Ex.no.: 1(a) **CEASER CIPHER**

Date:

Aim:

To implement symmetric key encryption using Ceaser cipher substitution method.

Algorithm:

1. Assign the 26 letters in alphabet to the variable named ALPHABET.
2. Convert the plaintext letters into lowercase.
3. To encrypt a plaintext letter, the first set of plaintext letters and slides it to LEFT by the number of positions of the secret shift.
4. The plaintext letter is then encrypted to the ciphertext letter on the sliding ruler underneath.
5. On receiving the ciphertext, the receiver who also knows the secret shift, positions his sliding ruler underneath the ciphertext alphabet and slides it to RIGHT by the agreed shift number, 3 in this case.
6. Then replaces the ciphertext letter by the plaintext letter on the sliding ruler underneath.

Program:

```
import java.util.Scanner;
public class ceasercipher
{
public static final String ALPHABET="abcdefghijklmnopqrstuvwxyz";
public static String encrypt(String plainText,int shiftKey)
{
plainText=plainText.toLowerCase();
String cipherText="";
for (int i=0; i<plainText.length();i++)
{
int charPosition=ALPHABET.indexOf(plainText.charAt(i));
int keyVal=(shiftKey+charPosition)%26;
char replaceVal=ALPHABET.charAt(keyVal);
cipherText+=replaceVal;
}
return cipherText;
}
public static String decrypt(String cipherText,int shiftKey)
{
cipherText = cipherText.toLowerCase();
String plainText = "";
for(int i=0;i<cipherText.length();i++)
{
```

```
int charPosition= ALPHABET. indexOf(cipherText. charAt(i));
int keyVal=(charPosition-shiftKey)%26;
if (keyVal< 0)
{
keyVal=ALPHABET.length()+keyVal;
}
char replaceVal=ALPHABET.charAt(keyVal);
plainText+=replaceVal;

}
return plainText;
}
public static void main(String[] args)
{
Scanner sc=new Scanner(System.in);
System.out.println("Enter the Plain text for Encryption: ");
String message=new String();
message=sc.next();
System.out.println("Encrypted message:Cipher Text="+encrypt(message,3));
System.out.println("Decrypted message:Plain Text="+decrypt (encrypt(message,3),3));
sc.close();
}
}
```

OUTPUT:

```
F:\bin>javac ceasercipher.java
F:\bin>java ceasercipher
Enter the Plain text for Encryption:
covid
Encrypted message:Cipher Text=frylg
Decrypted message:Plain Text=covid
```

Result:

Thus, the program to implement symmetric key encryption using Ceaser cipher substitution method was executed successfully and the output was verified.

Ex.no.: 1(b) **PLAYFAIR CIPHER**

Date:

Aim:

To implement symmetric key encryption using playfair cipher substitution method.

Algorithm:

1. Read the keyword.
2. Then create the key table of 5x5 grid of alphabets.
3. Read the word to encrypt.
4. If the input word should be even and then process it.
5. Then the plaintext message is split into pairs of two letters (digraphs).
6. If both the letters are in the same column, take the letter below each one.
7. If both letters are in the same row, take the letter to the right of each one.
8. If neither of the preceding two rules are true, form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.

Program:

```
import java.util.Scanner;
public class Playfair1
{
public static void main(String[] args)
{
Scanner in=new Scanner(System.in);
System.out.print("Enter keyword: ");
 String key=in.nextLine();
System.out.print("Enter message to encrypt: ");
String msg=in.nextLine();
PFEncryption pfEncryption = new PFEncryption();
pfEncryption.makeArray(key);
msg=pfEncryption.manageMessage(msg);
pfEncryption.doPlayFair(msg, "Encrypt");
String en=pfEncryption.getEncrypted();
System.out.println("Encrypting. .. \n\nThe encrypted text is: " + en);
System.out.println("==============================");
pfEncryption.doPlayFair(en, "Decrypt");
System.out.print("\nDecrypting... \n\nThe encrypted text is: " + pfEncryption.getDecrypted());
```

```
}
}
class PFEncryption
{
private char [][] alphabets= new char[5][5];
private char[] uniqueChar= new char[26];
private String ch="ABCDEFGHIKLMNOPQRSTUVWXYZ";
private String encrypted="";
private String decrypted="";
void makeArray(String keyword)
{
keyword=keyword.toUpperCase().replace("J","I");
boolean present, terminate=false;
int val=0;
int uniqueLen;
for (int i=0; i<keyword.length(); i++)
{
present=false;
uniqueLen=0;
if (keyword.charAt(i)!= ' ')
{
for (int k=0; k<uniqueChar.length; k++)
{
if (Character.toString(uniqueChar[k])==null)
{
break;
}
uniqueLen++;
}
for (int j=0; j<uniqueChar.length; j++)
{
if (keyword.charAt(i)==uniqueChar[j])
{
present=true;
}
}
if (!present)
{
uniqueChar[val]=keyword.charAt(i);
val++;
}
}
ch=ch.replaceAll(Character.toString(keyword.charAt(i)), "");
}
for (int i=0; i<ch.length(); i++)
{
```

```
uniqueChar[val]=ch.charAt(i);
val++;
}
val=0;
for (int i=0; i<5; i++)
{
for (int j=0; j<5; j++)
{
alphabets[i][j]=uniqueChar[val];
val++;
System.out.print(alphabets[i][j] + "\t");
}
System.out.println();
}
}
String manageMessage(String msg)
{
int val=0;
int len=msg.length()-2; String newTxt="";
String intermediate="";
while (len>=0)
{
intermediate=msg.substring(val, val+2);
if (intermediate.charAt(0)==intermediate.charAt(1))
{
newTxt=intermediate.charAt(0) + "x" + intermediate.charAt(1);
msg=msg.replaceFirst(intermediate,  newTxt);
len++;
}
len-=2;
val+=2;
}
if (msg.length()%2!=0)
{
msg=msg+'x';
}
return msg.toUpperCase().replaceAll("J","I").replaceAll(" ","");
}
void doPlayFair(String msg, String tag)
{
int val=0;
while (val<msg.length())
{
searchAndEncryptOrDecrypt(msg.substring(val,val+2),tag);
```

```
val+=2;
}}
void searchAndEncryptOrDecrypt(String doubblyCh, String tag)
{
char ch1=doubblyCh.charAt(0);
char ch2=doubblyCh.charAt(1);
int row1=0, col1=0, row2=0, col2=0;
for (int i=0; i<5; i++)
{
for (int j=0; j<5; j++)
{
if (alphabets[i][j]==ch1)
{
row1=i;
col1=j;
}
else if (alphabets[i][j]==ch2)
{
row2=i;
col2=j;
}       }    }
if (tag=="Encrypt")
encrypt(row1, col1, row2, col2);
else if(tag=="Decrypt")
decrypt(row1, col1, row2, col2);
}
void encrypt(int row1, int col1, int row2, int col2)
{
if (row1==row2)
{
col1=col1+1; col2=col2+1; if (col1>4)
col1=0;
 if (col2>4)
col2=0;
encrypted+=(Character.toString(alphabets[row1][col1])+ Character.toString(alphabets[row1][col2]));
}
else if(col1==col2)
{
row1=row1+1; row2=row2+1; if (row1>4)
row1=0;
if (row2>4)
row2=0; encrypted+=(Character.toString(alphabets[row1][col1])+Character.toString(alphabets[row2][col1]));
}
else
{ encrypted+=(Character.toString(alphabets[row1][col2])+ Character.toString(alphabets[row2][col1]));
}}
```

```
void decrypt(int row1, int col1, int row2, int col2)
{
if (row1==row2)
{
col1=col1-1; col2=col2-1; if (col1<0)
col1=4; if (col2<0) col2=4;
decrypted+=(Character.toString(alphabets[row1][col1])+Character.toString(alphabets[row1][col2]));
}
else if(col1==col2)
{
row1=row1-1; row2=row2-1;
if (row1<0)
row1=4;
if (row2<0)
row2=4;
decrypted+=(Character.toString(alphabets[row1][col1])+ Character.toString(alphabets[row2][col1]));
}
else
{
decrypted+=(Character.toString(alphabets[row1][col2])+
Character.toString(alphabets[row2][col1]));
}}
String getEncrypted( )
{
return encrypted;
}
String getDecrypted( )
{
return decrypted;
}}
```

OUTPUT:

```
F:\bin>javac Playfair1.java

F:\bin>java Playfair1
Enter keyword: INFOSEC
Enter message to encrypt: cryptography
I  N  F  O  S
E  C  A  B  D
G  H  K  L  M
P  Q  R  T  U
V  W  X  Y  Z
Encrypting....

The encrypted text is: AQVTYBKPERLW
========================================

Decrypting....

The encrypted text is: CRYPTOGRAPHY
```

Result:

Thus, the program to implement symmetric key encryption using playfair cipher substitution method was executed successfully and the output was verified.

Ex.no.: 1(c) **HILL CIPHER**

Date:

Aim:

To implement symmetric key encryption using Hill cipher substitution method.

Algorithm:

1. Obtain a plaintext message to encode in standard English with no spaces.
2. Split the plaintext into group of length three. To fill this, add X at the end.
3. Convert each group of letters with length three into plaintext vectors.
4. Replace each letter by the number corresponding to its position in the alphabet i.e.

A=1, B=2, C=3…Z=0.

5. Create the keyword in a 3*3 matrix.
6. Multiply the two matrices to obtain the cipher text of length three.
7. For decryption, convert each entry in the ciphertext vector into its plaintext vector by multiplying the cipher text vector and inverse of a matrix.
8. Thus plain text is obtained fromcorresponding plaintext vector by corresponding position in the alphabet.

Program:

```
import java.util.Scanner;
import javax.swing.JOptionPane;
public class hillcipher
{
//the 3x3 key matrix for 3 characters at once
public static int[][] keymat = new int[][]
{
{ 1, 2, 1 },
{ 2, 3, 2 },
{ 2, 2, 1 },
};
public static int[][] invkeymat = new int[][]
{
{ -1, 0, 1 },
{ 2, -1, 0 },
{ -2, 2, -1},
};
public static String key = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

public static void main(String[] args)
{
String text,outtext ="",outtext1="";
```

```
int ch,  n;
Scanner sc=new Scanner(System.in);
System.out.println("Enter the Plain text for Encryption: ");
//String text=new String();
text=sc.next();

text = text.toUpperCase();
text = text.replaceAll("\\s",""); //removing spaces

n = text.length() % 3;
if(n!=0)
{
for(int i = 1; i<= (3-n);i++)
{
text+= 'X';
}
}
System.out.println("Padded Text:" + text);
char[] ptextchars = text.toCharArray();
for(int i=0;i< text.length(); i+=3)
{
outtext += encrypt(ptextchars[i],ptextchars[i+1],ptextchars[i+2]);
}
System.out.println("Encypted Message: " + outtext);

char[] ptextchars1 = outtext.toCharArray();
for(int i=0;i< outtext.length(); i+=3)
{
outtext1 += decrypt(ptextchars1[i],ptextchars1[i+1],ptextchars1[i+2]);
}
System.out.println("Decrypted Message: " + outtext1);
}

private static String encrypt(char a, char b, char c)
{
String ret = "";
int x,y, z;
int posa = (int)a - 65;
 int posb = (int)b - 65;
int posc = (int)c - 65;
x = posa * keymat[0][0] + posb * keymat[1][0] + posc * keymat[2][0];
y = posa * keymat[0][1] + posb * keymat[1][1] + posc * keymat[2][1];
z = posa * keymat[0][2] + posb * keymat[1][2] + posc * keymat[2][2];
a = key.charAt(x%26);
b = key.charAt(y%26);
c = key.charAt(z%26); ret = "" + a + b + c; return ret;
```

```
}

private static String decrypt(char a, char b, char c)
{
String ret = "";
int x,y,z;
int posa = (int)a - 65;
int posb = (int)b - 65;
int posc = (int)c - 65;
x = posa * invkeymat[0][0]+ posb * invkeymat[1][0] + posc * invkeymat[2][0];
y = posa * invkeymat[0][1]+ posb * invkeymat[1][1] + posc * invkeymat[2][1];
z = posa * invkeymat[0][2]+ posb * invkeymat[1][2] + posc *  invkeymat[2][2];
a = key.charAt((x%26<0)?(26+x%26):(x%26));
b = key.charAt((y%26<0)?(26+y%26):(y%26));
c = key.charAt((z%26<0)?(26+z%26):(z%26));
ret = "" + a + b + c;
return ret;
}
}
```

Output:

```
F:\bin>javac hillcipher.java

F:\bin>java hillcipher
Enter the Plain text for Encryption:
mothertheresa
Padded Text:MOTHERTHERESAXX
Encypted Message: AAHXIGPPLJEROLR
Decrypted Message: MOTHERTHERESAXX

F:\bin>java hillcipher
Enter the Plain text for Encryption:
hilcipher
Padded Text:HILCIPHER
Encypted Message: TIIWGHXIG
Decrypted Message: HILCIPHER
```

Result:

Thus, the program to implement symmetric key encryption using Hill cipher substitution method was executed successfully and the output was verified.

Ex.no.: 1(d) **VIGENERE CIPHER**

Date:

Aim:

To implement symmetric key encryption using Ceaser cipher substitution method.

Algorithm:

1. The Vigenere cipher is a method of encrypting alphabetic text by using a series of different Caesar ciphers based on the letters of a keyword.
2. It is a simple form of polyalphabetic substitution.
3. To encrypt, a table of alphabets can be used, termed a Vigenere square, or Vigenere table.
4. It consists of the alphabet written out 26 times in different rows, each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible Caesar ciphers.
5. At different points in the encryption process, the cipher uses a different alphabet from one of the rows used.
6. The alphabet at each point depends on a repeating keyword.

Program:

```
public class vigenerecipher1
{
public static String encrypt(String text,final String key)
{
String res="";
for(int i=0,j=0; i< text.length(); i++)
{
char c=text.charAt(i);
if(c<'A'||c>'z')
continue;
res+=(char)((c+key.charAt(j)-2*'A')%26+'A');
j=++j%key.length();
}
return res;
}
public static String decrypt(String text,final String key)
{
String res="";
for(int i=0,j=0;i<text.length();i++)
{
```

```
char c=text.charAt(i);
if(c<'A'||c>'z')
continue;
res+=(char)((c-key.charAt(j)+26)%26+'A');
j=++j%key.length();
}
return res;
}
public static void main(String[] args)
{
System.out.println("Enter the key: ");
String key = System.console().readLine();
key = key.toUpperCase();
System.out.println("Enter the message for encrytption: ");
String message = System.console().readLine();
message = message.toUpperCase();
String encryptedMsg=encrypt(message,key);
System.out.println("String :"+message);
System.out.println("Encrypted message:Cipher Text=" +encryptedMsg);
System.out.println("Decrypted message:Plain Text="+decrypt(encryptedMsg,key));
}
}
```

Output:

```
F:\bin>javac vigenerecipher1.java
F:\bin>java  vigenerecipher1
Enter the key:
SECURITY
Enter the message for encrytption: CRYPTOGRAPHY
String :CRYPTOGRAPHY
Encrypted message:UVAJKWZPSTJS
Decrypted message:CRYPTOGRAPHY
```

Result:

Thus, the program to implement symmetric key encryption using vigenere cipher substitution method was executed successfully and the output was verified.

Ex.no.: 1(e) **RAIL FENCE CIPHER**

Date:

Aim:

To implement symmetric key encryption using rail fence cipher transposition method.

Algorithm:

1. In the rail fence cipher, the plaintext is written downwards and diagonally on successive "rails" of an imaginary fence, then moving up when we reach the bottom rail.
2. When we reach the top rail, the message is written downwards again until the whole plaintext is written out.
3. The message is then read off in rows.

Program:

```
class railfenceCipherHelper
{
int depth;
String encode(String msg, int depth) throws Exception
{
int r = depth;
int l = msg.length();
int c = l / depth;
int k = 0;
char mat[][] = new char[r][c]; String enc = "";
for (int i = 0; i< c; i++)
{
for (int j = 0; j < r; j++)
{
if (k != l)
{ mat[j][i] = msg.charAt(k++); }
else
{ mat[j][i] = 'X'; }
} }
for (int i = 0; i < r; i++)
{
for (int j = 0; j < c; j++)
{
enc += mat[i][j];
} }
return enc;
}
```

```
String decode(String encmsg,  int depth) throws Exception
{
int r = depth;
int l = encmsg.length();
int c = l / depth;
int k = 0;
char mat[][] = new char[r][c];

String dec = "";
for (int i = 0; i < r; i++)
{
for (int j = 0; j < c; j++)
{
mat[i][j] = encmsg.charAt(k++);
}
}
for (int i = 0; i< c; i++)
{
for (int j = 0; j < r; j++)
{
dec += mat[j][i];
}
}
return dec;
}
}
class railfencecipher
{
public static void main(String[] args) throws java.lang.Exception
{
railfenceCipherHelper rf = new railfenceCipherHelper();
String msg, enc, dec;
System.out.println("Enter the Plain text: ");
msg = System.console().readLine();
int depth = 2;
enc = rf.encode(msg, depth);
dec = rf.decode(enc, depth);
System.out.println("Plain Text:"+msg);
System.out.println("Encrypted Message-Cipher Text:"+enc);
System.out.printf("Decrypted Message-:"+dec);
}
}
```

Output:

```
F:\bin>javac railfencecipher.java

F:\bin>java  railfencecipher

Enter the Plain text:

attack at dawn

Plain Text attack at dawn

Encrypted Message-Cipher Text: atc tdwtaka an

Decrypted Message-: attack at dawn
```

Result:

Thus, the program to implement symmetric key encryption using rail fence cipher transposition method was executed successfully and the output was verified

Ex.no.: 1(f) **ROW COLUMN TRANSPOSITION TECHNIQUE**

Date:

Aim:

To implement symmetric key encryption using row column transposition method.

Algorithm:

1. Consider the plain text hello world, and let us apply the simple columnar transposition technique as shown below

h	e	l	l
o	w	o	r
l	d		

2. The plain text characters are placed horizontally and the cipher text is created with vertical format as: holewdlolr.
3. Now, the receiver has to use the same table to decrypt the cipher text to plain text.

Program:

```
import java.util.*;
class TransCipher
{
public static void main(String args[])
{
Scanner sc = new Scanner(System.in);
System.out.println("Enter the plain text");
String pl = sc.nextLine();
sc.close();
String s = "";
int start = 0;
for (int i= 0; i< pl.length(); i++)
{
if (pl.charAt(i) == ' ')
{
s = s + pl.substring(start, i);
start = i + 1;
}
}
s = s + pl.substring(start);
System.out.print(s);
System.out.println();
```

```
 // end of space deletion
int k = s.length();
int l = 0;
int col = 4;
int row = s.length() / col;
char ch[][] = new char[row][col];
for (int i = 0; i < row; i++)
 {
for (int j = 0; j < col; j++)
{
if (l < k)
{
ch[i][j] = s.charAt(l);
l++;
}
else
{
ch[i][j] = '#';
}
}
}
char trans[][] = new char[col][row];
for (int i = 0; i < row; i++)
{
for (int j = 0; j < col; j++)
{
trans[j][i] = ch[i][j];
```

```
}
}
for (int i = 0; i < col; i++)
{
for (int j = 0; j < row; j++)
{
System.out.print(trans[i][j]);
}
 }
System.out.println();
}
}
```

Output:

```
F:\bin>javac TransCipher.java
F:\bin>java  TransCipher
Enter the plain text altrozcarshervin
 altrozcarshervin
aorrlzsvtchiraen
a        l        t        r
 o        z        c        a
 r        s        h        e
r        v        I        n
```

Result:

Thus, the program to implement symmetric key encryption using row column transposition method was executed successfully and the output was verified

Ex.no.: 1(g) **DATA ENCRYPTION STANDARD (DES)**

Date:

Aim:

To apply Data Encryption Standard (DES) Algorithm for a practical application like User Message Encryption.

Algorithm:

1. Create a DES Key.
2. Create a Cipher instance from Cipher class, specify the following information and separated by a slash (/).
   - Algorithm name
   - Mode (optional)
   - Padding scheme (optional)
3. Convert String into Byte[] array format.
4. Make Cipher in encrypt mode, and encrypt it with Cipher.doFinal() method.
5. Make Cipher in decrypt mode, and decrypt it with Cipher.doFinal() method.

Program:

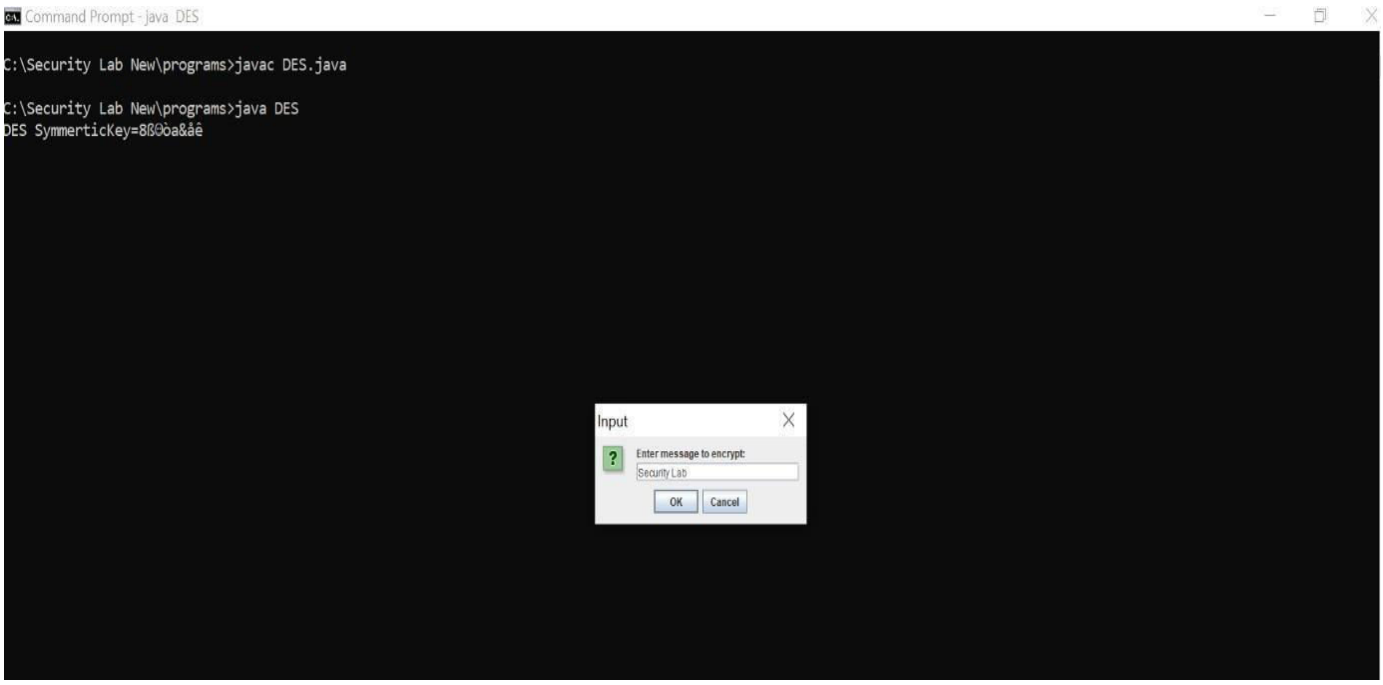
```
import javax.swing.*;
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Random;
class DES
{
byte[] skey=new byte[1000]; String skeystring;
static byte[] raw;
String inputmessage,encryptedata,decryptedmessage;
public DES()
{
try
{

generatesymmetrickey();
inputmessage=JOptionPane.showInputDialog(null,"Enter message to encrypt:");
byte[] ibyte =inputmessage.getBytes();
byte[] ebyte=encrypt(raw,  ibyte);
String encrypteddata=new String(ebyte); System.out.println("Encrypted message:"+encrypteddata);
```

```
JOptionPane.showMessageDialog(null,"EncryptedData"+"\n"+encrypteddata);
byte[] dbyte=decrypt(raw,ebyte);
String decryptedmessage=new String(dbyte);
System.out.println("Decrypted message:"+decryptedmessage);

JOptionPane.showMessageDialog(null,"Decrypted Data"+"\n"+decryptedmessage);

}

catch(Exception e)

{

System.out.println(e);

}

}

void generatesymmetrickey()

{

try

{

Random r = new Random();

int num=r.nextInt(10000);

String knum=String.valueOf(num);

byte[] knumb=knum.getBytes();

skey=getRawKey(knumb);

skeystring=new String(skey);

System.out.println("DES SymmerticKey="+skeystring);

}

catch(Exception e)

{

System.out.println(e);

}

}
```

```
private static byte[] getRawKey(byte[] seed) throws Exception
{
KeyGenerator kgen=KeyGenerator.getInstance("DES");
SecureRandom sr =SecureRandom.getInstance("SHA1PRNG");
sr.setSeed(seed);
kgen.init(56,sr);
SecretKey skey=kgen.generateKey();
raw=skey.getEncoded();
return raw;
}
private static byte[] encrypt(byte[] raw,byte[] clear) throws Exception
{
SecretKey seckey = new SecretKeySpec(raw, "DES");
Cipher cipher = Cipher.getInstance("DES");
cipher.init(Cipher.ENCRYPT_MODE,seckey);
byte[] encrypted=cipher.doFinal(clear);
return encrypted;
}
private static byte[] decrypt(byte[] raw,byte[] encrypted) throws Exception
{
SecretKey seckey = new SecretKeySpec(raw, "DES");
Cipher cipher = Cipher.getInstance("DES");
cipher.init(Cipher.DECRYPT_MODE,seckey);
byte[] decrypted = cipher.doFinal(encrypted);
return decrypted;
}
```

```
public static void main(String args[])
{
DES des=new DES();
}
}
```

Output:



Result:

Thus the java program for applying Data Encryption Standard (DES) Algorithm for a practical application of User Message Encryption is written and executed successfully.

