**PRACTICAL NO 1**

**Q.)Write programs to implement the following Substitution Cipher Techniques: -**

* **Caesar Cipher**
* **Monoalphabetic Cipher**

**#CaesarCipher.java**

package ceasercipher;

import java.util.Scanner;

public class CaeserCipher {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

int shift,i,n;

String str;

String str1="";

String str2="";

System.out.println("Enter the plaintext");

str=sc.nextLine();

str=str.toLowerCase();

n=str.length();

char ch1[]=str.toCharArray();

char ch3,ch4;

System.out.println("Enter the value by which each letter of the string is to be shifted");

shift=sc.nextInt();

System.out.println();

System.out.println("Encrypted text is");

for(i=0;i<n;i++)

{

if(Character.isLetter(ch1[i]))

{

ch3=(char)(((int)ch1[i]+shift-97)%26+97);

//System.out.println(ch1[i]+" = "+ch3);

str1=str1+ch3;

}

else if(ch1[i]==' ')

{

str1=str1+ch1[i];

}

}

System.out.println(str1);

System.out.println();

System.out.println("Decrypted text is");

char ch2[]=str1.toCharArray();

for(i=0;i<str1.length();i++)

{

if(Character.isLetter(ch2[i]))

{

if(((int)ch2[i]-shift)<97)

{

ch4=(char)(((int)ch2[i]-shift-97+26)%26+97);

}

else

{

ch4=(char)(((int)ch2[i]-shift-97)%26+97);

}

str2=str2+ch4;

}

else if(ch2[i]==' ')

{

str2=str2+ch2[i];

}

}

System.out.println(str2);

}

}

**Output:**

**#MonoalphabaticCipher.java**

package monoalphabeticcipher;

import java.util.Scanner;

public class MonoAlphabeticCipher {

public static char a[]={'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','q','r','s','t','u','v','w','x','y','z'};

public static char b[]={'D','K','V','Q','F','I','B','J','W','P','E','S','C','X','H','T','M','Y','A','U','O','L','R','G'};

public static String encryption(String s){

int n=s.length();

char c[]=new char[n];

int i;

for (i=0;i<n;i++){

for (int j = 0;j<26;j++){

if (a[j]==s.charAt(i)){

c[i]=b[j];

break;

}

}

}

return (new String(c));

}

public static String decryption(String c){

char d[]=new char[c.length()];

int i;

for (i=0;i<c.length();i++){

for (int j = 0;j<26;j++){

if (b[j]==c.charAt(i)){

d[i]=a[j];

break;

} } }

return (new String(d));

}

public static void main(String[] args) {

MonoAlphabeticCipher mn=new MonoAlphabeticCipher();

Scanner s=new Scanner(System.in);

System.out.println("Enter the plain text message:");

String en= mn.encryption(s .next().toLowerCase());

System.out.println("Encrypted Message:" +en);

System.out.println("Decrypted message of "+en+ "is:" +mn.decryption(en));

}

**PRACTICAL NO 2**

**Q.)Write programs to implement the following Substitution Cipher Techniques: -**

* **Vernam Cipher**
* **Playfair Cipher**

**#VernamCipher.java**

package vermancipher;

import java.io.DataInputStream;

import java.io.IOException;

import java.util.Random;

import static jdk.nashorn.internal.objects.NativeString.toUpperCase;

public class VermanCipher {

public static void main(String[] args) throws IOException {

System.out.println("\nReference - ");

for(int i=1;i<27;i++)

System.out.print((char)(i+64)+"-"+i+" ");

DataInputStream d=new DataInputStream(System.in);

System.out.print("\n\nEnter the Plain Text : ");

String plainText=d.readLine();

plainText=toUpperCase(plainText);

String modPlainText="";

//code to remove the whitespaces

for(int k=0;k<plainText.length();k++){

char c=plainText.charAt(k);

if(c>='A'&&c<='Z'){

modPlainText+=c;

}

}

//Generate Random one-time pad

String oneTimePad="";

String cipher="";

for(int k=0;k<modPlainText.length();k++){

Random rand=new Random();

int randNum=rand.nextInt(26)+1;

oneTimePad+=(char)(randNum+64);

}

System.out.println("\n||\tPlain\t||\tOne-Time\t||\tInitial\t\t||\tMod 27\t\t||\tCipher\t||");

System.out.println("||\tText\t||\tPad\t\t||\tTotal\t\t||\t(if>26)\t\t||\tText\t||");

System.out.println("----------------------------------------------------------------------------------------------------------");

for(int k=0;k<modPlainText.length();k++){

int a=(int)(modPlainText.charAt(k)-64);

int b=(int)(oneTimePad.charAt(k)-64);

if((a+b)>26)

{

System.out.println("||\t"+(char)(a+64)+" ("+a+")\t||\t"+(char)(b+64)+" ("+b+")\t\t||\t"+(a+b)+"\t\t||\t"+((a+b)-26)+"\t\t||\t"+(char)(((a+b)-26)+64)+"\t||");

cipher+=(char)(((a+b)-26)+64);

}

else{

System.out.println("||\t"+(char)(a+64)+" ("+a+")\t||\t"+(char)(b+64)+" ("+b+")\t\t||\t"+(a+b)+"\t\t||\t"+(a+b)+"\t\t||\t"+(char)((a+b)+64)+"\t||");

cipher+=(char)((a+b)+64);

}

}

System.out.println("\nCipher Text : "+cipher);

}

}

**Output:-**

**#PlayFairCipher.java**

package playfaircipher;

import java.io.DataInputStream;

import java.io.IOException;

import static java.lang.Character.toUpperCase;

public class PlayFairCipher {

public static void main(String[] args) throws IOException {

DataInputStream d=new DataInputStream(System.in);

System.out.println("Enter the keyword : ");

String keyword=d.readLine();

//Matrix formation

char[][] matrix=new char[5][5];

boolean keydone=false;

int i=0,j=0;

boolean filled=false;

for(int k=0;k<keyword.length()&&keydone!=true;k++){

filled=false;

char c=keyword.charAt(k);

c=toUpperCase(c);

if(c>='A'&&c<='Z'){

//checking that character is already inserted in matrix or not

for(int m=0;m<5;m++){

for(int n=0;n<5;n++){

if(matrix[m][n]==c)

filled=true;

}

}

if(!filled){

matrix[i][j]=c;

j++;

if(j==5){

i++;

j=0;

}

}

}

if(k==keyword.length()-1)

keydone=true;

}

for(char k='A';k<='Z';k++){

filled=false;

for(int m=0;m<5;m++){

for(int n=0;n<5;n++){

if(matrix[m][n]==k)

filled=true;

}

}

if(!filled){

if(k=='J')

continue;

matrix[i][j]=k;

j++;

if(j==5){

i++;

j=0;

}

}

}

System.out.print("\nTransformation Table\n");

for(i=0;i<5;i++){

for(j=0;j<5;j++){

System.out.print(" "+matrix[i][j]);

}

System.out.print("\n");

}

//Dividing whloe string in two-two character blocks

System.out.println("\nEnter the Plain Text : ");

String plainText=d.readLine();

plainText=plainText.toUpperCase();

String modPlainText="";

int l=0;

char lastChar='.';

for(int k=0;k<plainText.length();k++){

if(l==2)

l=0;

char c=plainText.charAt(k);

if(c>='A'&&c<='Z'){

if(lastChar==c&&l==1){

modPlainText+='X';

lastChar='X';

k--;

}

else{

modPlainText+=c;

lastChar=c;

}

l++;

if(l==2)

modPlainText+=" ";

}

}

if(l==1)

modPlainText+='X';

System.out.println("\nPlain Text : "+modPlainText);

// Encryption process starts here

l=0;

String cipherText="";

int x1=0,y1=0,x2=0,y2=0;

//label to come directly here from inner loops

//label always should be just above the loop

outer:

for(int k=0;k<modPlainText.length();k++){

if(l==2)

l=0;

char c=modPlainText.charAt(k);

if(c=='J')

c='I';

if(c>='A'&&c<='Z'){

for(int m=0;m<5;m++){

for(int n=0;n<5;n++){

if(c==matrix[m][n]){

if(l==0){

x1=m;y1=n;

l++;

continue outer;

}

if(l==1)

