

Thadomal Shahani Engineering College
Bandra (W.), Mumbai - 400 050.

CERTIFICATE

Certify that Mr./Miss Shirish Shetty
of comps Department, Semester VI with
Roll No.2103164 has completed a course of the necessary
experiments in the subject CSS under my
supervision in the **Thadomal Shahani Engineering College**
Laboratory in the year 2023 - 2024


Teacher In- Charge

Head of the Department

Principal

Date 4/6/24

CONTENTS

SR. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
1.)	Implementation of Extended Euclidean algorithm (L01)	1 - 4	17/1/24	
2.)	Implementation of Caesar Cipher (L01)	4 - 8	24/1/24	
3.)	Implementation of Playfair Cipher	8 - 12	31/1/24	
4.)	Implementation of Euler Totient Function		7/2/24	
5.)	Implementation of RSA cryptosystem.	12 - 20	21/2/24	
6.)	Implementation of Diffie Hellman Key exchange algorithm.	21 - 28	28/2/24	
7.)	Study the use of network reconnaissance tools and apply the following : WHOIS, dig, traceroute, nslookup.	29 - 35	6/3/24	Qashim 10/3/24 21/3/24
8.)	Study and implementation of packet sniffer tool: wireshark	36 - 45	13/3/24	
9.)	Design of personal Firewall using Iptables.	45 - 65	23/3/24	
10.)	Simulation of Buffer Overflow Attack	65 - 72	27/3/24	
11.)	Assignment - I	72 - 78	31/1/24	
12.)	Assignment - II	79 - 84	28/2/24	
13.)	Assignment - III.	84 - 89	10/4/24	

EXPERIMENT: 1

AIM: Implementation of Extended Euclidean algorithm.

THEORY:

The Extended Euclidean Algorithm is a method used to find the greatest common divisor (GCD) of two integers while simultaneously determining the coefficients that express this GCD as a linear combination of the original numbers. It extends the basic Euclidean Algorithm by calculating additional values that represent the coefficients of the GCD expression. These coefficients allow for the representation of the GCD as a linear combination of the input numbers, which is particularly useful in number theory and cryptography.

To understand the algorithm, here are the steps for it:

1. Initialization: Begin with the two numbers you want to find the GCD of, let's call them a and b . Initialize variables:

- $s1 = 1, s2 = 0$
- $t1 = 0, t2 = 1$
- $r1 = a, r2 = b$
- $q = \text{floor}(r1/2)$
- $r = r1 \% r2$
- $s = s1 - q * s2$
- $t = t1 - q * t2$

2. Iteration: Repeat until r becomes 0:

- Replace $r1$ with $r2$
- Replace $r2$ with r
- Calculate r by $r1 \% r2$
- Replace $s1$ with $s2$
- Replace $s2$ with s
- Calculate s with $s1 - q * s2$
- Replace $t1$ with $t2$
- Replace $t2$ with t

- Calculate t with $t1 - q * t2$

3. **Termination:** When r becomes 0, the last non zero remainder is the gcd of the number a and b . s and t are the coefficient of the linear equation of the combination of a and b .

- The formula for calculation of gcd of a and b is given as:
- $\text{Gcd}(a,b) = s * a + t * b$

Here's a simple example:

Let's find the GCD of $a=35$ and $b=15$.

1. Initialization:

- $s=0, old_s=1$
- $t=1, old_t=0$
- $r=15, old_r=35$

2. Iteration:

- $q=\text{floor}(35/15)=2, r=35-2\times15=5$
- Update: $old_r=15, r=5$
- Update: $old_s=0, s=1$
- Update: $0-2\times1=-2, old_t=1, t=-2$
- Next iteration: $q=\text{floor}(5/15)=3, r=15-3\times5=0$

3. Termination:

- $r=0$, so the GCD is the last nonzero remainder, which is 5.
- From the last iteration, $s=1$ and $t=-2$, so
$$\text{gcd}(35,15)=1\times35+(-2)\times15=35-30=5$$
$$\text{gcd}(35,15)=1\times35+(-2)\times15=35-30=5$$

CODE:

```
a = int(input("Enter the first number: "))
b = int(input("Enter the second number: "))
num1 = a
num2 = b

s1, s2, t1, t2 = 1, 0, 0, 1

'''s=s1-q*s2
t = t1 - q*t2'''

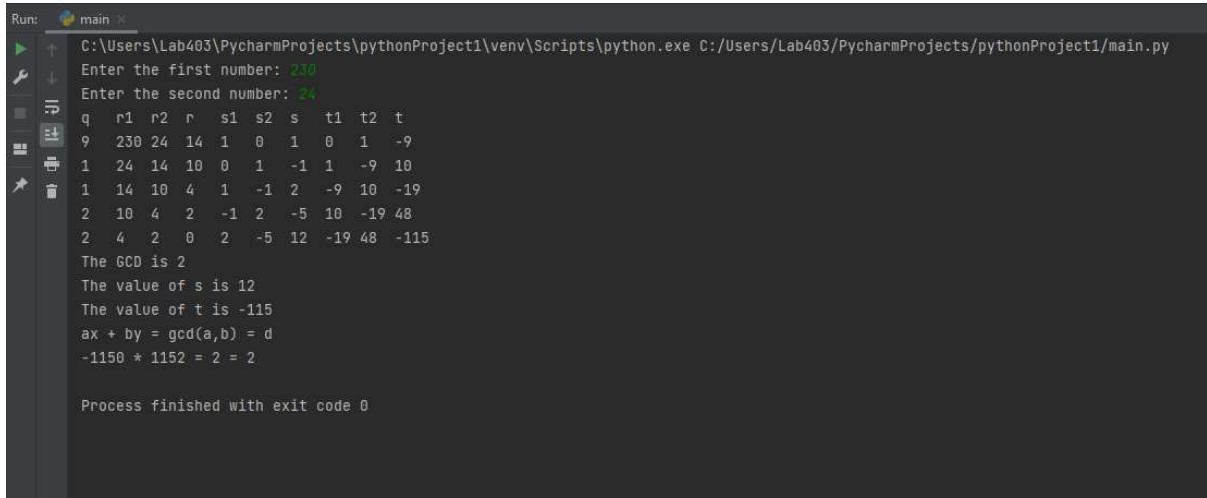
q = int(a / b)
r = int(a % b)
s = s1 - q * s2
t = t1 - q * t2
print("q\t r1\t r2\t r\t s1\t s2\t s\t t1\t t2\t t")
while(r != 0):
    print(f"{q}\t {r1}\t {r2}\t {r}\t {s1}\t {s2}\t {s}\t {t1}\t {t2}\t {t}")
    a = b
    b = r
    s1 = s2
    s2 = s
    t1 = t2
    t2 = t
    q = int(a / b)
    r = int(a % b)
    s = s1 - q * s2
    t = t1 - q * t2
print(f"\n{q}\t {a}\t {b}\t {r}\t {s1}\t {s2}\t {s}\t {t1}\t {t2}\t {t}")

print(f"The GCD is {b}")
print(f"The value of s is {s}")
print(f"The value of t is {t}")

print("ax + by = gcd(a,b) = d")
m1 = num1 * s2
m2 = num2 * t2
res = m1 + m2

print(m1, "* ", m2, " = ", res, " = ", b)
```

OUTPUT:



The screenshot shows the PyCharm IDE's Run tab with the following output:

```
Run: main ×
C:\Users\Lab403\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:/Users/Lab403/PycharmProjects/pythonProject1/main.py
Enter the first number: 230
Enter the second number: 24
      q   r1   r2   r   s1   s2   s   t1   t2   t
      9   230  24  14  1   0   1   0   1   -9
      1   24   14  10  0   1   -1   1   -9  10
      1   14   10   4   1   -1   2   -9  10  -19
      2   10   4   2   -1   2   -5  10  -19  48
      2   4   2   0   2   -5  12  -19  48  -115
The GCD is 2
The value of s is 12
The value of t is -115
ax + by = gcd(a,b) = d
-1150 * 1152 = 2 = 2

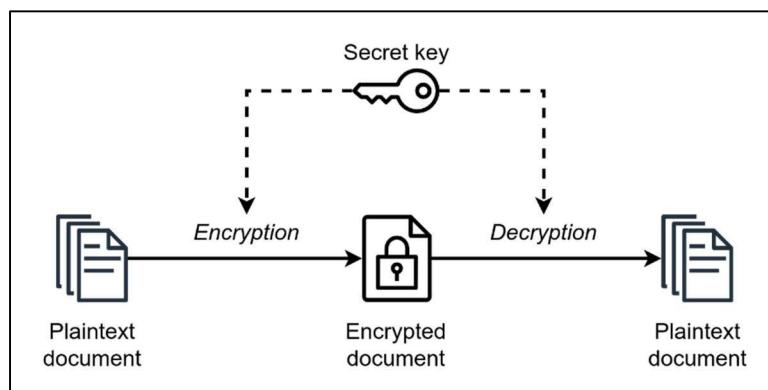
Process finished with exit code 0
```

EXPERIMENT NO: 2

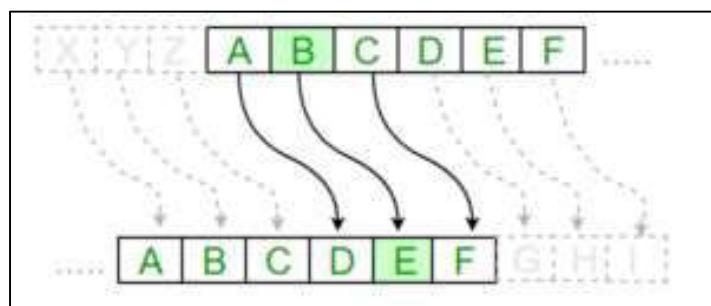
AIM: Implementation of Ceaser Cipher

THEORY:

In cryptography, a **symmetric key** is one that is used both to encrypt and decrypt information. This means that to decrypt information, one must have the same key that was used to encrypt it. The keys, in practice, represent a shared secret between two or more parties that can be used to maintain a private information link. This requirement that both parties have access to the secret key is one of the main drawbacks of symmetric key encryption, in comparison to public-key encryption.



In cryptography, a **substitution cipher** is a method of encrypting in which units of plaintext are replaced with the ciphertext, in a defined manner, with the help of a key; the "units" may be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth. The receiver deciphers the text by performing the inverse substitution process to extract the original message.



The **Caesar cipher** is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher where each letter in the plaintext is shifted a certain number of places down or up the alphabet. For example, with a shift of 3, 'A' would be replaced by 'D',

'B' would become 'E', and so on. The method is named after Julius Caesar, who allegedly used it to communicate with his generals.

A	B	C	D	E	F	G	H	I	J	K	L	M
↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
13	14	15	16	17	18	19	20	21	22	23	24	25

Table 2: Encoding English capital letters using integers from \mathbb{Z}_{26} .

Encryption in Ceaser Cipher is done using Modulus of 26 denoted as \mathbb{Z}_{26} . It uses the same key to encrypt and decrypt the message. Following are the steps for encryption:

1. Convert every character in the Plain Text into it's indices using the above table. For eg: if Plain Text is "AXE" then it's indices are: "0 23 4".
2. After getting the indices of every character, use the formula given below:

$$CT(i) = (PT(i) + (\text{key}) \bmod 26) \bmod 26$$

3. Using the formula given above the corresponding indices of Cipher Text.
4. Convert the indices into corresponding Characters using the above table. The cipher text is generated.

Decryption in Ceaser Cipher is done using the following steps:

1. Convert the Cipher text characters into corresponding indices using the above table.
2. After getting the indices use the following formula:

$$PT(i) = (CT(i) - (\text{key}) \bmod 26) \bmod 26$$

3. Using the formula the indices for plain text are generated.
4. Convert the indices into characters using the above table.

EXAMPLE:

$PT = "PARTH"$ key = 5

Encryption:

1. $P \ A \ R \ T \ H$
15 0 17 19 7
2. $CT(P) \Rightarrow (15 + 5) \% 26 \Rightarrow 20 \Rightarrow U$
 $CT(A) \Rightarrow (0 + 5) \% 26 \Rightarrow 5 \Rightarrow F$
 $CT(R) \Rightarrow (17 + 5) \% 26 \Rightarrow 22 \Rightarrow W$
 $CT(T) \Rightarrow (19 + 5) \% 26 \Rightarrow 24 \Rightarrow Y$
 $CT(H) \Rightarrow (7 + 5) \% 26 \Rightarrow 12 \Rightarrow M$
3. $CT: UFWMY$

Decryption: $CT = UFWMY$

$$PT(U) = (20 - 5) \% 26 = 15 \Rightarrow P$$

$$PT(F) = (5 - 5) \% 26 = 0 \Rightarrow A$$

$$PT(W) = (22 - 5) \% 26 = 17 \Rightarrow R$$

$$PT(Y) = (24 - 5) \% 26 = 19 \Rightarrow T$$

$$PT(M) = (12 - 5) \% 26 = 7 \Rightarrow I$$

$PT: PARTH$

```
import java.io.*;
public class CeaserCipher
{
    static String Encryption(String PlainText, int Key)
    {
        int len = PlainText.length();
        char[] pt,ct;
        pt = new char[len];
        ct = new char[len];
        pt = PlainText.toCharArray();

        for(int i = 0 ; i < len ; i++)
        {
            if(pt[i] >= 'A' && pt[i] <= 'Z')
            {
                ct[i] = (char)( ( ( (int)pt[i] - 65 ) + (Key%26)) % 26 ) + 65 ;
            }
            if(pt[i] >= 'a' && pt[i] <= 'z')
            {
                ct[i] = (char)( ( ( (int)pt[i] - 97 ) + (Key%26)) % 26 ) + 97 ;
            }
            if(pt[i] >= '0' && pt[i] <= '9')
            {
                ct[i] = (char)( ( ( (int)pt[i] - 48 ) + (Key%26)) % 10 ) + 48 ;
            }
            if(pt[i] >= '!' && pt[i] <= '/')
            {
                ct[i] = (char)( ( ( (int)pt[i] - 33 ) + (Key%26)) % 15 ) + 33 ;
            }
        }
    }
}
```

```
if(pt[i] == ' ')
{
    ct[i] = ' ';
}

String CipherText = new String(ct);
return CipherText;
}

static String Decryption(String CipherText, int Key)
{
    int len = CipherText.length();
    char[] pt,ct;
    pt = new char[len];
    ct = new char[len];
    ct = CipherText.toCharArray();

    for(int i = 0 ; i < len ; i++)
    {
        if(ct[i] >= 'A' && ct[i] <= 'Z')
        {
            pt[i] = (char)( ( ( (int)ct[i] + 26 - 65 ) - (Key%26)) % 26 ) + 65 );
        }
        if(ct[i] >= 'a' && ct[i] <= 'z')
        {
            pt[i] = (char)( ( ( (int)ct[i] + 26 - 97 ) - (Key%26)) % 26 ) + 97 );
        }
        if(ct[i] >= '0' && ct[i] <= '9')
        {
            pt[i] = (char)( ( ( (int)ct[i] + 20 - 48 ) - (Key%26)) % 10 ) + 48 );
        }
    }
}
```

```
    }

    if(ct[i] >= '!' && ct[i] <= '/') {
        pt[i] = (char)( ( ( (int)ct[i] + 30 - 33 ) - (Key%26)) % 15 ) + 33 );
    }

    if(ct[i] == ' ')
    {
        pt[i] = ' ';
    }
}

String PlainText = new String(pt);

return PlainText;
}

public static void main(String[] args) throws IOException
{
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    System.out.println("Enter the Plain Text to be encrypted: ");

    String PlainText = br.readLine();

    System.out.println("Enter the key to be used for the cipher: ");

    int Key = Integer.parseInt(br.readLine());

    String CipherText = Encryption(PlainText, Key);

    System.out.println("After Encryption: "+CipherText);

    String newPlainText = Decryption(CipherText, Key);

    System.out.println("After Decryption on Cipher Text: "+newPlainText);
}
```

The screenshot shows a terminal window with the following interface elements:

- Top navigation bar with tabs: PROBLEMS (2), OUTPUT, DEBUG CONSOLE, TERMINAL (underlined), PORTS.
- Script content:
 - PS D:\DS\Practise> `cd "d:\DS\Practise\" ; if ($?)`
 - Enter the Plain Text to be encrypted:
TSEC C12 #\$
 - Enter the key to be used for the cipher:
23
 - After Encryption: QPBZ Z45 +,
 - After Decryption on Cipher Text: TSEC C12 #\$
 - PS D:\DS\Practise> █

EXPERIMENT NO: 3

AIM: Implementation of Playfair Cipher.

THEORY: The Polyalphabetic cipher is a cryptographic technique that uses multiple substitution alphabets to encrypt plaintext characters. It differs from monoalphabetic ciphers by using a variety of substitution rules, making it more secure. The most famous Polyalphabetic cipher is the Vigenère cipher, invented by Blaise de Vigenère in the 16th century. It uses a keyword to determine which alphabet to use for each letter of the plaintext, repeating the keyword if necessary. Other Polyalphabetic ciphers include:

1. Playfair Cipher
2. Vigenère Cipher
3. Hill Cipher

The Playfair Cipher is a symmetric encryption technique that operates on pairs of letters, rather than single letters like traditional ciphers. It uses a 5x5 matrix called a key table, which contains a keyword without repeating letters followed by the remaining letters of the alphabet (excluding 'J'). The encryption process involves breaking the plaintext into pairs of letters (digraphs), applying specific rules to each pair, and then mapping them onto the key table to obtain the ciphertext. The rules include handling pairs with identical letters, pairs with letters in the same row, and pairs with letters in the same column. By using these rules and the key table, the Playfair Cipher provides a more secure method of encryption compared to monoalphabetic ciphers.

Algorithm: Initially, we are provided with a Plain text which is to be encrypted. We are also given a key using which the Plain Text can be encrypted.

1. Create a 5x5 Matrix. Fill the matrix with Plain Text such that its letters must not repeat in the matrix. For example if the key text is “SECURITY” then the matrix will be filled as:

S	E	C	U	R
I	T	Y		

2. Fill the rest of the cells with alphabets such that it must not be present in the Key Text. Fill the cells from A-Z accordingly.

3. Since there are only 25 cells in the matrix, only 25 alphabets can be filled in the matrix. Therefore, I and J are placed in the same cell and can be interchanged during encryption.