Ex.no.: 1(h) **AES ALGORITHM**

Date:

Aim:

To apply Advanced Encryption Standard (AES) Algorithm for a practical application like URL Encryption.

Algorithm:

1. AES is based on a design principle known as a substitution–permutation.
2. AES does not use a Feistel network like DES, it uses variant of Rijndael.
3. It has a fixed block size of128 bits, and a key size of 128, 192, or 256 bits.
4. AES operates on a 4 × 4 column-major order array of bytes, termed the state

Program:

```
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class AES
{
private static SecretKeySpec secretKey;
private static byte[] key;
public static void setKey(String myKey)
{
MessageDigest sha = null;
try
{
key = myKey.getBytes("UTF-8");
sha = MessageDigest.getInstance("SHA-1");
key = sha.digest(key);
```

```
key = Arrays.copyOf(key, 16);
secretKey = new SecretKeySpec(key, "AES");
}
catch (NoSuchAlgorithmException e)
{
e.printStackTrace();
} catch (UnsupportedEncodingException e)
{
e.printStackTrace();
}
}
public static String encrypt(String strToEncrypt, String secret)
{
try
{
setKey(secret);
Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
cipher.init(Cipher.ENCRYPT_MODE, secretKey);
return Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));
}
catch (Exception e)
{
System.out.println("Error while encrypting: " + e.toString());
}
return null;
}

public static String decrypt(String strToDecrypt, String secret)
{
try {
setKey(secret);
Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
cipher.init(Cipher.DECRYPT_MODE, secretKey);
return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
}
```

```
 catch (Exception e)
{
System.out.println("Error while decrypting: " + e.toString());
}
return null;
}
public static void main(String[] args) {

System.out.println("Enter the secret key: ");
String secretKey = System.console().readLine();

System.out.println("Enter the original URL: ");
 String originalString = System.console().readLine();

String encryptedString = AES.encrypt(originalString, secretKey);
String decryptedString = AES.decrypt(encryptedString, secretKey);

System.out.println("URL Encryption Using AES Algorithm\n ----------");
System.out.println("Original URL : " + originalString);
System.out.println("Encrypted URL : " + encryptedString);
System.out.println("Decrypted URL : " + decryptedString);
}
}
```

Output:

```
C:\Security Lab New\programs>java AES
Enter the secret key:
annaUniversity
Enter the original URL:
www.annauniv.edu
URL Encryption Using AES Algorithm
Original URL : www.annauniv.edu
Encrypted URL : vibpFJW6Cvs5Y+L7t4N6YWWe07+JzS1d3CU2h3mEvEg=
Decrypted URL : www.annauniv.edu
```

Result:

Thus the java program for applying Advanced Encryption Standard (AES) Algorithm for a practical application of URL encryption is written and executed successfully.

Ex.no.: 2(a) **RSA ALGORITHM**

Date:

Aim:

To implement a RSA algorithm using HTML and Javascript.

Algorithm:

1. Choose two prime number p and q.
2. Compute the value of n and t.
3. Find the value of public key e.
4. Compute the value of private key d.
5. Do the encryption and decryption

a. Encryption is given as,

$c = t^e \mod n$

b. Decryption is given as,

$t = c^d \mod n$

Program:

```
import java.math.*;
import java.util.*;

class RSA1 {
	public static void main(String args[])
	{
		int p, q, n, z, d = 0, e, i;

		// The number to be encrypted and decrypted
			Scanner sc= new Scanner(System.in);
			System.out.println("Enter the message to be encrypted:");
		int msg = sc.nextInt();
			System.out.println("Enter the value of p first prime no.:");
		p = sc.nextInt();
			System.out.println("Enter the value of q second prime no.:");
		q = sc.nextInt();
```

```
double c;
BigInteger msgback;

n = p * q;
z = (p - 1) * (q - 1);
System.out.println("the value of z = " + z);

for (e = 2; e < z; e++) {

        // e is for public key exponent
        if (gcd(e, z) == 1) {
                break;
        }
}
System.out.println("the value of e = " + e);
for (i = 0; i <= 9; i++) {
        int x = 1 + (i * z);

        // d is for private key exponent
        if (x % e == 0) {
                d = x / e;
                break;
        }
}
System.out.println("the value of d = " + d);
c = (Math.pow(msg, e)) % n;
System.out.println("Encrypted message is : " + c);

// converting int value of n to BigInteger
BigInteger N = BigInteger.valueOf(n);

// converting float value of c to BigInteger
BigInteger C = BigDecimal.valueOf(c).toBigInteger();
msgback = (C.pow(d)).mod(N);
System.out.println("Decrypted message is : "+ msgback);
```

```
    }

    static int gcd(int e, int z)
    {
        if (e == 0)
            return z;
        else
            return gcd(z % e, e);
    }
}
```

Output:

```
G:\MCE_IT\Network security\LAB>javac RSA1.java

G:\MCE_IT\Network security\LAB>java RSA1
Enter the message to be encrypted:
88
Enter the value of p first prime no.:
17
Enter the value of q second prime no.:
11
the value of z = 160
the value of e = 3
the value of d = 107
Encrypted message is : 44.0
Decrypted message is : 88

G:\MCE_IT\Network security\LAB>
```

```
G:\MCE_IT\Network security\LAB>java RSA1
Enter the message to be encrypted:
9
Enter the value of p first prime no.:
5
Enter the value of q second prime no.:
11
the value of z = 40
the value of e = 3
the value of d = 27
Encrypted message is : 14.0
Decrypted message is : 9
```

Result:

Thus the RSA algorithm was executed successfully and the output was verified.

Ex.no.: 2(b) **DIFFIE-HELLMAN KEY EXCHANGE ALGORITHM**

Date:

Aim:

To implement Diffie-Hellman Key Exchange algorithm.

Algorithm:

1. Sender and receiver publicly agree to use a modulus p and base g which is a primitive root modulo p.
2. Sender chooses a secret integer x then sends Bob R1 = g^x mod p
3. Receiver chooses a secret integer y, then sends Alice R2 = g^y mod p
4. Sender computes k1 = B^x mod p
5. Receiver computes k2 = A^y mod p
6. Sender and Receiver now share a secret key.

Program:

```
import java.io.*;
import java.math.BigInteger;
class dh
{
public static void main(String[]args)throws IOException
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter prime number:");
BigInteger p=new BigInteger(br.readLine());

System.out.print("Enter primitive root of "+p+":");
BigInteger g=new BigInteger(br.readLine());

System.out.println("Enter value for x less than "+p+":");
BigInteger x=new BigInteger(br.readLine());
BigInteger R1=g.modPow(x,p);
System.out.println("R1="+R1);

System.out.print("Enter value for y less than "+p+":");
BigInteger y=new BigInteger(br.readLine());
```

```
BigInteger R2=g.modPow(y,p);
System.out.println("R2="+R2);

BigInteger k1=R2.modPow(x,p);
System.out.println("Key calculated at Sender's side:"+k1);
BigInteger k2=R1.modPow(y,p);
System.out.println("Key calculated at Receiver's side:"+k2);
System.out.println("Diffie-Hellman secret key was calculated.");
}
}
```

Output:

```
C:\Security Lab New\programs>javac dh.java

C:\Security Lab New\programs>java dh
Enter prime number:
11
Enter primitive root of 11:7
Enter value for x less than 11:
3
R1=2
Enter value for y less than 11:6
R2=4
Key calculated at Sender's side:9
Key calculated at Receiver's side:9
Diffie-Hellman secret key was calculated.
```

Result:

Thus the Diffie-Hellman key exchange algorithm was executed successfully and the output was verified.

Ex.no.: 3 **DIGITAL SIGNATURE SCHEME**

Date:

Aim:

To implement the signature scheme - Digital Signature Standard.

Algorithm:

1. Declare the class and required variables.
2. Create the object for the class in the main program.
3. Access the member functions using the objects.
4. Implement the SIGNATURE SCHEME - Digital Signature Standard.
5. It uses a hash function.
6. The hash code is provided as input to a signature function along with a random number K generated for the particular signature.
7. The signature function also depends on the sender's private key.
8. The signature consists of two components.
9. The hash code of the incoming message is generated.
10. The hash code and signature are given as input to a verification function.

Program:

```
import java.util.*;
import java.math.BigInteger;
class dsaAlg {
final static BigInteger one = new BigInteger("1");
final static BigInteger zero = new BigInteger("0");
public static BigInteger getNextPrime(String ans)
{
BigInteger test = new BigInteger(ans);
while (!test.isProbablePrime(99))
{
test = test.add(one);
}
return test;
```

```
}
public static BigInteger findQ(BigInteger n)
{
BigInteger start = new BigInteger("2");
while (!n.isProbablePrime(99))
{
while (!((n.mod(start)).equals(zero)))
{
start = start.add(one);
}
n = n.divide(start);
}
return n;
}

public static BigInteger getGen(BigInteger p, BigInteger q, Random r)
{
BigInteger h = new BigInteger(p.bitLength(), r);
h = h.mod(p);
return h.modPow((p.subtract(one)).divide(q), p);
}
public static void main (String[] args) throws java.lang.Exception
{
Random randObj = new Random();
BigInteger p = getNextPrime("10600"); /* approximate prime */
BigInteger q = findQ(p.subtract(one));
BigInteger g = getGen(p,q,randObj);
System.out.println(" \n simulation of Digital Signature Algorithm \n");
 System.out.println(" \n global public key components are:\n");
System.out.println("\np is: " + p);
System.out.println("\nq is: " + q);
System.out.println("\ng is: " + g);
BigInteger x = new BigInteger(q.bitLength(), randObj);
x = x.mod(q);
BigInteger y = g.modPow(x,p);
```

```
BigInteger k = new BigInteger(q.bitLength(), randObj);
k = k.mod(q);
BigInteger r = (g.modPow(k,p)).mod(q);
BigInteger hashVal = new BigInteger(p.bitLength(),randObj);
BigInteger kInv = k.modInverse(q);
BigInteger s = kInv.multiply(hashVal.add(x.multiply(r)));
s = s.mod(q);
System.out.println("\nsecret information are:\n");
System.out.println("x (private) is:" + x);
System.out.println("k (secret) is: " + k);
System.out.println("y (public) is: " + y);
System.out.println("h (rndhash) is: " + hashVal);
System.out.println("\n generating digital signature:\n");
System.out.println("r is : " + r);
System.out.println("s is : " + s);
BigInteger w = s.modInverse(q);
BigInteger u1 = (hashVal.multiply(w)).mod(q);
BigInteger u2 = (r.multiply(w)).mod(q);
BigInteger v = (g.modPow(u1,p)).multiply(y.modPow(u2,p));
v = (v.mod(p)).mod(q);
System.out.println("\nverifying digital signature (checkpoints)\n:");
System.out.println("w is : " + w);
System.out.println("u1 is : " + u1);
System.out.println("u2 is : " + u2);
 System.out.println("v is : " + v);
if (v.equals(r))
{
System.out.println("\nsuccess: digital signature is verified!\n " + r);
}
else
{
System.out.println("\n error: incorrect digital signature\n ");
}
}
}
```

Output:

```
C:\Security Lab New\programs>javac dsaAlg.java
C:\Security Lab New\programs>java dsaAlg
simulation of Digital Signature Algorithm
global public key components are:
p is: 10601
q is: 53
g is: 6089
secret information are:
x (private) is:6
k (secret) is: 3
y (public) is: 1356
h (rndhash) is: 12619
generating digital signature:
r is : 2
s is :41
verifying digital signature (checkpoints):
w  is  : 22
u1  is  : 4
u2 is : 44
 v is : 2
success: digital signature is verified!
2
```

Result:

Thus the Digital Signature Standard Signature Scheme has been implemented and executed successfully.

Ex. No: 4 **Installation of Wireshark and observe data transferred in client-server**

Date: **communication using UDP/TCP and identify the UDP/TCP datagram.**

Aim

To install wireshark and observe data transferred in client server communication using UDP/TCP and identify the packets.

Procedure:

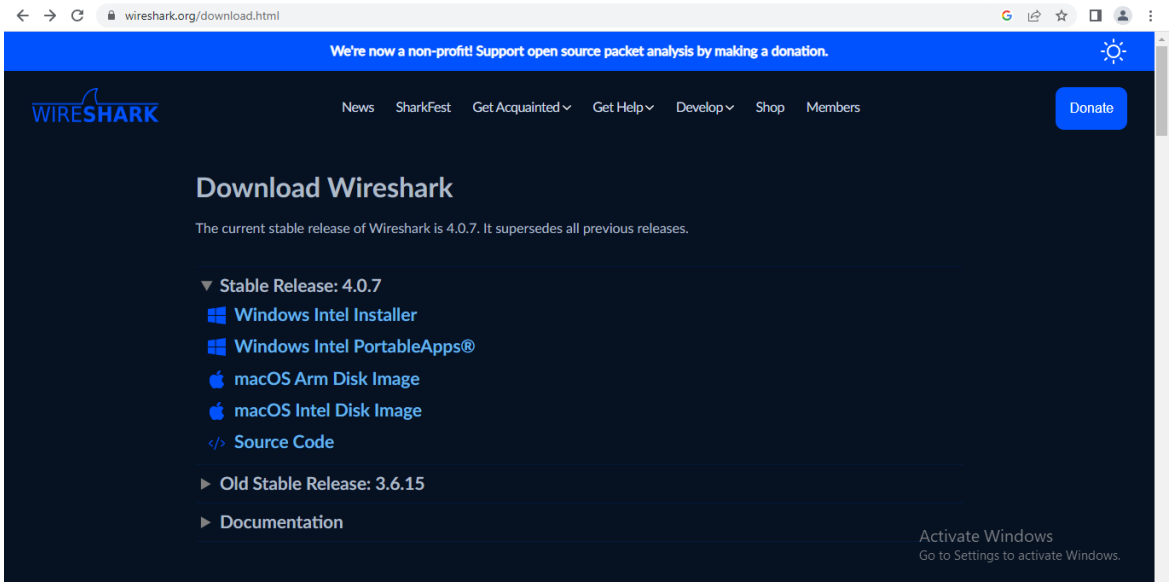
Wireshark:

- Wireshark is a network packet analyzer. A network packet analyzer presents captured packet data in as much detail as possible.
- A network packet analyzer is a measuring device for examining what's happening inside a network cable.
- Wireshark is an open-source packet analyzer, which is used for education, analysis, software development, communication protocol development, and network troubleshooting.
- It is used to track the packets so that each one is filtered to meet our specific needs. It is commonly called as a sniffer, network protocol analyzer, and network analyzer. It is also used by network security engineers to examine security problems.

Steps to install wireshark:

Follow the below steps to install Wireshark on Windows:

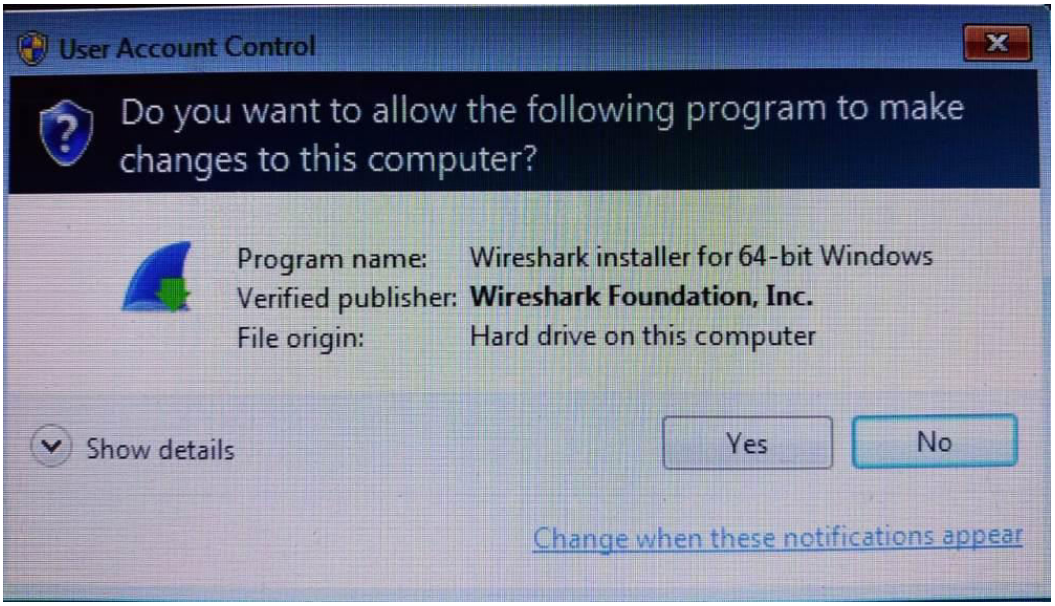
Step 1: Go to the official web page ***https://www.wireshark.org/download.html*** using any browser.



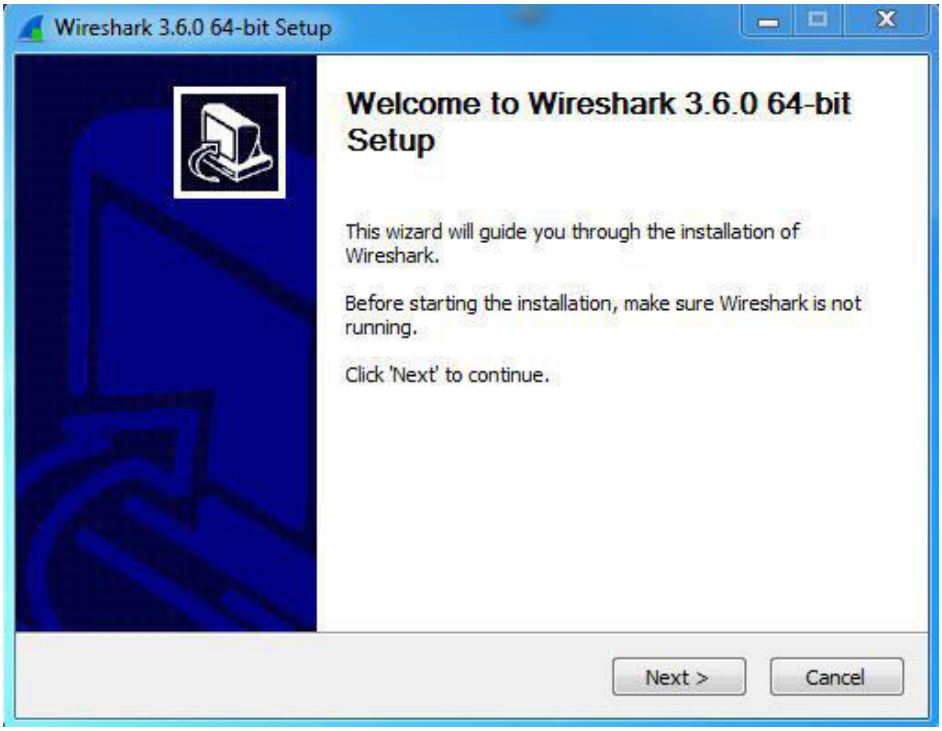
Step 2: Click on Windows Intel installer option, an executable file will be downloaded.

Step 3: Run the executable file in your system.

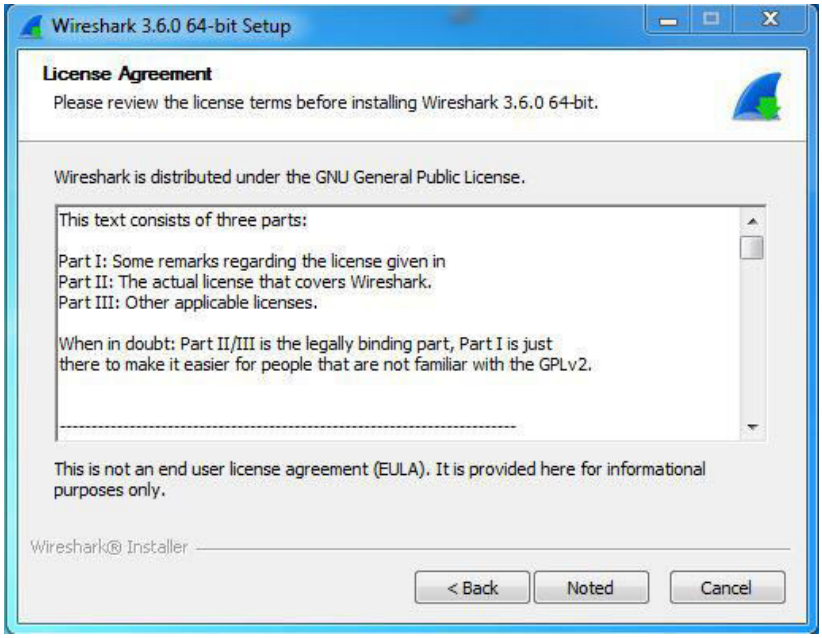
Step 4: It will prompt confirmation to make changes to your system. Click on Yes.



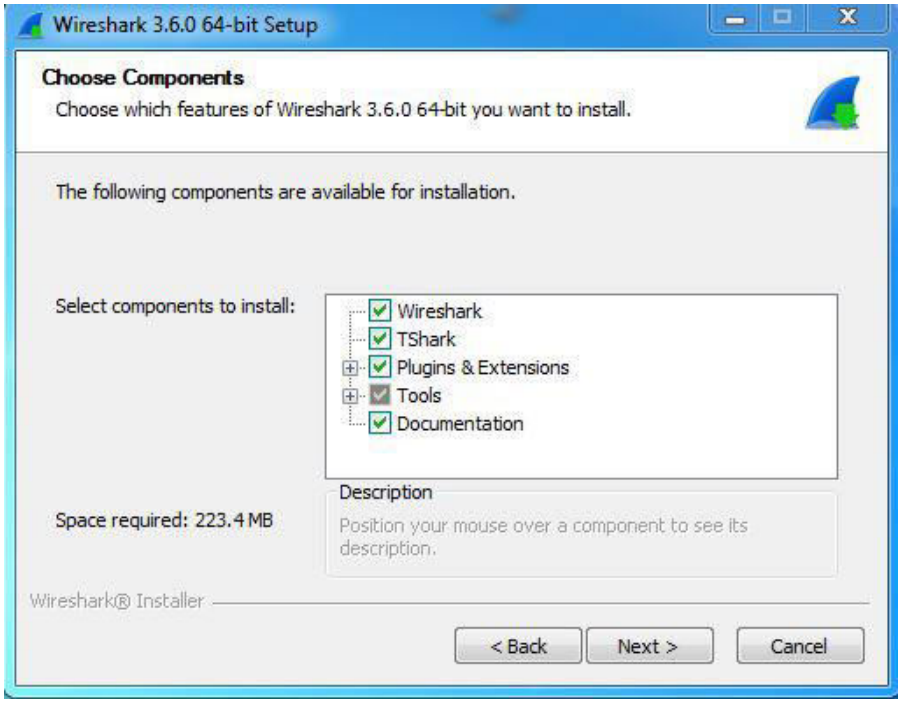
Step 5: Setup screen will appear, click on Next.



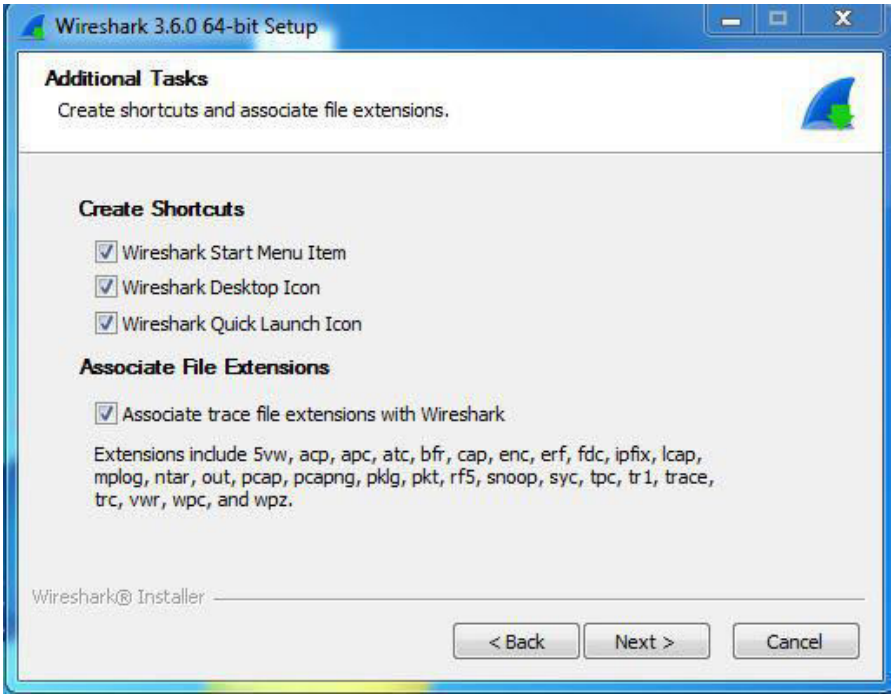
Step 6: The next screen will be of License Agreement, click on Noted.



