Practical no 2: Substitution Cipher

Aim: Write programs to implement the following Substitution Cipher Techniques: Vernam Cipher and Playfair Cipher

1. Vernam Cipher

Code:

```
import java.util.*;
public class Vernam {

   public static int binaryToDecimal(String
binaryString) {
      int decimalValue = 0;
      int base = 1;

      for (int i = binaryString.length() - 1; i >= 0;
i--) {
         char digitChar = binaryString.charAt(i);
         int digit =
Character.getNumericValue(digitChar);

         decimalValue += digit * base;
         base = base * 2;
      }

      return decimalValue;
   }

   public static int[] decimaltobinary(int num){
      int[] binary = new int[7];
      int id = 0;
      while (num > 0) {
         binary[id++] = num % 2;
         num = num / 2;
      }
      return binary;
   }

   public static String vernam_encrypt(String
pt,String key){
      Boolean len_flag=false;
      String cipher_text="";

      if ((key.length())==(pt.length())){
         len_flag=true;
      }
      else{
         len_flag=false;
         System.out.println("Length of Plain
text and key is not equal!");
      }

      if (len_flag==true){
         int[] pt_binary = new int[7];
         int[] key_binary = new int[7];
         for (int i=0;i<pt.length();i++){
            int pt_ascii= (int)pt.charAt(i);
            int key_ascii=(int)key.charAt(i);

            pt_binary=decimaltobinary(pt_ascii);

key_binary=decimaltobinary(key_ascii);


            String new_text="";
            for (int j=pt_binary.length-1;j>=0;j--){
                  if (pt_binary[j]==key_binary[j]){
                     new_text+="0";
                  }
                  else{
                     new_text+="1";
                  }
               }
```

```
            int
new_decimal=binaryToDecimal(new_text);
            if(new_decimal>25){
               new_decimal=new_decimal-26+65;
            }
            else{
               new_decimal=new_decimal+65;
            }



cipher_text=cipher_text+(char)(new_decimal);
         }

      }
      return cipher_text;
   }

   public static void main(String[] args){

      Scanner sc=new Scanner(System.in);
      System.out.print("Enter the plain text: ");
      String pt=sc.next();
      System.out.print("Enter the key: ");
      String key=sc.next();

      System.out.print("Encrypted Text: ");

System.out.print(vernam_encrypt(pt.toUpperC
ase(),key.toUpperCase()));
   }
}
```

Output:

```
Output - INSPractical (run)
run:
Enter the plain text: Dhruv
Enter the key: hello
Encrypted Text: MNEZZBUILD SUCCESSFUL (total time: 14 seconds)
```

2. Playfair Cipher

Code:

```
class Basic{
 String
allChar="ABCDEFGHIJKLMNOPQRSTUVWXYZ";
 boolean indexOfChar(char c)
{
 for(int i=0;i < allChar.length();i++)
 {
  if(allChar.charAt(i)==c)
                    return true;
 }
 return false;
}
}


class PlayFair{
      Basic b=new Basic();
      char keyMatrix[][]=new char[5][5];
```

```
boolean repeat(char c)
{
   if(!b.indexOfChar(c))
   {
      return true;
   }
      for(int i=0;i < keyMatrix.length;i++)
      {
         for(int j=0;j <
keyMatrix[i].length;j++)
         {
            if(keyMatrix[i][j]==c || c=='J')
               return true;
         }
      }
      return false;
}

void insertKey(String key)
{
    key=key.toUpperCase();
    key=key.replaceAll("J", "I");
    key=key.replaceAll(" ", "");
    int a=0,b=0;

    for(int k=0;k < key.length();k++)
    {
         if(!repeat(key.charAt(k)))
         {
            keyMatrix[a][b++]=key.charAt(k);
            if(b>4)
            {
               b=0;
               a++;
            }
         }
      }

      char p='A';

      while(a < 5)
      {
         while(b < 5)
         {
            if(!repeat(p))
            {
               keyMatrix[a][b++]=p;

            }
           p++;
         }
         b=0;
         a++;
      }
       System.out.print("----------------------
--Key Matrix-------------------");
      for(int i=0;i < 5;i++)
      {
         System.out.println();
         for(int j=0;j < 5;j++)
         {
```

```
                System.out.print("\t"+keyMatrix[i][
j]);
            }
        }
        System.out.println("\n-------------------
-------------------------------------");

    }

    int rowPos(char c)
    {
        for(int i=0;i < keyMatrix.length;i++)
            {
                for(int j=0;j <
keyMatrix[i].length;j++)
                {
                    if(keyMatrix[i][j]==c)
                        return i;
                }
            }
        return -1;
    }

    int columnPos(char c)
    {
        for(int i=0;i < keyMatrix.length;i++)
            {
                for(int j=0;j <
keyMatrix[i].length;j++)
                {
                    if(keyMatrix[i][j]==c)
                        return j;
                }
            }
        return -1;
    }

    String encryptChar(String plain)
    {
        plain=plain.toUpperCase();
        char a=plain.charAt(0),b=plain.charAt(1);
        String cipherChar="";
        int r1,c1,r2,c2;
        r1=rowPos(a);
        c1=columnPos(a);
        r2=rowPos(b);
        c2=columnPos(b);

        if(c1==c2)
        {
            ++r1;
            ++r2;
            if(r1>4)
                r1=0;

            if(r2>4)
                r2=0;
            cipherChar+=keyMatrix[r1][c2];
            cipherChar+=keyMatrix[r2][c1];
        }
        else if(r1==r2)
        {
```

```
        ++c1;
        ++c2;
        if(c1>4)
            c1=0;

        if(c2>4)
            c2=0;
        cipherChar+=keyMatrix[r1][c1];
        cipherChar+=keyMatrix[r2][c2];

    }
    else{
        cipherChar+=keyMatrix[r1][c2];
        cipherChar+=keyMatrix[r2][c1];
    }
    return cipherChar;
}




String Encrypt(String plainText,String key)
{
    insertKey(key);
    String cipherText="";
    plainText=plainText.replaceAll("j", "i");
    plainText=plainText.replaceAll(" ", "");
    plainText=plainText.toUpperCase();
    int len=plainText.length();
    //
System.out.println(plainText.substring(1,2+1));
        if(len/2!=0)
        {
            plainText+="X";
            ++len;
        }

        for(int i=0;i < len-1;i=i+2)
        {
            cipherText+=encryptChar(plainText.su
bstring(i,i+2));
            cipherText+=" ";
        }
        return cipherText;
    }



    String decryptChar(String cipher)
    {
        cipher=cipher.toUpperCase();
        char
a=cipher.charAt(0),b=cipher.charAt(1);
        String plainChar="";
        int r1,c1,r2,c2;
        r1=rowPos(a);
        c1=columnPos(a);
        r2=rowPos(b);
        c2=columnPos(b);

        if(c1==c2)
        {
            --r1;
```

```
        --r2;
        if(r1 < 0)
            r1=4;

        if(r2 < 0)
            r2=4;
        plainChar+=keyMatrix[r1][c2];
        plainChar+=keyMatrix[r2][c1];
    }
    else if(r1==r2)
    {
        --c1;
        --c2;
        if(c1 < 0)
            c1=4;

        if(c2 < 0)
            c2=4;
        plainChar+=keyMatrix[r1][c1];
        plainChar+=keyMatrix[r2][c2];

    }
    else{
        plainChar+=keyMatrix[r1][c2];
        plainChar+=keyMatrix[r2][c1];
    }
    return plainChar;
}

    String Decrypt(String cipherText,String
key)
    {
        String plainText="";
        cipherText=cipherText.replaceAll("j",
"i");
        cipherText=cipherText.replaceAll(" ", "");
        cipherText=cipherText.toUpperCase();
        int len=cipherText.length();
        for(int i=0;i < len-1;i=i+2)
        {
            plainText+=decryptChar(cipherText.su
bstring(i,i+2));
            plainText+=" ";

        }
        return plainText;
    }
}
class PlayFairCipher2{
    public static void main(String
args[])throws Exception
    {
        PlayFair p=new PlayFair();
        Scanner scn=new Scanner(System.in);
        String key,cipherText,plainText;

        System.out.println("Enter plaintext:");
        plainText=scn.nextLine();
```

```
        System.out.println("Enter Key:");
        key=scn.nextLine();

        cipherText=p.Encrypt(plainText,key);

        System.out.println("Encrypted text:");
        System.out.println("----------------------------------------------------------------------\n"+cipherText);
        System.out.println("----------------------------------------------------------------------");

        String encryptedText=p.Decrypt(cipherText, key);
        System.out.println("Decrypted text:" );
        System.out.println("----------------------------------------------------------------------\n"+encryptedText);
        System.out.println("----------------------------------------------------------------------");

    }
}
```