{

x2=m;y2=n;

// Replacing plain text with cipher text

if(x1==x2){

y1++; y2++;

if(y1==5)

y1=0;

if(y2==5)

y2=0;

cipherText+=matrix[x1][y1];

cipherText+=matrix[x2][y2];

}

else if(y1==y2){

x1++; x2++;

if(x1==5)

x1=0;

if(x2==5)

x2=0;

cipherText+=matrix[x1][y1];

cipherText+=matrix[x2][y2];

}

else{

cipherText+=matrix[x1][y2];

cipherText+=matrix[x2][y1];

}

l++;

continue outer;

}

}

}

}

}

else

cipherText+=c;

}

System.out.println("\nCipher Text : "+cipherText+"\n");

}

}

**Output:-**

**PRACTICAL NO. 3**

**Q)Write programs to implement the following Transposition Cipher Techniques: -**

* **Rail Fence Cipher**
* **Simple Columnar Technique**

**#RailFenceCipher.java**

package railfencecipher;

import java.util.Scanner;

public class RailFenceCipher {

public static void main(String[] args) {

int i=0,j=0;

String[] output=new String[10];

String ans = null;

Scanner sc = new Scanner(System.in);

System.out.println("Input String ");

String input=sc.next();

System.out.println("enter key");

int key=sc.nextInt();

int len=input.length();

System.out.println("Input String "+input);

for(i=0;i<len;i++)

{

for( j=0;j<key && i<len;j++){

//output[j]=""+input.charAt(i);

if(output[j]==null){

output[j]=""+input.charAt(i);

}

else{

output[j]=output[j]+input.charAt(i);

}

i=i+1;

}

for( j=key-2;j>=1 && i<len;j--){

//output[j]=""+input.charAt(i);

output[j]=output[j]+input.charAt(i);

i=i+1;

}

i=i-1;

}

for(j=0;j<key;j++){

if(j==0){

ans=""+output[j];

}

else{

ans+=output[j];

}

//ans+=output[j];

}

System.out.println("Ciphered Text:"+ans);

}

}

**Output:-**

**#SimpleColumnarCipher.java**

package simplecolumnarcipher;

import java.io.DataInputStream;

import java.io.IOException;

import java.util.Arrays;

import java.util.Scanner;

public class SimpleColumnarCipher {

public static String sortString(String inputString){

char tempArray[]=inputString.toCharArray();

Arrays.sort(tempArray);

return new String(tempArray);

}

public static void main(String[] args) throws IOException{

DataInputStream d=new DataInputStream(System.in);

String plaintext="",key,ciphertext="";

int pos=0;

System.out.print("\nEnter 1 for Encryption and 2 for Decryption : ");

int choice=Integer.parseInt(d.readLine());

if(choice==1){

System.out.print("\nEnter the Plaintext : ");

plaintext=d.readLine();

System.out.print("\nEnter the key : ");

key=d.readLine();

int y=key.length();

int strlen=plaintext.length();

int x=(strlen/y)+1;

char[][] matrix=new char[x][y];

String sorted=sortString(key);

for(int i=0;i<x;i++){

for(int j=0;j<y;j++){

if(pos<strlen){

matrix[i][j]=plaintext.charAt(pos);

pos++;

System.out.print(matrix[i][j]+"\t");

}

else

matrix[i][j]=' ';

}

System.out.println("\n");

}

for(int j=0;j<y;j++){

int k=key.indexOf(sorted.charAt(j));

for(int i=0;i<x;i++){

ciphertext+=matrix[i][k];

}

}

System.out.print("Ciphertext : "+ciphertext);

}

if(choice==2){

System.out.print("\nEnter the Ciphertext : ");

ciphertext=d.readLine();

System.out.print("\nEnter the key : ");

key=d.readLine();

int y=key.length();

int strlen=ciphertext.length();

int x=(strlen/y)+1;

char[][] matrix=new char[x][y];

String sorted=sortString(key);

for(int j=0;j<y;j++){

int k=key.indexOf(sorted.charAt(j));

for(int i=0;i<x;i++){

if(pos<strlen){

matrix[i][k]=ciphertext.charAt(pos);

pos++;

}

else

matrix[i][k]=' ';

}

}

for(int i=0;i<x;i++){

for(int j=0;j<y;j++){

plaintext+=matrix[i][j];

}

}

System.out.println("\nPlaintext : "+plaintext);

} }

}

**Output:**

**PRACTICAL NO 4**

**Q.)Write program to encrypt and decrypt strings using –**

* **DES Algorithm**
* **AES Algorithm**

**#DES.java**

package des;

import java.util.Scanner;

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

class DesEncrypter {

Cipher ecipher;

Cipher dcipher;

DesEncrypter(SecretKey key) throws Exception {

ecipher = Cipher.getInstance("DES");

dcipher = Cipher.getInstance("DES");

ecipher.init(Cipher.ENCRYPT\_MODE, key);

dcipher.init(Cipher.DECRYPT\_MODE, key);

}

public String encrypt(String str) throws Exception {

byte[] utf8 = str.getBytes("UTF8");

byte[] enc = ecipher.doFinal(utf8);

return new sun.misc.BASE64Encoder().encode(enc);

}

public String decrypt(String str) throws Exception {

byte[] dec = new sun.misc.BASE64Decoder().decodeBuffer(str);

byte[] utf8 = dcipher.doFinal(dec);

return new String(utf8, "UTF8");

}

}

public class DES {

public static void main(String[] args) throws Exception {

SecretKey key = KeyGenerator.getInstance("DES").generateKey();

DesEncrypter encrypter = new DesEncrypter(key);

System.out.print("Enter text: ");

Scanner sc =new Scanner(System.in);

String encrypted = encrypter.encrypt(sc.nextLine());

System.out.println("Encrypted text: "+encrypted);

String decrypted = encrypter.decrypt(encrypted);

System.out.println("Decrypted text: "+decrypted);

}

}

**Output:-**

**#AES.java:**

package aes;

import java.util.Scanner;

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import javax.xml.bind.DatatypeConverter;

public class AES {

public static void main(String[] args) throws Exception {

System.out.print("Enter text: ");

Scanner sc = new Scanner(System.in);

String plainText = sc.nextLine();

SecretKey secKey = getSecretEncryptionKey();

byte[] cipherText = encryptText(plainText, secKey);

String decryptedText = decryptText(cipherText, secKey);

System.out.println("AES Key (Hex Form):"+bytesToHex(secKey.getEncoded()));

System.out.println("Encrypted Text (Hex Form):"+bytesToHex(cipherText));

System.out.println("Descrypted Text:"+decryptedText);

}

public static SecretKey getSecretEncryptionKey() throws Exception{

KeyGenerator generator = KeyGenerator.getInstance("AES");

generator.init(128); // The AES key size in number of bits

SecretKey secKey = generator.generateKey();

return secKey;

}

public static byte[] encryptText(String plainText,SecretKey secKey)

throws Exception{

Cipher aesCipher = Cipher.getInstance("AES");

aesCipher.init(Cipher.ENCRYPT\_MODE, secKey);

byte[] byteCipherText = aesCipher.doFinal(plainText.getBytes());

return byteCipherText;

}

public static String decryptText(byte[] byteCipherText, SecretKey secKey) throws Exception {

Cipher aesCipher = Cipher.getInstance("AES");

aesCipher.init(Cipher.DECRYPT\_MODE, secKey);

byte[] bytePlainText = aesCipher.doFinal(byteCipherText);

return new String(bytePlainText);

}

private static String bytesToHex(byte[] hash) {

return DatatypeConverter.printHexBinary(hash);

}

}

**Output:-**

**PRACTICAL NO 5**

**Q.)Write a program to implement RSA algorithm to perform encryption / decryption of a given string.**

**#RSA.java**

package rsa;

import java.io.DataInputStream;

import java.io.IOException;

import java.math.BigInteger;

import java.util.Random;

public class RSA {

private BigInteger p;

private BigInteger q;

private BigInteger N;

private BigInteger phi;

private BigInteger e;

private BigInteger d;

private int bitlength = 1024;

private Random r;

public RSA()

{

r = new Random();

p = BigInteger.probablePrime(bitlength, r);

q = BigInteger.probablePrime(bitlength, r);

