INS Journal
TYCS

Maharashtra College
Prof. Ismail H. Popatia

Index

Implementing Substitution and Transposition Ciphers

Aim: To study and implement the Substitution and Transposition Ciphers

Theory:

Substitution Cipher:
The Substitution Cipher is one of the simplest and oldest methods of encrypting messages. It falls under the category of symmetric key encryption, meaning the same key is used for both encryption and decryption. In a Substitution Cipher, each letter in the plaintext (original message) is replaced by another letter or symbol to create the ciphertext (encrypted message). This method is called substitution because each letter is substituted with another according to a predetermined rule.

Caesar Cipher:
One of the most famous examples of a Substitution Cipher is the Caesar Cipher, named after Julius Caesar, who is believed to have used this method to protect his confidential correspondence. The Caesar Cipher involves shifting each letter in the plaintext by a fixed number of positions in the alphabet.

For example, with a shift of 3, the letter 'A' is substituted with 'D', 'B' with 'E', 'C' with 'F', and so on. This process wraps around the alphabet, so 'X' becomes 'A', 'Y' becomes 'B', and 'Z' becomes 'C'. The shift value is often referred to as the key, and it determines the mapping from plaintext to ciphertext.

Encryption Process:
To encrypt a message using the Caesar Cipher, follow these steps:

Choose a shift value (key) for the cipher.
Take the plaintext message and, for each letter:

a) Determine its position in the alphabet.
b) Shift the position by the key value.
c) Map the new position back to a letter in the alphabet.
d) Replace the original letter with the mapped letter to obtain the ciphertext.

For example, with a shift of 3, the plaintext "HELLO" would become "KHOOR" in ciphertext.

Decryption Process:
To decrypt a message encrypted with the Caesar Cipher, the recipient needs to know the shift value (key) that was used. The decryption process is the reverse of the encryption process:

Obtain the ciphertext message.
For each letter:

a) Determine its position in the alphabet.
b) Shift the position back by the key value (subtract the key).
c) Map the new position back to a letter in the alphabet.
d) Replace the original letter with the mapped letter to obtain the plaintext.

Using the same shift of 3, the ciphertext "KHOOR" would be decrypted as "HELLO".

Transposition Cipher:
The Transposition Cipher is another type of encryption method that operates by rearranging the characters or blocks of characters in the plaintext to form the ciphertext. 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 plaintext by writing them in a zigzag pattern along a set number of "rails." The rails are imaginary horizontal lines on which the plaintext characters are placed.

Encryption Process:
To encrypt a message using the Railfence Cipher, follow these steps:

a) Choose the number of rails (often referred to as the key) for the cipher.
b) Write the plaintext message diagonally along the rails from top to bottom and left to right.
c) Once the last rail is reached, reverse the direction and continue writing diagonally upwards until the first rail is reached again.
d) Read the characters in the zigzag pattern from left to right and from top to bottom to obtain the ciphertext

Decryption Process:
To decrypt a message encrypted with the Railfence Cipher, the recipient needs to know the number of rails (key) used during encryption. The decryption process is the reverse of the encryption process:

a) Write the ciphertext diagonally along the rails, just as it was done during encryption.
b) Read the characters from left to right and from top to bottom to obtain the plaintext.

Code:Python code for implementing Caesar Cipher

```
#A python program to illustrate Caesar Cipher Technique
def encrypt(text,s):
        result = ""

        # traverse text
        for i in range(len(text)):
                char = text[i]

                # Encrypt uppercase characters
                if (char.isupper()):
                        result += chr((ord(char) + s-65) % 26 + 65)

                # Encrypt lowercase characters
                else:
                        result += chr((ord(char) + s - 97) % 26 + 97)

        return result

#check the above function
text=input(" Enter the text to encrypt ")
s = 3
print("Text : " + text)
str(s)
print( "Cipher: " + encrypt(text,s))
```

Code: Python code for implementing Railfence Cipher

```
string=input("enter a string")
def RailFence(txt):
   result=""
   for i in range(len(string)):
      if(i%2==0):
         result+=string[i]
   for i in range(len(string)):
      if(i%2!=0):
         result += string[i]
   return result
print(RailFence(string))
```

Code: Java code for implementing Railfence Cipher

```
public class railfence {
public static void main(String args[])
    {
        String input = "ismile";
        String output = "";
intlen = input.length(),flag = 0;

System.out.println("Input String : " + input);
for(int i=0;i<len;i+=2) {

output += input.charAt(i);
        }
for(int i=1;i<len;i+=2) {

output += input.charAt(i);
        }

System.out.println("Ciphered Text : "+output);
    }
}
```

Click on the following link for video demonstration of the given practical

1) https://youtu.be/1txPRg0qIbg (Railfence Cipher, JAVA)
2) https://youtu.be/W31HhqLQLRI (Railfence Cipher, Python)
3) https://youtu.be/zcGqbAWWv-Q (Caesar Cipher)

OR

Scan the QR-code for video demonstration of the Caesar Cipher practical

Scan the QR-code for video demonstration of the Railfence practical Using JAVA

Scan the QR-code for video demonstration of the Railfence practical Using Python

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. It is based on the mathematical problem of factoring large prime numbers, which is computationally difficult and forms the foundation of RSA's security.

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, ϕ(N), where ϕ(N) = (p - 1) * (q - 1).
d) Choose an integer, e, such that 1 < e <ϕ(N) and e is coprime with ϕ(N). This means that e and ϕ(N) should have no common factors other than 1.
e) Find the modular multiplicative inverse of e modulo ϕ(N), denoted as d. In other words, d is an integer such that (d * e) % ϕ(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.

Encryption Process:

To encrypt a message using RSA encryption, follow these steps:

a) Obtain the recipient's public key, which includes the modulus, N, and the public exponent, e.
b) Represent the plaintext message as an integer, M, where 0 ≤ M < N.
c) Compute the ciphertext, C, using the encryption formula: $C = M^e \bmod N$.

Decryption Process:

To decrypt a message encrypted with RSA encryption, the recipient uses their private key. Follow these steps:

a) Obtain the recipient's private key, which includes modulus, N, and the private exponent, d.
b) Receive the ciphertext, C.
c) Compute the plaintext, M, using the decryption formula: $M = C^d \bmod N$.

Security Considerations:

RSA encryption relies on the difficulty of factoring large prime numbers. The security of RSA is based on the assumption that factoring large numbers is computationally infeasible within a reasonable time frame. Breaking RSA encryption requires factoring the modulus, N, into its constituent prime factors, which becomes exponentially more difficult as N grows larger.

To ensure the security of RSA, it is essential to use sufficiently large prime numbers for key generation and to protect the private key from unauthorized access.

Code: Python code for implementing RSA Algorithm

```
fromCrypto.PublicKey import RSA
fromCrypto.Cipher import PKCS1_OAEP
importbinascii

keyPair = RSA.generate(1024)

pubKey = keyPair.publickey()
print(f"Public key:  (n={hex(pubKey.n)}, e={hex(pubKey.e)})")
pubKeyPEM = pubKey.exportKey()
print(pubKeyPEM.decode('ascii'))

print(f"Private key: (n={hex(pubKey.n)}, d={hex(keyPair.d)})")
privKeyPEM = keyPair.exportKey()
print(privKeyPEM.decode('ascii'))

#encryption
msg = 'Ismile Academy'
encryptor = PKCS1_OAEP.new(pubKey)
encrypted = encryptor.encrypt(msg)
print("Encrypted:", binascii.hexlify(encrypted))
```

Click on the following link for video demonstration of the given practical

https://youtu.be/b9RYoJ4y9Cs

OR

Scan the QR-code for video demonstration of the RSA Algorithm

Message Authentication Codes (MAC)

Aim: To study and implement the Message Authentication Code for ensuring the message intergrity 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. The receiver can verify the integrity and authenticity of the message by recomputing the MAC using the same key and hash function and comparing it with the received MAC.

MAC can be implement using various algorithms, we consider MD5 and SHA1

MD5 Algorithm:
MD5 (Message Digest Algorithm 5) is a widely used cryptographic hash function. Although it has been widely used historically, it is now considered to have vulnerabilities and is not recommended for security-critical applications. Nonetheless, it serves as an educational example for understanding MAC and cryptographic hash functions.

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: ConcatenatedData = M || K (|| denotes concatenation).
c) Apply the MD5 algorithm to the ConcatenatedData to obtain the MAC: MAC = MD5(ConcatenatedData).
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: ConcatenatedData = 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.

Security Considerations:
MD5 is no longer considered secure for cryptographic purposes due to vulnerabilities that have been discovered. It is susceptible to collision attacks, where two different inputs produce the same hash value. Therefore, it is recommended to use stronger hash functions, such as SHA-256 or SHA-3, for MAC generation in real-world applications.

Additionally, the security of MAC relies on the confidentiality and integrity of the secret key. If an attacker gains access to the secret key, they can generate valid MACs and forge messages.

SHA1 (Secure Hash Algorithm):
SHA is a family of cryptographic hash functions designed by the National Security Agency (NSA) in the United States. It provides secure one-way hashing and is widely used for various security applications. Examples include SHA-256 and SHA-3, which are stronger and more secure than MD5 or SHA-1.

MAC Generation Process:
To generate a MAC using the SHA algorithm, such as SHA-256, 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: ConcatenatedData = M || K (|| denotes concatenation).
c) Apply the SHA algorithm (e.g., SHA-256) to the ConcatenatedData to obtain the MAC: MAC = SHA-256(ConcatenatedData).
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: ConcatenatedData = M || K.
c) Apply the SHA algorithm (e.g., SHA-256) to the ConcatenatedData to compute the recalculated MAC: RecalculatedMAC = SHA-256(ConcatenatedData).
d) Compare the RecalculatedMAC with the received MAC. If they match, the message is considered authentic and intact.

Security Considerations:
The security of MAC relies on the confidentiality and integrity of the secret key. If an attacker gains access to the secret key, they can generate valid MACs and forge messages. Therefore, it is crucial to employ strong key management practices to protect the secret key.

Additionally, the security of the MAC depends on the security of the underlying hash function. Strong hash functions like SHA-256 are designed to resist collision attacks and other cryptographic vulnerabilities.

Code: Python code for implementing MD5 Algorithm

```
importhashlib
result = hashlib.md5(b'Ismile')
result1 = hashlib.md5(b'Esmile')
# printing the equivalent byte value.
print("The byte equivalent of hash is : ", end ="")
print(result.digest())
print("The byte equivalent of hash is : ", end ="")
print(result1.digest())
```

Code: Python code for implementing SHA Algorithm

```
importhashlib
str = input(" Enter the value to encode ")
result = hashlib.sha1(str.encode())
print("The hexadecima equivalent if SHA1 is :  ")
print(result.hexdigest())
```

Click on the following link for video demonstration of the given practical

https://youtu.be/SYe3sNQydIg (MD5)
https://youtu.be/rWO6ajImQIg (SHA)

OR

Scan the QR-code for video demonstration of the MD5 algorithm

Scan the QR-code for video demonstration of the SHA algorithm

Digital Signatures

Aim: To study and implement the Digital Signature algorithm

Theory:

Digital Signature:

Digital signatures provide a means of ensuring message integrity and authenticity in secure communication. 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. It allows the recipient to verify the integrity of the message and authenticate the signer's identity.

RSA Algorithm:

RSA (Rivest-Shamir-Adleman) is an asymmetric encryption algorithm widely used for secure communication. It is based on the mathematical problem of factoring large prime numbers, which is computationally difficult. RSA consists of a key pair: a public key for encryption and a private key for decryption and digital signing.

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.

Security Considerations:

The security of digital signatures relies on the following considerations:

1. Key Management: The private key used for generating the digital signature must be kept confidential and securely stored. Unauthorized access to the private key could compromise the security of the digital signature.
2. Hash Function Security: The choice of a secure cryptographic hash function is critical for ensuring the integrity of the message and preventing hash function vulnerabilities.
3. Key Length: The security of RSA is directly related to the key length used. Longer key lengths offer higher security against brute-force attacks.
4. Certificate Authorities: In real-world scenarios, digital signatures are often used with X.509 certificates issued by trusted certificate authorities (CAs). CAs validate the identity of the signer and bind it to the public key, providing a trusted mechanism for digital signature verification.

Digital signatures, based on asymmetric encryption algorithms like RSA, provide a powerful mechanism for ensuring message integrity and authenticity in secure communication. Understanding the principles of digital signatures, including the key generation process and verification steps, is crucial for undergraduate students studying practical cryptography. Additionally, awareness of key management, hash function security, and the role of trusted certificate authorities enhances the understanding of real-world digital signature implementations.

Code: Python code for implementing SHA Algorithm

```
fromCrypto.Signature import PKCS1_v1_5
fromCrypto.Hash import SHA256
fromCrypto.PublicKey import RSA
from Crypto import Random

defgenerate_signature(private_key, message):
   # Load the private key
key = RSA.importKey(private_key)

   # Generate SHA-256 hash of the message
hashed_message = SHA256.new(message.encode('utf-8'))

   # Create a signature using the private key
signer = PKCS1_v1_5.new(key)
signature = signer.sign(hashed_message)

return signature

defverify_signature(public_key, message, signature):
   # Load the public key
key = RSA.importKey(public_key)

   # Generate SHA-256 hash of the message
hashed_message = SHA256.new(message.encode('utf-8'))

   # Verify the signature using the public key
```

```
verifier = PKCS1_v1_5.new(key)
returnverifier.verify(hashed_message, signature)


# Generate RSA key pair
random_generator = Random.new().read
key_pair = RSA.generate(2048, random_generator)

# Extract public and private keys
public_key = key_pair.publickey().export_key()
private_key = key_pair.export_key()

# Example usage
message = "Hello, World!"

# Generate a digital signature
signature = generate_signature(private_key, message)
print("Generated Signature:", signature)

# Verify the digital signature
is_valid = verify_signature(public_key, message, signature)
print("Signature Verification Result:", is_valid)
```

Click on the following link for video demonstration of the given practical

https://youtu.be/zggIfS5sNtg

OR

Scan the following QR-code for

Key Exchange using Diffe-Hellman

Aim: To study and implement the Diffe-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. The shared key can then be used for symmetric encryption to ensure confidentiality, integrity, and authenticity of the communication. One widely used key exchange algorithm is the Diffie-Hellman algorithm.

Key Exchange Techniques:
Key exchange techniques enable secure key establishment between two parties. There are two main types of key exchange techniques:

1) Symmetric Key Exchange
2) Asymmetric Key Exchange

Symmetric Key Exchange:
In symmetric key exchange, both parties share a pre-established secret key. This key is typically distributed using a secure out-of-band method. Once the secret key is shared, it can be used for secure communication. However, this approach requires prior key sharing and becomes impractical for scenarios where a large number of participants need to securely communicate

Asymmetric Key Exchange:
Asymmetric key exchange, also known as public key exchange, overcomes the limitations of symmetric key exchange by using asymmetric encryption algorithms. It allows two parties who have never communicated before to establish a shared secret key without any prior key sharing. Asymmetric key exchange is based on the concept of public and private key pairs, where the public key is widely known and the private key is kept secret.

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.