Output:

```
Output - INSPractical (run)
run:
Enter plaintext:
Hello
Enter Key:
4
-------------------------Key Matrix-------------------
        A       B       C       D       E
        F       G       H       I       K
        L       M       N       O       P
        Q       R       S       T       U
        V       W       X       Y       Z
-------------------------------------------------------
Encrypted text:
-------------------------------------------------------
KC QQ NY
-------------------------------------------------------
Decrypted text:
-------------------------------------------------------
HE LL OX
-------------------------------------------------------
BUILD SUCCESSFUL (total time: 16 seconds)
```

Practical no 4: DES and AES Algorithm

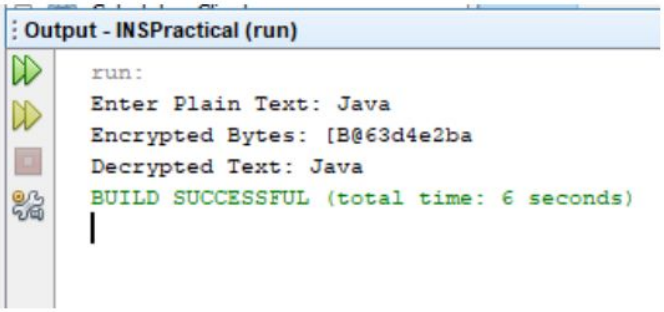
Aim: Write programs to encrypt and decrypt strings using - DES Algorithm and AES Algorithm

A) DES Algorithm

Code:

```
import java.security.InvalidKeyException;
import
java.security.NoSuchAlgorithmException;
import java.util.Scanner;
import javax.crypto.*;

public class DESAlgorithm {

   public static void main(String[] args) throws
NoSuchAlgorithmException,
InvalidKeyException,
NoSuchPaddingException,
IllegalBlockSizeException,
BadPaddingException{

      //Asking for text as input
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter Plain Text: ");
      String plain=sc.next();

      //Generate Key
      KeyGenerator
mygen=KeyGenerator.getInstance("DES");
      SecretKey mykey=mygen.generateKey();

      //Cipher object to encrypt and decrypt
the text
      Cipher
descipher=Cipher.getInstance("DES");

descipher.init(Cipher.ENCRYPT_MODE,mykey);
      byte[] mybytes=plain.getBytes();
      byte[] myEncrypted =
descipher.doFinal(mybytes);

descipher.init(Cipher.DECRYPT_MODE,mykey);
      byte[] myDecrypted =
descipher.doFinal(myEncrypted);

      System.out.println("Encrypted Bytes:
"+myEncrypted);
      System.out.println("Decrypted Text:
"+new String(myDecrypted));
   }
}
```

Output:



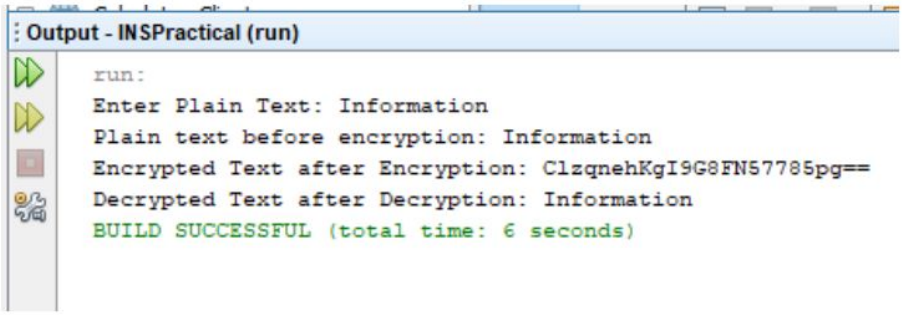
B) AES Algorithm

Code:

```
import java.security.InvalidKeyException;
import
java.security.NoSuchAlgorithmException;
import java.util.Base64;
import java.util.Scanner;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;


public class DESAlgorithm {

   static Cipher cipher;

   public static void main(String[] args) throws
NoSuchAlgorithmException,
InvalidKeyException,
NoSuchPaddingException,
IllegalBlockSizeException,
BadPaddingException, Exception{

      //Asking for text as input
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter Plain Text: ");
      String plain=sc.next();

      KeyGenerator
keygen=KeyGenerator.getInstance("AES");
      keygen.init(128);
      cipher = Cipher.getInstance("AES");
      SecretKey mykey=keygen.generateKey();
      System.out.println("Plain text before
encryption: "+plain);

      String
encryptedText=encrypt(plain,mykey);
      System.out.println("Encrypted Text after
Encryption: "+encryptedText);

      String
decryptedText=decrypt(encryptedText,mykey);
      System.out.println("Decrypted Text after
Decryption: "+decryptedText);
   }

   public static String encrypt(String plain,
SecretKey mykey) throws Exception,
InvalidKeyException,
IllegalBlockSizeException,
BadPaddingException{

      byte[] plaintext_byte=plain.getBytes();
      cipher.init(Cipher.ENCRYPT_MODE,mykey);
      byte[]
encryptedByte=cipher.doFinal(plaintext_byte);
      Base64.Encoder
encoder=Base64.getEncoder();
      String
encryptedText=encoder.encodeToString(encry
ptedByte);
      return encryptedText;
   }
```

```
    public static String decrypt(String
encryptedText, SecretKey mykey) throws
Exception, InvalidKeyException,
IllegalBlockSizeException,
BadPaddingException{
        Base64.Decoder
decoder=Base64.getDecoder();
        byte[]
encryptedTextByte=decoder.decode(encrypted
Text);
        cipher.init(Cipher.DECRYPT_MODE,mykey);
        byte[]
decryptedByte=cipher.doFinal(encryptedTextB
yte);
        String decryptedText=new
String(decryptedByte);
        return decryptedText;
    }
}
```

Output:



Practical no 5: RSA Algorithm

Aim: Write a program to implement RSA Algorithm to perform encryption/decryption of a given string

Code:

```
public class RsaAlgorithm {

    public static void main(String[] args) {
        int p = 3;
        int q = 7;
        int n = p * q;
        int phi = (p - 1) * (q - 1);

        int e = 2;
        while (gcd(e, phi) != 1) {
            e++;
        }

        int k = 2;
        double d = (1 + (k * phi)) / e;


        int w = 1;
        while ((d * e) % phi != 1) {
            w++;
        }

        System.out.println("Public Key: (n = " + n
+ ", e = " + e + ")");
        System.out.println("Private Key: (n = " + n
+ ", d = " + w + ") "+w);

        int plaintext = 5; // Message to encrypt
        int ciphertext = (int) Math.pow(plaintext,
e) % n; // Encrypting
        int decryptedText = (int)
Math.pow(ciphertext, d) % n; // Decrypting

        System.out.println("Plaintext: " +
plaintext);
        System.out.println("Ciphertext: " +
ciphertext);
        System.out.println("Decrypted Text: " +
decryptedText);
    }

    // Function to calculate GCD
    public static int gcd(int a, int b) {
        if (b == 0) {
            return a;
        }
        return gcd(b, a % b);
    }
}
```

Output:

Output - INSPractical (run)

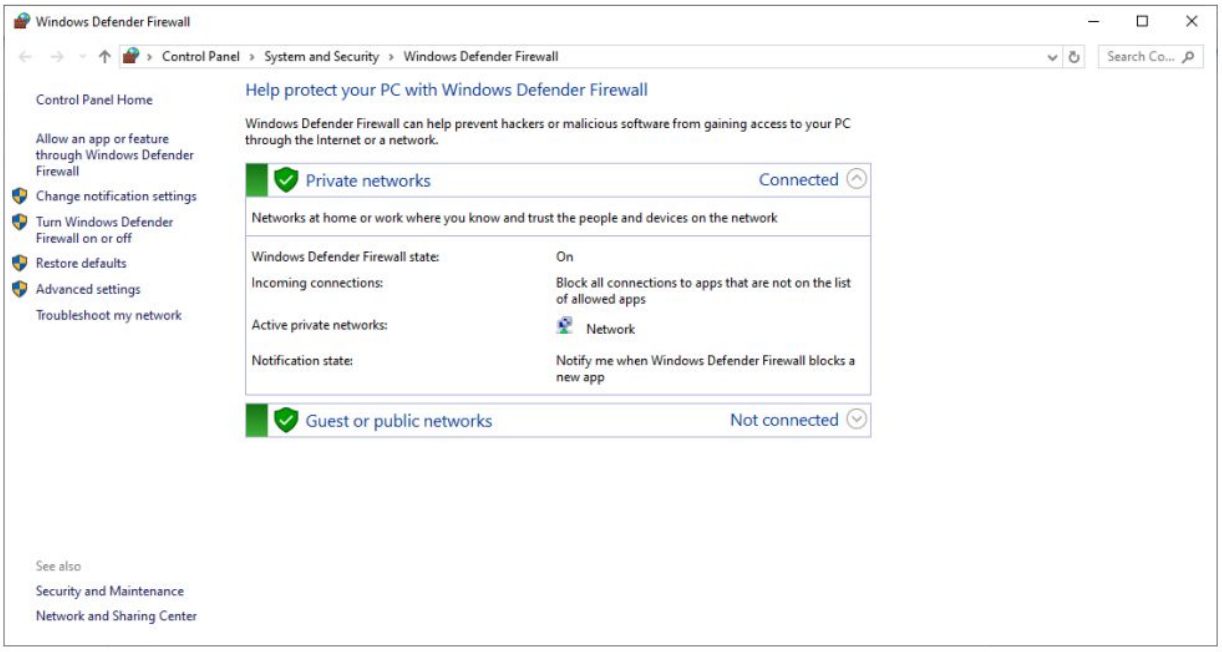
```
run:
Public Key: (n = 21, e = 5)
Private Key: (n = 21, d = 1) 1
Plaintext: 5
Ciphertext: 17
Decrypted Text: 5
BUILD SUCCESSFUL (total time: 0 seconds)
```