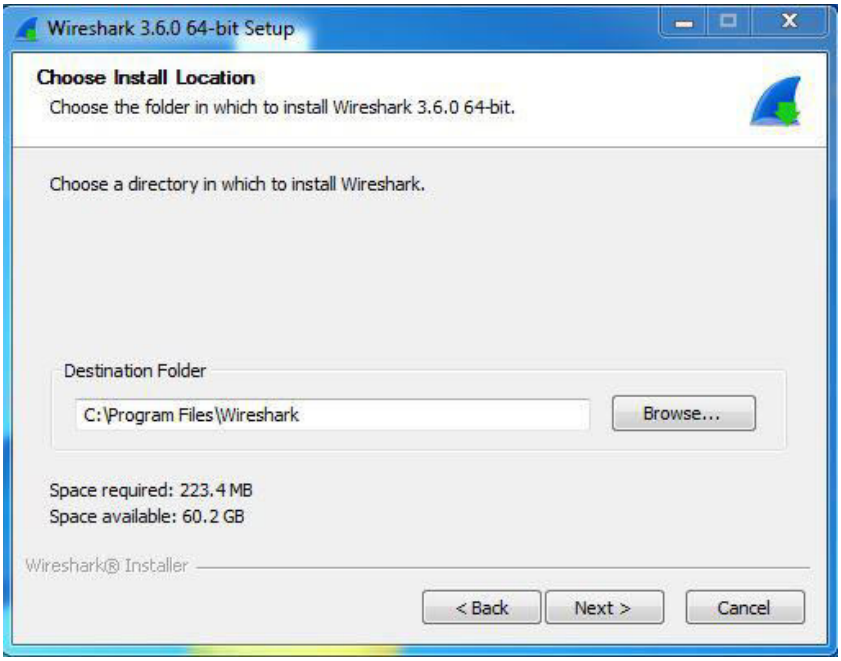
Step 7: This screen is for choosing components, all components are already marked so don't change anything just click on the Next button.



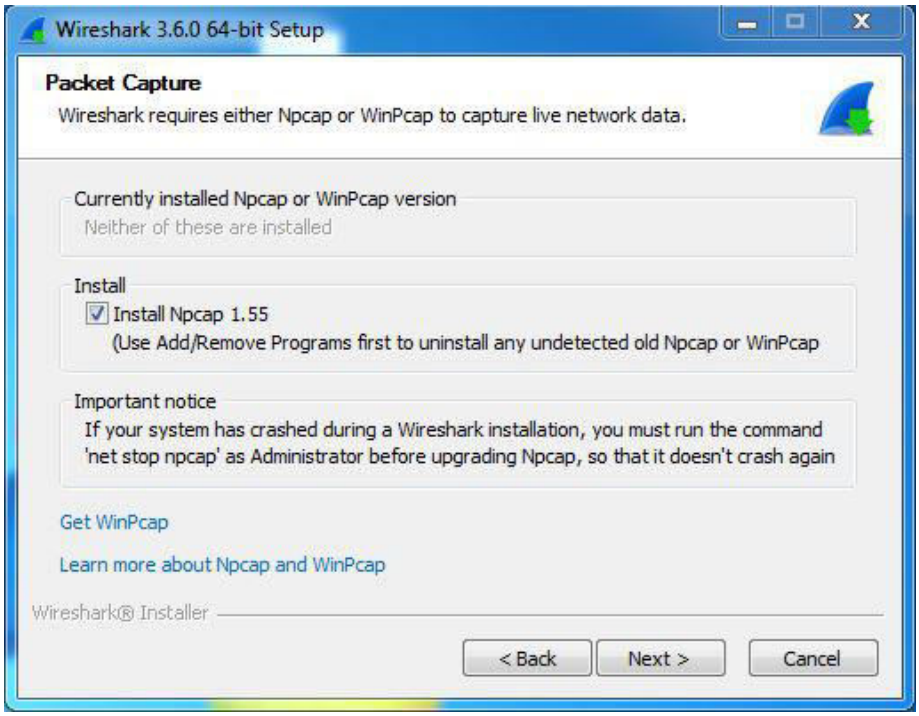
Step 8: This screen is of choosing shortcuts like start menu or desktop icon along with file extensions which can be intercepted by Wireshark, tick on needed boxes and click on Next button.



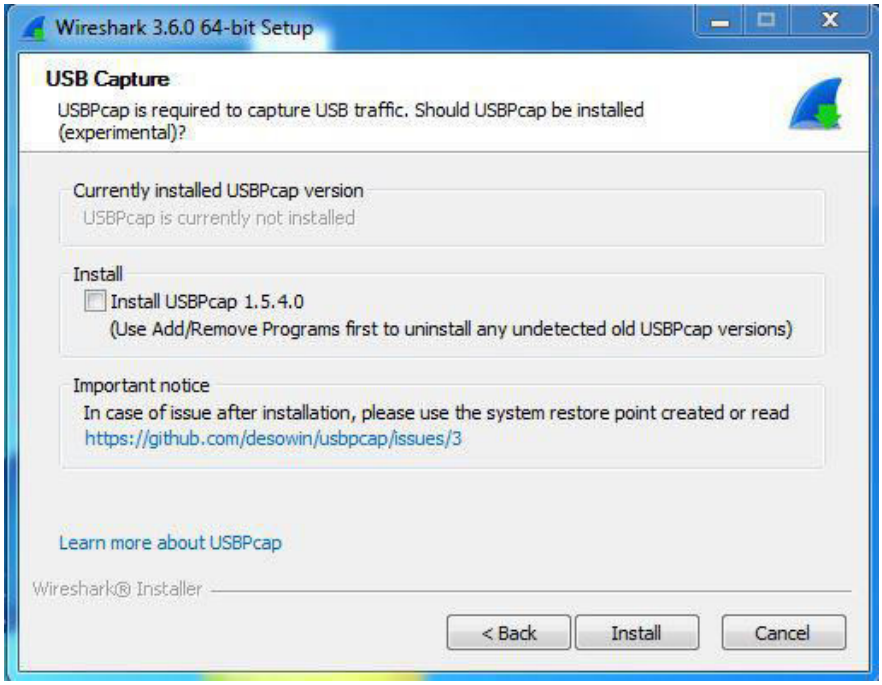
Step 9: You need to choose the destination folder by browsing to the location where you need to install wireshark. By default, it will install under C:\Program Files\Wireshark folder as shown below. Once chosen, Click on Next to proceed.



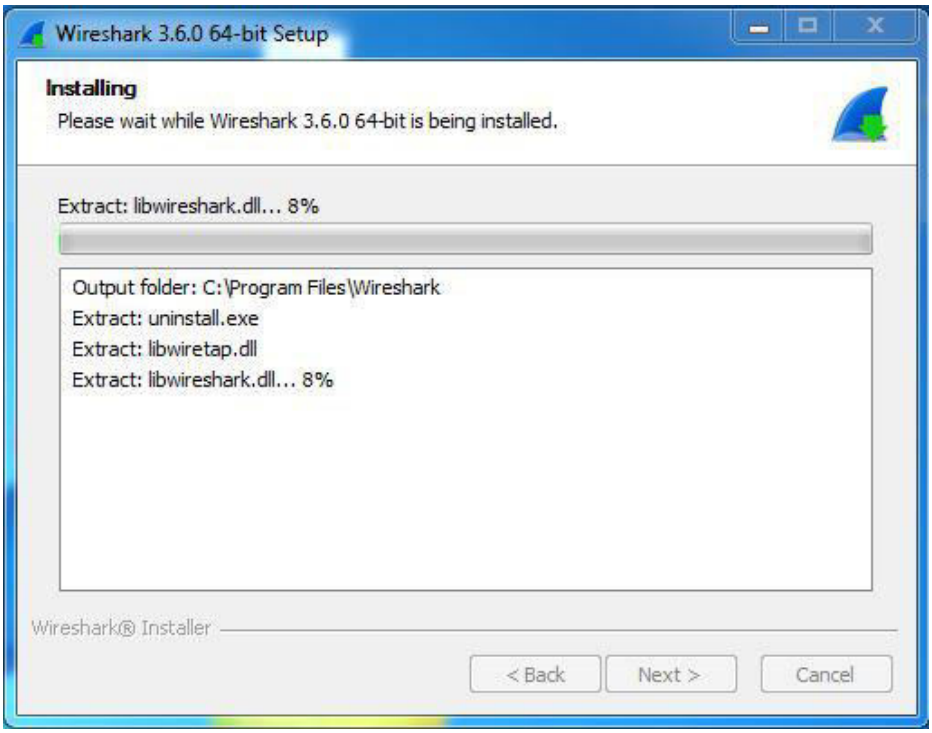
Step 10: Next screen has an option to install Npcap which is used with Wireshark to capture packets *pcap* means packet capture so the install option is already checked don't change anything and click the next button.



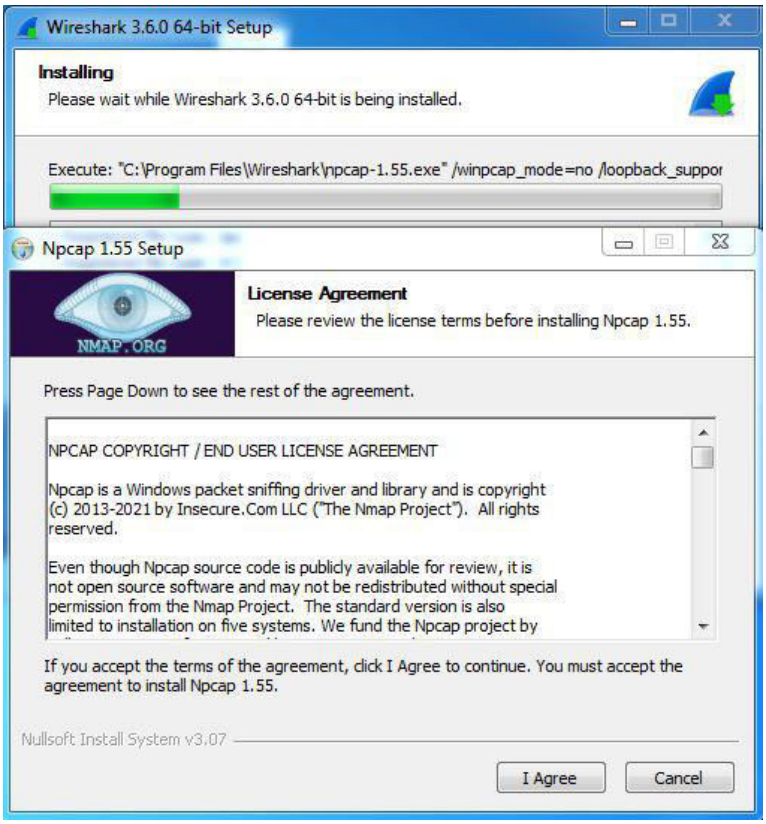
Step 11: Next screen is about USB network capturing so it is one's choice to use it or not, click on Install.



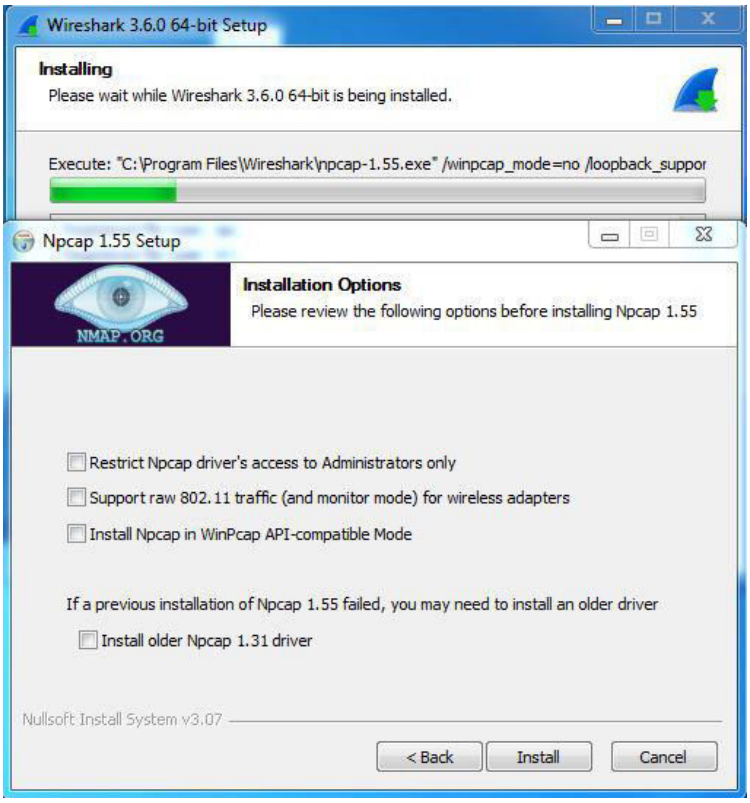
Step 12: After this installation process will start.



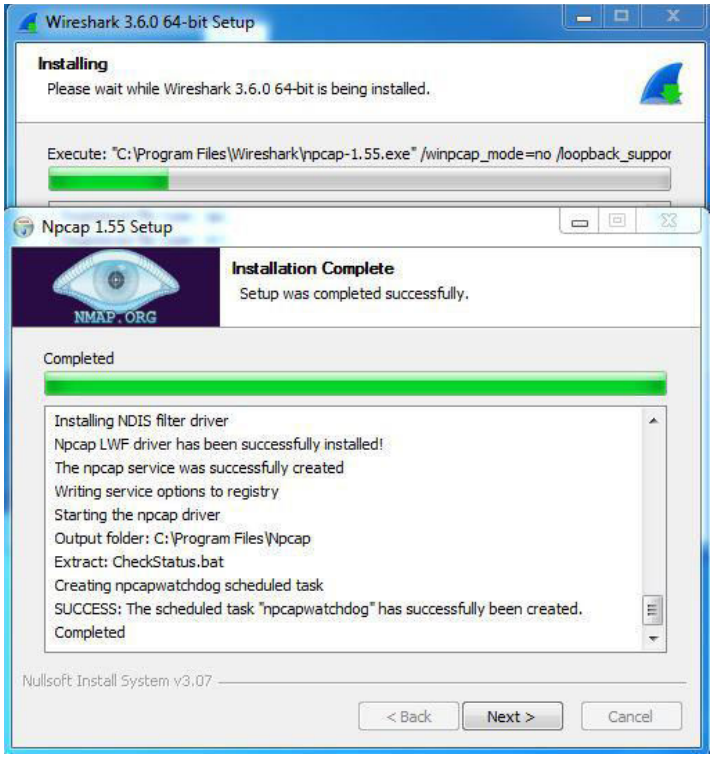
Step 13: This installation will prompt for Npcap installation as already checked so the license agreement of Npcap will appear to click on the ***I Agree*** button.



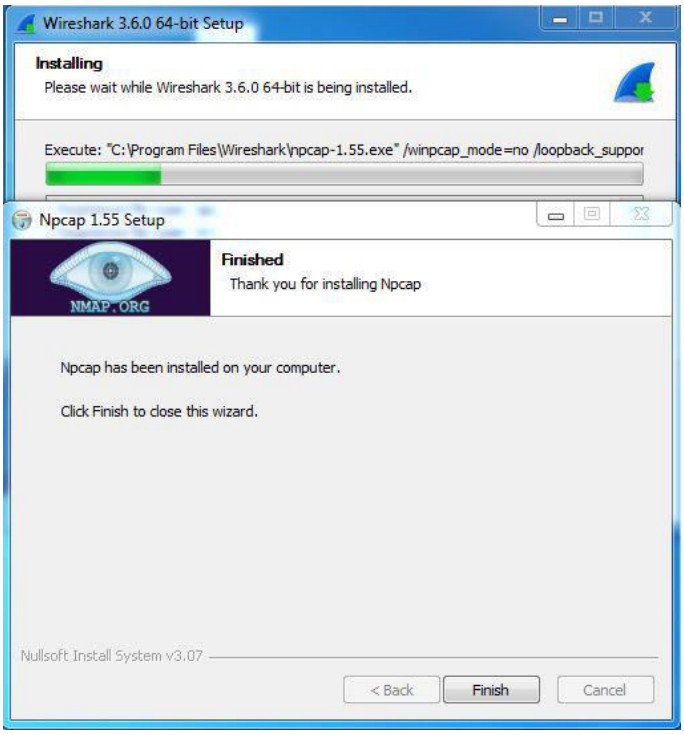
Step 14: Next screen is about different installing options of *npcap*, don't do anything click on Install.



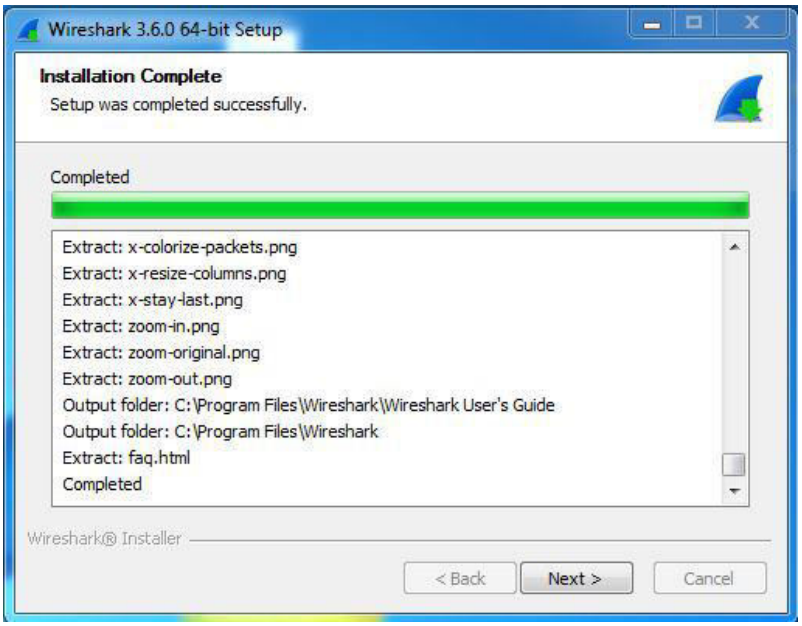
Step 15: After this installation process will complete click on the Next button.



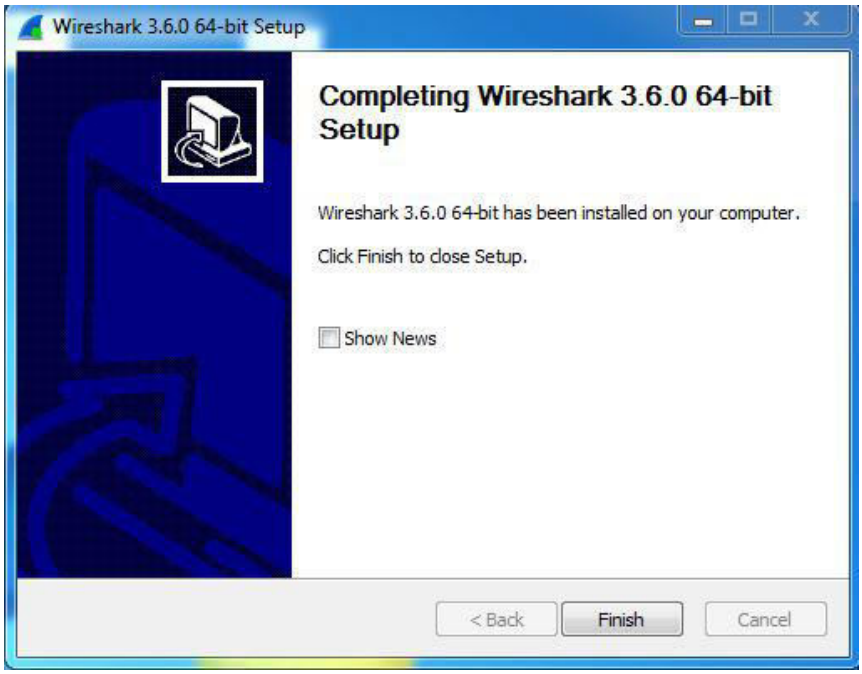
Step 16: Click on Finish after the installation process is complete.



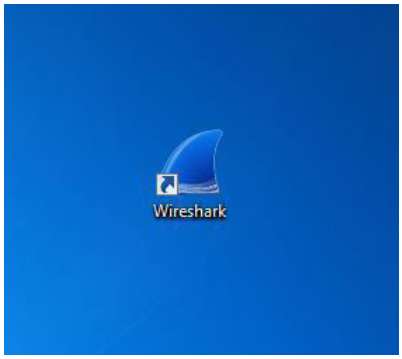
Step 17: After this installation process of Wireshark will complete click on the Next button.



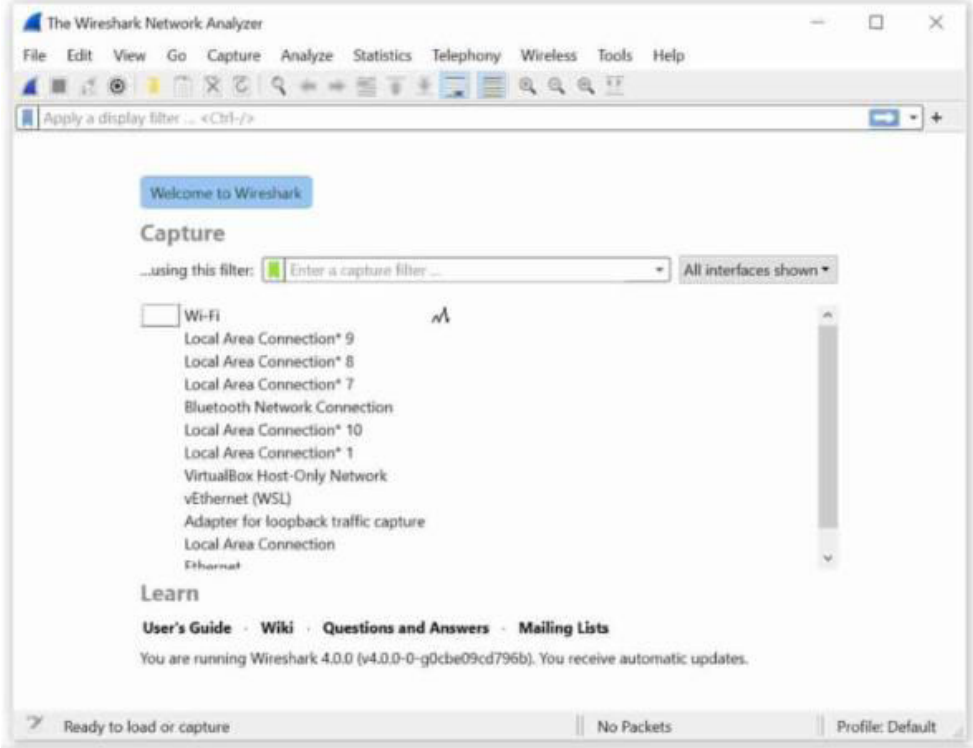
Step 18: Click on Finish after the installation process of Wireshark is complete.



Step 19: Wireshark is successfully installed on the system and an icon is created on the desktop.



Step 20: After successful installation, the first launch of wireshark should look like below. You need to select the Ethernet interface from where you need to capture the packets.

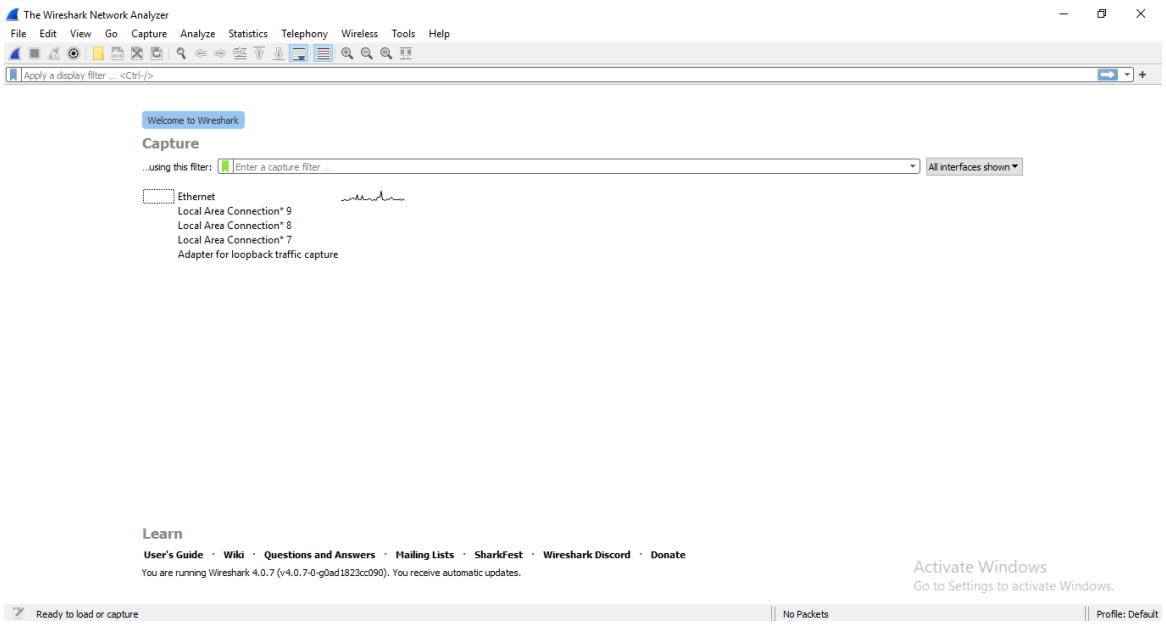


Analyzing the network using wireshark:

Wireshark is a packet sniffer and analysis tool. It captures network traffic from Ethernet, Bluetooth, wireless (IEEE.802.11), token ring, and frame relay connections, among others, and stores that data for offline analysis.