Key Generation:

a) Alice calculates the shared secret key using Bob's public key:
b) Secret Key: $K = B^a \bmod p$
c) Bob calculates the shared secret key using Alice's public key:
d) Secret Key: $K = A^b \bmod p$

Key Agreement:

Both Alice and Bob have calculated the same shared secret key, K, independently.
They can now use K as the shared secret key for symmetric encryption algorithms to ensure secure communication.

Security Considerations:

The security of the Diffie-Hellman algorithm relies on the following considerations:

1) Large Prime Numbers: The security of the algorithm is based on the difficulty of the discrete logarithm problem. Using large prime numbers ensures the security of the shared secret key.
2) Public Key Distribution: The public keys exchanged during the key exchange process should be authenticated to prevent man-in-the-middle attacks. Techniques like digital signatures or certificate authorities can be used for authentication.
3) Key Length: Longer key lengths provide stronger security against brute-force attacks. It is important to use an appropriate key length for the prime number to ensure the desired security level.

Conclusion:

Key exchange techniques play a crucial role in establishing secure communication channels between parties. The Diffie-Hellman algorithm, as an example of asymmetric key exchange, allows two parties to securely establish a shared secret key over an insecure channel. Understanding the principles of key exchange, including the Diffie-Hellman algorithm and its security considerations, is essential for undergraduate students studying practical cryptography..

Code: Python code for implementing Diffie-Hellman Algorithm

```
from random import randint
if __name__ == '__main__':

   P = 23
   G = 9

print('The Value of P is :%d'%(P))
print('The Value of G is :%d'%(G))

   a = 4
print('Secret Number for Alice is :%d'%(a))

   x = int(pow(G,a,P))

   b = 6
print('Secret Number for Bob is :%d'%(b))

   y = int(pow(G,b,P))

ka = int(pow(y,a,P))

kb = int(pow(x,b,P))

print('Secret key for the Alice is : %d'%(ka))
print('Secret Key for the Bob is : %d'%(kb))
```

Click on the following link for video demonstration of the given practical

https://youtu.be/LYtbQiy7OzM

Scan the QR-code for video demonstration of the Diffie-Hellman Algorithm	

Click on the following link for a simple animated video to explain the working of Diffie-Hellman Algorithm

https://youtu.be/ozjG1CtP-1c

Scan the QR-code for animated video to explain the working of Diffie-Hellman Algorithm	

IP Security (IPSec) Configuration

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

Theory:

Some theoretical aspects of IPSec and the concept of an IPSec VPN tunnel:

1. IPSec Overview:
   - IPSec (Internet Protocol Security) is a comprehensive suite of protocols and standards used for securing communication over IP networks, such as the Internet.
   - It ensures the confidentiality, integrity, and authenticity of data transmitted between devices or networks.

2. Security Goals of IPSec:
   - Confidentiality: IPSec achieves data privacy through encryption.
   - Integrity: It guarantees that data remains unaltered during transit.
   - Authentication: IPSec verifies the identity of communicating parties to prevent unauthorized access and impersonation.

3. Components of IPSec:
   - IPSec comprises multiple protocols and elements, including Authentication Header (AH), Encapsulating Security Payload (ESP), Security Associations (SAs), and key management protocols.

4. IPSec VPN Tunnel:
   - An IPSec VPN tunnel is a secure, encrypted connection established between two endpoints or networks over the Internet or untrusted networks.
   - It is created using the IPSec suite to provide a secure and private channel for data transmission.

5. Establishing a VPN Tunnel:
   - The process begins with the negotiation and establishment of Security Associations (SAs) between the endpoints.
   - These SAs define parameters like encryption methods, authentication, and shared keys.

6. Modes of Operation:
   - VPN tunnels can operate in either Transport Mode (securing data payload) or Tunnel Mode (securing entire IP packets, including headers).
   - Transport Mode is often used for host-to-host communication, while Tunnel Mode is suitable for network-to-network connections.

7. Data Encryption and Authentication:
   - Data transmitted through the VPN tunnel is encrypted using algorithms specified in the SAs, ensuring data privacy.
   - Authentication and data integrity checks prevent tampering or unauthorized access.

8. Routing and Secure Communication:
   - Once established, the VPN tunnel allows secure data routing between the endpoints or networks.

- Applications and services on either side can communicate securely, even over untrusted networks like the Internet.

9. Use Cases:
   - IPSec VPN tunnels are used for various purposes, including remote access VPNs, site-to-site VPNs, secure data transfer, and protecting real-time communication like VoIP and video conferencing.

10. Key Management:
   - Secure key management is critical for the long-term security of IPSec VPN tunnels.
   - Keys can be generated manually or through automated key exchange protocols like Internet Key Exchange (IKE).

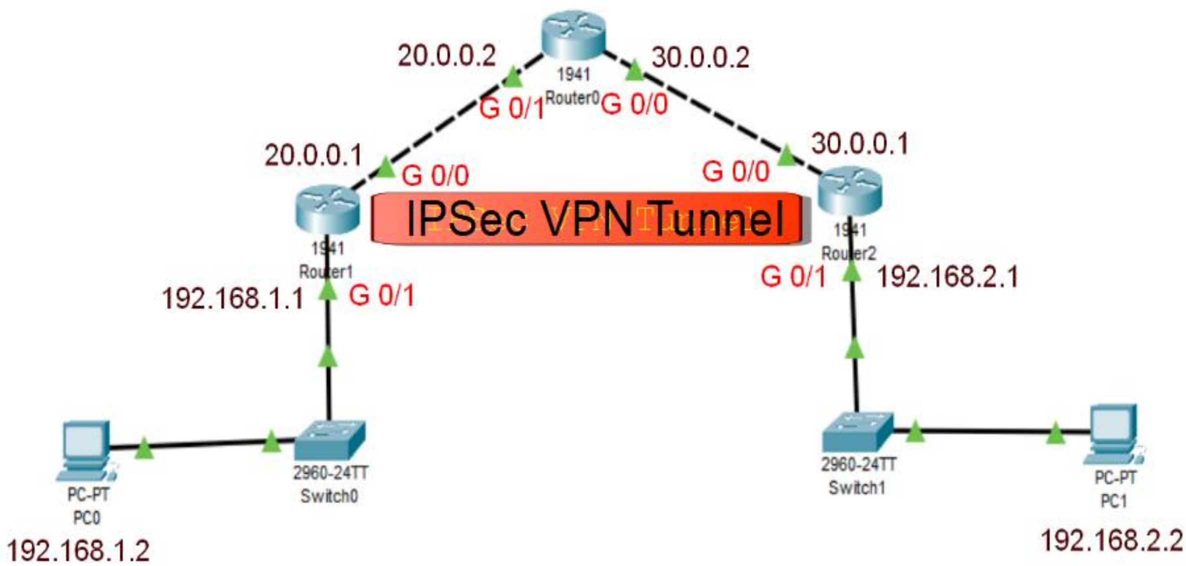
11. Security Policies:
   - Organizations define security policies that determine when and how IPSec should be applied to protect specific types of traffic or communication.

12. Interoperability:
   - IPSec is widely adopted, ensuring interoperability between different vendors' equipment and making it a versatile choice for securing networks and data.

Understanding the principles of IPSec and IPSec VPN tunnels is essential for designing, deploying, and managing secure communication in various network environments, ensuring data remains confidential, unaltered, and protected from unauthorized access.

Topology:

We use the following topology for the present case



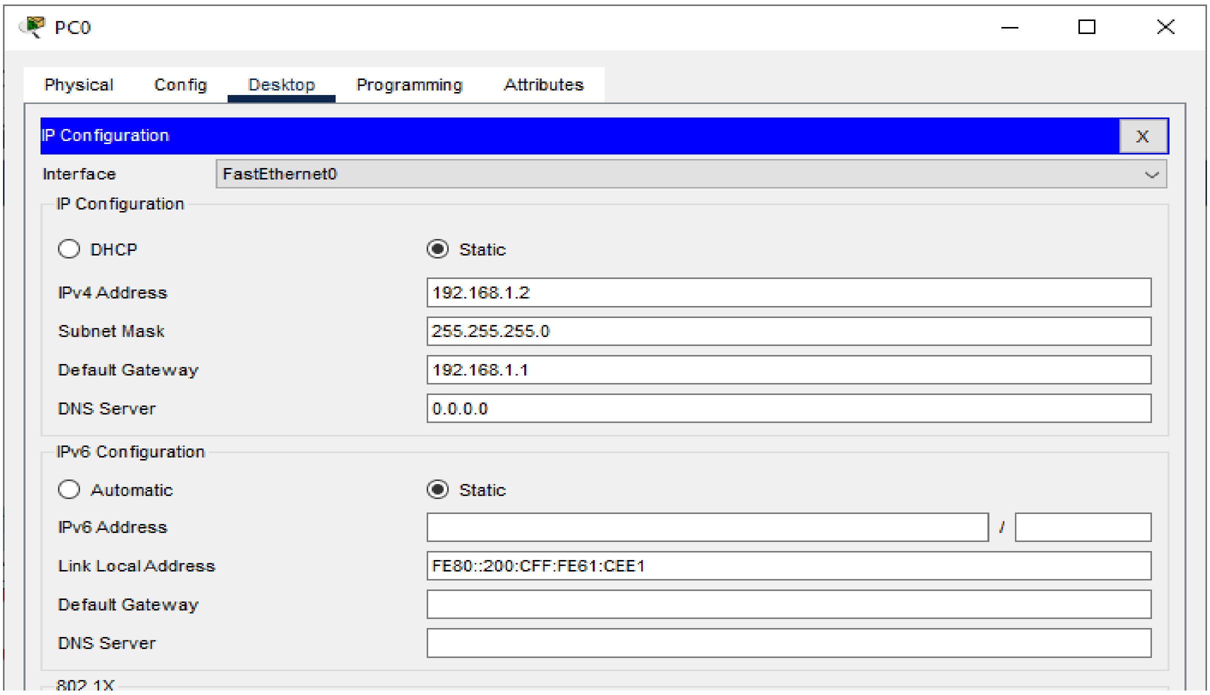
ISAKMP Policy Parameters

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

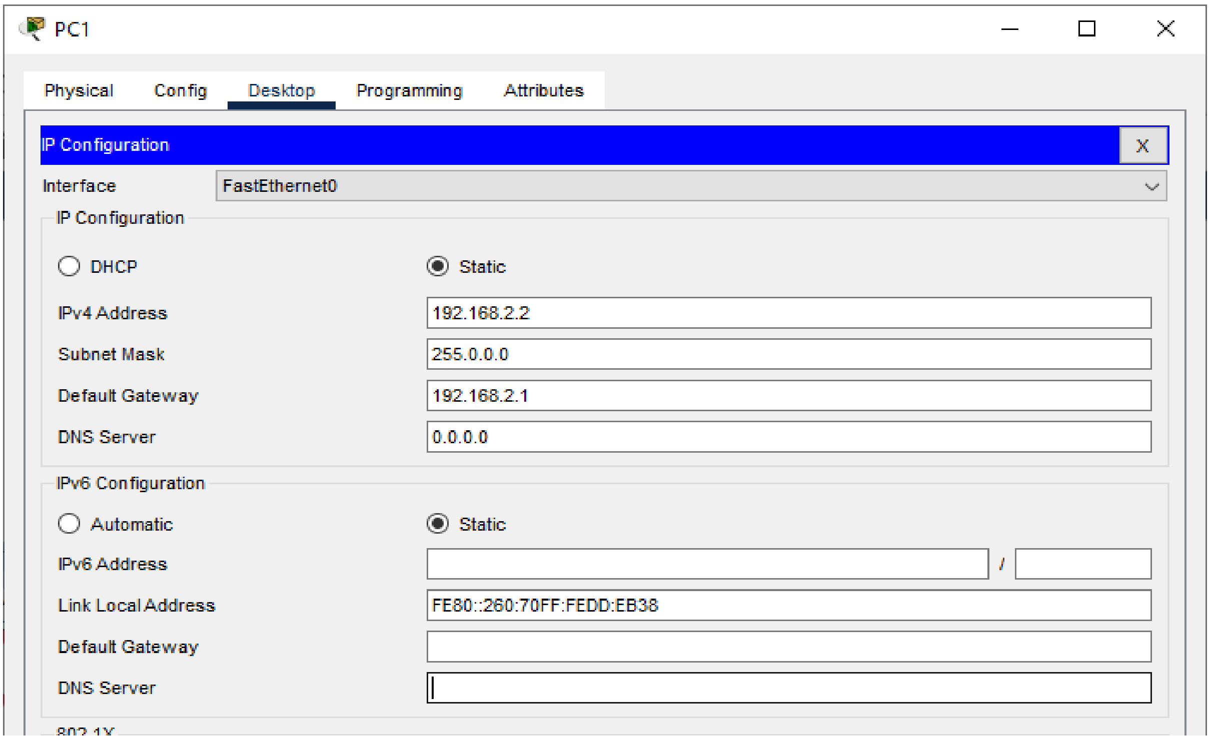
IPSec Policy Parameters

Parameters	R1	R2
Transform Set Name	VPN-SET	VPN-SET
ESP Transform Encryption	esp-aes	esp-aes
ESP Transform Authentication	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:

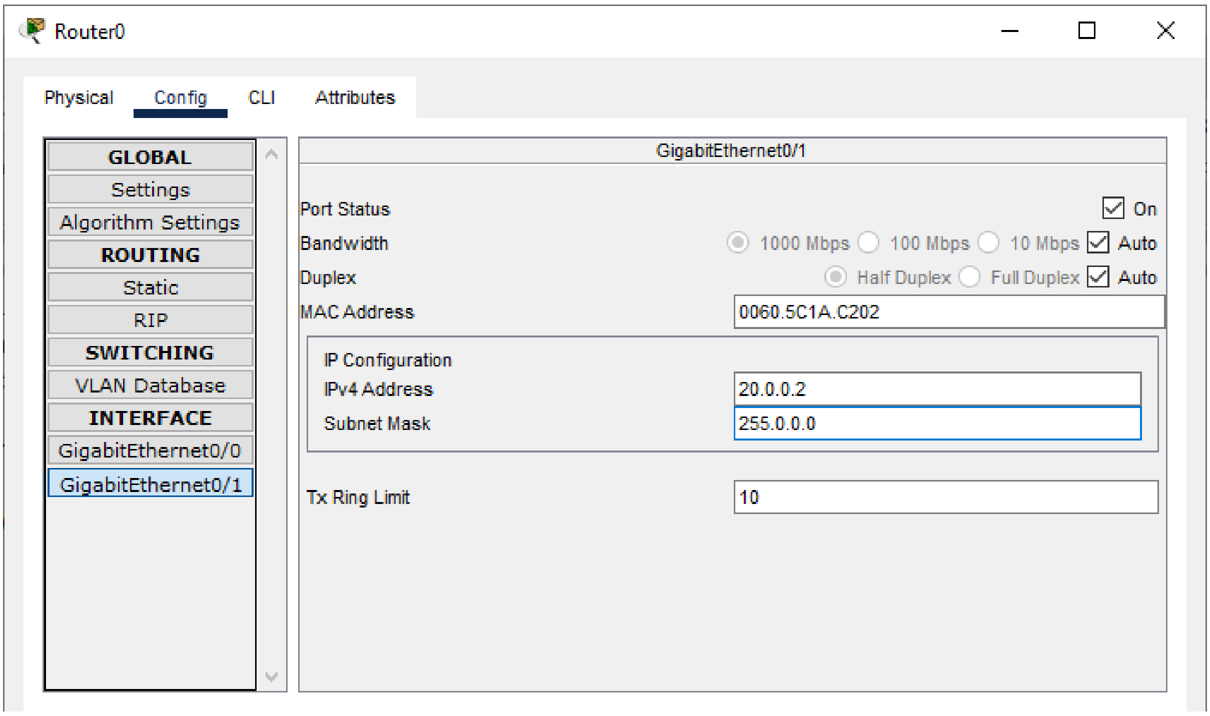


Configuring PC1:

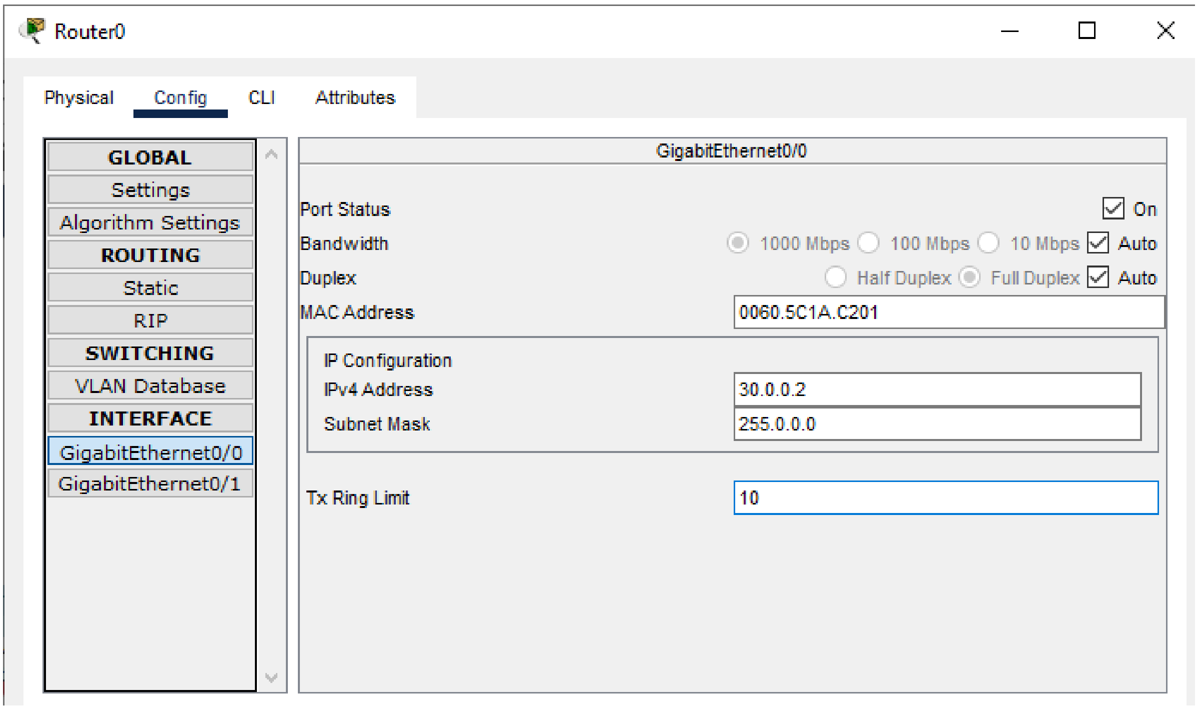


Configuring Router0:

Interface GigabitEthernet0/1:

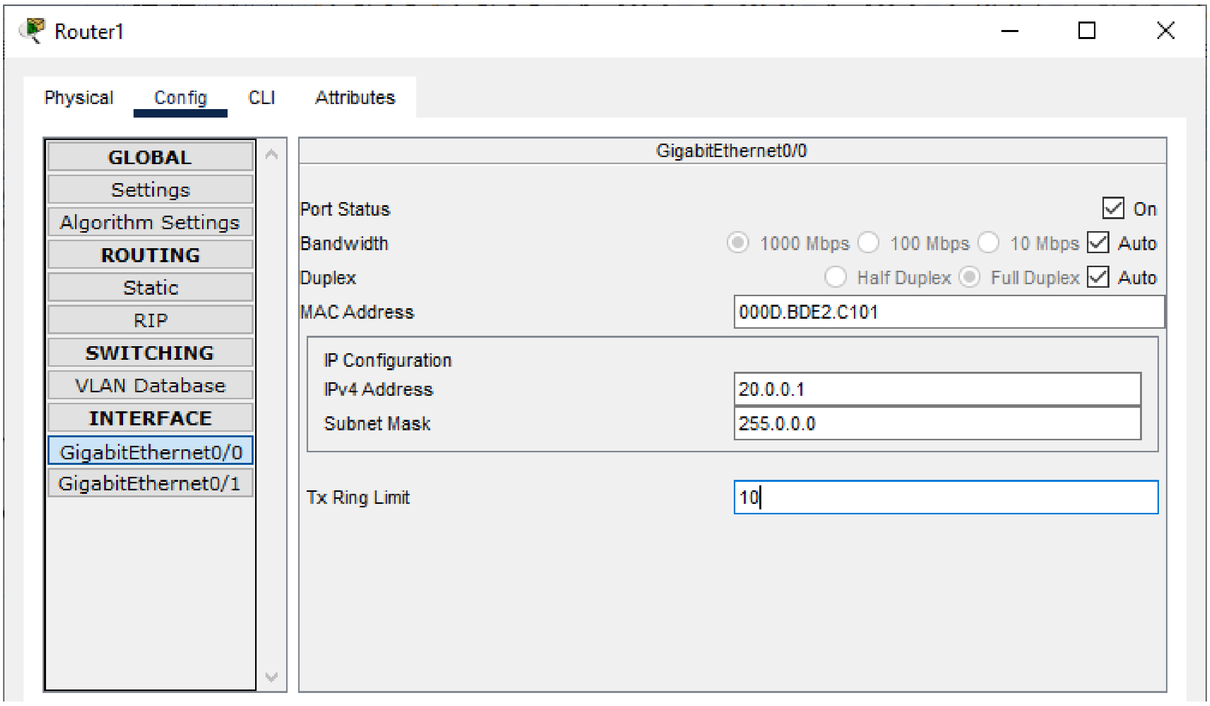


Interface GigabitEthernet0/0:

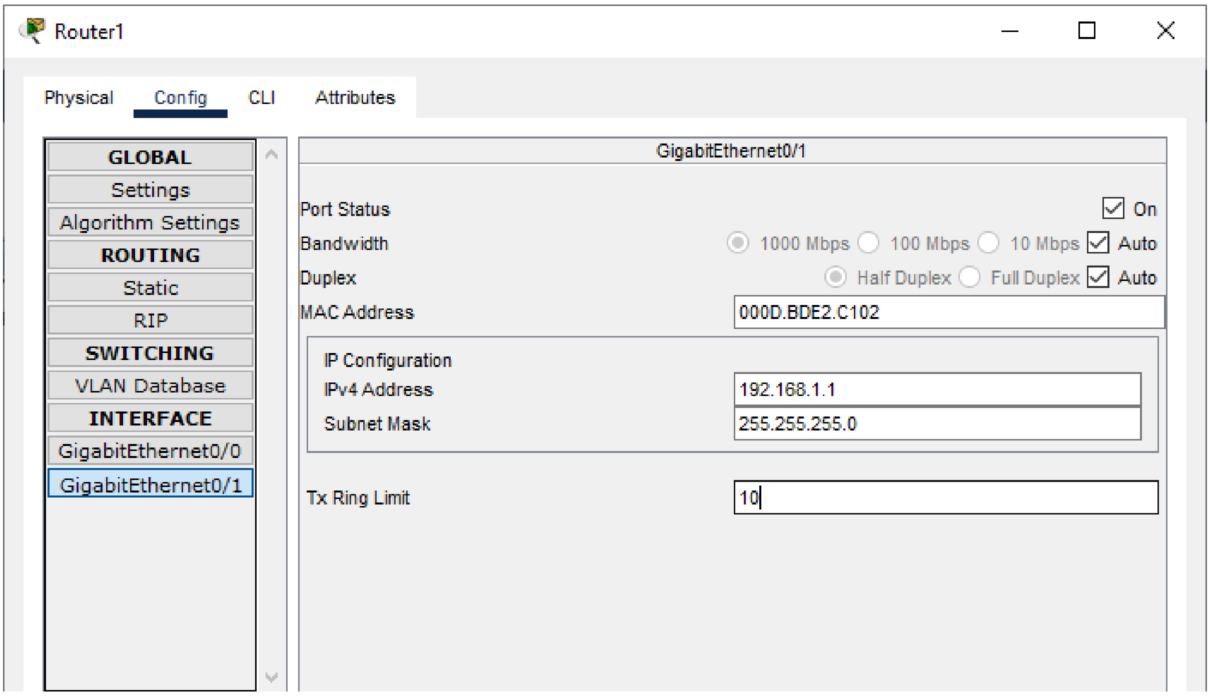


Configuring Router1:

Interface GigabitEthernet0/0:

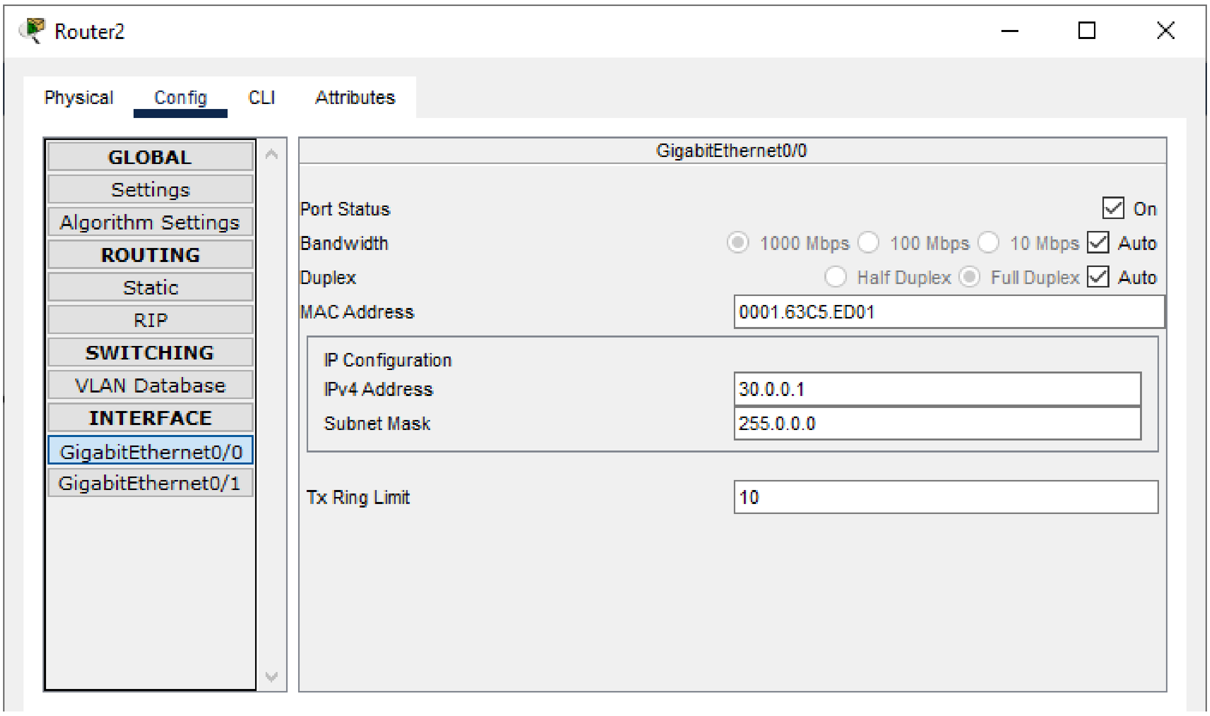


Interface GigabitEthernet0/1:

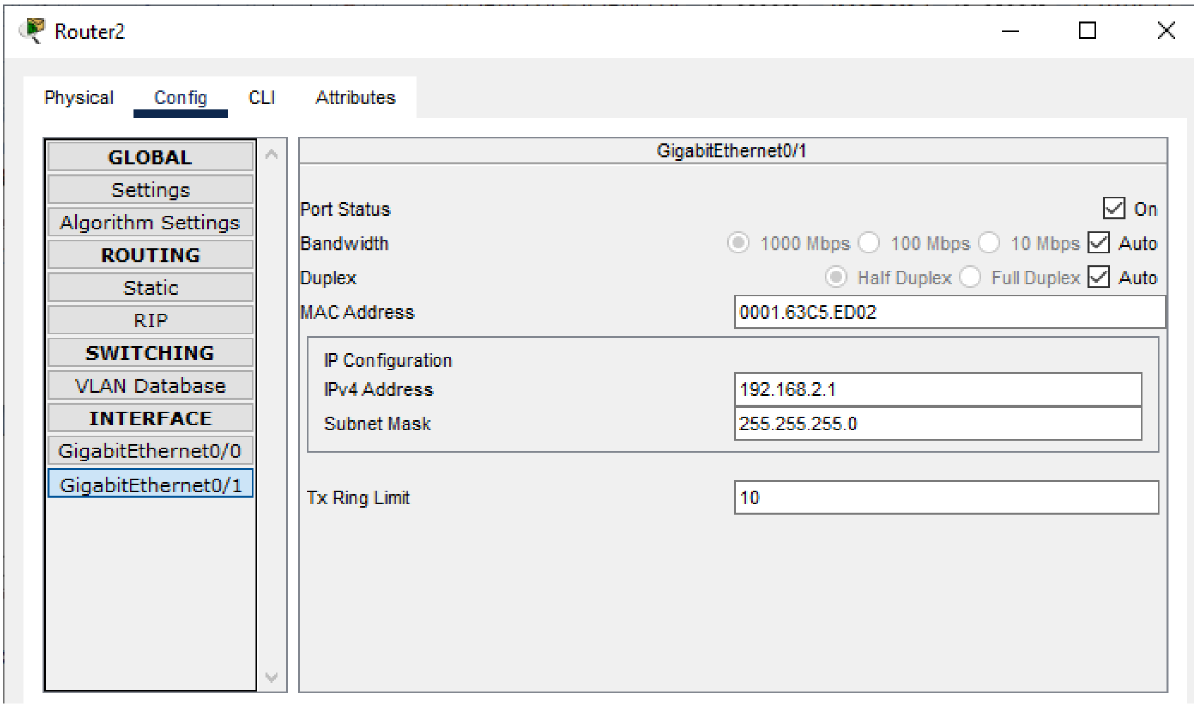


Configuring Router2:

Interface GigabitEthernet0/0:



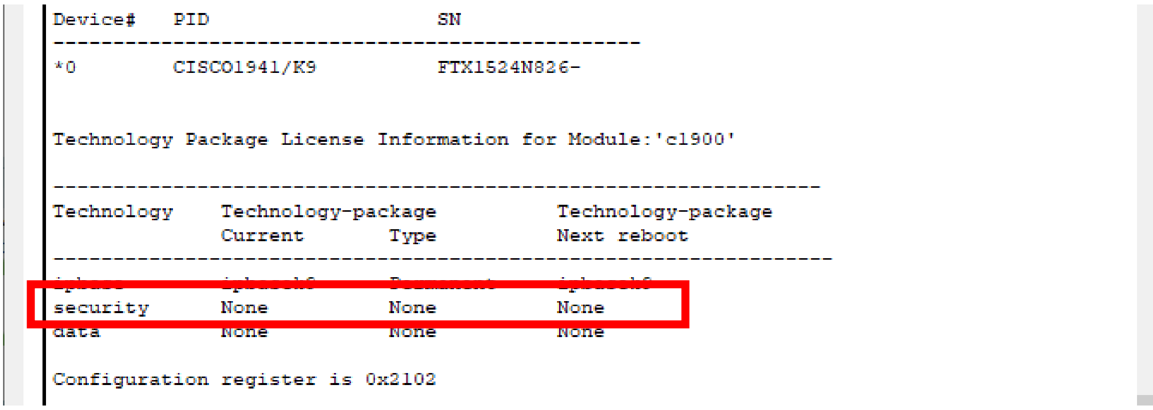
Interface GigabitEthernet0/1:



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



```
Device#   PID                   SN
-------------------------------------------------
*0        CISCO1941/K9          FTX1524N826-

Technology Package License Information for Module:'c1900'

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

Configuration register is 0x2102
```

(We see that the security feature is not enabled, hence we need to enable the security package

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

```
Technology Package License Information for Module:'c1900'

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

Configuration register is 0x2102
```

(The security package is enabled)

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

```
Device#   PID                   SN
-------------------------------------------------
*0        CISCO1941/K9          FTX1524N826-


Technology Package License Information for Module:'c1900'

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

Configuration register is 0x2102
```

(We see that the security feature is not enabled, hence we need to enable the security package

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

```
Technology Package License Information for Module:'c1900'

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

Configuration register is 0x2102
```

(The security package is enabled)

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)#

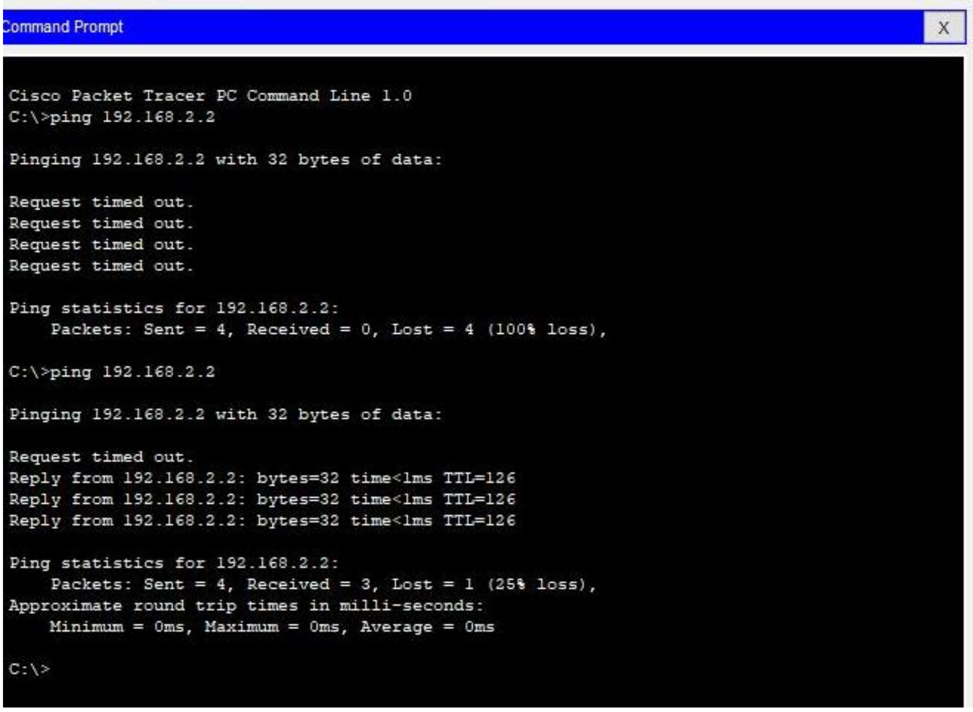
R1>enable
R1#configure terminal
R1(config)#crypto map IPSEC-MAP 10 ipsec-isakmp
R1(config-crypto-map)#set peer 30.0.0.1
R1(config-crypto-map)#set pfs group5
R1(config-crypto-map)#set security-association lifetime seconds 86400
R1(config-crypto-map)#set transform-set R1->R2

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



```
Command Prompt

Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:

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

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

C:\>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:

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

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>
```

