**Information security (CS403)**

**Regulation – R16**

**Lab manual for the Academic Year**

**(2019-20)**

**IV B.Tech. I Semester**

**DEPARTMENT OF**

**COMPUTER SCIENCE & ENGINEERING**



**Vadlamudi, 522 213, AP, India**

**Name of the Faculty: MRS.V.ANUSHA,ASST.PROFESOR, Dept. of CSE**

**PREFACE**

**Pre-requisite:** Any Programming Language

**About the course / Lab:**

This lab focuses towards the introduction of network security using various cryptographic algorithms and understanding network security applications. It also focuses on the practical applications that have been implemented and are in use to provide email and web security.

.

**Relevance to industry:**

The students will have knowledge and understanding of security aspects such as Classical encryption techniques, Block ciphers and the Data Encryption Standard, Basics of finite fields, Advanced Encryption Standard, Contemporary symmetric ciphers, Confidentiality using symmetric encryption, Basics of number theory, Key management, Public key cryptosystems, Message authentication, Hash functions and algorithms, Digital signatures and authentication protocols, Network security practice, Applications, E-Mail, IP and web security, System security, Intruders, Malicious software, Firewalls and develop their skills in: the programming of symmetric and/or asymmetric ciphers ,their use in the networks and knows protocols used in Web Security and Transport layer Securitywhich help the them to become security professionals .

**Latest Technologies:**

* IOT
* CLOUD
* DEEP LEARNING

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Description** | **Page No.** |
| **1** | **Syllabus** | **4** |
| **2** | **Course objectives and outcomes** | **5** |
| **3** | **List of experiments with codes** | **6 - 53** |
| **4** | **Viva voce questions** | **54 - 56** |

**Syllabus**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.no** | **Lab. No** | **Topic** | **Page No.** |
| **1** | **Lab-1** | Implement ceaser, Playfair and rail fence ciphers. |  |
| **2** | **LAB-2** | Develop simplified data encryption standard algorithm (S-DES) |  |
| **3** | **LAB-3** | Write a program to implement RSA algorithm. |  |
| **4** | **LAB-4** | Demonstrate the usage of wireshark to identify abnormal activity in network communication. |  |
| **5** | **LAB-5** | Demonstrate usage of NMAP (Zenmap) tool in network scanning. |  |
| **6** | **LAB-6** | . Demo of eavesdropping attack and its prevention using SSH. |  |
| **7** | **LAB-7** | Configuration and deployment of firewall. |  |

**COURSE OBJECTIVES**

1. To understand basics of Cryptography and Network Security.

2. To be able to secure a message over insecure channel by various means.

3. To learn about how to maintain the Confidentiality, Integrity and Availability of a data.

4. To understand various protocols for network security to protect against the threats in the networks.

**OUTCOMES:**

After successful completion of the course, the learners would be able to

1. Provide security of the data over the network.

2. Do research in the emerging areas of cryptography and network security.

3. Implement various networking protocols.

4. Protect any network from the threats in the world.

**LAB-1:**

**AIM:** Write a program for Ceaser cipher ,Playfair& Railfence encryption and decryption using files

**Objective:** To provide confidentiality to the message and to protect the message against Release of message contents attacks

**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS

Packages: Turbo/Borland/GNU - C/C++/Java

**Procedure:**

Algorithm Encryption:

1. Open a file which contains the plain text in read mode

2. Create a new file to which the cipher to be written.

3. Read one by one character of file-1 and call encrypt function write the cipher character in file2.

4. Close the files.

Algorithm Decryption:

1. Open a file which contains the cipher text in read mode

2. Read one by one character of file and call decrypt function

3. Close the file.

Note: Use an integer digit from 1-26 key. The same key is used for Encryption and Decryption.

Encrypt function:

Read the key

if character is between A to Z .

code = character + key;

cipher\_character = to\_char(code);

Decryption function:

if character is between A to Z .

code= character – key;

code=code+26;

original\_character = to\_char(code);

**Example:**

Key = 3

Replace ABCDEFGHI………………………………..WXYZ with

CDEFGHI………………………………..WXYZAB respectively.

In Encryption replace A with C , B with D and so on.

If we consider plain text : Vignan

The cipher is :Ykiqcq

**Implementation:**

//Ceaser Cipher

#include<stdio.h>

#include<conio.h>

FILE \*source,\*dest;

void encrypt();

void decrypt();

void main()

{

int choice;

int k;

printf("Enter Key from 1-26");

scanf("%d",&k);

do

{

clrscr();

printf("\n\n\t\tCeaser Cipher\n\nEnter your chice:\n");

printf("1.Encryption\n2.Decryption\n3.Exit.\n\nYour Choice:");

scanf("%d",&choice);

switch(choice)

{

case 1:encrypt(k);

break;

case 2:decrypt(k);

break;

default:exit(0);

}

getch();

}while(choice);

}

void encrypt(int k)

{

char fname[15],ch;

int n;

printf("\n\nEnter the name of file to be encrypted:\n");

flushall();

gets(fname);

flushall();

source = fopen(fname,"r");

dest = fopen("Dest.txt","w");

while ((ch=getc(source))!=EOF)

{

n=((toupper(ch)-65)+k)% 26;

ch=(char)(n+65);

putc(ch,dest);

}

fclose(dest);

fclose(source);

printf("\n\nThe file has been encrypted...\n\nThe contents are:\n");

dest=fopen("Dest.txt","r");

while((ch=getc(dest))!=EOF)

printf("%c",ch);

}

void decrypt(int k)

{

char fname[15],ch;

int i,n;

printf("\n\nEnter the name of file to be decrypted:\n");

flushall();

gets(fname);

flushall();

dest = fopen(fname,"r");

printf("\nDecrypted contents are : ");

while ((ch=getc(dest))!=EOF)

{

n=(toupper(ch)-65)-k;

if(n<0)

{

n=n+26;

}

n=n%26;

ch=(char)(n+65);

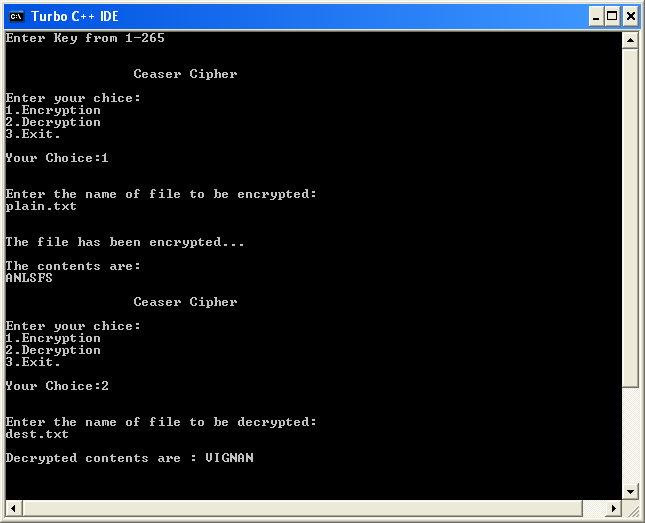
putchar(ch);

}

fclose(source);

}

**Sample Output:**



**LAB-1b**

**AIM:** Implementation of Play Fair Cipher Encryption

**Objective:** To provide confidentiality to the message and protect the message against attacks

**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS

Packages: Turbo/Borland/GNU - C/C++

**Analyzing the Problem:**

By analyzing the problem it is found that two basic steps are required for implementing the data

encryption using Play Fair cipher

1) Generate Key matrix

2) Encrypt the data using encryption rule and key matrix

Step1: Generating Key matrix

To Generate the key matrix take any random key of any length and form a 5X5matrix. Go on filling the rows of the matrix with the key characters (if repeatingcharacter occurs then ignore it). Fill the remaining matrix with alphabets from A to Z(except those already occurred in the key). For example for the key “monarchy” we have the matrix as follow

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| M | O | N | A | R |
| C | H | Y | B | D |
| E | F | G | I/J | K |
| L | P | Q | S | T |
| U | V | W | Y | Z |

Step 2: Encrypt the data using encryption rule and key matrix

To Encrypt the data take two characters at time from plain text file and encrypt itusing one of the following rules.

Encryption rules

1) Repeating plain text letters that would fall in the same pair are separated withfiller letter,such as x.( i.e. Balloon becomes Ba, lx, lo, on)

2) If both the characters are in the same raw then replace each with the character toits right, with

the last character followed by the first, in the matrix.

3) If both the characters are in the same column then replace each with the characterbelow it, withthe bottom character followed by the top, in the matrix.

4) Otherwise each plain text letter is replaced by the letter that lies in its own rowand the column occupied by the other plain text letter

Example:Using key as “monarchy” we have

- Encryption of AR as RM

**-** Encryption of MU as CM

- Encryption of BP as IM

**Designing the Solution:**

For this solution we have to implement the following functions given below.

1) Input function for key & Plain Text.

2) Matrix generation.

3) Encryption function for generating Cipher Text.

4) Print function for printing Cipher Text Output.

