Practical No.1: Implementing Substitution and Transposition Ciphers

Aim: To study and implement the Substitution and Transposition Ciphers

Theory:

Substitution Cipher: In a Substitution Cipher, each letter in the plain text (original message) is replaced by another letter or symbol to create the cipher text (encrypted message). This method is called substitution because each letter is substituted with another according to a predetermined rule.

Caesar Cipher: The Caesar Cipher involves shifting each letter in the plaintext by a fixed number of positions in the alphabet. The shift value is often referred to as the key, and it determines the mapping from plain text to cipher text.

Transposition Cipher: Unlike the Substitution Cipher, which substitutes each letter with another, the Transposition Cipher preserves the original letters but changes their order. One of the well-known examples of a Transposition Cipher is the Railfence Cipher.

Railfence Cipher: The Railfence Cipher is a basic form of a Transposition Cipher that rearranges the letters of the plain text by writing them in a zigzag pattern along a set number of "rails." The rails are imaginary horizontal lines on which the plain text characters are placed.

Code: Python code for implementing Caesar Cipher

Caesar_cipher.py

```
def encrypt(msg, key):
   enc_msg = ""
   for ch in msg:
      if(ch.isupper()):
         new_ch = chr((ord(ch)-ord("A")+key)%26 + ord("A"))
         enc_msg += new_ch
      if(ch.islower()):
         new_ch = chr((ord(ch)-ord("a")+key)%26 + ord("a"))
         enc_msg += new_ch
   return enc_msg
def decrypt(msg, key):
   dec_msg = ""
   for ch in msg:
      if(ch.isupper()):
```

```
            new_ch = chr((ord(ch)-ord("A")-key)%26 + ord("A"))
            dec_msg += new_ch
        if(ch.islower()):
            new_ch = chr((ord(ch)-ord("a")-key)%26 + ord("a"))
            dec_msg += new_ch
    return dec_msg
text = input("Enter the text to encrpyt: ")
key = int(input("Enter the key for encrpytion: "))
enc = encrypt(text, key)
dec = decrypt(enc, key)
print(f"\nThe encrpyted text is: {enc}\nThe decryption of encrypted text is: {dec}")
```

OUTPUT:

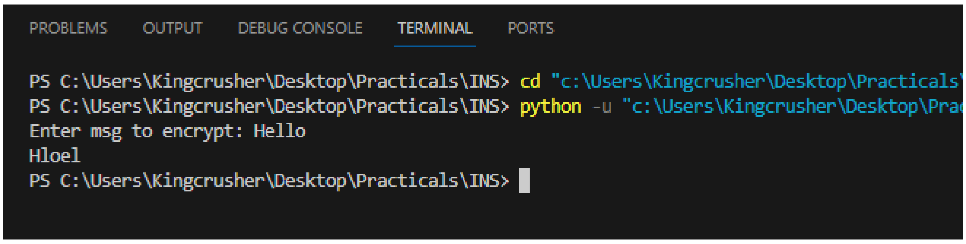
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Kingcrusher\Desktop\Practicals\INS> cd "c:\User
PS C:\Users\Kingcrusher\Desktop\Practicals\INS> python -u "
Enter the text to encrpyt: Hello
Enter the key for encrpytion: 3

The encrpyted text is: Khoor
The decryption of encrypted text is: Hello
PS C:\Users\Kingcrusher\Desktop\Practicals\INS>
```

Code: Python code for implementing Railfence Cipher

Railfence_cipher.py

```
# Railfence cipher
def encrpyt(msg):
    enc_msg = ""
    rails = ["", ""]
    for i in range(len(msg)):
        idx = i%2
        rails[idx] += msg[i]
    for m in rails:
        enc_msg += m
    return enc_msg
msg = input("Enter msg to encrypt: ")
print(encrpyt(msg))
```

OUTPUT:



Practical No.2: RSA Encryption and Decryption

Aim: To study and implement the RSA Encryption and Decryption

Theory: RSA (Rivest-Shamir-Adleman) Algorithm: RSA is a widely used asymmetric encryption algorithm that provides secure communication over untrusted networks.

Key Generation: The RSA algorithm involves the generation of a public-private key pair. The key generation process consists of the following steps:

a) Select two distinct prime numbers, p and q.

b) Compute the modulus, N, by multiplying p and q: $N = p * q$.

c) Calculate Euler's function, $\phi(N)$, where $\phi(N) = (p - 1) * (q - 1)$.

d) Choose an integer, e, such that $1 < e < \phi(N)$ and e is coprime with $\phi(N)$. This means that e and $\phi(N)$ should have no common factors other than 1.

e) Find the modular multiplicative inverse of e modulo $\phi(N)$, denoted as d. In other words, d is an integer such that $(d * e) \% \phi(N) = 1$.

f) The public key consists of the modulus, N, and the public exponent, e. The private key consists of the modulus, N, and the private exponent, d.

Code: Python code for implementing RSA Algorithm

RSA.py

```
import math

class RSA:

    def __init__(self, prime1, prime2) -> None:

        self.p = prime1

        self.q = prime2

        self.N = self.p*self.q

        self.phi = (self.p-1)*(self.q-1)

        self.e = self.getE()

        self.d = self.getD()


    def isPrime(self, n):

        for i in range(2, int(math.sqrt(n))):

            if(n%1==0):
```

```
                return False
        return True
    def getE(self):
        n = -1
        for num in range(2,self.N):
            if self.isPrime(num):
                if(self.phi%num==0):
                    continue
                else:
                    n = num
                    break
        return n
    def getD(self):
        d = 1
        while((d*self.e)%self.phi != 1):
            d += 1
        return d
    def encrypt(self, msg):
        enc_msg = (msg**self.e)%self.N
        return enc_msg
    def decrypt(self, msg):
        dec_msg = (msg**self.d)%self.N
        return dec_msg
p, q = 7, 17
r = RSA(p, q)
print(f"The value of p is: {r.p}\n The value of q is: {r.q}")
print(f"The value of N is {r.N}\n The value of phi is {r.phi}")
print(f"The value of E is: {r.e}\n The value of D is {r.d}")
```

print(f"The encryprion of PT is: {r.encrypt(10)}")

print(f" The decryprion of CT is: {r.decrypt(40)}")

OUTPUT:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Kingcrusher\Desktop\Practicals\INS> cd "c:\Users\K
PS C:\Users\Kingcrusher\Desktop\Practicals\INS> python -u "c:\
The value of p is: 7
 The value of q is: 17
The value of N is 119
 The value of phi is 96
The value of E is: 5
 The value of D is 77
The encryprion of PT is: 40
 The decryprion of CT is: 10
PS C:\Users\Kingcrusher\Desktop\Practicals\INS>
```

Practical No.3: Message Authentication Codes (MAC)

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

Theory:

Message Authentication Code (MAC): MAC is a technique used to ensure the integrity and authenticity of messages exchanged between two parties. It involves the use of a secret key and a cryptographic hash function to generate a tag or code that can be appended to the message.

MAC Generation Process:

To generate a MAC using the MD5 algorithm, follow these steps:

a) Both the sender and receiver must agree on a secret key, K, which is known only to them.

b) Concatenate the message, M, and the secret key, K: Concatenated Data = M || K (|| denotes concatenation).

c) Apply the MD5 algorithm to the Concatenated Data to obtain the MAC: MAC = MD5(Concatenated Data).

d) MAC Verification Process:

To verify the integrity and authenticity of a received message using the MAC, follow these steps:

a) Receive the message, M, and the MAC, MAC.

b) Concatenate the received message, M, with the secret key, K: Concatenated Data = M || K.

c) Apply the MD5 algorithm to the ConcatenatedData to compute the recalculated MAC: RecalculatedMAC = MD5(ConcatenatedData).

d) Compare the RecalculatedMAC with the received MAC. If they match, the message is considered authentic and intact.

Code: To implement MAC

MAC.py

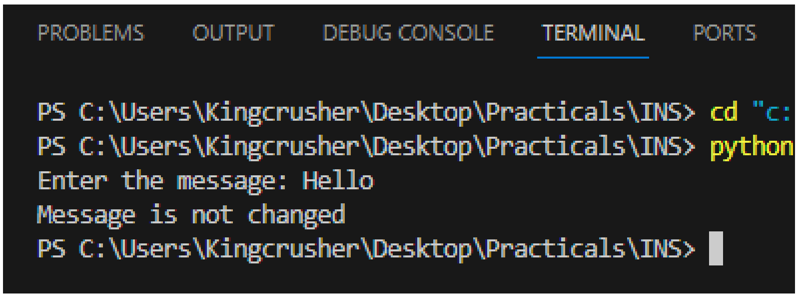
```
import hashlib

def MAC(msg):

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

   return result.hexdigest()

def encrypt(msg, key):

   enc_msg = ""
```

```
    for ch in msg:
        if(ch.isupper()):
            new_ch = chr((ord(ch)-ord("A")+key)%26 + ord("A"))
            enc_msg += new_ch
        if(ch.islower()):
            new_ch = chr((ord(ch)-ord("a")+key)%26 + ord("a"))
            enc_msg += new_ch
    mac = MAC(enc_msg)
    return [enc_msg, mac]
def decrypt(msg, key, mac):
    dmac = MAC(msg)
    if(dmac==mac):
        print("Message is not changed")
    else:
        print("Message is changed")
    dec_msg = ""
    for ch in msg:
        if(ch.isupper()):
            new_ch = chr((ord(ch)-ord("A")-key)%26 + ord("A"))
            dec_msg += new_ch
        if(ch.islower()):
            new_ch = chr((ord(ch)-ord("a")-key)%26 + ord("a"))
            dec_msg += new_ch
    return dec_msg
```