N = p.multiply(q);

phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));

e = BigInteger.probablePrime(bitlength / 2, r);

while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0)

{

e.add(BigInteger.ONE);

}

d = e.modInverse(phi);

}

public RSA(BigInteger e, BigInteger d, BigInteger N)

{

this.e = e;

this.d = d;

this.N = N;

}

public static void main(String[] args) throws IOException {

RSA rsa = new RSA();

DataInputStream in = new DataInputStream(System.in);

String teststring;

System.out.println("Enter the plain text:");

teststring = in.readLine();

System.out.println("Encrypting String: " + teststring);

System.out.println("String in Bytes: "

+ bytesToString(teststring.getBytes()));

// encrypt

byte[] encrypted = rsa.encrypt(teststring.getBytes());

System.out.println("Encrypted String : "+encrypted);

// decrypt

byte[] decrypted = rsa.decrypt(encrypted);

System.out.println("Decrypting Bytes: " + bytesToString(decrypted));

System.out.println("Decrypted String: " + new String(decrypted));

}

private static String bytesToString(byte[] encrypted)

{

String test = "";

for (byte b : encrypted)

{

test+=Byte.toString(b);

}

return test;

}

// Encrypt message

public byte[] encrypt(byte[] message)

{

return (new BigInteger(message)).modPow(e, N).toByteArray();

}

// Decrypt message

public byte[] decrypt(byte[] message)

{

return (new BigInteger(message)).modPow(d, N).toByteArray();

}

}

**Output:-**

**PRACTICAL NO. 6**

**Q.)Write a program to implement the Diffie-Hellman Key Agreement algorithm to generate symmetric keys.**

**Code:**

package diffie\_hellman;

import java.util.Scanner;

public class Diffie\_Hellman {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

System.out.println("Enter the First prime number(g): ");

int g=sc.nextInt();

System.out.println("Enter the Second prime number(n)");

int n=sc.nextInt();

System.out.println("Choose 1st secret no(Alice)");

int a=sc.nextInt();

System.out.println("Choose 2nd secret no(BOB)");

int b=sc.nextInt();

int A = (int)Math.pow(n,a)%g;

int B = (int)Math.pow(n,b)%g;

int S\_A = (int)Math.pow(B,a)%g;

int S\_B =(int)Math.pow(A,b)%g;

if(S\_A==S\_B)

{

System.out.println("Alice and Bob can communicate with each other!!!");

System.out.println("They share a secret no = "+S\_A);

}

else

{

System.out.println("ALice and Bob cannot communicate with each other!!!");

}

}

}

**PRACTICAL NO 7**

**Q.)Write a program to implement the MD5 algorithm compute the message digest.**

**Code:**

package md5;

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.Scanner;

public class MD5 {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

System.out.print("Enter a text : ");

String i=sc.nextLine();

System.out.println(md5(i));

System.out.println("For null :" + md5(""));

System.out.println("For simple text :"+ md5("This is my text"));

System.out.println("For simple numbers : " + md5("12345"));

}

public static String md5(String input) {

String Md5 = null;

if(null == input)

return null;

try {

//Create MessageDigest object for MD5

MessageDigest digest = MessageDigest.getInstance("MD5");

//Update input string in message digest

digest.update(input.getBytes(), 0, input.length());

//Converts message digest value in base 16 (hex)

Md5 = new BigInteger(1, digest.digest()).toString(16);

}

catch (NoSuchAlgorithmException e) {

e.printStackTrace();

}

return Md5;

}

}

**PRACTICAL NO 8**

**Q.)Write a program to calculate HMAC-SHA1 Signature**

**Code:**

package hmac.sha1;

import java.io.DataInputStream;

import java.io.IOException;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import java.security.SignatureException;

import javax.crypto.Mac;

import javax.crypto.spec.SecretKeySpec;

import javax.xml.bind.DatatypeConverter;

public class HMACSHA1 {

public static byte[] calculateRFC2104HMAC(String data, String key)

throws SignatureException, NoSuchAlgorithmException, InvalidKeyException

{

SecretKeySpec signingKey = new SecretKeySpec(key.getBytes(), "HmacSHA1");

Mac mac = Mac.getInstance("HmacSHA1");

mac.init(signingKey);

byte[] hex = mac.doFinal(data.getBytes());

return hex;

}

public static void main(String[] args) throws IOException, SignatureException, NoSuchAlgorithmException, InvalidKeyException {

System.out.print("Enter plain text: ");

DataInputStream ds=new DataInputStream(System.in);

String plain=ds.readLine();

System.out.print("Enter key text: ");

String key=ds.readLine();

byte[] hmac = calculateRFC2104HMAC(plain, key);

System.out.println(DatatypeConverter.printHexBinary(hmac));

}

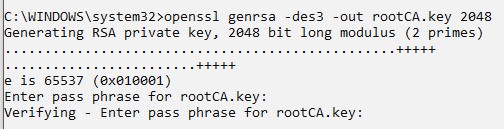
}

**Practical -9**

**Write a program to implement SSL.**

It takes two terminal commands to generate a root certificate. The first command is to create a private key. This can be accomplished by running the following command:

C:\WINDOWS\system32>**openssl genrsa -des3 -out rootCA.key 2048**



This creates a key, 2048 bits long, The -des3 parameter species to use the Tripple DES algorithm to encrypt the key and will require you to enter a password in order for the key le to be created. Be sure to remember the password you enter or you will have to generate a new key.

Now to generate the root certificate:

C:\WINDOWS\system32> **openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.pem**

**Description of parameters:**

-new: create a new request

-nodes: don’t encrypt the output key

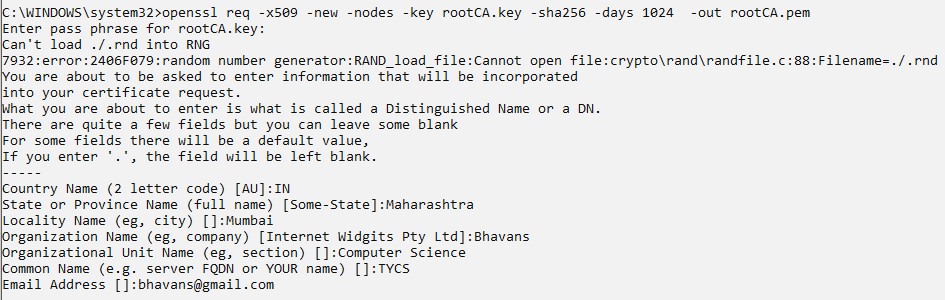
-x509: species the kind of certificate to make -key: the le with the private key to use

-sha256: this is the hashing algorithm. When you omit this it will default to the SHA1 algorithm which will result in the browser generating a warning

-days: the number of days the certificate should be valid for. Use as high a number as you feel comfortable with for your development environment

-out: the name of the le to write the certificate to

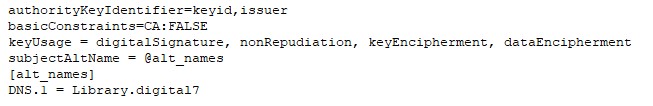
When you run the command, you will be asked to provide some information. This will be included in the certificate and is public information. I used to the following to create the certificate:



# Creating the SSL certificate

With the root certificate covered it is time to generate the SSL certificate. Because only an X.509 v3 certificate carries SAN information it requires a little more work than when creating an X.509 v1 certificate.

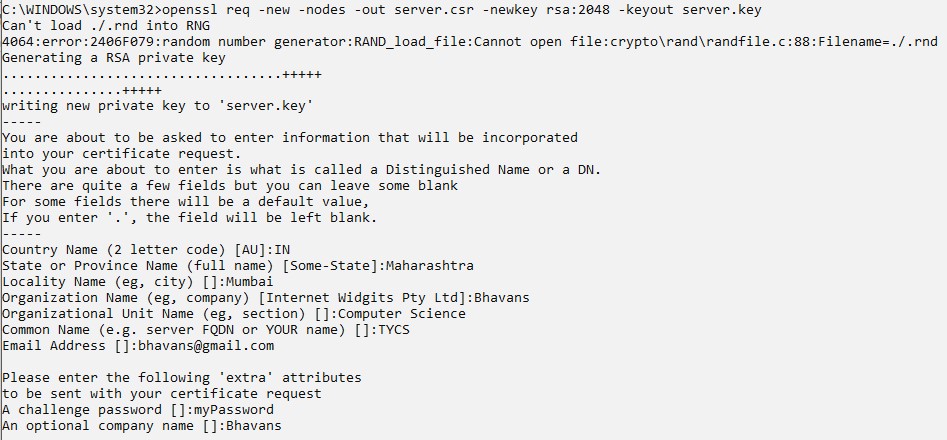
The extra work is the additional le needed with the v3 information. I’ve created a file named v3.ext and put the following information in there:



It is the alt\_names section which deserves most of the attention. This is where the domains can be specified for which the certificate is valid. I need this one domain (Library.digital7) but you could add an extra entry.