Practical no 8: Configure Windows Firewall

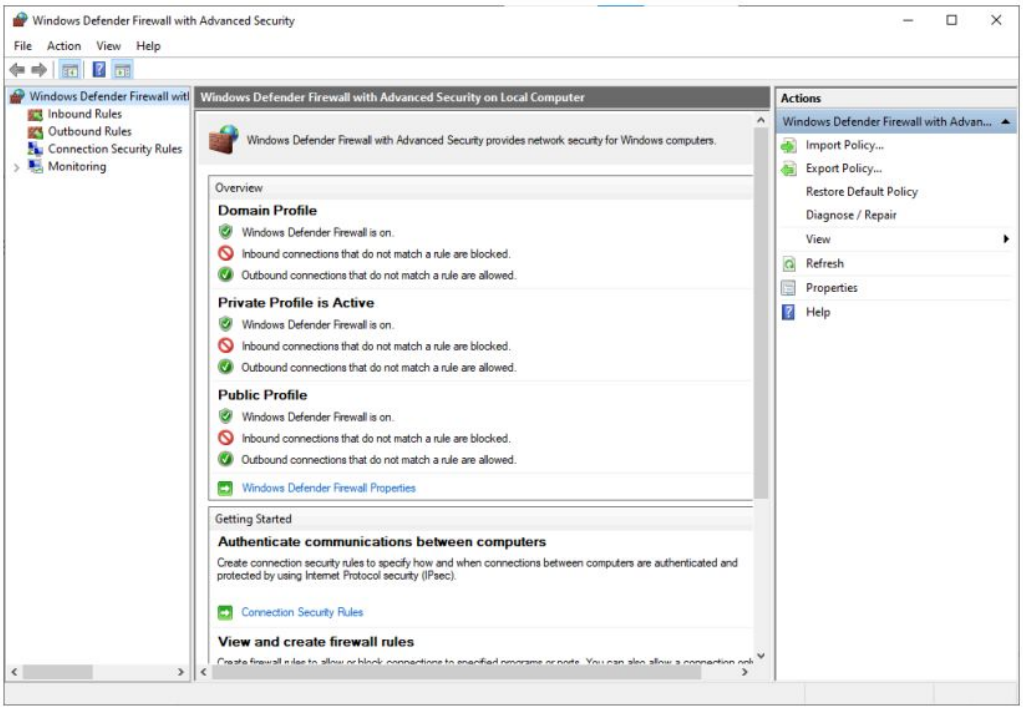
Aim: Configure Windows Firewall to block : Port, An program and a website

