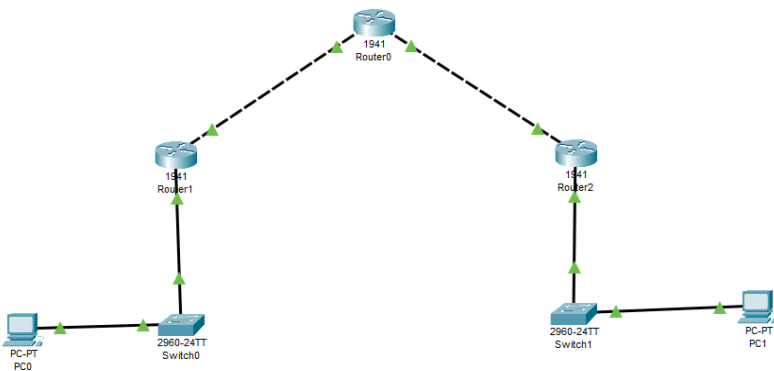


Connect the Router 1 with Router 0 and Router 0 with Router 2 using copper cross wire as given below.





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
```

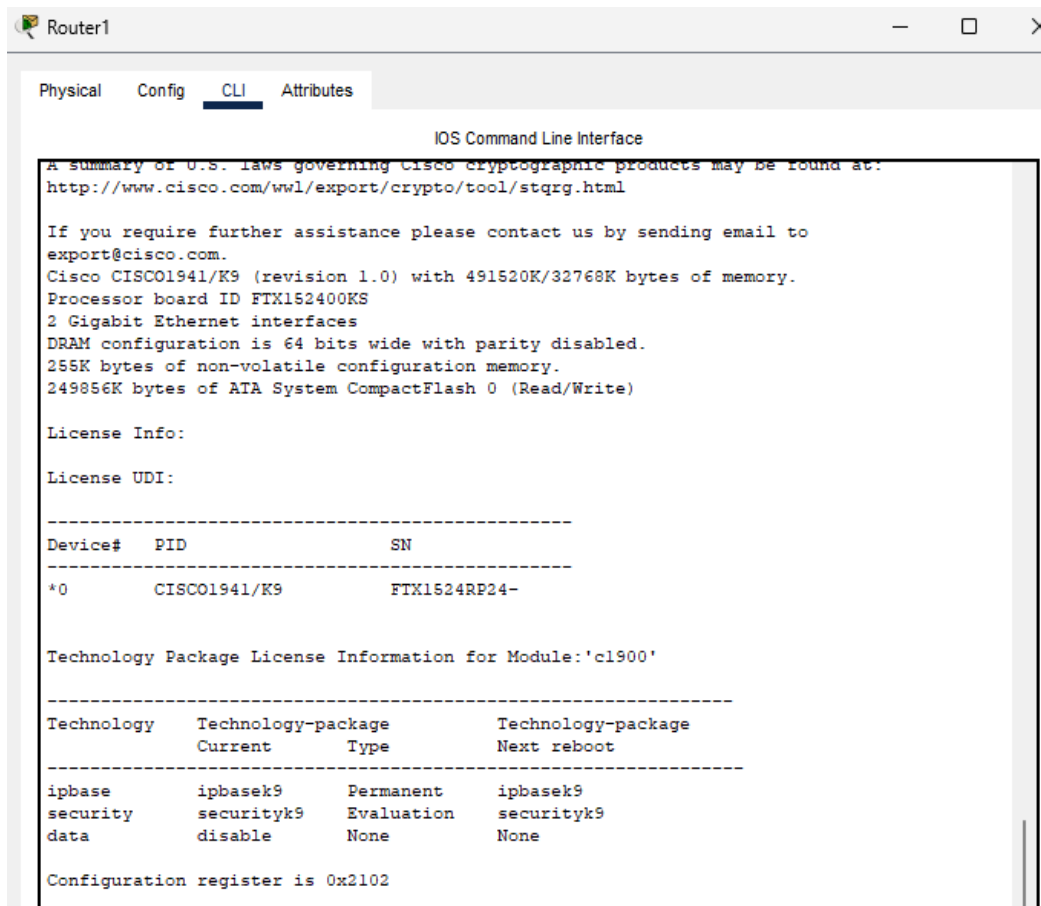
Router2

```
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-



The screenshot shows the CLI interface of Router1. The 'CLI' tab is selected. The output displays system information, license information, and technology package details.

IOS Command Line Interface

A summary of U.S. laws governing Cisco cryptographic products may be found at:
<http://www.cisco.com/wwl/export/crypto/tool/stqrg.html>

If you require further assistance please contact us by sending email to
export@cisco.com.

Cisco CISC01941/K9 (revision 1.0) with 491520K/32768K bytes of memory.
Processor board ID FTX152400KS
2 Gigabit Ethernet interfaces
DRAM configuration is 64 bits wide with parity disabled.
256K bytes of non-volatile configuration memory.
249856K bytes of ATA System CompactFlash 0 (Read/Write)

License Info:

License UDI:

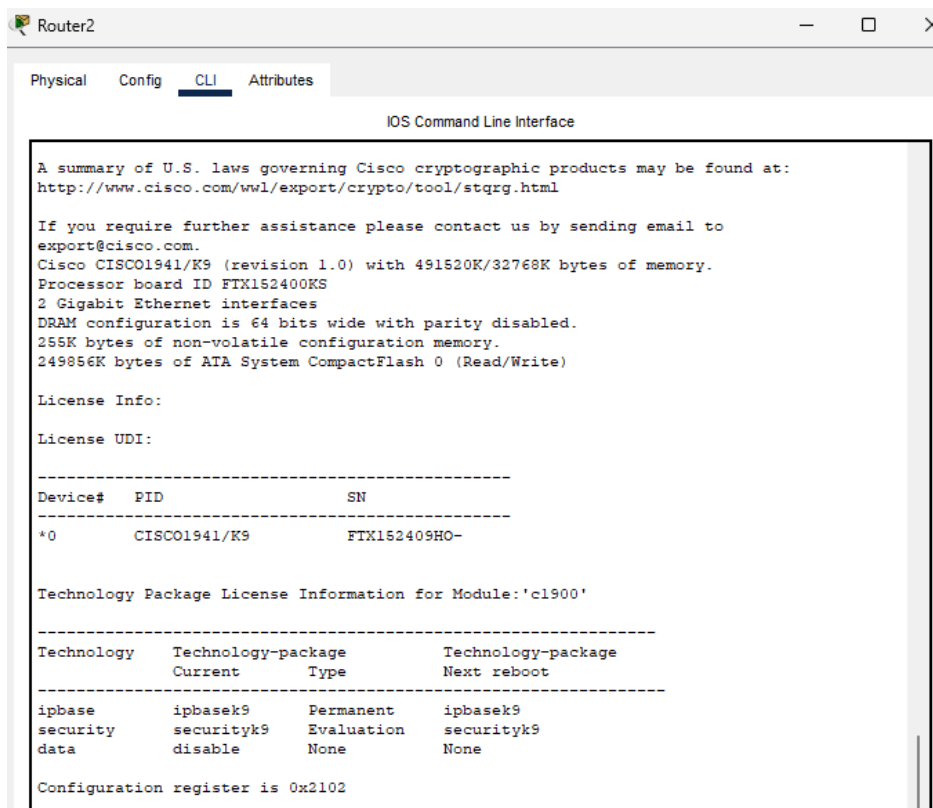
Device#	PID	SN
*0	CISC01941/K9	FTX1524RP24-

Technology Package License Information for Module: 'c1900'

Technology	Technology-package Current	Technology-package Type	Technology-package Next reboot
ipbase	ipbasek9	Permanent	ipbasek9
security	securityk9	Evaluation	securityk9
data	disable	None	None

Configuration register is 0x2102

Router2-



Router2

Physical Config **CLI** Attributes

IOS Command Line Interface

A summary of U.S. laws governing Cisco cryptographic products may be found at:
<http://www.cisco.com/wwl/export/crypto/tool/stqrg.html>

If you require further assistance please contact us by sending email to
export@cisco.com.