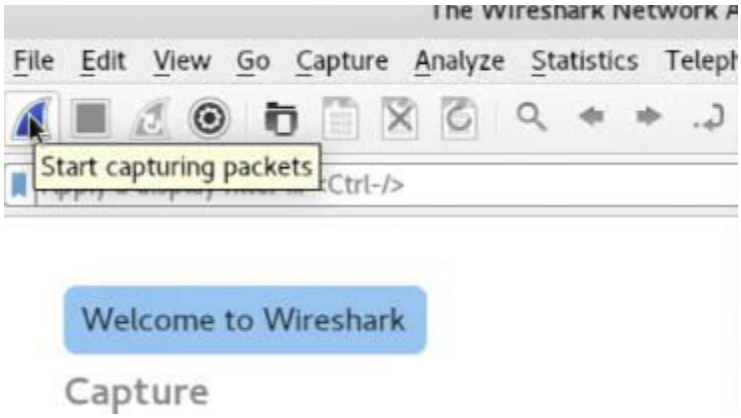
Capturing data packets on Wireshark

- When we open Wireshark, we can see a screen showing us a list of all the network connections that can be monitored.
- It also has a capture filter field to only capture the network traffic we want to see.

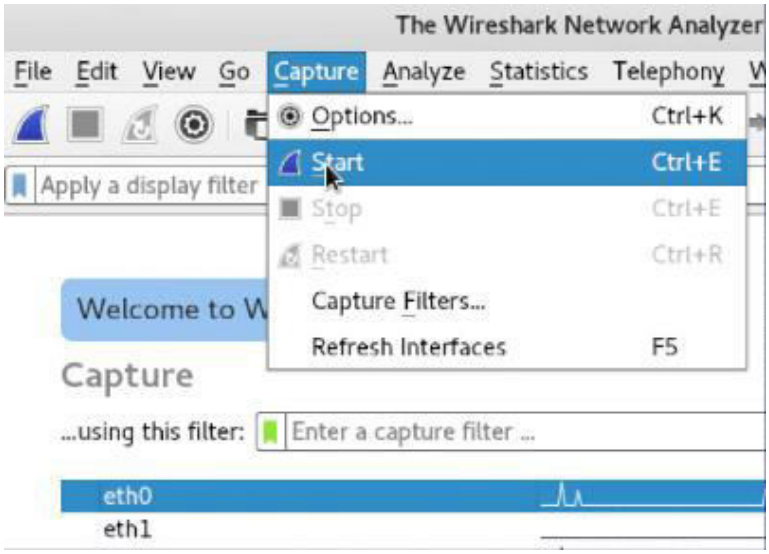


- We can select one of the network interfaces by clicking on it. For e.g. in the above window **'Ethernet'** is showing the traffic, it is active and can be selected.
- Once select the network interface, you can start the capture, and there are several ways to do that.

1. Click the first button (**blue fin shaped button**) on the toolbar, titled "**Start capturing packets**."

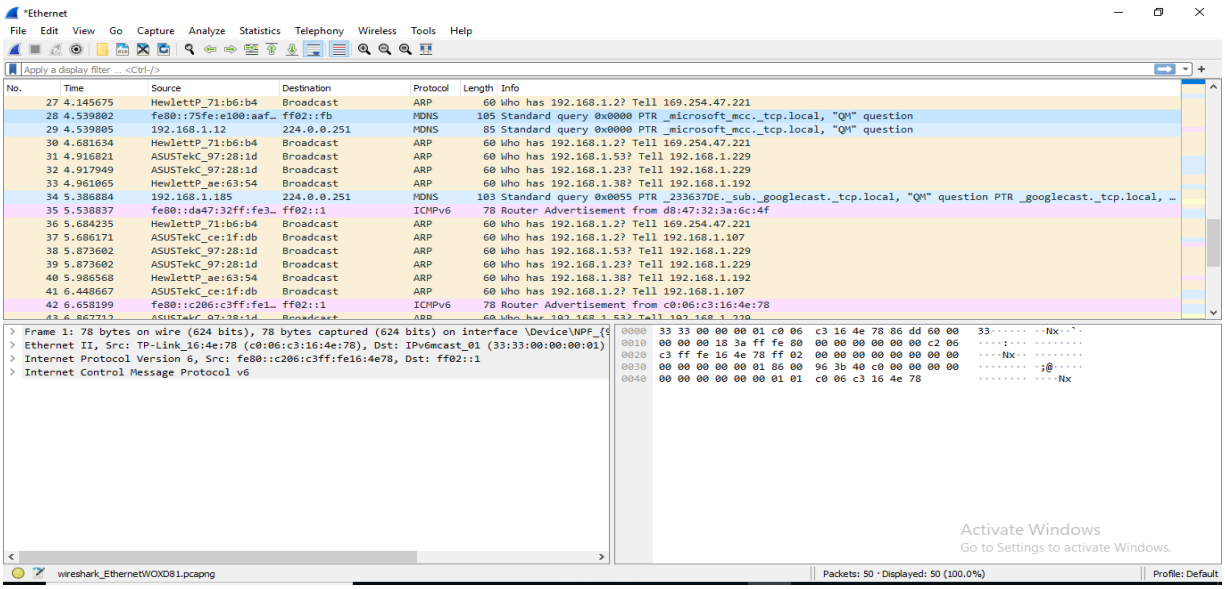


2. We can select the menu item **Capture -> Start.**

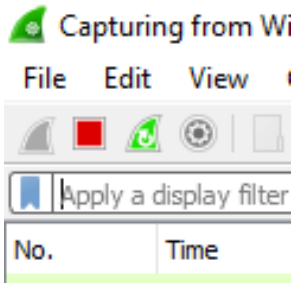


3. We can also use the shortcut key **Ctrl + E.**

- During the capture, Wireshark will show you the packets captured in real-time.



- It will continue listening to all the data packets, and we will get much data.
- If we want to see a particular data, then we can click on the **red button**.

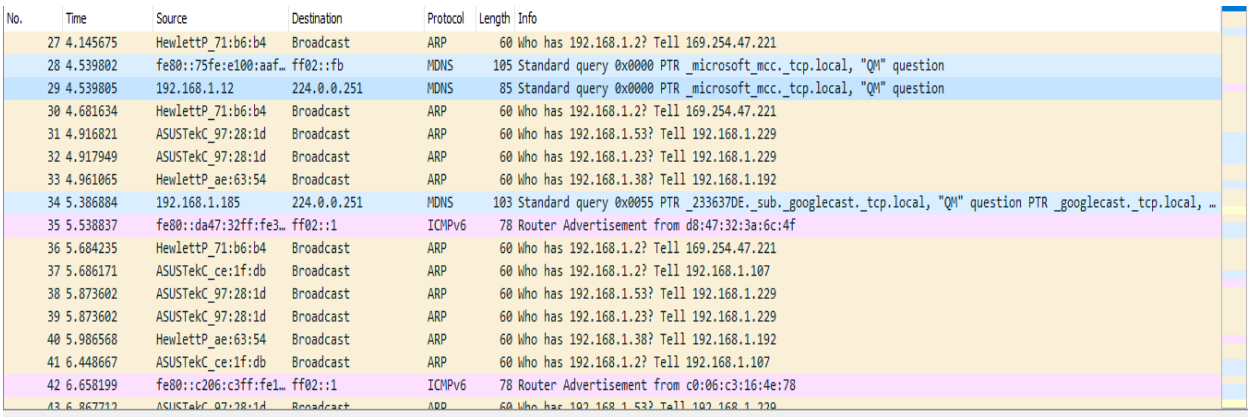


- The traffic will become stationary, and we can note the parameters like time, source, destination, the protocol being used, length, and the Info.

The screen/interface of the Wireshark is divided into **five parts**:

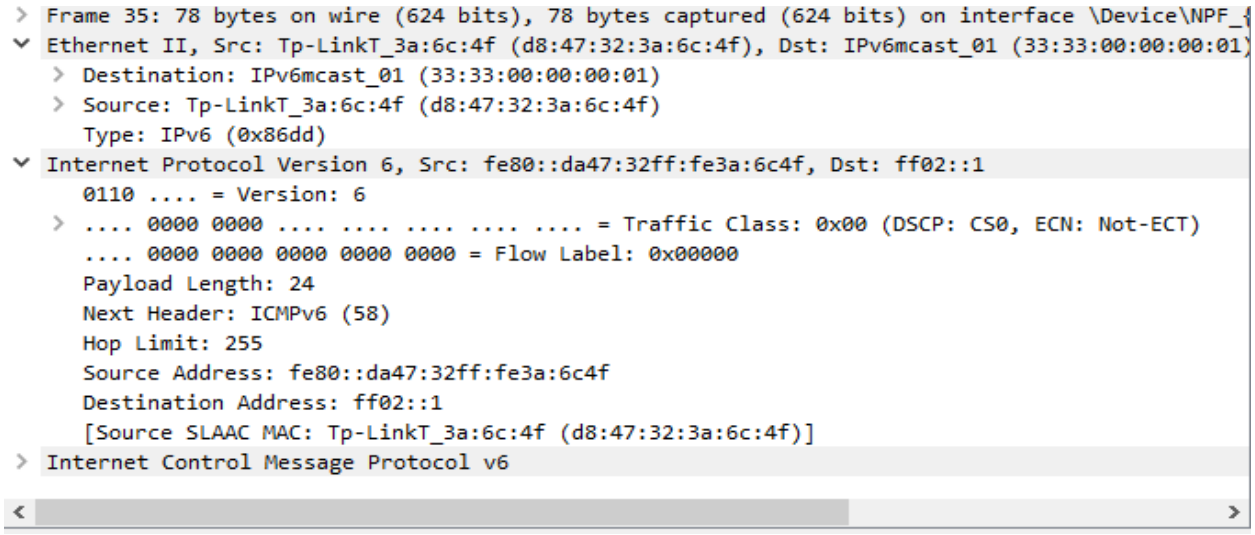
1. The **First part** contains a **menu bar and the options** displayed below it. This part is at the top of the window. File and the capture menus options are commonly used in Wireshark. The capture menu allows starting the capturing process. And the File menu is used to open and save a capture file.

2. The **second part** is **the packet listing** window. It determines the packet flow or the captured packets in the traffic. It includes the packet number, time, source, destination, protocol, length, and info. We can sort the packet list by clicking on the column name.

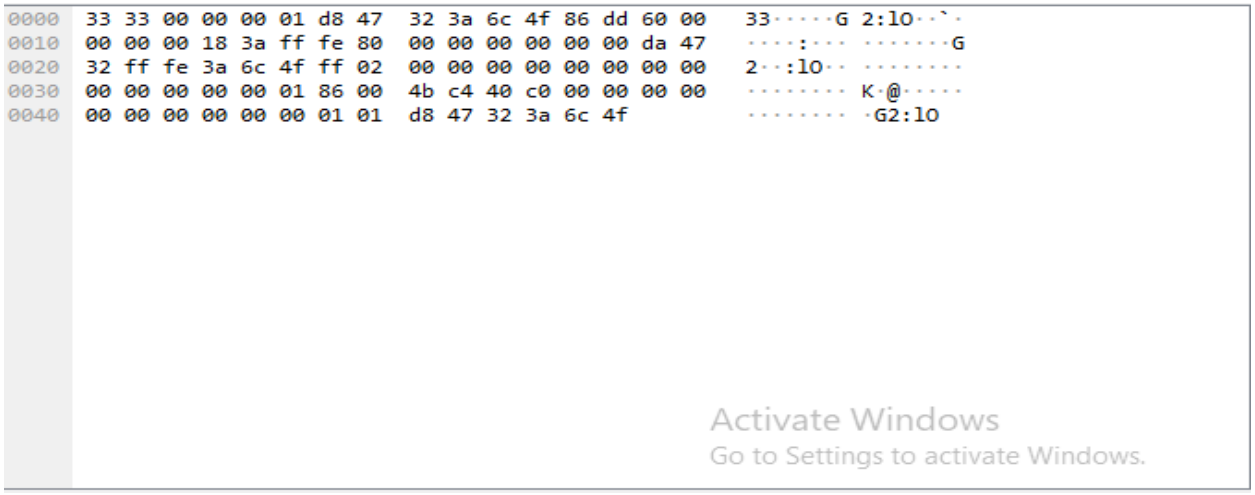


No.	Time	Source	Destination	Protocol	Length	Info
27	4.145675	HewlettP_71:b6:b4	Broadcast	ARP	60	Who has 192.168.1.2? Tell 169.254.47.221
28	4.539802	fe80::75fe:e100:aaf…	ff02::fb	MDNS	105	Standard query 0x0000 PTR _microsoft_mcc._tcp.local, "QM" question
29	4.539805	192.168.1.12	224.0.0.251	MDNS	85	Standard query 0x0000 PTR _microsoft_mcc._tcp.local, "QM" question
30	4.681634	HewlettP_71:b6:b4	Broadcast	ARP	60	Who has 192.168.1.2? Tell 169.254.47.221
31	4.916821	ASUSTekC_97:28:1d	Broadcast	ARP	60	Who has 192.168.1.53? Tell 192.168.1.229
32	4.917949	ASUSTekC_97:28:1d	Broadcast	ARP	60	Who has 192.168.1.23? Tell 192.168.1.229
33	4.961065	HewlettP_ae:63:54	Broadcast	ARP	60	Who has 192.168.1.38? Tell 192.168.1.192
34	5.386884	192.168.1.185	224.0.0.251	MDNS	103	Standard query 0x0055 PTR _233637DE._sub._googlecast._tcp.local, "QM" question PTR _googlecast._tcp.local, …
35	5.538837	fe80::da47:32ff:fe3…	ff02::1	ICMPv6	78	Router Advertisement from d8:47:32:3a:6c:4f
36	5.684235	HewlettP_71:b6:b4	Broadcast	ARP	60	Who has 192.168.1.2? Tell 169.254.47.221
37	5.686171	ASUSTekC_ce:1f:db	Broadcast	ARP	60	Who has 192.168.1.2? Tell 192.168.1.107
38	5.873602	ASUSTekC_97:28:1d	Broadcast	ARP	60	Who has 192.168.1.53? Tell 192.168.1.229
39	5.873602	ASUSTekC_97:28:1d	Broadcast	ARP	60	Who has 192.168.1.23? Tell 192.168.1.229
40	5.986568	HewlettP_ae:63:54	Broadcast	ARP	60	Who has 192.168.1.38? Tell 192.168.1.192
41	6.448667	ASUSTekC_ce:1f:db	Broadcast	ARP	60	Who has 192.168.1.2? Tell 192.168.1.107
42	6.658199	fe80::c206:c3ff:fe1…	ff02::1	ICMPv6	78	Router Advertisement from c0:06:c3:16:4e:78
43	6.867712	ASUSTekC_97:28:1d	Broadcast	ARP	60	Who has 192.168.1.53? Tell 192.168.1.229

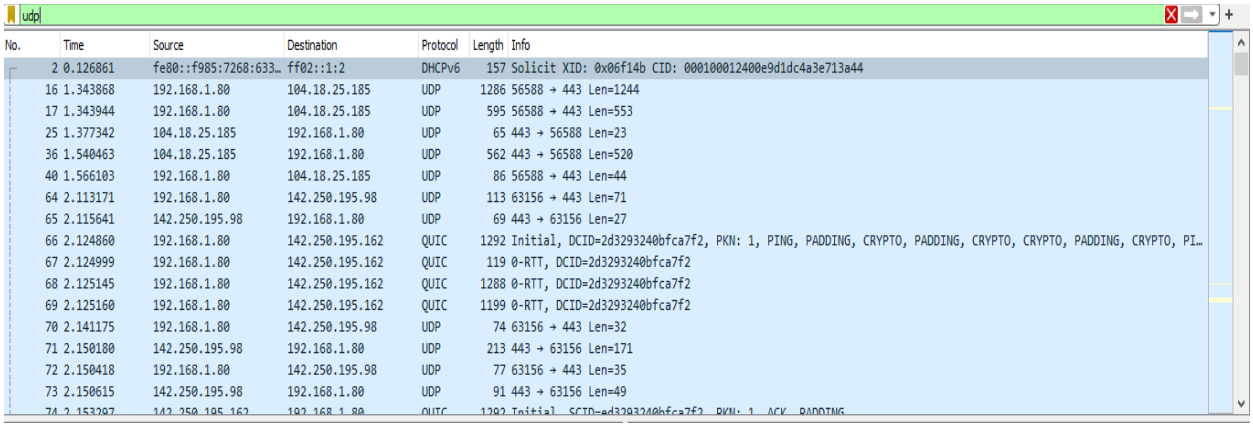
3. **Third** is the **packet header - detailed** window. It contains detailed information about the components of the packets. The protocol info can also be expanded or minimized according to the information required.



4. The **fourth part** is called the **packet contents** window, which displays the content in ASCII and hexadecimal format.



5. At last, **Fifth part** is the **filter field** which is at the top of the display. The captured packets on the screen can be filtered based on any component according to your requirements.
For example, if we want to see only the packets with the UDP protocol, we can apply filters to that option. All the packets with UDP as the protocol will only be displayed on the screen, shown below:



Analyzing data packets on Wireshark:

When you click on a packet, the other two panes change to show you the details about the selected packet. You can also tell if the packet is part of a conversation. Here are details about each column in the top pane:

No.: This is the number order of the packet captured. The bracket indicates that this packet is part of a conversation.

Time: This column shows how long after you started the capture this particular packet was captured.

Source: This is the address of the system that sent the packet.

Destination: This is the address of the packet destination.

Protocol: This is the type of packet. For example: TCP, DNS, DHCPv6, or ARP.

Length: This column shows you the packet's length, measured in bytes.

Info: This column shows you more information about the packet contents, which will vary depending on the type of packet.

Most used Filters in Wireshark:

Whenever we type any commands in the filter command box, it turns **green** if your command is correct. It turns **red** if it is incorrect or the Wireshark does not recognize your command.

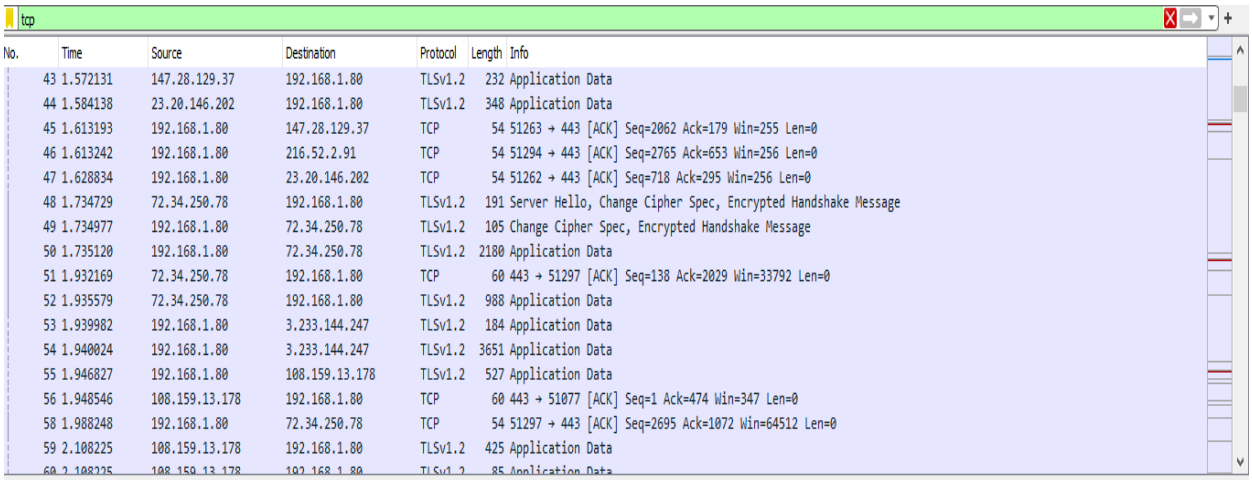
Filters	Description
1. ip.addr Example- ip.addr==10.0.10.142 ip.src	It is used to specify the IP address as the source or the destination. This example will filter based on this IP address as a source and a destination. If we want for a particular source or destination then, It is used for the source filter.

ip.dst	It is used for the destination.
2. protocol – dns , http Example- dns or http 'Dns and http' is never used.	This command filters based on the protocol. It requires the packet to be either dns protocol or http protocol and will display the traffic based on this. We would not use the command 'dns and http' because it requires the packet to be both, dns as well as http, which is impossible.
3. tcp.port Example: tcp.port==443	It sets filter based on the specific port number. It will filter all the packets with this port number.
4. udp.port	It is same as tcp.port. Instead, udp is used.
5. tcp.analysis.flags	Wireshark can flag TCP problems. This command will only display the issues that Wireshark identifies. Example, packet loss, tcp segment not captured, etc. are some of the problems. It quickly identifies the problem and is widely used.
6. !() For example, !(arp or dns or icmp)	It is used to filter the list of protocols or applications, in which we are not interested. It will remove arp, dns, and icmp, and only the remaining will be left or it clean the things that may not be helpful.
7. Select any packet. Right-click on it and select 'Follow' and then select' TCP stream.'	It is used if you want to work on a single connection on a TCP conversation. Anything related to the single TCP connection will be displayed on the screen.
8. tcp contains the filter For example- tcp contains Facebook Or	It is used to display the packets which contain such words. In this, Facebook word in any packet in this trace file i.e., finding the devices, which are talking to Facebook. This command is useful if you are looking for a

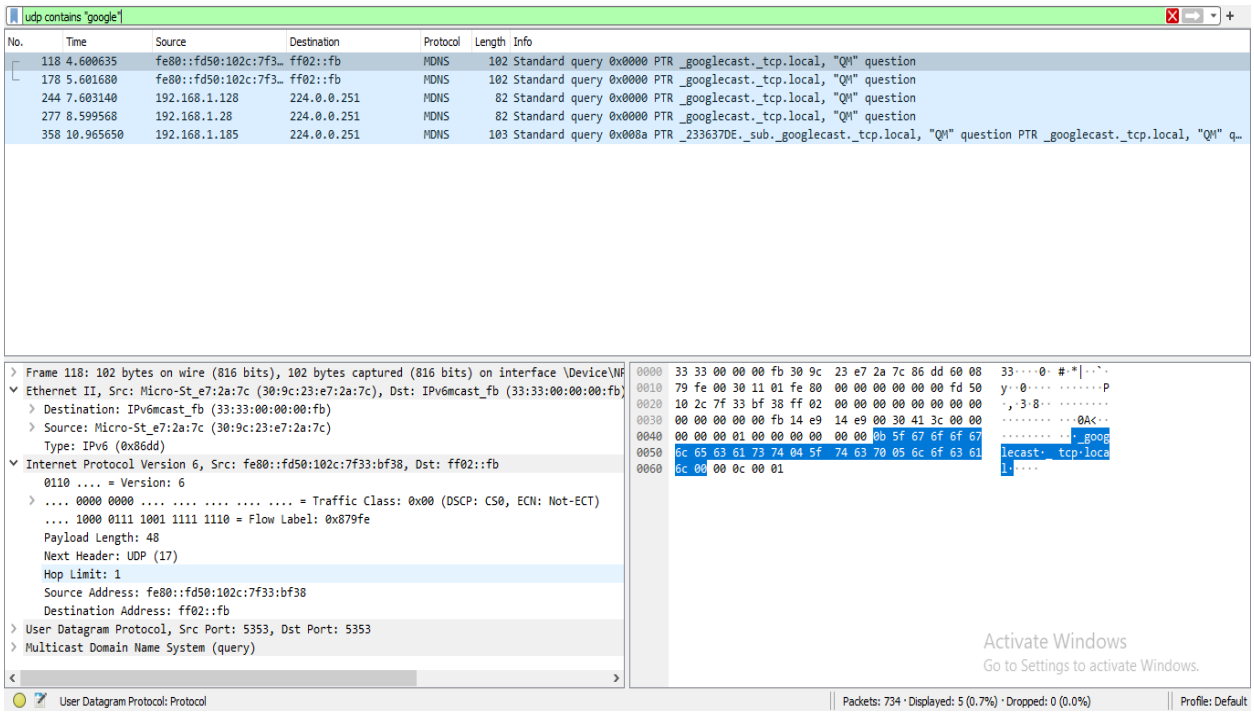
udp contains Facebook	username, word, etc.
http.request For the responses or the response code, you can type http.**response.code** = =200	It will display all the http requests in the trace file. You can see all the servers, the client is involved.
tcp.flags.syn==1 tcp.flags.reset	This will display all the packets with the sync built-in tcp header set to 1. This will show all the packets with tcp resets.

Eg.