**Implementation:**

/\*PLAY FAIR CIPHER\*/

#include <stdio.h>

#define siz 5

void playfair(char ch1,char ch2, char mat[siz][siz])

{

int j,m,n,p,q,c,k;

for(j=0,c=0;(c<2)||(j<siz);j++)

for(k=0;k<siz;k++)

if(mat[j][k] == ch1)

m=j,n=k,c++;

else if(mat[j][k] == ch2)

p=j,q=k,c++;

if(m==p)

{

n++; q++;

if(n==siz) n=0;

if(q==siz) q=0;

printf("%c%c",mat[m][n],mat[p][q]);

}

else if(n==q)

{

m++; p++;

if(m==siz) m=0;

if(q==siz) p=0;

printf("%c%c",mat[m][n],mat[p][q]);

}

else

{

printf("%c%c",mat[m][q],mat[p][n]);

}

}

void main()

{

char mat[siz][siz],key[20],str[25]={0};

int m,n,i,j;

char temp;

printf("Enter Key String:");

gets(key);

printf("Enter Plain text");

gets(str);

m=n=0;

for(i=0;key[i]!='\0';i++)

{

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

if(key[j] == key[i]) break;

if(key[i]=='j') key[i]='i';

if(j>=i)

{

mat[m][n++] = key[i];

if(n==siz)

n=0,m++;

}

}

for(i=97;i<=122;i++)

{

for(j=0;key[j]!='\0';j++)

if(key[j] == i)

break;

else if(i=='j')

break;

if(key[j]=='\0')

{

mat[m][n++] = i;

if(n==siz) n=0,m++;

}

}

printf("\n\nMatrix :\n");

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

{

for(j=0;j<siz;j++)

printf("%c\t",mat[i][j]);

printf("\n");

}

printf("\n\nEntered text :%s\nCipher Text :",str);

for(i=0;str[i]!='\0';i++)

{

temp = str[i++];

if(temp == 'j') temp='i';

if(str[i]=='\0')

playfair(temp,'x',mat);

else

{

if(str[i]=='j') str[i]='i';

if(temp == str[i])

{

playfair(temp,'x',mat);

i--;

}

else

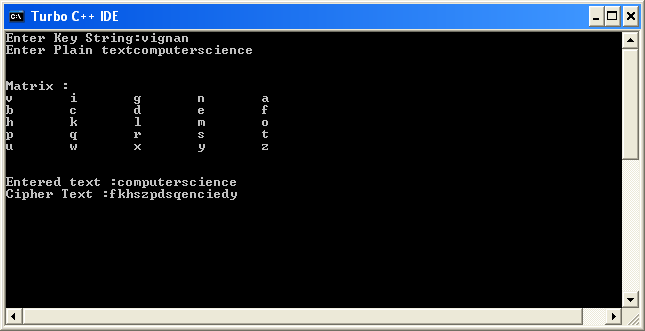
playfair(temp,str[i],mat);

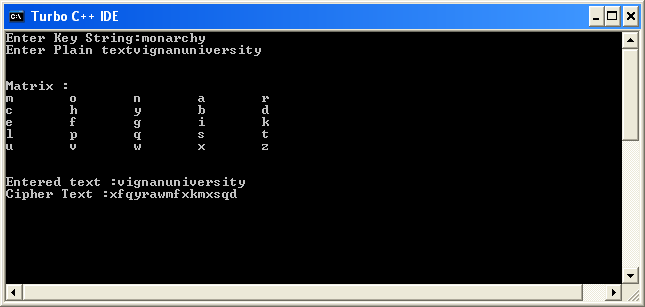
}

}

}

**Sample Output:**

****



**LAB-1c:**

**AIM:** Implementation of encryption and decryption using railfence cipher

**Objective:** To provide confidentiality to the message and to protect the message against Release of message contents attacks

**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS

Packages: Turbo/Borland/GNU - C/C++

**Procedure:**

Rail Fence cipher is a Transposition cipher. Encryption is the result by changing the position of the message. In this particular scheme the message is written in two rows. That is the first character is written in the first row, second character is written in the second row and so on. To get the cipher read the message off, row by row, first row followed by second row.

Algorithm Encryption:

1. Read plain text

2. Consider CT as a temporary string to which cipher is copied

3. Copy all the even indexed letters of the plain text to CT

4. Copy all the odd indexed letters of the plain text to CT

5. CT contains the cipher

Algorithm Decryption:

1. Read cipher text, CT

2. Consider PT as a temporary string to which plain text is copied

3. k=strlen(CT)/2

4. i=0,j=0;

5. PT[i]=CT[i]

6. PT[i+1]=CT[k]

7. i++,j++,k++

8. Repeat steps 5,6,7 till the end of the char is reached in CT

9. PT contains the plain text derived based on cipher

**Example:**

If we consider plain text : VIGNANUNIVERSITY

Intermediate text representation is:

V G A U I E S T

I N N N V R I Y

The cipher is : V G A U I E S T I N N N V R I Y

The plain text : VIGNANUNIVERSITY

**Implementation:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

int i,j,k,l;

char a[20],c[20],d[20];

clrscr();

printf("\nEnter the input string : ");

scanf("%s",&a);

l=strlen(a);

for(i=0,j=0;i<l;i++)

{

if(i%2==0)

{

c[j]=a[i];

printf("%c ",c[j]);

j++;

}

}

printf("\n");

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

{

if(i%2==1)

{

c[j]=a[i];

printf("%c ",c[j]);

j++;

}

}

c[j]='\0';

printf("\nCipher text after applying rail fence :");

printf("\n%s",c);

/\*Deciphering\*/

if(l%2==0)

k=l/2;

else

k=(l/2)+1;

for(i=0,j=0;i<k;i++)

{

d[j]=c[i];

j=j+2;

}

for(i=k,j=1;i<l;i++)

{

d[j]=c[i];

j=j+2;

}

d[l]='\0';

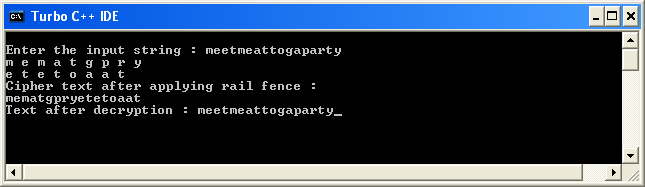
printf("\nText after decryption : ");

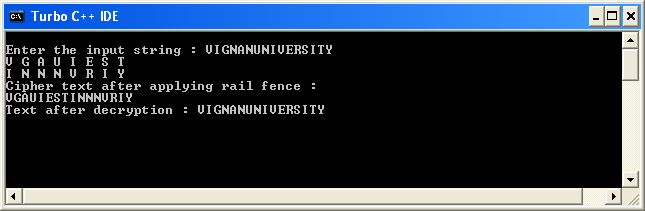
printf("%s",d);

getch();

}

**Sample Outputs:**

****



**LAB-2:**

**AIM:** Implementation of encryption and decryption using S-DES algorithm.

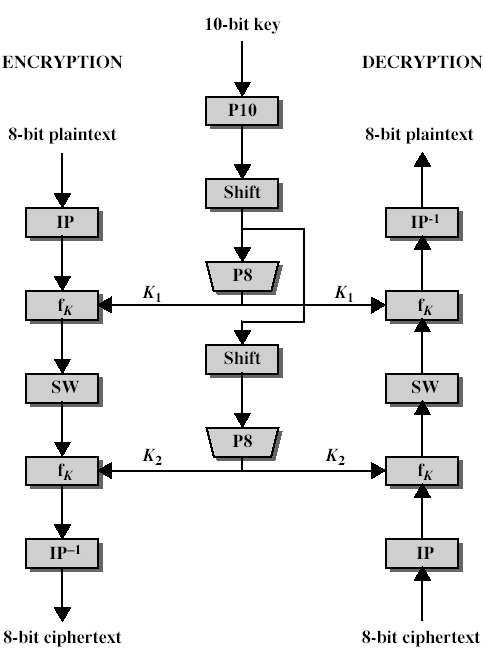
**Objective:** To provide confidentiality to the message and to protect the message against Release of message contents attacks. S-DES is a Symmetric key encryption algorithm.

**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS

Packages: Turbo/Borland/GNU - C/C++

**Procedure:**

S-DES is a simplified version of DES. S-DES algorithm is used for the academic purpose. S-DES uses bit wise operation on message letters to encrypt the data so it is more powerful against the cryptanalysis attacks. This algorithm takes 8-bitof the message as input, also takes 10 bit key and produces 8 bit cipher text. This algorithm has two rounds. It generates 2, 8-bit keys that are to be used in each round. Following figure shows the functional details of S-DES.



**Design:**

Algorithm to generate key:

As there are two rounds we have to generate two keys from the given 10-bit key

1: Apply permutation function P10 on 10 bit key

2: Divide the result into two parts each containing 5-bit, call them L0 and L1

3: Apply one bit Circular Left Shift on both L0 and L1

4: L0 and L1 together will form out 10-bit number

5: Apply permutation function P8 on result to select 8 out of 10 bits for key K1 (for the

first round)

6: Again apply two bit Circular Left Shift to L0 and L1

7: Combine the result, which will form out 10-bit number

8: Apply permutation function P8 on result to select 8 out of 10 bits for key K2 (for the

second round)

Algorithm for Encryption:

1: Get 8 bit message text (M) apply Initial permutation function (IP)

2: Divide IP(M) into nibbles L0 and R0

3: Apply function Fk on L0

4: XOR the result with R0 ( That is R0 (+) Fk(L0))

5: Swap the result with RO

6: Repeat the step 1 to 5for the second round

7:Apply (IP-1) on the result to get the encrypted data

Algorithm for function Fk:

1: Give the 4-bit input to EP (Expansion function) the result will be a 8-bit expanded data

2: XOR the 8-bit expanded data with 8-bit key (K1 for the first round and K2 for the

second round)

2: Divide result into upper (P1) and lower (P2) nibble

3: Apply compression function S0 to P0 and S1 to P1, which will compress the 4-bit input

to 2-bit output

4: Combine 2-bit output from S0 and S1 to form a 4-bit digit

5: Apply permutation function P4 to 4-bit result

Functions used in S-DES:

P10 = 3 5 2 7 4 10 1 9 8 6

P8 = 6 3 7 4 8 5 10 9

P4 = 2 4 3 1

IP = 2 6 3 1 4 8 5 7

IP-1 = 4 1 3 5 7 2 8 6

EP = 4 1 2 3 2 3 4 1

S0:

1 0 3 2

3 2 1 0

0 2 1 3

3 1 3 2

S1:

0 1 2 3

2 0 1 3

3 0 1 0

2 1 0 3

**Example:**

Plain text: 10001011

Key: 0000011011

Key1:11100100

Key2:01011100

Cipher Text:11110001

Plain Text: 10001011

**Implementation:**

#include <stdio.h>

int l[4],r[4],keys[2][8],ct[8];

void sbox(int sip[],int p[],int sbno,int i)

{

int sbox[2][4]`[4]={1,0,3,2,3,2,1,0,0,2,1,3,3,1,3,2,0,1,2,3,2,0,1,3,3,0,1,0,2,1,0,3};

int rw,c,sop;

rw = sip[3]+sip[0]\*2;

c = sip[2]+sip[1]\*2;

sop = sbox[sbno][rw][c]; //sop gives decimal value of S-Box Output

for(;sop!=0;sop/=2)

p[i--]=sop%2;

}

void cmp\_fun(int round)

{

int EP[]={4,1,2,3,2,3,4,1},i,epd[8];

int slip[4],srip[4];

int p[4]={0},p4[]={2,4,3,1},np[4];

for(i=0;i<8;i++) // E/P Permutation

epd[i]=r[EP[i]-1];

for(i=0;i<8;i++)//Performing XOR with Key

if(i<4)

slip[i] = epd[i]^keys[round][i]; // Using Key \_ 1=>0

else

srip[i-4] = epd[i]^keys[round][i];

sbox(slip,p,0,1);//Calling SBox 1, 0->SBOX 1

sbox(srip,p,1,3);//Calling SBox 1, 1->SBOX 2

for(i=0;i<4;i++) //P4 permutation

np[i]=p[p4[i]-1];

for(i=0;i<4;i++)

l[i] = l[i]^np[i];