Cisco CISC01941/K9 (revision 1.0) with 491520K/32768K bytes of memory.
Processor board ID FTX152400KS
2 Gigabit Ethernet interfaces
DRAM configuration is 64 bits wide with parity disabled.
255K bytes of non-volatile configuration memory.
249856K bytes of ATA System CompactFlash 0 (Read/Write)

License Info:

License UDI:

Device#	PID	SN
*0	CISC01941/K9	FTX152409HO-

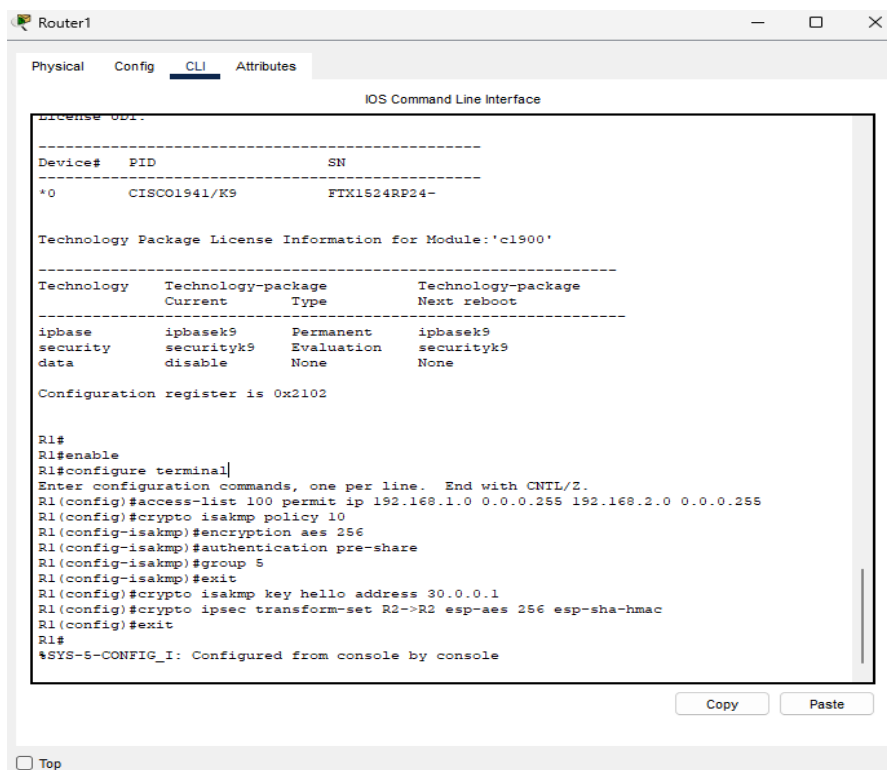
Technology Package License Information for Module:'c1900'

Technology	Technology-package Current	Technology-package Type	Technology-package Next reboot
ipbase	ipbasek9	Permanent	ipbasek9
security	securityk9	Evaluation	securityk9
data	disable	None	None

Configuration register is 0x2102

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

Router1-



Router1

Physical Config **CLI** Attributes

IOS Command Line Interface

License UDI:

Device#	PID	SN
*0	CISC01941/K9	FTX1524RP24-

Technology Package License Information for Module:'c1900'

Technology	Technology-package Current	Technology-package Type	Technology-package Next reboot
ipbase	ipbasek9	Permanent	ipbasek9
security	securityk9	Evaluation	securityk9
data	disable	None	None

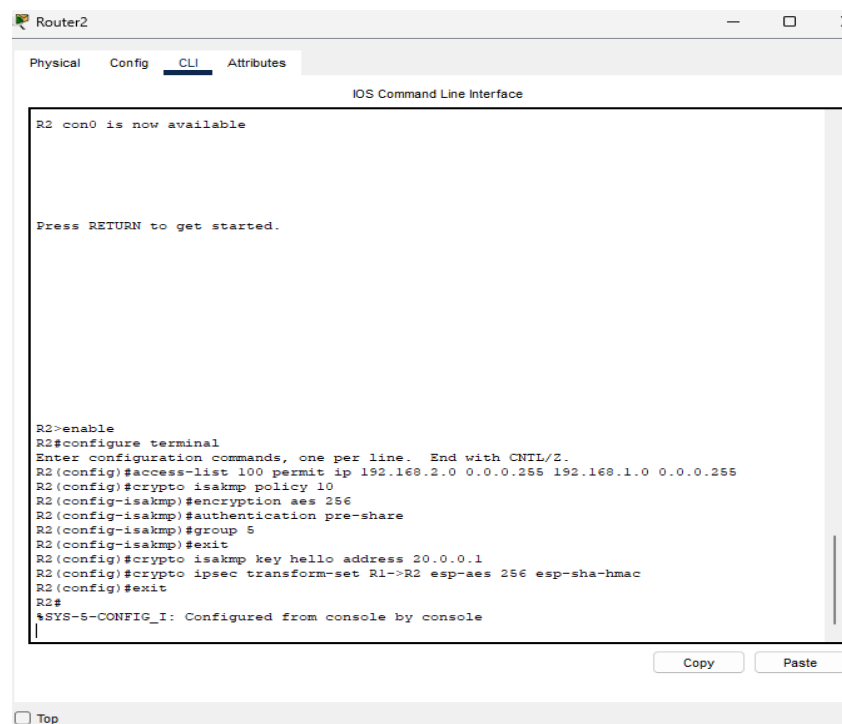
Configuration register is 0x2102

```
R1#
R1#enable
R1#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#access-list 100 permit ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
R1(config)#crypto isakmp policy 10
R1(config-isakmp)#encryption aes 256
R1(config-isakmp)#authentication pre-share
R1(config-isakmp)#group 5
R1(config-isakmp)#exit
R1(config)#crypto isakmp key hello address 30.0.0.1
R1(config)#crypto ipsec transform-set R2->R2 esp-aes 256 esp-sha-hmac
R1(config)#exit
R1#
%SYS-5-CONFIG_I: Configured from console by console
```