With this le created and stored in the same folder as the root CA key and certificate it is time to head back to the terminal. The first step is to create a private key for the SSL certificate and a certificate signing request. These two tasks can be combined into a single command:

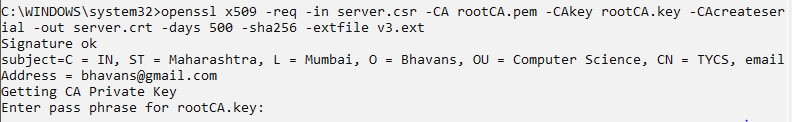
C:\WINDOWS\system32>**openssl req -new -nodes -out server.csr -newkey rsa:2048 -keyout server.key**



It is quite similar to the command used to create the root certificate. The -newkey and -keyout are to specify the kind of private key to generate, and the file to store it in.

Now that a private key and certificate signing request have been created it is possible to issue the certificate with the previously generated root certificate.

C:\WINDOWS\system32>**openssl x509 -req -in server.csr -CA rootCA.pem -CAkey rootCA.key CAcreateserial -out server.crt -days 500 -sha256 -extfile v3.ext**



# Preparing the certificate for IIS

IIS needs the SSL certificate along with the private key in order to be able to use the certificate. Right now I’ve created a server.key and a server.crt file and these need to be combined into a single file. This can be accomplished with the following terminal command:

C:\WINDOWS\system32>**openssl pkcs12 -inkey server.key -in server.crt -export -out server.pfx**

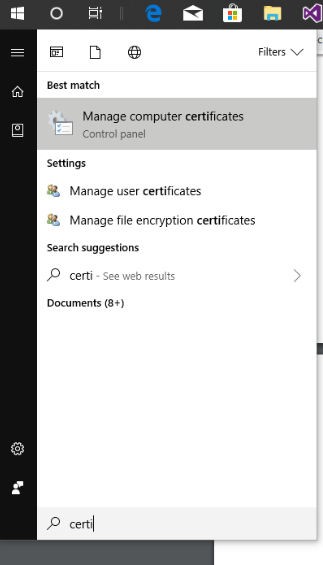


When the command is executed it will ask for an export password, this will be needed again when importing the resulting server.pfx into the windows certificate store.

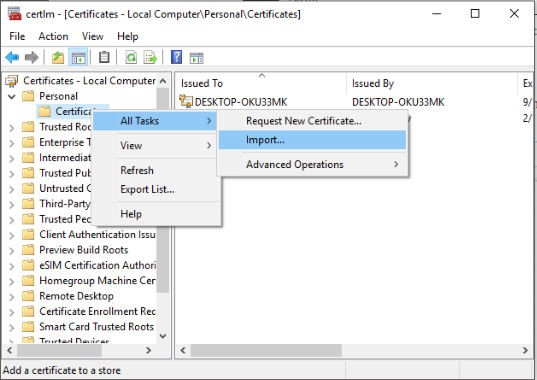
With this command executed all the keys and certificates to get a fully functioning SSL certificate are generated. All that is left to do is importing the certificates and configuring IIS.

# Configuring the Windows certificate store

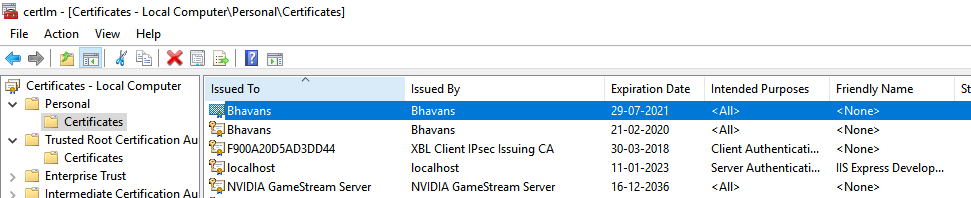
In order to be able to use the certificate for the website, the certificate need to be imported into the Windows certificate store. My virtual machine runs Windows 10, it may work a little certificate on other versions. When you open the start menu in Windows 10 and you type “certificates”, Windows comes up with two relevant suggestions: “Manage computer certificates” and “Manage user certificates”. Both will be needed to install the SSL certificate.



The window for managing the computer certificates looks something like this:

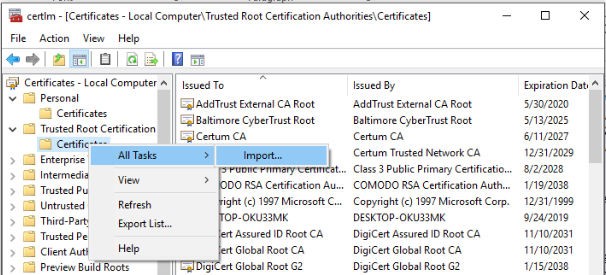


When the context menu for Personal is accessed there is an option Import… under All Tasks. Selecting this item will start a wizard to select and import a certificate. In this certificate store both the rootCA.pem and server.pfx certificate need to be imported. By importing server.pfx the SSL certificate becomes selectable in IIS, importing rootCA.pem will stop IIS from generating warnings the certificate chain is not complete.

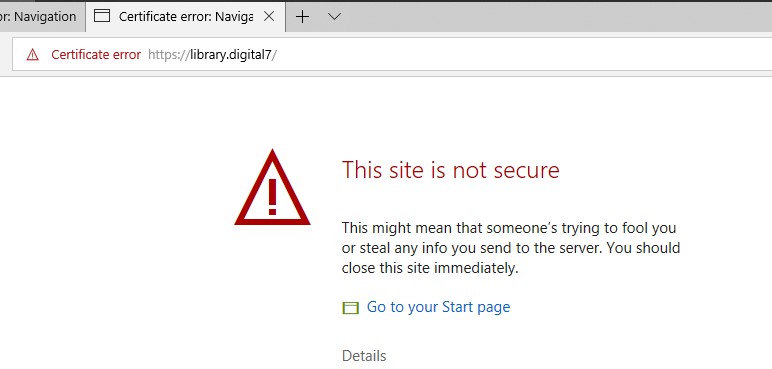


With both certificates installed they will be listed in the application. More importantly, it is now possible to select them in IIS when creating an HTTPS binding and not get any warning messages from IIS In order to inform Windows it can trust certificates issued with the self created root certificate, the root certificate should be imported under personal certificates. This application looks the same as the one for managing the computer certificates. The big difference is the location where the root certificate should be imported into: Trusted Root Certification Authorities.

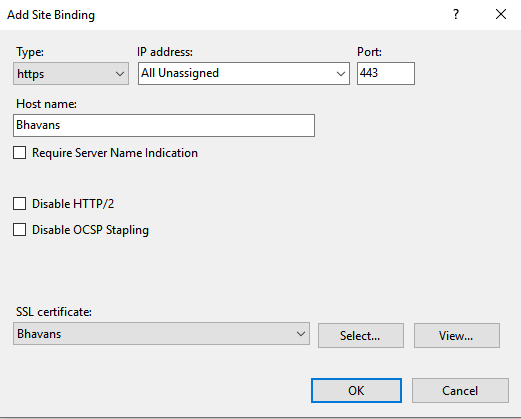
Importing the rootCA.pem certificate in this location informs that accepting an CA certificate from an unknown origin is dangerous and to make sure the certificate is actually legit.