If (msg is not modified):

```
msg = input("Enter the message: ")

enc = encrypt(msg, 2)

dec = decrypt(enc[0], 2, enc[1])
```

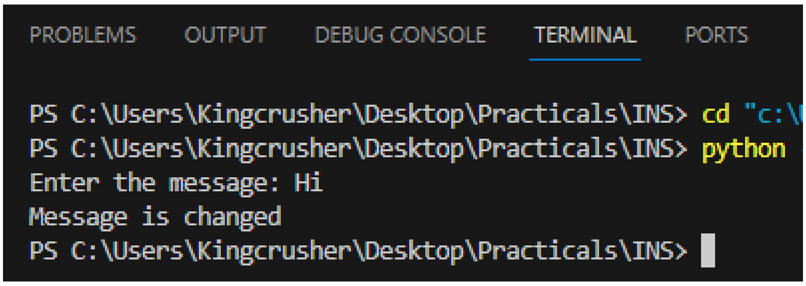
OUTPUT:



If (msg is modified):

```
msg = input("Enter the message: ")

enc = encrypt(msg, 2)

enc[0] = "hello"

dec = decrypt(enc[0], 2, enc[1])
```

OUTPUT:



Practical No.4: Digital Signatures

Aim: To study and implement the Digital Signature algorithm

Theory:

Digital Signature: A digital signature is a cryptographic technique that uses asymmetric encryption algorithms, such as RSA (Rivest-Shamir-Adleman), to bind the identity of the signer with the content of a message.

Digital Signature Generation Process: To generate a digital signature using RSA, follow these steps:

a) The signer generates a key pair: a private key (d) and a public key (e, N).

b) The signer computes the hash value of the message using a cryptographic hash function, such as SHA-256, to ensure data integrity.

c) The signer applies a mathematical function to the hash value using their private key (d) to generate the digital signature

Digital Signature Verification Process: To verify the authenticity and integrity of a received message using a digital signature, follow these steps:

a) The recipient obtains the signer's public key (e, N).

b) The recipient computes the hash value of the received message using the same cryptographic hash function.

c) The recipient applies a mathematical function to the received digital signature using the signer's public key (e, N).

d) The recipient compares the computed signature with the received digital signature. If they match, the message is considered authentic and intact.

Code: To implement digital signature using RSA

Digital_signature.py

```
import math, hashlib

class RSA:

    def __init__(self, prime1, prime2) -> None:

        self.p = prime1

        self.q = prime2

        self.N = self.p*self.q

        self.phi = (self.p-1)*(self.q-1)

        self.e = self.getE()
```

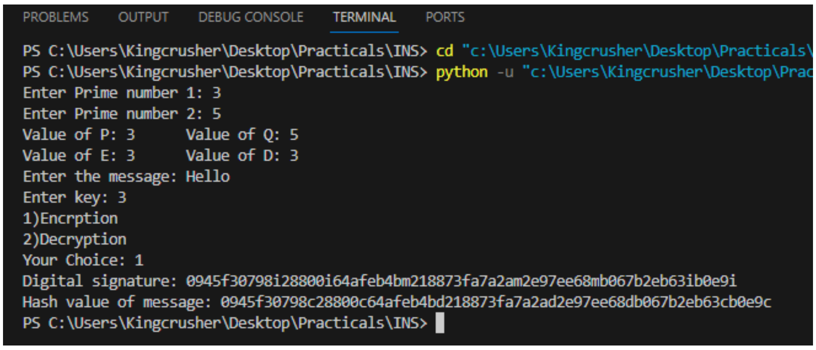
```
    self.d = self.getD()
def isPrime(self, n):
    for i in range(2, int(math.sqrt(n))):
        if(n%1==0):
            return False
    return True
def getE(self):
    n = -1
    for num in range(2,self.N):
        if self.isPrime(num):
            if(self.phi%num==0):
                continue
            else:
                n = num
                break
    return n
def getD(self):
    d = 1
    while((d*self.e)%self.phi != 1):
        d += 1
    return d
def encrypt(self, msg):
    letters=""
    for i in msg:
        if(i.isdigit()!=True):
            posi=ord(i)-ord("a")
            ct=((posi**self.e)%self.N)%26
            letters=letters+(chr(ct+97))
```

```
            else:
                letters = letters + i
        return letters
    def decrypt(self, msg):
        letters=""
        for i in msg:
            if(i.isdigit()!=True):
                posi=ord(i)-ord("a")
                pt=((posi**self.d)%self.N)%26
                letters=letters+(chr(pt+97))
            else:
                letters = letters + i
        return letters
def hashcode(msg,key):
    msgKey=msg+key
    hashed=hashlib.sha256(msgKey.encode("UTF-8")).hexdigest()
    return hashed
p=int(input("Enter Prime number 1: "))
q=int(input("Enter Prime number 2: "))
rsa=RSA(p,q)
if(rsa.isPrime(p) and rsa.isPrime(q)):
    print(f"Value of P: {rsa.p}     Value of Q: {rsa.q}\nValue of E: {rsa.e}     Value of D: {rsa.d}")
    msg=input("Enter the message: ")
    key=input("Enter key: ")
    choice=int(input("1)Encrption\n2)Decryption\nYour Choice: "))
    if choice==1:
        hash_value = hashcode(msg,key)
        digi_sign = rsa.encrypt(hash_value)
```

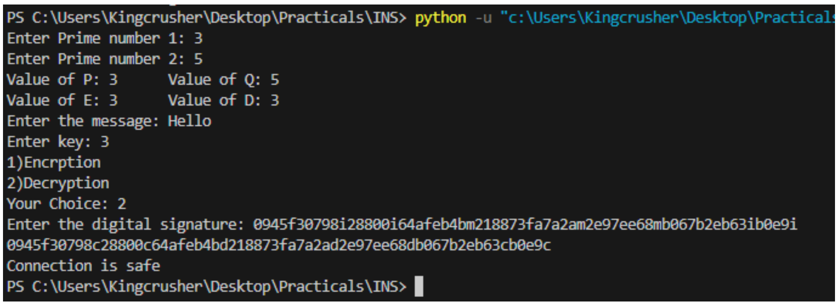
```
        print(f"Digital signature: {digi_sign}\nHash value of message: {hash_value}")
    elif choice==2:
        digi_sign = input("Enter the digital signature: ")
        hash_value = rsa.decrypt(digi_sign)
        print(hash_value)
        hashValueOfMsg = hashcode(msg,key)
        if(hash_value == hashValueOfMsg):
            print("Connection is safe")
        else:
            print("Connection is not safe")
else:
    print("Numbers are not prime")
```

OUTPUT:

For encryption:



For decryption:



Practical No.5: Key Exchange using Diffie-Hellman

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

Theory:

Key Exchange: Key exchange is a fundamental concept in cryptography that allows two parties to securely establish a shared secret key over an insecure communication channel.

Diffie-Hellman Algorithm: The Diffie-Hellman algorithm is a widely used asymmetric key exchange algorithm. It enables two parties to securely establish a shared secret key over an insecure communication channel.

High-level working explanation of the Diffie-Hellman algorithm:

a) Select a large prime number, p, and a primitive root modulo p, g. These values are publicly known.

b) Each party, Alice and Bob, generates a private key, a and b, respectively.

c) Both Alice and Bob calculate their public keys:

d) Alice: $A = g^a \bmod p$

e) Bob: $B = g^b \bmod p$

f) Alice and Bob exchange their public keys over the insecure channel.

Code: To implement Diffie – Helmen key exchange

Diff-hell.py

```
def isprime(n):
    if(n<2):
        return False
    for i in range(2,n//2):
        if n%i==0:
            return False
    return True
def exc(n, g):
    x = int(input("Enter your choice of number: "))
    A = (g**x)%n
    return [A, x]
```

```
def exc2(n,x,b):
    k1 = (b**x)%n
    print(f"The key to use for cryptography is {k1}")
n = int(input("Enter a prime number 'n1': "))
g = int(input("Enter a prime number 'n2': "))
if isprime(n) and isprime(g):
    print("For user A:")
    A = exc(n,g)
    print("For user B:")
    B = exc(n,g)
    print("For user A:")
    exc2(n,A[1],B[0])
    print("For user B:")
    exc2(n,B[1],A[0])
else:
    print("Entered number is not prime")
```

OUTPUT:

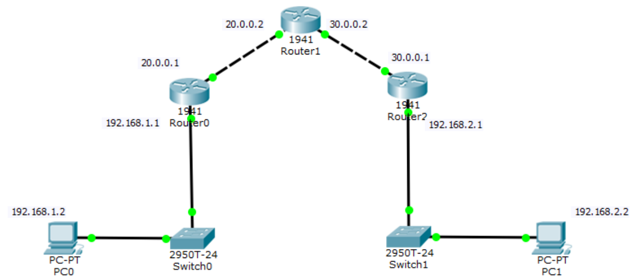
```
PS C:\Users\Kingcrusher\Desktop\Practicals\INS> cd "c:\Users\King
PS C:\Users\Kingcrusher\Desktop\Practicals\INS> python -u "c:\Use
Enter a prime number 'n1': 3
Enter a prime number 'n2': 5
For user A:
Enter your choice of number: 7
For user B:
Enter your choice of number: 11
For user A:
The key to use for cryptography is 2
For user B:
The key to use for cryptography is 2
PS C:\Users\Kingcrusher\Desktop\Practicals\INS>
```

Practical No.6: IP Security (IPSec) Configuration

Aim: To Configure IPSec on network devices to provide secure communication and protect against unauthorized access and attacks.