Copy Paste

☐ Top

Router2-

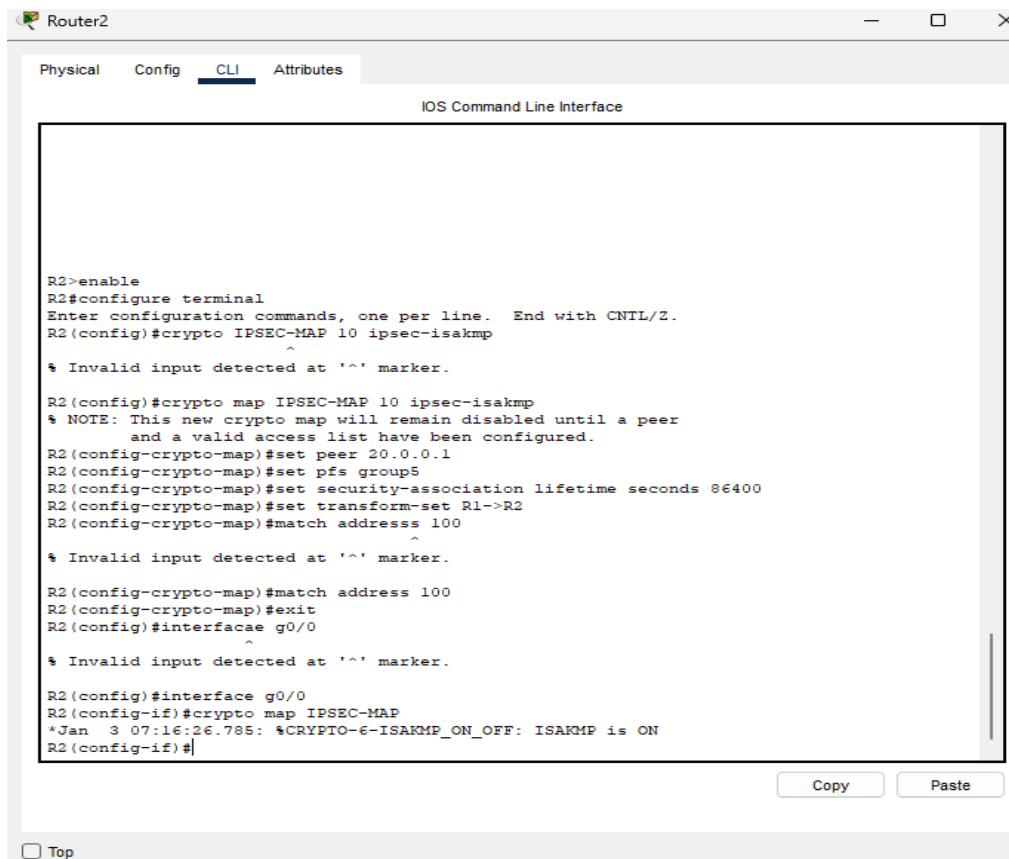


Step 5: Create a Crypto Map-

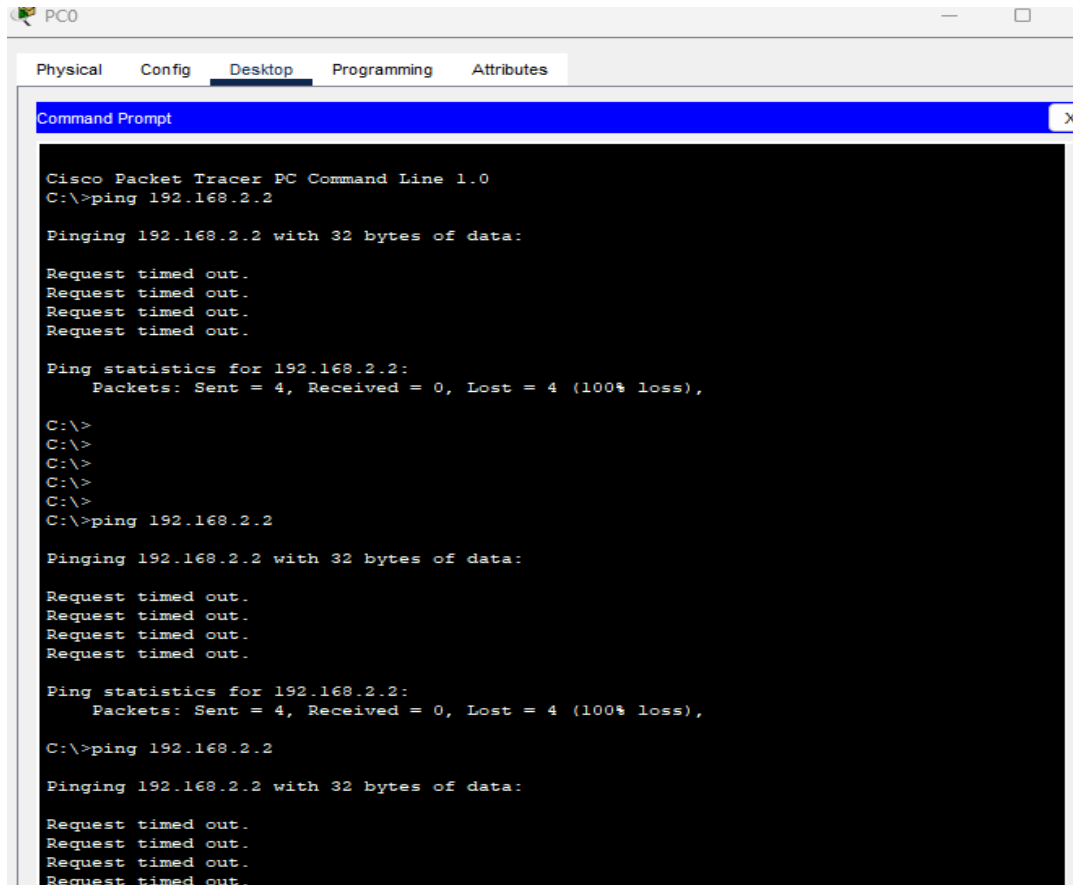
Router1-

```
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-

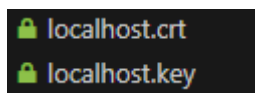


PRACTICAL 7

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:

[illegible]

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.

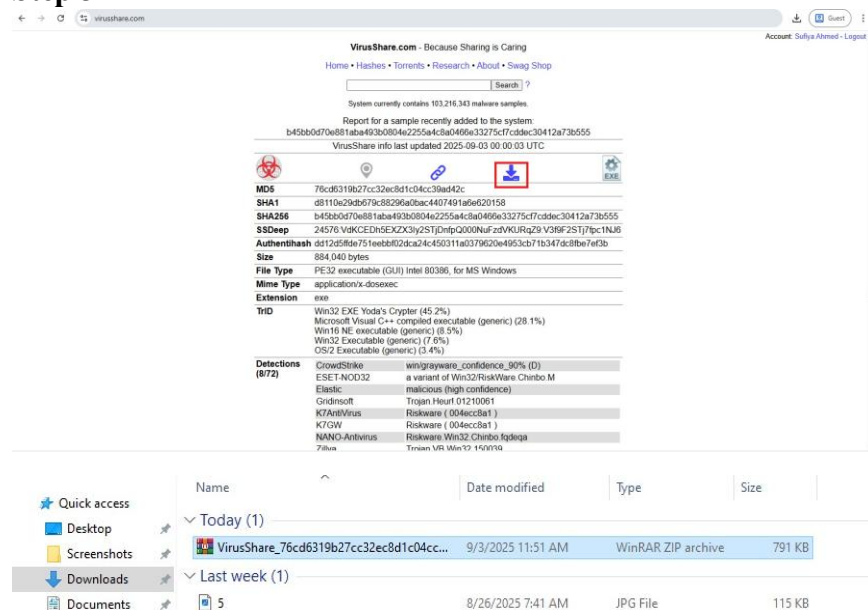
Step 1: Visit the website VirusShare.com for downloading the virus samples.

Step 2: Login:

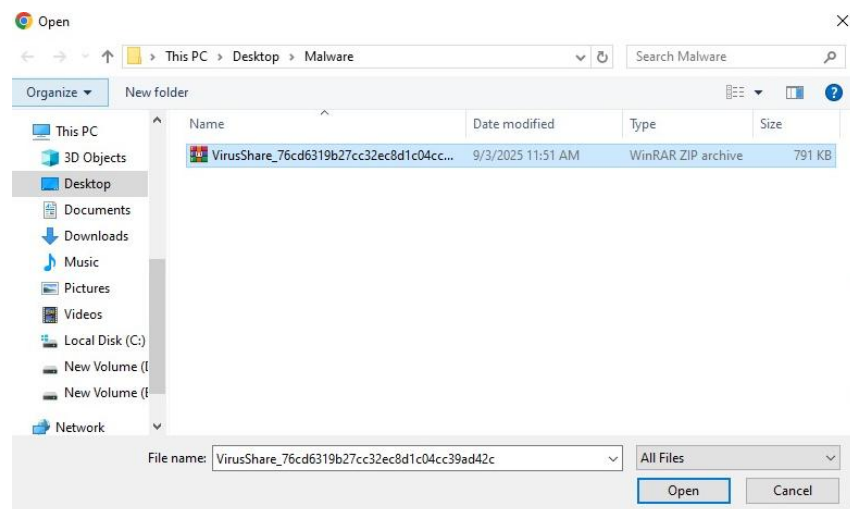
Username: Sufiya111

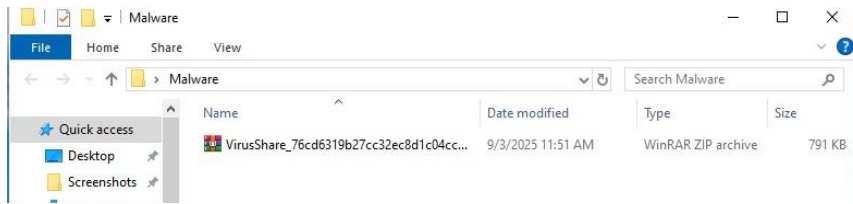
Password: INSPrac9

Step 3: Click on download.

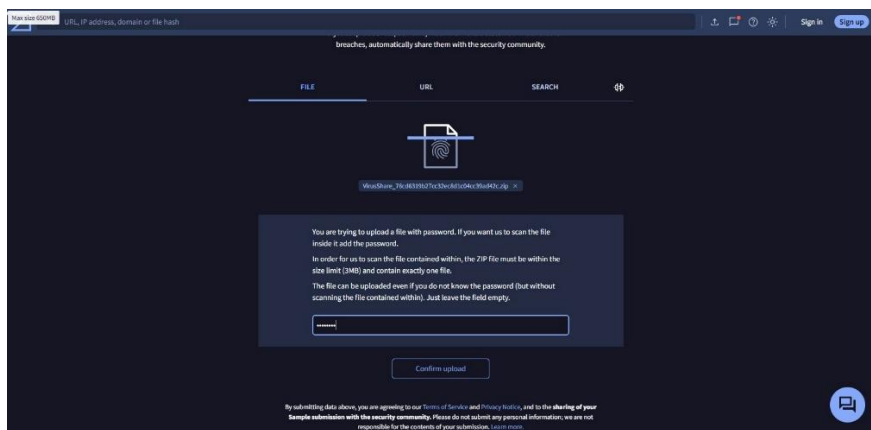


Step 4: Create a folder on the desktop name as Malware.

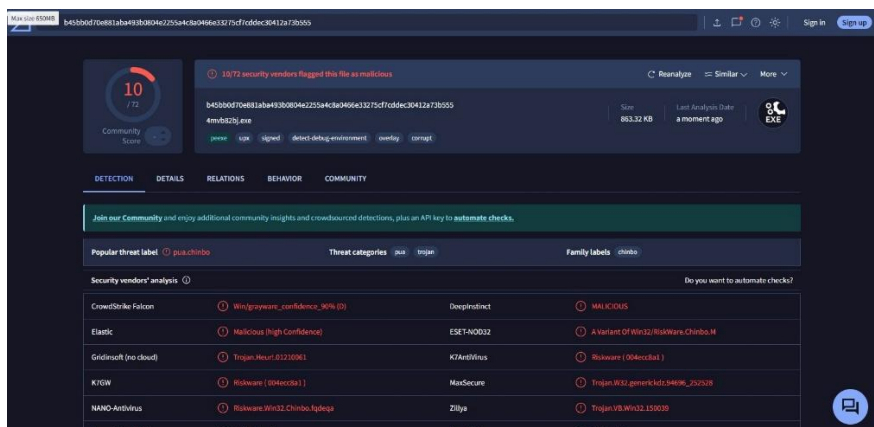




Step 5: Now scan the zip file on the website www.virustotal.com. Choose a file and scan it.



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

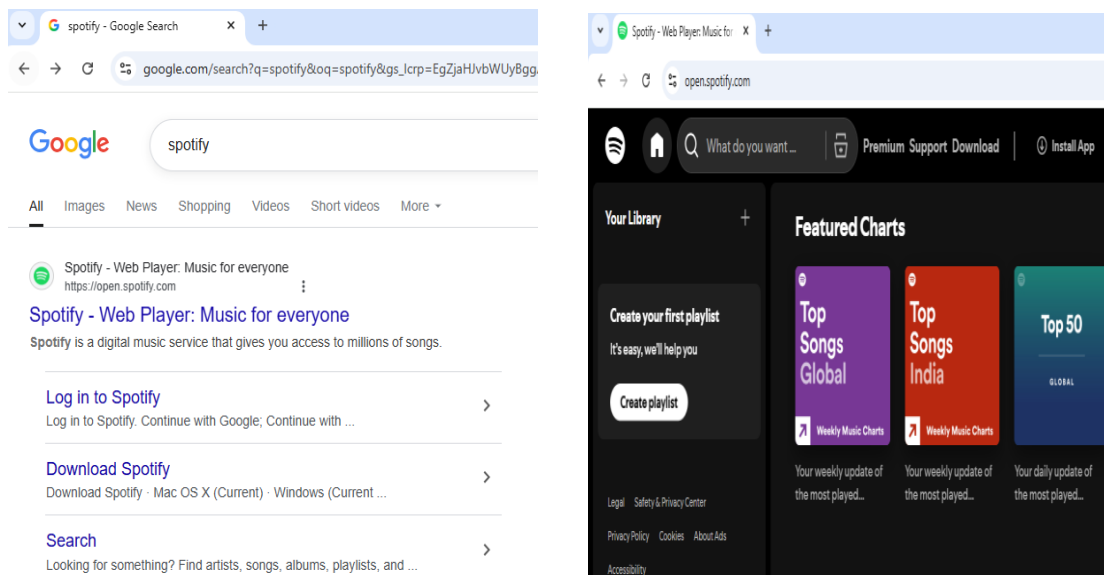
- 1) A Port
- 2) A Program
- 3) A Website

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.

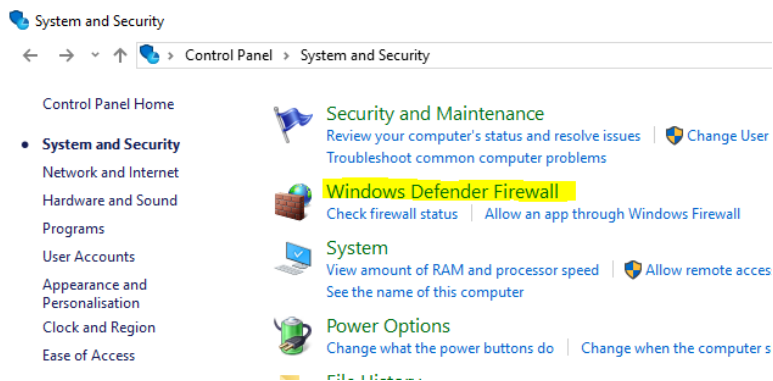
Port Number	Protocol	Application
20	TCP	FTP data
21	TCP	FTP control
22	TCP	SSH
25	TCP	SMTP
53	UDP, TCP	DNS
80	TCP	HTTP (W/W)
110	TCP	POP3
443	TCP	SSL

For Inbound-

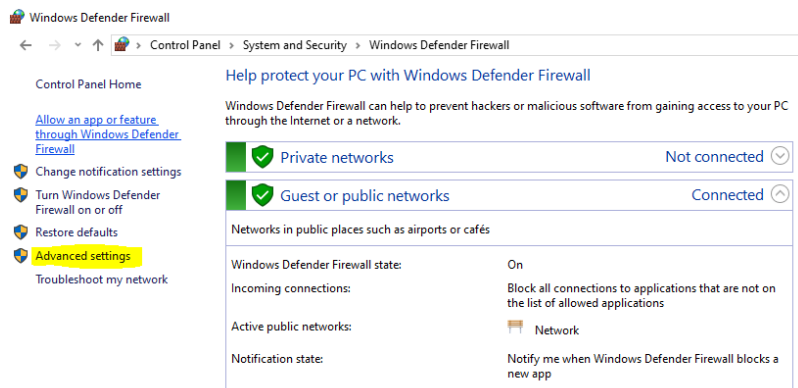
Step 1: We access any website through the browser and confirm that the HTTP/HTTPS protocols are working. Go to Browser > Search Anything > Open > Working.



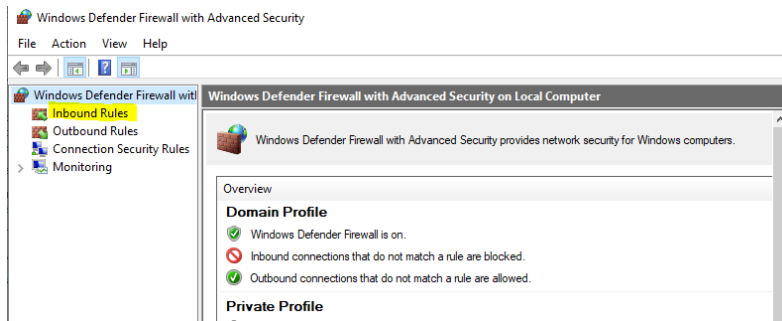
Step 2: We open 'Windows Defender Firewall'



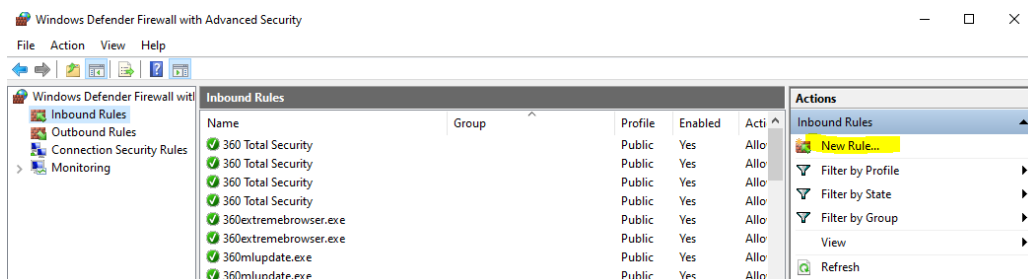
Step 3: Next, we click on ‘Advanced settings’



Step 4: Next go to ‘Inbound rules’



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



Step 6: Select 'Port' and click 'Next'

New Inbound Rule Wizard

Rule Type

Select the type of firewall rule to create.

Steps:

- Rule Type
- Protocol and Ports
- Action
- Profile
- Name

What type of rule would you like to create?

☐ **Program**
Rule that controls connections for a program.

☒ **Port**
Rule that controls connections for a TCP or UDP port.

☐ **Predefined:**
@FirewallAPI.dll,80200
Rule that controls connections for a Windows experience.

☐ **Custom**
Custom rule.

< Back **Next >** Cancel

Step 7: Give Port '80, 443' and click 'Next'