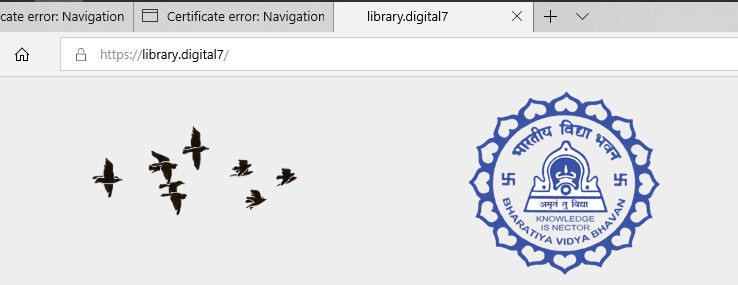


# Before adding SSL certificate



# After adding the SSL certificate



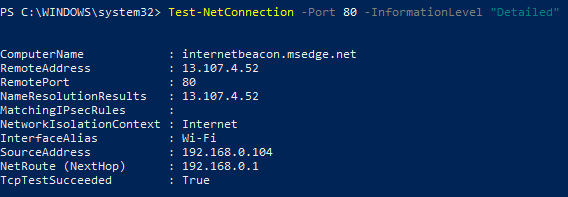


Practical 10

Configure Windows Firewall to block: (a)A port (b) A Program (c) A website.

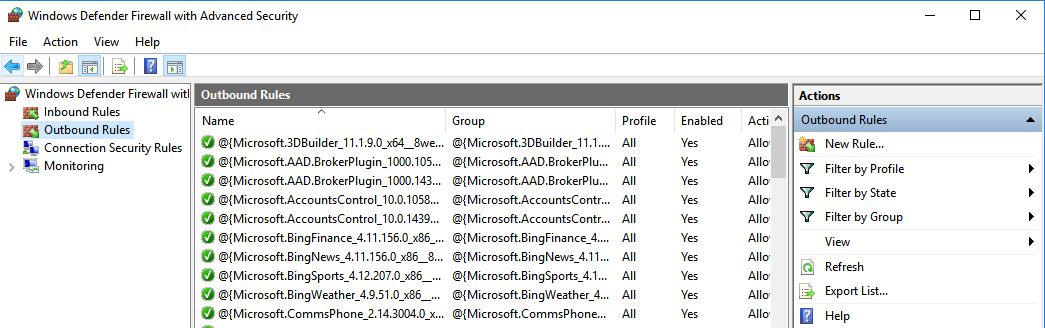
**(a) Blocking a port**

Before blocking port first test it.



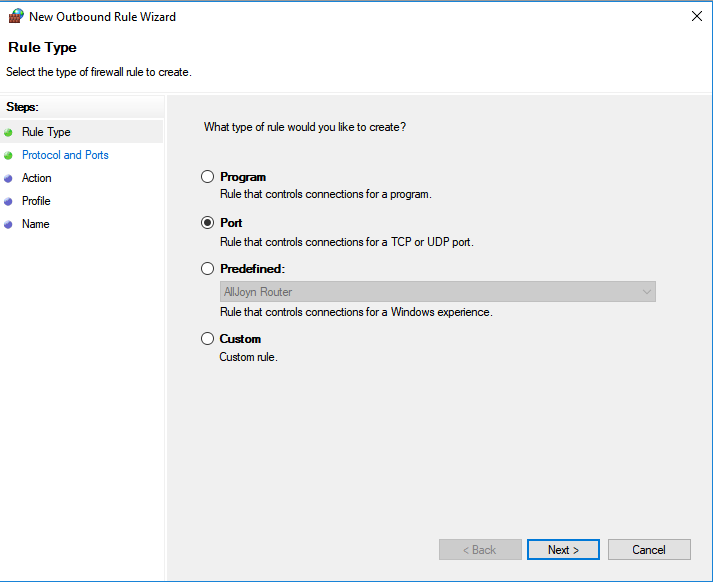
1.Open Windows Defender Firewall with Advanced Security

2. Firewall window shows a list of rules on the left side. From the list, select Outbound Rules to display the Outbound rules section.

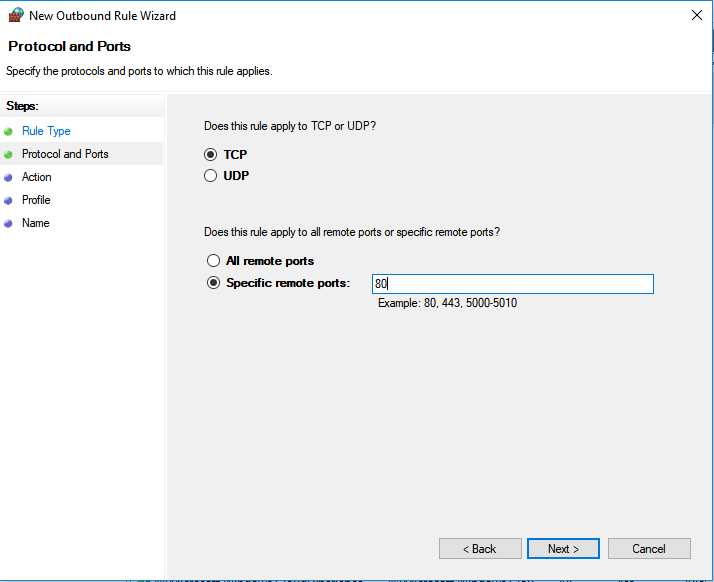


3. From the right pane select the ‘New Rule’ option.

4. This will open the ‘New Outbound Rule Wizard’ window. Select ‘Port’ as the new Rule Type and click Next.

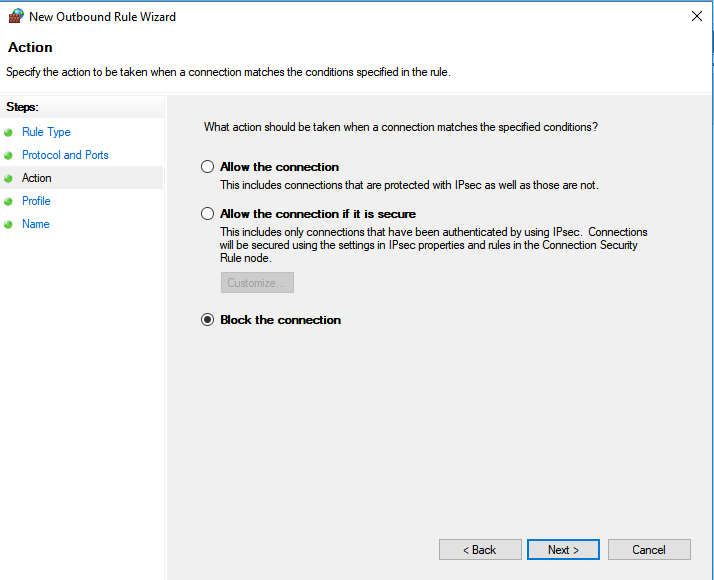


5. Here we are blocking a TCP port hence select TCP radiobutton. Click on specific local ports. Then choose port 80 as shown in the screenshot below.