Theory:

Topology: We use the following topology for the present case



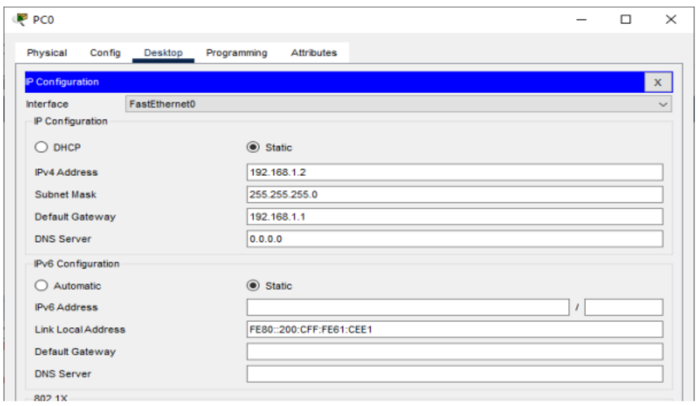
ISAKMP Policy Parameters

Parameters	Parameter Options and Defaults	R1	R2
Key Distribution Method	Manual or ISAKMP	ISAKMP	ISAKMP
Encryption Algorithm	DES. 3DES or AES	AES-256	AES-256
Hash Algorithm	MD5 or SHA-1	SHA-1	SHA-1
Authentication Method	Pre-shared Key or RSA	Pre-shared	Pre-shared
Key Exchange	DH Group 1, 2 or 5	Group 5	Group 5
ISE SA Lifetime	86400 seconds or less	86400	86400
ISAKMP Key	User defined	ismile	ismile

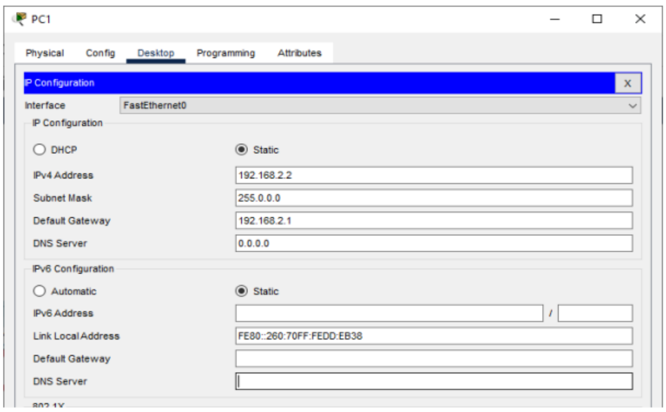
IPSec Policy Parameters

Parameters	R1	R2
Transform Set Name	VPN-SET	VPN-SET
ESP Transform Encryption	esp-aes	esp-aes
ESP Transform Authenticatior	esp-sha-hmac	esp-sha-hmac
Peer IP Address	30.0.0.1	20.0.0.1
Traffic to be Encrypted	R1->R2	R2->R1
Crypto Map Name	IPSEC-MAP	IPSEC-MAP
SA Establishment	ipsec-isakmp	ipsec-isakmp

Configuring PC0:

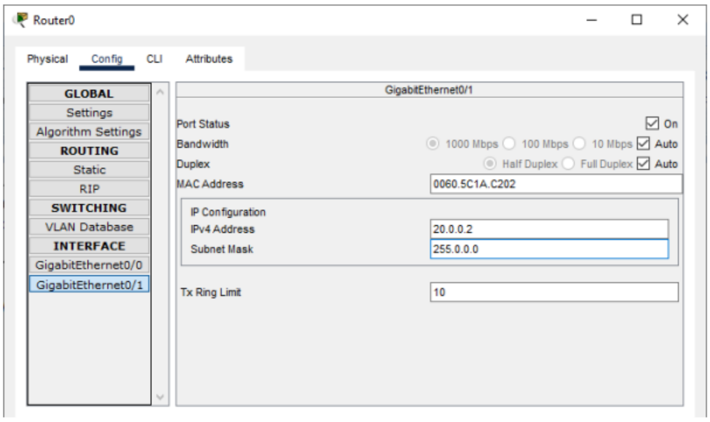


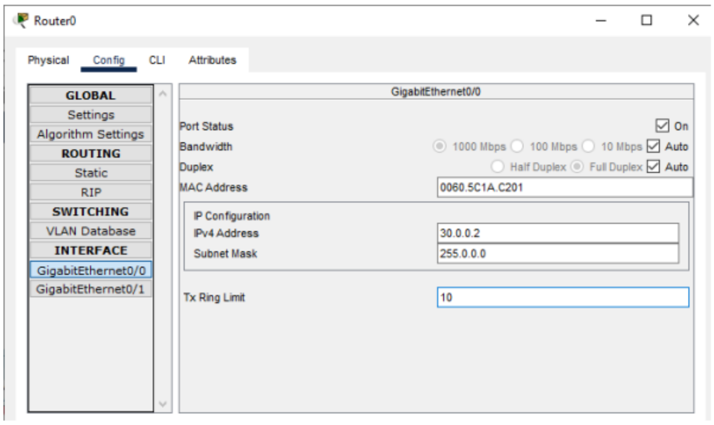
Configuring PC1:



Configuring Router0:

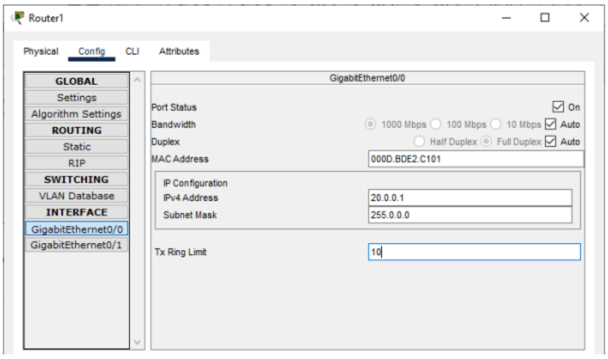
Interface GigabitEthernet0/1: **Interface GigabitEthernet0/0:**

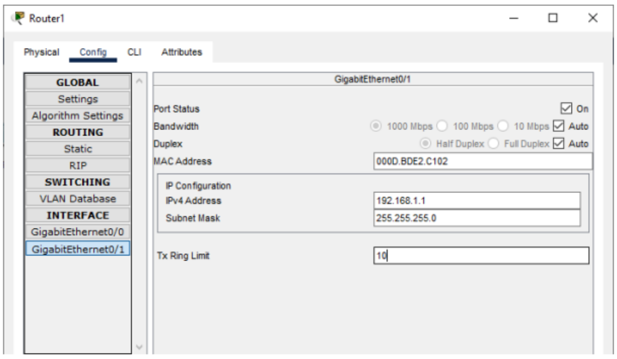




Configuring Router1:

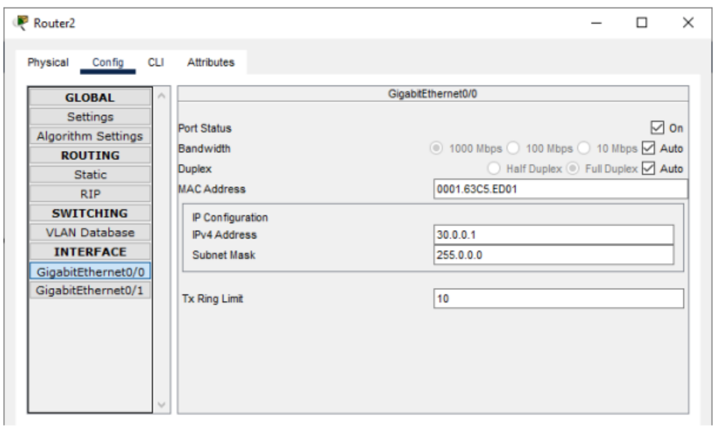
Interface GigabitEthernet0/0: **Interface GigabitEthernet0/1:**

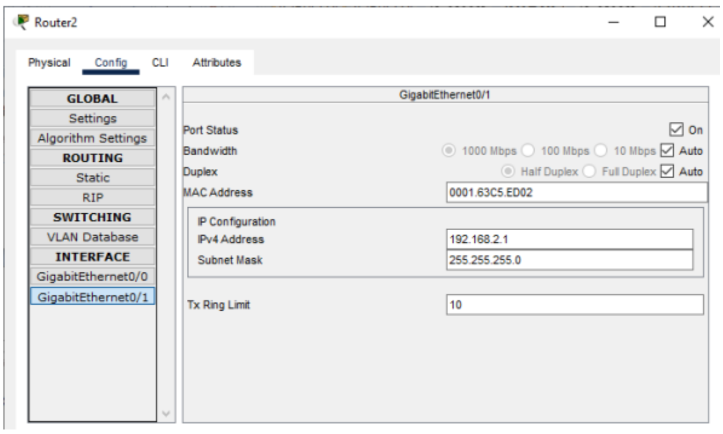




Configuring Router2:

Interface GigabitEthernet0/0: **Interface GigabitEthernet0/1:**

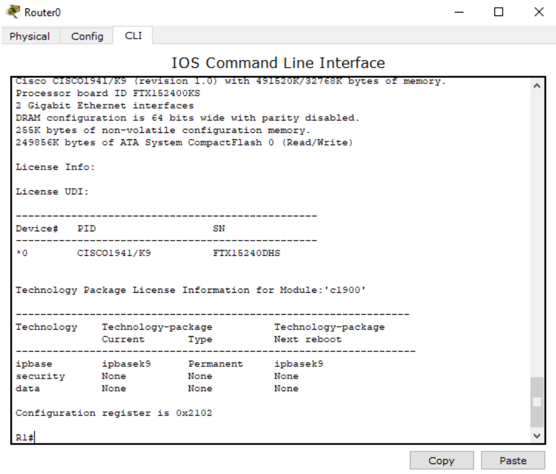




Checking and Enabling the Security features in Router R1 and R2: Enter the following command in the CLI mode of Router1

Router(config)#ip route 0.0.0.0 0.0.0.0 20.0.0.2

Router(config)#hostname R1



R1(config)#exit

R1#show version

R1#

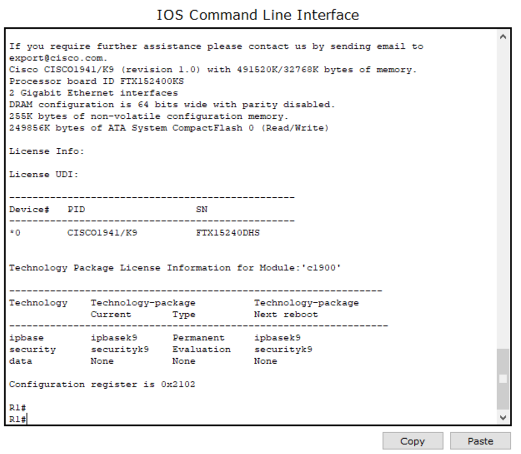
R1#configure terminal

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

R1(config)#

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

R1(config)#exit



R1#

R1#copy run startup-config

R1#reload

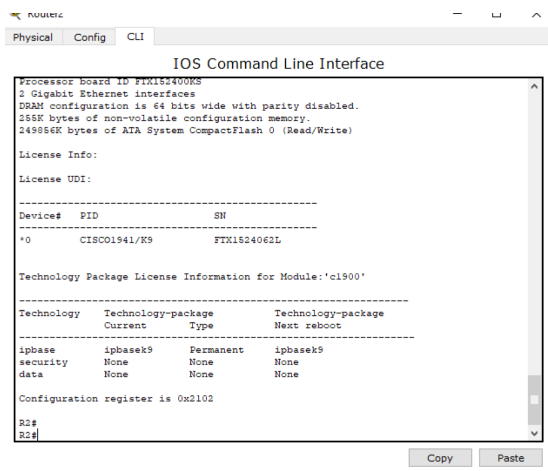
R1>enable

R1#show version

Enter the following command in the CLI mode of Router2

Router(config)#ip route 0.0.0.0 0.0.0.0 30.0.0.2

Router(config)#hostname R2



R2(config)#exit

R2#show version

R2#

R2#configure terminal

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

R2(config)#

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

R2(config)#exit

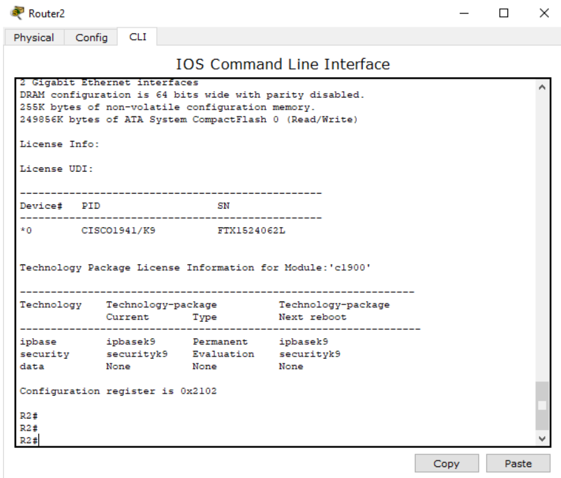
R2#

R2#copy run startup-config

R2#reload

R2>enable

R2#show version



Enter the following command in the CLI mode of Router0

Router>enable

Router#configure terminal

Router(config)#hostname R0

R0(config)#

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

R1#configure terminal

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

R1(config)#crypto isakmp policy 10

R1(config-isakmp)#encryption aes 256

R1(config-isakmp)#authentication pre-share

R1(config-isakmp)#group 5

R1(config-isakmp)#exit

R1(config)#crypto isakmp key ismile address 30.0.0.1

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

R1(config)#

R2#

R2#configure terminal

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

R2(config)#crypto isakmp policy 10 R2(config-isakmp)#encryption aes 256

R2(config-isakmp)#authentication pre-share

R2(config-isakmp)#group 5

R2(config-isakmp)#exit

R2(config)#crypto isakmp key ismile address 20.0.0.1

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

R2(config)#

R1>enable

R1#configure terminal

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

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

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

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

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

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

R1(config-crypto-map)#exit

R1(config)#interface g0/0

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

R2>enable

R2#configure terminal

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

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

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

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

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

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

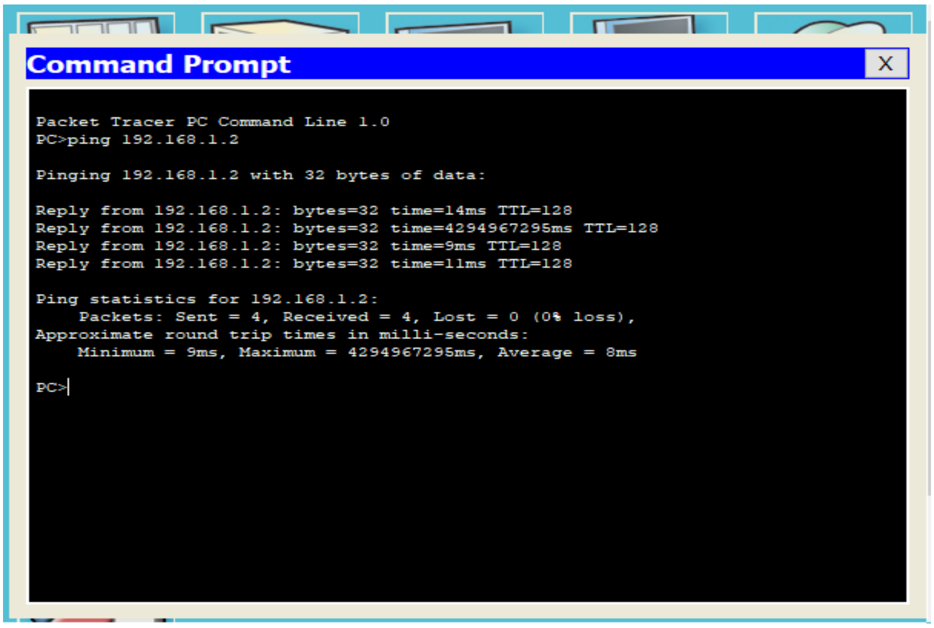
R2(config-crypto-map)#exit

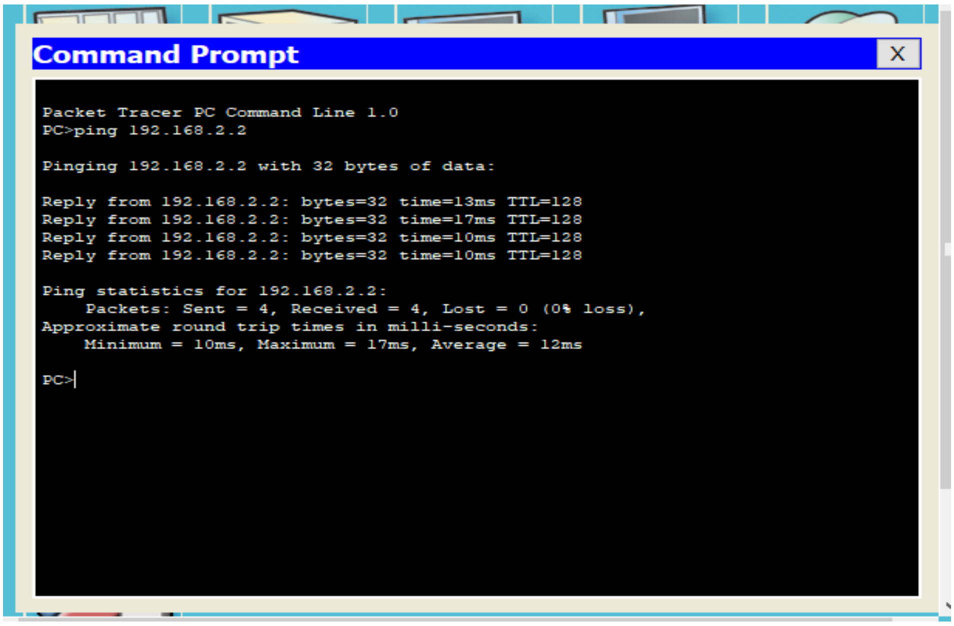
R2(config)#interface g0/0

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

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