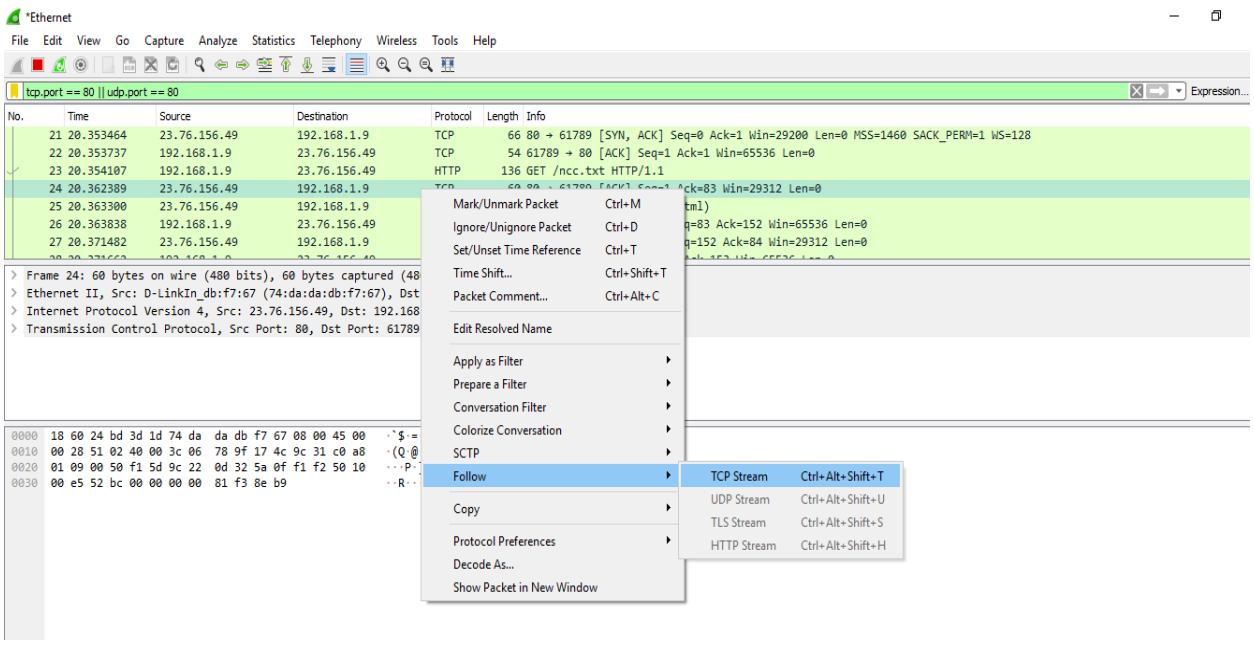
TCP packet capture:



UDP containing the keyword:

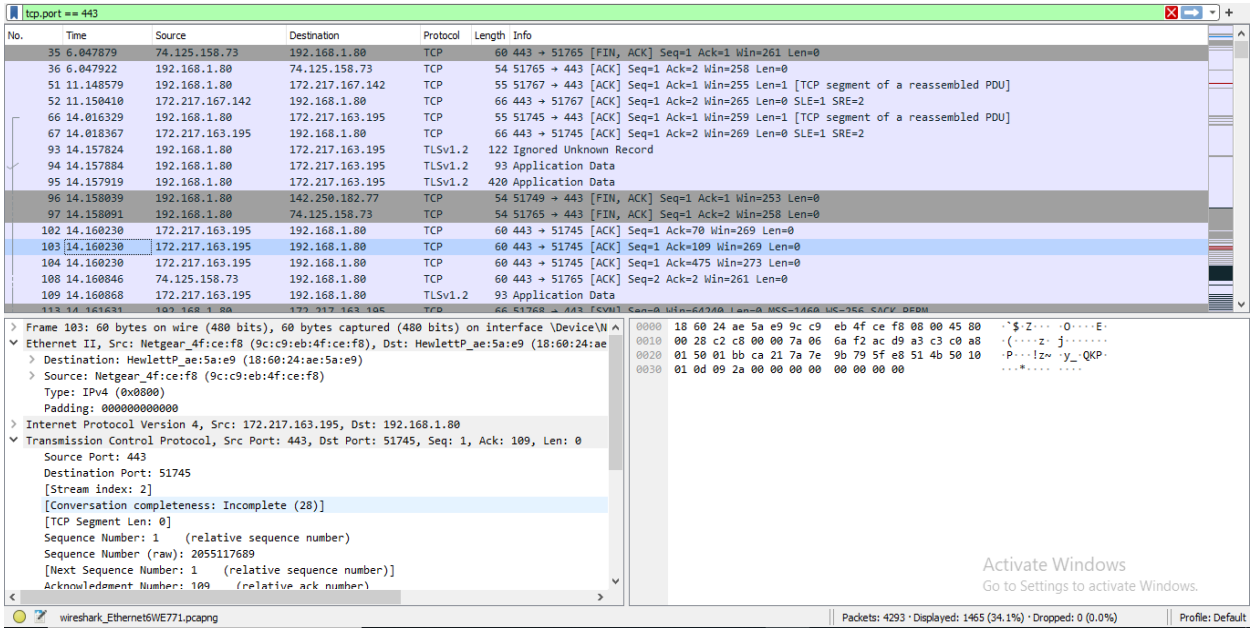


TCP Stream:

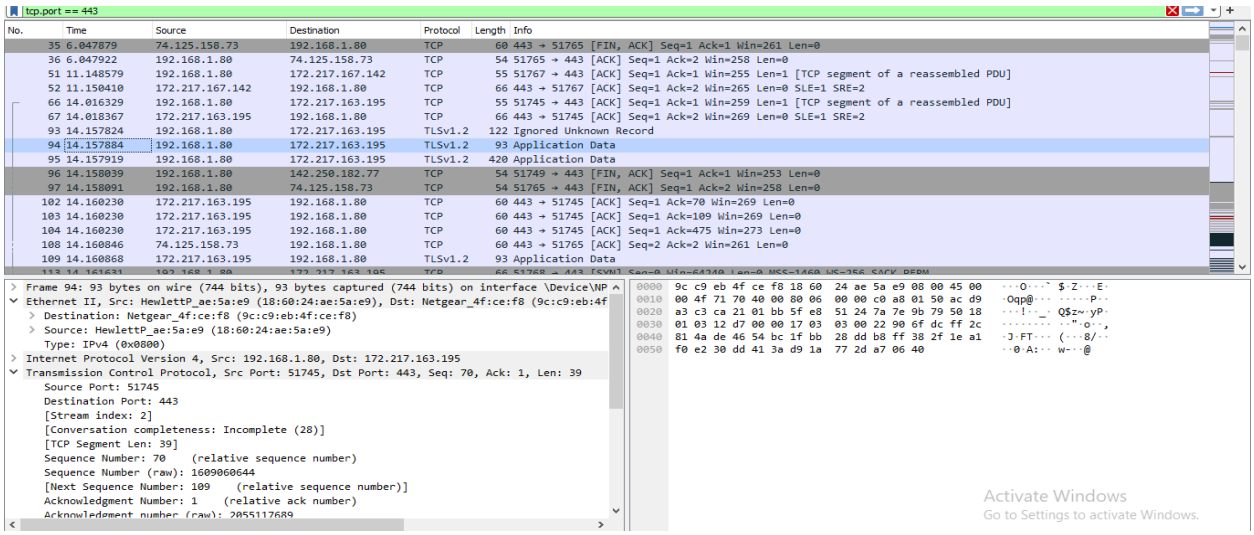


Tcp port

tcp port = = 443
Lists all the tcp connections from port 443



In the above figure, **source port:443**



In the above figure, we can find the **destination port: 443**

Result:

Thus, the wireshark installation was carried out successfully, and we observed the data transfer between client and server using wireshark. The TCP / UDP packets were analyzed successfully using wireshark.

Ex.no.: 5 **SHA-1 ALGORITHM**

Date:

Aim:

To calculate the message digest (hash) of a text using the SHA-1 algorithm in Java.

Algorithm:

1. Append Padding bits.
2. Append Length - 64 bits are appended to the end.
3. Prepare Processing Functions.
4. Prepare Processing Constants.
5. Initialize Buffers.
6. Processing Message in 512-bit blocks (L blocks in total message).

Program:

```
import java.util.Scanner;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
public class sha1
{
public static void main(String[] args)throws NoSuchAlgorithmException
{
Scanner sc = new Scanner(System.in);
System.out.println("Enter the String:");
String message = new String();
message = sc.next();
System.out.println("Mesage Digest is=");
System.out.println(sha1(message));
}
static String sha1(String input)throws NoSuchAlgorithmException
{
MessageDigest mDigest = MessageDigest.getInstance("SHA1");
byte[] result = mDigest.digest(input.getBytes());
```

```
 StringBuffer sb = new StringBuffer();
for(int i = 0;i<result.length;i++)
{
sb.append(Integer.toString((result[i] & 0xff) + 0x100, 16).substring(1));
}
return sb.toString();
}
}
```

Output:

```
C:\Security Lab New\programs>java sha1
Enter the String:
CORONA VIRUS DISEASE
Mesage Digest Is =
7690b7ccb987f4b3f32d2b9e7e8a69db2d0ded02
```

Result:

Thus the Secure Hash Algorithm (SHA-1) has been executed successfully and the output was verified successfully.

Ex. no.: 6(a) **Performing packet sniffing (Eavesdropping) using wireshark**

Date:

Aim:

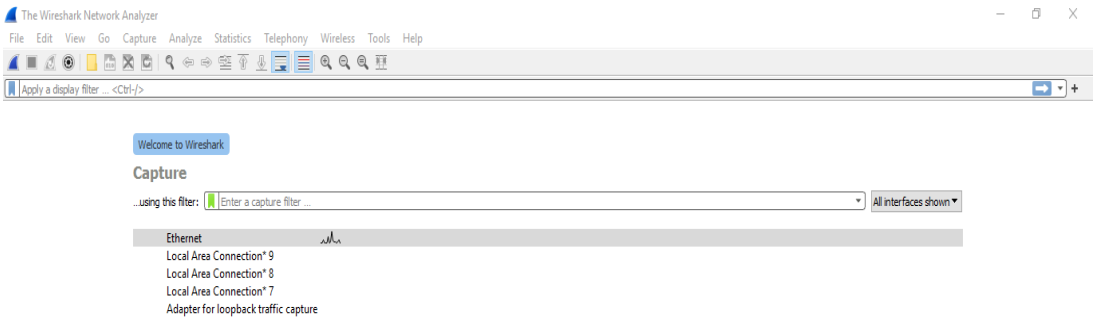
To use the wireshark tool and carry out an experiment on packet sniffing (eavesdropping) attack.

Procedure:

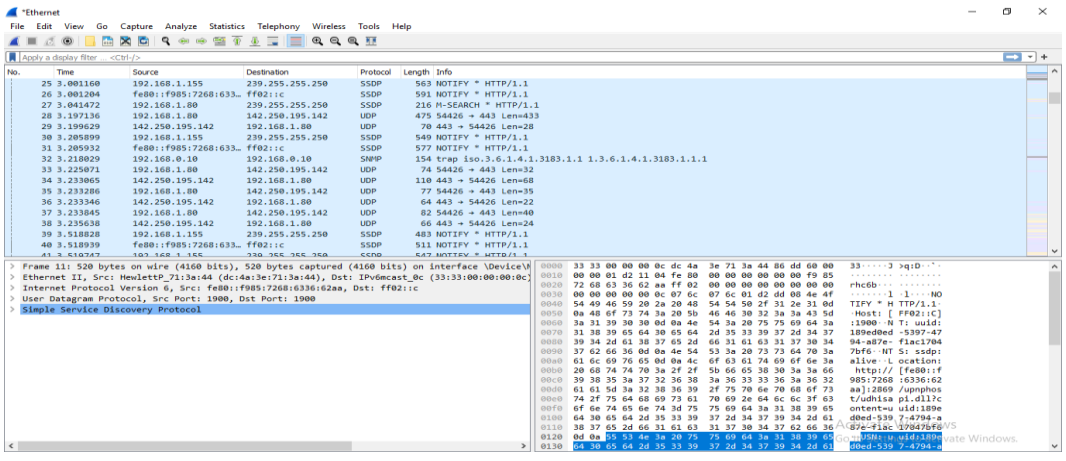
- Wireshark is a packet sniffing program that administrators can use to isolate and troubleshoot problems on the network.
- Packet sniffing is defined as the process to capture the packets of data flowing across a computer network.
- The Packet sniffer is a device or software used for the process of sniffing.
- It can also be used to capture sensitive data like usernames and passwords.
- It can also be used in wrong way (hacking) to eaves drop.

Step 1: Open the Wireshark Application.

Step 2: Select the Ethernet interface from the list.

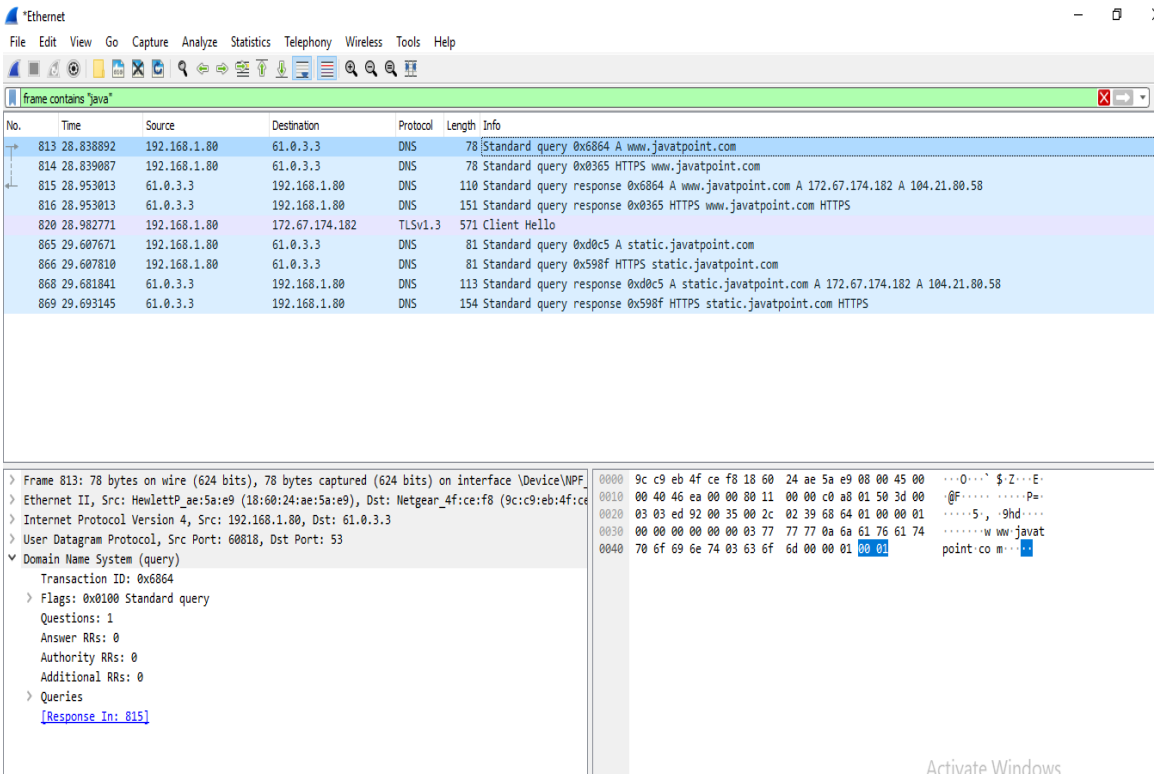


Step 3: Click on the capture button and start capturing the packets.



Step 4: Open the browser and type the address of any website (eg. www.javatpoint.com) , the traffic will start showing, and exchange of the packets will also start.

Step 5: In the filter tab apply **"http"** or **"dns"** filters, and we will see the packets exchanged in the network, for filtering a particular packet we can use the keywords associated with it. e.g**. frame contains "java"**

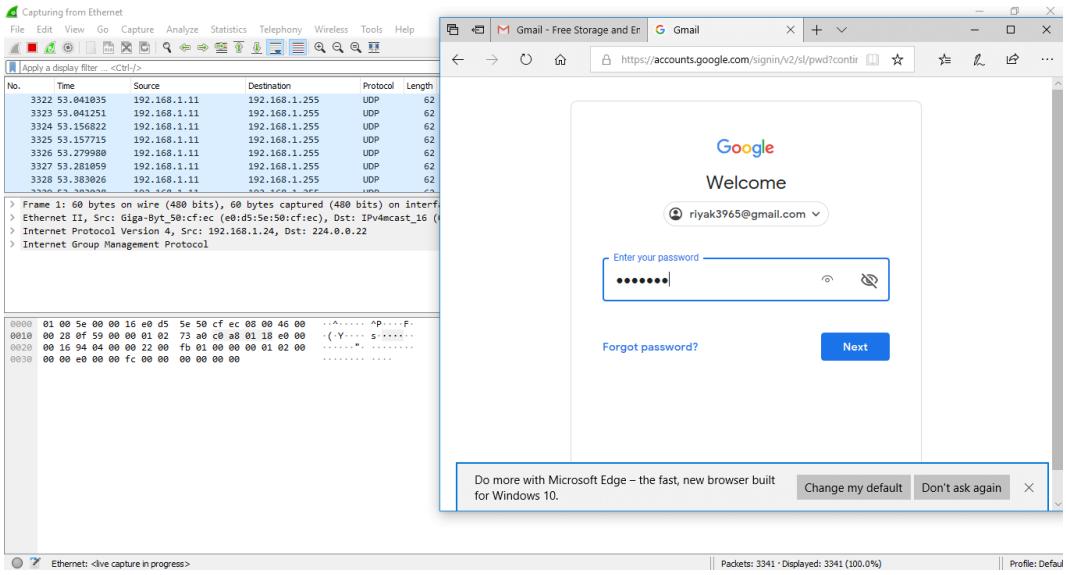


User Name and Password sniffing

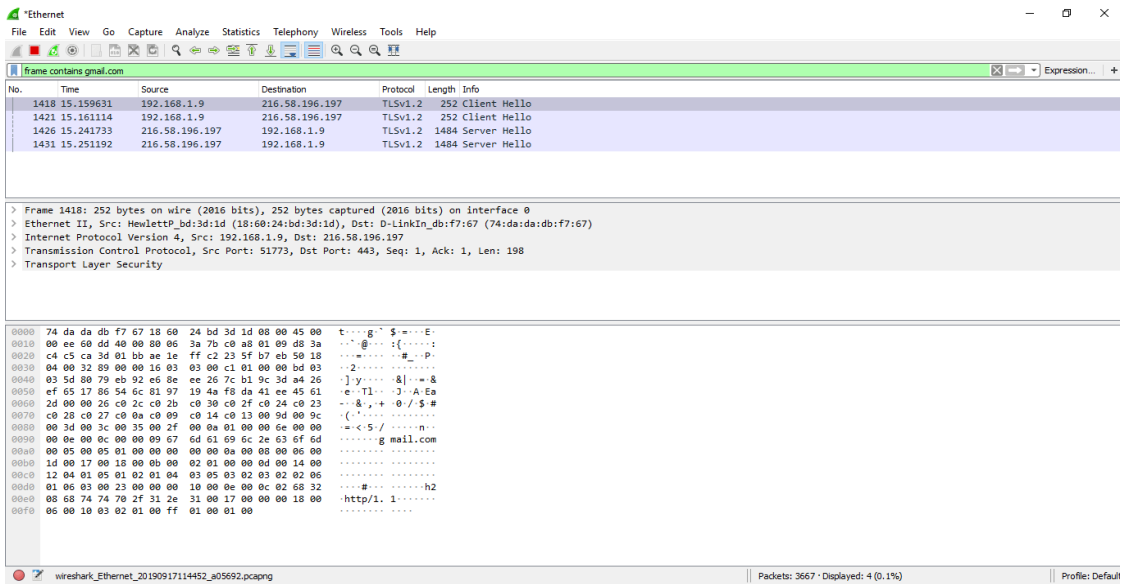
It is the process used to know the passwords and username for the particular website. Let's take an example of **gmail.com**. Below are the steps:

Step 1: Open the Wireshark and select the suitable interface.

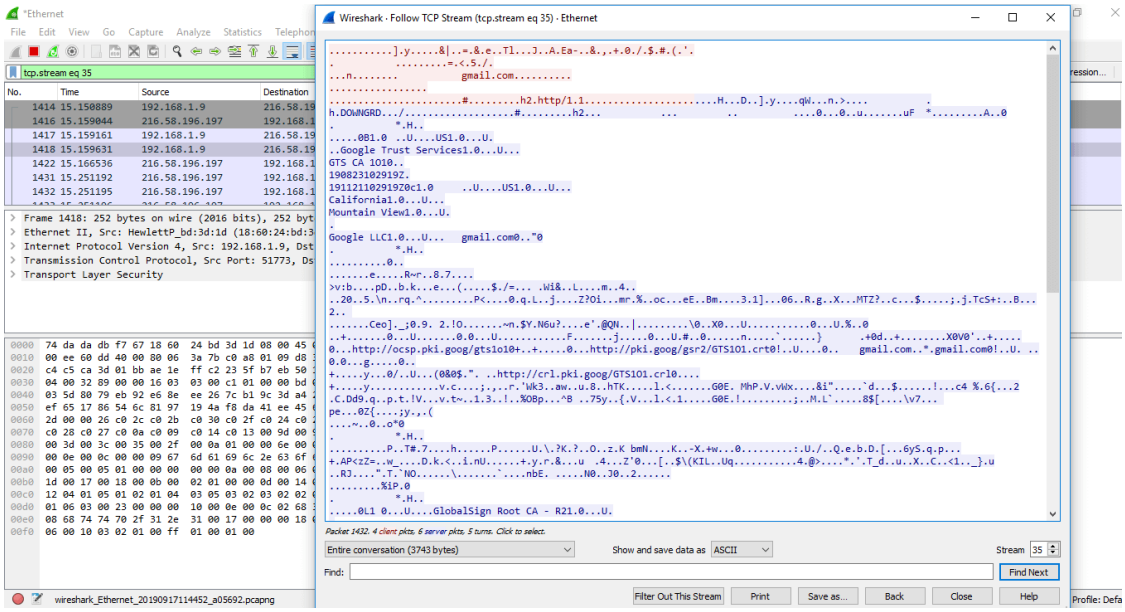
Step 2: Open the browser and enter the web address, we have entered gmail.com, which is highly secured.



Step 3: In the filters block of wireshark tool, enter the command **frame contains " gmail.com."**



Step 4: Right-click on the particular network and select 'Follow', and then 'TCP Stream.' You can see that all the data is secured in the encrypted form.



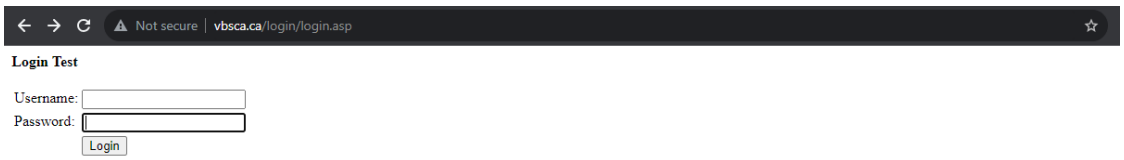
Since gmail.com is very secure, the password is stored in encrypted form and it is difficult to predict the password from it.

If it is an **unsecured network** then, the data will not be encrypted and we can view the user id and password.

Eg. **vbsca.ca**

Step 1: Open the Wireshark and select the suitable interface.

Step 2: Open the browser and enter the web address, vbsca.ca , the webpage will be has shown below.

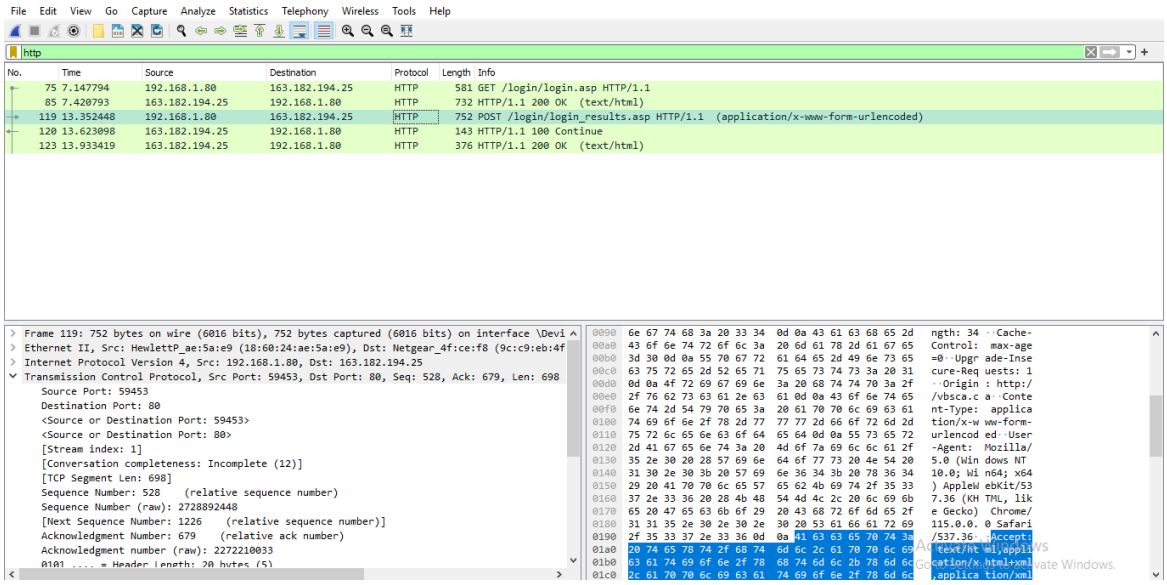


Type the user name as "**admin**" and password "**1234**", enter login.