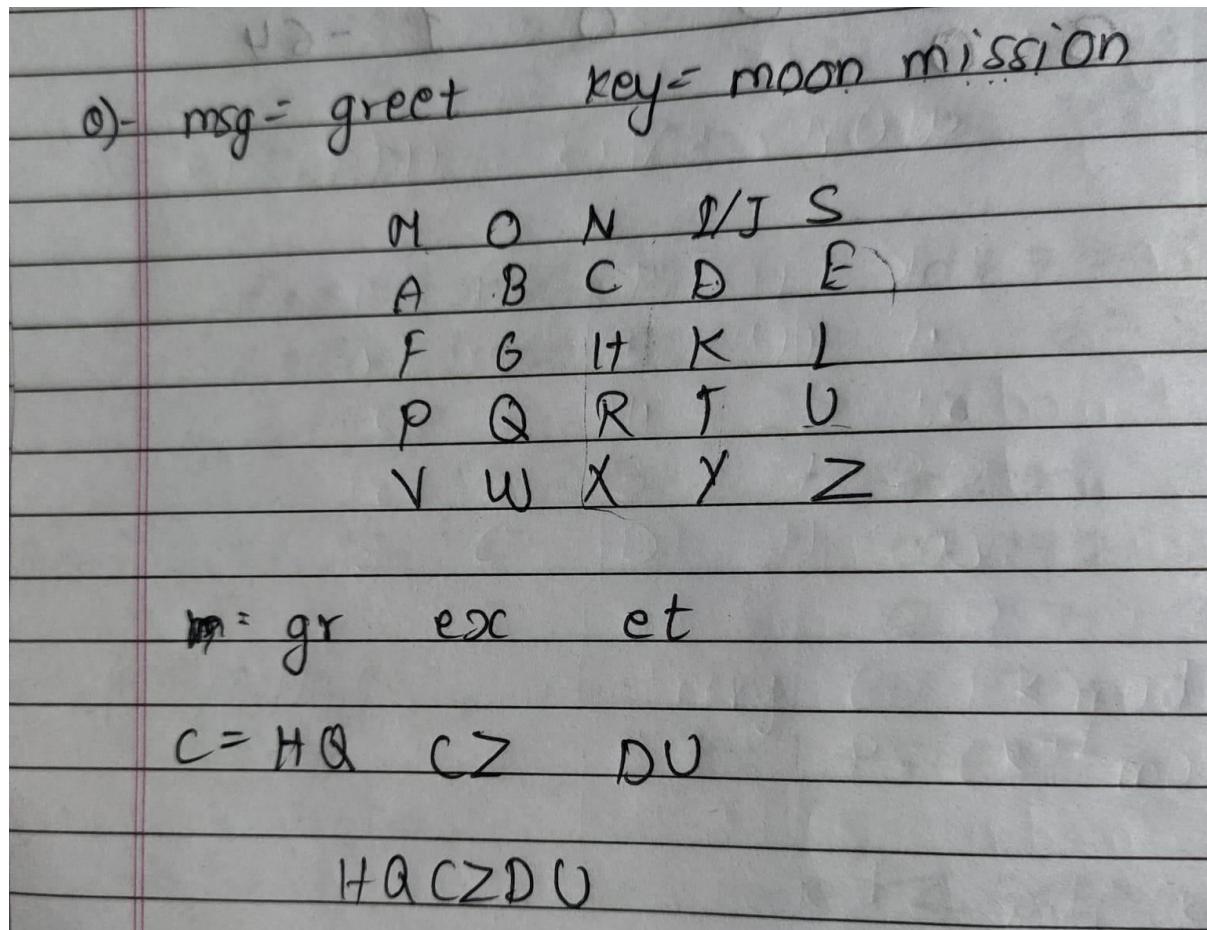
S	E	C	U	R
I	T	Y	A	B
D	F	G	H	K
L	M	N	O	P
Q	V	W	X	z

4. The next step is creating pairs of the Plain Text for the algorithm. If there are alphabets which are repeating then fill a bogus character in between them. For example, if the Plain Text is "HELLO WORLD" then HE, LX, LO, WO, RL, DX.
5. There are three rules to be followed while encryption:
- If the Letter in the pair is in the same row, then consider the immediate next letter.
 - If the Letter in the pair is in the same column, then consider the immediate next letter.
 - Make a rectangle in the matrix and then consider the extremes in the rectangle.

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

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

Example:



```
import java.util.*;  
  
public class playFair  
{  
  
    static char[] alphabetChecker(char keyArr[])  
    {  
  
        char result[] = new char[30];  
  
        int index = 0;  
  
        for(char c = 'A'; c <= 'Z'; c++)  
        {  
  
            if(c == 'J')  
                continue;  
  
            int t = 0;  
  
            for(char w : keyArr)  
            {  
  
                if(c == w)  
                    break;  
  
                else  
                    t++;  
            }  
  
            if(t == keyArr.length)  
            {  
  
                result[index] = c;  
  
                index++;  
            }  
        }  
  
        return result;  
    }  
  
    static char[] removeDuplicate(String key)  
    {  
  
        int len = 0, index = 0;  
    }
```

```
Set<Character> arr = new LinkedHashSet<>();
char temp[] = key.toCharArray();
for(char c : temp)
{
    arr.add(c);
    len++;
}
char newKey[] = new char[len];
for(char c : arr)
{
    newKey[index] = c;
    index++;
}
return newKey;
}
```

```
static char[][] createMatrix(char keyArr[])
{
    char result[][] = new char[5][5];
    int index = 0, ind = 0;
    char alpha[] = alphabetChecker(keyArr);
    for(int i = 0; i < 5 ; i++)
    {
        for(int j = 0; j < 5; j++)
        {
            if(index < keyArr.length-1)
            {
                result[i][j] = keyArr[index];
                index++;
            }
            else if(index >= keyArr.length-1)
```

```
{  
    result[i][j] = alpha[ind];  
    ind++;  
}  
}  
}  
return result;  
}  
  
static char[] pairMaker(String plainText)  
{  
    char beforePair[] = plainText.toCharArray();  
    char newPair[] = new char[30];  
    ArrayList<Character> arr = new ArrayList<Character>(20);  
  
    for(char c : beforePair)  
    {  
        arr.add(c);  
    }  
  
    for(int i = 0 ; i < arr.size() - 1 ; i++)  
    {  
        if(arr.get(i) == arr.get(i+1))  
        {  
            arr.add(i+1,'X');  
        }  
    }  
  
    if(arr.size() % 2 != 0)  
    {  
        arr.add('X');  
    }  
}
```

```
}

for(int i = 0; i < arr.size(); i++)
{
    newPair[i] = arr.get(i);
}

return newPair;
}

static int[] findPosition(char c, char keyArr[])
{
    char matrix[][] = createMatrix(keyArr);
    int res[] = new int[2];
    for(int i = 0 ; i < 5 ; i++)
    {
        for(int j = 0 ; j < 5 ; j++)
        {
            if(matrix[i][j] == c)
            {
                res[0] = i;
                res[1] = j;
                break;
            }
        }
    }
    return res;
}

static String encryptPlayFair(String plainText, char keyArr[])
{
    char newPair[] = pairMaker(plainText);
```

```
char matrix[][] = createMatrix(keyArr);

char a,b;

StringBuilder cipherText = new StringBuilder();

for(int i = 0 ; i < plainText.length() + 2 ; i+=2)

{

    a = newPair[i];

    b = newPair[i+1];



    int firstPos[] = findPosition(a, keyArr);

    int secondPos[] = findPosition(b, keyArr);



    if(firstPos[0] == secondPos[0]) // same row

    {

        cipherText.append(matrix[firstPos[0]][(firstPos[1] + 1) % 5]);

        cipherText.append(matrix[secondPos[0]][(secondPos[1] + 1) % 5]);

    }

    else if(firstPos[1] == secondPos[1]) // same column

    {

        cipherText.append(matrix[(firstPos[0] + 1) % 5][firstPos[1]]);

        cipherText.append(matrix[(secondPos[0] + 1) % 5][secondPos[1]]);

    }

    else

    {

        cipherText.append(matrix[firstPos[0]][secondPos[1]]);

        cipherText.append(matrix[secondPos[0]][firstPos[1]]);

    }

}

return cipherText.toString();

}
```

```
public static void main(String args[])
{
    String key = "THADOMAL";
    String plainText = "BALLOON";
    char keyArr[] = removeDuplicate(key);
    char res[][] = createMatrix(keyArr);
    char newPair[] = pairMaker(plainText);
    String encString = "";

    System.out.println("The Key Matrix is: ");
    System.out.println();
    for(int i = 0 ; i < 5 ; i++)
    {
        for(int j = 0 ; j < 5 ; j++)
        {
            System.out.print(res[i][j]+" ");
        }
        System.out.println();
    }
    System.out.println();
    System.out.println("The Plaintext after checking duplicates: ");
    for(char c : newPair)
        System.out.print(c+" ");
    System.out.println();
    encString = encryptPlayFair(plainText, keyArr);
    System.out.println("String after Encryption is: "+encString);
}
```

OUTPUT:

```
PS D:\DS\Practise> & 'C:\Program Files\Java\jdk-s\parth\AppData\Roaming\Code\User\workspaceStorage\playFair'
```

```
The Key Matrix is:
```

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

```
The Plaintext after checking duplicates:
```

```
B A L X L O X O N X
```

```
String after Encryption is: IBBWEHZAIZ
```

```
PS D:\DS\Practise>
```

EXPERIMENT NO: 4

AIM: Implementation of Euler's Totient Function.

THEORY: Euler's Totient function, $\phi(n)$, named after the renowned Swiss mathematician Leonhard Euler, is a fundamental arithmetic function in number theory. It plays a crucial role in various mathematical disciplines, including number theory, cryptography, and algorithm design. This function is designed to determine the count of positive integers less than or equal to a given positive integer n that are coprime (relatively prime) to n .

The primary objective of Euler's Totient function is to quantify the degree of coprimality of a positive integer n with respect to other integers. By computing $\phi(n)$, one can understand the number of positive integers less than or equal to n that do not share any common factors with n apart from 1. This information is valuable in several mathematical contexts, including prime factorization, modular arithmetic, and RSA encryption.

Euler's Totient function operates on the principle of coprimality, where two integers are considered coprime if their greatest common divisor (GCD) equals 1. The function $\phi(n)$ calculates the count of positive integers less than or equal to n that are coprime to n . It achieves this by leveraging the prime factorization of n . By examining the prime factors of n , $\phi(n)$ computes the count of integers that do not contain those factors, thus capturing the coprime nature of these numbers.

The Euler's Totient Function deals with three cases, each case handled with different formulas:

1. If the number is prime itself:

If the number is prime itself, the Euler's Totient Function can be given as:

$$\phi(n) = n - 1$$

2. If the number is not prime but factorizable:

If the number is not a prime number, factorize the number into its prime factors.

Suppose for a given number the factors are p and q then Euler's Totient is given as:

$$\phi(n) = \phi(p * q) = (p-1) * (q-1)$$

p must not be equal to q

3. If the factors are repeating:

If the prime factors for a given number are repeating then the following formula should be used:

$$\phi(p^k) = (p^k) - (p^{k-1})$$

Example:

P Euler Totient: $\phi(n)$

R1: $\phi(n) = n - 1$ if n is prime
 $\phi(n) = \phi(p) * \phi(q)$ if $p \neq q$ & p, q are prime

$$\phi(p^e) = p^e - p^{e-1}$$

(i) $\phi(17) = 17 - 1 = 16$

(ii) $\phi(35) = \phi(5) * \phi(7) \Rightarrow 4 * 6 = 24$

(iii) $\phi(240) = \phi(2^4) * \phi(3) * \phi(5) =$
 $= (2^4 - 2^3) * 2 * 4$
 $\Rightarrow 8 * 2 * 4 = \underline{\underline{64}}$

(iv) $\phi(144) = \phi(2^4) * \phi(3^2)$
 $= (2^4 - 2^3) * (3^2 - 3^1)$
 $= 8 * 6 = \underline{\underline{48}}$

2	240
2	120
2	60
2	30
3	15
5	1

2	144
2	72
2	36
2	18
3	9
3	3
1	1

```
import java.io.*;  
  
public class EulersTotient {  
  
    public static int eulerTotient(int n) {  
  
        int result = n;  
  
        for (int p = 2; p * p <= n; ++p) {  
  
            if (n % p == 0) {  
  
                while (n % p == 0)  
  
                    n /= p;  
  
                result -= result / p;  
  
            }  
  
        }  
  
        if (n > 1)  
  
            result -= result / n;  
  
        return result;  
    }  
  
    public static int gcd(int a, int b) {  
  
        if (b == 0)  
  
            return a;  
  
        return gcd(b, a % b);  
    }  
  
    public static void main(String[] args) throws IOException {  
  
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  
  
        System.out.println("Enter the number whose Euler's Totient is to be found out: ");  
  
        int num = Integer.parseInt(br.readLine());  
  
        int result = eulerTotient(num);  
  
        System.out.println("The Euler's Totient is: " + result);  
    }  
}
```

```
System.out.println("Relatively Prime numbers to "+num+" are: ");
for(int i = 0 ; i < num ; i++)
{
    if(gcd(num, i) == 1)
    {
        System.out.print(i+" ");
    }
}
}
```

OUTPUT:

```
PS D:\DS\Practise> d:; cd 'd:\DS\Practise'; & 'C:\Program F
nMessages' '-cp' 'C:\Users\parth\AppData\Roaming\Code\User\w
s\Practise_1b69d4b2\bin' 'EulersTotient'
Enter the number whose Euler's Totient is to be found out:
15
The Euler's Totient is: 8
Relatively Prime numbers to 15 are:
1 2 4 7 8 11 13 14
PS D:\DS\Practise> d:; cd 'd:\DS\Practise'; & 'C:\Program F
nMessages' '-cp' 'C:\Users\parth\AppData\Roaming\Code\User\w
s\Practise_1b69d4b2\bin' 'EulersTotient'
Enter the number whose Euler's Totient is to be found out:
48
The Euler's Totient is: 16
Relatively Prime numbers to 48 are:
1 5 7 11 13 17 19 23 25 29 31 35 37 41 43 47
PS D:\DS\Practise> 
```

EXPERIMENT NO: 5

AIM: Implement RSA cryptosystems.

THEORY:

The RSA (Rivest-Shamir-Adleman) cryptosystem is one of the most widely used asymmetric encryption algorithms in modern cryptography. It was invented in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman. RSA is based on the computational difficulty of factoring large prime numbers, which forms the basis of its security.

The RSA algorithm is primarily used for achieving the security goals of confidentiality and integrity in cryptographic systems. While it can contribute to authentication and non-repudiation in certain contexts, these goals are typically addressed using additional cryptographic mechanisms when using RSA.

1. Confidentiality:

RSA achieves confidentiality through asymmetric encryption. The public key, used for encryption, is widely distributed, while the private key, used for decryption, is kept secret.

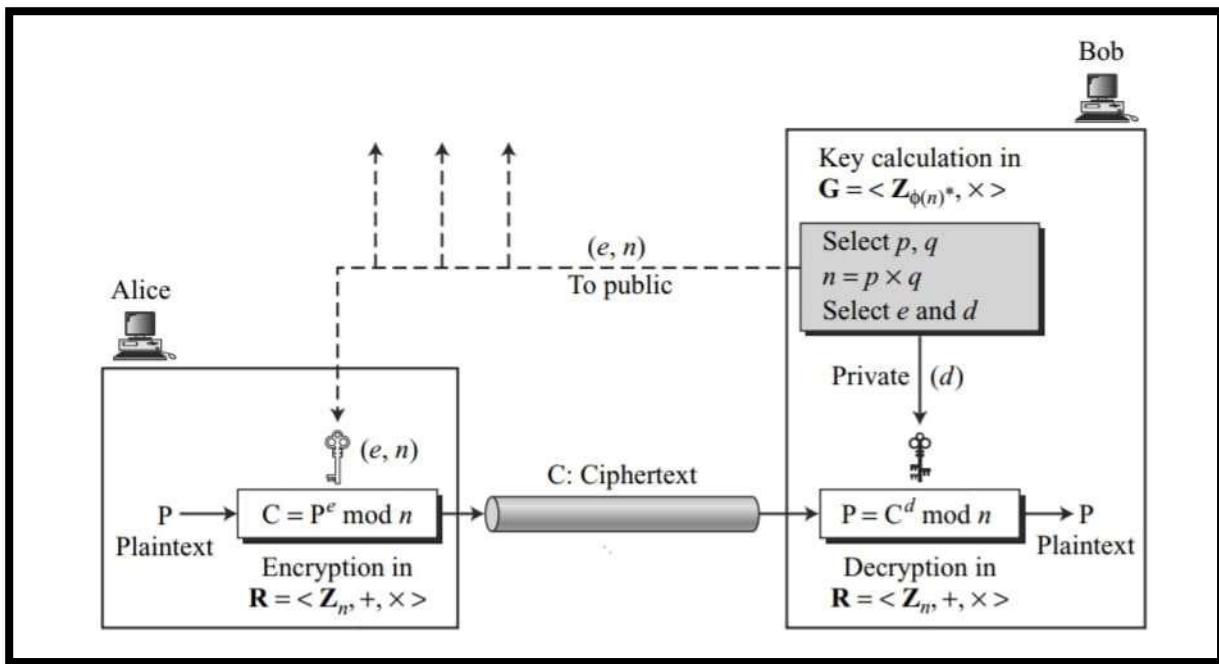
2. Integrity:

RSA alone does not directly address integrity. However, it is commonly used in combination with cryptographic hashing algorithms and digital signatures to achieve integrity.

3. Non-repudiation:

Non-repudiation ensures that a sender cannot deny sending a message and that a recipient cannot deny receiving it.

RSA-based digital signatures can provide non-repudiation by binding the sender's identity to the message.



ALGORITHM:

1. Key Generation:

- Choose distinct prime numbers p and q .
- Compute $n = p * q$.
- Compute $\phi(n) = (p-1) * (q-1)$.
- Choose an integer e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$.
- Compute the modular multiplicative inverse d of e modulo $\phi(n)$, such that $d * e \equiv 1 \pmod{\phi(n)}$.

2. Encryption:

- Convert the plaintext message M into an integer m such that $0 \leq m < n$.
- Compute the ciphertext $c \equiv m^e \pmod{n}$.

3. Decryption:

- Decrypt the ciphertext c using the private key d to recover the plaintext $m \equiv c^d \pmod{n}$.

EXAMPLE:

RSA Algo: ① key Generation by Alice

select p, q $p \text{ and } q \text{ both prime}$

calc. $n = p \times q$

calc. $\phi(n) = (p-1)(q-1)$

Select integer e $\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$

calc. $d = e^{-1} \pmod{\phi(n)}$

Public key (n, e) $d = e^{-1} \pmod{\phi(n)}$

Private key (n, d) $PR = (d, n)$

Encryption by bob with Alices Publickey

PT: $M < n$

CT: $C = M^e \pmod{n}$

Decryption by Alice with Alice's Public key

CT: C

PT: $M = C^d \pmod{n}$

② Perform Encryption and Decryption using RSA algo:
where your message $m = 85$, where p and q are 17 and 11

Step 1: Key generation algo:
i) $p \& q$ should be prime and $p \neq q$

Step 2: $n = 17 \times 11 = 187$

Step 3: $\phi(n) = (p-1)(q-1) = 160$

Step 4: e $\gcd(e, \phi(n)) = 1$

$$\begin{array}{r} q \\ 53 \\ 3 \end{array} \quad \begin{array}{r} r_1 \\ 160 \\ 3 \end{array} \quad \begin{array}{r} r_2 \\ 3 \\ 1 \end{array} \quad \begin{array}{r} r \\ 1 \\ 0 \end{array} \quad \therefore [e = 3]$$

$\xrightarrow{\text{gcd}}$

Step 5: $d + e = 1 \pmod{\phi(n)}$
 $d * 3 = 1 \pmod{160}$

$t = 1 - 3(-53)$

$r_1 r_2 t_1 t_2$
 $53 160 3 1 0 1 -53$

$-53 + 160 = 107 \therefore d = 107$
 $107 * 3 = 321 \pmod{160}$
 $= 1$

Public key = $\{3, 167\}$
Private key = $\{107, 167\}$

I Encryption

$C^T = M^e \pmod{n}$
 $= 88^3 \pmod{167} = 88^3 \% 167$
 $= 44$

II Decryption

$P^T = C^d \pmod{n}$
 $P^T = 44^{107} \pmod{167}$ - 44^{107} is a very big value
we will take $e=7$.