Output: Pinging PC2(192.168.2.2) from PC1 and then PC1(192.168.1.2) from PC2





Practical No.7: Malware Analysis and Detection

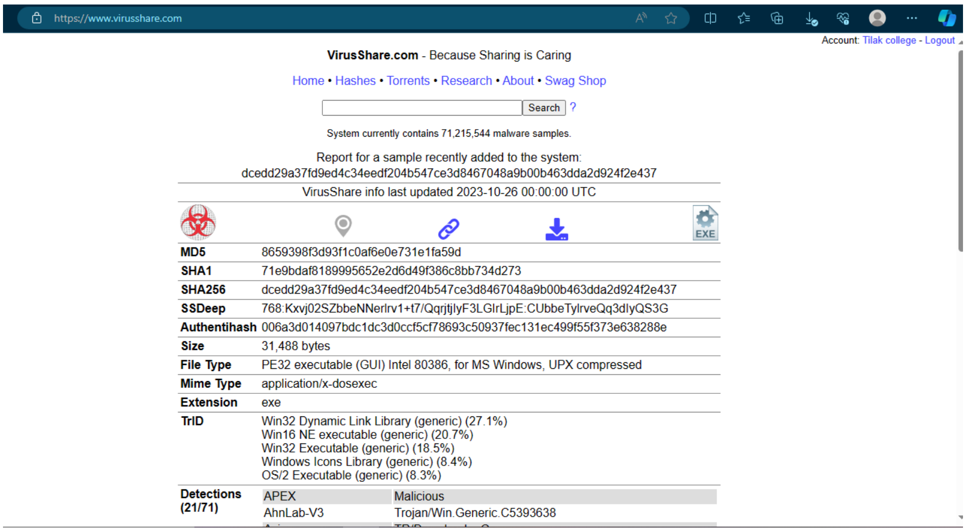
Aim: To do Detect and Analize Malware (Clean Samples)

Theory:

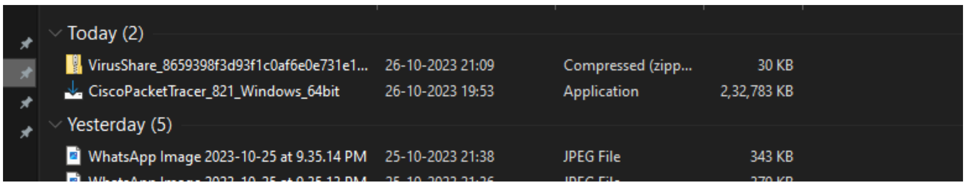
Malware, short for "malicious software," refers to a broad category of software programs or code specifically designed to infiltrate, damage, disrupt, or gain unauthorized access to computer systems, networks, and digital devices.

Analysis: For analyzing the Malware, we need one. A clean sample of the Malware needs to be downloaded from a trusted website, the downloading and analysis is demonstrated by the following steps

1) We select the website www.virusshare.com for downloading the clean sample of Malware (an account needs to be created for the same). Any other source can be selected to download the Malware (clean sample and authorized site)

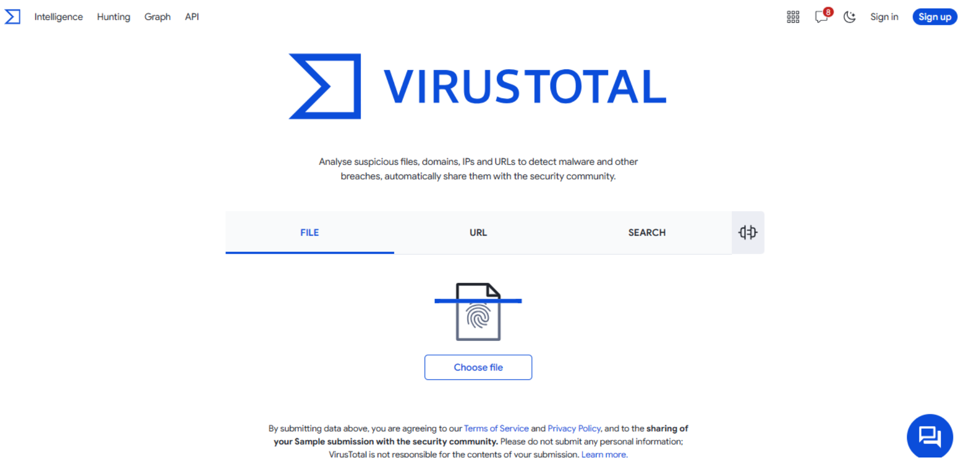


2) By clicking the above download icon the Malware gets downloaded in ZIP format.



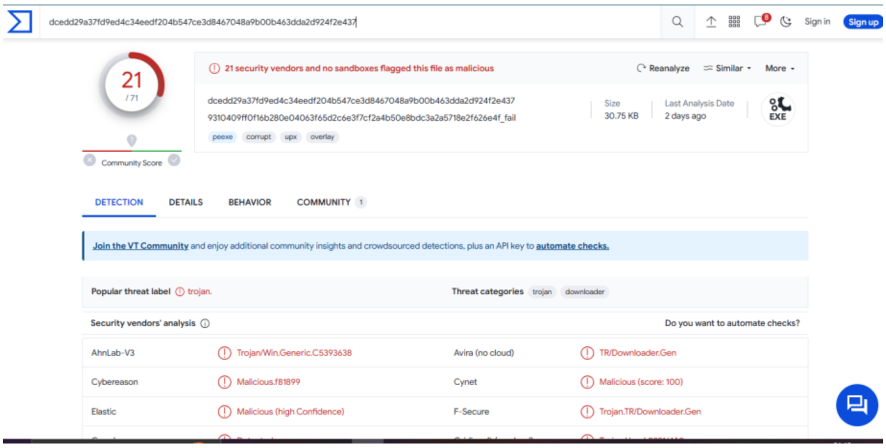
3) For unzip the password is "infected", there is no need to unzip the file, we create a folder "Malware" on desktop and save the file in the folder

4) In order to analyse the Malware, we select the website www.virustotal.com



5) Click on “Choose File” and select the file from the location (ZIP file will do, if asks for password enter infected)

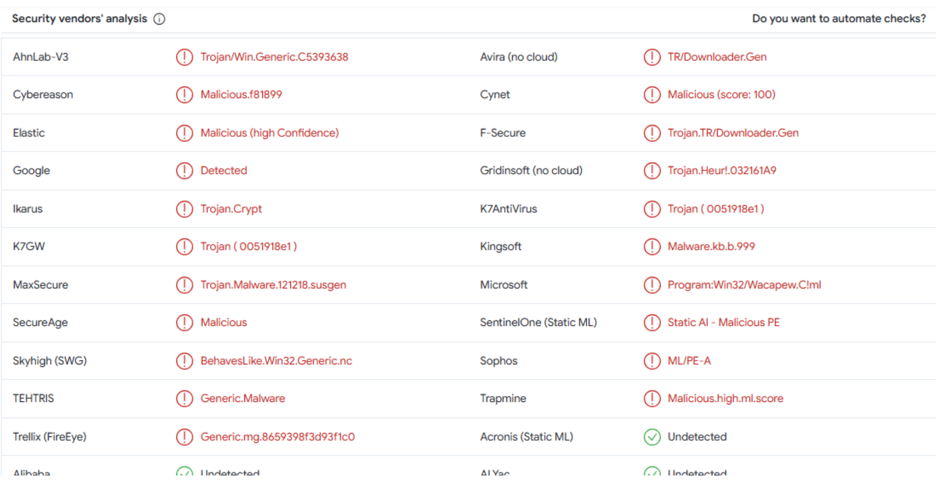
6) We get the following after the upload is complete



We interpret the following findings

a) 21 security vendors out of 71 flagged this file as malicious

b) The detection tab shows the threats-type which were flagged by the vendors for e.g.



Security vendors' analysis

Do you want to automate checks?

AhnLab-V3	Trojan/Win.Generic.C5393638	Avira (no cloud)	TR/Downloader.Gen
Cybereason	Malicious.f81899	Cynet	Malicious (score: 100)
Elastic	Malicious (high Confidence)	F-Secure	Trojan.TR/Downloader.Gen
Google	Detected	Gridinsoft (no cloud)	Trojan.Heur!.032161A9
Ikarus	Trojan.Crypt	K7AntiVirus	Trojan (0051918e1)
K7GW	Trojan (0051918e1)	Kingsoft	Malware.kb.b.999
MaxSecure	Trojan.Malware.121218.susgen	Microsoft	Program:Win32/Wacapew.C!ml
SecureAge	Malicious	SentinelOne (Static ML)	Static AI - Malicious PE
Skyhigh (SWG)	BehavesLike.Win32.Generic.nc	Sophos	ML/PE-A
TEHTRIS	Generic.Malware	Trapmine	Malicious.high.ml.score
Trellix (FireEye)	Generic.mg.8659398f3d93f1c0	Acronis (Static ML)	Undetected
Alibaba	Undetected	ALYac	Undetected

c) The details tab gives the following information

i. Basic properties

ii. History

iii. Compiler products

iv. Header

v. Sections

vi. Imports

vii. Exports

viii. Overlays

d) The Behavior tab gives the following information

i. Activity summary

ii. MITRE ATT&CK Tactics and Techniques

iii. Behavior Similarity Hashes iv. Process and service actions

Practical No.8: Firewall Configuration and Rule based Filtering

Aim: To configure and test firewall rules to control network traffic, filter packets based on specified criteria, and protect network resources from unauthorized access.