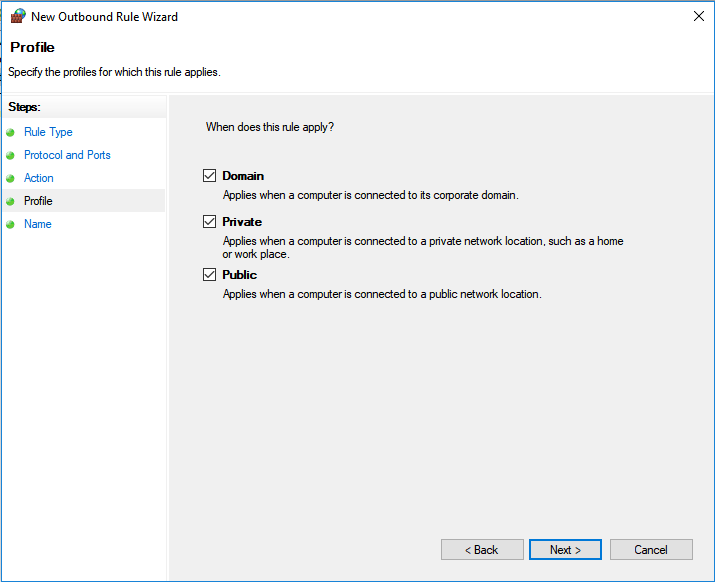


6. Click Next to continue.

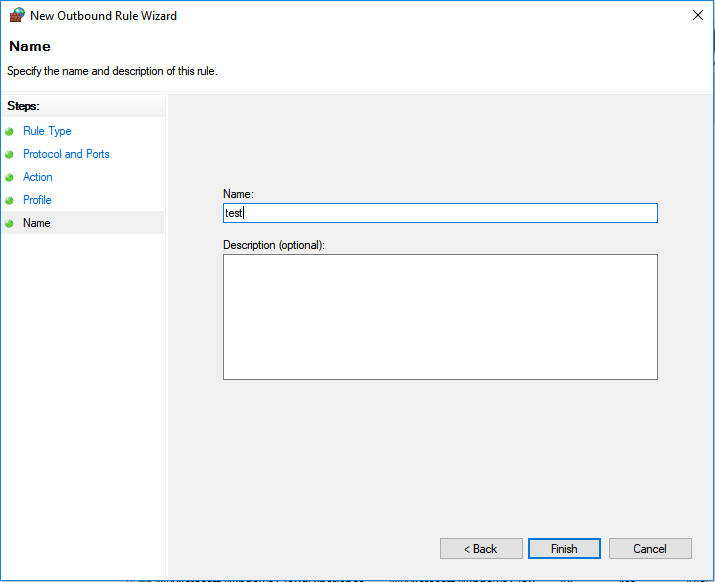
7. Next, select ‘Block the connection’ as the Action and click Next.



8. Select all the profiles available for different type of connections (Domain, Private and Public) and Click Next to continue.



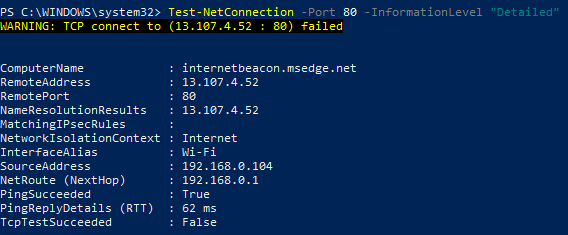
9. Give a name of your choice to the new rule. If you want, you can add the description to the new rule. Description is optional.



10.Finally, click the Finish button.

11. Repeat all the steps above for inbound rule to block port from accepting connections.

To test if port 80 is blocked or not:

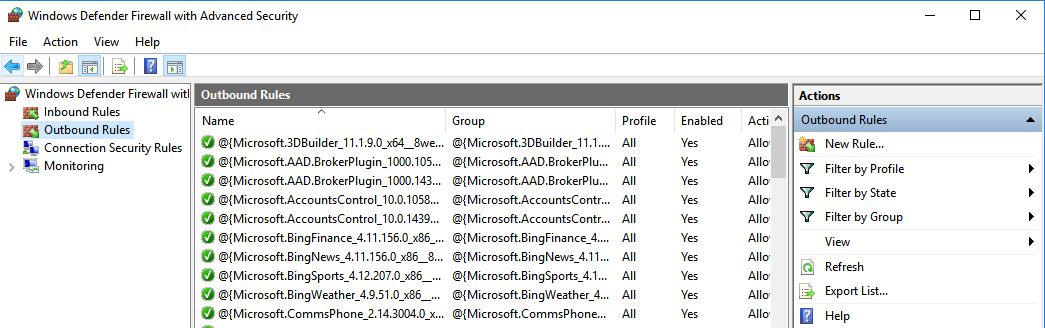


**(b) Blocking a program**

**We will block Google Chrome browser from accessing to internet.**

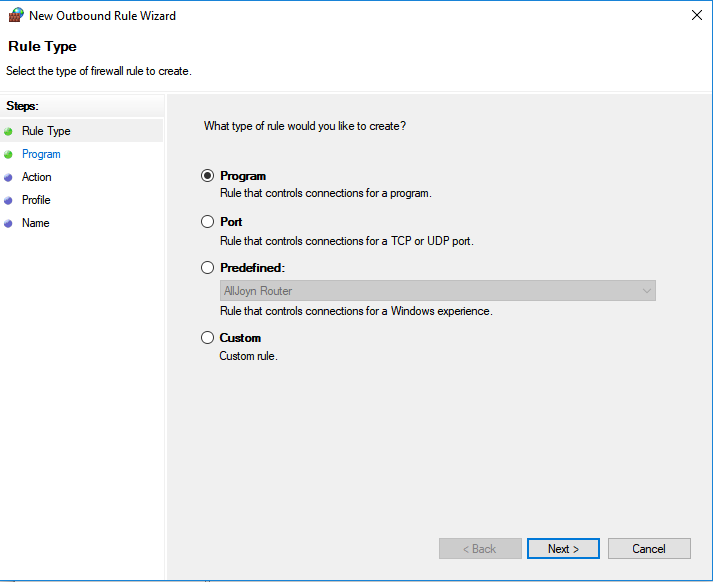
1.Open Windows Defender Firewall with Advanced Security

2. Firewall window shows a list of rules on the left side. From the list, select Outbound Rules to display the Outbound rules section.

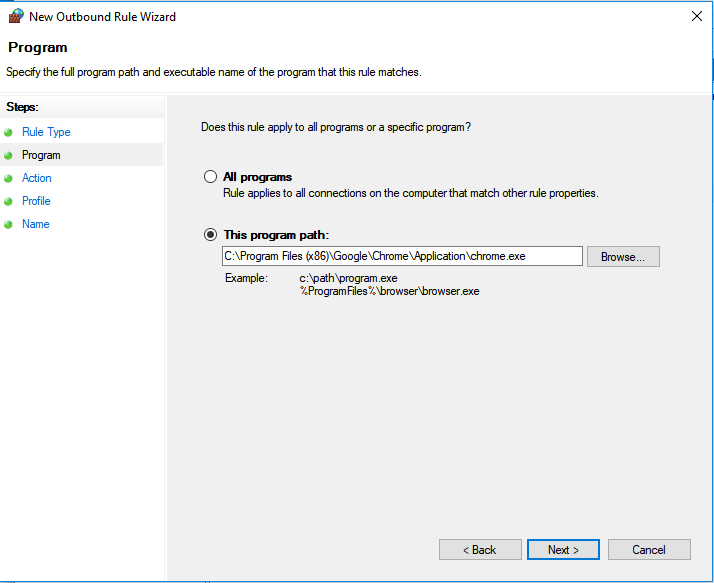


3. From the right pane select the ‘New Rule’ option.

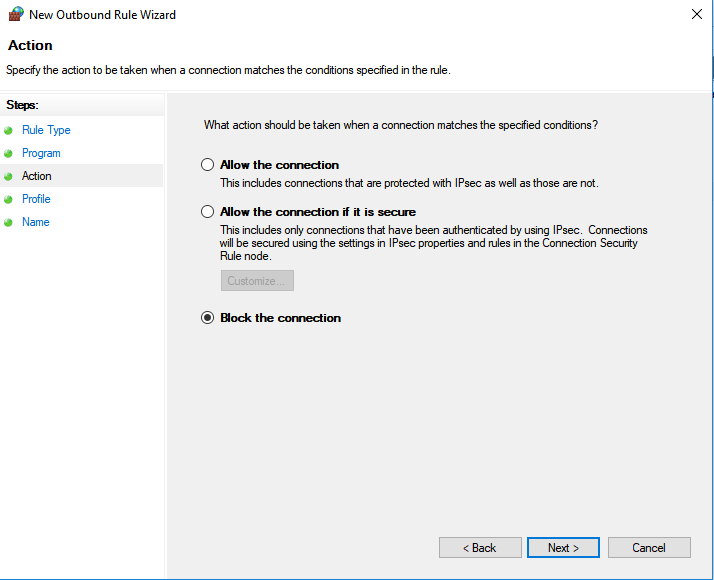
4.This will open the ‘New Outbound Rule Wizard’ window. Select ‘Program’ as the new Rule Type and click Next.