Encr: $C^T = 88^7 \pmod{167}$
 $= ((88^6 \% 167) * (88^2 \% 167) * (88^4 \% 167)) \% 167$

$C^T = \underline{\underline{11}}$

Dec: $11^{23} \pmod{167}$ $11^{23} = 11^1 * 11^2 * 11^4 * 11^{16}$
 $(11^6 \% 167) = 11$ $(11 * 11) \pmod{167} = 55$
 $(11^4 \% 167) = 121$ $(55 * 55 \% 167) \pmod{167} = \underline{\underline{154}}$
 $11^6 = 11^4 * 11^2 * 11^4$

$(11273570 \% 167) = \underline{\underline{88}}$

```
import java.io.*;
import java.util.*;

class RSA
{
    static boolean isPrime(int x)
    {
        for(int i = 2 ; i < x ; i++)
        {
            if(x%i == 0)
                return false;
        }
        return true;
    }

    static int gcd(int a, int b)
    {
        if(b == 0)
            return a;
        return gcd(b, a%b);
    }

    static void gcdTablePrinter(int phi, int e)
    {
        int q = 0,r1 = 0,r2 = 0,r = 1;
        r1 = phi;
        r2 = e;
        System.out.println("q\tr1\tr2\tr");
        while(r!=0)
```

```
{  
    q = r1/r2;  
    r = r1%r2;  
    System.out.println(q+"\t"+r1+"\t"+r2+"\t"+r);  
    r1 = r2;  
    r2 = r;  
}  
  
System.out.println(q+"\t"+r1+"\t"+r2+"\t"+r);  
}  
  
  
static int inverseFinder(int phi, int e)  
{  
    int q=0,r1,r2,r=1,t=0,t1=0,t2=1;  
    r1 = phi;  
    r2 = e;  
    System.out.println("q\tr1\tr2\tr\tr1\tr2\tr");  
    while(r != 0)  
    {  
        q = r1/r2;  
        r = r1%r2;  
        t = t1 - q*t2;  
        System.out.println(q+"\t"+r1+"\t"+r2+"\t"+r+"\t"+t1+"\t"+t2+"\t"+t);  
        r1 = r2;  
        r2 = r;  
        t1 = t2;  
        t2 = t;  
    }  
    System.out.println(q+"\t"+r1+"\t"+r2+"\t"+r+"\t"+t1+"\t"+t2+"\t"+t);  
    if(t1 < 0)  
}
```

```
{  
    return (t1 + phi);  
}  
return t1;  
}  
  
static int power(int base, int exponent)  
{  
    int result = 1;  
    for(int i = 0 ; i < exponent; i++)  
    {  
        result = result*base;  
    }  
    return result;  
}  
  
static int RSAencyrpt(int m, int e, int n)  
{  
    int temp = power(m, e);  
    int result = temp % n;  
    return result;  
}  
static int RSAdecrypt(int c, int d, int n)  
{  
    int temp = power(c,d);  
    int result = temp%n;  
    return result;  
}  
public static void main(String args[]) throws IOException
```

```
{  
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  
  
    System.out.println("Enter p, q and Message to be encrypted: ");  
  
    ArrayList<Integer> arr = new ArrayList<Integer>(3);  
  
    for(int i = 0; i < 3; i++)  
  
    {  
  
        int num = Integer.parseInt(br.readLine());  
  
        arr.add(num);  
  
    }  
  
    if(isPrime(arr.get(0)) == false || isPrime(arr.get(1)) == false || arr.get(0) == arr.get(1))  
  
        System.out.println("RSA encryption is not possible");  
  
    else  
  
    {  
  
        int n = arr.get(0)* arr.get(1);  
  
        int phi = (arr.get(0) - 1)*(arr.get(1) - 1);  
  
        System.out.println("n = "+ arr.get(0)+" * "+arr.get(1)+" = "+n);  
  
        System.out.println("phi(n) = "+ (arr.get(0) - 1) +" * "+(arr.get(1) - 1) +" = "+phi);  
  
        int e = 2;  
  
        System.out.println("GCD Table: ");  
  
        while(e < phi)  
  
        {  
  
            if(gcd(e, phi) == 1)  
  
                break;  
  
            else  
  
                e++;  
  
        }  
  
        gcdTablePrinter(phi, e);  
  
        System.out.println("The Inverse Table: ");  
  
        int d = inverseFinder(phi, e);  
    }
```

```
System.out.println("The public key is: ("+e+","+n+")");  
System.out.println("The private key is: ("+d+","+n+")");  
int cipher = RSAencyrpt(arr.get(2), e, n);  
int pt = RSAdecrypt(cipher, d, n);  
System.out.println("After Encyrption: "+cipher);  
System.out.println("After Decyрption: "+pt);  
}  
}  
}
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\DS\Practise> d:; cd 'd:\DS\Practise'; & 'C:\ProgramsInExceptionMessages' '-cp' 'C:\Users\parth\AppData\Roaming\redhat.java\jdt_ws\Practise_1b69d4b2\bin' 'RSA'  
Enter p, q and Message to be encrypted:  
11  
3  
2  
n = 11 * 3 = 33  
phi(n) = 10 * 2 = 20  
GCD Table:  


| q | r1 | r2 | r |
|---|----|----|---|
| 6 | 20 | 3  | 2 |
| 1 | 3  | 2  | 1 |
| 2 | 2  | 1  | 0 |
| 2 | 1  | 0  | 0 |

  
The Inverse Table:  


| q | r1 | r2 | r | t1 | t2  | t   |
|---|----|----|---|----|-----|-----|
| 6 | 20 | 3  | 2 | 0  | 1   | -6  |
| 1 | 3  | 2  | 1 | 1  | -6  | 7   |
| 2 | 2  | 1  | 0 | -6 | 7   | -20 |
| 2 | 1  | 0  | 0 | 7  | -20 | -20 |

  
The public key is: (3,33)  
The private key is: (7,33)  
After Encyrption: 8  
After Decyрption: 2  
PS D:\DS\Practise>
```

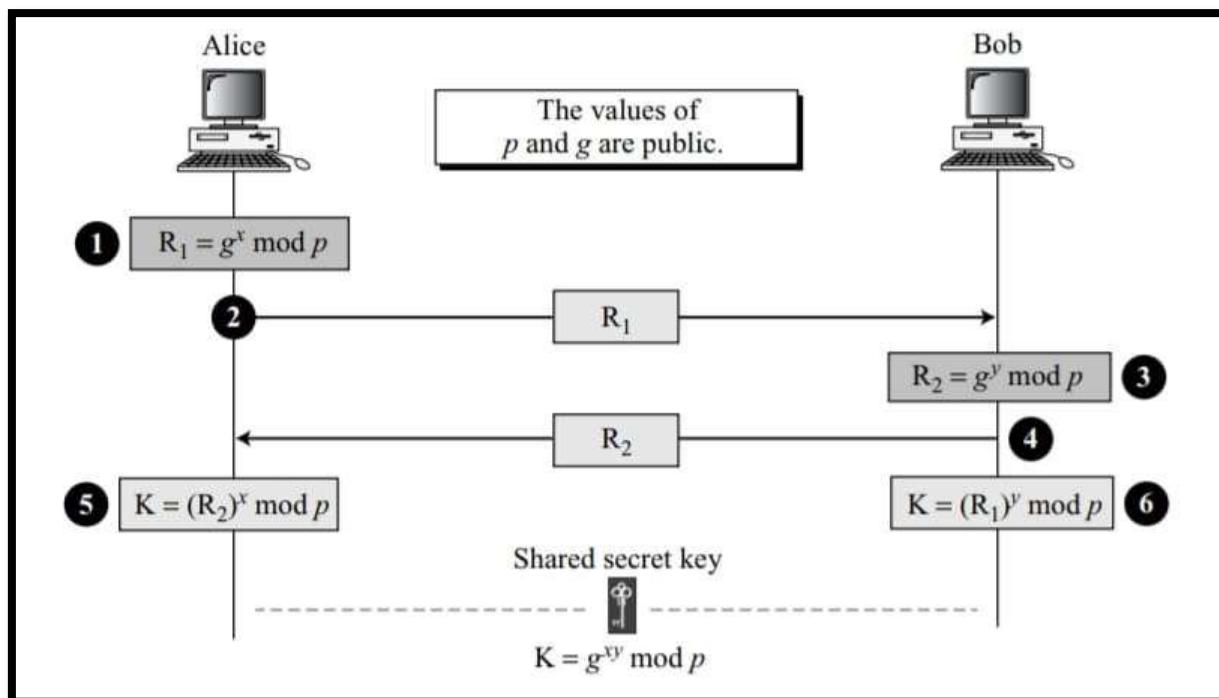

EXPERIMENT NO: 6

AIM : Implement Diffie Hellman Key Exchange Algorithm.

THEORY :

The Diffie-Hellman key exchange algorithm is a fundamental method used in cryptography to securely exchange cryptographic keys over a public channel. It was developed independently by Whitfield Diffie and Martin Hellman in 1976 and is one of the first public-key protocols. The beauty of the Diffie-Hellman algorithm lies in its ability to establish a shared secret key between two parties without the need for any prior communication or shared secret.

In the Diffie-Hellman protocol two parties create a symmetric session key without the need of a KDC. Before establishing a symmetric key, the two parties need to choose two numbers p and g . The first number, p , is a large prime number on the order of 300 decimal digits (1024 bits). The second number, g , is a generator of order $p - 1$ in the group $\langle Z_p^*, \times \rangle$. These two (group and generator) do not need to be confidential. They can be sent through the Internet; they can be public.



Diagrammatic Representation of Diffie Hellman Key Exchange Algorithm

ALGORITHM:

The steps are as follows:

1. Alice chooses a large random number x such that $0 \leq x \leq p - 1$ and calculates

$$R_1 = g^x \bmod p.$$

2. Bob chooses another large random number y such that $0 \leq y \leq p - 1$ and calculates

$$R_2 = g^y \bmod p.$$

3. Alice sends R_1 to Bob. Note that Alice does not send the value of x ; she sends only R_1 .

4. Bob sends R_2 to Alice. Again, note that Bob does not send the value of y , he sends only R_2 .

5. Alice calculates $K = (R_2)^x \bmod p.$

6. Bob also calculates $K = (R_1)^y \bmod p.$

The value g must be a cyclic generator. For example: if $p = 5$ then the value of g will be 2

$$2^1 \bmod 5 = 2$$

$$2^2 \bmod 5 = 4$$

$$2^3 \bmod 5 = 3$$

$$2^4 \bmod 5 = 1$$

$$P = 7$$

for A, $1 \leq x \leq 6$
 consider $x = 2$

for B, $1 \leq y \leq 6$
 consider $y = 3$

cyclic generator of 7:

$2^1 \bmod 7 = 2$	$3^1 \bmod 7 = 3$
$2^2 \bmod 7 = 4$	$3^2 \bmod 7 = 2$
$2^3 \bmod 7 = 1$	$3^3 \bmod 7 = 6$
$2^4 \bmod 7 = 2$	$3^4 \bmod 7 = 4$
$2^5 \bmod 7 = 4$	$3^5 \bmod 7 = 5$
$2^6 \bmod 7 = 1$	$3^6 \bmod 7 = 1$

$\therefore 3$ is cyclic generator

$$R_1 = 3^2 \bmod 7 = 2$$

$$R_2 = 3^3 \bmod 7 = 6$$

key for A = $6^2 \bmod 7 = 1$
key for B = $2^3 \bmod 7 = 1$

```
import java.util.*;

public class DiffieHellman {

    static int gcd = -1;

    static int mod_inv = -1;

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter p: ");
        int p = sc.nextInt();

        if(!isPrime(p)){
            System.out.println("It is not prime");
            System.exit(0);
        }

        System.out.println("It is prime");

        System.out.println("Enter the value is x in range(1, "+(p-1)+"): ");
        int x = sc.nextInt();

        if(x<=0 || x>=p){
            System.out.println("Not allowed");
            System.exit(0);
        }

        System.out.println("Enter the value is y in range(1, "+(p-1)+"): ");
        int y = sc.nextInt();

        if(y<=0 || y>=p){
            System.out.println("Not allowed");
            System.exit(0);
        }
    }
}
```

}

```
System.out.println("Generator are: ");
int g = getGenerator(p);
System.out.println("Generator is: "+g);

for(int i=1; i<p; i++){
    System.out.println(g+"i mod "+p+" = "+power(g, i, p));
}

int k1 = power(g, x, p);
int k2 = power(g, y, p);

System.out.println("Key 1 is : "+k1);
System.out.println("Key 2 is: "+k2);

int final_key1 = power(k1, y, p);
int final_key2 = power(k2, x, p);

System.out.println("Keys are: "+final_key1+", "+final_key2);
}

public static int power(int a, int b, int mod) {
    int result = 1;
    int count = 0;
    int x = a;
    while (b > 0) {
        // If the current bit of b is set, multiply the result by a
        if ((b & 1) == 1)
            result = (result * a) % mod;
    }
}
```

```
//if((b&1) == 1)
    //System.out.println("we calculate "+ x + "^" + (1<<count));
    // Square the value of a and reduce it modulo mod
    a = (a * a) % mod;

    // Right shift b to move to the next bit
    b >>= 1;
    count++;
}

return result;
}

static void recur(int r1, int r2, int s1, int s2, int t1, int t2){

    int q = r1/r2;
    int r = r1%r2;

    int s = s1 - q*s2;
    int t = t1 - q*t2;

    //System.out.println(q+ " +r1+" +r2+" +r+" "+s1+" "+s2+" "+s+" "+t1+" "+t2+" "+t+" ");
    if(r>0)
        recur(r2, r, s2, s, t2, t);
    else{
        //System.out.println("equation is given by: ax+by\nWhere,");
        //System.out.println("x = "+s2+"\ny=" +t2);
        gcd = r2;
        mod_inv = t2;
    }
}

static boolean isPrime(int a){
```

```
for(int i=2; i<a; i++)  
    if(a%i==0)  
        return false;  
  
    return true;  
}  
  
  
static int getGenerator(int p){  
    int ans = -1;  
    for(int i=2; i<p; i++){  
        Set<Integer> set = new HashSet<Integer>();  
        for(int j=1; j<p; j++){  
            set.add(power(i, j, p));  
        }  
  
        if(set.size()==p-1){  
            System.out.print(i+" ");  
            ans = i;  
        }  
    }  
    System.out.println("");  
  
    return ans;  
}
```

The screenshot shows a terminal window with the following tabs at the top: PROBLEMS (3), OUTPUT, DEBUG CONSOLE, TERMINAL (underlined), and PORTS. The terminal content is as follows:

```
Enter p:  
17  
It is prime  
Enter the value is x in range(1, 16):  
2  
Enter the value is y in range(1, 16):  
3  
Generator are:  
3 5 6 7 10 11 12 14  
Generator is: 14  
14^1 mod 17 = 14  
14^2 mod 17 = 9  
14^3 mod 17 = 7  
14^4 mod 17 = 13  
14^5 mod 17 = 12  
14^6 mod 17 = 15  
14^7 mod 17 = 6  
14^8 mod 17 = 16  
14^9 mod 17 = 3  
14^10 mod 17 = 8  
14^11 mod 17 = 10  
14^12 mod 17 = 4  
14^13 mod 17 = 5  
14^14 mod 17 = 2  
14^15 mod 17 = 11  
14^16 mod 17 = 1  
Key 1 is : 9  
Key 2 is: 7  
Keys are: 15, 15  
PS D:\DS\Practise>
```

Experiment – 6

Aim: Study of various Reconnaissance Tools like WHOIS, traceroute, dig, nslookup to gather information about networks and domain registrars.

Theory:

Whois

Command: whois tsec.edu

Explanation:

The `whois` command sends a query to a WHOIS server, which is responsible for maintaining and providing access to domain registration information. The server returns information such as:

1. Registrar Information: This includes the name of the organization that registered the domain, their contact information, and possibly their website.
2. Domain Registration Details: This includes the date when the domain was registered and the date when it will expire.
3. Registrant Information: This may include the name, address, email, and phone number of the person or organization that registered the domain.
4. Name Servers: These are the authoritative servers responsible for resolving the domain to its associated IP address.
5. Domain Status: This indicates whether the domain is active, on hold, pending transfer, or has expired.
6. DNS Records: Some WHOIS servers provide information about the DNS records associated with the domain, such as A records, MX records, and NS records.

Screenshot:

```
admin@admini-OptiPlex-3050:~/Desktop$ whois tsec.edu
This Registry database contains ONLY .EDU domains.
The data in the EDUCAUSE Whois database is provided
by EDUCAUSE for information purposes in order to
assist in the process of obtaining information about
or related to .edu domain registration records.

The EDUCAUSE Whois database is authoritative for the
.edu domain.

A Web interface for the .EDU EDUCAUSE Whois Server is
available at: http://whois.educause.edu

By submitting a Whois query, you agree that this information
will not be used to allow, enable, or otherwise support
the transmission of unsolicited commercial advertising or
solicitations via e-mail. The use of electronic processes to
harvest information from this server is generally prohibited
except as reasonably necessary to register or modify .edu
domain names.

-----
Domain Name: TSEC.EDU

Registrant:
    Thadomal Sahani Engineering College
    P.G Kher Marg, Bandra(W)
    Mumbai, Maharashtra 400 050
    India

Administrative Contact:
    Dr. Gopakumaran Thampi
    Thadomal Shahani Engineering College
    Nari Gurshahani Marg, Bandra(W)
    Mumbai, 400050
    India
    +91.2226495808
```

Dig

The dig command is a versatile tool for querying DNS (Domain Name System) servers to retrieve information about domain names, such as IP addresses, name servers, and DNS records.

Command: dig www.google.com

Explanation:

1. DNS Resolution: The primary purpose of using `dig` is to resolve domain names to IP addresses. For example, when you type `www.google.com` in a web browser, your computer needs to know the corresponding IP address to establish a connection to Google's servers. `dig` helps you understand this process by showing you the IP address associated with the domain.
2. Name Servers: DNS servers store information about domain names and their corresponding IP addresses. When you run `dig`, it also provides information about the authoritative name servers responsible for the domain. These name servers hold the official records for the domain and can provide additional details if needed.
3. Additional Records: Apart from the IP address, `dig` can also retrieve other DNS records associated with the domain, such as MX (Mail Exchange) records for email servers, TXT records for text information like SPF (Sender Policy Framework) records used for email authentication, and more. These additional records can be crucial for understanding how a domain is configured and used.
4. Debugging and Troubleshooting: `dig` is a valuable tool for network administrators and developers for debugging and troubleshooting DNS-related issues. By inspecting the output of `dig`, you can identify DNS misconfigurations, check DNS propagation status, and diagnose connectivity problems.

Screenshot:

```
; <>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 10306
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;www.google.com.           IN      A

;; ANSWER SECTION:
www.google.com.        140     IN      A      142.250.199.132

;; Query time: 35 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Tue Mar 19 11:21:01 IST 2024
;; MSG SIZE rcvd: 59
```

Command: dig google.com +noquestion

Explanation:

In this command, google.com is the domain name being queried. The +noquestion option is used to suppress the display of the question section in the output.

When you run this command, it sends a DNS query to a DNS server to retrieve information about the domain, such as its IP address, authoritative name servers, and other DNS records. By using the +noquestion option, the output will not display the original query that was sent to the DNS server.

Screenshot:

```
; <>> DiG 9.18.18-0ubuntu0.22.04.2-Ubuntu <>> google.com +noquestion
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6635
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;;
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 65494
;; ANSWER SECTION:
google.com.          262      IN      A       142.251.42.46
;;
;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Wed Mar 20 13:31:06 IST 2024
;; MSG SIZE  rcvd: 55
```

Command: dig google.com +nocomments

Explanation:

In this command, google.com is the domain name being queried. By using the +nocomments option, the output will not display any comments that might be included in the DNS server's response.

Screenshot:

```
admini@admini-OptiPlex-3050:~/Desktop$ dig google.com +nocomments
; <>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <>> google.com +nocomments
;; global options: +cmd
;google.com.           IN      A
google.com.          187     IN      A      172.217.167.174
;; Query time: 28 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Tue Mar 19 11:31:59 IST 2024
;; MSG SIZE  rcvd: 55
```

Command:

dig google.com +noauthority

Explanation:

In this command, google.com is the domain name being queried. The +noauthority option is employed to suppress the display of authority section in the output.