Theory:

We would use firewall to block

1) A Port

2) A Website

Part 1: Blocking the HTTP and HTTPS (Port 80 and Port 443) using the Firewall

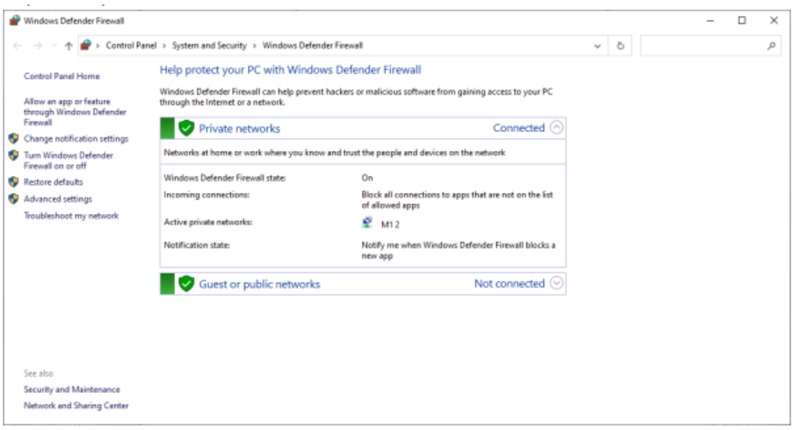
Before starting with the blocking port process, we note that the applications running at the server end are identified with the well-known Port numbers, some of the commonly used are as follows

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

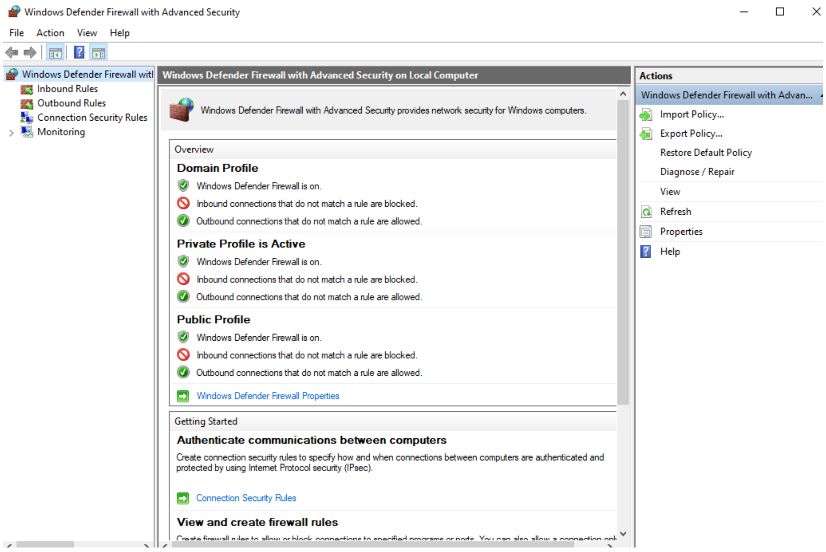
We perform the blocking Port operation as follows:

Step 1: We access any website through the browser and confirm that the HTTP/HTTPS protocols are working.

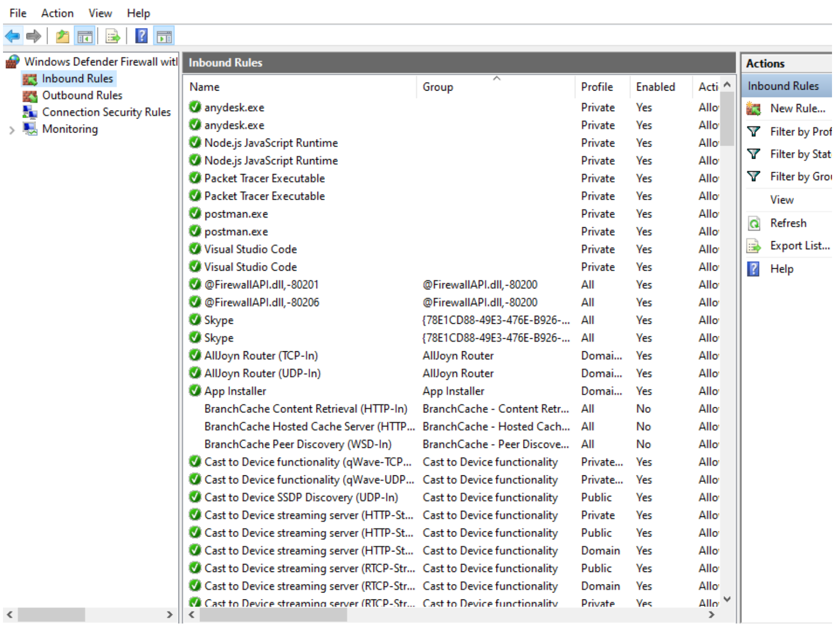
Step 2: We open 'Windows Defender Firewall'



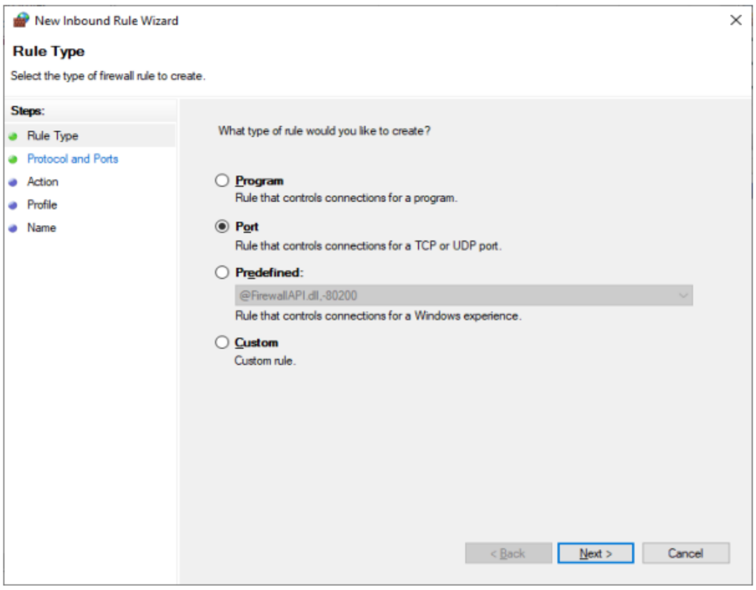
Next we click on 'Advanced settings'



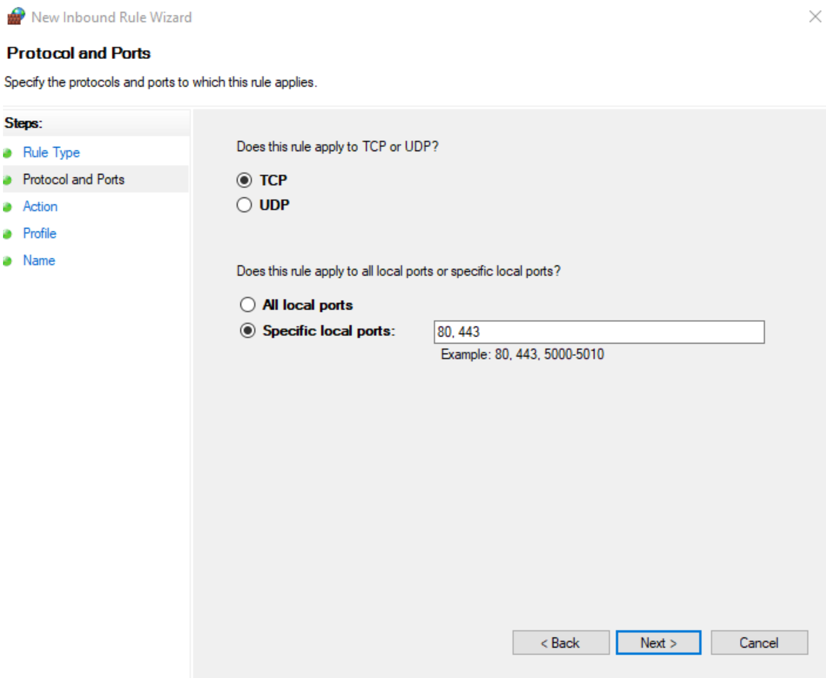
Next we click on ‘Inbound Rules’