A) Port

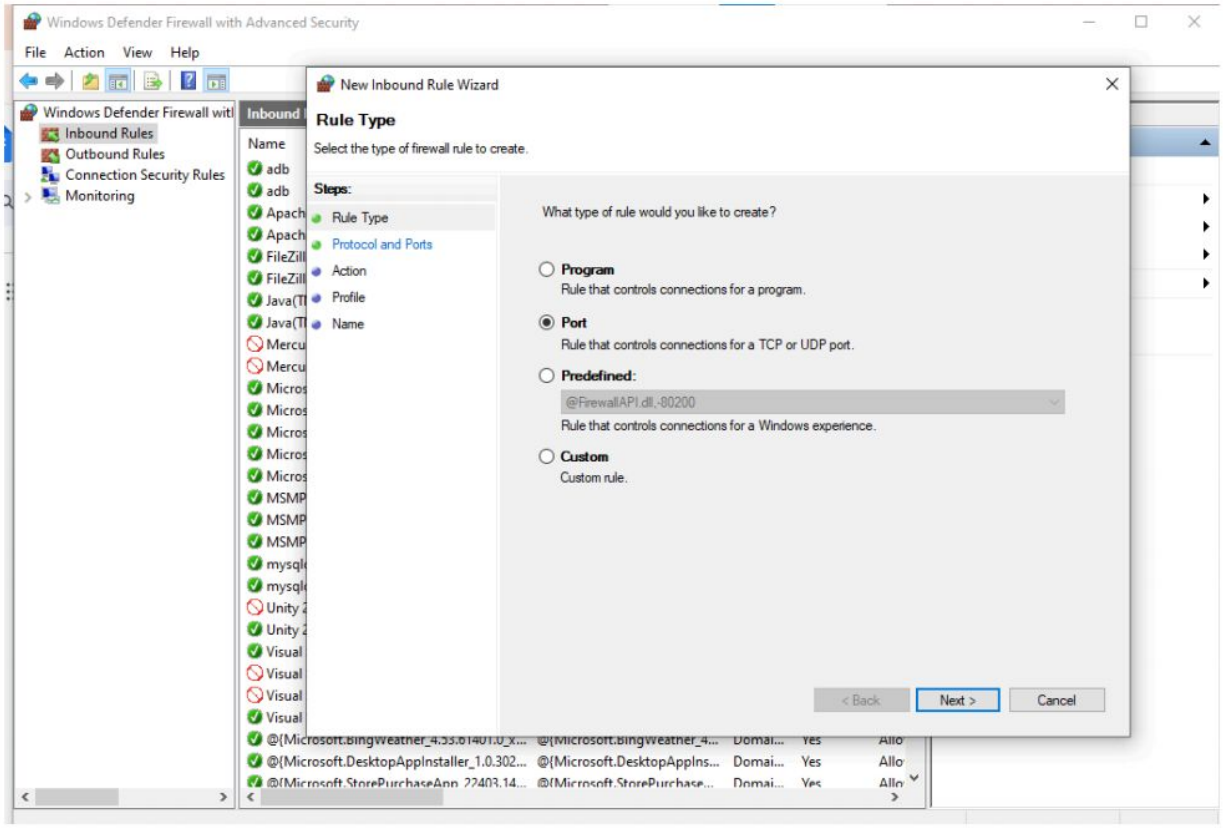
Step 1: Open windows firewall defender



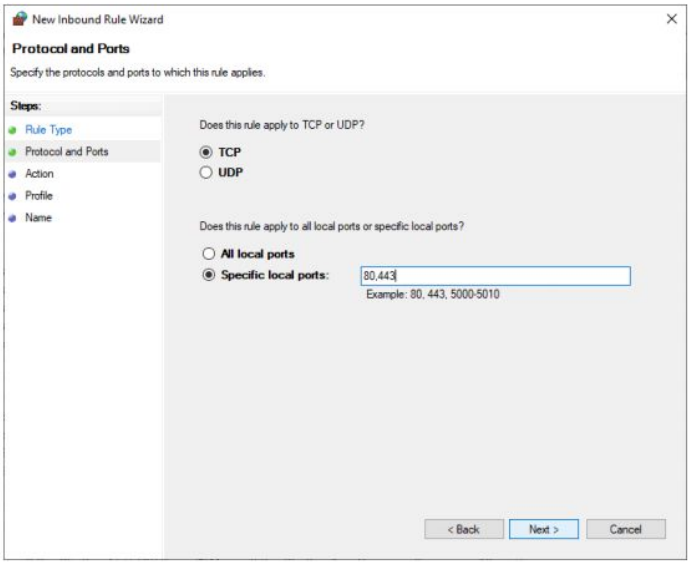
Step 2: Change to advanced settings



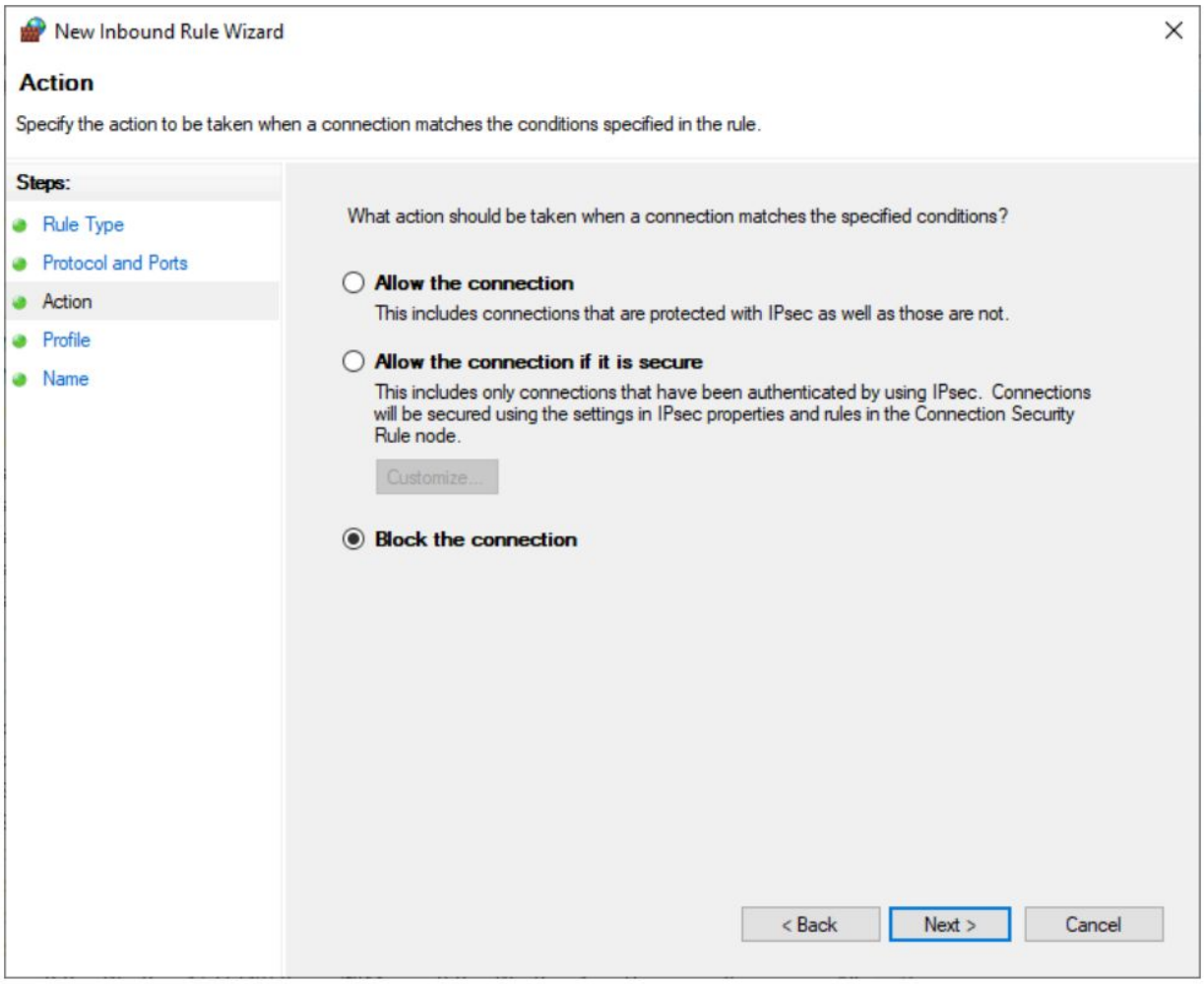
Step 3: Right click on inbound rules and create new rule and select port as rule type