Screenshot:

```
admini@admini-OptiPlex-3050:~/Desktop$ dig google.com +noauthority
; <>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <>> google.com +noauthority
;; global options: +cmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 20274
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;google.com.           IN      A

;; ANSWER SECTION:
google.com.          27     IN      A      172.217.167.174

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Tue Mar 19 11:34:39 IST 2024
;; MSG SIZE  rcvd: 55
```

Command: dig google.com +noadditional

Explanation:

When this command is executed, it sends a DNS query to a DNS server to obtain details about the domain, such as its IP address, authoritative name servers, and additional DNS records. By utilizing the +noadditional option, the output will not include any additional information beyond the essential response to the query.

Screenshot:

```
admin@admini-OptiPlex-3050:~/Desktop$ dig google.com +noadditional
; <>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <>> google.com +noadditional
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64277
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;;
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;google.com.           IN      A
;;
;; ANSWER SECTION:
google.com.        172     IN      A      172.217.167.174
;;
;; Query time: 96 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Tue Mar 19 11:36:44 IST 2024
;; MSG SIZE  rcvd: 55
```

Command:

```
dig google.com +nostats
```

Explanation:

By utilizing the +nostats option, the output will not include any statistical information regarding the query execution.

Screenshot:

```
admini@admini-OptiPlex-3050:~/Desktop$ dig google.com +nostats
; <>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <>> google.com +nostats
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29741
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;;
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;google.com.           IN      A
;;
;; ANSWER SECTION:
google.com.        92      IN      A      172.217.167.174
```

Command:

Dig google.com +noanswer

Explanation:

By using the +noanswer option, the output will not include the actual DNS records associated with the domain. This can be useful when you only want to see metadata about the query without the actual response.

Screenshot:

```
admini@admini-OptiPlex-3050:~/Desktop$ dig google.com +noanswer
; <>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <>> google.com +noanswer
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62281
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;;
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;google.com.           IN      A
;;
;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Tue Mar 19 11:38:57 IST 2024
;; MSG SIZE  rcvd: 55
```

Command:

Dig google.com MX +noall +answer

Explanation:

In this command, google.com is the domain name being queried. The MX option specifies that only mail exchange (MX) records should be queried. The +noall option is used to suppress all output sections except the answer section. Finally, the +answer option is used to display only the answer section in the output.

When you run this command, it sends a DNS query to a DNS server to retrieve the MX records associated with the domain google.com. By using the +noall and +answer options, the output will only include the MX records found in the answer section, excluding any other sections such as the question, authority, and additional sections.

Screenshot:

```
admin@admini-OptiPlex-3050:/Desktop$ dig google.com MX +noall +answer
google.com.          264     IN      MX      10 smtp.google.com.
```

Command:

Dig google.com NS +noall +answer

Explanation:

The NS option specifies that only name server (NS) records should be queried. The +noall option is used to suppress all output sections except the answer section. Finally, the +answer option is used to display only the answer section in the output.

Screenshot:

```
admini@admini-OptiPlex-3050:~/Desktop$ dig google.com NS +noall +answer
google.com.          104603  IN      NS      ns4.google.com.
google.com.          104603  IN      NS      ns2.google.com.
google.com.          104603  IN      NS      ns3.google.com.
google.com.          104603  IN      NS      ns1.google.com.
```

Command:

```
dig -t ANY google.com +noall +answer
```

Explanation:

The command utilizes the -t ANY option to specify that all types of DNS records (ANY) should be queried for the domain google.com. The +noall option is used to suppress all output sections except the answer section, and the +answer option is used to display only the answer section in the output.

When you execute this command, it sends a DNS query to a DNS server to retrieve all types of DNS records associated with the domain google.com, including A, AAAA, MX, NS, and others. By using the +noall and +answer options, the output will only include the DNS records found in the answer section, excluding any other sections such as the question, authority, and additional sections.

Screenshot:

```
admini@admini-OptiPlex-3050:~/Desktop$ dig -t ANY google.com +noall +answer
google.com.          200    IN      A       142.250.67.238
google.com.          236    IN      AAAA    2404:6800:4009:814::200e
google.com.          214    IN      MX      10 smtp.google.com.
google.com.          59     IN      SOA     ns1.google.com. dns-admin.google.com. 616768135 900 900 1800 60
google.com.          84407   IN      NS      ns2.google.com.
google.com.          84407   IN      NS      ns4.google.com.
google.com.          84407   IN      NS      ns3.google.com.
google.com.          84407   IN      NS      ns1.google.com.
```

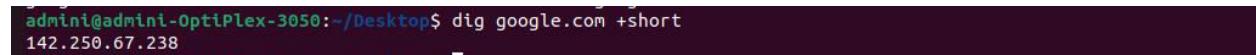
Command:

```
dig google.com +short
```

Explanation:

The `+short` option is used to display only the IP addresses in a compact format, without additional information such as DNS record types or server responses.

Screenshot:



```
admini@admini-OptiPlex-3050:~/Desktop$ dig google.com +short
142.250.67.238
```

Command:

```
Dig -x 209.132.183.81
```

Explanation:

The `Dig` command uses the `-x` option to perform a reverse DNS lookup, also known as a PTR (Pointer) query.

Reverse DNS lookups are often used to verify the identity of an IP address, find the domain name associated with an IP, or perform troubleshooting tasks. In this case, the command will return the domain name (if any) associated with the IP address 209.132.183.81.

Screenshot:

```
admin@admini-OptiPlex-3050:~/Desktop$ dig x 209.132.183.81

; <>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <>> x 209.132.183.81
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 27298
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;x.           IN      A

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Tue Mar 19 11:49:18 IST 2024
;; MSG SIZE rcvd: 30

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34279
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;209.132.183.81.          IN      A

;; ANSWER SECTION:
209.132.183.81.      0      IN      A      209.132.183.81

;; Query time: 63 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Tue Mar 19 11:49:18 IST 2024
;; MSG SIZE rcvd: 59
```

Traceroute

The **traceroute** command is a network diagnostic tool used to trace the route that packets take from your computer to a specified destination, typically a domain name or IP address. Here's how it works:

1. **Sending Packets:** When you execute the **traceroute** command followed by a destination (e.g., **google.com**), your computer sends out a series of packets towards that destination.
2. **Incrementing TTL:** Each packet has a Time To Live (TTL) value set initially to 1. When the first packet is sent, it reaches the first router along the path, where the TTL is decremented to 0. The router then discards the packet and sends back an ICMP (Internet Control Message Protocol) Time Exceeded message to your computer.
3. **Determining Hops:** Your computer receives the ICMP Time Exceeded message, indicating the first hop along the route to the destination. It then sends another packet with a TTL of 2, which reaches the second router along the path. This

process continues, incrementing the TTL for each subsequent packet, until the destination is reached or the maximum number of hops is reached.

4. **Displaying Results:** **traceroute** displays the list of routers (or "hops") along the path to the destination. It shows the IP addresses of the routers, along with the round-trip time (RTT) for each hop, indicating how long it took for the packets to reach that router and receive a response.
5. **Identifying Network Issues:** Traceroute is commonly used for diagnosing network connectivity issues. By examining the list of routers and RTT values, network administrators can identify potential bottlenecks, routing problems, or points of failure along the path to the destination.
6. **Different Protocols:** Depending on the operating system and version of **traceroute** being used, it may employ different protocols such as ICMP, UDP, or TCP to send the packets. Each protocol has its advantages and limitations, and may be chosen based on network configurations and security policies.

Command:

```
traceroute google.com
```

Explanation:

The traceroute command is used to trace the route that packets take from your computer to a specified destination, such as a domain name or IP address. In this command, google.com is the destination domain name.

When you run this command, your computer sends out a series of packets with increasing TTL (Time To Live) values towards the destination. Each router along the path decrements the TTL value, and if it reaches zero, the router discards the packet and sends back an ICMP (Internet Control Message Protocol) Time Exceeded message. By receiving these messages, traceroute can determine the routers along the path to the destination.

Screenshot:

```
traceroute to google.com (142.251.42.46), 30 hops max, 60 byte packets
1 _gateway (192.168.32.1)  0.872 ms  0.812 ms  0.768 ms
2 192.168.34.1 (192.168.34.1)  2.458 ms  2.415 ms  2.373 ms
3 203.212.25.1 (203.212.25.1)  2.331 ms  2.291 ms  2.250 ms
4 203.212.24.53 (203.212.24.53)  3.495 ms  3.456 ms  3.414 ms
5 175.100.177.53 (175.100.177.53)  5.380 ms  5.338 ms  5.188 ms
6 172.16.2.202 (172.16.2.202)  5.869 ms  *
7 175.100.188.22 (175.100.188.22)  4.565 ms  4.608 ms  4.557 ms
8  *
9 142.251.64.14 (142.251.64.14)  7.756 ms 142.251.64.8 (142.251.64.8)  4.934 ms  4.897 ms
10 142.251.69.43 (142.251.69.43)  4.288 ms 142.251.69.45 (142.251.69.45)  9.666 ms  5.347 ms
11 bom12s20-in-f14.1e100.net (142.251.42.46)  5.290 ms  5.247 ms 192.178.110.199 (192.178.110.199)  5.964 ms
```

NsLookup

The **nslookup** command-line tool is used to query Domain Name System (DNS) servers to obtain domain name or IP address information. It stands for "name server lookup". Here's how it works:

- Querying DNS Servers:** When you run **nslookup** followed by a domain name or an IP address, the command queries the DNS servers configured on your system to resolve the domain name or perform a reverse DNS lookup for the IP address.
- Obtaining DNS Records:** **nslookup** can retrieve various types of DNS records associated with a domain name, including A (IPv4 address), AAAA (IPv6 address), MX (mail exchange), NS (name server), CNAME (canonical name), and TXT (text) records, among others.
- Interactive Mode:** **nslookup** can also be used in interactive mode, where you can enter multiple queries without having to invoke the command multiple times. In interactive mode, you can specify the type of record you want to query and choose the DNS server to query.
- Diagnosing DNS Issues:** **nslookup** is a valuable tool for diagnosing DNS-related issues such as DNS resolution failures, misconfigurations, and DNS server connectivity problems. By querying different DNS servers and examining the responses, network administrators can troubleshoot and resolve DNS issues efficiently.

Command:

nslookup google.com

Explanation:

When you execute this command, your system sends a DNS query to the default DNS server configured on your system (or the one specified in your network settings). The DNS server then responds with information about the domain **google.com**, such as its IP address(es), authoritative name servers, and other DNS records.

Screenshot:

```
admini@admini-OptiPlex-3050:~/Desktop$ nslookup google.com
Server:      127.0.0.53
Address:     127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 172.217.167.174
Name:   google.com
Address: 2404:6800:4009:810::200e
```

Command:

```
nslookup -query = mx google.com
```

Explanation:

The **nslookup** command with the **-query=mx** option is used to specifically query the mail exchange (MX) records for the domain **google.com**.

In this command:

- **-query=mx** specifies that the query type is MX records, which are used to identify the mail servers responsible for receiving email messages for the domain.
- **google.com** is the domain name being queried.

When you execute this command, your system sends a DNS query to the default DNS server configured on your system (or the one specified in your network settings) to retrieve the MX records associated with the domain **google.com**. The DNS server then responds with information about the mail servers configured for handling email for the domain.

Screenshot:

```
Server:      127.0.0.53
Address:     127.0.0.53#53

Non-authoritative answer:
google.com      mail exchanger = 10 smtp.google.com.

Authoritative answers can be found from:
smtp.google.com internet address = 142.251.175.27
smtp.google.com internet address = 74.125.24.26
smtp.google.com internet address = 74.125.24.27
smtp.google.com internet address = 142.251.10.27
smtp.google.com internet address = 142.251.175.26
smtp.google.com has AAAA address 2404:6800:4003:c1c::1a
smtp.google.com has AAAA address 2404:6800:4003:c03::1b
smtp.google.com has AAAA address 2404:6800:4003:c03::1a
smtp.google.com has AAAA address 2404:6800:4003:c1c::1b
```

Command:

```
nslookup -type = ns google.com
```

Explanation:

The nslookup command with the -type=ns option is used to specifically query the name server (NS) records for the domain google.com.

In this command:

-type=ns specifies that the query type is NS records, which are used to identify the authoritative name servers for the domain.

google.com is the domain name being queried.

Screenshot:

```
Server:      127.0.0.53
Address:     127.0.0.53#53

Non-authoritative answer:
google.com      nameserver = ns1.google.com.
google.com      nameserver = ns3.google.com.
google.com      nameserver = ns2.google.com.
google.com      nameserver = ns4.google.com.

Authoritative answers can be found from:
ns1.google.com  internet address = 216.239.32.10
ns1.google.com  has AAAA address 2001:4860:4802:32::a
ns3.google.com  internet address = 216.239.36.10
ns3.google.com  has AAAA address 2001:4860:4802:36::a
ns2.google.com  internet address = 216.239.34.10
ns2.google.com  has AAAA address 2001:4860:4802:34::a
ns4.google.com  internet address = 216.239.38.10
ns4.google.com  has AAAA address 2001:4860:4802:38::a
```

Command:

```
#nslookup -type = soa google.com
```

Explanation:

-type=soa specifies that the query type is SOA record, which is used to identify the authoritative name server for the domain and contains various administrative information about the zone.

Screenshot:

```
Server:      127.0.0.53
Address:     127.0.0.53#53

Non-authoritative answer:
google.com
    origin = ns1.google.com
    mail addr = dns-admin.google.com
    serial = 616561579
    refresh = 900
    retry = 900
    expire = 1800
    minimum = 60

Authoritative answers can be found from:
google.com      nameserver = ns2.google.com.
google.com      nameserver = ns4.google.com.
google.com      nameserver = ns1.google.com.
google.com      nameserver = ns3.google.com.
ns3.google.com  internet address = 216.239.36.10
ns3.google.com  has AAAA address 2001:4860:4802:36::a
ns2.google.com  internet address = 216.239.34.10
ns2.google.com  has AAAA address 2001:4860:4802:34::a
ns1.google.com  internet address = 216.239.32.10
ns1.google.com  has AAAA address 2001:4860:4802:32::a
ns4.google.com  internet address = 216.239.38.10
ns4.google.com  has AAAA address 2001:4860:4802:38::a
```

Command:

```
nslookup -type = any google.com
```

Explanation:

The nslookup command with the -type=any option is used to query all available DNS records (ANY) for the domain google.com. When you execute this command, your system sends a DNS query to the default DNS server configured on your system (or the one specified in your network settings) to retrieve all DNS records associated with the domain google.com. The DNS server then responds with information about all available

DNS record types, including A (IPv4 address), AAAA (IPv6 address), MX (mail exchange), NS (name server), SOA (start of authority), and others.

Screenshot:

```
Server:      127.0.0.53
Address:     127.0.0.53#53

Non-authoritative answer:
Name:  google.com
Address: 142.251.42.46
Name:  google.com
Address: 2404:6800:4009:830::200e
google.com      mail exchanger = 10 smtp.google.com.
google.com
    origin = ns1.google.com
    mail addr = dns-admin.google.com
    serial = 616561579
    refresh = 900
    retry = 900
    expire = 1800
    minimum = 60
google.com      nameserver = ns3.google.com.
google.com      nameserver = ns2.google.com.
google.com      nameserver = ns4.google.com.
google.com      nameserver = ns1.google.com.
```

Extra Commands:

Command:

ipconfig

Explanation:

The IPCONFIG network command provides a comprehensive view of information regarding the IP address configuration of the device we are currently working on.

The IPConfig command also provides us with some variation in the primary command that targets specific system settings or data, which are:

IPConfig/all - Provides primary output with additional information about network adapters.

IPConfig/renew - Used to renew the system's IP address.

IPConfig/release - Removes the system's current IP address.x`

Screenshot:

```
C:\Users\Vikas>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet 2:

  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 1:

  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 2:

  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

  Connection-specific DNS Suffix  . :
  IPv6 Address . . . . . : 2409:40c0:56:358:a43c:a278:e39d:5c77
  Temporary IPv6 Address . . . . . : 2409:40c0:56:358:999a:2fab:9ee4:ba3b
  Link-local IPv6 Address . . . . . : fe80::7dcf:dfec:6168:c8a3%13
  IPv4 Address . . . . . : 192.168.107.48
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . : fe80::64c4:3aff:fe15:f0a6%13
```

Command:

Ping www.google.com

Explanation:

The Ping command is one of the most widely used commands in the prompt tool, as it allows the user to check the connectivity of our system to another host.

This command sends four experimental packets to the destination host to check whether it receives them successfully, if so, then, we can communicate with the destination host. But in case the packets have not been received, that means, no communication can be established with the destination host.

Screenshot:

```
C:\Users\Vikas>ping www.google.com

Pinging www.google.com [2404:6800:4009:82b::2004] with 32 bytes of data:
Reply from 2404:6800:4009:82b::2004: time=713ms
Reply from 2404:6800:4009:82b::2004: time=376ms
Reply from 2404:6800:4009:82b::2004: time=200ms
Reply from 2404:6800:4009:82b::2004: time=718ms

Ping statistics for 2404:6800:4009:82b::2004:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 200ms, Maximum = 718ms, Average = 501ms
```

Command:

Tracert www.google.com

Explanation:

The TRACERT command is used to trace the route during the transmission of the data packet over to the destination host and also provides us with the “hop” count during transmission.

Using the number of hops and the hop IP address, we can troubleshoot network issues and identify the point of the problem during the transmission of the data packet.

Screenshot:

```
C:\Users\Vikas>tracert www.google.com

Tracing route to www.google.com [2404:6800:4009:82b::2004]
over a maximum of 30 hops:

 1  250 ms    16 ms     3 ms  2409:40c0:56:358::d5
 2  524 ms    923 ms   266 ms  2405:200:5201:0:3924:0:3:87
 3  524 ms    617 ms   173 ms  2405:200:5201:0:3925::ff09
 4  216 ms    615 ms   213 ms  2405:200:801:200::1fb6
 5  *         *         * Request timed out.
 6  48 ms     406 ms   149 ms  2001:4860:1:1::3c8
 7  384 ms    366 ms   241 ms  2001:4860:1:1::3c8
 8  29 ms     29 ms    30 ms  2404:6800:811b::1
 9  723 ms    195 ms   214 ms  2001:4860:0:1::160
10  821 ms    489 ms   204 ms  2001:4860:0:1::3129
11  215 ms    899 ms   563 ms  bom12s18-in-x04.1e100.net [2404:6800:4009:82b::2004]

Trace complete.
```

Command:

netstat

Explanation:

The Netstat command as the name suggests displays an overview of all the network connections in the device. The table shows detail about the connection protocol, address, and the current state of the network.