}

void left\_shift(int keyip[],int nob)

{

int t1,t2,i;

while(nob>0)

{

t1=keyip[0],t2=keyip[5];

for(i=0;i<9;i++)

if(i<4)

keyip[i] =keyip[i+1];

else if(i>4)

keyip[i] = keyip[i+1];

keyip[4]=t1,keyip[9]=t2;

nob--;

}

}

void gen\_keys()

{

int key[10],i,keyip[10];

int p10[]={3,5,2,7,4,10,1,9,8,6},p8[]={6,3,7,4,8,5,10,9};

printf("Enter Key :");

for(i=0;i<10;i++)

scanf("%d", &key[i]);

for(i=0;i<10;i++) // Permutation P10

keyip[i] = key[p10[i]-1];

left\_shift(keyip,1); // Left Shifting (Array,No of bts)

printf("\nKey1 :");

for(i=0;i<8;i++){ //Permuting P8 on key1

keys[0][i] = keyip[p8[i]-1];// Key1 Generated!!

printf("%d",keys[0][i]);

}

left\_shift(keyip,2);// Generating Key2 . .

printf("\nKey2 :");

for(i=0;i<8;i++){

keys[1][i] = keyip[p8[i]-1];// Key2 Generated!!

printf("%d",keys[1][i]);

}

}

void En\_De(int pt[],int c)

{

int ip[]={2,6,3,1,4,8,5,7},ipi[]={4,1,3,5,7,2,8,6},t[8],i;

for(i=0;i<8;i++)// Performing Permutation on input bits!!

if(i<4)

l[i]=pt[ip[i]-1];

else

r[i-4] = pt[ip[i]-1];

cmp\_fun(c);//Round 0+1 using key 0+1

for(i=0;i<4;i++) //Swapping left & right

r[i]=l[i]+r[i],l[i]=r[i]-l[i],r[i]=r[i]-l[i];

printf("\n\n");

cmp\_fun(!c); // Round 1+1 wid key1+1 wid swapped bits

for(i=0;i<8;i++)

if(i<4) t[i]=l[i];

else t[i]=r[i-4];

for(i=0;i<8;i++)

ct[i] = t[ipi[i]-1];

}

void main()

{

int pt[8]={0},i;

printf("Enter plain text binary bits:");

for(i=0;i<8;i++)

scanf("%d",&pt[i]);

gen\_keys(); // Generating Keys key1 & key2

En\_De(pt,0);

printf("\nCipher Text :");

for(i=0;i<8;i++)

printf("%d",ct[i]);

En\_De(ct,1);

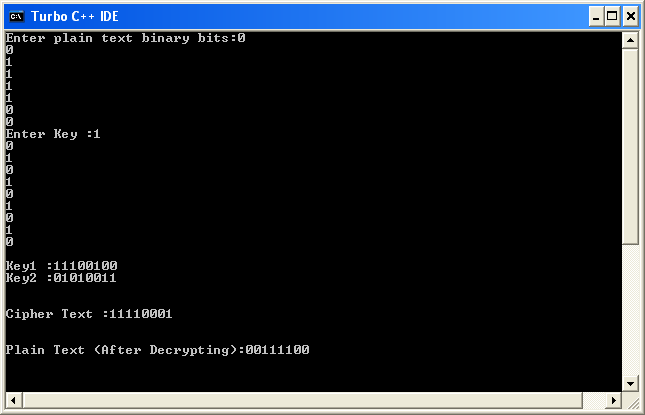
printf("\nPlain Text (After Decrypting):");

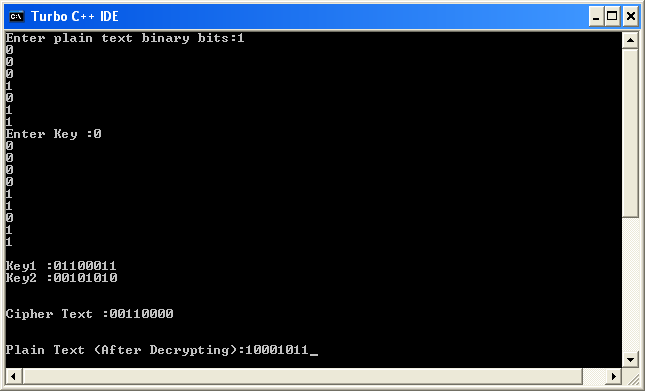
for(i=0;i<8;i++)

printf("%d",ct[i]);

}

**Sample outputs:**

****

****

**LAB-3:**

**AIM:** Implement RSA asymmetric (public key and private key)-

EncryptionImplement Euclidean and Extended Euclidean algorithm for calculating the GCD

.

**Objective:** To provide confidentiality to the message and to protect the message against Release of message contents attacks. RSA is a public key encryption algorithm.

**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS

Packages: Turbo/Borland/GNU - C/C++

**Procedure:**

ItwasdevelopedbyRivest, ShamirandAdleman.Thisalgorithm makes use ofanexpressionwithexponentials.Plaintextisencryptedin blocks. Witheachblockhavinga binaryvalue lessthansomenumbern. For some plaintextblockM andciphertextblockC:

Encryption : C =Memodn

Decryption : M =Cdmodn

Publickey ofKU={e,n}anda privatekeyofKR={d,n}.

**Algorithm:**

Key generation:

Step1: Selecttwoprime numbers,p,q.

Step2: Calculaten= p\*q

Step3: Calculate Ф(n) =(p-1)(q-1)

Step4: SelectesuchthateisrelativelyprimetoФ(n), gcd(e, Ф(n))=1 andlessthanФ(n);

Step5: Determinedsuchthatd=e-1modФ(n)

Encryption:

Step1: Read Plain Text, M

Step 2: Find C= M\*M mod n

Step 3: Repeat Step 2 for e times

Step 4: C contains cipher

Decryption:

Step1: Read Cipher Text, C

Step 2: Find M= C\*C mod n

Step 3: Repeat Step 2 for d times

Step 4: M contains plain text

**Example:**

Selecttwoprime numbers,p=17andq =11.

Calculaten= p\*q= 17\*11= 187

Calculate Ф(n) =(p-1)(q-1) = 16\*10=160.

SelectesuchthateisrelativelyprimetoФ(n)=160andlessthanФ(n);choose e =7.

Determinedsuchthated≡1modФ(n)andd<160.thecorrectvalueisd= 23,because23\*7=161=1mod160.

Consider plaintext: VIGNAN

V=22

ENCRYPTION : C =Memodn

C= 227 mod 187 = 14

Decryption : M =Cdmodn

M=1423 mod 187 = 22

REMAINING LETTERS ARE LEFT TO THE STUDENTS AS AN EXCERCISE

**Implementation:**

/\* C program for the Implementation Of RSA Algorithm \*/

#include<stdio.h>

#include<conio.h>

int phi,M,n,e,d,C,FLAG;

gcd(int a,int b)

{

int temp = 0;

while(b != 0)

{

temp = a;

a = b;

b = temp % b;

}

return a;

}

int isprime(int n)

{

int i,count=1;

for(i=2;i<n;i++)

{

if(n%i==0)

count=0;

break;

}

return count;

}

int check()

{

int i;

if((e<=1)||(e>=phi))

{

FLAG = 1;

return;

}

if(gcd(e,phi)!=1)

{

FLAG = 1;

return;

}

if(isprime(e)==0)

{

FLAG = 1;

return;

}

FLAG = 0;

return;

}

void encrypt()

{

int i;

C = 1;

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

C=C\*M%n;

C = C%n;

printf("\n\tEncrypted keyword : %d",C);

}

void decrypt()

{

int i;

M = 1;

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

M=M\*C%n;

M = M%n;

printf("\n\tDecrypted keyword : %d",M);

}

void main()

{

int i,p,q,s;

clrscr();

printf("Enter Two Relatively Prime Numbers\t: ");

scanf("%d%d",&p,&q);

n = p\*q;

phi=(p-1)\*(q-1);

printf("\n\tF(n)\t= %d",phi);

do

{

printf("\n\nEnter e value\t: ");

scanf("%d",&e);

check();

}while(FLAG==1);

printf("GCD",gcd(e,phi));

d = 1;

for(i=1;i<phi;i++)

{

if((e\*i)%phi==1)

{

d=i;

break;

}

}

printf("\n\tPublic Key\t: {%d,%d}",e,n);

printf("\n\tPrivate Key\t: {%d,%d}",d,n);

printf("\n\nEnter The Plain Text\t: ");

scanf("%d",&M);

encrypt();

//printf("\n\nEnter the Cipher text\t: ");

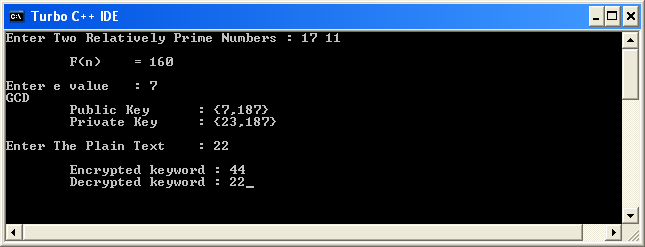
//scanf("%d",&C);

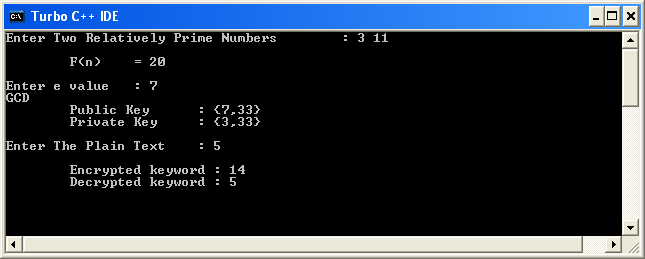
decrypt();

getch();

}

**Sample outputs:**





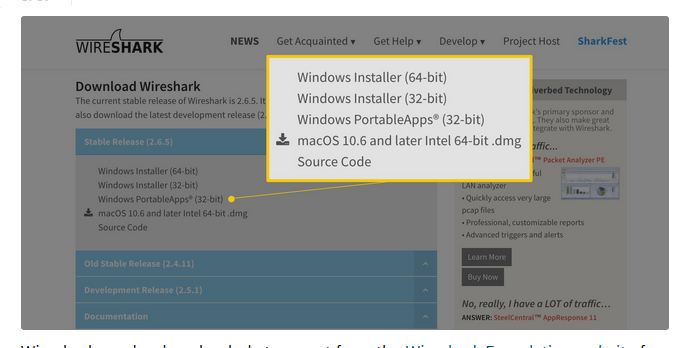
**LAB-4**

**AIM:**. Demonstrate the usage of Wireshark to identify abnormal activity in network

Communication

Wireshark is a free application you use to capture and view the data traveling back and forth on your network. It provides the ability to drill down and read the contents of each packet and is filtered to meet your specific needs. It is commonly used to troubleshoot network problems and to develop and test software. This [open-source](https://www.lifewire.com/what-is-open-source-software-4147547) protocol analyzer is widely accepted as the industry standard, winning its fair share of awards over the years.