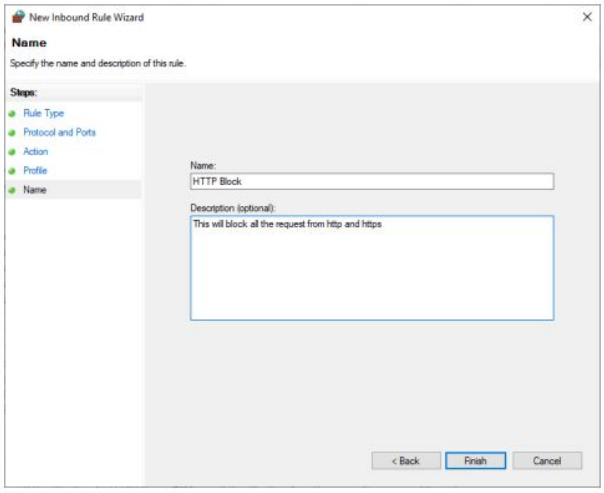
Step 4: Enter the port as 80,443 as 80 used by http and 443 is used by https



Step 5: Select block the connection in action type

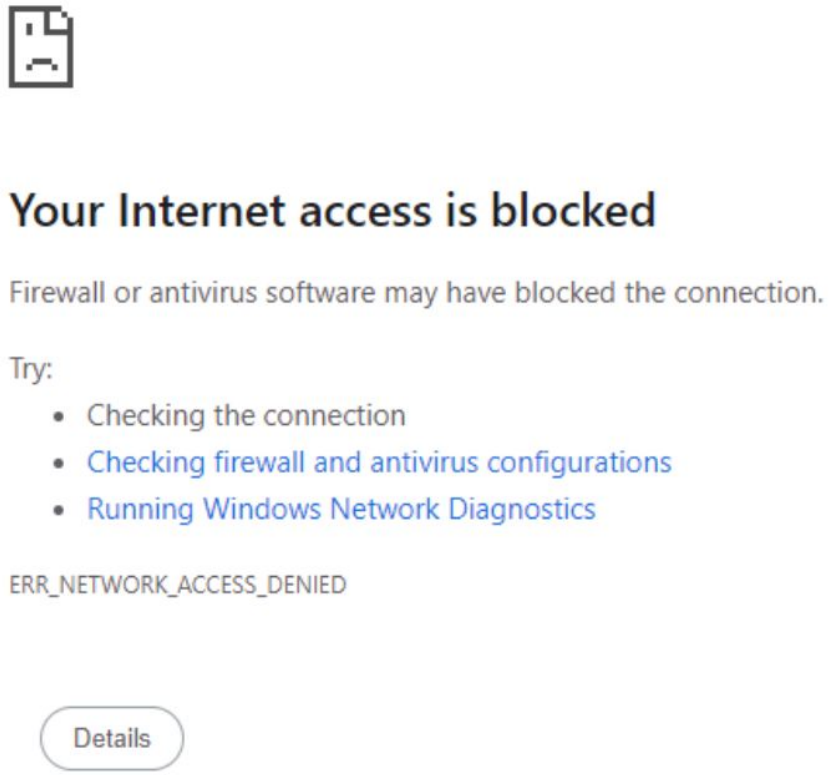


Step 6: Give name and description to the rule



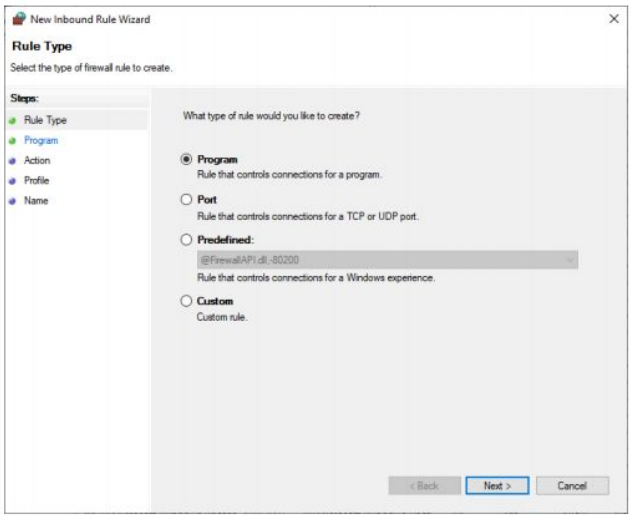
Step 7: Right click on the Outbound rules and perform the same procedure to create the same rule