Protocol and Ports

Specify the protocols and ports to which this rule applies.

Steps:

- Rule Type
- Protocol and Ports
- Action
- Profile
- Name

Does this rule apply to TCP or UDP?

☒ **TCP**

☐ **UDP**

Does this rule apply to all local ports or specific local ports?

☐ **All local ports**

☒ **Specific local ports:**
Example: 80, 443, 5000-5010

< Back **Next >** Cancel

Step 8: Block the connection

New Inbound Rule Wizard

Action

Specify the action to be taken when a connection matches the conditions specified in the rule.

Steps:

- Rule Type
- Protocol and Ports
- Action
- Profile
- Name

What action should be taken when a connection matches the specified conditions?

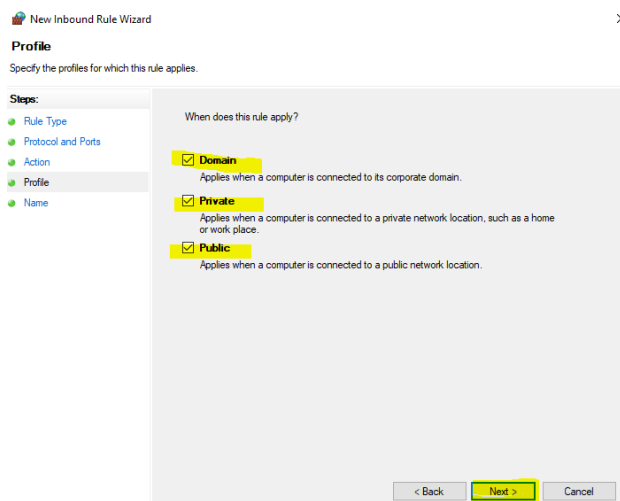
☐ **Allow the connection**
This includes connections that are protected with IPsec as well as those are not.

☐ **Allow the connection if it is secure**
This includes only connections that have been authenticated by using IPsec. Connections will be secured using the settings in IPsec properties and rules in the Connection Security Rule node.
[Customize](#)

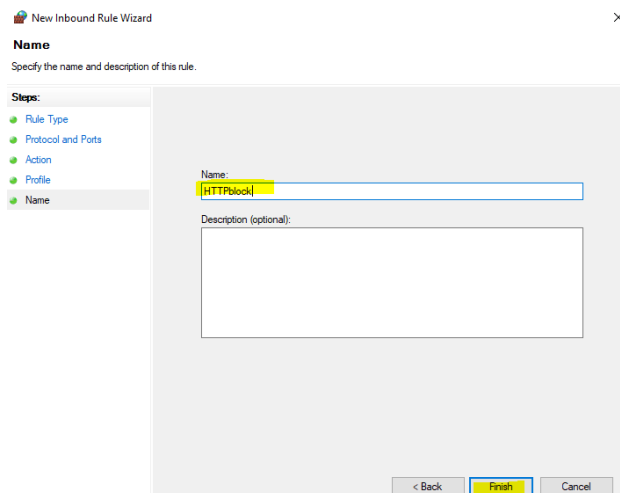
☒ **Block the connection**

< Back **Next >** Cancel

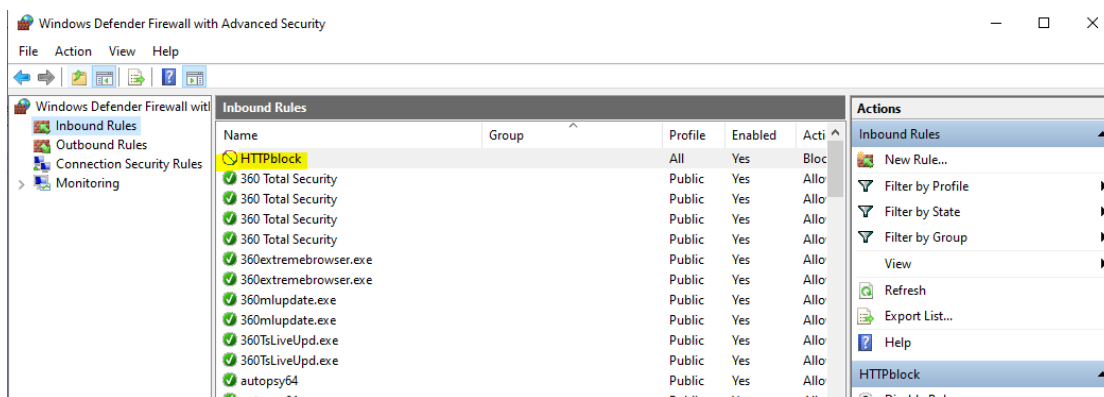
Step 9: Check all the rules application are enable



Step 10: Name the rule



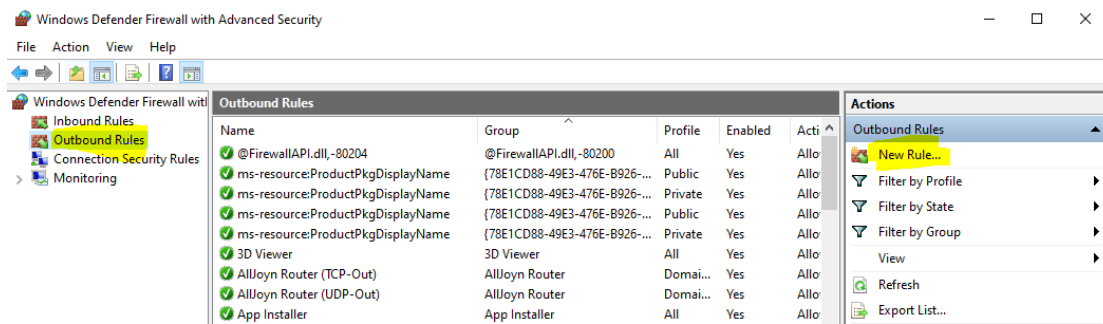
New rule for inbound is added.



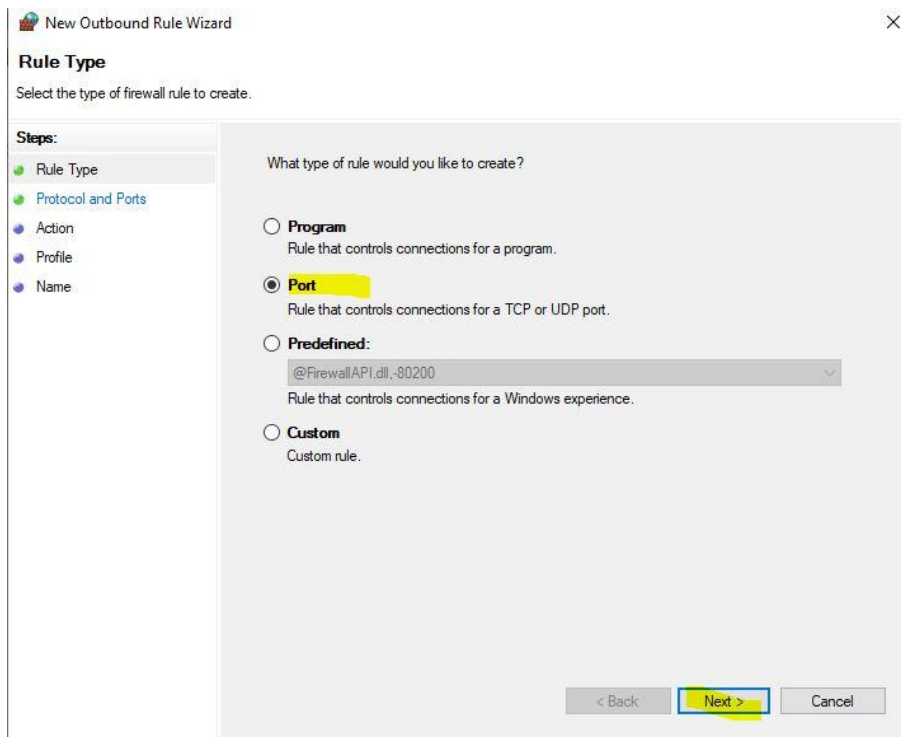
For outbound-

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

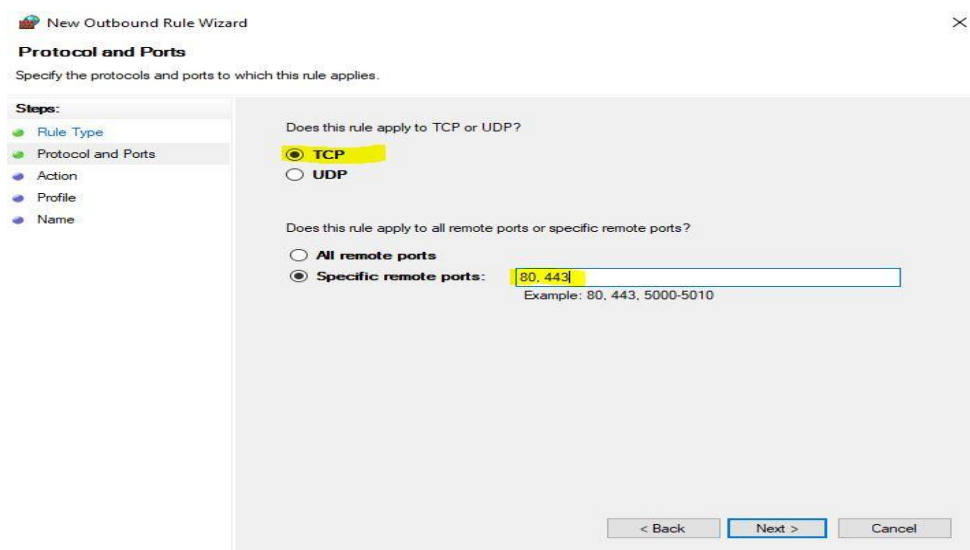
Step 1: Create a 'new rule' for Outbound



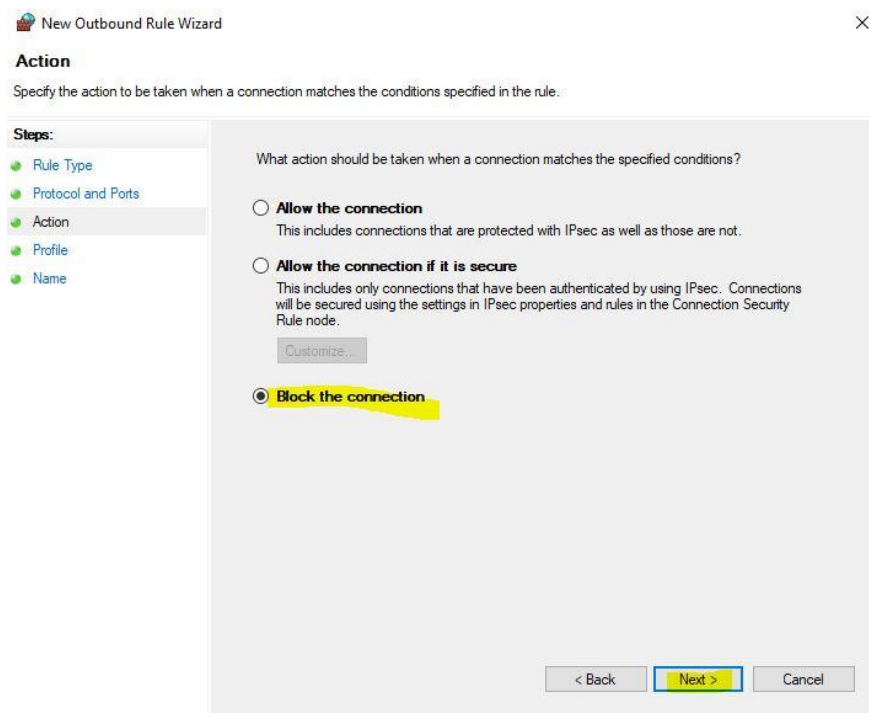
Step 2: Select 'Port' and click 'Next'



Step 3: Give Port '80, 443' and click 'Next'



Step 4: Block the connection



New Outbound Rule Wizard

Action

Specify the action to be taken when a connection matches the conditions specified in the rule.

Steps:

- Rule Type
- Protocol and Ports
- Action
- Profile
- Name

What action should be taken when a connection matches the specified conditions?

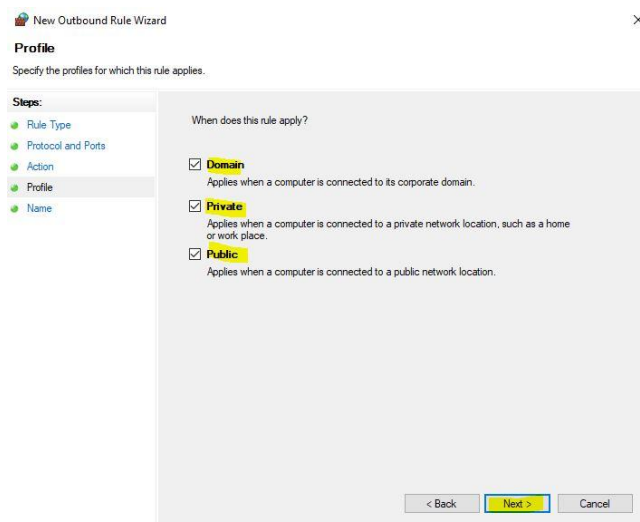
☐ **Allow the connection**
This includes connections that are protected with IPsec as well as those are not.

☐ **Allow the connection if it is secure**
This includes only connections that have been authenticated by using IPsec. Connections will be secured using the settings in IPsec properties and rules in the Connection Security Rule node.