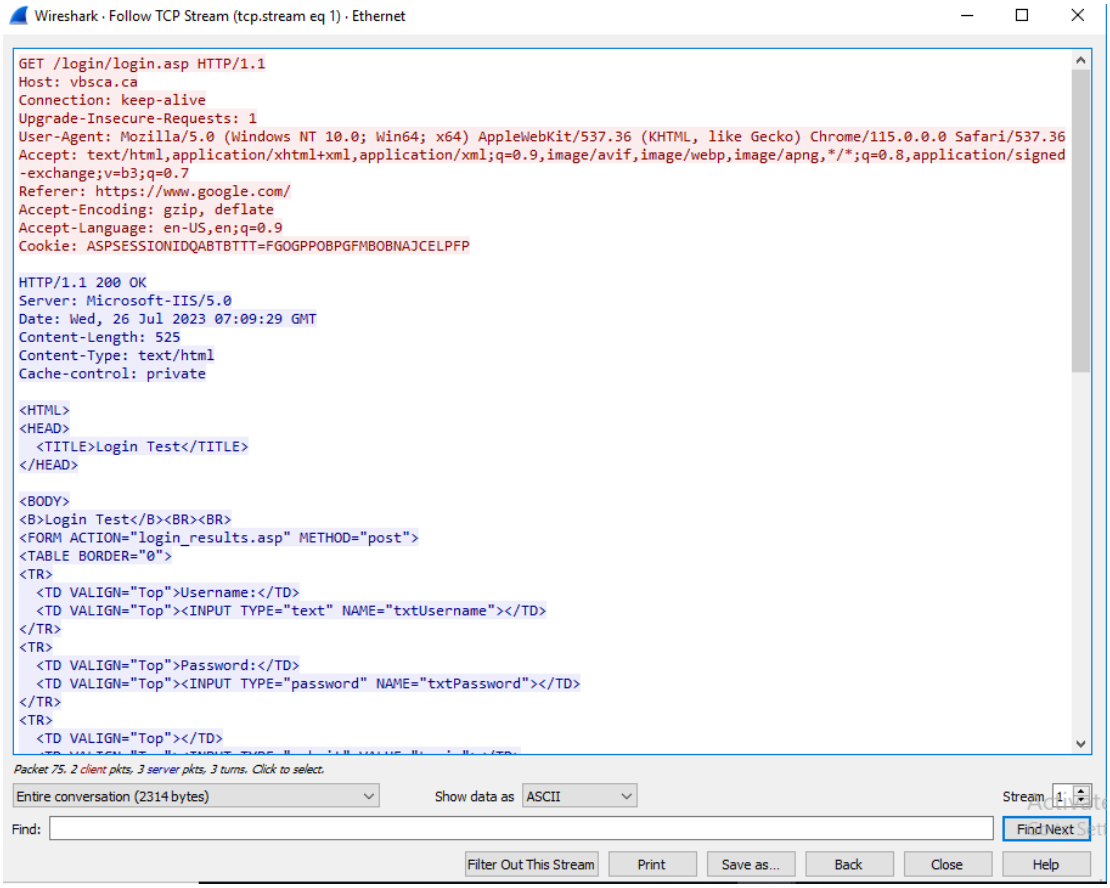
Login Test

Sorry, but the username that you entered does not exist.

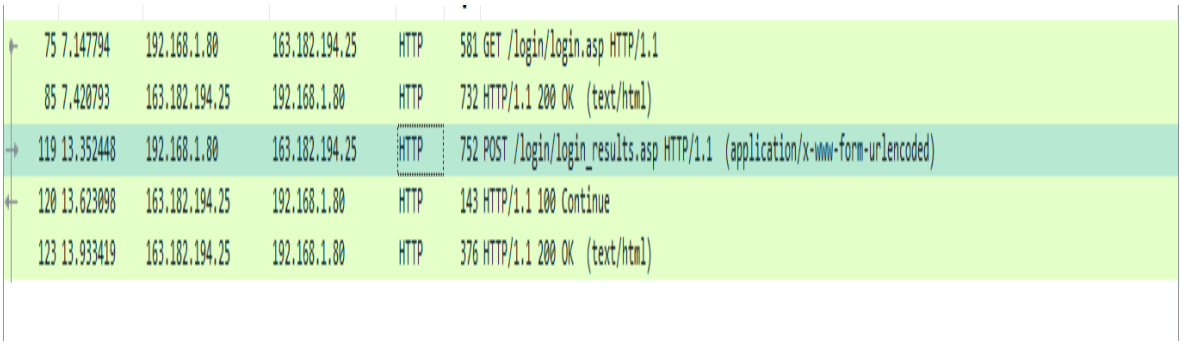
Step 3: In the filters block of wireshark tool, enter the command " http", to filter all the http packets.

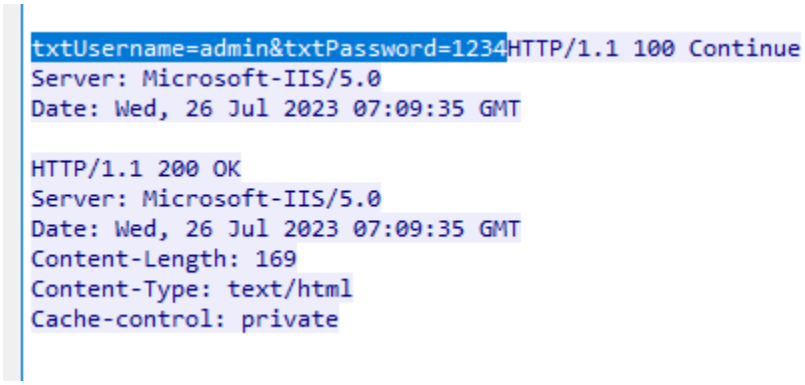


Step 4: Right-click on the particular network and select 'Follow', and then 'TCP Stream.' The data will not be encrypted and readable format.



If we select the packet with the '**post'** keyword and follow it, then we can find the user name and password entered in the browser.





This is possible since it is an unsecure website.

Result:

Thus, the experiment to carry out packet snipping (eavesdropping) attack using wireshark tool was carried out successfully and the output was verified.

Ex.no.: 6(b) **Simulating dictionary attack on password**

Date:

Aim:

To write a program that simulates a dictionary attack on a password by trying out a list of commonly used passwords and their variations.

Dictionary attack:

A dictionary attack is a method of breaking into a password-protected computer, network or other IT resource by systematically entering every word in a dictionary as a password. A dictionary attack can also be used in an attempt to find the key necessary to decrypt an encrypted message or document.

Algorithm:

Step 1: Start the program.

Step 2: Import the hashlib python library for generating the hash of the password.

Step 3: Generate a sample list of commonly used passwords and their variations.

Step 4: Encode the password to be attacked and compute its hash using SHA 256.

Step 5: Using loop, try out all possible combinations of common passwords and their variations.

Step 6: Compute the hash of the possible password generated.

Step 7: Compare the hash of the original password and the possible password, if there is a match return the password found, else return password is not found message.

Step 8: Stop the program.

Program:

```
import hashlib

# List of commonly used passwords and their variations

common_passwords = ["password", "password123", "letmein", "qwerty", "123456", "abc123", "admin", "welcome", "monkey", "sunshine"]

password_variations = ["", "123", "1234", "12345", "123456", "!", "@", "#", "$", "%", "^", "&", "*", "(", ")", "-", "_", "+", "=", "/", "\\", "|", "[", "]", "{", "}", "<", ">"]
```

```
# Hash of the password to be attacked
pas=input("Enter the password to be attacked:\n")
hashed_password = hashlib.sha256(pas.encode()).hexdigest()
# Try out all possible combinations of common passwords and their variations
for password in common_passwords:
    for variation in password_variations:
        possible_password = password + variation
        hashed_possible_password = hashlib.sha256(possible_password.encode()).hexdigest()
        if hashed_possible_password == hashed_password:
            print(f"Password found: {possible_password}")
            break
    else:
        continue
    break
else:
    print("Password not found")
```

Output:

```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

========== RESTART: C:/Users/Exam.DESKTOP-8RVNVAC.000/Desktop/ex1.py ==========
Enter the password to be attacked:
password
Password found: password

========== RESTART: C:/Users/Exam.DESKTOP-8RVNVAC.000/Desktop/ex1.py ==========
Enter the password to be attacked:
admin123
Password found: admin123

========== RESTART: C:/Users/Exam.DESKTOP-8RVNVAC.000/Desktop/ex1.py ==========
Enter the password to be attacked:
exam
Password not found
```

Result:

Thus the program to simulate a dictionary attack on password was executed successfully and the output was verified

Ex.no.: 7 **MAN IN THE MIDDLE ATTACK USING ARP POISONING**

Date:

Aim:

To implement man in the middle attack using ARP poisoning.

ARP Poisoning:

- Address Resolution Protocol (ARP) is a stateless protocol used for resolving IP addresses to machine MAC addresses. All network devices that need to communicate on the network broadcast ARP queries in the system to find out other machines' MAC addresses.
- The ARP cache is an important part of the ARP protocol. Once a mapping between a MAC address and an IP address is resolved as the result of executing the ARP protocol, the mapping will be cached. Therefore, there is no need to repeat the ARP protocol if the mapping is already in the cache.
- However, because the ARP protocol is stateless, the cache can be easily poisoned by maliciously crafted ARP messages. Such an attack is called the ARP cache poisoning attack.
- Fundamentally, there is no built-in form of authentication in ARP, therefore replies can be easily spoofed. By sending false ARP replies, it is easy to redirect traffic from a victim to yourself.
- At this point one can perform several attacks. One could drop the traffic, effectively performing a denial-of-service, listen to the traffic and forward it, sniffing the entire victim's traffic or could also modify the traffic before sending it.
- ARP Poisoning is also known as **ARP Spoofing**.

Procedure:

- **Ettercap** tool is used to perform ARP poisoning in LAN environment
- We need a VMware workstation installed with Kali Linux/Ubuntu OS and Ettercap tool to sniff the local traffic in LAN.

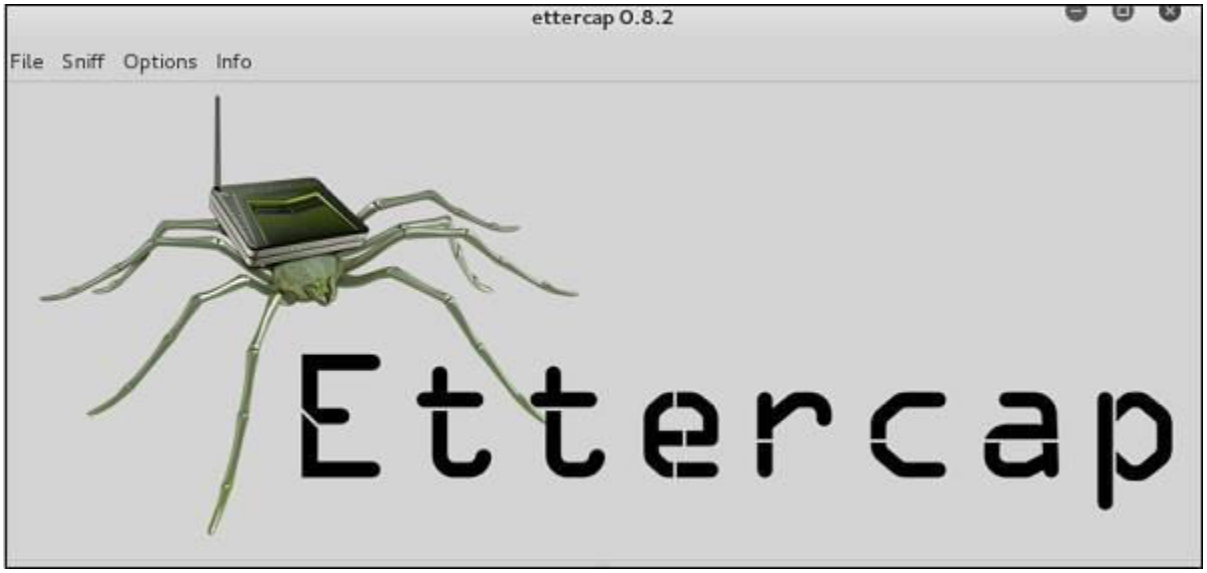
Step 1 − Install the VMware workstation and install the Kali Linux operating system.

Step 2 − Login into the Kali Linux using username pass "root, toor".

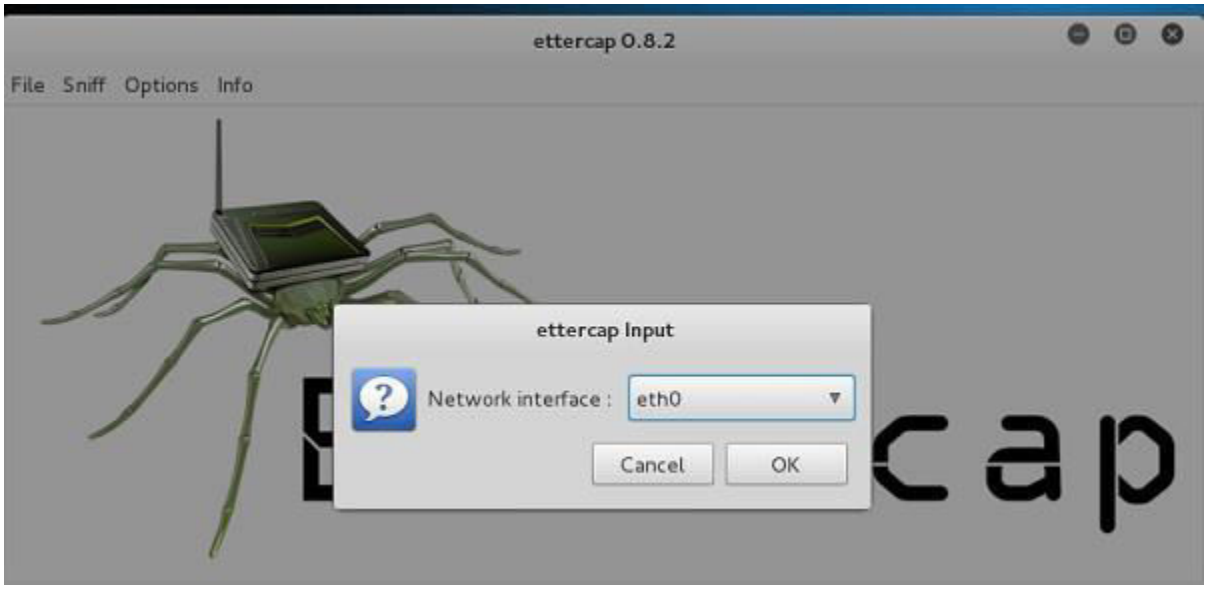
Step 3 − Make sure you are connected to local LAN and check the IP address by typing the command **ifconfig** in the terminal.

```
root@kali:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:cf:f8:e7
          inet addr:192.168.121.128  Bcast:192.168.121.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fecf:f8e7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:70 errors:0 dropped:0 overruns:0 frame:0
          TX packets:54 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4963 (4.8 KiB)  TX bytes:8868 (8.6 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:16 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:960 (960.0 B)  TX bytes:960 (960.0 B)
```

Step 4 − Open up the terminal and type “Ettercap –G” to start the graphical version of Ettercap.

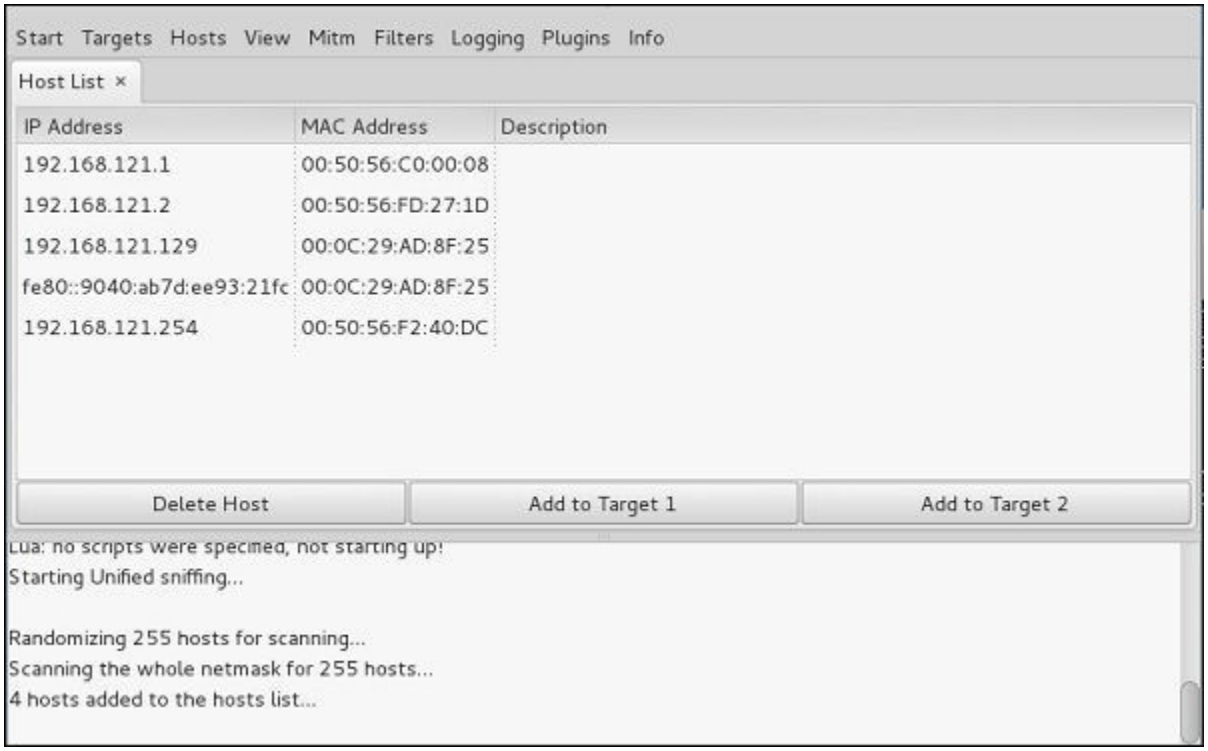


Step 5 − Now click the tab “sniff” in the menu bar and select “unified sniffing” and click OK to select the interface. We are going to use “eth0” which means Ethernet connection.



Step 6 − Now click the "hosts" tab in the menu bar and click "scan for hosts". It will start scanning the whole network for the alive hosts.

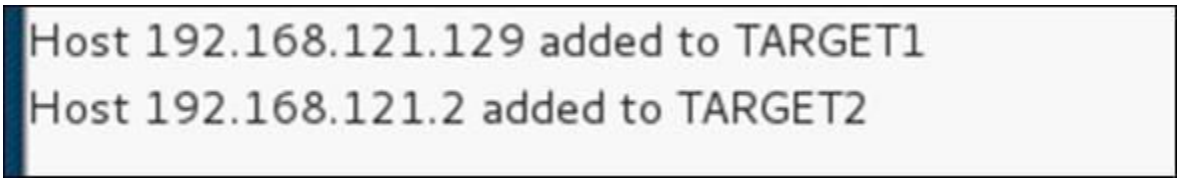
Step 7 − Next, click the "hosts" tab and select "hosts list" to see the number of hosts available in the network. This list also includes the default gateway address. We have to be careful when we select the targets.



Step 8 − Now we have to choose the targets. In MITM, our target is the host machine, and the route will be the router address to forward the traffic. In an MITM attack, the attacker intercepts the network and sniffs the packets. So, we will add the victim as "target 1" and the router address as "target 2."

In VMware environment, the default gateway will always end with "2" because "1" is assigned to the physical machine.

Step 9 − In this scenario, our target is "192.168.121.129" and the router is "192.168.121.2". So we will add target 1 as **victim IP** and target 2 as **router IP**.



Step 10 − Now click on "MITM" and click "ARP poisoning". Thereafter, check the option "Sniff remote connections" and click OK.

Step 11 − Click "start" and select "start sniffing". This will start ARP poisoning in the network which means we have enabled our network card in "promiscuous mode" and now the local traffic can be sniffed.

Step 12 − Now it's time to see the results; if our victim logged into some websites. We can see the results in the toolbar of Ettercap.

Result:

Thus, man in the middle attack by ARP poisoning has been successfully implemented.

Ex.no.: 8 **INTRUSION DETECTION SYSTEM (IDS)**

Date:

Aim:

To demonstrate Intrusion Detection System (IDS) using Snort software tool.

STEPS ON CONFIGURING AND INTRUSION DETECTION:

1. Download Snort from the Snort.org website. (http ://www.snort.org/snort-downloads)

2. Download Rules(https://www.snort.org/snort-rules). You must register to get the rules. (You should download these often)

3. Double click on the .exe to install snort. This will install snort in the "C:\Snort" folder.It is important to have WinPcap (http s://www.winpcap.org/insta ll/) installed

4. Extract the Rules file. You will need WinRAR for the .gz file.

5. Copy all files from the "rules" folder of the extracted folder. Now paste the rules into "C:\Snort\rules" folder.

6. Copy "snort.conf" file from the "etc" folder of the extracted folder. You must paste it into "C:\Snort\etc" folder. Overwrite any existing file. Remember if you modify your snort.conf file and download a new file, you must modify it for Snort to work.

7. Open a command prompt (cmd.exe) and navigate to folder "C:\Snort\bin" folder. (at the Prompt, type cd\snort\bin)

8. To start (execute) snort in sniffer mode use following command:

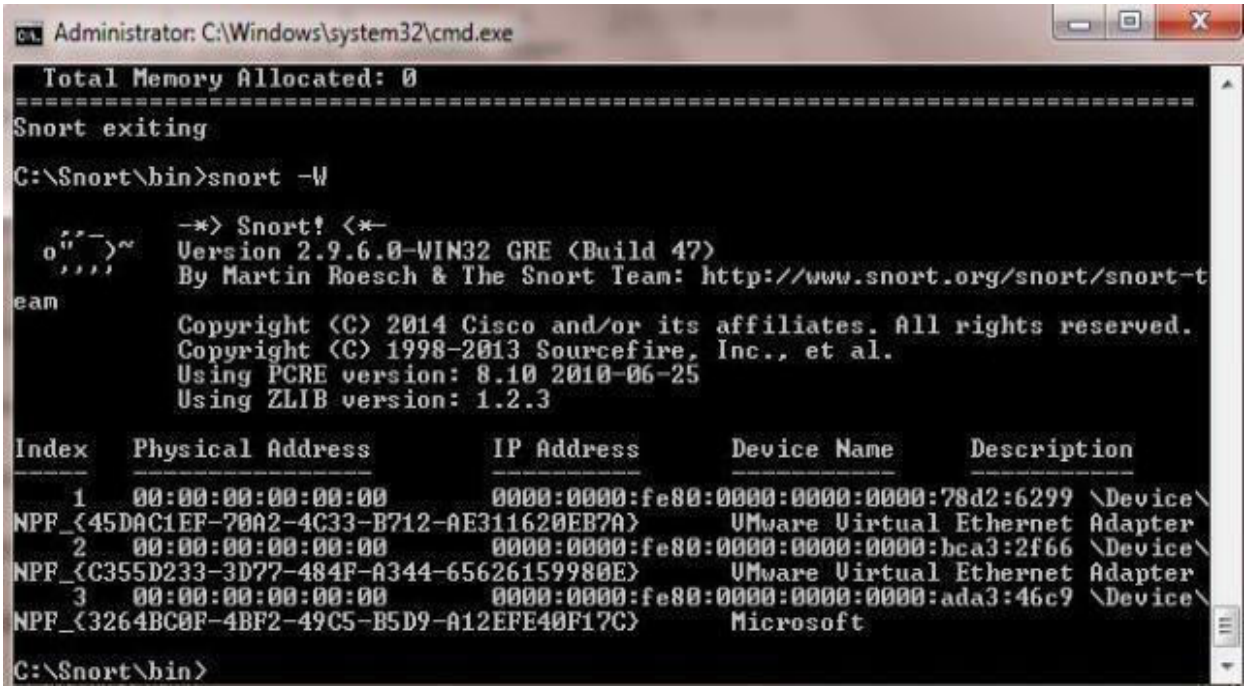
snort -dev -i 3

-i indicates the interface number. You must pick the correct interface number. In my case, it is 3.

-dev is used to run snort to capture packets on your network.

To check the interface list, use following command:

snort –W



Finding an interface

You can tell which interface to use by looking at the Index number and finding Microsoft. As you can see in the above example, the other interfaces are for VMWare. My interface is 3.

9. To run snort in IDS mode, you will need to configure the file “snort.conf” according to your network environment.

10. To specify the network address that you want to protect in snort.conf file, look for the following line.

var HOME_NET 192.168.1.0/24 (You will normally see any here)

11. You may also want to set the addresses of DNS_SERVERS, if you have some on your network.

Example:

example snort

12. Change the RULE_PATH variable to the path of rules folder. var RULE_PATH c:\snort\rules

path to rules

13. Change the path of all library files with the name and path on your system. and you must change the path of snort_dynamicpreprocessorvariable. C:\Snort\lib\snort_dynamiccpreprocessor

You need to do this to all library files in the “C:\Snort\lib” folder. The old path might be: “/usr/local/lib/…”. you will need to replace that path with your system path. Using C:\Snort\lib

14. Change the path ofthe “dynamicengine” variable value in the “snort.conf” file.

Example:

dynamicengine C:\Snort\lib\snort_dynamicengine\sf_engine.dll

15 Add the paths for “include classification.config” and “include reference.config” files.

include c:\snort\etc\classification.config include c:\snort\etc\reference.config

16. Remove the comment (#) on the line to allow ICMP rules, if it is commented with a #.

include $RULE_PATH/icmp.rules

17. You can also remove the comment of ICMP-info rules comment, if it is commented. include $RULE_PATH/icmp-info.rules

18. To add log files to store alerts generated by snort, search for the “output log” test in snort.conf and add the following line:

output alert_fast: snort-alerts.ids

19. Comment (add a #) the whitelist $WHITE_LIST_PATH/white_list.rules and the blacklist

Change the nested_ip inner , \ to nested_ip inner #, \

20. Comment out (#) following lines:

#preprocessor normalize_ip4

#preprocessor normalize_tcp: ips ecn stream

#preprocessor normalize_icmp4

#preprocessor normalize_ip6

#preprocessor normalize_icmp6

21. Save the “snort.conf” file.

22. To start snort in IDS mode, run the following command:

snort -c c:\snort\etc\snort.conf -l c:\snort\log -i 3 (Note: 3 is used for my interface card)

If a log is created, select the appropriate program to open it. You can use WordPard or

NotePad++ to read the file.