Step 8: Access any http or https link and check whether it is accessible or not

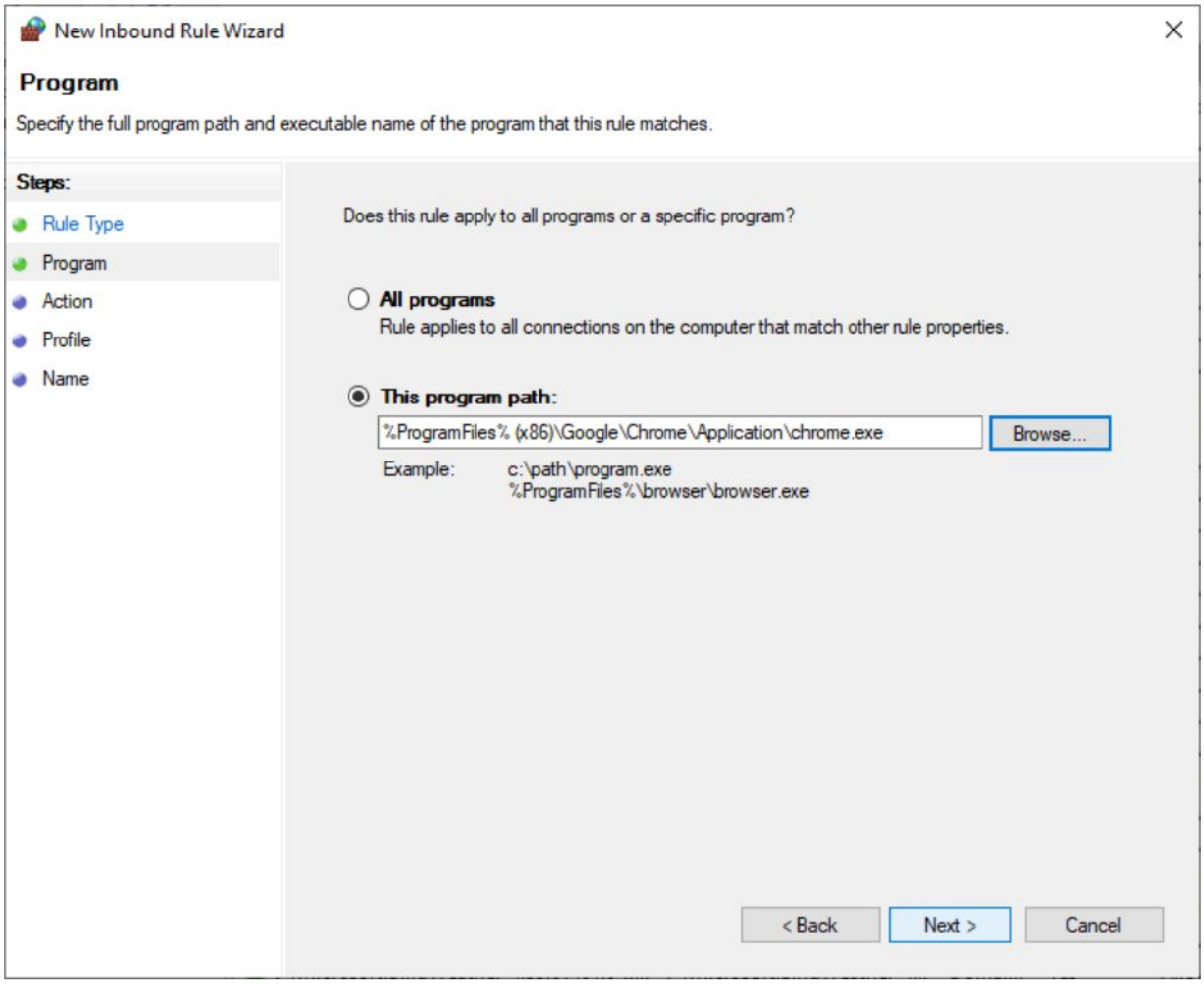


B. Program

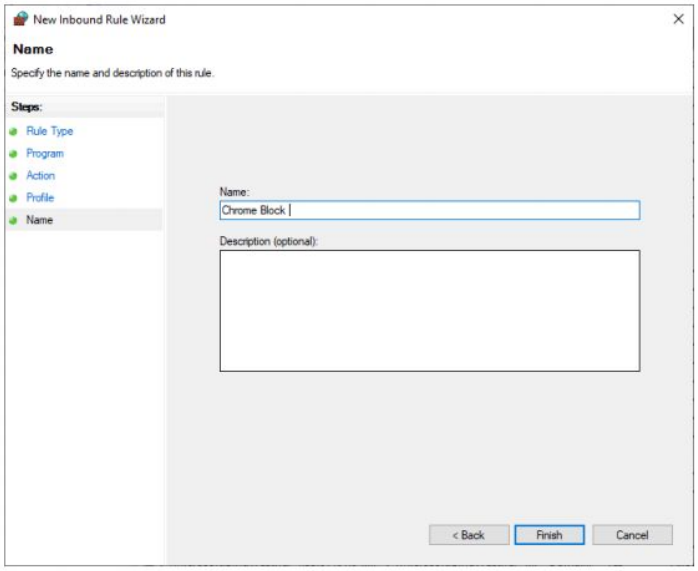
Step 1: Right click on Inbound and create new rule and select rule type as program



Step 2: Select the path of the program (Here selected the chrome.exe) from the file explorer



Step 3: Name the rule

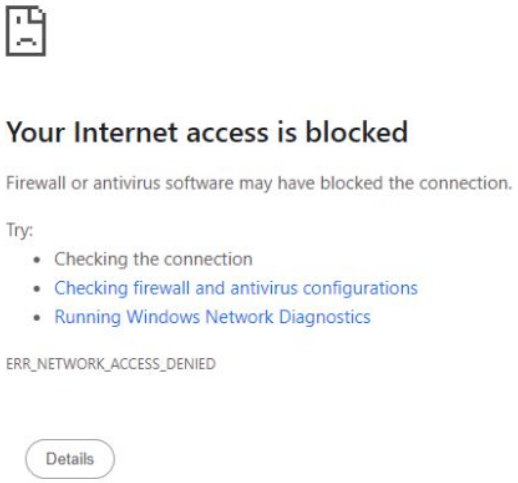


Step 4: Check the rule in the rules list

Name	Group	Profile	Enabled	Action	Override
Chrome Block		All	Yes	Block	No
adb		Private	Yes	Allow	No

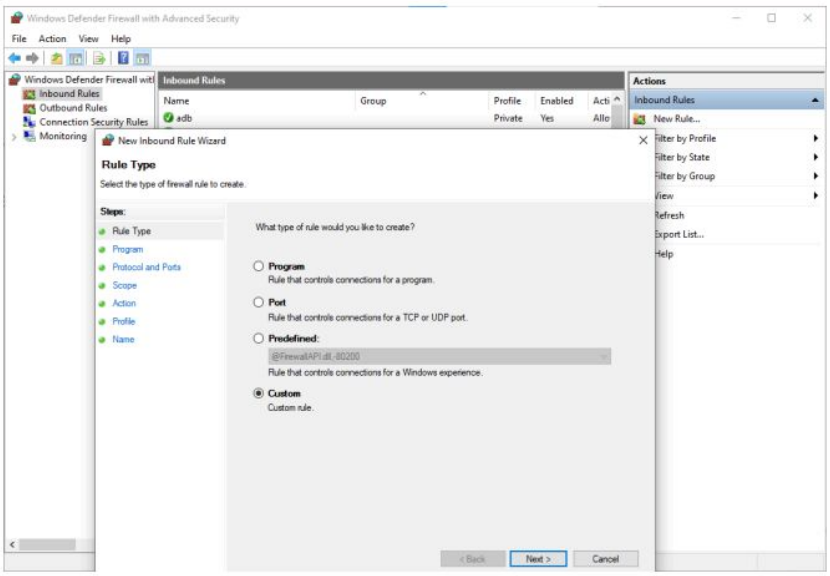
Step 5: Right click on the Outbound rules and follow the above same procedure and create a new rule for program