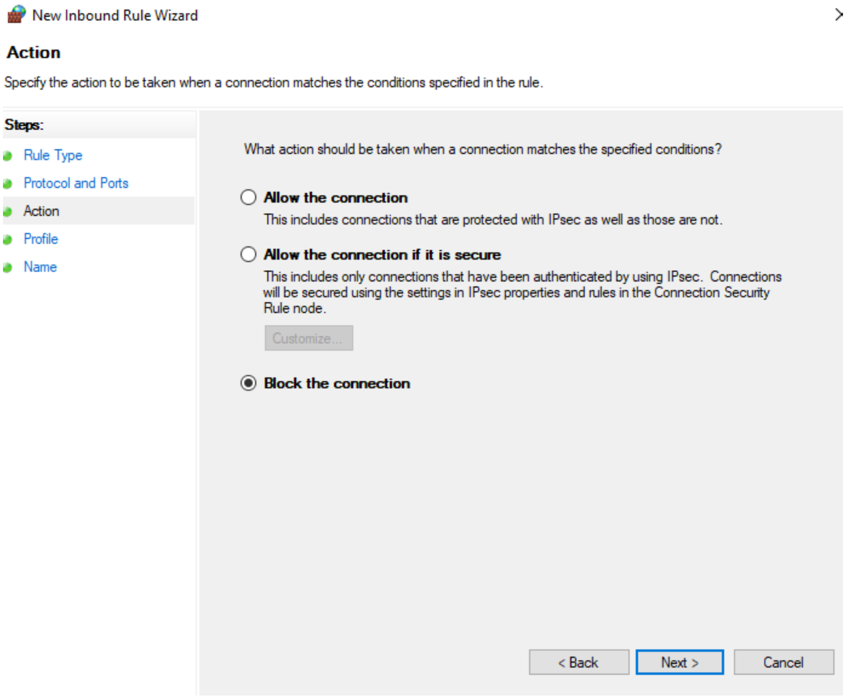
Then click on ‘New Rule’



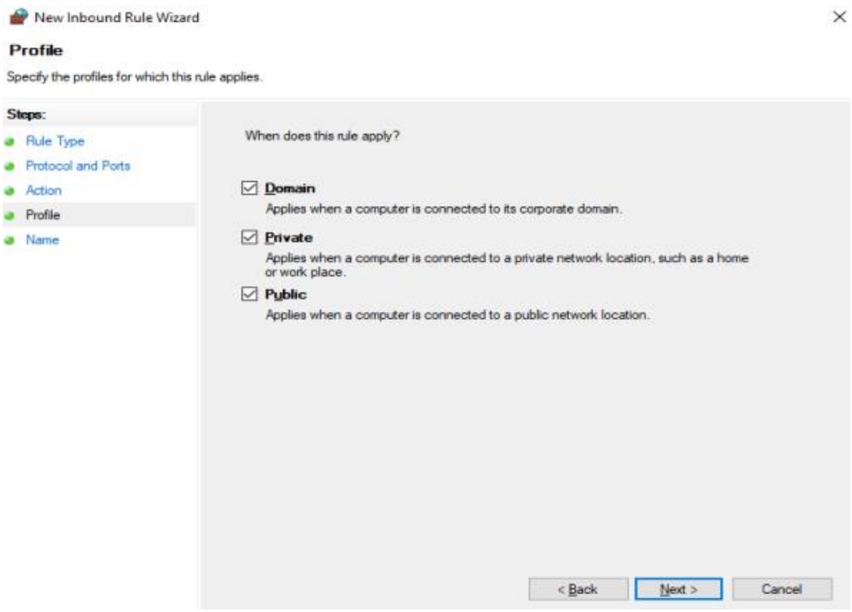
Select the radio button 'Port' and click 'Next' and enter the following



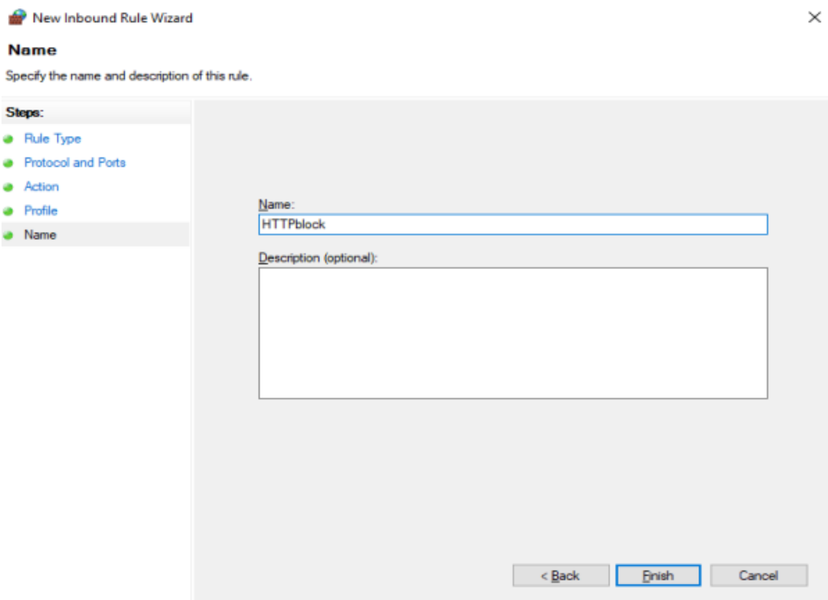
After next, we need to finalize the rule



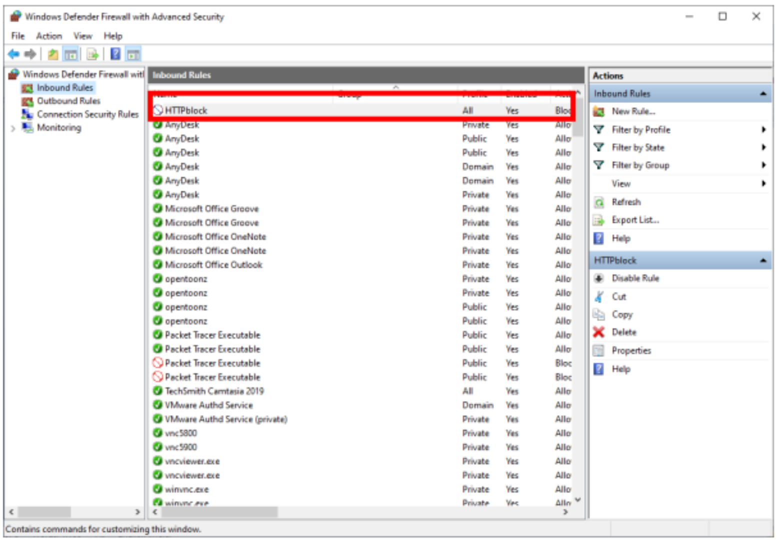
Click ‘Next’ and we get the following



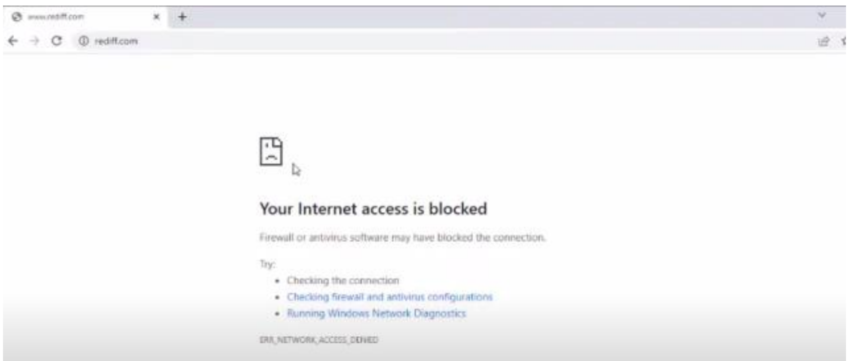
After clicking the ‘Next’ button we need to name the rule and click finish



The Inbound rule is added

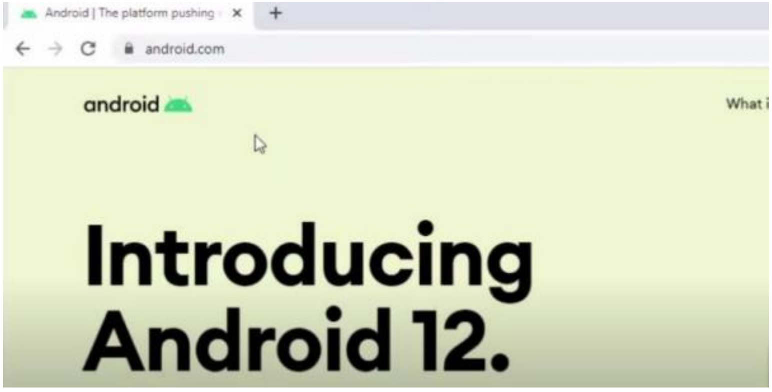


We repeat all the above steps for creating 'Outbound Rules', and then try to access the internet. We see that the accessed is blocked