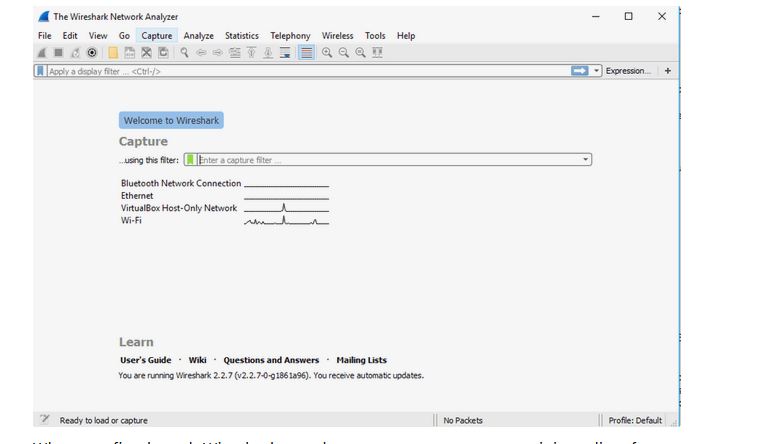
Originally known as Ethereal, Wireshark has a user-friendly interface that can display data from hundreds of different [protocols](https://www.lifewire.com/definition-of-protocol-network-817949) on all major network types. Data packets can be viewed in real time or analyzed offline. Wireshark supports dozens of capture/trace file formats supported including [CAP](https://www.lifewire.com/cap-file-2622694) and [ERF](https://www.lifewire.com/erf-file-2621096). Integrated decryption tools allow you to view encrypted packets for several popular protocols including [WEP](https://www.lifewire.com/definition-of-wired-equivalent-privacy-816575) and [WPA/WPA2](https://www.lifewire.com/definition-of-wifi-protected-access-816576).



### Downloading and Installing Wireshark

Wireshark can be downloaded at no cost from the [Wireshark Foundation website](https://wireshark.org/#download) for both macOS and Windows operating systems. Unless you are an advanced user, it is recommended that you only download the latest stable release. During the Windows setup process, you should choose to install WinPcap if prompted, as it includes a library required for live data capture.

The application is also available for Linux and most other UNIX-like platforms including [Red Hat](https://www.lifewire.com/best-linux-distributions-for-beginners-4174526), Solaris, and FreeBSD. The binaries required for these operating systems can be found toward the bottom of the download page in the [Third-Party Packages](https://www.wireshark.org/download.html#thirdparty) section. You can also download Wireshark's source code from this page.



### How to Capture Data Packets

When you first launch Wireshark, a welcome screen appears containing a list of available network connections on your current device. In this example, you'll notice that the following connection types are shown: Bluetooth Network Connection, Ethernet, VirtualBox Host-Only Network, and Wi-Fi. Displayed to the right of each is an EKG-style line graph that represents live traffic on that respective network.

To begin capturing packets, select one or more of the networks by clicking on your choice and using the **Shift** or **Ctrl** keys if you want to record data from multiple networks simultaneously. After a connection type is selected for capturing purposes, its background is shaded in either blue or gray. Click on **Capture** in the main menu located toward the top of the Wireshark interface. When the drop-down menu appears, select the **Start** option.

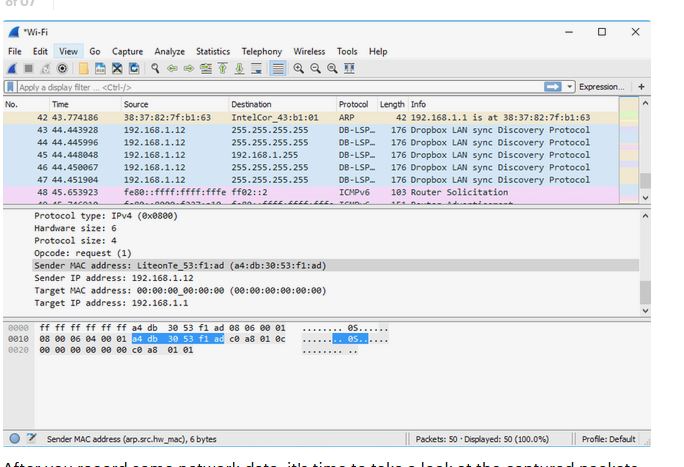
You can also initiate packet capturing via one of the following shortcuts.

* **Keyboard:** Press ​**Ctrl** + **E.**
* **Mouse:** To begin capturing packets from one particular network, double-click on its name.
* **Toolbar:** Click on the blue shark fin button located on the far left side of the Wireshark toolbar.

The live capture process begins, and Wireshark displays the packet details as they are recorded. To Stop capturing:

* **Keyboard:** Press **Ctrl** + **E**
* **Toolbar:** Click on the red **Stop** button located next to the shark fin on the Wireshark toolbar.

### Viewing and Analyzing Packet Contents



After you record some network data, it's time to take a look at the captured packets. The captured data interface contains three main sections: the packet list pane, the packet details pane, and the packet bytes pane.

### Packet List

The packet list pane, located at the top of the window, shows all packets found in the active capture file. Each packet has its own row and corresponding number assigned to it, along with each of these data points.

* **Time:** The timestamp of when the packet was captured is displayed in this column. The default format is the number of seconds or partial seconds since this specific capture file was first created. To modify this format to something that may be a bit more useful, such as the actual time of day, select the **Time Display Format**option from Wireshark's View menu located at the top of the main interface.
* **Source:** This column contains the address (IP or other) where the packet originated.
* **Destination:** This column contains the address that the packet is being sent to.
* **Protocol:** The packet's protocol name, such as TCP, can be found in this column.
* **Length:** The packet length, in bytes, is displayed in this column.
* **Info:** Additional details about the packet are presented here. The contents of this column can vary greatly depending on packet contents.

When a packet is selected in the top pane, you may notice one or more symbols appear in the first column. Open or closed brackets and a straight horizontal line indicate whether a packet or group of packets are all part of the same back-and-forth conversation on the network. A broken horizontal line signifies that a packet is not part of said conversation.

### Packet Details

The details pane, found in the middle, presents the protocols and protocol fields of the selected packet in a collapsible format. In addition to expanding each selection, you can apply individual Wireshark filters based on specific details and follow streams of data based on protocol type via the details context menu, which is accessible by right-clicking your mouse on the desired item in this pane.

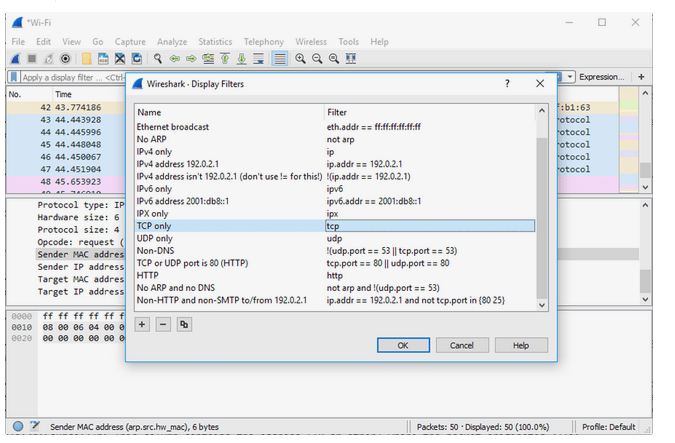
### Packet Bytes

At the bottom is the packet bytes pane, which displays the raw data of the selected packet in a hexadecimal view. This [hex dump](https://www.lifewire.com/xxd-linux-command-unix-command-4097149) contains 16 hexadecimal bytes and 16 ASCII bytes alongside the data offset.

Selecting a specific portion of this data automatically highlights its corresponding section in the packet details pane and vice versa. Any bytes that cannot be printed are instead represented by a period.

You can choose to show this data in bit format as opposed to hexadecimal by right-clicking anywhere within the pane and selecting the appropriate option from the context menu.

### Using Wireshark Filters



One of the most important feature sets in Wireshark is its filter capability, especially when you're dealing with files that are significant in size. Capture filters can be set before the fact, instructing Wireshark to only record those packets that meet your specified criteria.

Filters can also be applied to a capture file that has already been created so that only certain packets are shown. These are referred to as display filters.

Wireshark provides a large number of predefined filters by default, letting you narrow down the number of visible packets with just a few keystrokes or mouse clicks. To use one of these existing filters, place its name in the **Apply a display filter** entry field located directly below the Wireshark toolbar or in the **Enter a capture filter** entry field located in the center of the welcome screen.

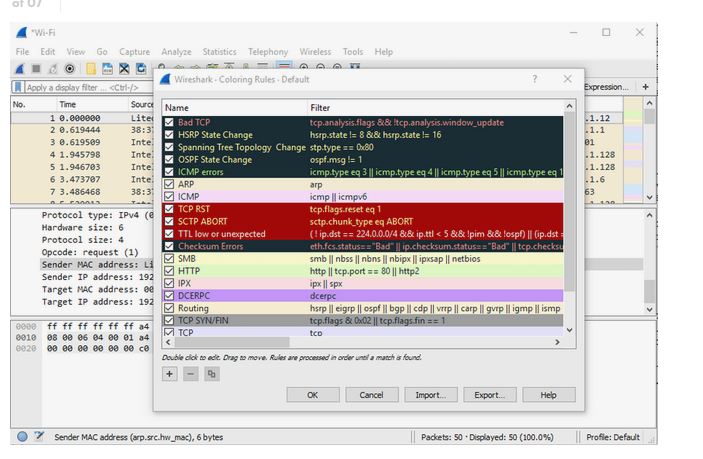
There are multiple ways to achieve this. If you already know the name of your filter, type it into the appropriate field. For example, if you only want to display TCP packets, you type **tcp**. Wireshark's autocompleting feature shows suggested names as you begin typing, making it easier to find the correct moniker for the filter you're seeking.