Screenshot:

```
C:\Users\Vikas>netstat

Active Connections

  Proto  Local Address        Foreign Address      State
  TCP    127.0.0.1:59792      LAPTOP-R73NGRAQ:59795 ESTABLISHED
  TCP    127.0.0.1:59795      LAPTOP-R73NGRAQ:59792 ESTABLISHED
  TCP    192.168.107.48:60451 162.247.243.29:https ESTABLISHED
  TCP    [::1]:1521           LAPTOP-R73NGRAQ:49672 ESTABLISHED
  TCP    [::1]:49672          LAPTOP-R73NGRAQ:1521 ESTABLISHED
  TCP    [2409:40c0:56:358:999a:2fab:9ee4:ba3b]:54148 [64:ff9b::14c6:778f]:https ESTABLISHED
  TCP    [2409:40c0:56:358:999a:2fab:9ee4:ba3b]:59709 whatsapp-cdn6-shv-01-bom2:https ESTABLISHED
  TCP    [2409:40c0:56:358:999a:2fab:9ee4:ba3b]:59729 [64:ff9b::a2fe:370]:4444 ESTABLISHED
  TCP    [2409:40c0:56:358:999a:2fab:9ee4:ba3b]:59915 sa-in-f188:5228 ESTABLISHED
  TCP    [2409:40c0:56:358:999a:2fab:9ee4:ba3b]:59916 sa-in-f188:5228 ESTABLISHED
  TCP    [2409:40c0:56:358:999a:2fab:9ee4:ba3b]:59917 sa-in-f188:5228 ESTABLISHED
  TCP    [2409:40c0:56:358:999a:2fab:9ee4:ba3b]:59918 [64:ff9b::14d4:5875]:https ESTABLISHED
  TCP    [2409:40c0:56:358:999a:2fab:9ee4:ba3b]:60166 [2603:1030:210:5::119]:https ESTABLISHED
  TCP    [2409:40c0:56:358:999a:2fab:9ee4:ba3b]:60398 li781-4:https ESTABLISHED
  TCP    [2409:40c0:56:358:999a:2fab:9ee4:ba3b]:60401 li695-222:https ESTABLISHED
  TCP    [2409:40c0:56:358:999a:2fab:9ee4:ba3b]:60435 a184-86-248-59:https ESTABLISHED
  TCP    [2409:40c0:56:358:999a:2fab:9ee4:ba3b]:60437 [2606:4700:8d7b:5e0d:d6d7:493:d5a4:f5d7]:https ESTABLISHED
  TCP    [2409:40c0:56:358:999a:2fab:9ee4:ba3b]:60438 [2606:4700:8d7b:5e0d:d6d7:493:d5a4:f5d7]:https ESTABLISHED
  TCP    [2409:40c0:56:358:999a:2fab:9ee4:ba3b]:60445 [64:ff9b::144b:20ff]:https ESTABLISHED
```

Shirish Shetty 2103164 C32

Experiment – 8

Aim: Examine the use of packet sniffer tool: Wireshark

Theory:

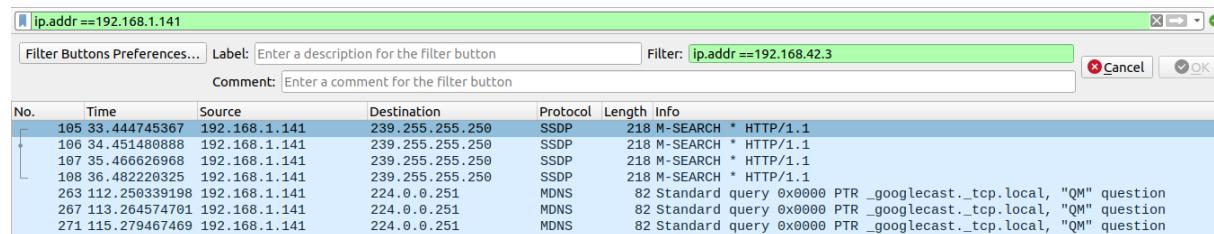
Command:

ip.addr == 192.168.42.3

Explanation:

The Wireshark filter ip.addr == 192.168.42.3 is used to display network traffic involving the IP address 192.168.42.3. Suppose you want to analyze network traffic to and from a particular device in your network, identified by the IP address 192.168.42.3. Applying this filter will narrow down the displayed packets to only those involving communication with that specific IP address, making it easier to analyze its network behavior.

Screenshot:



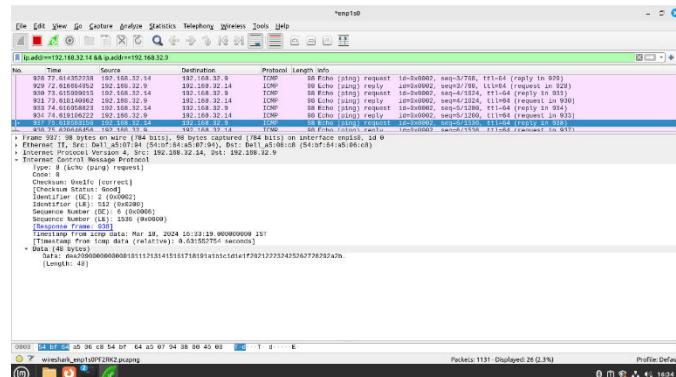
Command:

ip.addr == 10.0.5.119 && ip.addr == 91.189.94.25

Explanation:

This filter will capture network traffic where both the source and destination IP addresses match the specified values simultaneously. This filter can be useful for troubleshooting network communication between two specific hosts. It allows you to focus on traffic exchanged exclusively between these two IP addresses, helping to diagnose any issues or monitor communication between them.

Screenshot:



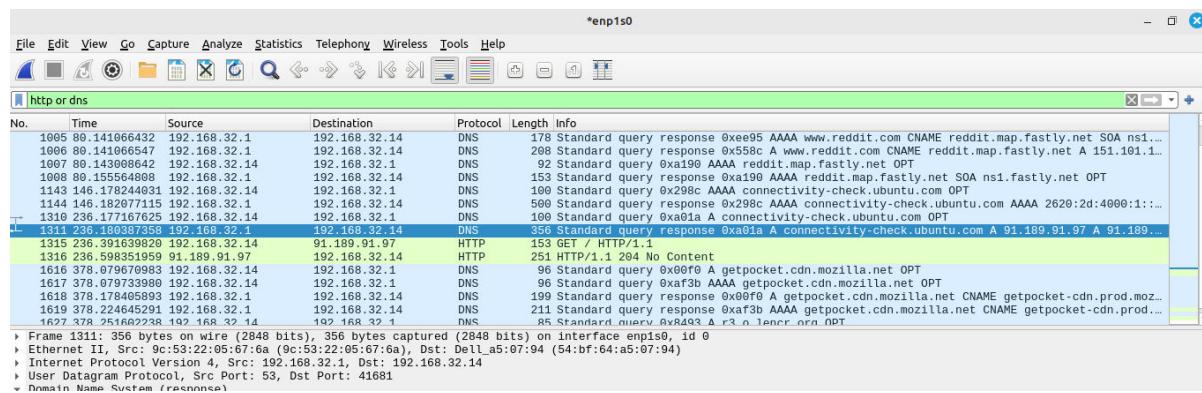
Command:

http or dns

Explanation:

This filter will capture network traffic that matches either the HTTP or DNS protocol. It allows you to monitor and analyze web browsing activity (HTTP) as well as DNS queries and responses. You can use this filter to monitor HTTP traffic to analyze web browsing activity, including requests and responses exchanged between clients and servers. This filter also allows you to capture DNS traffic to analyze DNS queries and responses, helping to troubleshoot DNS-related issues or monitor DNS activity on the network.

Screenshot:



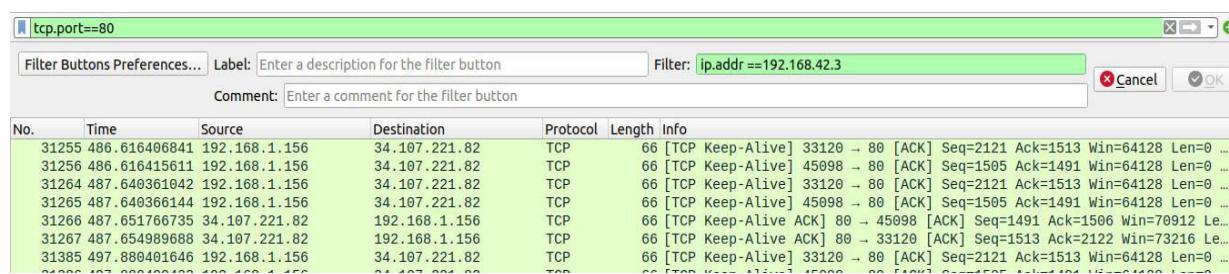
Command:

tcp.port==4000

Explanation:

This Wireshark filter is used to capture network traffic that is transmitted over TCP (Transmission Control Protocol) and is specifically targeted at a particular port number.

Screenshot:



Command:

tcp.flags.reset==0

Explanation:

This Wireshark filter is used to capture TCP (Transmission Control Protocol) packets where the Reset (RST) flag is not set. In TCP, the Reset (RST) flag is used to terminate a connection abruptly or to indicate an error condition. When the RST flag is set in a TCP packet, it signifies the termination of the connection.

By filtering packets where the Reset (RST) flag is not set (i.e., `tcp.flags.reset==0`), you can capture TCP traffic that is not associated with connection termination or error conditions. This may include normal data transmission, connection establishment, or connection teardown without using the Reset flag.

Screenshot:

No.	Time	Source	Destination	Protocol	Length	Info
32196	538.321027229	192.168.1.156	31.13.79.53	TCP	66	48106 -> 443 [ACK] Seq=145291 Ack=368416 Win=393344 Len=0 TSval=19025...
32197	538.325494948	31.13.79.53	192.168.1.156	TCP	2826	443 -> 48106 [ACK] Seq=368416 Ack=145291 Win=386560 Len=2760 TSval=14...
32198	538.325516489	192.168.1.156	31.13.79.53	TCP	66	48106 -> 443 [ACK] Seq=145291 Ack=371176 Win=393472 Len=0 TSval=19025...
32199	538.325527796	31.13.79.53	192.168.1.156	TLSv1.3	1392	Application Data
32200	538.333284461	31.13.79.53	192.168.1.156	TLSv1.3	4152	Application Data
32201	538.333352748	192.168.1.156	31.13.79.53	TCP	66	48106 -> 443 [ACK] Seq=145291 Ack=376588 Win=392576 Len=0 TSval=19025...
32202	538.340098466	31.13.79.53	192.168.1.156	TLSv1.3	4152	Application Data
32203	538.341202000	192.168.1.156	31.13.79.53	TCP	66	48106 -> 443 [ACK] Seq=145291 Ack=386560 Win=393472 Len=0 TSval=19025...

Command:

```
http.request
```

Explanation:

This filter specifies that the captured packets must contain HTTP request messages. By applying the `http.request` filter, you capture only packets that contain HTTP request messages. This can include requests for web pages, images, scripts, stylesheets, or any other resources hosted on a web server.

Screenshot:

No.	Time	Source	Destination	Protocol	Length	Info
107	35.466626968	192.168.1.141	239.255.255.250	SSDP	218	M-SEARCH * HTTP/1.1
108	36.482220325	192.168.1.141	239.255.255.250	SSDP	218	M-SEARCH * HTTP/1.1
152	68.29828226	192.168.2.102	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
154	68.863957001	192.168.1.151	239.255.255.250	SSDP	215	M-SEARCH * HTTP/1.1
159	69.303016577	192.168.2.102	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
165	69.872329741	192.168.1.151	239.255.255.250	SSDP	215	M-SEARCH * HTTP/1.1
170	70.304095390	192.168.2.102	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1

Command:

```
!(arp or icmp or dns)
```

Explanation:

This Wireshark filter is used to capture packets that are not ARP (Address Resolution Protocol), ICMP (Internet Control Message Protocol), or DNS (Domain Name System) traffic. It allows you to analyze actual data communication between devices on the network without being distracted by protocol-specific messages used for network management or resolution.

Screenshot:

!arp or icmp or dns						
No.	Time	Source	Destination	Protocol	Length	Info
3	0.001472	2606:4700:8d7b:5e0d..	2409:40c2:1007:c810..	TCP	74	443 → 64917 [ACK] Seq=1 Ack=1 Win=8 Len=0
4	0.002644	2606:4700:8d7b:5e0d..	2409:40c2:1007:c810..	TLSv1.2	372	Application Data
5	0.002644	2606:4700:8d7b:5e0d..	2409:40c2:1007:c810..	TLSv1.2	118	Application Data
6	0.002644	2606:4700:8d7b:5e0d..	2409:40c2:1007:c810..	TLSv1.2	105	Application Data
7	0.002723	2409:40c2:1007:c810..	2606:4700:8d7b:5e0d..	TCP	74	64917 → 443 [ACK] Seq=1 Ack=374 Win=257 Len=0
8	0.010570	2409:40c2:1007:c810..	2606:4700:8d7b:5e0d..	TCP	86	64920 → 443 [SYN] Seq=0 Win=64800 Len=0 MSS=1350 WS=256 SACK_PERM
9	0.262600	2409:40c2:1007:c810..	2606:4700:8d7b:5e0d..	TCP	86	64921 → 443 [SYN] Seq=0 Win=64800 Len=0 MSS=1350 WS=256 SACK_PERM
10	0.264119	2606:4700:8d7b:5e0d..	2409:40c2:1007:c810..	TCP	74	443 → 64918 [ACK] Seq=1 Ack=1 Win=7 Len=0
11	0.264119	2606:4700:8d7b:5e0d..	2409:40c2:1007:c810..	TLSv1.3	1374	Server Hello, Change Cipher Spec
13	0.264237	104.26.15.76	192.168.45.48	TCP	66	443 → 64919 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1300 SACK_PERM WS=8192

Command:

not (tcp.port == 80) and not (tcp port == 25)

Explanation:

This filter is useful when you want to capture network traffic excluding web browsing (HTTP) and email communication (SMTP).

Screenshot:

not(tcp.port==80)&¬(tcp.port==25)						
Filter Buttons Preferences...		Label:	Enter a description for the filter button	Filter:	ip.addr == 192.168.42.3	
No.	Time	Source	Destination	Protocol	Length	Info
316	138.746185279	0.0.0.0	255.255.255.255	DHCP	346	DHCP Request - Transaction ID 0x3d8ec87d
317	138.821916577	fe80::7ec0:91e5:d50..	ff02::16	ICMPv6	110	Multicast Listener Report Message v2
318	140.620790914	fe80::7ec0:91e5:d50..	ff02::16	ICMPv6	110	Multicast Listener Report Message v2
319	140.629836852	0.0.0.0	255.255.255.255	DHCP	346	DHCP Request - Transaction ID 0xed4fd8a1
320	140.696616384	fe80::7ec0:91e5:d50..	ff02::16	ICMPv6	110	Multicast Listener Report Message v2
321	140.756901802	Dell_2a:d6:44	Broadcast	ARP	60	Who has 192.168.1.248? Tell 192.168.1.193
322	141.492800872	fe80::7ec0:91e5:d50..	ff02::16	ICMPv6	110	Multicast Listener Report Message v2

Command:

ftp

Explanation:

This command filters the displayed packets to show only those related to FTP (File Transfer Protocol) traffic. When you enter ftp into the Wireshark capture filter field, Wireshark will display packets that are part of FTP communication, including commands, responses, data transfers, and control messages.

Screenshot:

```
C:\Users\Vikas\Downloads>ftp 192.168.45.48
Connected to 192.168.45.48.
220-FileZilla Server 1.8.1
220 Please visit https://filezilla-project.org/
202 UTF8 mode is always enabled. No need to send this command
User (192.168.45.48:(none)): user
331 Please, specify the password.
Password:
230 Login successful.
ftp> |
```

ftp	Source	Destination	Protocol	Length	Info
ftp	192.168.45.48	192.168.45.48	FTP	72	Request: PORT 192,168,45,48,200,101
9 20...	192.168.45.48	192.168.45.48	FTP	74	Response: 200 PORT command successful.
11 20...	192.168.45.48	192.168.45.48	FTP	50	Request: NLST
17 20...	192.168.45.48	192.168.45.48	FTP	73	Response: 150 Starting data transfer.
76 20...	192.168.45.48	192.168.45.48	FTP	70	Response: 226 Operation successful
84 20...	192.168.45.48	192.168.45.48	FTP	50	Request: QUIT
88 20...	192.168.45.48	192.168.45.48	FTP	58	Response: 221 Goodbye.
101 20...	192.168.45.48	192.168.45.48	FTP	121	Response: 220-FileZilla Server 1.8.1
103 20...	192.168.45.48	192.168.45.48	FTP	58	Request: OPTS UTF8 ON
107 20...	192.168.45.48	192.168.45.48	FTP	107	Response: 207 UTF8 mode is always enabled. No need to send this command
109 20...	192.168.45.48	192.168.45.48	FTP	55	Request: USER user
115 20...	192.168.45.48	192.168.45.48	FTP	79	Response: 331 Please, specify the password.
117 20...	192.168.45.48	192.168.45.48	FTP	54	Request: PASS 123
123 20...	192.168.45.48	192.168.45.48	FTP	67	Response: 230 Login successful.

Experiment No. 9

Aim: Setting up personal Firewall using iptables

Theory:-

Description

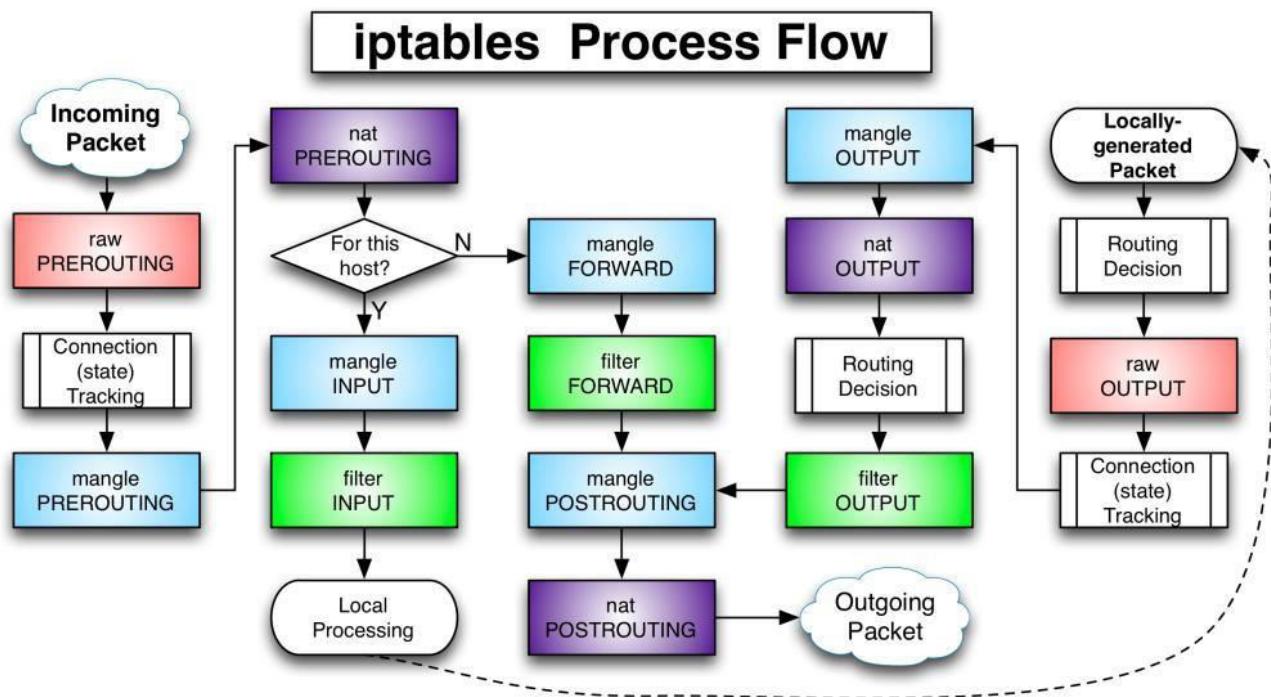
Iptables is used to set up, maintain, and inspect the tables of IP packet filter rules in the Linux kernel. Several different tables may be defined. Each table contains a number of built-in chains and may also contain user-defined chains.

Each chain is a list of rules which can match a set of packets. Each rule specifies what to do with a packet that matches. This is called a 'target', which may be a jump to a user-defined chain in the same table.