[Customize...](#)

☒ **Block the connection**

< Back **Next >** Cancel

Step 5: Check all the rules application are enable



New Outbound Rule Wizard

Profile

Specify the profiles for which this rule applies.

Steps:

- Rule Type
- Protocol and Ports
- Action
- Profile
- Name

When does this rule apply?

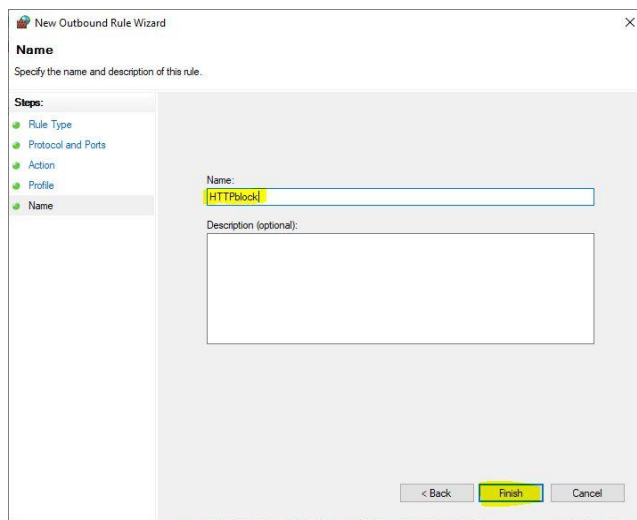
☒ **Domain**
Applies when a computer is connected to its corporate domain.

☒ **Private**
Applies when a computer is connected to a private network location, such as a home or work place.

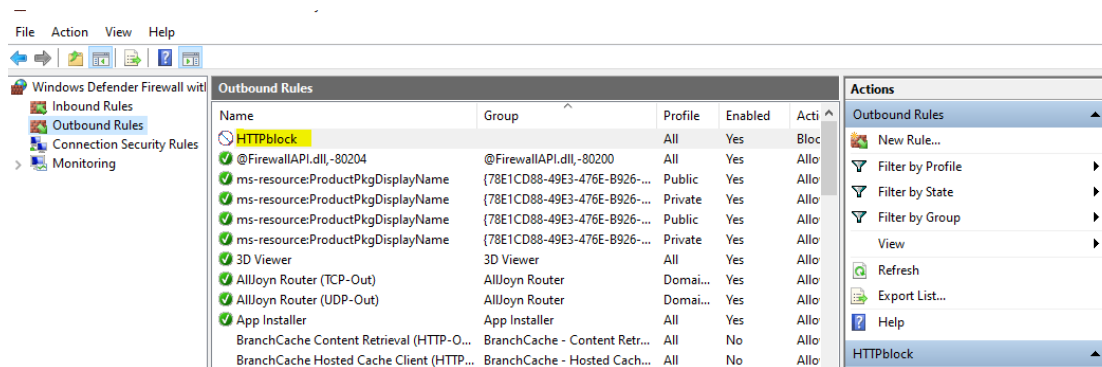
☒ **Public**
Applies when a computer is connected to a public network location.

< Back **Next >** Cancel

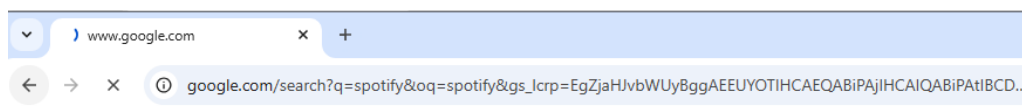
Step 6: Name the rule



New rule for outbound is added.



Now the access is blocked-



This site can't be reached

The web page at https://www.google.com/search?q=spotify&oq=spotify&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTIHCAEQABiPAjIHCAIQABiPatIBCDQwNTVqMGoxqAIAsAIB&sourceid=chrome&ie=UTF-8

q=spotify&oq=spotify&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTIHCAEQABiPAjIHCAIQABiPatIBCDQwNTVqMGoxqAIAsAIB&sourceid=chrome&ie=UTF-8 might be temporarily down or it may have moved permanently to a new web address.

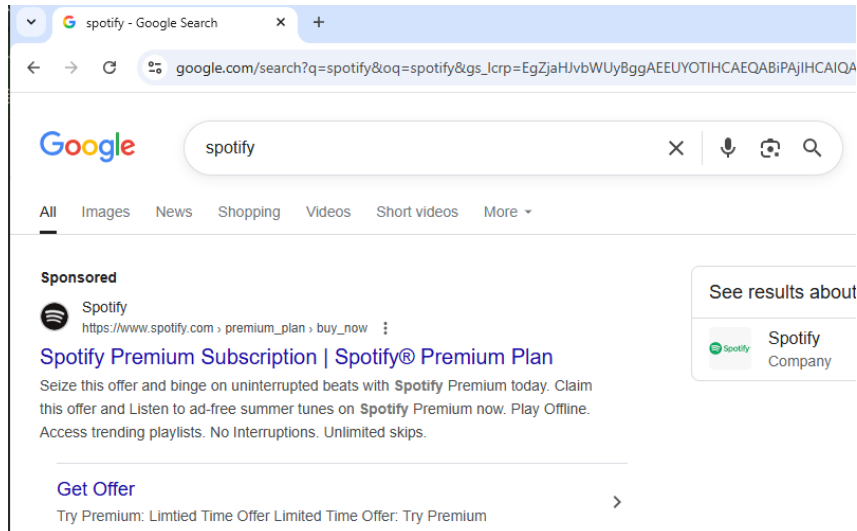
ERR_QUIC_PROTOCOL_ERROR

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.

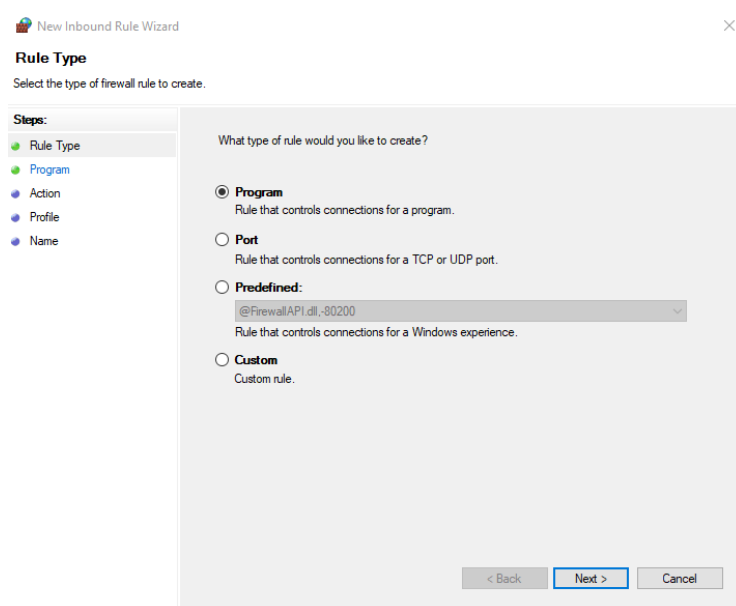
Step 1: Open Windows Defender Firewall.

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.

New Inbound Rule Wizard

Program

Specify the full program path and executable name of the program that this rule matches.

Steps:

- Rule Type
- Program
- Action
- Profile
- Name

Does this rule apply to all programs or a specific program?

☐ **All programs**
Rule applies to all connections on the computer that match other rule properties.

☒ **This program path:**

Example: c:\path\program.exe
 %ProgramFiles%\browser\browser.exe

< Back Next > Cancel

Step 7: Now click on block the connection.

New Inbound Rule Wizard

Action

Specify the action to be taken when a connection matches the conditions specified in the rule.

Steps:

- Rule Type
- Program
- Action
- Profile
- Name