Another way to choose a filter is to click on the bookmark-like icon positioned on the left side of the entry field. This presents a menu containing some of the most commonly used filters as well as an option to **Manage Capture Filters** or **Manage Display Filters**. If you choose to manage either type, an interface appears allowing you to add, remove, or edit filters.

You can also access previously used filters by selecting the down arrow on the right side of the entry field to display a history drop-down list.

Once set, capture filters are applied as soon as you begin recording network traffic. To apply a display filter, you click on the right arrow button found on the far right side of the entry field.

### Color Rules

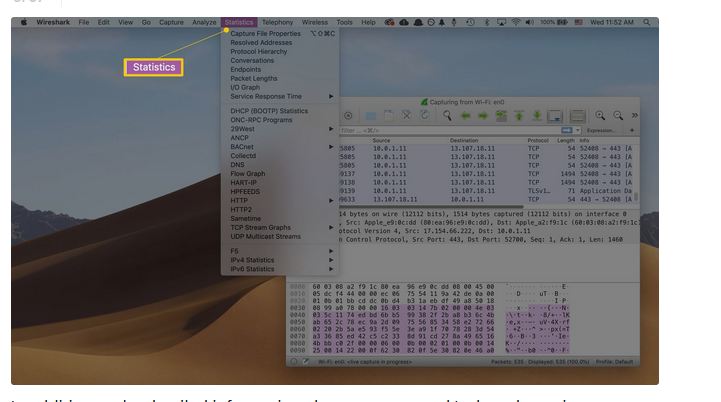


While Wireshark's capture and display filters allow you to limit which packets are recorded or shown on the screen, its colorization functionality takes things a step further by making it easy to distinguish between different packet types based on their individual hue. This handy feature lets you quickly locate certain packets within a saved set by their row color in the packet list pane.

Wireshark comes with about 20 default coloring rules built in, each of which can be edited, disabled, or deleted if you wish. You can also add new shade-based filters through the coloring-rules interface, accessible from the **View** menu. In addition to defining a name and filter criteria for each rule, you are also asked to associate both a background color and a text color.

Packet colorization can be toggled off and on via the **Colorize Packet List** option, also found in the **View** menu.

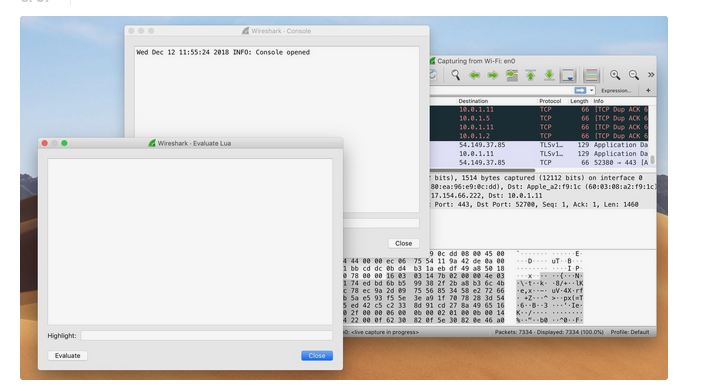
### Statistics



In addition to the detailed information about your network's data shown in Wireshark's main window, several other useful metrics are available via the **Statistics** drop-down menu found toward the top of the screen. These include size and timing information about the capture file itself, along with dozens of charts and graphs ranging in topic from packet conversation breakdowns to load distribution of HTTP requests.

Display filters can be applied to many of these statistics via their interfaces, and the results can be exported to several common file formats including [CSV](https://www.lifewire.com/csv-file-2622708), [XML](https://www.lifewire.com/what-is-an-xml-file-2622560), and TXT.

### Advanced Features

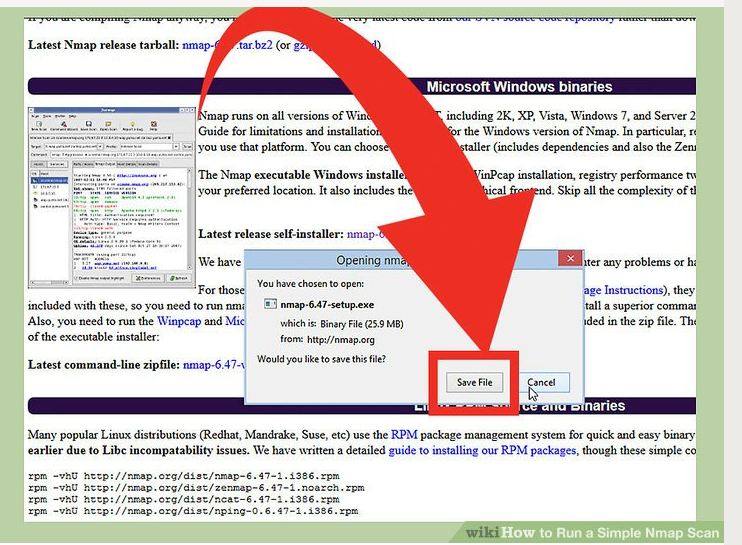


In addition to Wireshark's main functionality, there is also a collection of additional features available in this powerful tool typically reserved for advanced users. This includes the ability to write your own protocol dissectors in the Lua programming language.

**LAB-5**

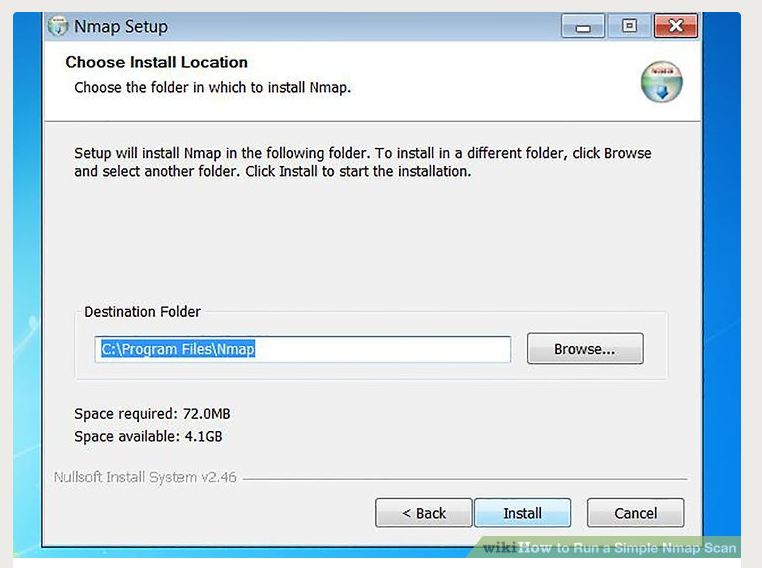
**AIM:**Demonstrate usage of NMAP (Zenmap) Tool in Network Scanning.

### Using Zenmap

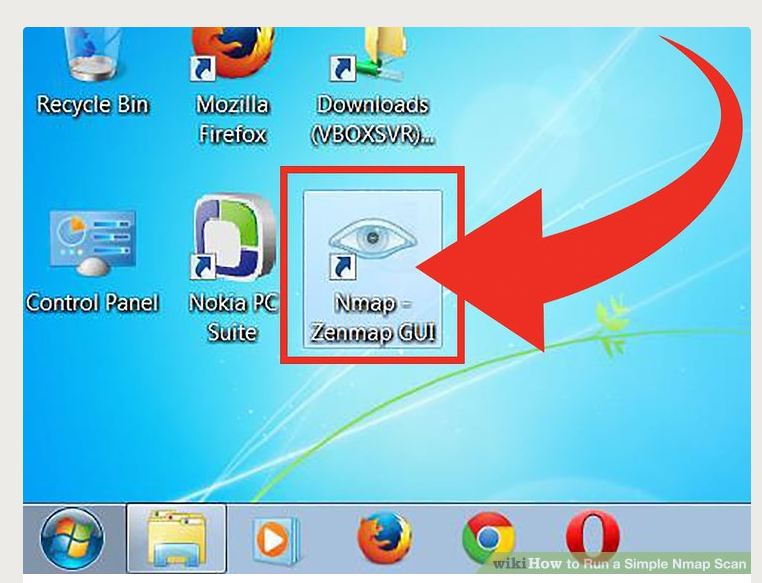
.

**1.Download the Nmap installer.** This can be found for free from the developer’s website. It is highly recommended that you download directly from the developer to avoid any potential viruses or fake files. Downloading the Nmap installer includes Zenmap, the graphical interface for Nmap which makes it easy for newcomers to perform scans without having to learn command lines.

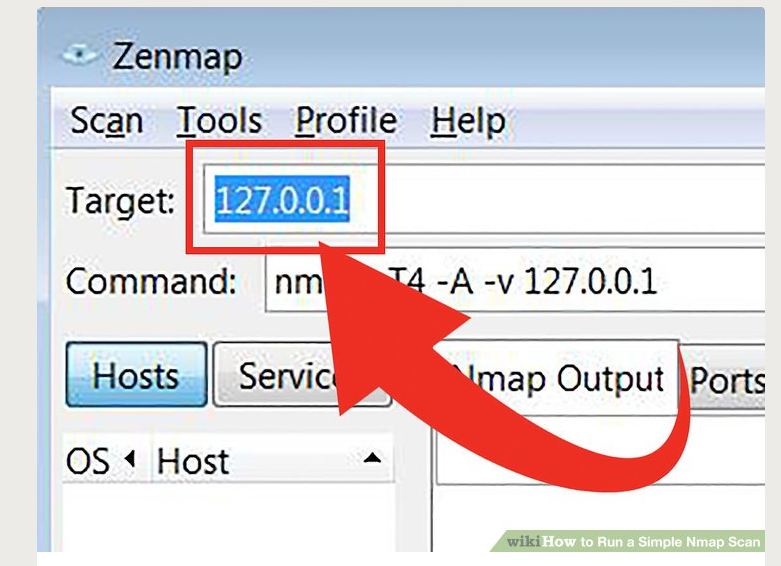
* The Zenmap program is available for Windows, Linux, and Mac OS X. You can find the installation files for all operating systems on the Nmap website.



**2.Install Nmap.** Run the installer once it is finished downloading. You will be asked which components you would like to install. In order to get the full benefit of Nmap, keep all of these checked. Nmap will not in any adware or spyware.