Targets

A firewall rule specifies criteria for a packet, and a target. If the packet does not match, the next rule in the chain is examined; if it does match, then the next rule is specified by the value of the target, which can be the name of a user-defined chain or one of the special values ACCEPT, DROP, QUEUE, or RETURN.

ACCEPT means to let the packet through. DROP means to drop the packet on the floor. QUEUE means to pass the packet to userspace. (How the packet can be received by a userspace process differs by the particular queue handler. 2.4.x and 2.6.x kernels up to 2.6.13 include the ip_queue queue handler. Kernels 2.6.14 and later additionally include the nfnetlink_queue queue handler. Packets with a target of QUEUE will be sent to queue number '0' in this case. Please also see the NFQUEUE target as described later in this man page.) RETURN means stop traversing this chain and resume at the next rule in the previous (calling) chain. If the end of a built-in chain is reached or a rule in a built-in chain with target RETURN is matched, the target specified by the chain policy determines the fate of the packet.



All packets inspected by iptables pass through a sequence of built-in tables (queues) for processing. Each of these queues is dedicated to a particular type of packet activity and is controlled by an associated packet transformation/filtering chain.

1. Filter Table

Filter is default table for iptables.

Iptables's filter table has the following built-in chains.

- INPUT chain – Incoming to firewall. For packets coming to the local server.
- OUTPUT chain – Outgoing from firewall. For packets generated locally and going out of the local server.
- FORWARD chain – Packet for another NIC on the local server. For packets routed through the local server.

2. NAT Table

This table is consulted when a packet that creates a new connection is encountered.

Iptable's NAT table has the following built-in chains.

- PREROUTING chain – Alters packets before routing. i.e Packet translation happens immediately after the packet comes to the system (and before routing). This helps to translate the destination ip address of the packets to something that matches the routing on the local server. This is used for DNAT (destination NAT).

- POSTROUTING chain – Alters packets after routing. i.e Packet translation happens when the packets are leaving the system. This helps to translate the source ip address of the packets to something that might match the routing on the destination server. This is used for SNAT (source NAT).
- OUTPUT chain – NAT for locally generated packets on the firewall.

3. Mangle Table

Iptables's Mangle table is for specialized packet alteration. This alters QOS bits in the TCP header. Mangle table has the following built-in chains.

- PREROUTING chain
- OUTPUT chain
- FORWARD chain
- INPUT chain
- POSTROUTING chain

4. Raw Table

Iptable's Raw table is for configuration exemptions. Raw table has the following built-in chains.

- PREROUTING chain
- OUTPUT chain

5. Security Table

This table is used for Mandatory Access Control (MAC) networking rules, such as those enabled by the SECMARK and CONNSECMARK targets. Mandatory Access Control is implemented by Linux Security Modules such as SELinux. The security table is called after the filter table, allowing any Discretionary Access Control (DAC) rules in the filter table to take effect before MAC rules. This table provides the following built-in chains: INPUT (for packets coming into the box itself), OUTPUT (for altering locally-generated packets before routing), and FORWARD (for altering packets being routed through the box).

Chains

Tables consist of *chains*, Rules are combined into different chains. The kernel uses chains to manage packets it receives and sends out. A chain is simply a checklist of rules which are lists of rules which are followed in order. The rules operate with an if-then -else structure.

Input – This chain is used to control the behaviour for incoming connections. For example, if a user attempts to SSH into your PC/server, iptables will attempt to match the IP address and port to a rule in the input chain.

Forward – This chain is used for incoming connections that aren't actually being delivered locally. Think of a router – data is always being sent to it but rarely actually destined for the router itself; the data is just forwarded to its target.

Output – This chain is used for outgoing connections. For example, if you try to ping howtogeek.com, iptables will check its output chain to see what the rules are regarding ping and howtogeek.com before making a decision to allow or deny the connection attempt.

Targets:

ACCEPT: Allow packet to pass through the firewall.

DROP: Deny access by the packet.

REJECT: Deny access and notify the server.

QUEUE: Send packets to user space.

RETURN: jump to the end of the chain and let the default target process it

iptables command Switch	Description
-L	Listing of rules present in the chain
-n	Numeric output of addresses and ports
-v	Displays the rules in verbose mode
-t <table->	If you don't specify a table, then the filter table is assumed. As discussed before, the possible built-in tables include: filter, nat, mangle
-j <target>	Jump to the specified target chain when the packet matches the current rule.
-A	Append rule to end of a chain
-F	Flush. Deletes all the rules in the selected table
-p <protocol-type>	Match protocol. Types include, icmp, tcp, udp, and all
-s <ip-address>	Match source IP address
-d <ip-address>	Match destination IP address
-i <interface-name>	Match "input" interface on which the packet enters.
-o <interface-name>	Match "output" interface on which the packet exits

1. List iptables rules: `iptables -L` - Displays the current list of iptables rules, initially empty.
2. Block outgoing traffic: `iptables -I OUTPUT -s <your ip> -d <neighbour ip> -p <protocol> -j <action>` - Blocks outgoing traffic to a specific destination for a specific protocol from a machine.
3. Allow outgoing traffic: `iptables -I OUTPUT -s <your ip> -d <neighbour ip> -p <protocol> -j ACCEPT` - Allows outgoing traffic to a specific destination for a specific protocol from a machine.
4. Block outgoing traffic temporarily: `iptables -I OUTPUT -s <your ip> -d <neighbour ip> -p <protocol> -j REJECT` - Temporarily blocks outgoing traffic to a specific destination for a specific protocol from a machine.
5. Block incoming traffic: `iptables -I INPUT -s <neighbour ip> -d <firewall ip> -p <protocol> -j DROP` - Blocks incoming traffic from a particular destination for a specific protocol to the machine.

6. Allow incoming traffic: `iptables -I INPUT -s <neighbour ip> -d <firewall ip> -p <protocol> -j ACCEPT` - Allows incoming traffic from a particular destination for a specific protocol to the machine.

7. Clear iptables rules: `iptables -F` - Clears all rules in iptables.

8. Block specific URL: `iptables -t filter -I OUTPUT -m string --string facebook.com -j REJECT --algo kmp` - Blocks access to a specific URL (e.g., facebook.com) using string matching with the KMP algorithm.

EXTRA:

9. Block all incoming and outgoing traffic by default:

- `iptables -P INPUT DROP`
- `iptables -P OUTPUT DROP`

Sets the default policy for incoming and outgoing traffic to DROP, effectively blocking all traffic unless explicitly allowed by rules.

10. Allow established connections:

- `iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT`
- `iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT`

Allows incoming and outgoing traffic that is part of an established or related connection, ensuring that established connections are not blocked.

Steps:-

1. Get root access: \$ sudo su root

2. # apt-get install iptables

Commands:-

1. To see the list of iptables rules

iptables -L

. Initially it is empty

```
Processing triggers for libtcm-bin (2.37-12) ...
root@kali:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

2. To block outgoing traffic to a particular destination for a specific protocol from a machine

Syntax: iptables -I OUTPUT -s <your ip> -d <neighbour ip> -p <protocol> -j <action>

Open one terminal and Ping the neighbour. Let the ping run.

```
#ping 192.168.208.6
```

Open another terminal and run the iptables command

```
# iptables -I OUTPUT -s 192.168.208.18 -d 192.168.208.6 -p icmp -j DROP
```

2. To allow outgoing traffic to a particular destination for a specific protocol from a machine

```
# iptables -I OUTPUT -s 192.168.208.18 -d 192.168.208.6 -p icmp -j ACCEPT
```

3. To block outgoing traffic to a particular destination for a specific protocol from a machine for sometime

```
# iptables -I OUTPUT -s 192.168.208.18 -d 192.168.208.6 -p icmp -j REJECT
```

```
root@admini-OptiPlex-3050:/home/admini/Desktop# ping 192.168.1.159
PING 192.168.1.159 (192.168.1.159) 56(84) bytes of data.
64 bytes from 192.168.1.159: icmp_seq=1 ttl=64 time=0.107 ms
64 bytes from 192.168.1.159: icmp_seq=2 ttl=64 time=0.504 ms
64 bytes from 192.168.1.159: icmp_seq=3 ttl=64 time=0.283 ms
64 bytes from 192.168.1.159: icmp_seq=4 ttl=64 time=0.519 ms
^C
--- 192.168.1.159 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3053ms
rtt min/avg/max/mdev = 0.107/0.353/0.519/0.170 ms
root@admini-OptiPlex-3050:/home/admini/Desktop# iptables -I OUTPUT -s 192.168.1.160 -d 192.168.1.159 -p icmp -j DROP
root@admini-OptiPlex-3050:/home/admini/Desktop# ping 192.168.1.159
PING 192.168.1.159 (192.168.1.159) 56(84) bytes of data.
^C
--- 192.168.1.159 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2045ms
root@admini-OptiPlex-3050:/home/admini/Desktop# iptables -I OUTPUT -s 192.168.1.160 -d 192.168.1.159 -p icmp -j ACCEPT
root@admini-OptiPlex-3050:/home/admini/Desktop# ping 192.168.1.159
PING 192.168.1.159 (192.168.1.159) 56(84) bytes of data.
64 bytes from 192.168.1.159: icmp_seq=1 ttl=64 time=0.208 ms
64 bytes from 192.168.1.159: icmp_seq=2 ttl=64 time=0.501 ms
64 bytes from 192.168.1.159: icmp_seq=3 ttl=64 time=0.502 ms
64 bytes from 192.168.1.159: icmp_seq=4 ttl=64 time=0.519 ms
^C
--- 192.168.1.159 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3068ms
rtt min/avg/max/mdev = 0.208/0.432/0.519/0.129 ms
root@admini-OptiPlex-3050:/home/admini/Desktop#
```

Allow the traffic again by using ACCEPT instead of REJECT

```
root@OptiPlex-3050:/home/admini/Desktop# iptables -I OUTPUT -s 192.168.1.160 -d 192.168.1.159 -p icmp -j REJECT
root@OptiPlex-3050:/home/admini/Desktop# ping 192.168.1.159
PING 192.168.1.159 (192.168.1.159) 56(84) bytes of data.
From 192.168.1.160 icmp_seq=1 Destination Port Unreachable
ping: sendmsg: Operation not permitted
From 192.168.1.160 icmp_seq=2 Destination Port Unreachable
ping: sendmsg: Operation not permitted
From 192.168.1.160 icmp_seq=3 Destination Port Unreachable
ping: sendmsg: Operation not permitted
From 192.168.1.160 icmp_seq=4 Destination Port Unreachable
ping: sendmsg: Operation not permitted
From 192.168.1.160 icmp_seq=5 Destination Port Unreachable
ping: sendmsg: Operation not permitted
From 192.168.1.160 icmp_seq=6 Destination Port Unreachable
ping: sendmsg: Operation not permitted
From 192.168.1.160 icmp_seq=7 Destination Port Unreachable
ping: sendmsg: Operation not permitted
From 192.168.1.160 icmp_seq=8 Destination Port Unreachable
ping: sendmsg: Operation not permitted
From 192.168.1.160 icmp_seq=9 Destination Port Unreachable
ping: sendmsg: Operation not permitted
^C
--- 192.168.1.159 ping statistics ---
9 packets transmitted, 0 received, +9 errors, 100% packet loss, time 8186ms

root@OptiPlex-3050:/home/admini/Desktop# iptables -I OUTPUT -s 192.168.1.160 -d 192.168.1.159 -p icmp -j ACCEPT
root@OptiPlex-3050:/home/admini/Desktop# ping 192.168.1.159
PING 192.168.1.159 (192.168.1.159) 56(84) bytes of data.
64 bytes from 192.168.1.159: icmp_seq=1 ttl=64 time=0.373 ms
64 bytes from 192.168.1.159: icmp_seq=2 ttl=64 time=0.428 ms
64 bytes from 192.168.1.159: icmp_seq=3 ttl=64 time=0.326 ms
64 bytes from 192.168.1.159: icmp_seq=4 ttl=64 time=0.176 ms
^C
--- 192.168.1.159 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3050ms
rtt min/avg/max/mdev = 0.176/0.325/0.428/0.093 ms
root@OptiPlex-3050:/home/admini/Desktop#
```

4. To block incoming traffic from particular destination for a specific protocol to machine

Syntax: `iptables -I INPUT -s <neighbour ip> -d <firewall ip> -p <protocol> -j <action>`

Open one terminal and Ping the neighbour. Let the ping run.

#`ping 192.168.208.6`

Open another terminal and run the iptables command

#`iptables -I INPUT -s 192.168.208.6 -d 192.168.208.18 -p icmp -j DROP`

5. To allow incoming traffic from particular destination for a specific protocol to machine

Syntax: `iptables -I INPUT -s <neighbour ip> -d <firewall ip> -p <protocol> -j <action>`

Open another terminal and run the iptables command

#`iptables -I INPUT -s 192.168.208.6 -d 192.168.208.18 -p icmp -j ACCEPT`

Check the ping status on the other terminal

```
root@OptiPlex-3050:/home/admini/Desktop# iptables -I INPUT -s 192.168.1.158 -d 192.168.1.160 -p icmp -j DROP
root@OptiPlex-3050:/home/admini/Desktop# iptables -I INPUT -s 192.168.1.158 -d 192.168.1.160 -p icmp -j ACCEPT
root@OptiPlex-3050:/home/admini/Desktop#
```

```
root@admini-OptiPlex-3050: ~ admini@admini-OptiPlex-3050: ~ admini@admini-OptiPlex-3050: ~ admini@admini-OptiPlex-3050: ~ admini@admini-OptiPlex-3050: ~
admini@admini-OptiPlex-3070: ~ $ ping 192.168.1.160
PING 192.168.1.160 (192.168.1.160) 56(84) bytes of data.
64 bytes from 192.168.1.160: icmp_seq=1 ttl=64 time=0.830 ms
64 bytes from 192.168.1.160: icmp_seq=2 ttl=64 time=0.543 ms
64 bytes from 192.168.1.160: icmp_seq=3 ttl=64 time=0.533 ms
64 bytes from 192.168.1.160: icmp_seq=4 ttl=64 time=0.173 ms
64 bytes from 192.168.1.160: icmp_seq=5 ttl=64 time=0.397 ms
^C
--- 192.168.1.160 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4101ms
rtt min/avg/max/mdev = 0.173/0.495/0.830/0.214 ms
admini@admini-OptiPlex-3070: ~ $ ping 192.168.1.160
PING 192.168.1.160 (192.168.1.160) 56(84) bytes of data.
^C
--- 192.168.1.160 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4101ms

admini@admini-OptiPlex-3070: ~ $ ping 192.168.1.160
PING 192.168.1.160 (192.168.1.160) 56(84) bytes of data.
64 bytes from 192.168.1.160: icmp_seq=1 ttl=64 time=0.350 ms
64 bytes from 192.168.1.160: icmp_seq=2 ttl=64 time=0.544 ms
64 bytes from 192.168.1.160: icmp_seq=3 ttl=64 time=0.544 ms
64 bytes from 192.168.1.160: icmp_seq=4 ttl=64 time=0.353 ms
64 bytes from 192.168.1.160: icmp_seq=5 ttl=64 time=0.540 ms
64 bytes from 192.168.1.160: icmp_seq=6 ttl=64 time=0.544 ms
64 bytes from 192.168.1.160: icmp_seq=7 ttl=64 time=0.511 ms
64 bytes from 192.168.1.160: icmp_seq=8 ttl=64 time=0.362 ms
64 bytes from 192.168.1.160: icmp_seq=9 ttl=64 time=0.533 ms
^C
--- 192.168.1.160 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8196ms
rtt min/avg/max/mdev = 0.350/0.475/0.544/0.085 ms
admini@admini-OptiPlex-3070: ~
```

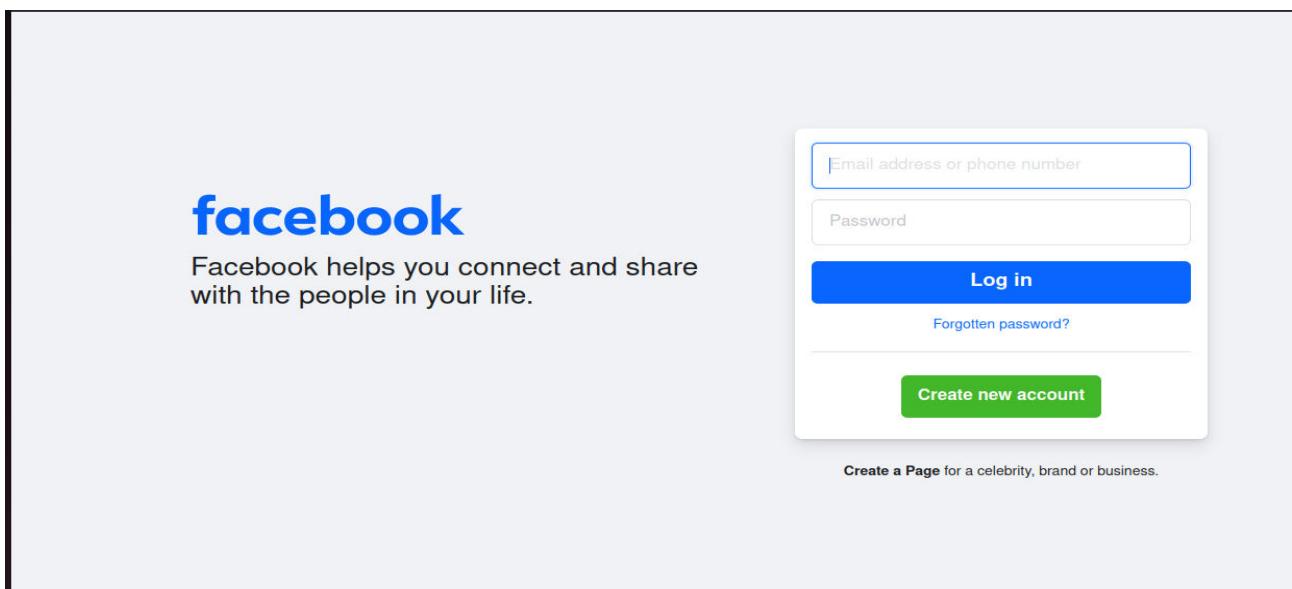
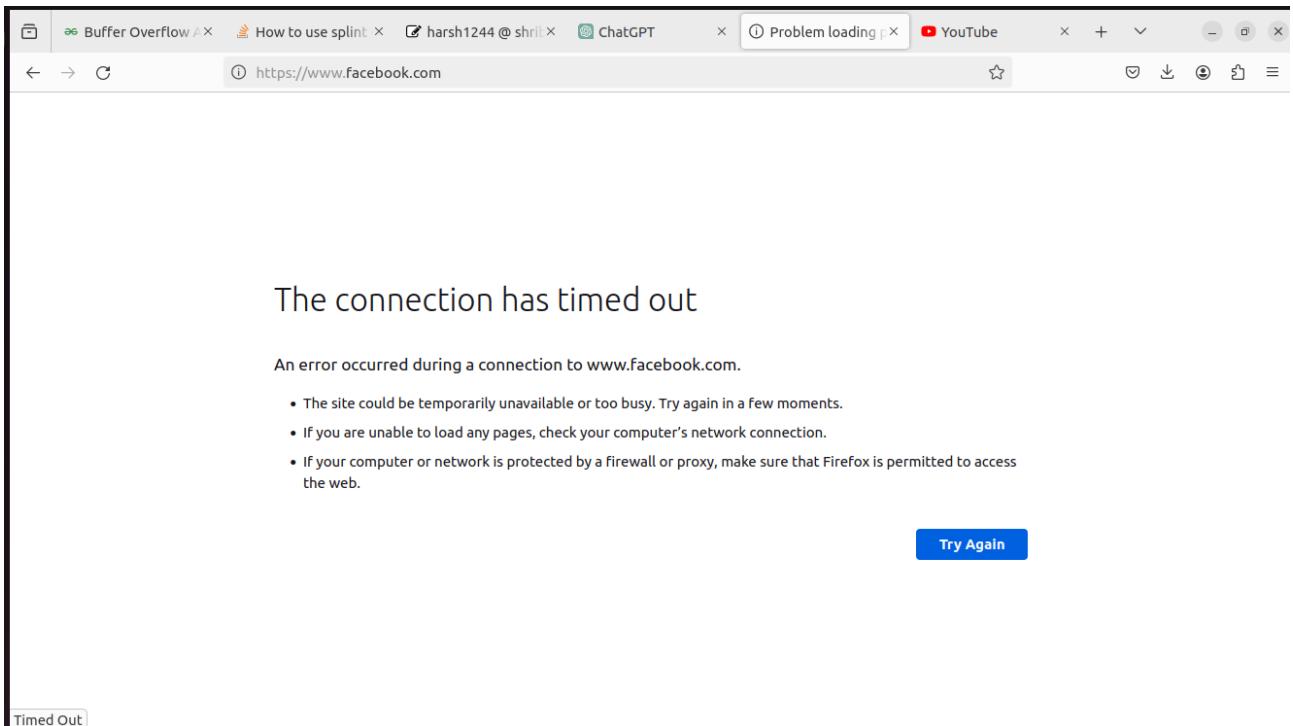
6. To clear the rules in iptables