To generate Log files in ASCII mode, you can use following command while running snort in

IDS mode:

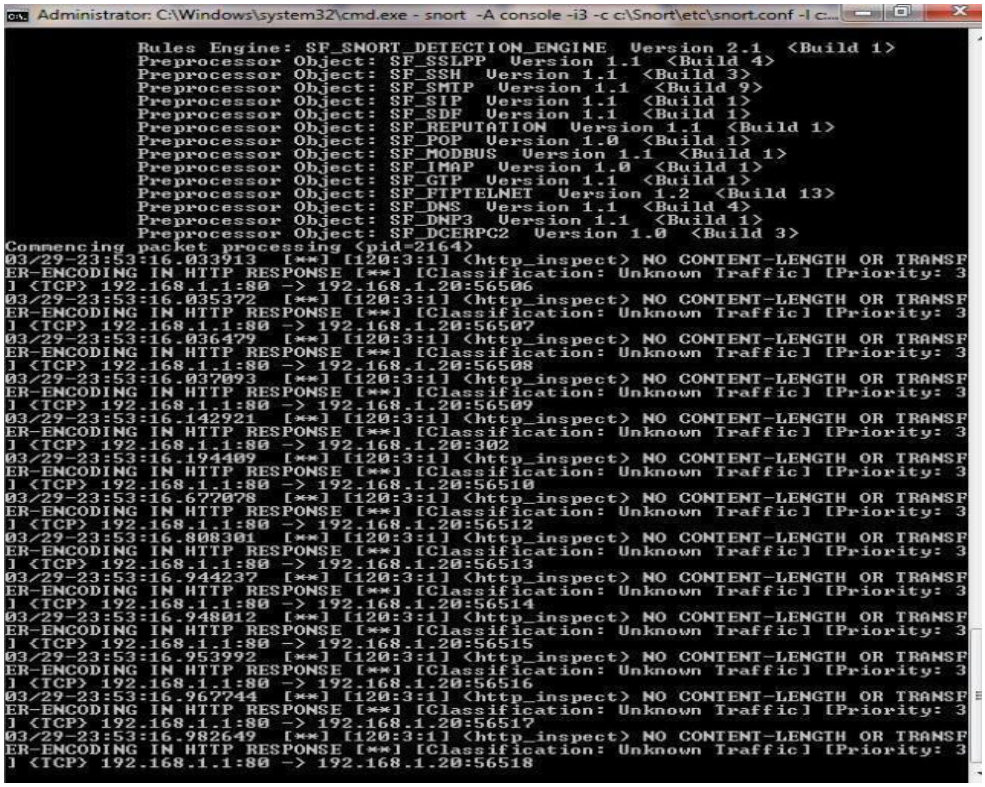
snort -A console -i3 -c c:\Snort\etc\snort.conf -l c:\Snort\log -K ascii

23. Scan the computer that is running snort from another computer by using PING or NMap

(ZenMap).

After scanning or during the scan you can check the snort-alerts.ids file in the log folder to insure it is logging properly. You will see IP address folders appear.

Snort monitoring traffic –



Result:

Thus the Intrusion Detection System (IDS) has been demonstrated using the Open Source Intrusion Detection Tool Snort.

Ex.no.: 9 **EXPLORING DIFFERENT NETWORK MONITORING TOOLS**

Date:

Aim:

To explore the different tools used for network monitoring.

Procedure:

Network monitoring:

Network monitoring refers to the practice of monitoring computer networks, both local area networks (LANs) and wide area networks (WANs), to ensure their smooth operation, performance, and security. It involves the continuous surveillance and analysis of network components, such as routers, switches, servers, and other network devices, to gather data and detect any abnormalities or issues.

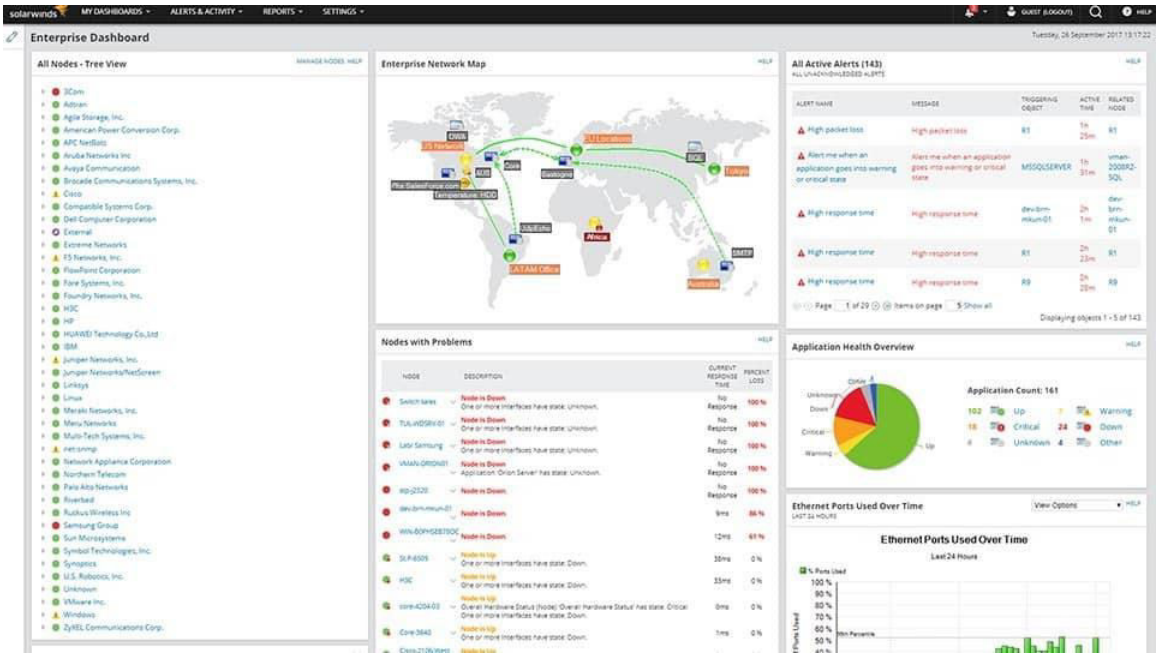
Network monitoring tools:

Network monitoring tools and software are used to collect and analyze network data, including network traffic, bandwidth usage, latency, packet loss, error rates, and device statistics. These tools provide valuable insights into network performance metrics, allowing administrators to identify bottlenecks, troubleshoot issues, and optimize network resources.

The following are some of the commonly used network monitoring tools:

1. SolarWinds Network Performance Monitor:

- SolarWinds Network Performance Monitor is a comprehensive network performance monitoring tool that can monitor the status of devices with SNMP. It can automatically discover network devices connected to the network.
- It identifies all devices connected to the network, maps them, and watches out for performance issues. This system will alert you of gathering problems and send a notification.
- Any devices, applications, or services that have been discovered can also be viewed on a network topology map where we can see how the infrastructure links together.
- The NetPath feature allows you to trace packet transfers hop-by-hop, which can help to diagnose the origin of performance network issues more effectively.



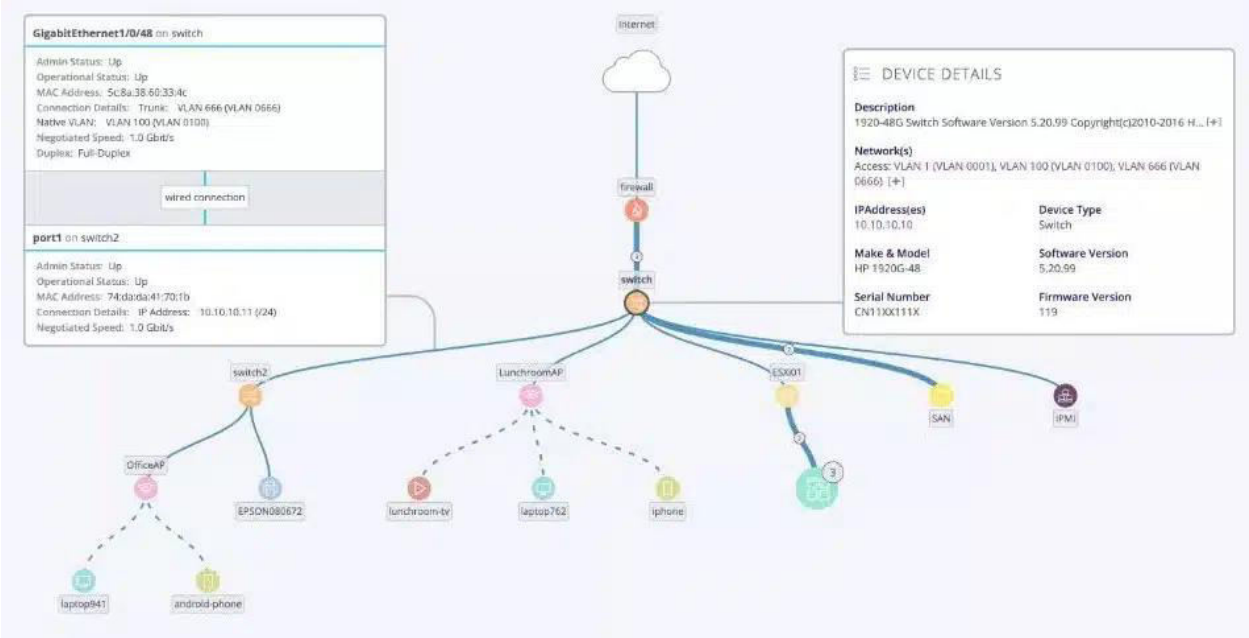
The main dashboard monitors the availability and performance of connected network devices

The network maps show broken connections that need attention in red.

2. Auvik:

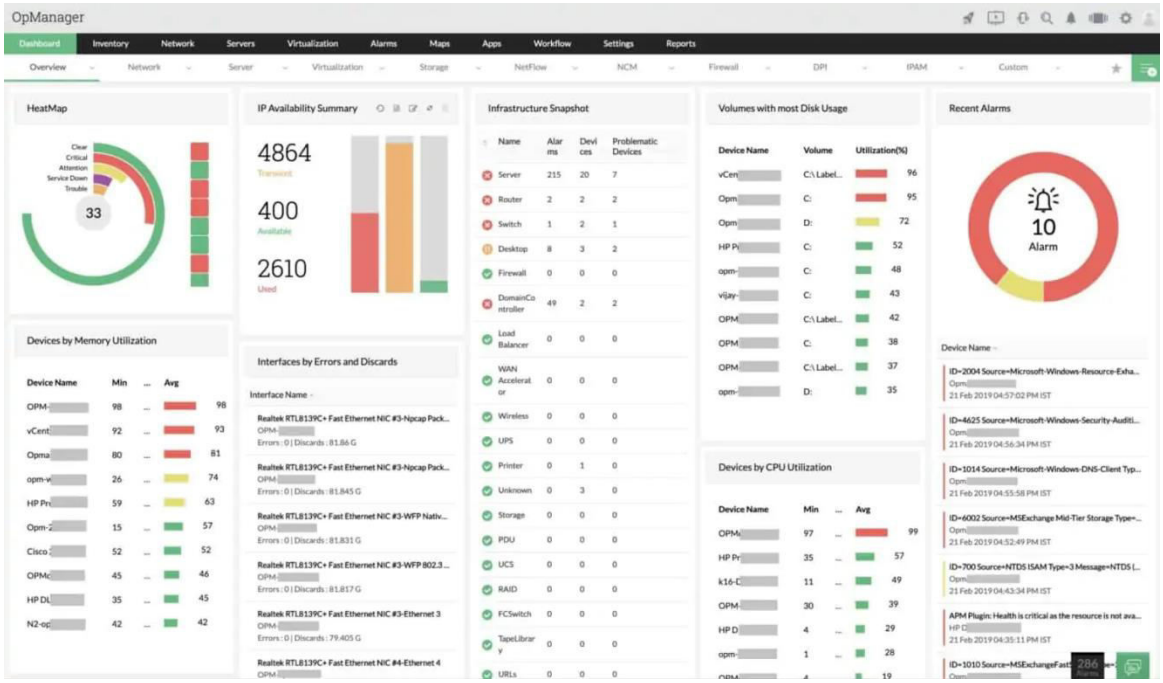
- Auvik is a cloud-based network monitoring system that includes a number of system management tools.
- The Auvik package is able to monitor multiple sites and centralizes their control. This makes the package ideal for monitoring a WAN.
- The great feature of Auvik is that its higher plan provides both traffic analysis and network device performance monitoring.

- From its cloud location, this package can watch over multiple sites, providing alerts that buy time to head off system disasters.



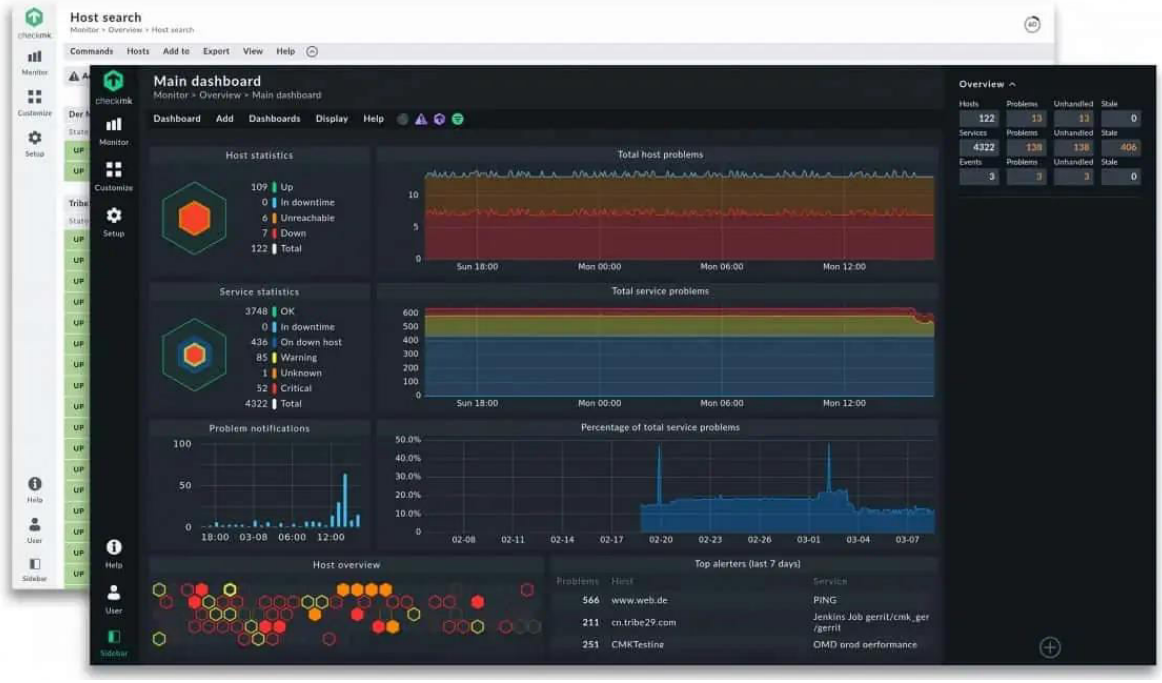
3. ManageEngine OpManager:

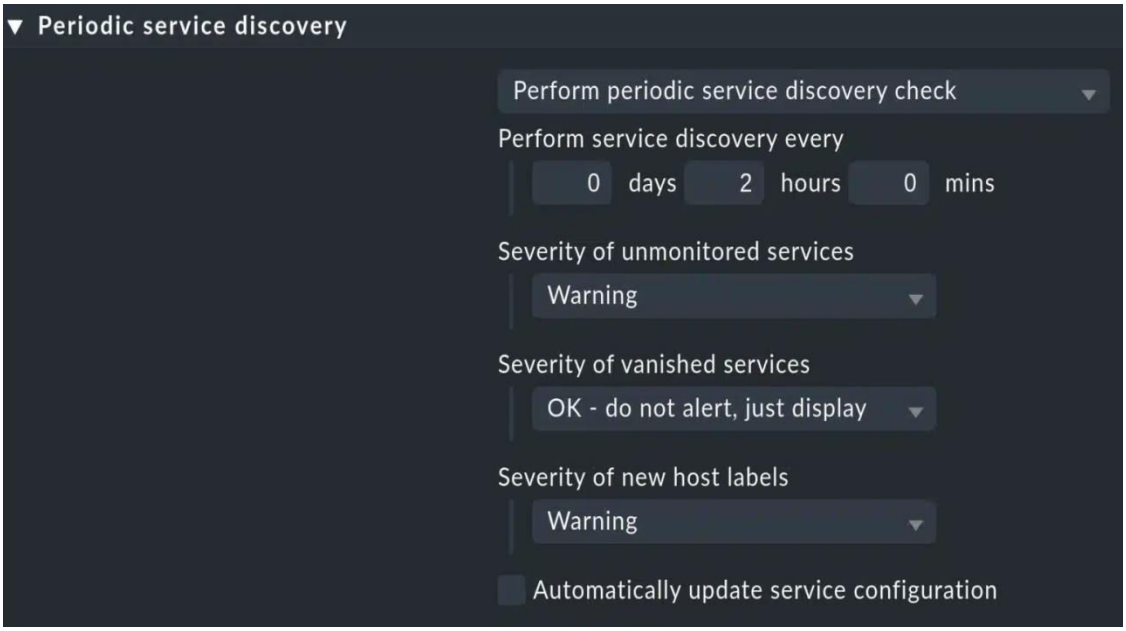
- ManageEngine OpManager is a network monitoring solution that can monitor the performance of network devices, servers, routers, switches, and virtual machines in real-time.
- Customizable dashboards provide over 200 widgets for you to create a unique monitoring experience.
- The ManageEngine OpManager system is one of the few detailed network monitoring systems that offers a graphical user interface for Linux.
- While most Linux network monitors are command-line systems, the high-quality graphs and charts of OpManager make status recognition easy.



4. Checkmk :

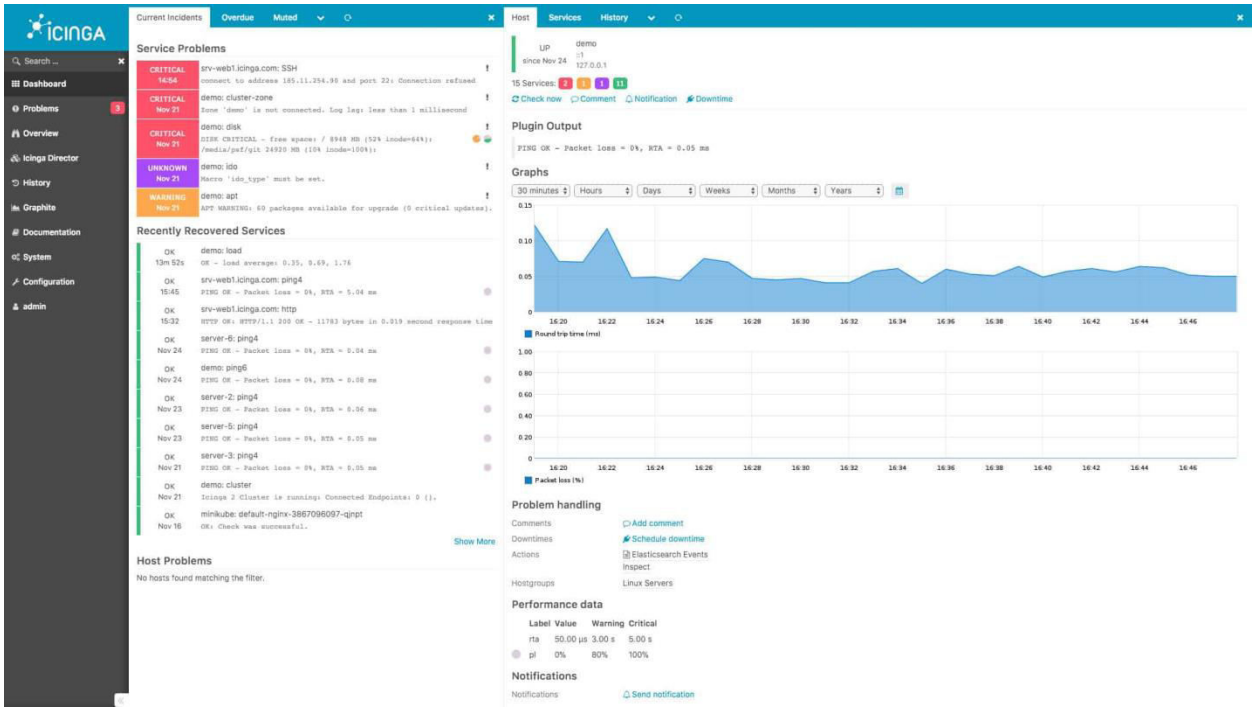
- Checkmk is a system monitoring package that is able to track the performance of networks, servers, and applications.
- The network monitor can be used for LANs and wireless networks, so it can also be useful for activity on networks that use both wired and wireless technology.
- Checkmk stands out due to its automated service discovery for infrastructure monitoring simplifying the initial setup and any ongoing management.
- When you connect Checkmk to a server, it automatically detects the server's operating system and the services running on it, and then suggests appropriate checks for those services.





5. Icinga:

- Icinga is an open-source network monitoring tool that monitors the performance of your network, cloud-service, and data center.
- The software is web-based and can be configured through the GUI or with the Domain Specific Language (DSL). Having the choice between the two gives you the power to monitor however you want.



Result:

Thus, some of the commonly used network monitoring tools have been explored and their features were studied.

Ex.no.: 10(a) **STUDY TO CONFIGURE FIREWALL**

Date:

Aim:

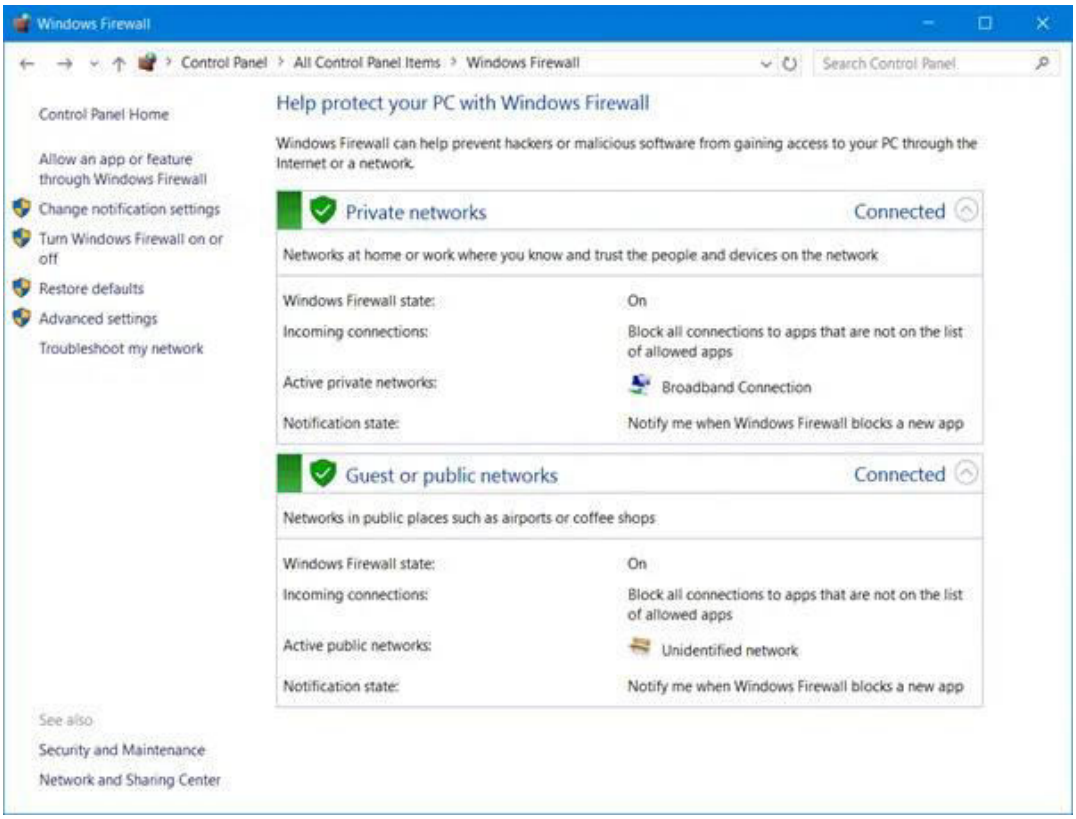
To study about the configuration of firewall.

Procedure:

A firewall is a software or hardware that checks information from the Internet or a network. Depending on your firewall settings, it blocks or allows it to pass through to your computer. A firewall can help prevent hackers or malicious software from gaining access to your computer through a network or the Internet. A firewall can also help stop your computer from sending malicious software to other computers.

How to configure Windows Firewall:

You can customize most settings of your Windows Firewall through the left pane of the Firewall applet in Control Panel.



1. Turn on or off Windows Firewall

This setting is selected by default. When Windows Firewall is On, most programs are blocked from communicating through the firewall.
To turn off Windows Firewall, open Control Panel and click on the Windows Firewall applet. Here, clicking on the Turn Firewall On or Off setting in Control Panel will enable or disable the Windows Firewall on your computer.

2. Block all incoming firewall connections, including those in the list of allowed programs

This setting blocks all unsolicited attempts to connect to your computer. Use this setting when you need maximum protection for your computer, such as when you connect to a public network in a hotel or airport, or when a computer worm is spreading over the Internet. With this setting, you are not notified when Windows Firewall blocks programs, and programs in the list of allowed programs are ignored. When you block all incoming connections, you can still view most web pages, send and receive an e-mail, and send and receive instant messages.