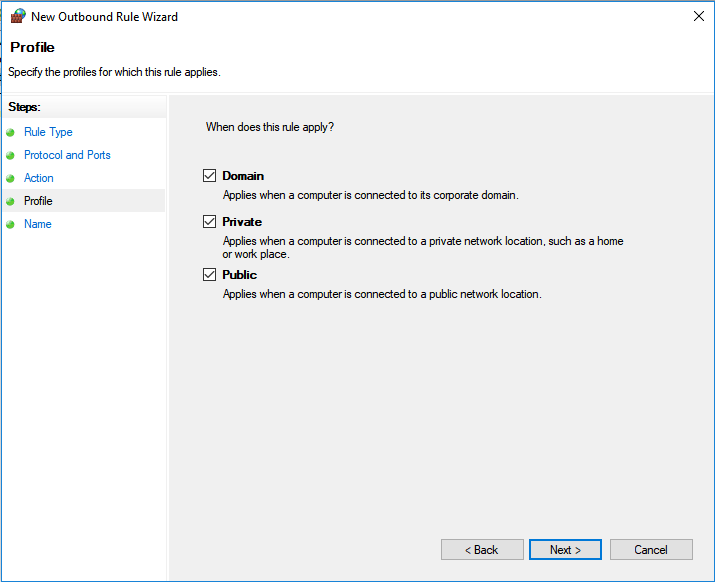
5. Select This program path radiobutton and give full path to the program you want to block and click on Next.



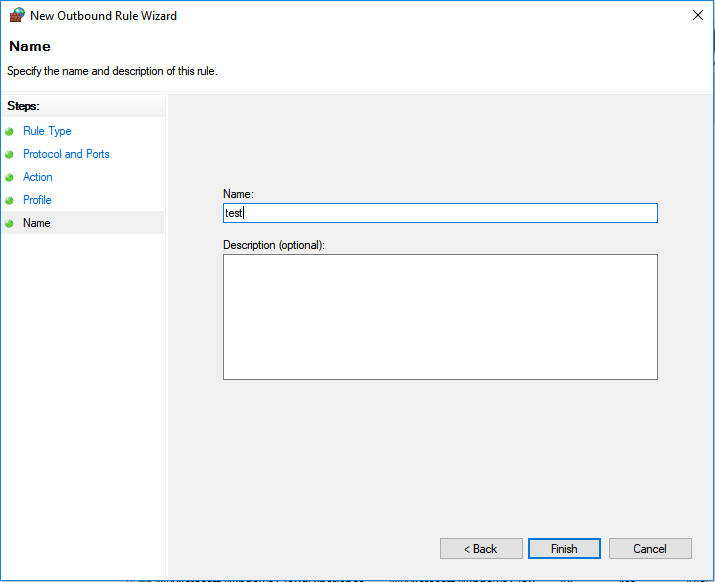
6. Select Block the Connection and click on Next.



7. Select all the profiles available for different type of connections (Domain, Private and Public) and Click Next to continue.



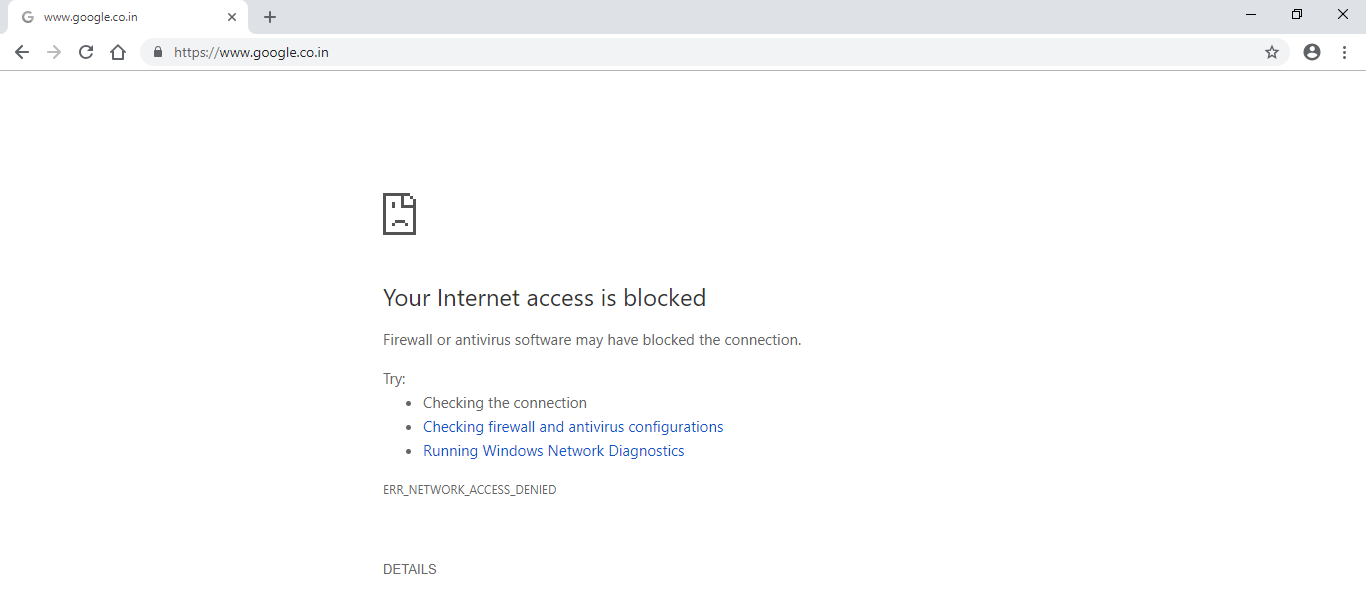
8. Give a name of your choice to the new rule. If you want, you can add the description to the new rule. Description is optional.



9. Finally, click the Finish button.

10. Repeat all the steps above for inbound rule to block a program.

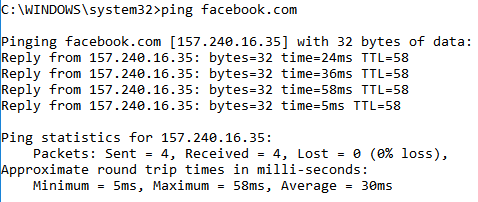
**Testing if Chrome is getting Internet access:**



**(c) Blocking a website**

**We will block access to facebook.com**

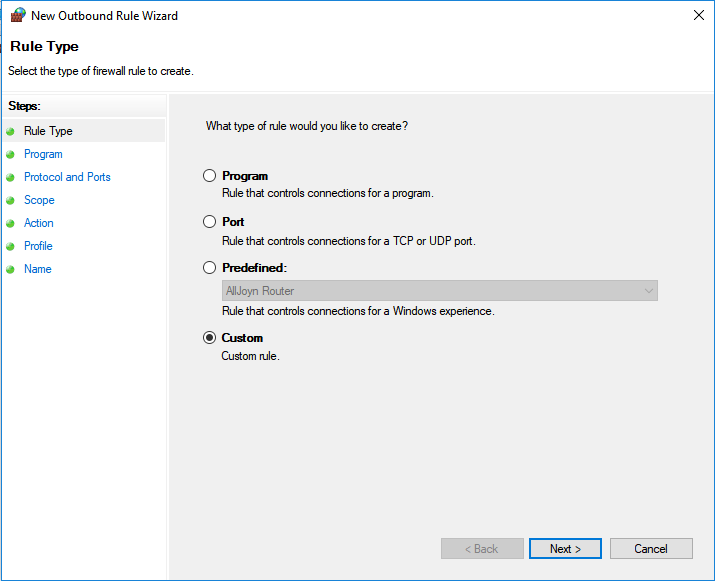
1. For blocking access to any website IP address of that site is required. We can get it by using ping command as follows:



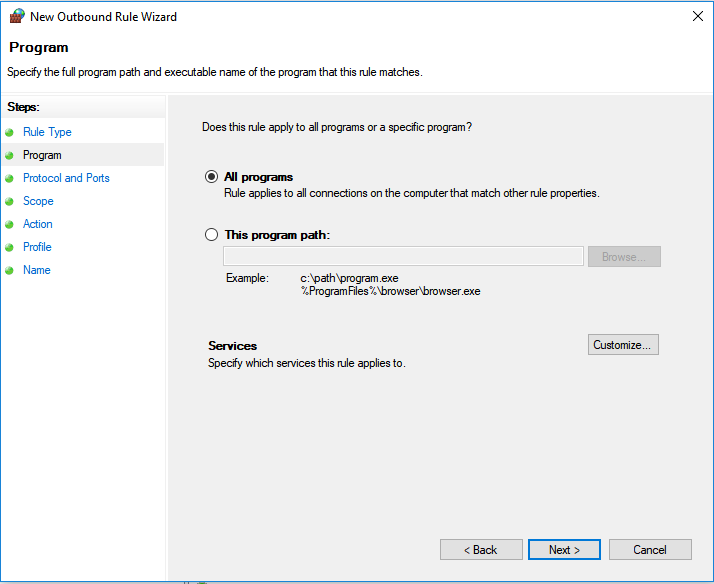
2. Open Windows Defender Firewall with Advanced Security

3. Firewall window shows a list of rules on the left side. From the list, select Outbound Rules to display the Outbound rules section.