```
# iptables -F
```

7. To block specific URL from machine

```
# iptables -t filter -I OUTPUT -m string --string facebook.com -j REJECT --algo kmp
```

```
root@admini-OptiPlex-3050:/home/admini/Desktop# iptables -F
root@admini-OptiPlex-3050:/home/admini/Desktop# iptables -t filter -I OUTPUT -m string --string facebook.com -j REJECT --algo kmp
root@admini-OptiPlex-3050:/home/admini/Desktop# iptables -t filter -I OUTPUT -m string --string facebook.com -j ACCEPT --algo kmp
root@admini-OptiPlex-3050:/home/admini/Desktop#
```



It will block facebook.com by performing string matching. The algorithm used for string matching is KMP.

If we change target *from REJECT to ACCEPT*, the site can be visited again.

EXTRA :

To block all incoming and outgoing traffic by default:

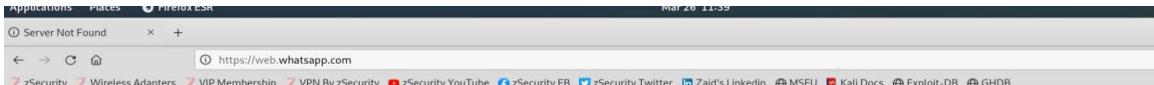
```
iptables -P INPUT DROP  
iptables -P OUTPUT DROP
```

These commands set the default policy for incoming and outgoing traffic to DROP, effectively blocking all traffic unless explicitly allowed by rules.

To

```
root@kali:~# ping 192.168.232.128  
PING 192.168.232.128 (192.168.232.128) 56(84) bytes of data.  
64 bytes from 192.168.232.128: icmp_seq=1 ttl=64 time=0.618 ms  
64 bytes from 192.168.232.128: icmp_seq=2 ttl=64 time=0.043 ms  
64 bytes from 192.168.232.128: icmp_seq=3 ttl=64 time=0.039 ms  
64 bytes from 192.168.232.128: icmp_seq=4 ttl=64 time=0.039 ms  
64 bytes from 192.168.232.128: icmp_seq=5 ttl=64 time=0.087 ms  
64 bytes from 192.168.232.128: icmp_seq=6 ttl=64 time=0.040 ms  
64 bytes from 192.168.232.128: icmp_seq=7 ttl=64 time=0.040 ms  
64 bytes from 192.168.232.128: icmp_seq=8 ttl=64 time=0.040 ms  
^V64 bytes from 192.168.232.128: icmp_seq=9 ttl=64 time=0.048 ms  
64 bytes from 192.168.232.128: icmp_seq=10 ttl=64 time=0.040 ms  
64 bytes from 192.168.232.128: icmp_seq=11 ttl=64 time=0.040 ms  
^Z  
[1]+ Stopped ping 192.168.232.128  
root@kali:~# iptables -P INPUT DROP  
root@kali:~# iptables -P OUTPUT DROP  
root@kali:~# ping 192.168.232.128  
PING 192.168.232.128 (192.168.232.128) 56(84) bytes of data.  
^Z^Z  
[2]+ Stopped ping 192.168.232.128  
root@kali:~#
```

allow



established connections:

```
cssCopy code  
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT  
iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

These commands allow incoming and outgoing traffic that is part of an established or related connection, ensuring that established connections are not blocked.

```
[1]+  Stopped                  ping 192.168.232.128
root@kali:~/Desktop# iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
root@kali:~/Desktop# ping 192.168.232.128
PING 192.168.232.128 (192.168.232.128) 56(84) bytes of data.
64 bytes from 192.168.232.128: icmp_seq=1 ttl=64 time=0.059 ms
64 bytes from 192.168.232.128: icmp_seq=2 ttl=64 time=0.031 ms
64 bytes from 192.168.232.128: icmp_seq=3 ttl=64 time=0.029 ms
64 bytes from 192.168.232.128: icmp_seq=4 ttl=64 time=0.030 ms
^Z
[2]+  Stopped                  ping 192.168.232.128
```

Observations:

1. In case of OUTPUT chain, for DROP and REJECT chain, at source machine we get two different messages.
For DROP – ‘Operation Not Permitted’. Here No acknowledgement is provided.
For REJECT – ‘Destination Port Unreachable’. Here acknowledgement is given.
2. In case of INPUT chain for DROP and REJECT chain at source machine we get two different responses as follows:
For DROP – No message. Here No acknowledgement is provided.
For REJECT – ‘Destination Port Unreachable’. Here acknowledgement is given.

Experiment 10

Aim: Simulate Buffer overflow attack using Splint

Explanation:

Buffer overflow is a critical flaw found in certain implementations of the C programming language. These bugs occur when data is written beyond the bounds of a buffer or array, leading to the corruption of the program's stack. This often results in altering the return address of a function call to a clandestine memory location where malicious code is inserted. There are primarily two types of buffer overflow attacks: stack-based and heap-based. Heap-based attacks inundate the memory allocated for a program, but their complexity makes them relatively uncommon. Stack-based buffer overflows, on the other hand, are prevalent.

Splint is a tool used for statically analyzing C programs to uncover security vulnerabilities and programming errors. It performs various lint checks, including detecting unused declarations, type inconsistencies, usage before definition, unreachable code, ignored return values, execution paths lacking returns, potential infinite loops, and cases of fall-through. Additionally, Splint leverages annotations within the source code to enable more powerful checks. These annotations, presented as structured comments, document assumptions about functions, variables, parameters, and types. By exploiting this supplementary information, Splint enhances traditional lint checks and provides greater flexibility in detecting bugs. Moreover, Splint allows developers to customize the level of scrutiny according to the specific requirements of their projects.

Splint identifies numerous issues in C programs, such as dereferencing potentially null pointers, utilizing undefined storage, mismatched types with higher precision than typical C compilers offer, breaches of information hiding, memory management errors like dangling references and memory leaks, hazardous aliasing, inconsistencies in modifying or accessing global variables with specified interfaces, problematic control flow including probable infinite loops, fall-through scenarios, incomplete switches, and dubious statements. Furthermore, it detects buffer overflow vulnerabilities, unsafe implementations or calls of macros, and deviations from prescribed naming conventions.

Steps :

1. Installation

```
$ sudo apt-get install splint
```

2. Checking Vulnerability

```
$ splint program1.c
```

Program1.c is the program whose vulnerability is to be checked.

```
#include <stdio.h>

#include <string.h>

int main(void)

{

    char buff[15];

    int pass = 0;

    printf("\n Enter the password : \n");

    gets(buff);

    if(strcmp(buff, "thegeekstuff"))

    {

        printf ("\n Wrong Password \n");

    }

    else

    {

        printf ("\n Correct Password \n");

        pass = 1;}}
```

Program 3

```
#include <stdio.h>

#include <string.h>

char password[] = "password";

int get_password() {

    int auth_ok = 0;

    char buff[16];

    printf("Enter password: ");

    scanf("%s", buff);

    if(strncmp(buff, password, sizeof(password)) == 0)

        auth_ok = 1;

    return auth_ok; }

void success() {

    printf("Success!

\n");

}
```

Shirish Shetty 2103164 C32

```
int main(int argc, char** argv) {  
    int res = get_password();  
  
    if (res == 0) {  
  
        printf("Failure \n");  
  
        return 0;  
  
    }  
  
    success();  
  
    return 0;  
}
```

Shirish Shetty 2103164 C32

Output :

Program 1

```
admini@admini-OptiPlex-3050:~/Desktop$ gcc test.c -o test.out
test.c: In function 'main':
test.c:8:2: warning: implicit declaration of function 'gets'; did you mean 'fgets'? [-Wimplicit-function-declaration]
  8 |   gets(buff);
     |   ^
     |   fgets
/usr/bin/ld: /tmp/ccGvhMXM.o: in function 'main':
test.c:(.text+0x3e): warning: the 'gets' function is dangerous and should not be used.
admini@admini-OptiPlex-3050:~/Desktop$ ./test.out

Enter the password :
lab403

Wrong Password
admini@admini-OptiPlex-3050:~/Desktop$ splint test.c
Splint 3.1.2 --- 21 Feb 2021

test.c: (in function main)
test.c:8:2: Use of gets leads to a buffer overflow vulnerability. Use fgets
      instead: gets
      Use of function that may lead to buffer overflow. (Use -bufferoverflowhigh to
      inhibit warning)
test.c:8:2: Return value (type char *) ignored: gets(buff)
      Result returned by function call is not used. If this is intended, can cast
      result to (void) to eliminate message. (Use -retvalother to inhibit warning)
test.c:9:5: Test expression for if not boolean, type int:
      strcmp(buff, "thegeekstuff")
      Test expression type is not boolean or int. (Use -predboolint to inhibit
      warning)
test.c:18:3: Path with no return in function declared to return int
      There is a path through a function declared to return a value on which there
      is no return statement. This means the execution may fall through without
      returning a meaningful result to the caller. (Use -noret to inhibit warning)

Finished checking --- 4 code warnings
admini@admini-OptiPlex-3050:~/Desktop$ ./test.out

Enter the password :
thegEEKstuff

Correct Password
admini@admini-OptiPlex-3050:~/Desktop$ splint test.c
Splint 3.1.2 --- 21 Feb 2021

test.c: (in function main)
test.c:8:2: Use of gets leads to a buffer overflow vulnerability. Use fgets
      instead: gets
      Use of function that may lead to buffer overflow. (Use -bufferoverflowhigh to
      inhibit warning)
test.c:8:2: Return value (type char *) ignored: gets(buff)
      Result returned by function call is not used. If this is intended, can cast
      result to (void) to eliminate message. (Use -retvalother to inhibit warning)
test.c:9:5: Test expression for if not boolean, type int:
      strcmp(buff, "thegeekstuff")
      Test expression type is not boolean or int. (Use -predboolint to inhibit
      warning)
test.c:18:3: Path with no return in function declared to return int
      There is a path through a function declared to return a value on which there
      is no return statement. This means the execution may fall through without
      returning a meaningful result to the caller. (Use -noret to inhibit warning)

Finished checking --- 4 code warnings
admini@admini-OptiPlex-3050:~/Desktop$
```

Shirish Shetty 2103164 C32

Program 2 :

```
root@kali:~# gcc program2.c -o p2.out
```

```
program2.c:2:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
```

```
2 | main()
```

```
| ^~~~
```

```
program2.c: In function ‘main’:
```

```
program2.c:6:9: warning: character constant too long for its type
```

```
6 | buff[5]='abcdefghijklmnophsgkgfks';
```

```
| ~~~~~
```

```
program2.c:6:9: warning: overflow in conversion from ‘int’ to ‘char’ changes value from ‘1734765427’ to ‘115’ [-Woverflow]
```

```
root@kali:~# ./p2.out
```

```
My stack looks like:
```

```
0xffff9e6a7098
```

```
0xffff9e6a70a8
```

```
0x5595409e6dd8
```

```
(nil)
```

```
0xfb5769cab10
```

```
(nil)
```

```
0xfb5769f8ab0
```

```
0x1
```

```
0xfb5767f36ca
```

```
0xffff9e6a7080
```

```
s
```

```
My new stack looks like:
```

```
0x559540a302a0
```

```
(nil)
```

```
(nil)
```

```
0x63
```

```
(nil)
```

```
(nil)
```

```
0x7fb5769f8ab0
```

```
0x73
```

```
0x7fb5767f36ca
```

```
0xffff9e6a7080
```

```
root@kali:~# splint program2.c
```

```
Splint 3.1.2 --- 21 Feb 2021
```

```
program2.c: (in function main)
```

```
program2.c:5:1: No argument corresponding to printf format code 1 (%p):
```

```
"My stack looks
```

```
like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"
```

```
Types are incompatible. (Use -type to inhibit warning)
```

```
program2.c:5:33: Corresponding format code
```

```
program2.c:5:1: No argument corresponding to printf format code 2 (%p):
```

```
"My stack looks
```

```
like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"
```

```
program2.c:5:37: Corresponding format code
```

```
program2.c:5:1: No argument corresponding to printf format code 3 (%p):
```

```
"My stack looks
```

```
like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"
```

```
program2.c:5:41: Corresponding format code
```

```
program2.c:5:1: No argument corresponding to printf format code 4 (%p):
```

```
"My stack looks
```

```
like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"
```

```
program2.c:5:45: Corresponding format code
```

```
program2.c:5:1: No argument corresponding to printf format code 5 (%p):
```

```
"My stack looks
```

```
like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"
```

Shirish Shetty 2103164 C32

program2.c:5:49: Corresponding format code

program2.c:5:1: No argument corresponding to printf format code 6 (%p):

"My stack looks

like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"

program2.c:5:53: Corresponding format code

program2.c:5:1: No argument corresponding to printf format code 7 (%p):

"My stack looks

like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"

program2.c:5:57: Corresponding format code

program2.c:5:1: No argument corresponding to printf format code 8 (%p):

"My stack looks

like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"

program2.c:5:61: Corresponding format code

program2.c:5:1: No argument corresponding to printf format code 9 (%p):

"My stack looks

like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"

program2.c:5:65: Corresponding format code

program2.c:5:1: No argument corresponding to printf format code 10 (%p):

"My stack looks

like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"

program2.c:5:69: Corresponding format code

program2.c:8:1: No argument corresponding to printf format code 1 (%p):

"My new stack looks

like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"

program2.c:8:36: Corresponding format code

program2.c:8:1: No argument corresponding to printf format code 2 (%p):

"My new stack looks

like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"

program2.c:8:40: Corresponding format code

program2.c:8:1: No argument corresponding to printf format code 3 (%p):

"My new stack looks

like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"

program2.c:8:44: Corresponding format code

program2.c:8:1: No argument corresponding to printf format code 4 (%p):

"My new stack looks

like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"

program2.c:8:48: Corresponding format code

program2.c:8:1: No argument corresponding to printf format code 5 (%p):

"My new stack looks

like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"

program2.c:8:52: Corresponding format code

program2.c:8:1: No argument corresponding to printf format code 6 (%p):

"My new stack looks

like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"

program2.c:8:56: Corresponding format code

program2.c:8:1: No argument corresponding to printf format code 7 (%p):

"My new stack looks

like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"

program2.c:8:60: Corresponding format code

program2.c:8:1: No argument corresponding to printf format code 8 (%p):

"My new stack looks

like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"

program2.c:8:64: Corresponding format code

program2.c:8:1: No argument corresponding to printf format code 9 (%p):

"My new stack looks

like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"

program2.c:8:68: Corresponding format code

program2.c:8:1: No argument corresponding to printf format code 10 (%p):

Shirish Shetty 2103164 C32

"My new stack looks
like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"

program2.c:8:72: Corresponding format code

program2.c:9:2: Path with no return in function declared to
return int

There is a path through a function declared to return a
value on which there

is no return statement. This means the execution may fall
through without

returning a meaningful result to the caller. (Use -freturnerror to
inhibit warning)

Finished checking --- 21 code warnings

Program 3 :

```
admin@admin-OptiPlex-3050:~/Desktop$ gcc program3.c -o p3.out
admin@admin-OptiPlex-3050:~/Desktop$ ./p3.out
Enter password: hellonandan
Failure
admin@admin-OptiPlex-3050:~/Desktop$ splint program3.c
Splint 3.1.2 --- 21 Feb 2021

program3.c: (in function get_password)
program3.c:10:5: Return value (type int) ignored: scanf("%s", buff)
    Result returned by function call is not used. If this is intended, can cast
    result to (void) to eliminate message. (Use -freturnerror to inhibit warning)
program3.c: (in function main)
program3.c:20:14: Parameter argc not used
    A function parameter is not used in the body of the function. If the argument
    is needed for type compatibility or future plans, use /*@unused@*/ in the
    argument declaration. (Use -fparamerror to inhibit warning)
program3.c:27: Parameter argv not used
program3.c:4:6: Variable exported but not used outside program3: password
    A declaration is exported, but not used outside this module. Declaration can
    use static qualifier. (Use -fexportlocal to inhibit warning)
program3.c:6:5: Function exported but not used outside program3: get_password
program3.c:14:1: Definition of get_password
program3.c:16:6: Function exported but not used outside program3: success
program3.c:18:1: Definition of success

Finished checking --- 6 code warnings
admin@admin-OptiPlex-3050:~/Desktop$
```

Extra Splint Programs and Outputs :

Buffer Overflow in Local Variables:

Buffer Overflow in Local Variables: When a program writes beyond the bounds of a local variable's buffer, causing stack corruption.

```
#include <stdio.h>
```

```
void vulnerable_function(char *input) {  
    char buffer[10];  
    printf("Before overflow: %s\n", buffer);  
    strcpy(buffer, input);  
    printf("After overflow: %s\n", buffer);  
}
```

```
int main() {  
    char input[20] = "Buffer Overflow!";  
    vulnerable_function(input);  
    return 0;  
}
```

```

root@kali:~# gcc extra1.c -o ex.out
extra1.c: In function 'vulnerable_function':
extra1.c:6:5: warning: implicit declaration of function 'strcpy' [-Wimplicit-function-declaration]
  6 |   strcpy(buffer, input);
     |   ^
extra1.c:2:1: note: include <string.h> or provide a declaration of 'strcpy'
  1 | #include <stdio.h>
     | +#include <string.h>
  2 |
extra1.c:6:5: warning: incompatible implicit declaration of built-in function 'strcpy' [-Wbuiltin-declaration-mismatch]
  6 |   strcpy(buffer, input);
     |   ^
extra1.c:6:5: note: include <string.h> or provide a declaration of 'strcpy'
root@kali:~# ./ex.out
Before overflow:
After overflow: Buffer Overflow!
Segmentation fault
root@kali:~# splint extra.c
Splint 3.1.2 --- 21 Feb 2021

Cannot open file: extra.c

Finished checking --- no code processed
root@kali:~# splint extra1.c
Splint 3.1.2 --- 21 Feb 2021

extra1.c: (in function vulnerable_function)
extra1.c:5:37: Passed storage buffer not completely defined (*buffer is
           undefined): printf (... , buffer, ... )
  Storage derivable from a parameter, return value or global is not defined.
  Use /*@out@*/ to denote passed or returned storage which need not be defined.
  (Use -comdef to inhibit warning)
extra1.c:3:6: Function exported but not used outside extra1:
           vulnerable_function
  A declaration is exported, but not used outside this module. Declaration can
  use static qualifier. (Use -exportlocal to inhibit warning)
  extra1.c:8:1: Definition of vulnerable_function

Finished checking --- 2 code warnings
root@kali:~#