Command Prompt

```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

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

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Reply from 192.168.1.2: bytes=32 time<1ms TTL=126
Reply from 192.168.1.2: bytes=32 time<1ms TTL=126
Reply from 192.168.1.2: bytes=32 time<1ms TTL=126
Reply from 192.168.1.2: bytes=32 time=1ms TTL=126

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

For the video demonstration of the above practical click on the link below or scan the QR-code

https://youtu.be/QHR6cbvB6X0

Malware Analysis and Detection

Aim: To do Detect and Analyse 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. Malware is created with malicious intent, often to steal sensitive information, gain control over systems, extort money, or cause harm to users or organizations. There are various types of malware, each with distinct characteristics and purposes. Some common types of malware include:

1. Viruses: Viruses are malicious programs that attach themselves to legitimate files or programs and spread by infecting other files. When infected files are executed, the virus replicates and spreads further, potentially causing damage to data and systems.

2. Worms: Worms are standalone programs that replicate and spread across computer networks without needing to attach themselves to other files. They often exploit security vulnerabilities to self-propagate and can cause network congestion and data loss.

3. Trojans: Trojans are deceptive programs that disguise themselves as legitimate software, tricking users into installing them. Once installed, Trojans can perform a variety of malicious activities, such as stealing sensitive information, opening backdoors, or launching attacks.

4. Ransomware: Ransomware encrypts a user's files or entire system and demands a ransom payment in exchange for providing the decryption key. It can lead to data loss and significant disruption if not properly managed.

5. Spyware: Spyware is designed to secretly gather information from a user's device, such as browsing habits, passwords, and personal data. This information is then sent to a remote attacker or entity.

6. Adware: Adware is software that displays unwanted advertisements, often in the form of pop-ups or banners, on a user's device. While not as malicious as other types of malware, it can be disruptive and invasive.

7. Keyloggers: Keyloggers record a user's keystrokes, allowing attackers to capture sensitive information like passwords, credit card numbers, and other confidential data.

8. Botnets: Botnets are networks of compromised computers or devices, known as "bots" or "zombies," controlled by a central attacker. Botnets are often used to launch coordinated attacks, distribute spam, or carry out other malicious activities.

9. Rootkits: Rootkits are designed to hide malicious activities from the user and security software. They can modify or replace core system files to gain unauthorized access and control over a system.

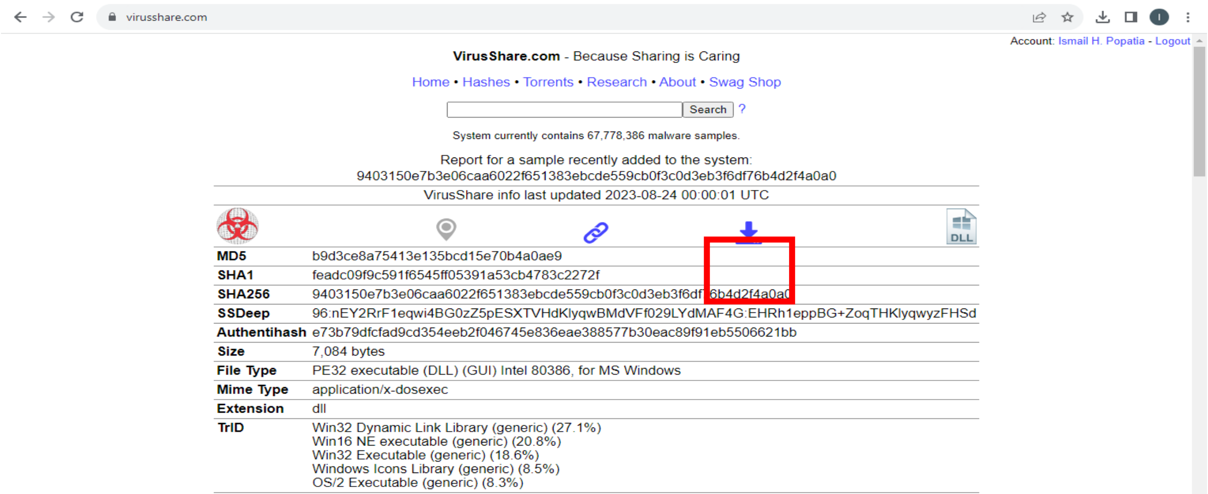
10. Backdoors: Backdoors provide unauthorized access to a compromised system. They can be used by attackers to maintain control over a system, often allowing them to return even after the initial breach is resolved.

Malware can enter systems through various vectors, including malicious email attachments, compromised websites, infected software downloads, and even through physical devices like infected USB drives. To protect against malware, it's essential to maintain strong cybersecurity practices, including using reputable antivirus and anti-malware software, keeping software up-to-date, avoiding suspicious links and downloads, and practicing safe browsing habits.

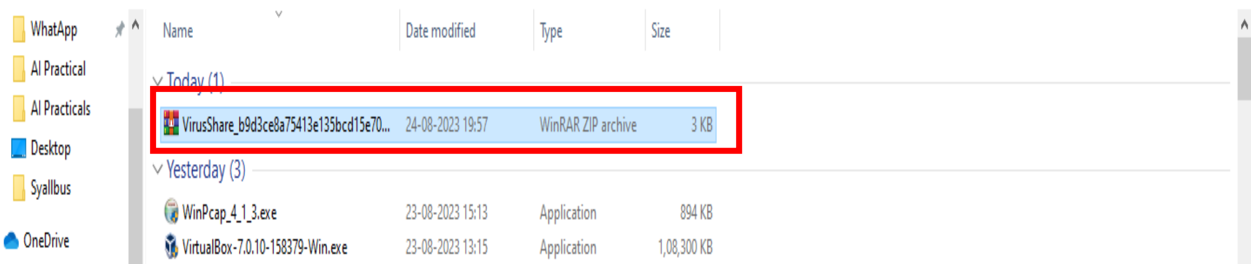
Analysis:

For analysing 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 authorised site)

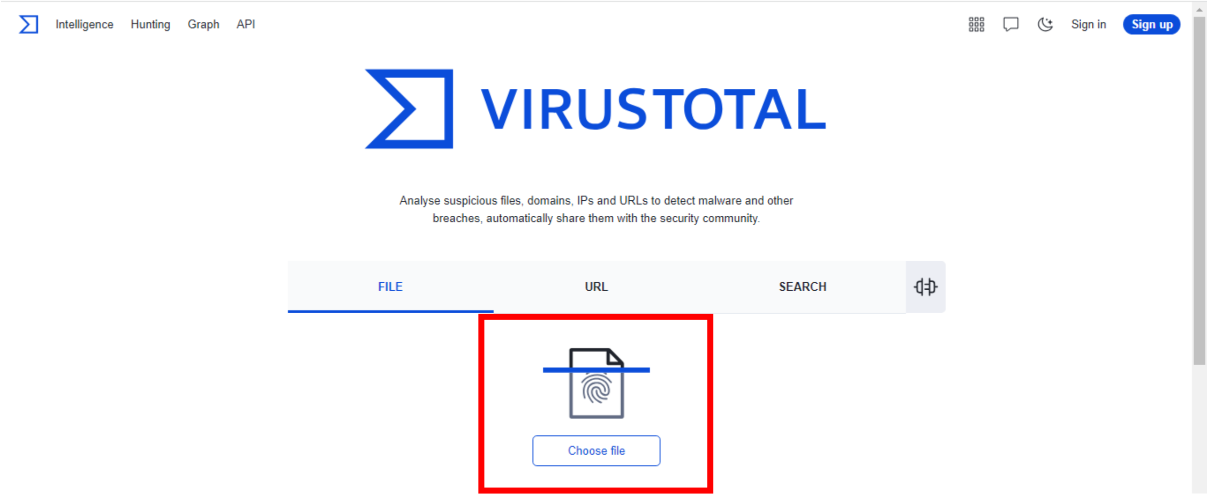


2) By clicking the above download icon the Malware gets downloaded in ZIP format.



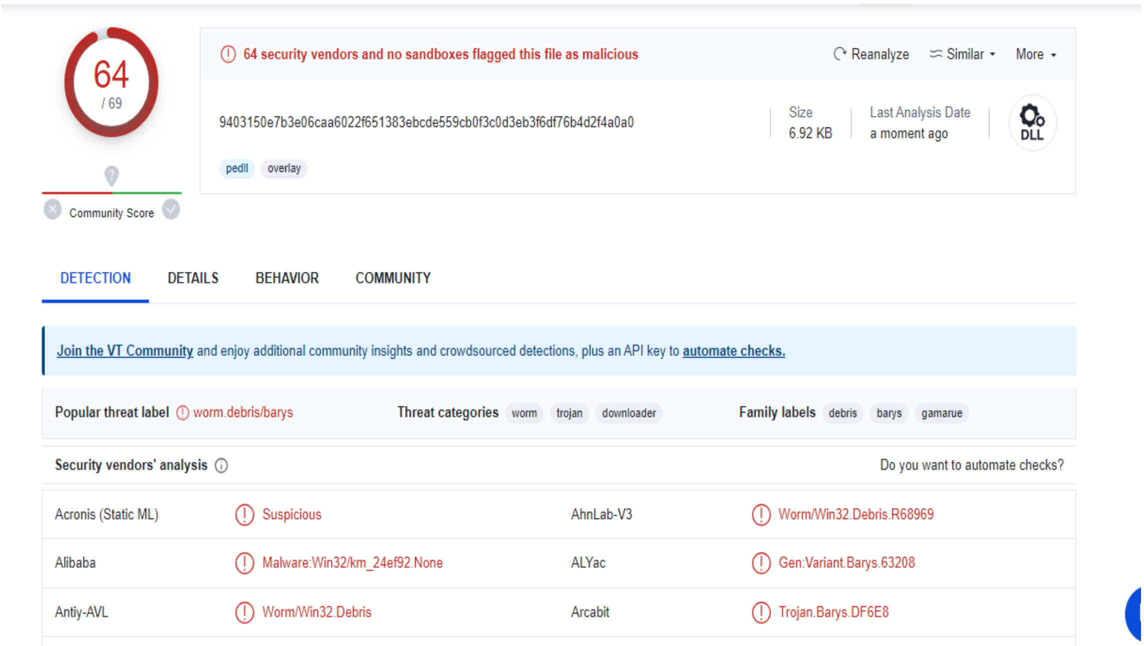