What action should be taken when a connection matches the specified conditions?

☐ **Allow the connection**
This includes connections that are protected with IPsec as well as those are not.

☐ **Allow the connection if it is secure**
This includes only connections that have been authenticated by using IPsec. Connections will be secured using the settings in IPsec properties and rules in the Connection Security Rule node.

☒ **Block the connection**

< Back Next > Cancel

Step 8: Tick all the checkboxes.

New Inbound Rule Wizard

Profile

Specify the profiles for which this rule applies.

Steps:

- Rule Type
- Program
- Action
- Profile
- Name

When does this rule apply?

☒ **Domain**
Applies when a computer is connected to its corporate domain.

☒ **Private**
Applies when a computer is connected to a private network location, such as a home or work place.

☒ **Public**
Applies when a computer is connected to a public network location.

< Back Next > Cancel

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

Steps:

- Rule Type
- Program
- Action
- Profile
- Name

Name:

Then click on finish:

< Back Finish Cancel

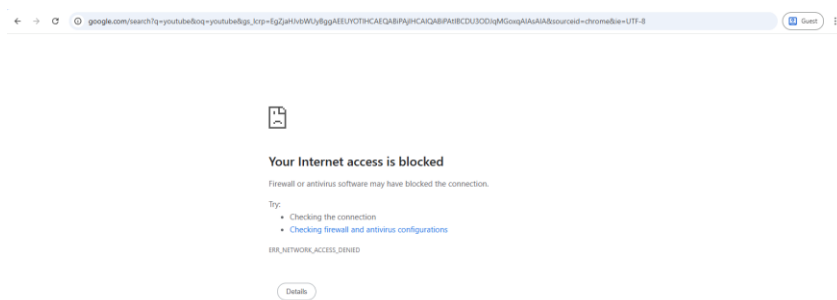
The Inbound rule is created:

Name	Group	Profile	Enabled
360TsLiveUpd.exe		Public	Yes
360TsLiveUpd.exe		Public	Yes
Apache HTTP Server		Public	Yes
Apache HTTP Server		Public	Yes
arduino ide.exe		Public	Yes
arduino ide.exe		Public	Yes
Firefox (C:\Program Files\Mozilla Firefox)		Private	Yes
Firefox (C:\Program Files\Mozilla Firefox)		Private	Yes
Free Download Manager		Private	Yes
Free Download Manager		Private	Yes
Free Download Manager		Domain	Yes
Free Download Manager		Domain	Yes
Free Download Manager		Public	Yes
Free Download Manager		Public	Yes
Half-Life Launcher		Public	Yes
Half-Life Launcher		Public	Yes
java		Public	Yes
java		Public	Yes
mdns-discovery.exe		Public	Yes
mdns-discovery.exe		Public	Yes
Node.js JavaScript Runtime		Public	Yes
Node.js JavaScript Runtime		Public	Yes
packettracer6		Public	Yes
packettracer6		Public	Yes
Port 3306		All	Yes
Port 33060		All	Yes
Port 33060		All	Yes
Portblock		All	Yes
Visual Studio Code		Public	Yes
Visual Studio Code		Public	Yes
ZBrush Executable		Public	Yes
ZBrush Executable		Public	Yes

Step 10: Follow the same steps for Outbound Rule.

Name	Group	Profile	Enabled
Portblock		All	Yes
@FirewallAPI.dll - 80204	@FirewallAPI.dll - 80200	All	Yes