```

Integer Overflow leading to Buffer Overflow:

Integer Overflow leading to Buffer Overflow: Occurs when an integer overflow condition leads to writing data beyond the bounds of a buffer, causing a buffer overflow.

```

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void vulnerable_function(int size) {

    char buffer[size];

    strcpy(buffer, "Buffer Overflow!");

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

}

int main() {

    int size = 10;

    vulnerable_function(size);

    return 0;
}

```

}

```
root@kali:~# gedit ext2.c
root@kali:~# gcc ext2.c -o ext2.out
root@kali:~# ./ext2.out
Buffer Overflow!
root@kali:~# splint ext2.c
Splint 3.1.2 --- 21 Feb 2021

ext2.c:5:6: Function exported but not used outside ext2: vulnerable_function
  A declaration is exported, but not used outside this module. Declaration can
  use static qualifier. (Use -exportlocal to inhibit warning)
  ext2.c:9:1: Definition of vulnerable_function

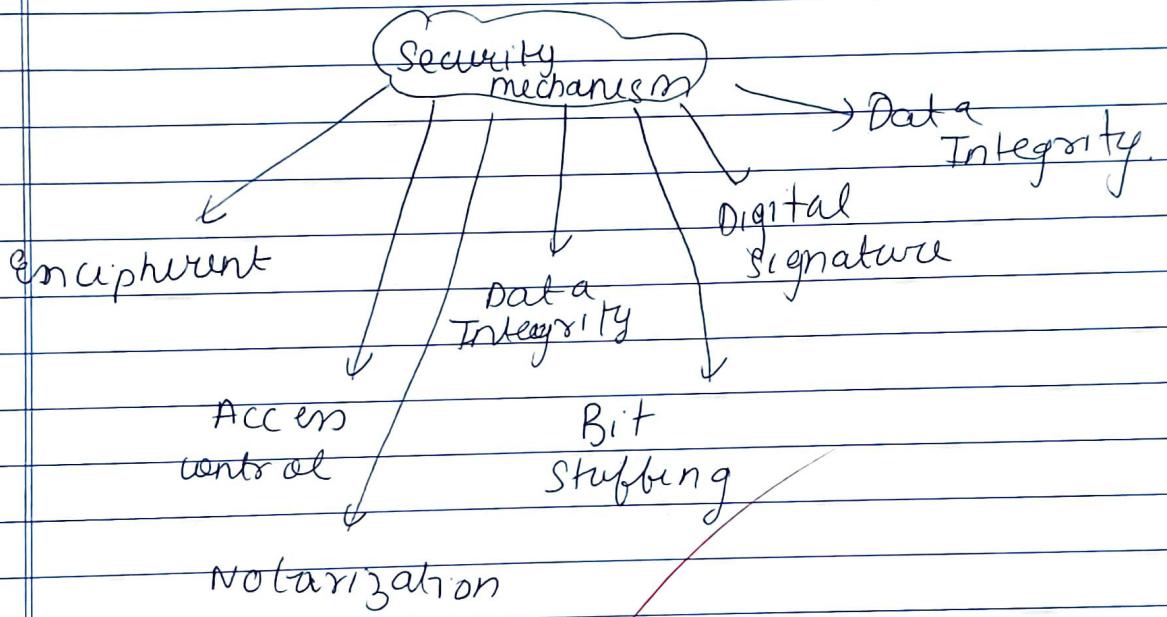
Finished checking --- 1 code warning
root@kali:~#
```

C32 -

Assignment - 1

* Security mechanism

- Network security field in computer technology that deals with ensuring security of computer network infrastructure.
- Security mechanism can also be termed as a set of processes that deal with recovery from security attack. Various mechanisms are designed to recover from these specific attacks at various protocol layer.



(1) Encipherment

→ Security mechanism deals with hiding & covering the data which helps data to become confidential. It is achieved applying mathematical calculation or algorithms which reconstruct information into not readable forms.

- Eg : - An source encryption message using protocol like PGP or S/MIME.
- A messaging app encryption conversation end to end such as WhatsApp or Signal.

② Access control :

→ This mechanism is used to stop unattended access to data which you are sending. It can be achieved by various techniques such as applying passwords, using firewalls.

- Eg : → An organization network requiring employees to authenticate with their credential before accessing sensitive system or data.

③ Authentication Exchange

→ This security mechanism deals with identity to known in communication. This is achieved at TCP/P layer where 2 way handshaking mechanism is used to ensure data is sent or not.

- Eg : → A website requiring user to enter username & password to access their account.

④ Notarization

This security mechanism involve use of trusted 3rd party in communication. This mediator keep record of request made by sender to receiver for later denied.

Eg:- A signed legal document such as contract, being witnessed by notary public to validate its authenticity & Identity of signature.

(5) Bit Stuffing :-

This security mechanism is used to add extra bits into data which being transmitted checks the Even & Odd parity.

Eg:- In HDLC, a "flag" sequence is inserted before & after data frame & bit stuffing is employed to avoid the occurrence of unintended flag sequences within the data.

(6) Data Integrity :-

- It appends value to data to which is created by data itself.

This similar to sending packet of information known to both sending & receiving parties and checked before & after data is received.

Eg:- checksum.

- verifying the integrity of downloaded files by comparing checksum generated before & after the transfer. If match, it ensures the data was corrupted during transmission.

Q2) What are various types of attack? Explain with example:

Types of Attacks :-

- ① Active Attacks.
- ② Passive Attacks.

① Active Attacks:-

- Types of cybersecurity attack in which an attacker attempts to alter, destroy or disrupt ~~or~~ normal operation of system.
- Involve the attacker taking direct action against target system.
- Can be more dangerous than passive attack, which involves simply monitoring eavesdropping on a system.

Types:-

- Masquerade
- Modification of message
- Repudiation
- Replay
- Denial of Service.

Passive Attacks:-

- Attempt to learn or make use of info from the system but also that does not affect system resources.
- Are in nature of eavesdropping.
- Goal of opponent is obtain info that is being transferred.
- Involve an attacker **passively** monitoring

destroying it

Eve : - Capturing : - Attacker listens in on network traffic to collect sensitive info.

Sniffing : - Attacker captures & analyzes data packet to steal info.

Types :-

- (1) Release of message content
- (2) Traffic analysis.

Q3) List few viruses on net & explain

(i) Ransomware :-

This type of malware encrypt user files and demand payment for their release. Recent variants have become increasingly sophisticated, targeting both individual and organizations.

(ii) Spyware :-

Spyware is designed to secretly observe and collect information about a user's browsing habits, passwords & other sensitive data. It can be harder to detect.

(iii) Trojans :-

Trojans disguise themselves as legitimate software but carry out malicious activities once installed on a device. They can create

or even turn the infected device into a botnet.

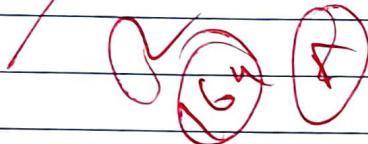
4.) Fileless malware :-

This type of malware operates in system memory w/o leaving traces on the hard drive.

It can be harder to detect and remove compared to traditional malware.

5.) Mobile malware:-

With the rising use of smartphone, malware targeting mobile devices, has become prevalent. This include malicious apps that steal personal info, send SMS etc



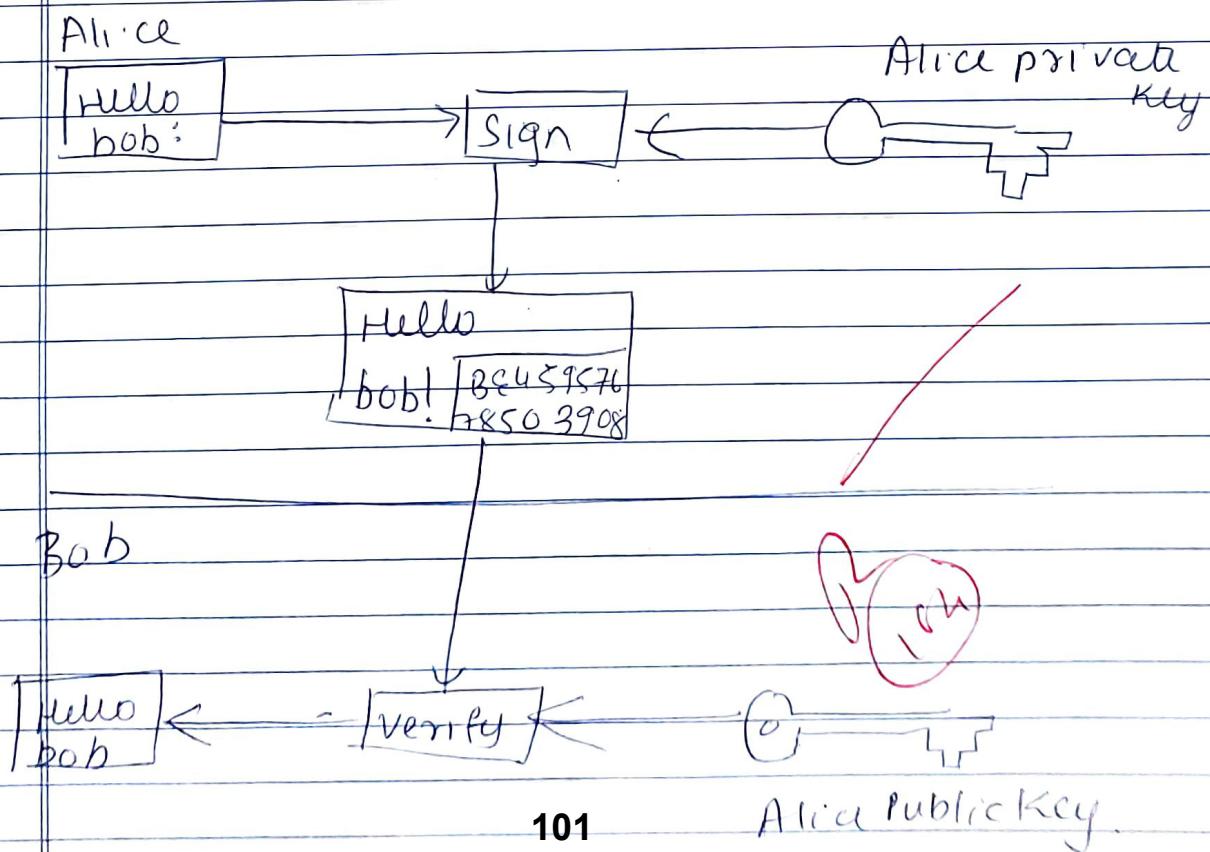
Assignment - 2.

* Digital Signature : -

In cryptography and system security are essential for ensuring data integrity & authenticity. These signature use public key algorithm to provide security.

When data is signed with digital signature, it can be verified by other, providing that the data originated from signer and was not altered.

Digital signature involve key and encryption using private key, hash algorithm like SHA 256 etc. Verification of digital signature involves using public key to confirm the authenticity of signed data.



Key components of digital signature include key generation, signing algorithms and signature verification algorithms. These schemes rely on public key cryptography, ensuring valid signature can be generated with private key. Digital signatures are used to validate messages, email, contracts and other digital data, providing security, authenticity and non-repudiation.

Digital signatures work by creating a unique digital fingerprint for each user or entity, enhancing transparency in online interactions and building trust between parties. They involve hash function, public key distribution, digital certificates issued by certificate authorities play a crucial role in verifying identities and associating public keys with individuals or entities.

Q) Digital certificates:-

Digital certificates play a crucial role in CSS by verifying the authenticity of digital message, devices & individuals. These certificates are electronic password that utilize public key infrastructure to ensure secure communication & data exchange online. They are issued by CA and serve two primary functions:- verifying the sender/receiver and providing encryption/decryption capabilities of data transmission.

Root CA certificate

Root CA public key

Root CA signature.

Issuer

CA certificate

CA's public key

Root CA signature

Issuer

Service

certificate

Service public key

CA signature

There are three main types of digital certification

- 1 Individual digital signature - Used to identify a person, sign electronic document, email etc
- 2 Service certification - Identify server & ensure secure communication over network
- 3 Encryption certificates - Used encrypt message using recipient's public key of data & identify during transmission.

Digital certificates are integral component of public key infrastructure, managing the distribution, authentication & revocation of these certificates. Digital certificates are essential for establishing trust in online interaction, securing website using SSL certificates, encryption emails & ensuring data integrity through digital signatures.

In summary, digital certificates are vital tool in CSS, providing

- authentication
- encryption
- Trust mechanism for secure electronic communication & data exchange in various online environments

Assignment No - 3

(Q1) Explain Some Software Vulnerabilities. How are they exploited to launch an attack.

Ans 1.) Buffer :- when a program writes more overflow data to a buffer than it can hold, causing it to overbounce into adjacent memory

2.) SQL Injection :- Attacker insert malicious SQL code into the input fields, exploiting vulnerabilities in database queries to access or manipulate database queries to access or manipulate data

3.) Cross Site Scripting :- Attacker inject malicious script into web pages viewed by other users, compromising their security or stealing the data

4.) Cross-site Request Forgery (CSRF) :- Attacker trick user into performing unintended actions on a web application where they are authenticated.

5.) Remote Code Execution :- Vulnerabilities that allow attackers to execute arbitrary code on a remote Server or a System.

6.) Directory Traversal :- Attacker manipulates input to gain unauthorized access to file and directory outside the web server's root directory.

7.) Information Disclosure :- Vulnerabilities that expose sensitive information such as passwords, user data or system configuration.

8.) Authentication Bypass :- Flaws that allow attacker to gain unauthorized access to system or application without providing valid credentials.

* Exploitation to launch an attack

1.) Buffer Overflow :- Attacker exploits programs that do not properly validate input size, allowing them to overflow the buffer allocated for a variable.
- By sending input that exceeds the buffer's full capacity, attacker overwrites adjacent memory addresses with malicious code or data.

- These overwritten data can include instructions to execute arbitrary commands or launch other attack, potentially leading to remote code execution on system.

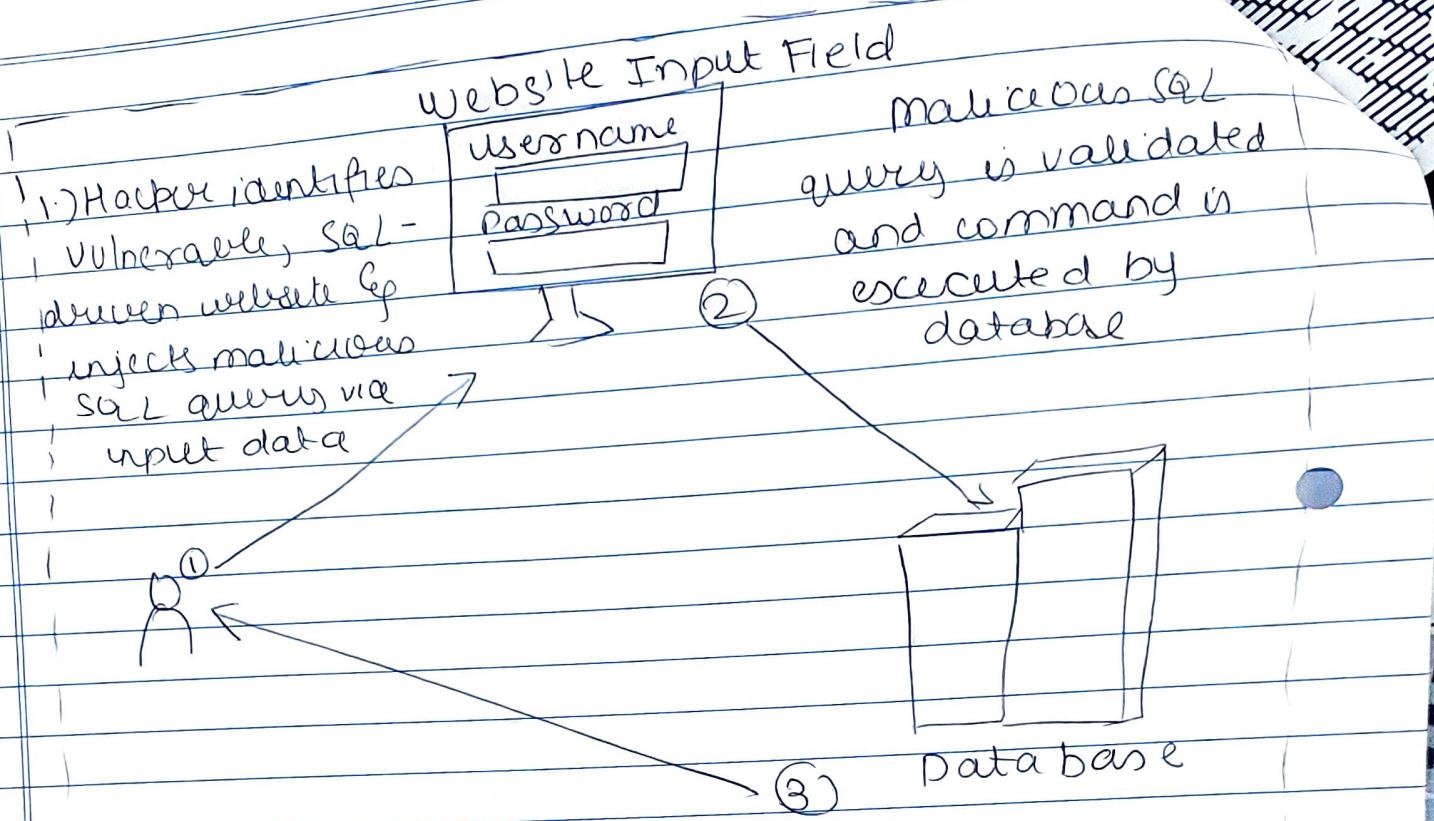
* Cross-Site Scripting (XSS)

- Attacker inject malicious script into web pages viewed by other users
- (Q) - These scripts can steal cookies, session token or other sensitive data from user, compromising their security.

Q.] Explain In Brief SQL Injection.

Ans → SQL Injection (SQLi) is web security vulnerability that allows an attacker to interface with the queries that an application makes to its database.

- They allow attacker to view data that they are not normally able to retrieve.
- In many cases, an attacker can modify or delete this data, causing persistent change to the application content or behaviour.
- In some situations, an attacker can escalate a SQL injection attack to compromise the underlying server or other backend infrastructure. It can also enable them to perform denial of service attack.



3) Hacker is granted access to view & alter records or potentially acts as a database administrator

* Impact of SQL injection attack

A successful SQL injection attack can result in unauthorized access to sensitive data, such as:

- (i) password
- (ii) credit card details
- (iii) personal user information

* How to detect SQL injection vulnerabilities

You can detect SQL injection manually using systematic of test against every entry point in the application. So this, you can usually submit typically submit.

- (i) The single quote character ('') & look for errors or other anomalies
- II (ii) Some SQL specific syntax that evaluates its base value to base value of entry point, different value & look for systematic difference in application response
- (iii) Boolean condition such as OR 1 = 1 & OR 1 = 2, and look for differences in application responses
- (iv) Payloads designed to trigger time delay when executed within a SQL query & look for difference in time taken to respond
- III (v) OAST payloads designed to trigger an out of bound network interaction when executed within a SQL query, and monitor any resulting interaction

* SQL injection example

SQL
DB

- (i) Retrieving hidden data, where you can modify a SQL query to return additional result
- (ii) Subverting application logic, where you can change a query to interface with application's logic
- (iii) UNION attack, where you can retrieve data from different database table.
- (iv) Blind SQL injection, where the results of a query you control are not returned in the application's response.