Step 6: Check the rule working by accessing chrome.exe



C) Website

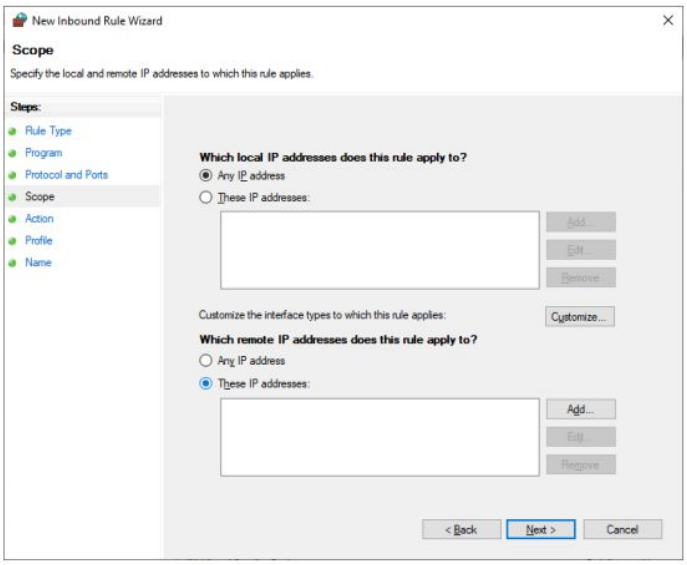
Step 1: Right click on inbound rules and select custom rule



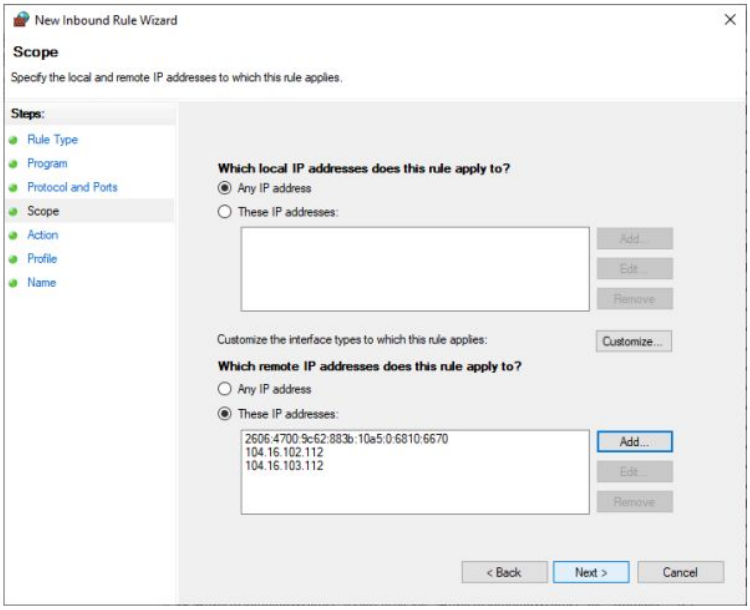
Step 2: open cmd and run the nslookup command for the website with its url (nslookup canva.com) and run the command and note down the addresses

```
C:\Users\user>nslookup canva.com
Server:  UnKnown
Address:  192.168.52.1

Non-authoritative answer:
Name:    canva.com
Addresses:  2606:4700:9c62:883b:10a5:0:6810:6670
          104.16.102.112
          104.16.103.112
```

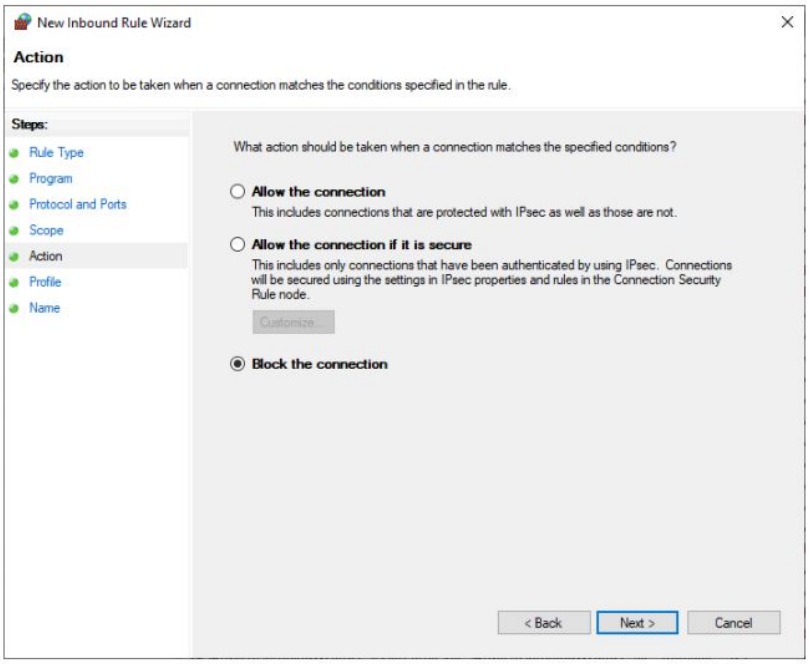
Step 3: Under the custom rule under Scope in the remote IP addresses select these IP addresses



Step 4: Add all the addresses noted from the command line



Step 5: Specify the action



Step 7: Do the same procedure with Outbound rules and create new rule for website and check the rule in rule list

Step 8: Accessing the specified url (canva.com) with block the access