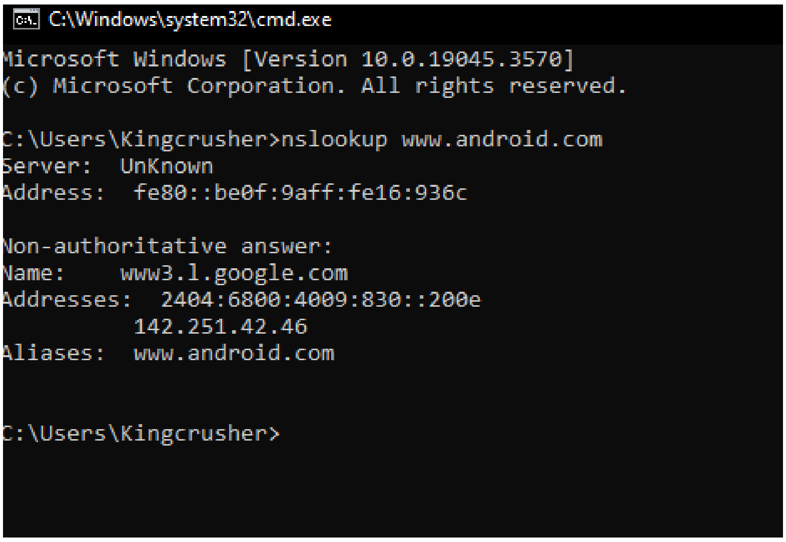


Part 2: Blocking the website www.android.com

We open the browser and access the website, which is now accessible



We find the IP addresses of the website using the following command

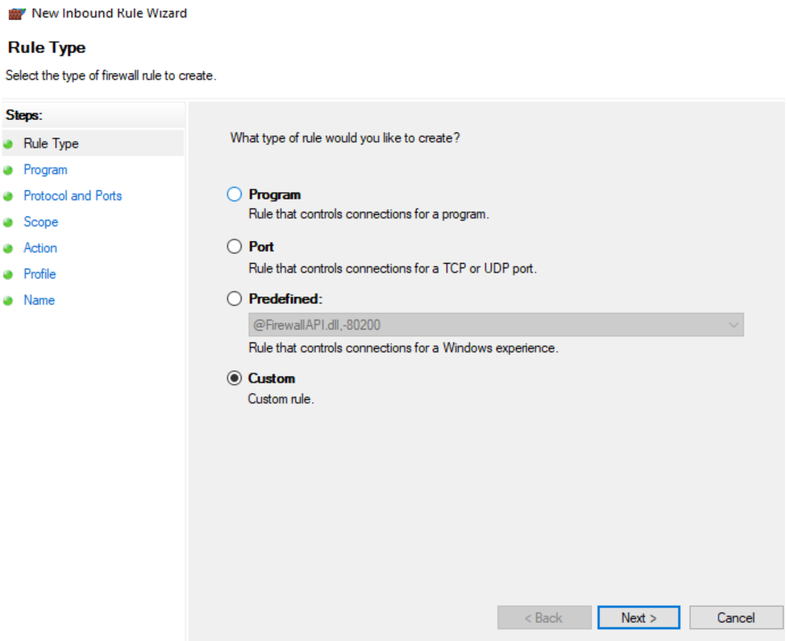


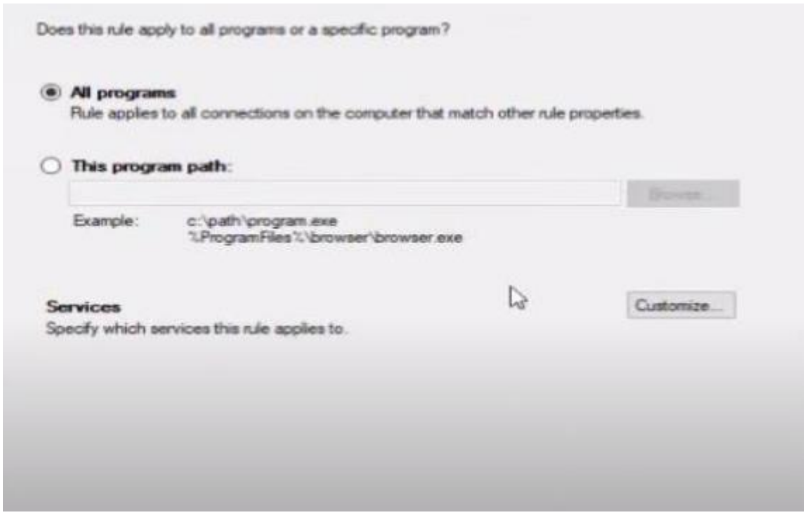
We save the IP addresses

IPv4 142.251.42.46

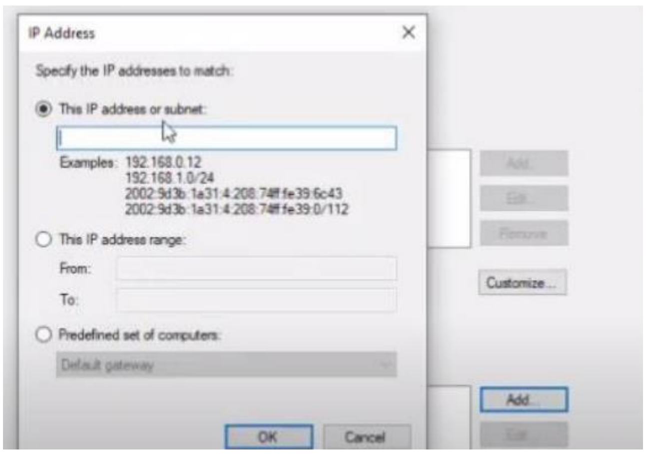
IPv6 2404:6800:4009:830::200e

We open the windows Firewall settings and apply the Inbound Rule

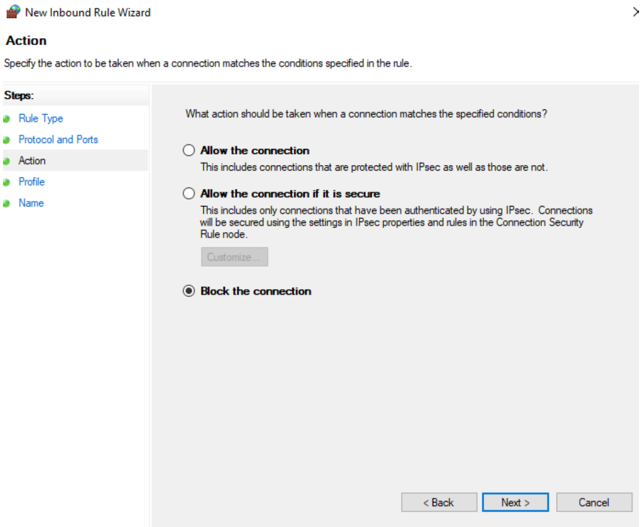




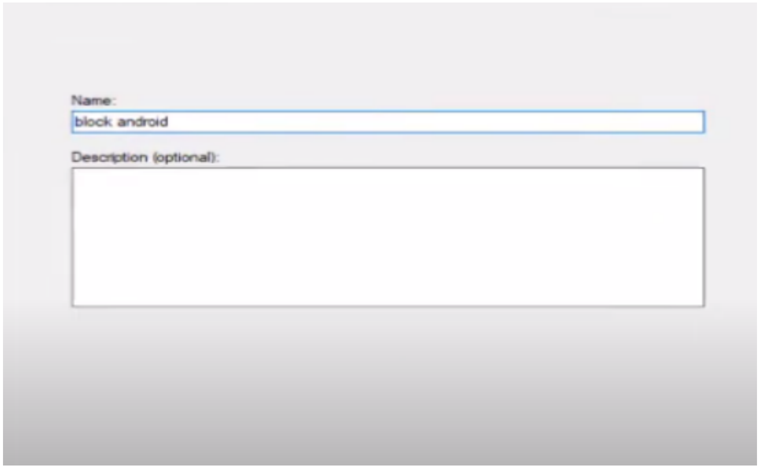
Insert the IP addresses both IPv4 and IPv6



Select Block connection

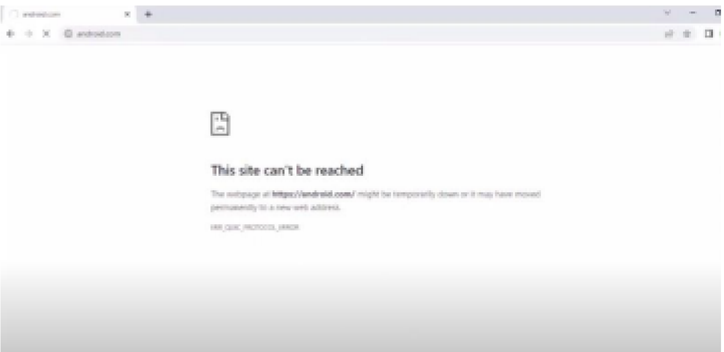


Provide a suitable name and finish



Repeat the above for Outbound Rules

Now if we try to access the website www.android.com , it would be blocked



Practical No. 9: Web Security with Secure Socket Layer(SSL)

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

Code:

server.py

```
import socket
server_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
LOCALHOST = '127.0.0.1'
port = 9990
server_socket.bind((LOCALHOST,port))
server_socket.listen()
print("Server started...")
client_sockets,addr=server_socket.accept()
while True:
    msg_received = client_sockets.recv(1024)
    msg_received = msg_received.decode()
    print("Client:", msg_received)
    msg_send = input("Me:")
    client_sockets.send(msg_send.encode("ascii"))
    if msg_send=="exit":
        break
client_sockets.close()
```

client.py

```
import socket
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
LOCALHOST = '127.0.0.1'
port = 9990
s.connect((LOCALHOST,port))
```

```python
print("New client created:")

while True:

    client_message = input("Me: ")

    s.send(client_message.encode())

    msg_received = s.recv(1024)

    msg_received = msg_received.decode()

    print("Server:",msg_received)

    if msg_received == 'exit':

        break

s.close()
```

OUTPUT:

server.py

```
C:\Windows\system32\cmd.exe

C:\Users\Tilak College\Desktop\ins>python server.py
Server started...
Client: Hi
Me:Hello
Client: How r u?
Me:fine
Client: Bye
Me:exit

C:\Users\Tilak College\Desktop\ins>
```

Client.py

```
C:\Windows\system32\cmd.exe

C:\Users\Tilak College\Desktop\ins>python client.py
New client created:
Me: Hi
Server: Hello
Me: How r u?
Server: fine
Me: Bye
Server: exit

C:\Users\Tilak College\Desktop\ins>
```