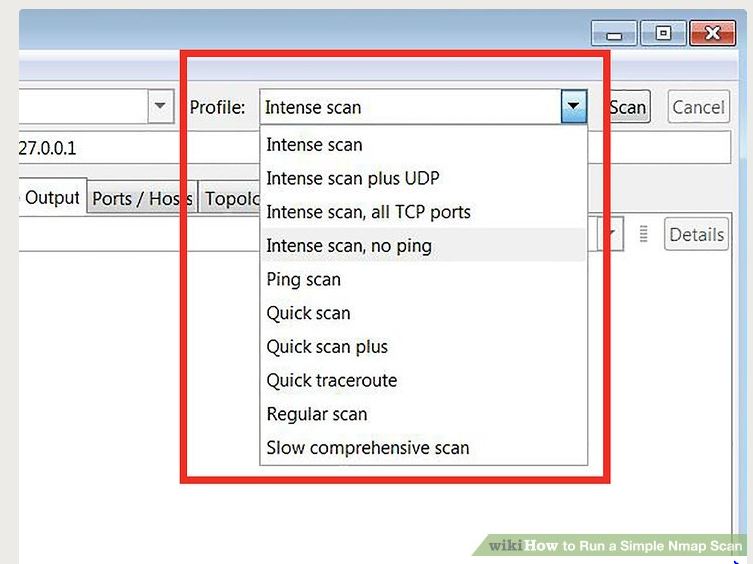


**3.Run the “Nmap – Zenmap” GUI program.** If you left your settings at default during installation, you should be able to see an icon for it on your desktop. If not, look in your Start menu. Opening Zenmap will start the program.



**4.Enter in the target for your scan.** The Zenmap program makes scanning a fairly simple process. The first step to running a scan is choosing your target. You can enter a domain (example.com), an IP address (127.0.0.1), a network (192.168.1.0/24), or a combination of those.

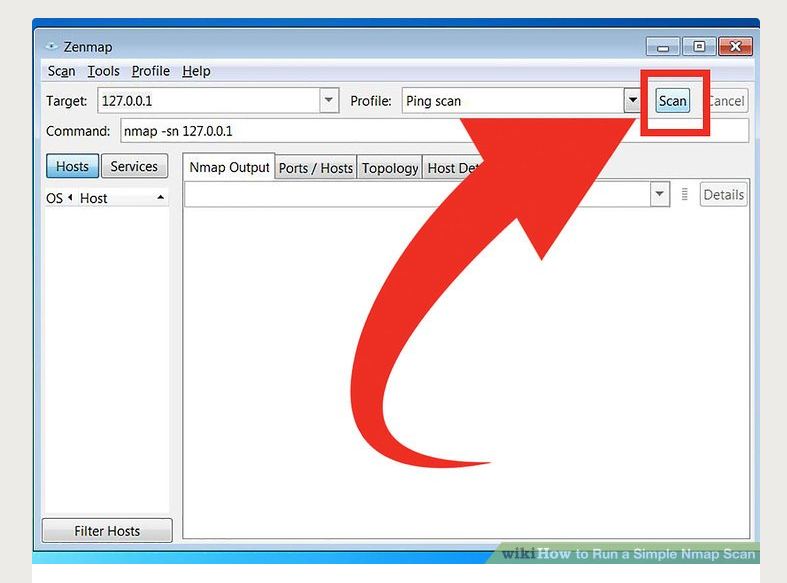
Depending on the intensity and target of your scan, running an Nmap scan may be against the terms of your internet service provider, and may land you in hot water. Always check your local laws and your ISP contract before performing Nmap scans on targets other than your own network.

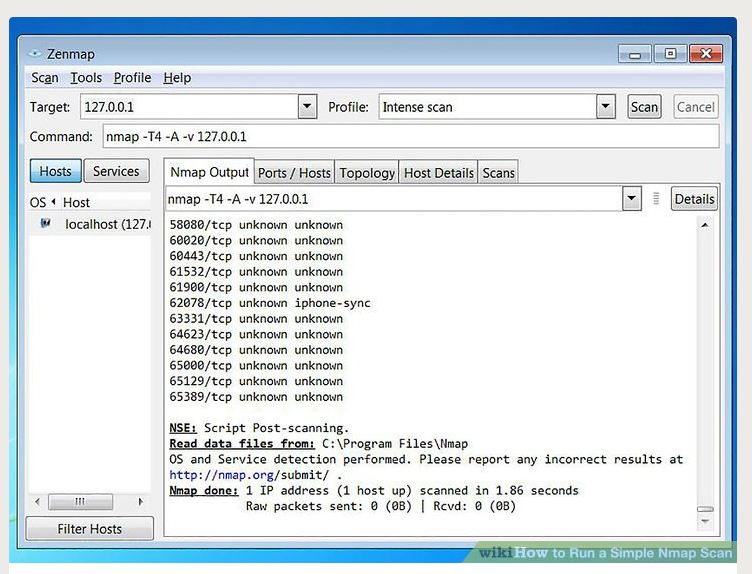


5.

**Choose your Profile.** Profiles are preset groupings of modifiers that change what is scanned. The profiles allow you to quickly select different types of scans without having to type in the modifiers on the command line. Choose the profile that best fits your needs:[[1]](https://www.wikihow.com/Run-a-Simple-Nmap-Scan#_note-1)

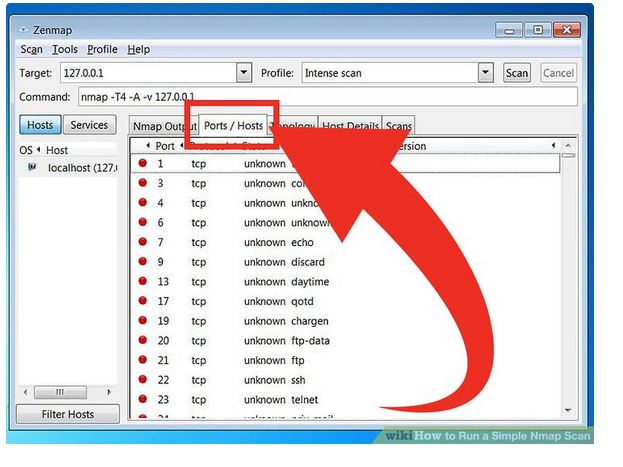
* **Intense scan** - A comprehensive scan. Contains Operating System (OS) detection, version detection, script scanning, traceroute, and has aggressive scan timing. This is considered an intrusive scan.
* **Ping scan** - This scan simply detects if the targets are online, it does not scan any ports.
* **Quick scan** - This is quicker than a regular scan due to aggressive timing and only scanning select ports.
* **Regular scan** - This is the standard Nmap scan without any modifiers. It will return ping and return open ports on the target.



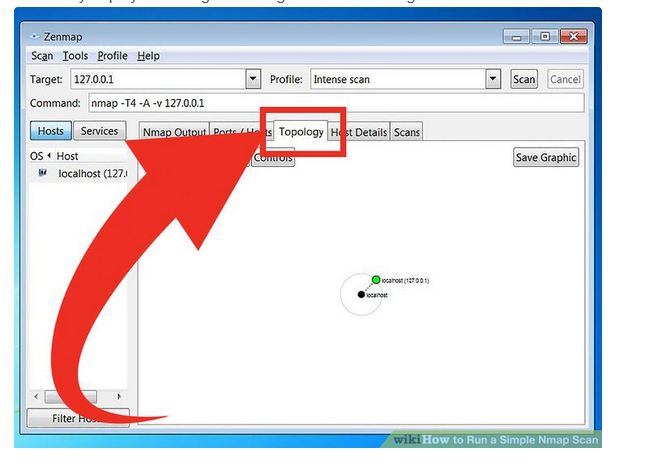
**6.Click Scan to start scanning.** The active results of the scan will be displayed in the Nmap Output tab. The time the scan takes will depend on the scan profile you chose, the physical distance to the target, and the target’s network configuration.

7.**Read your results.** Once the scan is finished, you’ll see the message “Nmap done” at the bottom of the Nmap Output tab. You can now check your results, depending on the type of scan you performed. All of the results will be listed in the main Nmap Output tab, but you can use the other tabs to get a better look at specific data.[[2]](https://www.wikihow.com/Run-a-Simple-Nmap-Scan#_note-2)

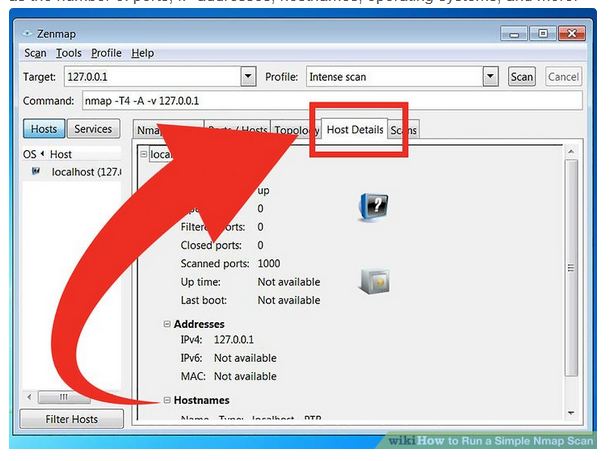
* **Ports/Hosts** - This tab will show the results of your port scan, including the services for those ports.



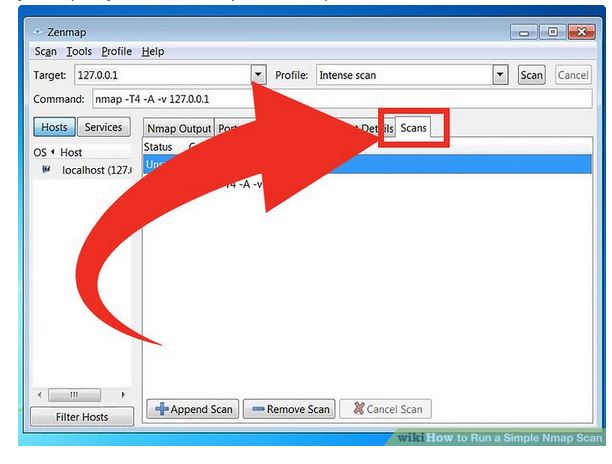
**Topology** - This shows the traceroute for the scan you performed. You can see how many hops your data goes through to reach the target.



**Host Details** - This shows a summary of your target learned through scans, such as the number of ports, IP addresses, hostnames, operating systems, and more.