ms-resource:AppTitle	{78E1CD88-49E3-476E-B926-...	All	Yes
ms-resource:AppTitle	{78E1CD88-49E3-476E-B926-...	All	Yes
ms-resource:AppTitle	{78E1CD88-49E3-476E-B926-...	All	Yes
ms-resource:AppTitle	{78E1CD88-49E3-476E-B926-...	All	Yes
ms-resource:ProductPkgDisplayName	{78E1CD88-49E3-476E-B926-...	Private	Yes
ms-resource:ProductPkgDisplayName	{78E1CD88-49E3-476E-B926-...	Private	Yes
ms-resource:ProductPkgDisplayName	{78E1CD88-49E3-476E-B926-...	Public	Yes
ms-resource:ProductPkgDisplayName	{78E1CD88-49E3-476E-B926-...	Public	Yes
Skype	{78E1CD88-49E3-476E-B926-...	All	Yes
Skype	{78E1CD88-49E3-476E-B926-...	All	Yes
3D Viewer		All	Yes
AllJoyn Router (TCP-Out)	AllJoyn Router	Domai...	Yes
AllJoyn Router (UDP-Out)	AllJoyn Router	Domai...	Yes
App Installer	App Installer	All	Yes
BranchCache Content Retrieval (HTTP-Out)	BranchCache - Content Retr...	All	No
BranchCache Hosted Cache Client (HTTP-Out)	BranchCache - Hosted Cach...	All	No
BranchCache Hosted Cache Server (HTTP-Out)	BranchCache - Hosted Cach...	All	No
BranchCache Peer Discovery (WSD-Out)	BranchCache - Peer Discove...	All	No
Captive Portal Flow	Captive Portal Flow	All	Yes
Cast to Device functionality (qWave-TCP-Out)	Cast to Device functionality	Private...	Yes
Cast to Device functionality (qWave-UDP-Out)	Cast to Device functionality	Private...	Yes
Cast to Device streaming server (RTP-Stre...	Cast to Device functionality	Public	Yes
Cast to Device streaming server (RTP-Stre...	Cast to Device functionality	Domain	Yes
Cast to Device streaming server (RTP-Stre...	Cast to Device functionality	Private	Yes
Cloud Identity (TCP-Out)	Cloud Identity	All	Yes
Connected Devices Platform - Wi-Fi Dire...	Connected Devices Platform	Public	Yes
Connected Devices Platform (TCP-Out)	Connected Devices Platform	Domai...	Yes
Connected Devices Platform (UDP-Out)	Connected Devices Platform	Domai...	Yes
Core Networking - DNS (UDP-Out)	Core Networking	All	Yes
Core Networking - Dynamic Host Config...	Core Networking	All	Yes

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

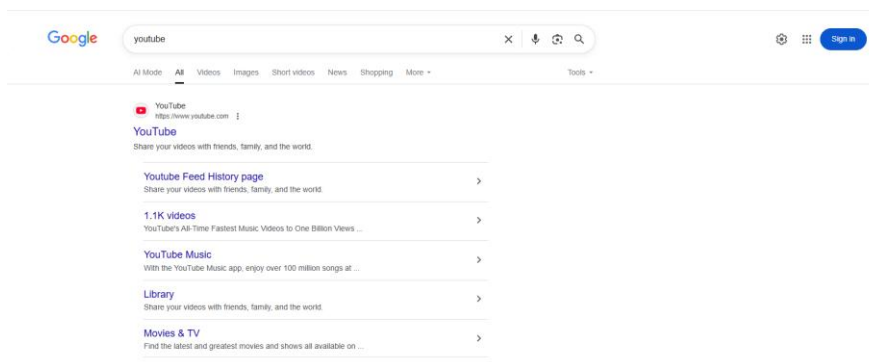


Now to unblock the Chrome:

Go to firewall > Go to inbound > Right click and delete the rule

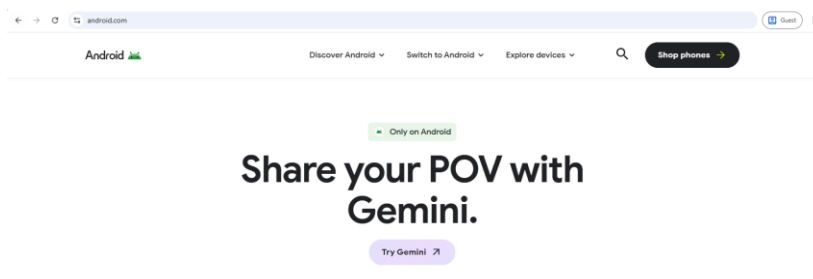
Go to firewall > Go to outbound > Right click and delete the rule

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