3) For unzip the password is “infected”, there is no need to unzip the file, we create a folder “Malware” on desktop and save the file in the folder

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) 64 security vendors out of 69 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?

Acronis (Static ML)	Suspicious	AhnLab-V3	Worm/Win32.Debris.R68969
Alibaba	Malware:Win32/km_24ef92.None	ALYac	Gen:Variant.Barys.63208
Antiy-AVL	Worm/Win32.Debris	Arcabit	Trojan.Barys.DF6E8
Avast	Win32:Debris-A [Wrm]	AVG	Win32:Debris-A [Wrm]
Avira (no cloud)	WORM/Debris.J.1	Baidu	Win32.Worm.Bundpil.an
BitDefender	Gen:Variant.Barys.63208	BitDefenderTheta	Gen:NN.ZedlaF.36350.aq5@aWbSzHn

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

Countermeasures:

Countermeasures are strategies, actions, or precautions taken to prevent or mitigate various risks, threats, or undesirable events. In the context of cyber-security and dealing with potential malware, viruses, and other online threats, here are some common countermeasures you can take:

1. Use Antivirus and Anti-Malware Software: Install reputable antivirus and anti-malware software on your devices. Keep the software updated to ensure you have the latest protection against known threats.

2. Keep Operating Systems and Software Updated: Regularly update your operating system, web browsers, plugins, and other software. Updates often include security patches that address vulnerabilities.

3. Use Strong and Unique Passwords: Use complex passwords that combine upper and lower case letters, numbers, and symbols. Avoid using common or easily guessable passwords. Consider using a password manager to securely store your passwords.

4. Enable Two-Factor Authentication (2FA): Whenever possible, enable two-factor authentication for your online accounts. This adds an extra layer of security by requiring a second form of verification in addition to your password.

5. Be Cautious with Email and Attachments: Be wary of unsolicited emails, especially those with attachments or links. Don't open attachments or click on links from unknown or suspicious sources. Verify the sender's authenticity before taking any action.

6. Use a Firewall: Enable firewalls on your devices and network. Firewalls help block unauthorized access and protect your system from external threats.

7. Regular Backups: Regularly back up your important data to an external source or a cloud storage service. In case of a malware attack or data loss, you'll have a copy of your important files.

9. Secure Wi-Fi Networks: Secure your home or office Wi-Fi network with a strong password and encryption. Avoid using public Wi-Fi networks for sensitive activities.

10. Use Ad-Blockers and Script Blockers: Install browser extensions that block ads and potentially malicious scripts. This can help prevent drive-by downloads and malvertising.

11. Disable Macros: Disable macros in office documents unless you're certain they are safe. Malicious macros are often used to deliver malware.

12. Download Software from Official Sources: Only download software from reputable and official sources. Be cautious of downloading software from unfamiliar websites.

13. Regularly Scan for Malware: Perform regular scans of your devices using reputable antivirus and anti-malware tools.

14. Use Virtual Private Networks (VPNs): When connecting to the internet, especially on public networks, use a VPN to encrypt your internet connection and enhance your privacy.

15. Implement Security Policies: If you're managing a network or a business, establish and enforce security policies for employees, including guidelines for safe browsing, email practices, and device usage.

For the video demonstration of the practical click on the link below or scan the QR-code

https://youtu.be/H2agWcwaqvM

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:

Firewalls are network security devices or software applications designed to monitor, filter, and control incoming and outgoing network traffic based on a set of predetermined security rules. The primary purpose of a firewall is to act as a barrier between a trusted internal network (such as a company's internal network) and untrusted external networks (like the internet), in order to prevent unauthorized access, data breaches, and other potential cyber threats.