**Scans** - This tab stores the commands of your previously-run scans. This allows you to quickly re-scan with a specific set of parameters.



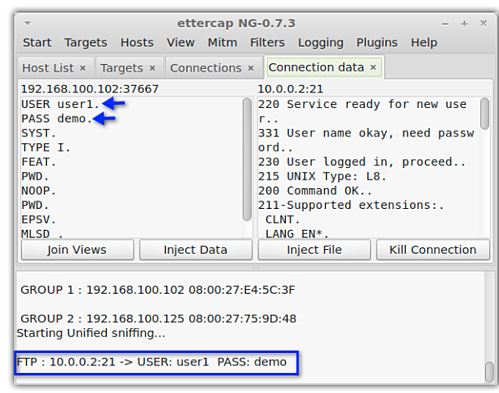
**LAB-6**

**AIM:**Demo of Eavesdropping attack and its Prevention using SSH.

**Understanding the risks of sending files over the Internet:**

Most of the methods we use for sending files over the Internet, like email or FTP, are actually very vulnerable to attacks. When we [FTP a file](https://www.jscape.com/blog/bid/101531/How-to-FTP-a-File), for example, the information is sent as plaintext. What this means is that, given the right tools, a hacker can easily carry out a man-in-the-middle attack, sniff our FTP connection, and view the information being sent - including our FTP username and password.

Here’s a screenshot of a packet sniffer (a network hacking tool) eavesdropping on an FTP connection.



 Once the crooks have obtained our login credentials, they can just simply login to our FTP server and grab whatever files they find.

To know more about sniffing, man-in-the-middle attacks, how such attacks are carried out, and how encrypted file transfers defeat them, read the article [Countering Packet Sniffers Using Encrypted FTP](https://www.jscape.com/blog/bid/91906/Countering-Packet-Sniffers-Using-Encrypted-FTP).

**What SFTP does**

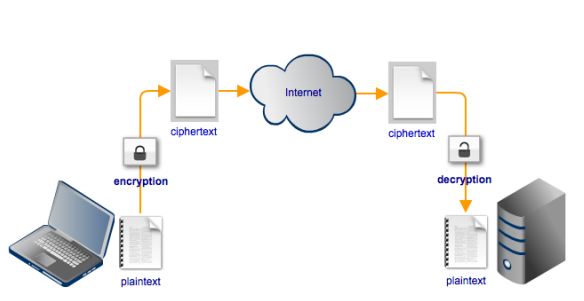
[SFTP](https://www.jscape.com/blog/bid/75602/Understanding-Key-Differences-Between-FTP-FTPS-and-SFTP) a.k.a. Secure File Transfer Protocol a.k.a. SSH File Transfer Protocol protects file transfers from various threats. It does this in two ways:

1. It encrypts the file transfer connection and

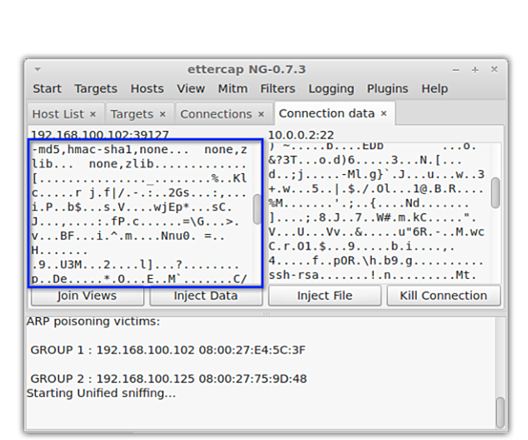
2. It provides stronger authentication

**SFTP encryption**

Encryption renders data unreadable. The data can only be made readable again after it has been decrypted. In an SSH file transfer, data is encrypted throughout the SSH connection. Decryption is done at both ends, i.e., at the server and at the client



Thus, any attempt to eavesdrop on an SFTP file transfer using a man-in-the-middle attack will not succeed.

Here’s a screenshot of the same hacking tool shown earlier, this time displaying an attempt to eavesdrop on an SFTP connection. Notice how the transmitted data is no longer comprehensible

**SFTP authentication**

To authenticate users connecting to the server, file transfer methods like FTP only require a username and password. Now we all know just how easy it is to obtain those login credentials using a packet sniffer. But if the connection is encrypted, then those login credentials are already safe, right? Wrong.

There are many sinister ways of obtaining passwords.

Crooks can perform various social engineering acts like shoulder surfing, phishing, or simply impersonating a legit user and calling a gullible Help Desk agent. More technically skilled individuals can even carry out a [brute force attack on the server](https://www.jscape.com/blog/bid/95157/Protecting-FTP-Passwords-from-Brute-Force-Attacks) itself and steal a bunch of usernames and passwords.

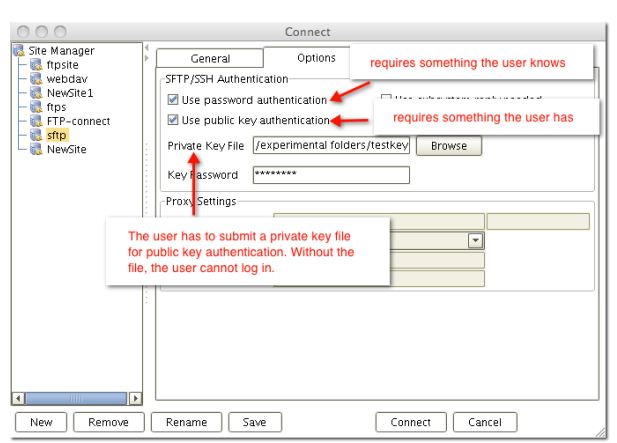
SFTP can make things more difficult for them crooks. That’s because an [SFTP server](https://www.jscape.com/products/file-transfer-servers/jscape-mft-server/) can apply two methods of authentication. The first one asks for something only the user is supposed to know: the user’s username and password. And the second one asks for something only the user is supposed to have: the user’s private key.

This second method, known as public key authentication, enhances the authentication process quite considerably. Even if an attacker is able to obtain a user’s login credentials, he won’t be able to login without the user’s private key.

**The role of a SSH secure file transfer client**

In order to upload or download files to/from a SFTP server, you would of course need a SFTP client or SSH secure file transfer client.

A SSH secure file transfer client typically supports not only SSH encryption but also public key authentication. In other words, it normally provides the option for attaching a private key file. In the SFTP client shown below, the user can enable password authentication, public key authentication, or both.



**LAB-7**

**AIM:Configuration and deployment of Firewall.**

 Controlling outbound network access is an important part of an overall network security plan. For example, you may want to limit access to web sites. Or, you may want to limit the outbound IP addresses and ports that can be accessed.

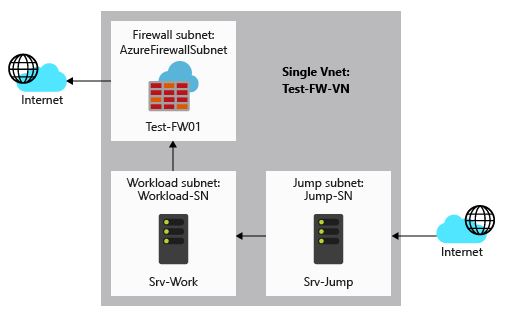
One way you can control outbound network access from an Azure subnet is with Azure Firewall. With Azure Firewall, you can configure:

* Application rules that define fully qualified domain names (FQDNs) that can be accessed from a subnet.
* Network rules that define source address, protocol, destination port, and destination address.

Network traffic is subjected to the configured firewall rules when you route your network traffic to the firewall as the subnet default gateway.

For this session, you create a simplified single VNet with three subnets for easy deployment. For production deployments, a [hub and spoke model](https://docs.microsoft.com/azure/architecture/reference-architectures/hybrid-networking/hub-spoke) is recommended, where the firewall is in its own VNet. The workload servers are in peered VNets in the same region with one or more subnets.

* **AzureFirewallSubnet** - the firewall is in this subnet.
* **Workload-SN** - the workload server is in this subnet. This subnet's network traffic goes through the firewall.
* **Jump-SN** - The "jump" server is in this subnet. The jump server has a public IP address that you can connect to using Remote Desktop. From there, you can then connect to (using another Remote Desktop) the workload server.



In this session, you learn how to:

* Set up a test network environment
* Deploy a firewall
* Create a default route
* Configure an application rule to allow access to [www.google.com](http://www.google.com/)
* Configure a network rule to allow access to external DNS servers
* Test the firewall

If you prefer, you can complete this tutorial using [Azure PowerShell](https://docs.microsoft.com/en-us/azure/firewall/deploy-ps).

If you don't have an Azure subscription, create a [free account](https://azure.microsoft.com/free/?WT.mc_id=A261C142F) before you begin.

## Set up the network

First, create a resource group to contain the resources needed to deploy the firewall. Then create a VNet, subnets, and test servers.

### Create a resource group

The resource group contains all the resources for the tutorial.

1. Sign in to the Azure portal at [https://portal.azure.com](https://portal.azure.com/).
2. On the Azure portal home page, select **Resource groups** > **Add**.
3. For **Resource group name**, type **Test-FW-RG**.
4. For **Subscription**, select your subscription.
5. For **Resource group location**, select a location. All subsequent resources that you create must be in the same location.
6. Select **Create**.

### Create a VNet

This VNet will contain three subnets.

1. From the Azure portal home page, select **Create a resource**.
2. Under **Networking**, select **Virtual network**.
3. For **Name**, type **Test-FW-VN**.
4. For **Address space**, type **10.0.0.0/16**.
5. For **Subscription**, select your subscription.
6. For **Resource group**, select **Test-FW-RG**.
7. For **Location**, select the same location that you used previously.
8. Under **Subnet**, for **Name** type **AzureFirewallSubnet**. The firewall will be in this subnet, and the subnet name **must** be AzureFirewallSubnet.
9. For **Address range**, type **10.0.1.0/24**.
10. Accept the other default settings, and then select **Create**.

**Note**

The minimum size of the AzureFirewallSubnet subnet is /26.

Create additional subnets

Next, create subnets for the jump server, and a subnet for the workload servers.

1. On the Azure portal home page, select **Resource groups** > **Test-FW-RG**.
2. Select the **Test-FW-VN** virtual network.
3. Select **Subnets** > **+Subnet**.
4. For **Name**, type **Workload-SN**.
5. For **Address range**, type **10.0.2.0/24**.
6. Select **OK**.

Create another subnet named **Jump-SN**, address range **10.0.3.0/24**.