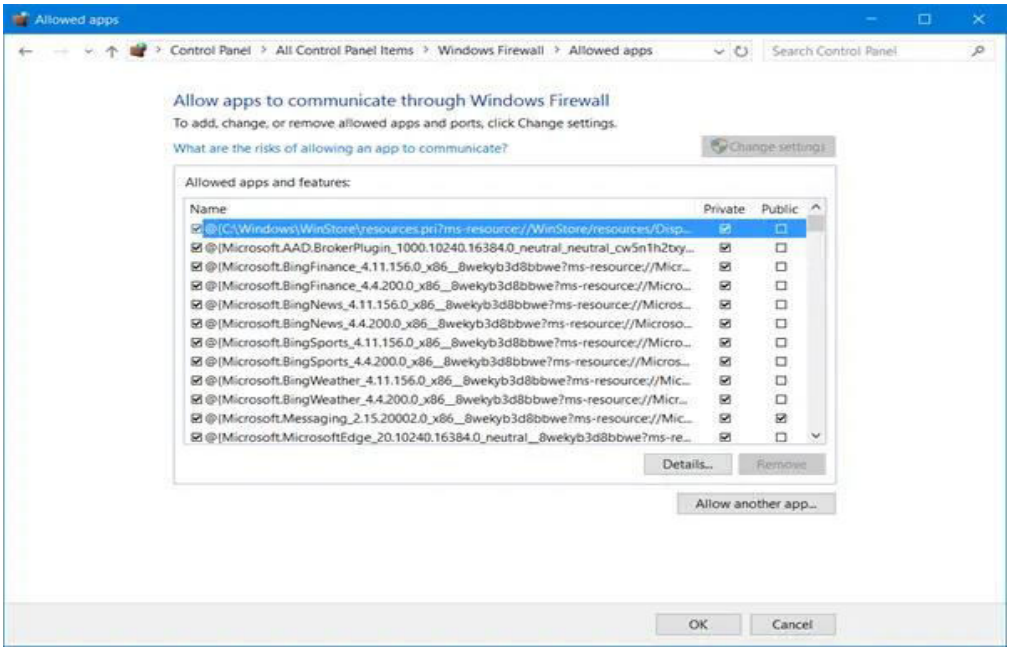
3. Turn off Windows Firewall

Avoid using this setting unless you have another firewall running on your computer. Turning off Windows Firewall might make your computer more vulnerable to damage from hackers and malicious software. Clicking on the Turn Firewall On or Off will let you enable or disable the Windows Firewall on your computer.

4. Block or Allow Programs through the Windows Firewall

By default, most programs are blocked by Windows Firewall to help make your computer more secure. To work properly, some programs might require you to allow them to communicate through the firewall. Here's how to do that:

Click Allow an app or feature through Windows Firewall. If you are prompted for an administrator password or confirmation, type the password or provide confirmation.



Select the check box next to the program you want to allow, select the network location types you want to allow communication on, and then click OK.

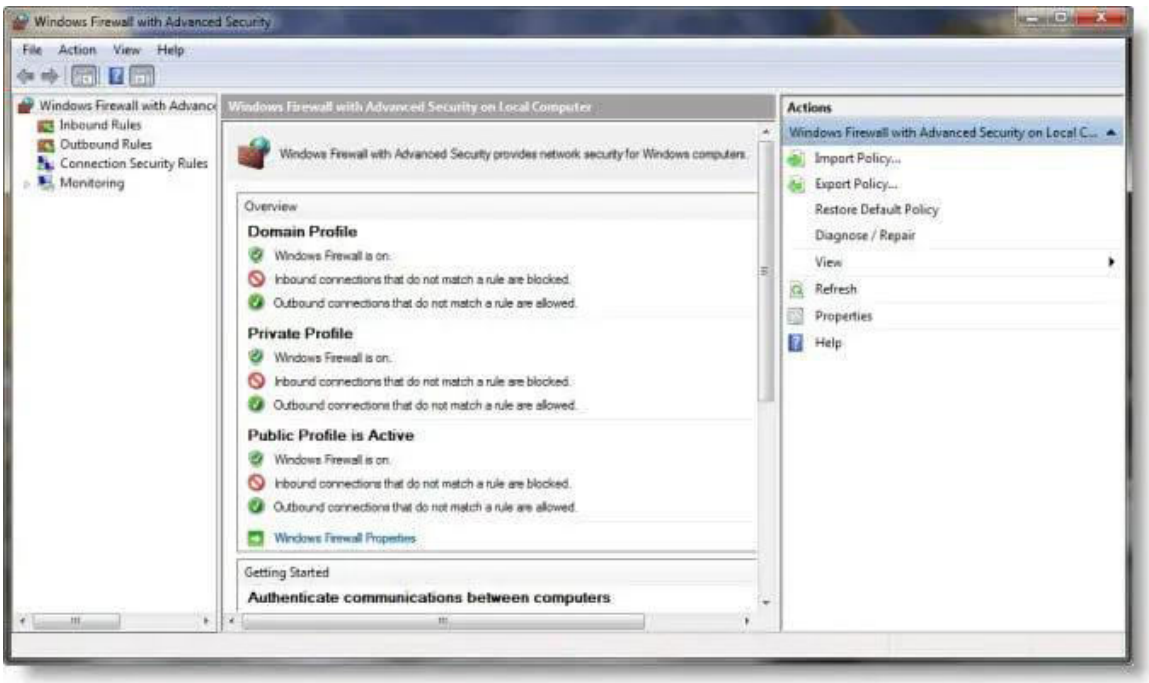
If you want to allow a program to communicate through the firewall, you can add it to the list of allowed programs. For example, you might not be able to send photos in an instant message until you add the instant messaging program to the list of allowed programs. To add or remove a program to the list, click on the Allow an app or feature through Windows Firewall link to open the following panel, where you will be able to get more details about allowed programs and allow another app to communicate through the firewall.

5. How to open a port in Windows Firewall

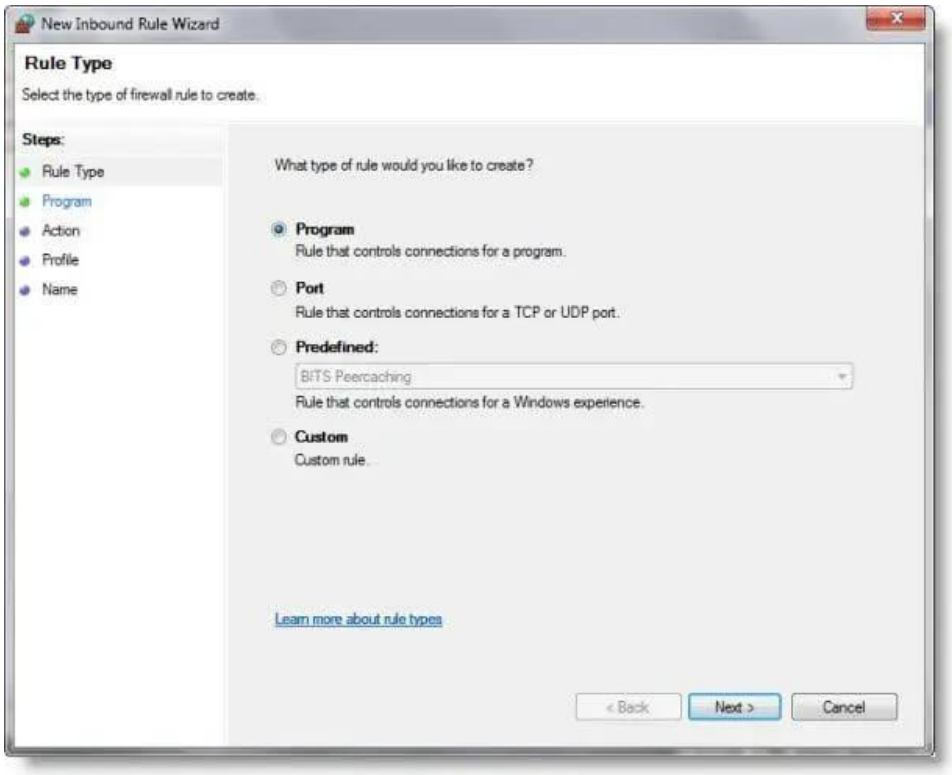
You can also block or open a Port in Windows Firewall. If Windows Firewall is blocking a program and you want to allow that program to communicate through the firewall, you can usually do that by selecting the program in the list of allowed programs (also called the exceptions list) in the Windows Firewall. To learn how to do this, see Allow a program to communicate through Windows Firewall.

However, if the program isn't listed, you might need to open a port. For example, to play a multiplayer game with friends online, you might need to open a port for the game so that the firewall allows the game information to reach your computer. A port stays open all the time, so be sure to close ports that you don't need any more.

Click to open Windows Firewall. In the left pane, click Advanced settings.



In the Windows Firewall with Advanced Security dialog box, in the left pane, click Inbound Rules, and then, in the right pane, click New Rule.



Follow the instructions on the screen to its logical conclusion.

Result:

Thus, the configuration of firewall has been studied successfully.

Ex.no.: 10(b) **STUDY TO CONFIGURE VPN**

Date:

Aim:

To study about the configuration of VPN.

Procedure:

Virtual private network:

A VPN, which stands for virtual private network, establishes a digital connection between your computer and a remote server owned by a VPN provider, creating a point-to-point tunnel that encrypts your personal data, masks your IP address, and lets you sidestep website blocks and firewalls on the internet.

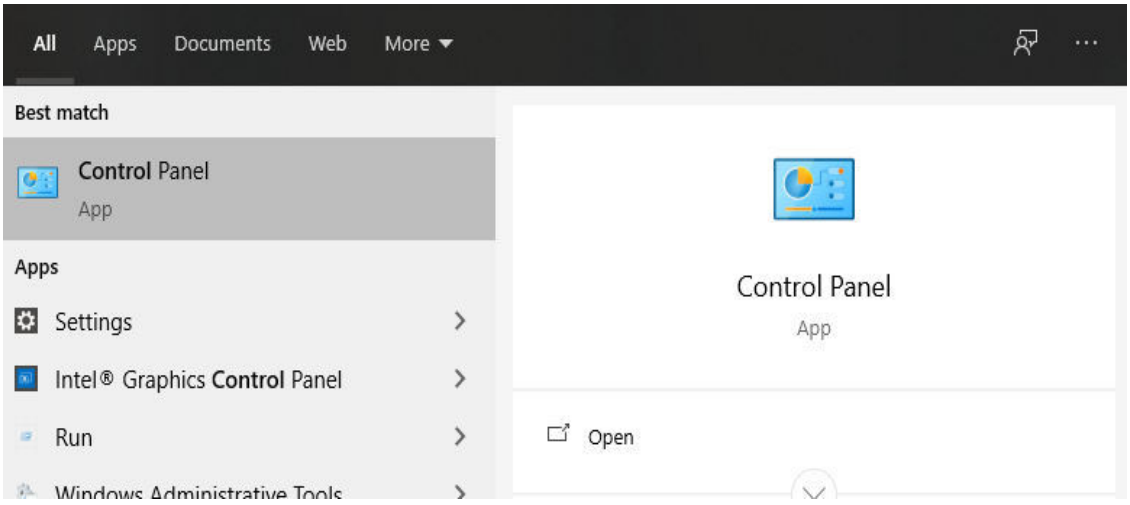
Set up a VPN on Windows:

Using the windows built-in client might pose an additional security risk. The third-party clients natively support OpenVPN and Wireguard. These won't work out of the box with a built-in client. Windows built-in client only supports IKEv2, L2TP, PPTP, and SSTP, so it's much easier (and safer) to install the app.

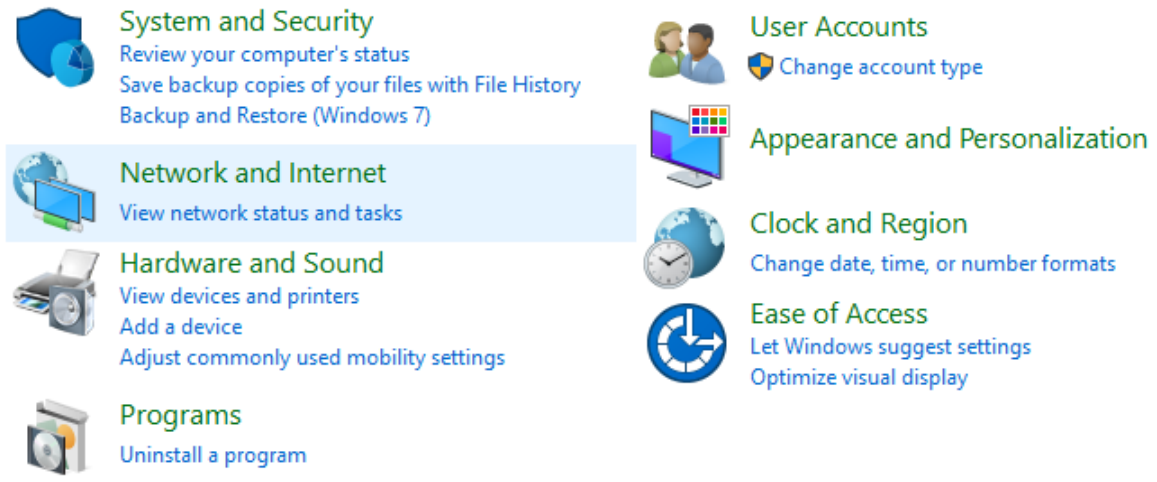
The steps are:

1. You'll have to find an outside server to use for your connection. You can set it up yourself, or you can use a third-party VPN service provider.

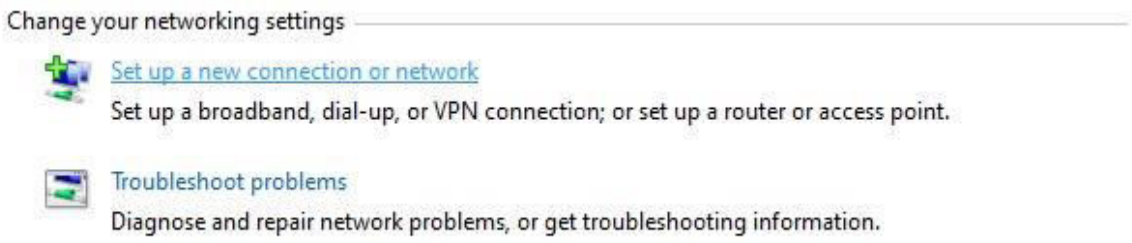
2. Click on the Windows taskbar, type in Control panel, and open it.



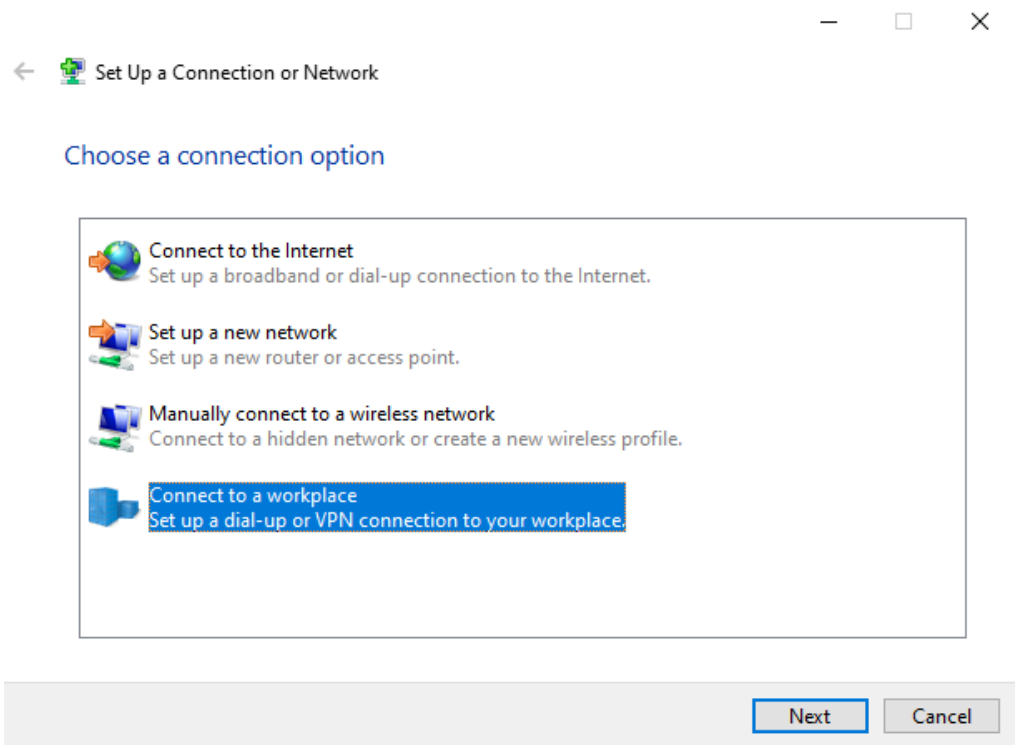
3. Click Network and Internet, then Network and Sharing Center.



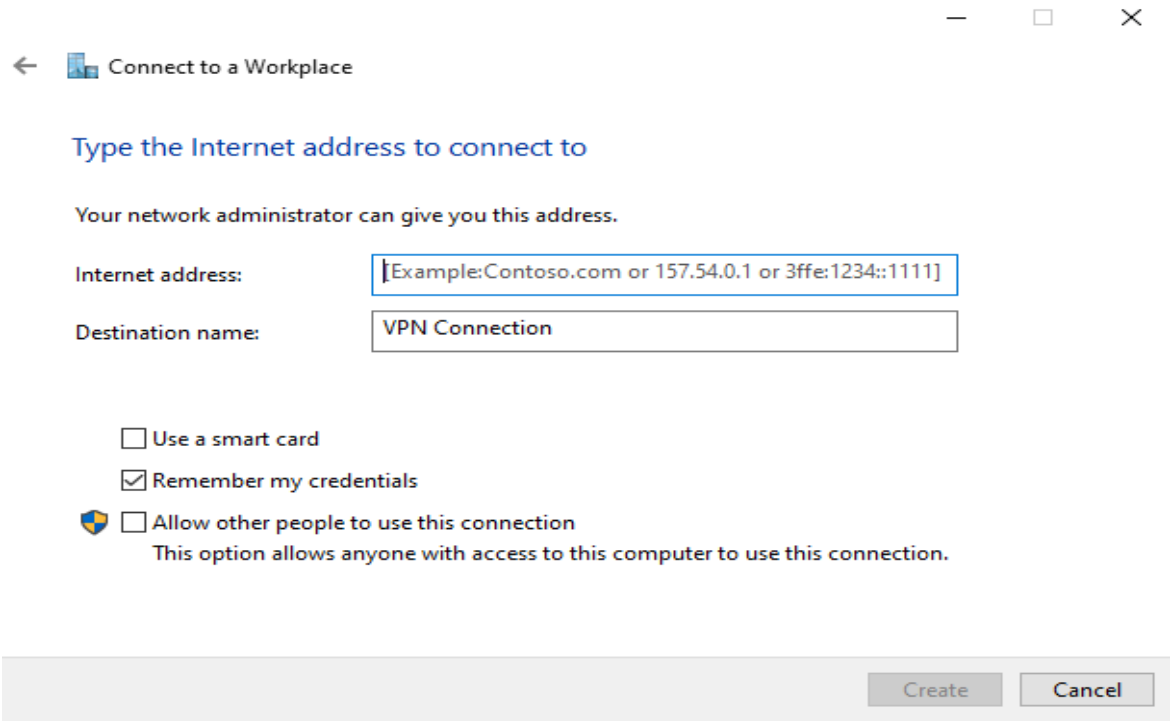
4. Under change your network settings section, click Set up a new connection or network.



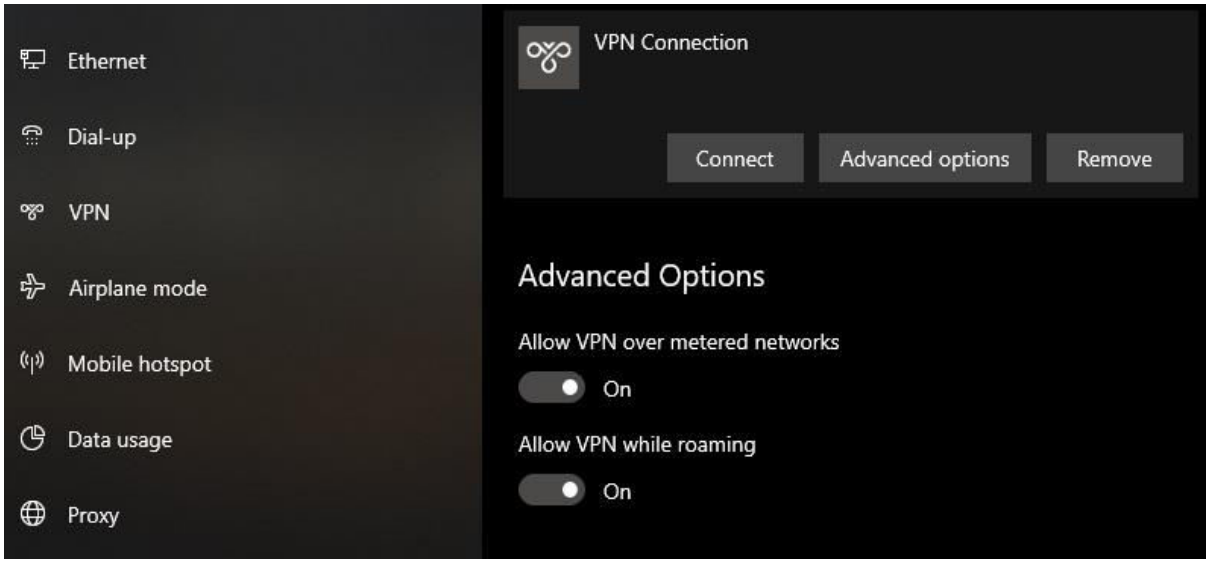
5. From the list, select Connect to a workplace, then Use my Internet connection (VPN).



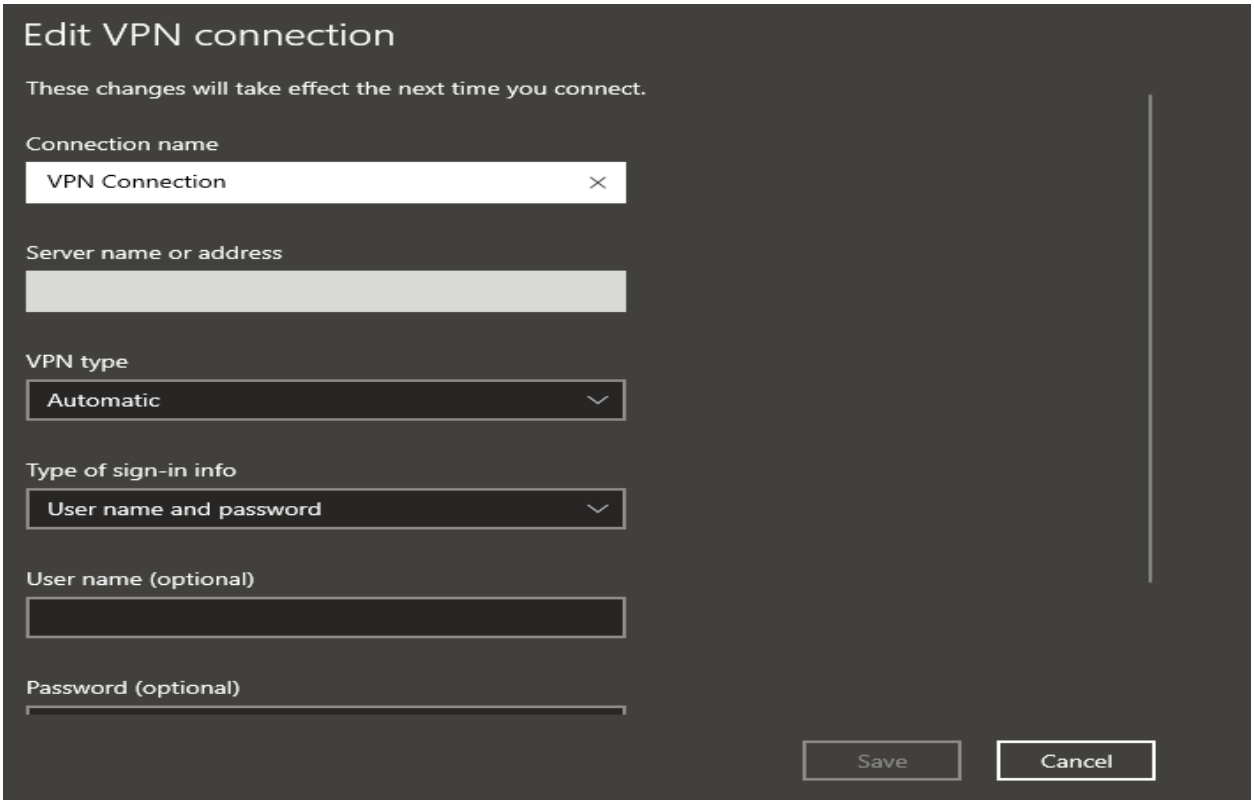
6. Enter your credentials. When you're finished, click Create. Mind that specific data encryption measures will significantly depend on your setup.



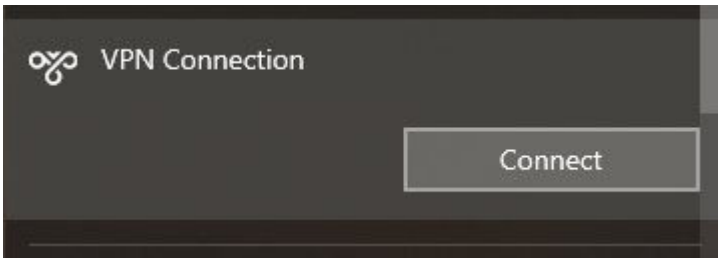
7. If you want to access your VPN, you'll have to open your Network settings. Then, go to a VPN tab, and select Advanced options.



8. In this newly opened window, fill in your credentials and click Save.



9. Now, click on your wifi connectivity icon, select your connection from the list, and click Connect.



Result:

Thus, the configuration of VPN has been studied successfully.