Firewalls operate by inspecting network packets (small units of data) as they pass through the firewall and making decisions about whether to allow or block the traffic based on a set of predefined rules. These rules can be configured to specify which types of traffic are permitted and which are denied. Firewalls can be implemented in various locations within a network, including at the network perimeter, on individual devices, or even within cloud environments.

There are several types of firewalls, including:

1. Packet Filtering Firewalls: These examine network packets and decide whether to allow or block them based on criteria like source and destination IP addresses, port numbers, and protocols. While they are relatively simple, they lack the ability to inspect the actual content of the data packets.

2. Stateful Inspection Firewalls: Also known as dynamic packet filtering firewalls, these maintain a state table to keep track of active connections and only allow incoming traffic that matches an existing, legitimate connection. This approach is more secure than basic packet filtering.

3. Proxy Firewalls: Proxy firewalls act as intermediaries between the internal network and the external network. They receive requests from internal users, then initiate and manage connections to external resources on behalf of those users. This can provide an additional layer of security by hiding internal network details.

4. Application Layer Firewalls: These operate at the application layer of the OSI model and can understand the context of the traffic, such as the specific application or service being accessed. They are capable of making more fine-grained decisions based on the actual content of the traffic.

5. Next-Generation Firewalls (NGFW): NGFWs combine traditional firewall functionality with advanced features such as intrusion prevention, deep packet inspection, and application-aware filtering. They are designed to provide more comprehensive protection against modern threats.

Firewalls play a crucial role in network security by helping organizations establish a strong defence against unauthorized access, malware, and various cyber attacks. However, it's important to note that while firewalls are an essential component of a comprehensive cyber-security strategy, they are not a standalone solution. They should be used in conjunction with other security measures such as antivirus software, intrusion detection systems, regular software updates, and user training to ensure a robust defence against evolving threats.

Using a firewall to block unauthorized access is an important aspect of securing your network and systems. Firewalls act as barriers between your network and potential threats from the internet or other external sources. Here's a step-by-step guide on how to use a firewall to block unauthorized access:

1. Choose the Right Firewall:
There are hardware firewalls (devices that are placed between your network and the internet) and software firewalls (installed on individual computers or servers). Choose the type that best suits your needs.

2. Install and Configure the Firewall:
For software firewalls, install the firewall software on the computers or servers you want to protect. For hardware firewalls, follow the manufacturer's instructions to set it up between your network and the internet.

3. Define Security Policies:
Determine which types of traffic are allowed and which are denied. This can involve specifying rules that allow or block traffic based on criteria such as IP addresses, ports, and protocols.

4. Block Unauthorized Access:
Create rules to block access from unauthorized IP addresses or ranges. For example, you might block IP addresses known for malicious activities or any IP address that doesn't have a legitimate reason to access your network.

5. Allow Necessary Traffic:
Configure rules to allow access to the services and applications that are essential for your business operations. For instance, if you have a web server, you'll need to allow incoming traffic on port 80 or 443.

6. Regularly Update Rules:
Keep your firewall rules up to date. New threats can emerge, and you might need to adjust your rules accordingly.

7. Use Application Layer Filtering:
Modern firewalls can inspect traffic at the application layer, allowing you to block specific applications or services. This can help prevent unauthorized access to potentially risky services.

8. Intrusion Detection and Prevention Systems (IDPS):
Consider integrating an IDPS with your firewall. These systems can detect and prevent intrusion attempts by analyzing network traffic and patterns.

9. Logging and Monitoring:
Enable logging on your firewall to keep track of connection attempts and blocked traffic. Regularly review the logs to identify potential security issues.

10. Testing and Adjusting:
Regularly test your firewall's effectiveness by attempting to access your network from different scenarios. This can help you identify any gaps in your security and adjust your rules accordingly.

11. Keep Firewall Software Updated:
If you're using software firewalls, make sure to keep the firewall software up to date with the latest security patches.

12. Educate Users:
Even with a firewall in place, user awareness is essential. Educate your users about safe online practices, such as not clicking on suspicious links or downloading unknown files.

Remember that while firewalls are a valuable part of network security, they are not a complete solution. It's important to adopt a multi-layered security approach that includes regular updates, patches, antivirus software, intrusion detection, and employee training.

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

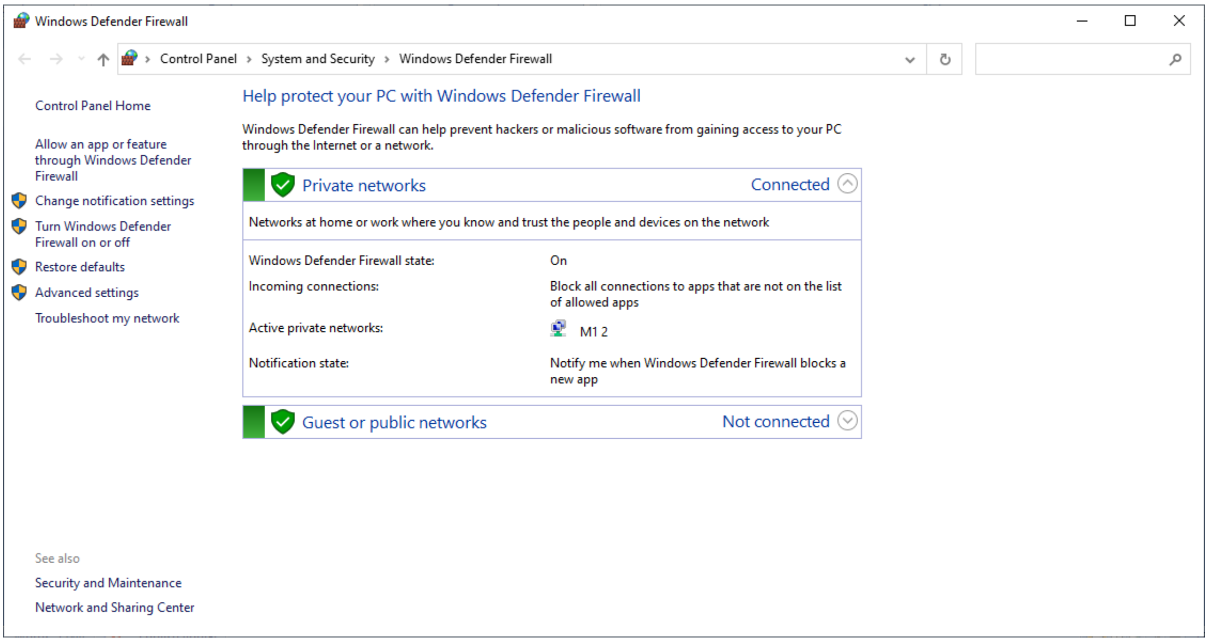
Before starting with the blocking port process, we note that the applications running at the server-end are identified with the well-known Port numbers, some of the commonly used are as follows

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

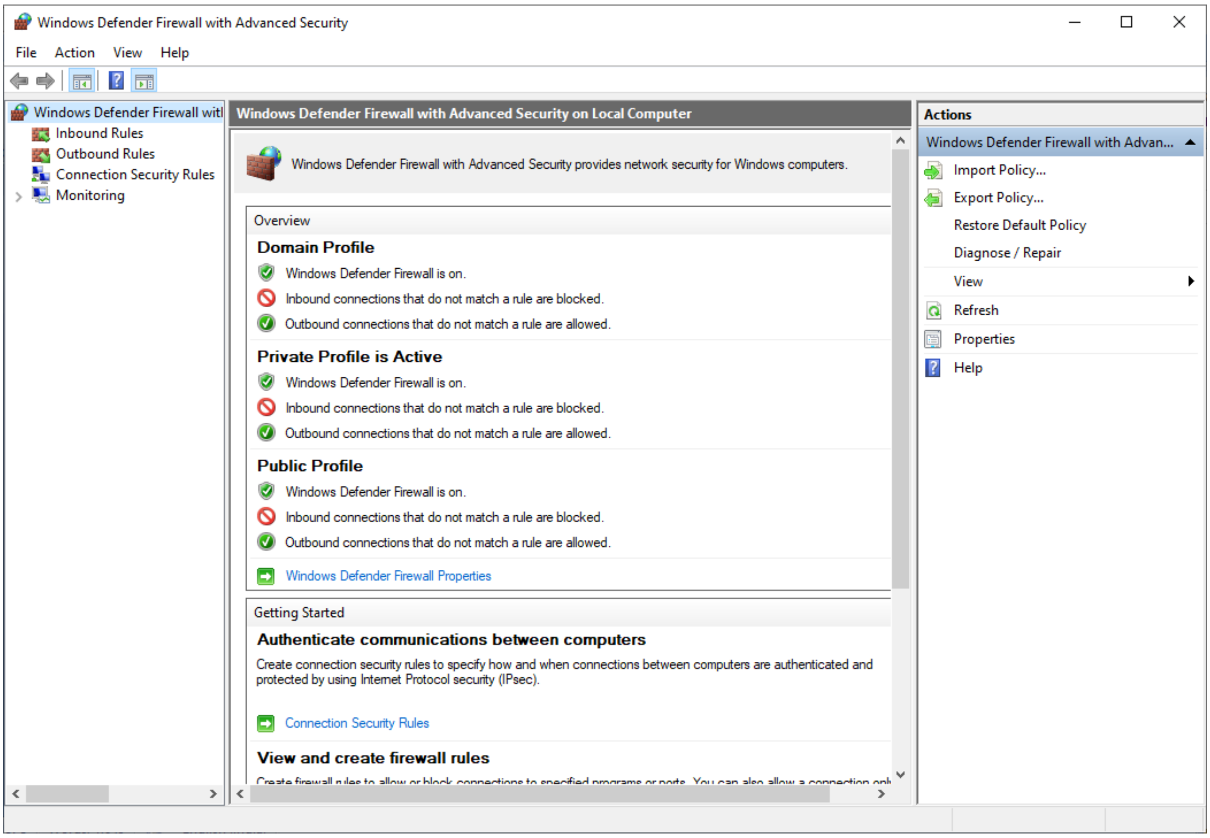
We perform the blocking Port operation as follows:

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

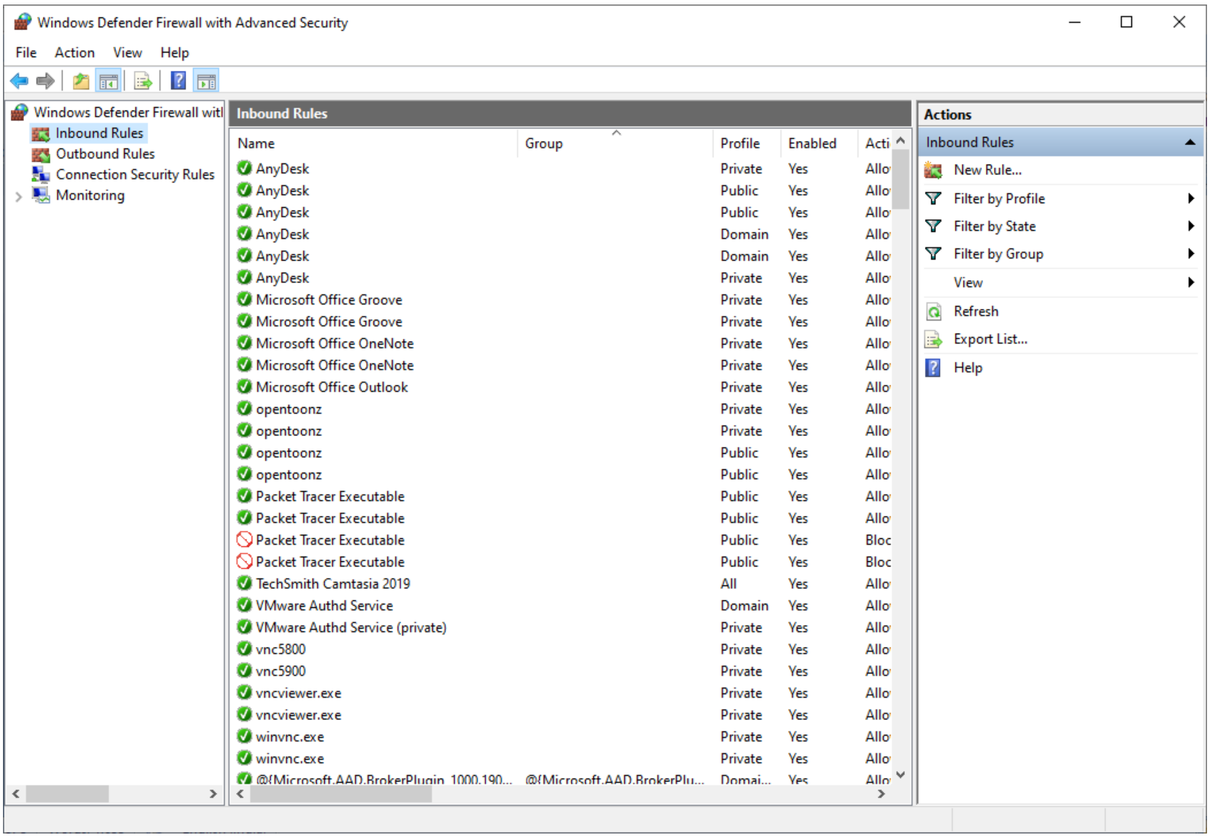
Step 2: We open 'Windows Defender Firewall'



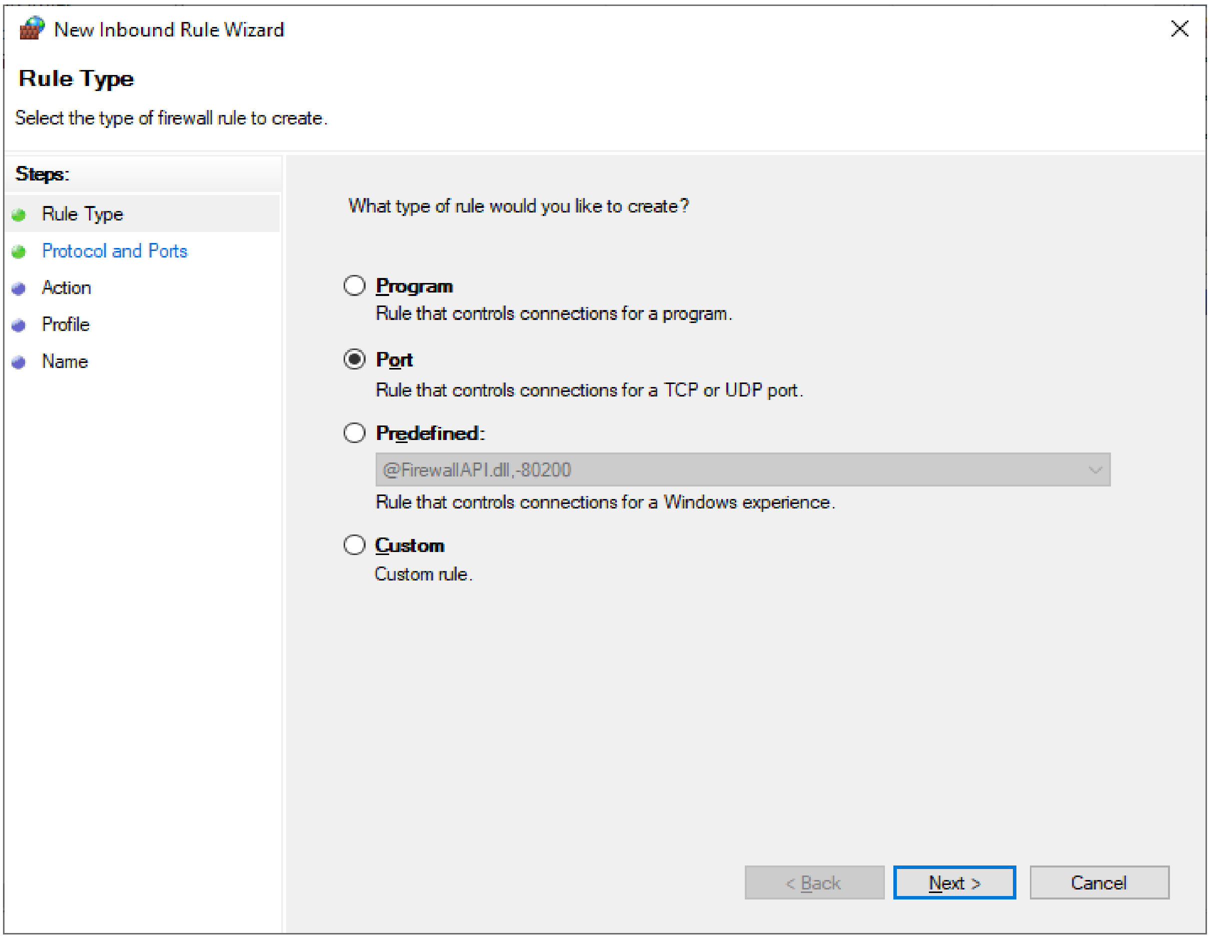
Next we click on 'Advanced settings'



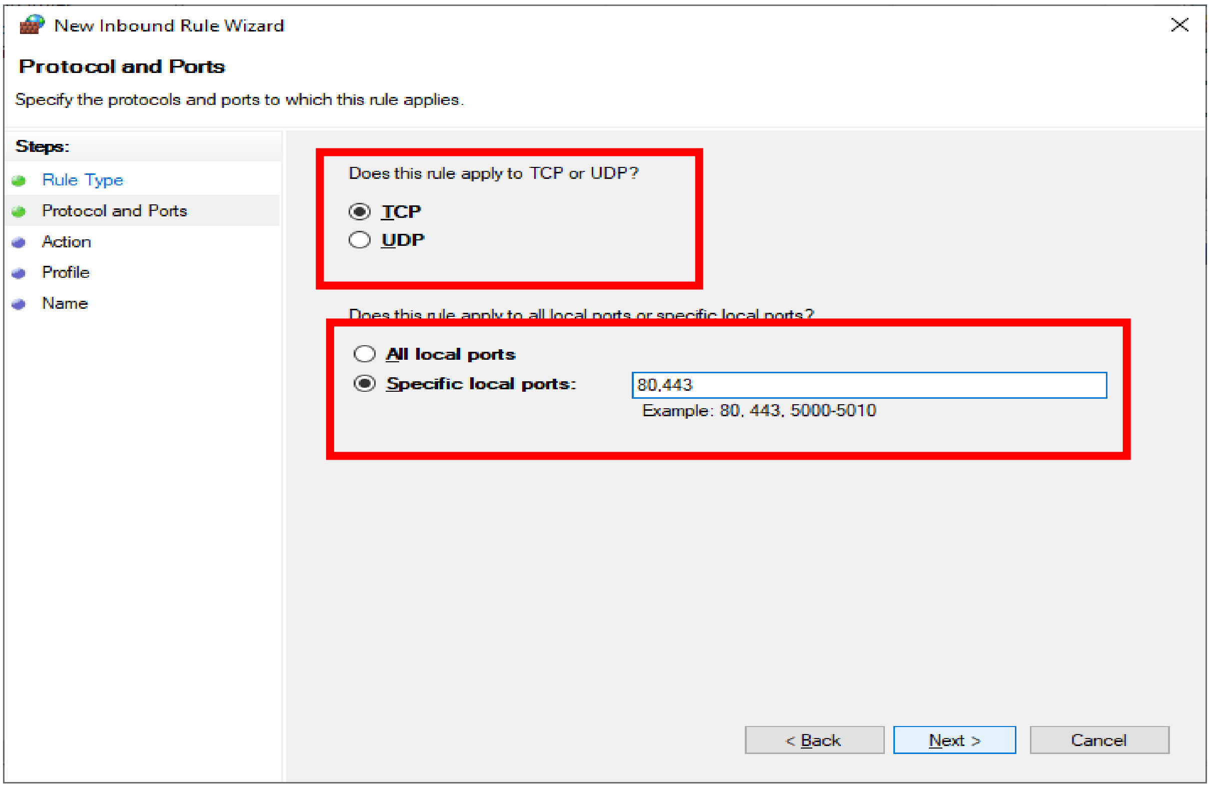
Next we click on 'Inbound Rules'



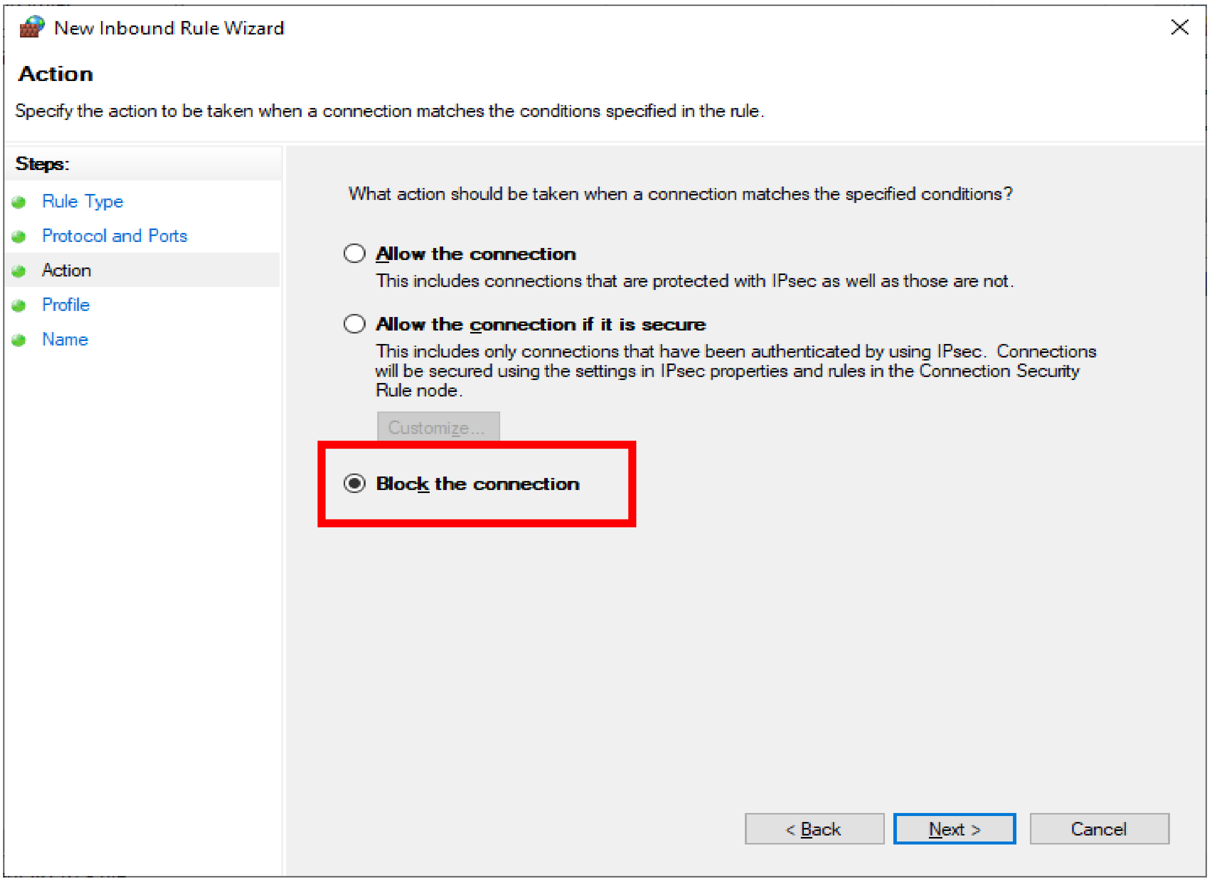
Then click on 'New Rule'



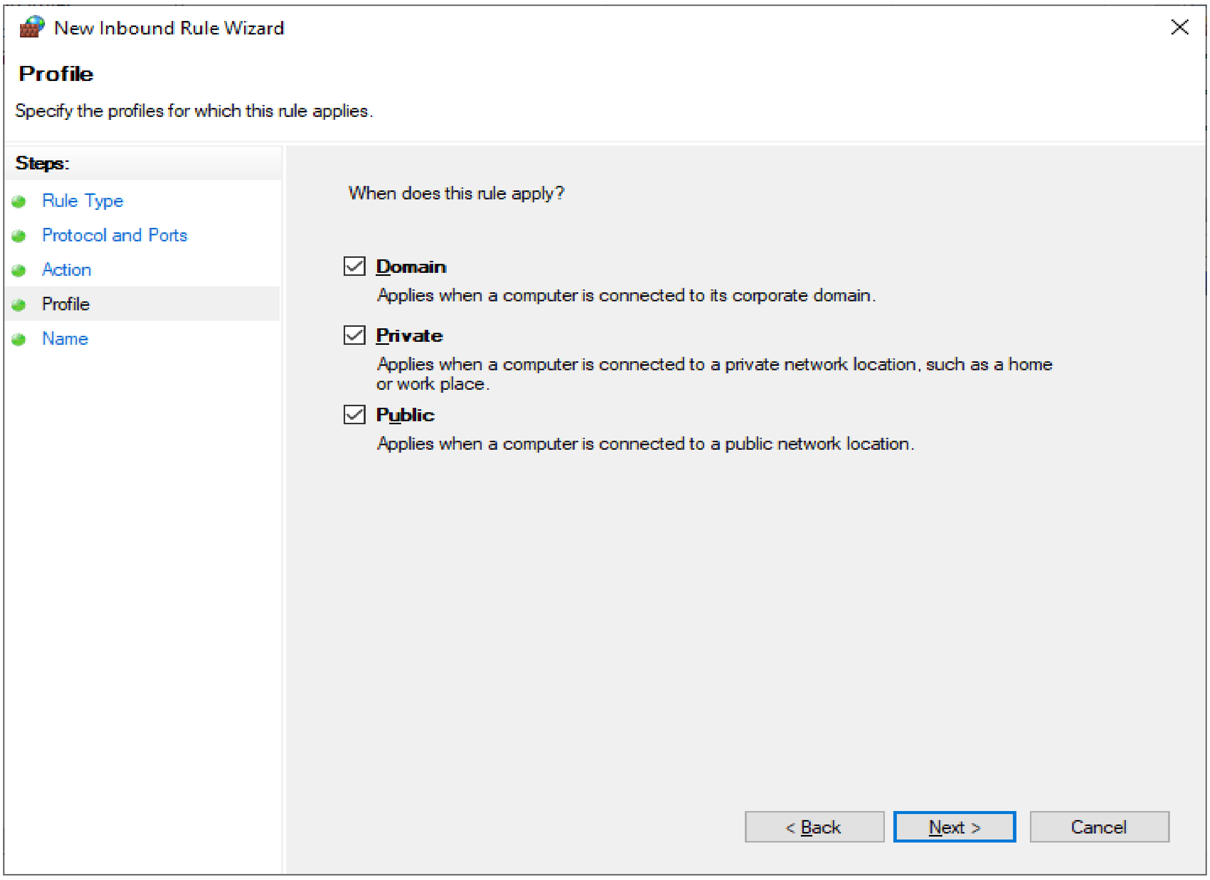
Select the radio button 'Port' and click 'Next' and enter the following



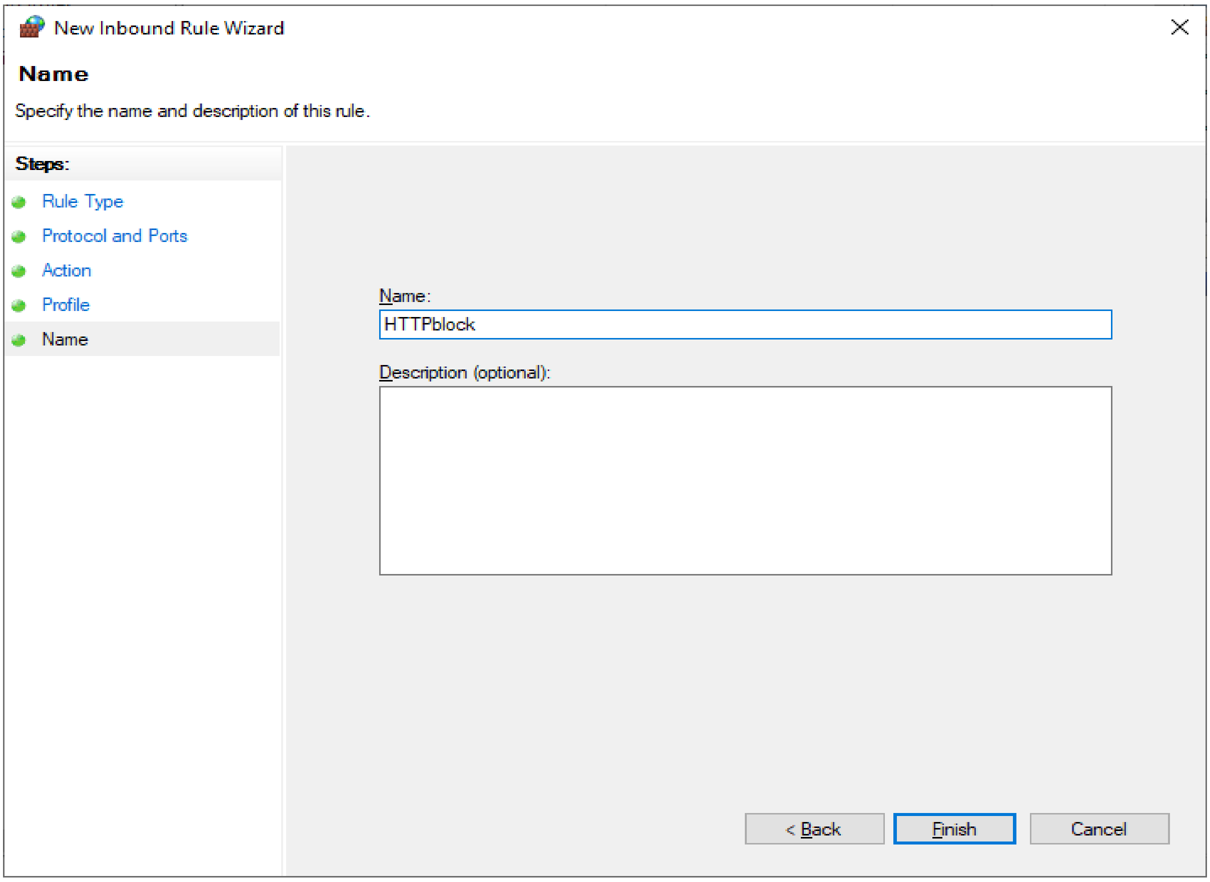
After next, we need to finalise the rule



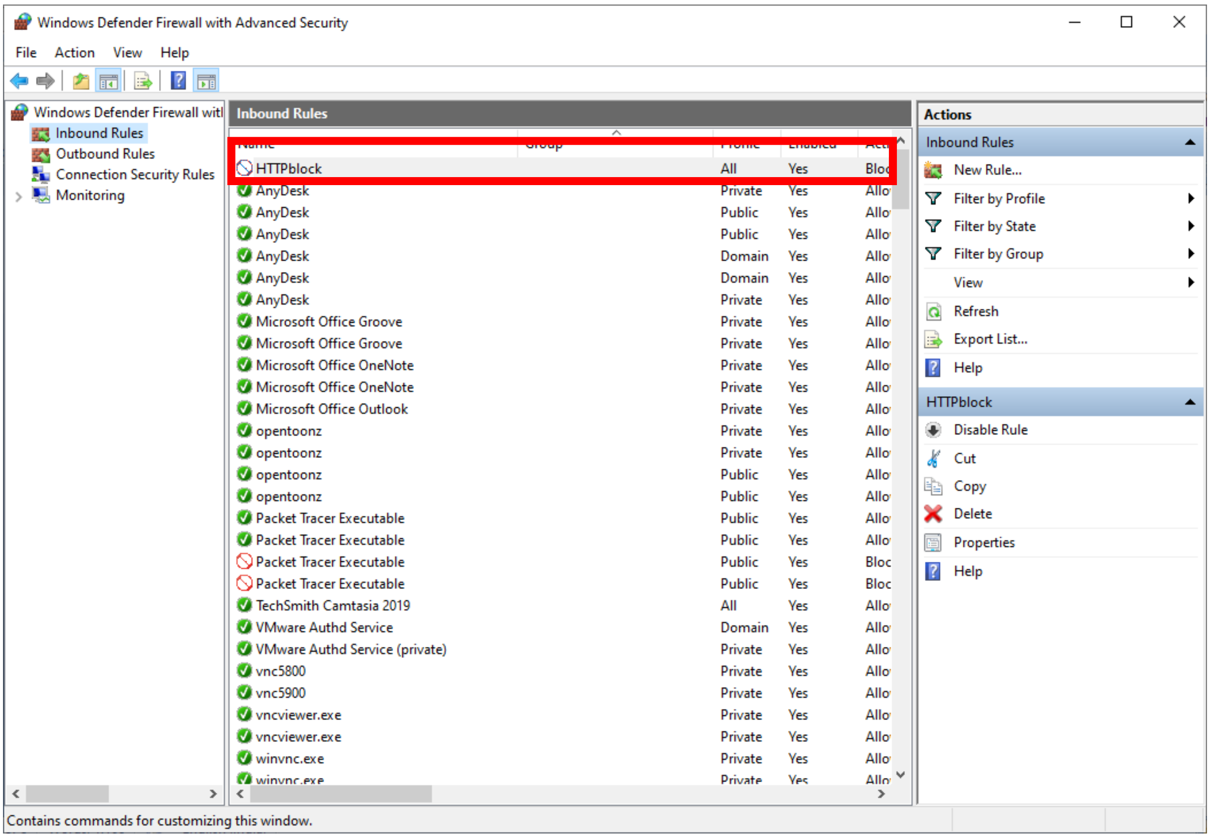
Click 'Next' and we get the following



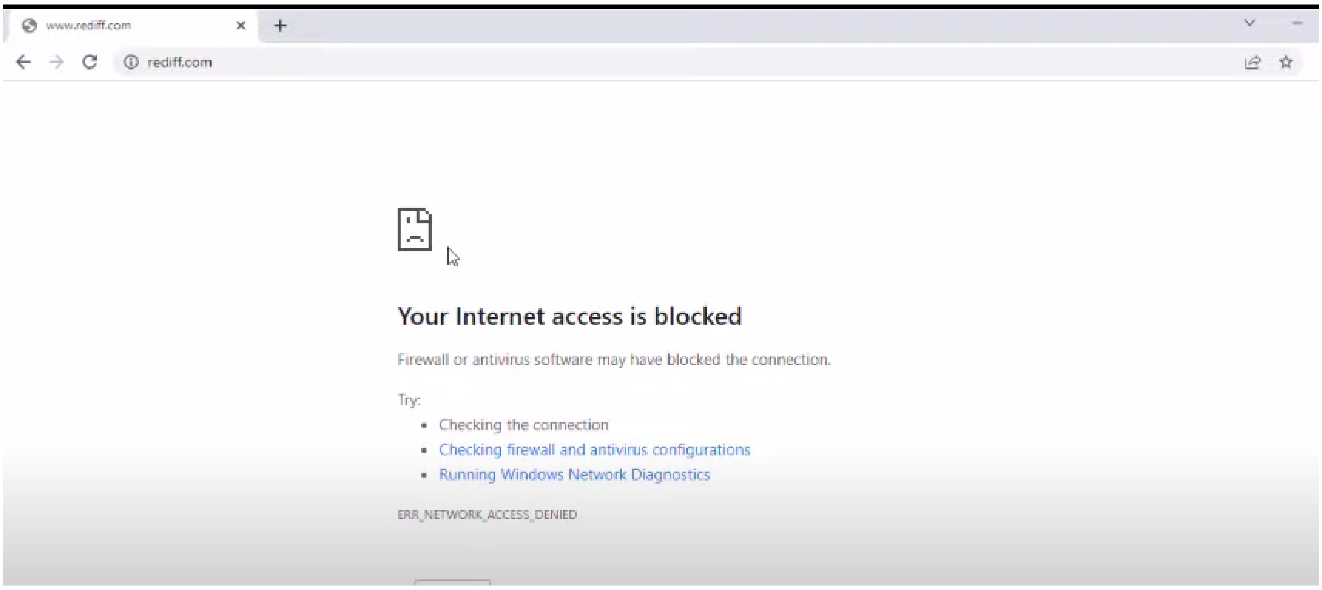
After clicking the 'Next' button we need to name the rule and click finish



The Inbound rule is added

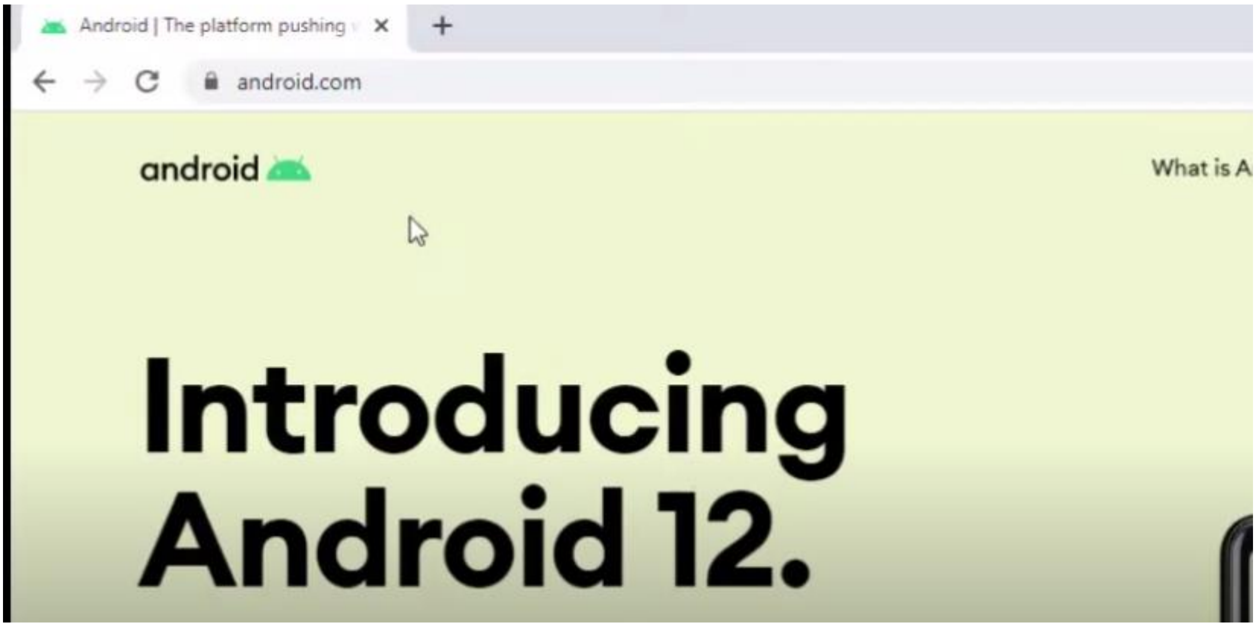


We repeat all the above steps for creating 'Outbound Rules', and then try to access the internet.
We see that the accessed is blocked

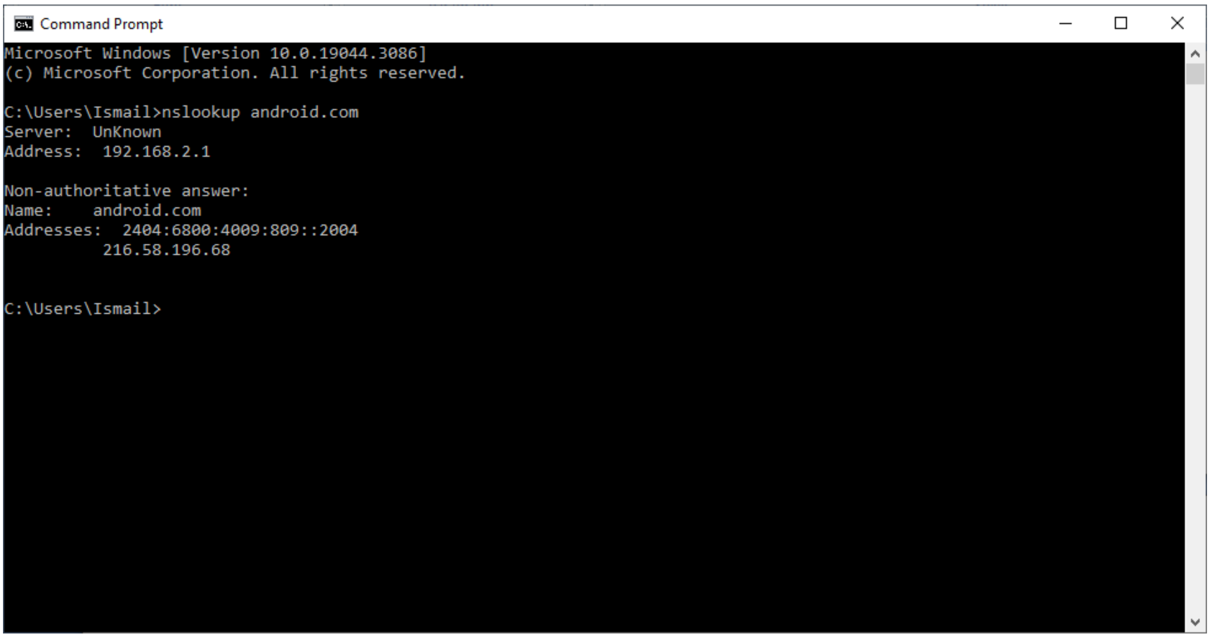


Part 2: Blocking the website www.android.com

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



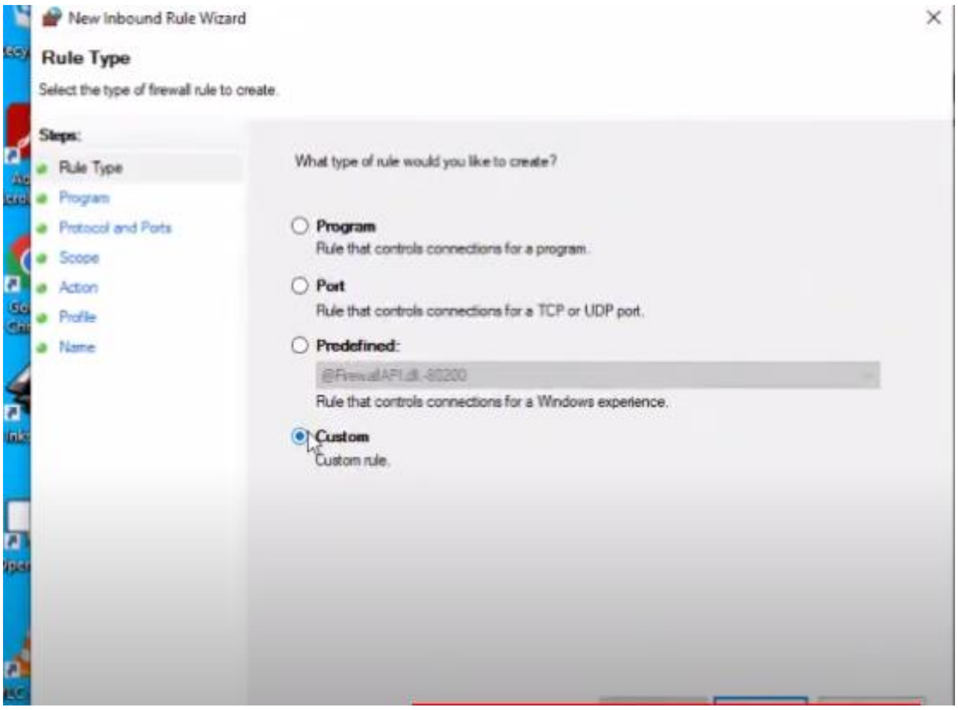
We find the IP addresses of the website using the following command

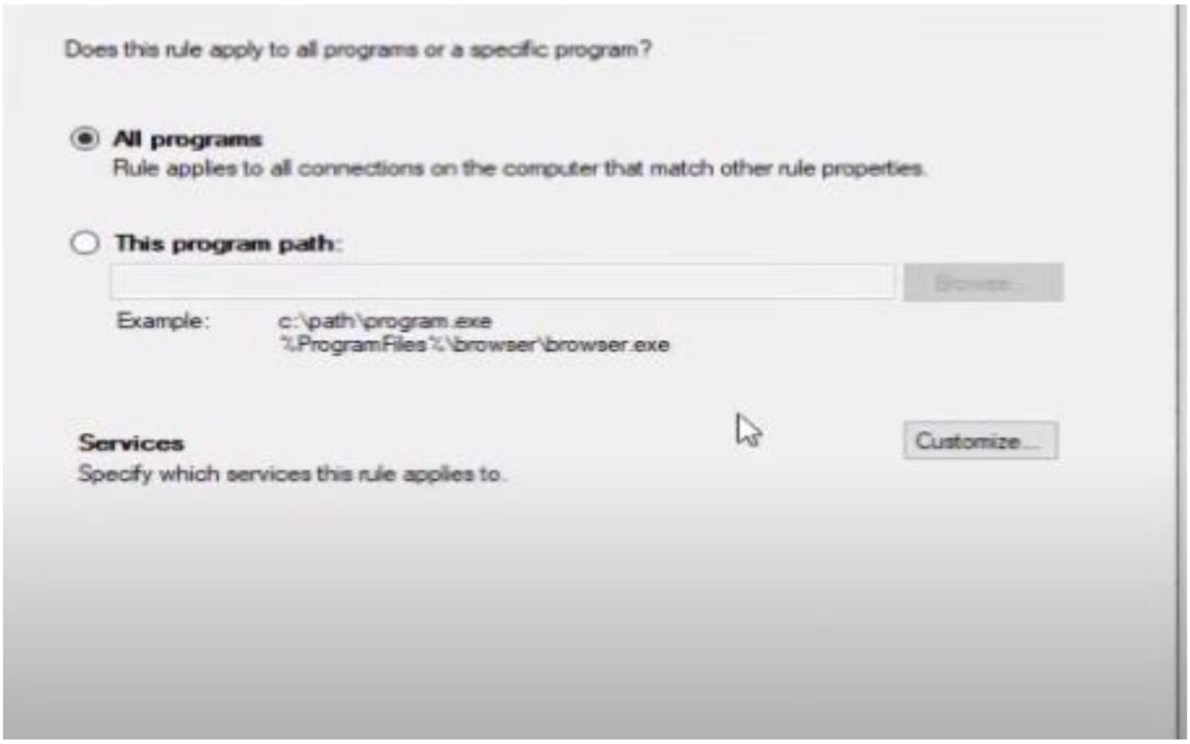


We save the IP addresses

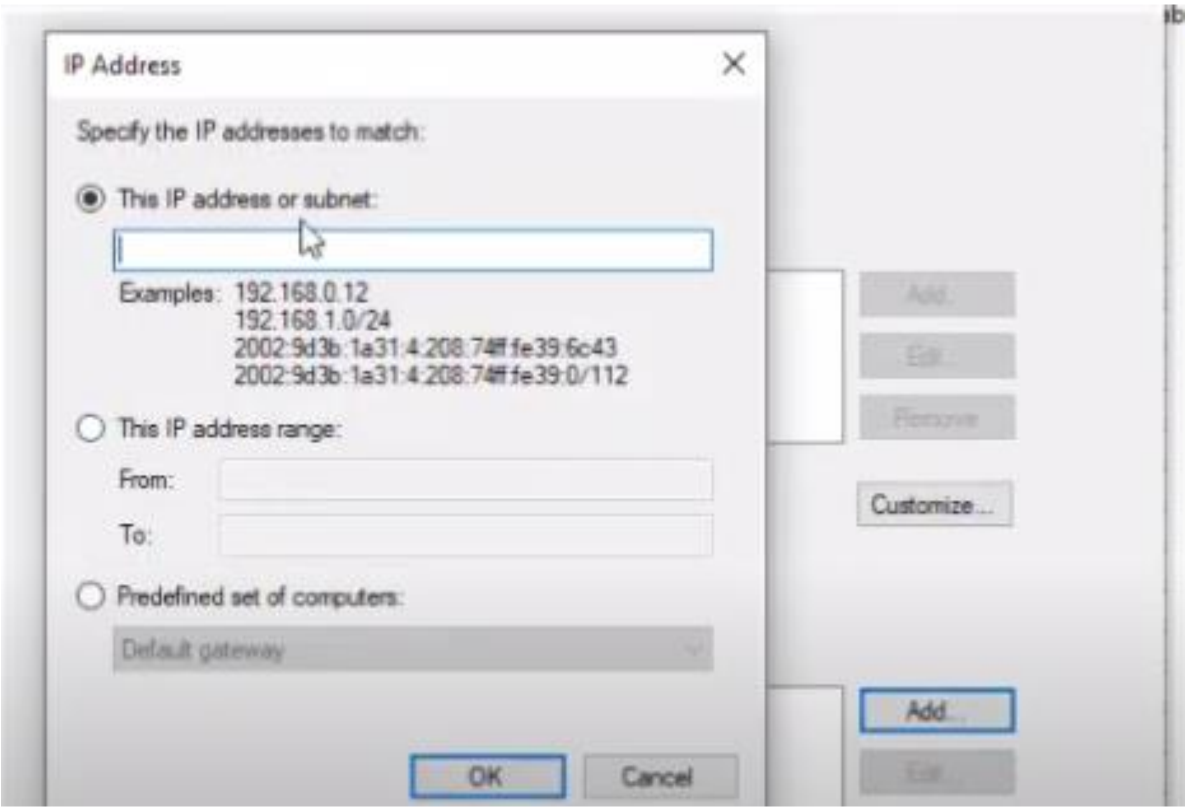
IPv4	216.58.196.68
IPv6	2404:6800:4009:809::2004

We open the windows Firewall settings and apply the Inbound Rule

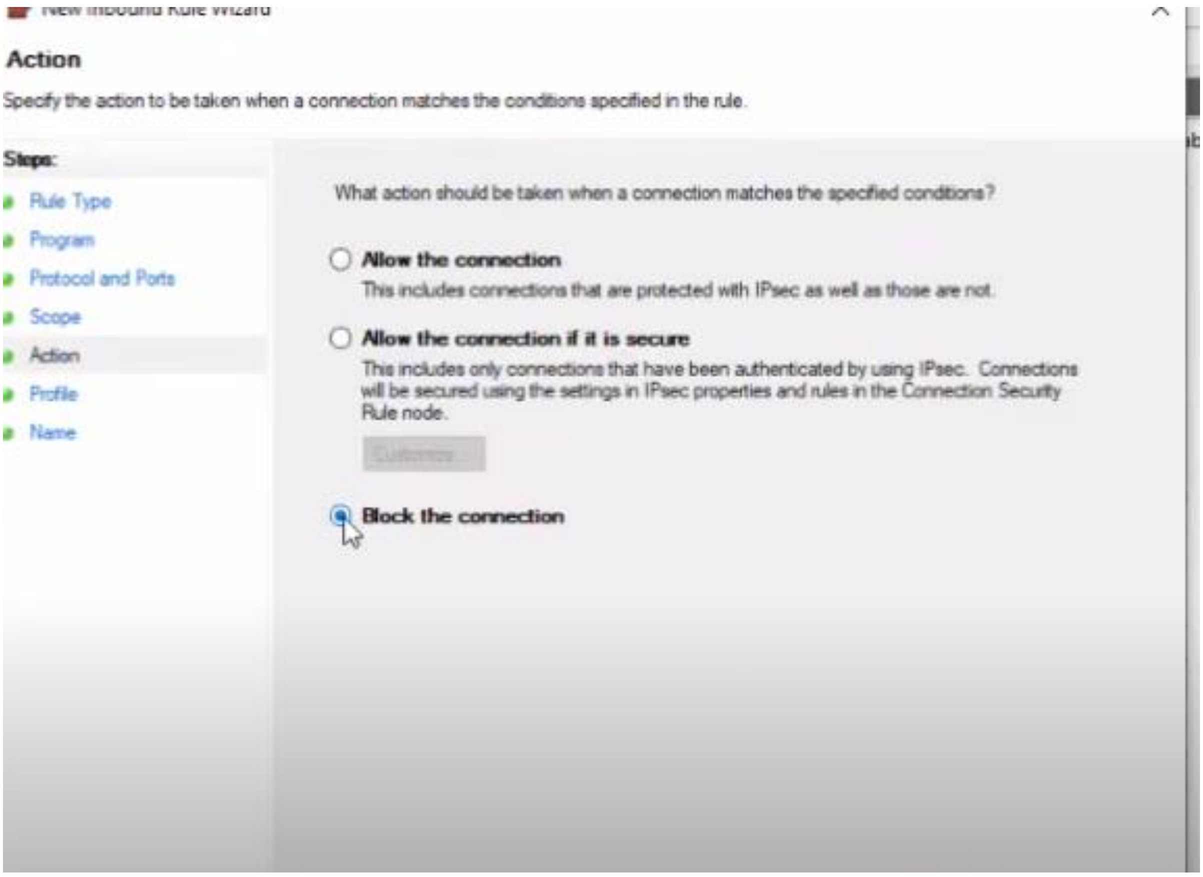




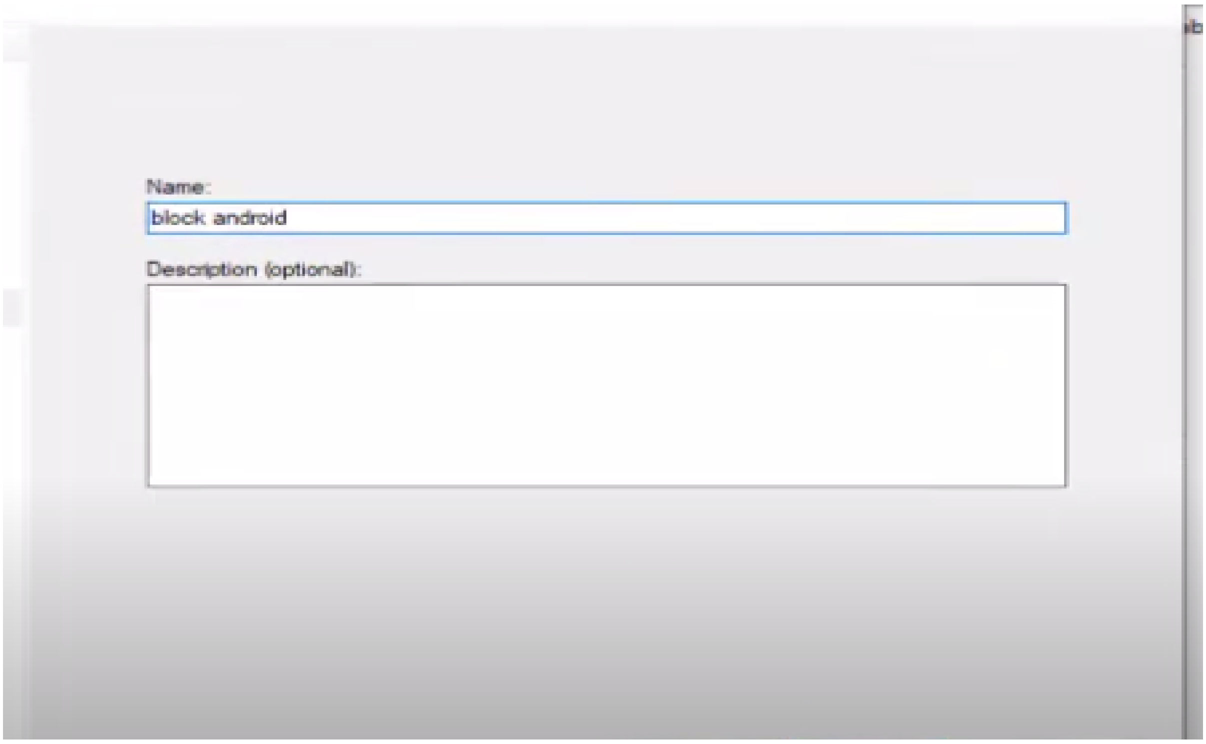
Insert the IP addresses both IPv4 and IPv6



Select Block connection

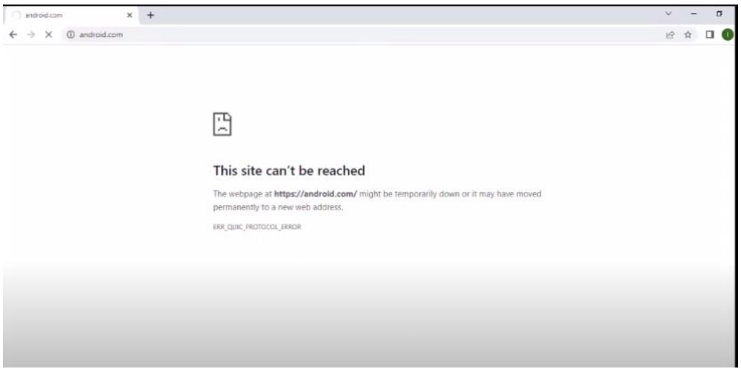


Provide a suitable name and finish



Repeat the above for Outbound Rules

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



For Video demonstration of the above practical click on the link below or scan the QR-code

https://youtu.be/94Pi87vrZfo