### Create virtual machines

Now create the jump and workload virtual machines, and place them in the appropriate subnets.

1. On the Azure portal, select **Create a resource**.
2. Select **Compute** and then select **Windows Server 2016 Datacenter** in the Featured list.
3. Enter these values for the virtual machine:

| **Setting** | **Value** |
| --- | --- |
| Resource group | **Test-FW-RG** |
| Virtual machine name | **Srv-Jump** |
| Region | Same as previous |
| Administrator user name | **azureuser** |
| Password | **Azure123456!** |

1. Under **Inbound port rules**, for **Public inbound ports**, select **Allow selected ports**.
2. For **Select inbound ports**, select **RDP (3389)**.
3. Accept the other defaults and select **Next: Disks**.
4. Accept the disk defaults and select **Next: Networking**.
5. Make sure that **Test-FW-VN** is selected for the virtual network and the subnet is **Jump-SN**.
6. For **Public IP**, accept the default new public ip address name (Srv-Jump-ip).
7. Accept the other defaults and select **Next: Management**.
8. Select **Off** to disable boot diagnostics. Accept the other defaults and select **Review + create**.
9. Review the settings on the summary page, and then select **Create**.

Use the information in the following table to configure another virtual machine named **Srv-Work**. The rest of the configuration is the same as the Srv-Jump virtual machine.

| **Setting** | **Value** |
| --- | --- |
| Subnet | **Workload-SN** |
| Public IP | **None** |
| Public inbound ports | **None** |

## Deploy the firewall

Deploy the firewall into the VNet.

1. From the portal home page, select **Create a resource**.
2. Type **firewall** in the search box and press **Enter**.
3. Select **Firewall** and then select **Create**.
4. On the **Create a Firewall** page, use the following table to configure the firewall:

| **Setting** | **Value** |
| --- | --- |
| Subscription | <your subscription> |
| Resource group | **Test-FW-RG** |
| Name | **Test-FW01** |
| Location | Select the same location that you used previously |

1. Select **Review + create**.
2. Review the summary, and then select **Create** to create the firewall.

This will take a few minutes to deploy.

1. After deployment completes, go to the **Test-FW-RG** resource group, and select the **Test-FW01** firewall.
2. Note the private IP address. You'll use it later when you create the default route.

## Create a default route

For the **Workload-SN** subnet, configure the outbound default route to go through the firewall.

1. From the Azure portal home page, select **All services**.
2. Under **Networking**, select **Route tables**.
3. Select **Add**.
4. For **Name**, type **Firewall-route**.
5. For **Subscription**, select your subscription.
6. For **Resource group**, select **Test-FW-RG**.
7. For **Location**, select the same location that you used previously.
8. Select **Create**.
9. Select **Refresh**, and then select the **Firewall-route** route table.
10. Select **Subnets** and then select **Associate**.
11. Select **Virtual network** > **Test-FW-VN**.
12. For **Subnet**, select **Workload-SN**. Make sure that you select only the **Workload-SN** subnet for this route, otherwise your firewall won't work correctly.
13. Select **OK**.
14. Select **Routes** and then select **Add**.
15. For **Route name**, type **fw-dg**.
16. For **Address prefix**, type **0.0.0.0/0**.
17. For **Next hop type**, select **Virtual appliance**.
18. For **Next hop address**, type the private IP address for the firewall that you noted previously
19. Select **OK**.

## Configure an application rule

This is the application rule that allows outbound access to [www.google.com](http://www.google.com/).

1. Open the **Test-FW-RG**, and select the **Test-FW01** firewall.
2. On the **Test-FW01** page, under **Settings**, select **Rules**.
3. Select the **Application rule collection** tab.
4. Select **Add application rule collection**.
5. For **Name**, type **App-Coll01**.
6. For **Priority**, type **200**.
7. For **Action**, select **Allow**.
8. Under **Rules**, **Target FQDNs**, for **Name**, type **Allow-Google**.
9. For **Source Addresses**, type **10.0.2.0/24**.
10. For **Protocol:port**, type **http, https**.
11. For **Target FQDNS**, type [**www.google.com**](http://www.google.com/)
12. Select **Add**.

Azure Firewall includes a built-in rule collection for infrastructure FQDNs that are allowed by default. These FQDNs are specific for the platform and can't be used for other purposes. For more information, see [Infrastructure FQDNs](https://docs.microsoft.com/en-us/azure/firewall/infrastructure-fqdns).

## Configure a network rule

This is the network rule that allows outbound access to two IP addresses at port 53 (DNS).

1. Select the **Network rule collection** tab.
2. Select **Add network rule collection**.
3. For **Name**, type **Net-Coll01**.
4. For **Priority**, type **200**.
5. For **Action**, select **Allow**.
6. Under **Rules**, for **Name**, type **Allow-DNS**.
7. For **Protocol**, select **UDP**.
8. For **Source Addresses**, type **10.0.2.0/24**.
9. For Destination address, type **209.244.0.3,209.244.0.4**
10. For **Destination Ports**, type **53**.
11. Select **Add**.

### Change the primary and secondary DNS address for the **Srv-Work**network interface

For testing purposes in this tutorial, configure the server's primary and secondary DNS addresses. This isn't a general Azure Firewall requirement.

1. From the Azure portal, open the **Test-FW-RG** resource group.
2. Select the network interface for the **Srv-Work** virtual machine.
3. Under **Settings**, select **DNS servers**.
4. Under **DNS servers**, select **Custom**.
5. Type **209.244.0.3** in the **Add DNS server** text box, and **209.244.0.4** in the next text box.
6. Select **Save**.
7. Restart the **Srv-Work** virtual machine.

## Test the firewall

Now, test the firewall to confirm that it works as expected.

1. From the Azure portal, review the network settings for the **Srv-Work** virtual machine and note the private IP address.
2. Connect a remote desktop to **Srv-Jump** virtual machine, and sign in. From there, open a remote desktop connection to the **Srv-Work** private IP address.
3. Open Internet Explorer and browse to [https://www.google.com](https://www.google.com/).
4. Select **OK** > **Close** on the Internet Explorer security alerts.

You should see the Google home page.

1. Browse to [https://www.microsoft.com](https://www.microsoft.com/).

You should be blocked by the firewall.

So now you've verified that the firewall rules are working:

* You can browse to the one allowed FQDN, but not to any others.
* You can resolve DNS names using the configured external DNS server.

## Clean up resources

You can keep your firewall resources for the next tutorial, or if no longer needed, delete the **Test-FW-RG** resource group to delete all firewall-related resources.

**VIVA VOICE QUESTIONS**

1. **Define security attack**.

Any action that compromises the security of information owned by an organization.

1. **Define security mechanism.**

A process (or a device incorporating such a process) that is designed to detect, prevent, or recover from a security attack.

1. **Define security service.**

A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization.

1. **Define Active Attack.**

A passive attack attempts to learn or make use of information from the system but does not affect system resources.

1. **Define Passive Attack.**

An active attack attempts to alter system resources or affect their operation.

1. **Define Masquerade.**

A masquerade takes place when one entity pretends to be a different entity.

1. **Define Replay.**

Replay involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect

1. **Define Modification of messages.**

Modification of messages simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect

1. **Define denial of service.**

The denial of service prevents or inhibits the normal use or management of communications facilities

1. **Define release of message contents.**

A telephonic conversation, an E-mail message or a transferred file may contain confidential data. A passive attack may monitor the contents of these transmission.

1. **Define traffic analysis.**

In this attack the eavesdropper analyzes the traffic, determine the location, identify communicating hosts, observes the frequency and length of message being exchanged. Using all these information they predict the nature of communication . All incoming and out going traffic of network is analysed but not altered.

1. **Define Confidentiality.**

Preserving authorized restrictions on information access

and disclosure, including means for protecting personal privacy and proprietary information. A loss of confidentiality is the unauthorized disclosure of information.

1. **Define Integrity.**

Guarding against improper information modification or destruction, including ensuring information nonrepudiation and authenticity. A loss of integrity is the unauthorized modification or destruction of information.

1. **Define Availability.**

Ensuring timely and reliable access to and use of information. A loss of availability is the disruption of access to or use of information or an information system.

1. **Define Encryption.**

The process of converting from plaintext to ciphertext is known as encryption

1. **Define Decryption.**

The process of restoring the plaintext from the ciphertext is known as decryption.

1. **Define Plaintext.**

An original message is known as the plaintext.

1. **Define Ciphertext.**

The coded message is called the ciphertext.

1. **Define Cryptography.**

Cryptography render the message unintelligible to outsiders by various transformations

of the text.

1. **Define Steganography.**

Steganography conceal the existence of the message

1. **List the types of Active Attacks.**

masquerade, replay, modification of messages, and denial of service.

1. **List the types of Passive Attacks.**

release of message contents and traffic analysis.

**23. List various security services.**

Authentication, Data Integrity, Access Control, Data Confidentiality, Non-repudiation

1. **List various security mechanisms.**

Encipherment, Digital Signature, Access Control, Data Integrity, Authentication Exchange, Traffic Padding, Routing Control, Notarization

1. **Describe the two kinds of attacks.**

Passive and Active Attacks

1. **What are the 2 basic building blocks of all encryption techniques?**

substitution and transposition.

1. **What are the types of substitution techniques?**

Caesar Cipher, Mono alphabetic Cipher, Poly alphabetic cipher, hill cipher, play fair cipher, one time pad cipher.

1. **What are the 2 authentication services?**

Peer entity authentication, Data origin authentication

1. **Define Substitution technique.**

A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols.

1. **Define Transposition technique.** A Transposition technique is kind of mapping is achieved by performing some sort of permutation on the plaintext letters
2. **Abbreviate DES.**

Data Encryption Standard

1. **Abbreviate NIST.**

National Institute of Standards and Technology

1. **Abbreviate FIPS.**

Federal Information Processing Standard

1. **What is length of plaintext in DES?**

64 bits

1. **What is length of key in DES?**

56 bits

1. **What is IP in DES?**

Initial Permutation

1. **What is size of key after permuted choice 2?**

48 bits

1. **Typical number of rounds in Feistel Cipher.**
2. **Define Block cipher.**

A block cipher is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length.

1. **Define Stream Cipher**.

A stream cipheris one that encrypts a digital data stream one bit or one byte at a time.

1. **Define Substitution.**

Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements.

1. **Define Permutation.**

A sequence of plaintext elements is replaced by a permutation of that sequence.

1. **PGP?**

Pretty good privacy.