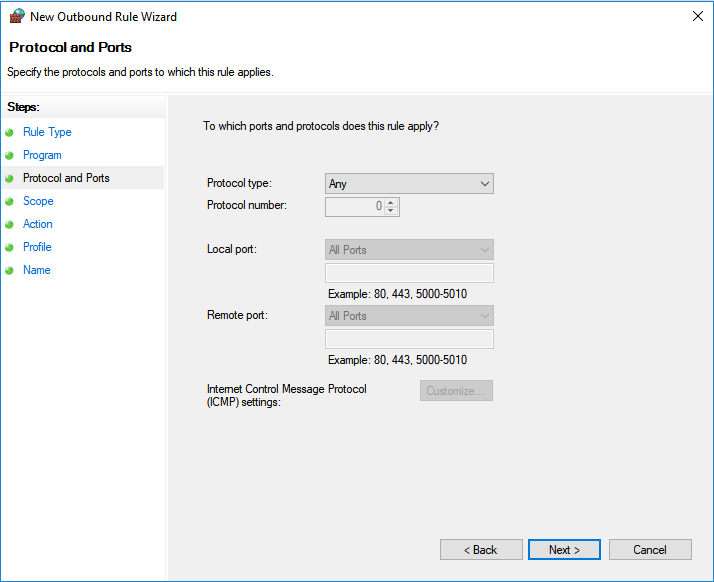
4. This will open the ‘New Outbound Rule Wizard’ window. Select ‘Custom’ as the new Rule Type and click Next.



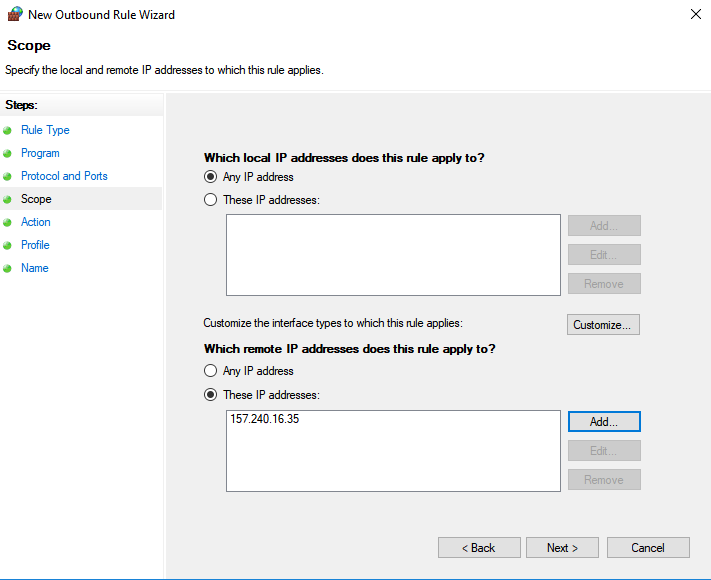
5. All Programs should be selected and click Next.



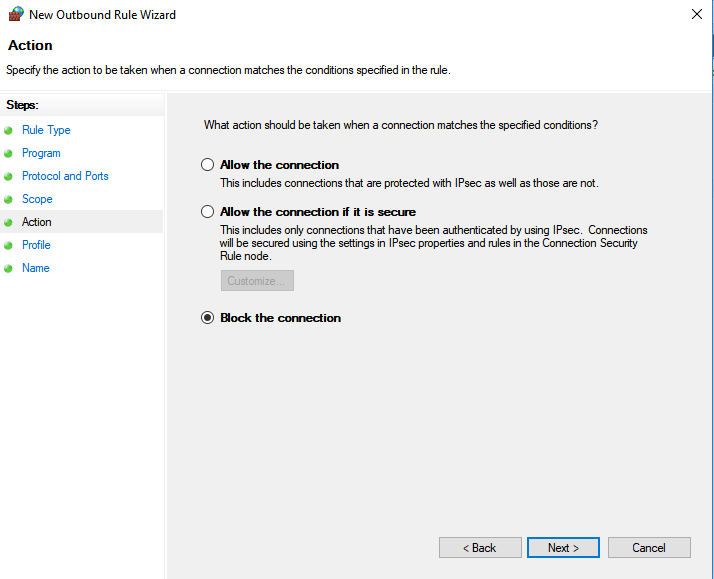
6. Click next on the Protocols and Ports page (leave as it is).



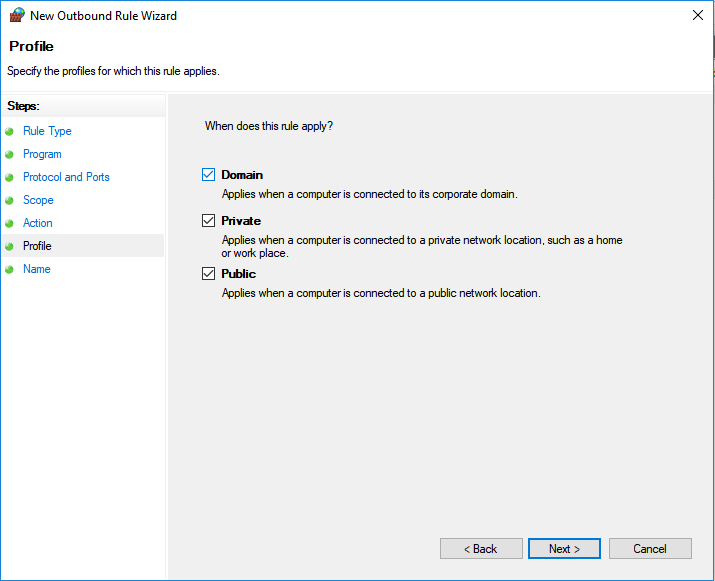
7. In the scope page enter the IP address of the site you want to block in the "Which remote IP addresses does this rule apply to?" section and click on Next.



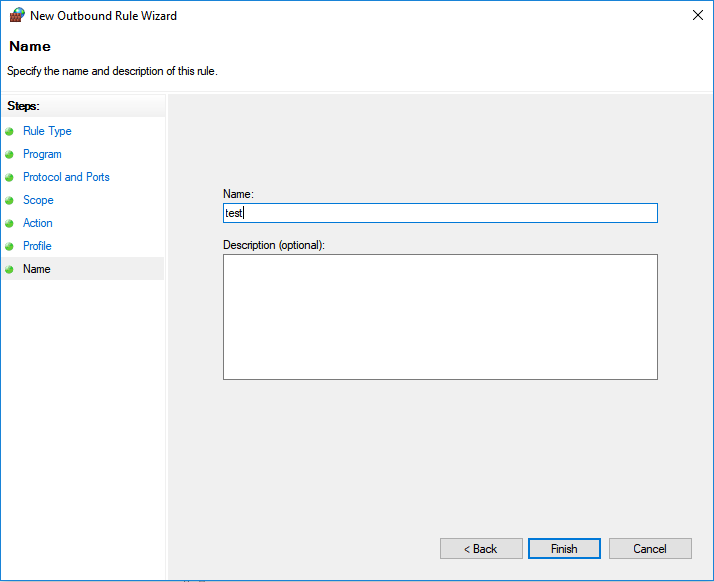
8. Select Block the connection on the Action page, click Next.



9. Leave all boxes checked on Profile page if you don't ever want to see the website on any network connection, modify if you want to allow on a Domain, Public or Private Network and click Next.



10. Enter a name for the new rule you created and click Finish.



11. Repeat all the steps above for inbound rule to block a website.

**Checking if facebook.com can be accessed or not:**