```
Command Prompt
Microsoft Windows [Version 10.0.19045.6159]
(c) Microsoft Corporation. All rights reserved.

C:\Users\abc>nslookup android.com
Server:  dns.google
Address:  8.8.8.8

Non-authoritative answer:
Name:    android.com
Addresses:  2404:6800:4009:82a::2004
          142.250.192.100

C:\Users\abc>
```

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

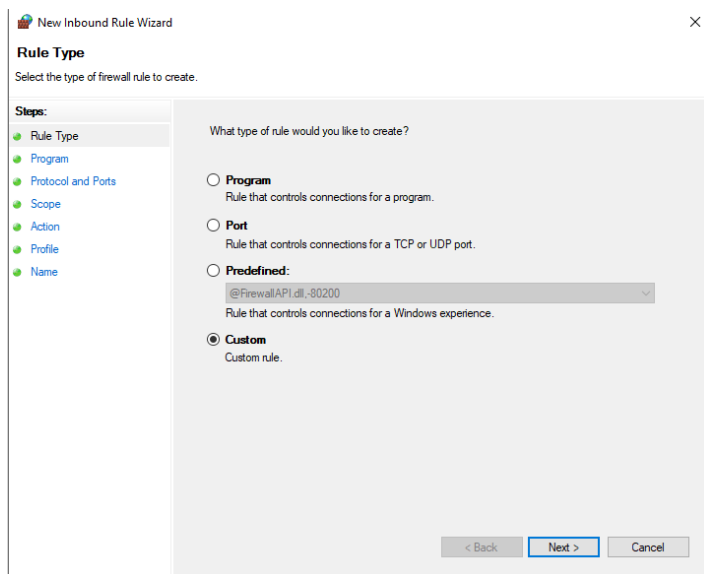
Step 1: Open Windows Defender Firewall.

Step 2: Go to Advance Settings

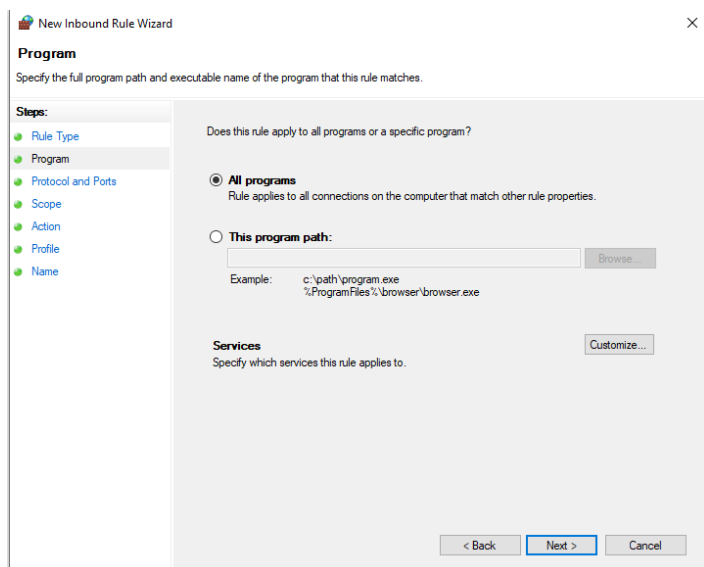
Step 3: Click on Inbound Rule

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

The screenshot shows the 'New Inbound Rule Wizard' window, specifically the 'Protocol and Ports' step. The left sidebar lists the steps: Rule Type, Program, Protocol and Ports (selected), Scope, Action, Profile, and Name. The main area is titled 'To which ports and protocols does this rule apply?'. It contains fields for 'Protocol type' (set to 'Any'), 'Protocol number' (set to '0'), 'Local port' (set to 'All Ports'), and 'Remote port' (set to 'All Ports'). Below these are examples of port ranges: 'Example: 80, 443, 5000-5010'. There is also a 'Customize...' button for 'Internet Control Message Protocol (ICMP) settings'. At the bottom, there are three buttons: '< Back', 'Next >', and 'Cancel'.

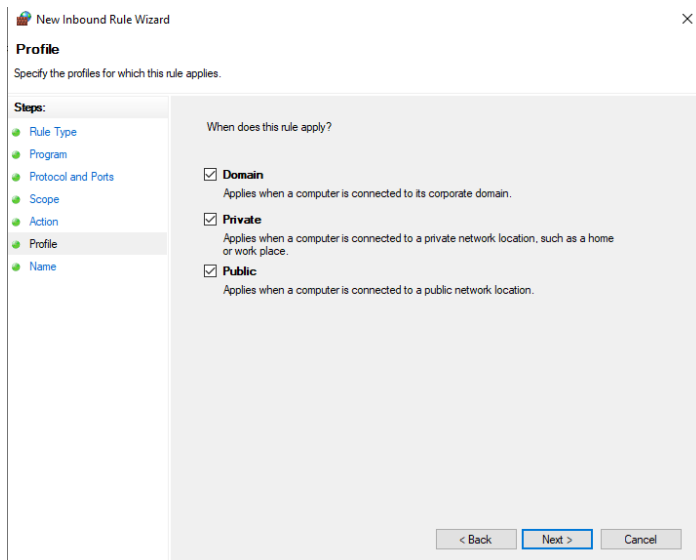
Step 8: Add both the addresses

The screenshot shows the 'New Inbound Rule Wizard' window, specifically the 'Scope' step. The left sidebar lists the steps: Rule Type, Program, Protocol and Ports, Scope (selected), Action, Profile, and Name. The main area is titled 'Specify the local and remote IP addresses to which this rule applies.' It contains two sections: 'Which local IP addresses does this rule apply to?' and 'Which remote IP addresses does this rule apply to?'. Both sections have radio buttons for 'Any IP address' and 'These IP addresses:'. Under 'These IP addresses:', there are text boxes for entering IP addresses and buttons for 'Add...', 'Edit...', and 'Remove...'. At the bottom, there are three buttons: '< Back', 'Next >', and 'Cancel'.

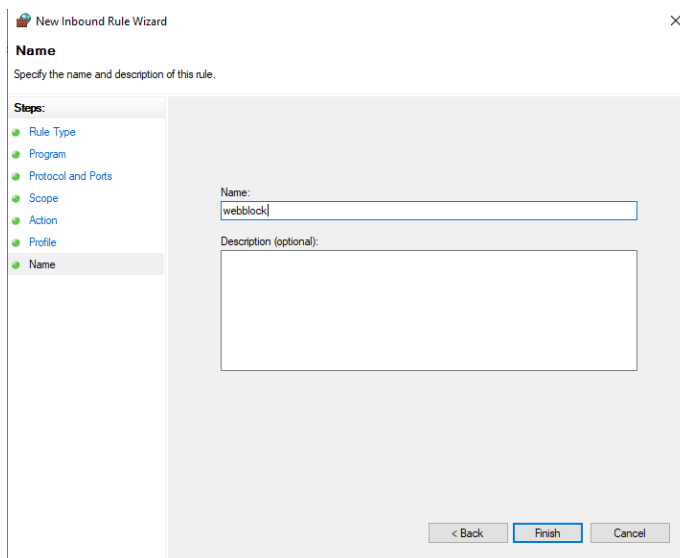
Step 9: Block the connection

The screenshot shows the 'New Inbound Rule Wizard' window, specifically the 'Action' step. The left sidebar lists the steps: Rule Type, Program, Protocol and Ports, Scope, Action (selected), Profile, and Name. The main area is titled 'Specify the action to be taken when a connection matches the conditions specified in the rule.' It contains a section titled 'What action should be taken when a connection matches the specified conditions?'. There are three radio buttons: 'Allow the connection', 'Allow the connection if it is secure', and 'Block the connection'. The 'Block the connection' option is selected. Below the radio buttons are 'Customize...' buttons. At the bottom, there are three buttons: '< Back', 'Next >', and 'Cancel'.

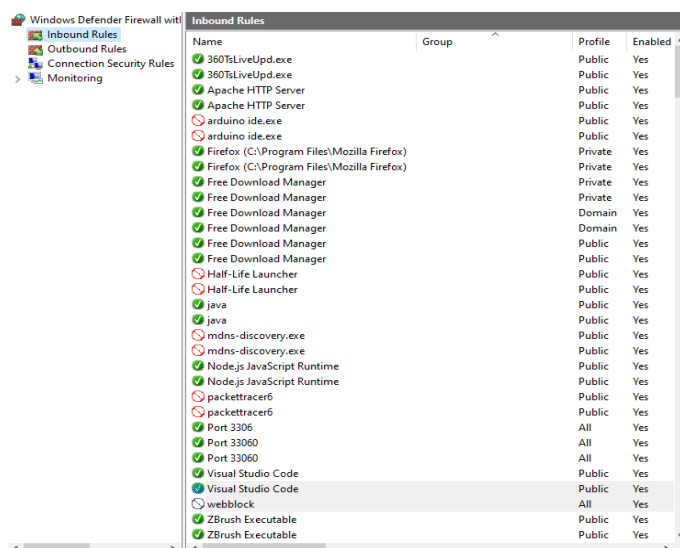
Step 10: Check all the boxes



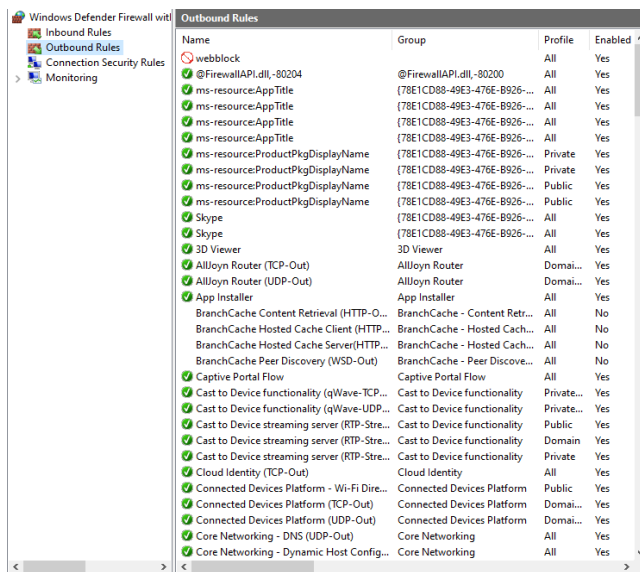
Step 11: Name the rule and click on finish.



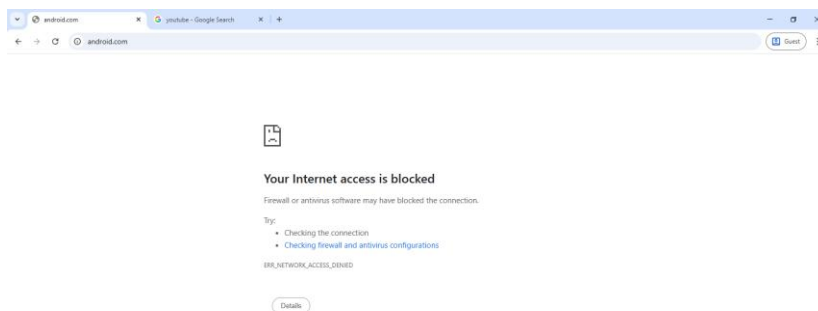
The Inbound Rule is created:



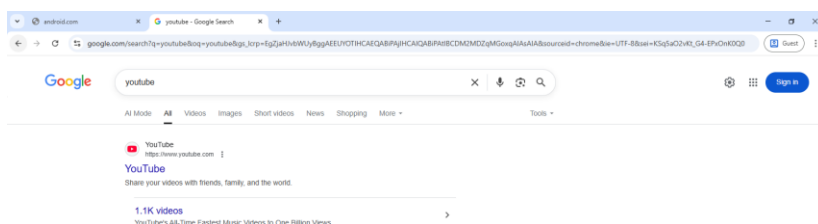
Now, follow the same steps for creating the outbound rule:



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-

