Thadomal Shahani Engineering College

Bandra (W.), Mumbai - 400 050.

CERTIFICATE

Certify that Mr./Miss Parth Sandeep Dabholkar

of Computer Engg. Department, Semester VI with

Roll No. 2103032 has completed a course of the necessary

experiments in the subject Cryptography and System Security (CSS) under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year 2023 - 2024

Teacher In- Charge 16/4/24

Head of the Department

Date 16/4/24

Principal

THADOMAL SHAHANI TSEC ENGINEERING SOLLEGE

CONTENTS

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* = *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:
- Gcd(a,b) = s * a + t * b

Here's a simple example:

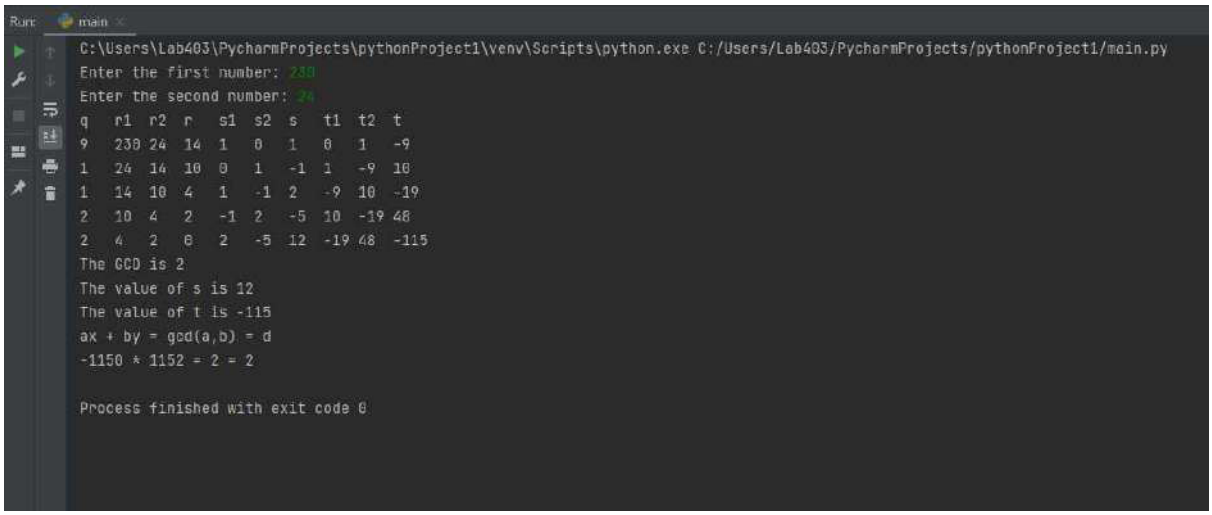
Let's find the GCD of a=35 and b=15.

1. Initialization:
   - s=0,old_s=1
   - t=1,old_t=0
   - r=15,old_r=35
2. Iteration:
   - q=floor(35/15)=2, r=35−2×15=5
   - Update: old_r=15,r=5
   - Update: old_s=0,s=1
   - Update: 0−2×1=−2, old_t=1,t=−2
   - Next iteration: q=floor(515)=3, r=15−3×5=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 gcd(35,15)=1×35+(−2)×15=35−30=5gcd(35,15)=1×35+(−2)×15=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\tr1\tr2\tr\ts1\ts2\ts\tt1\tt2\tt")
while(r != 0):
    print(f"{q}\t{a}\t{b}\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"{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:

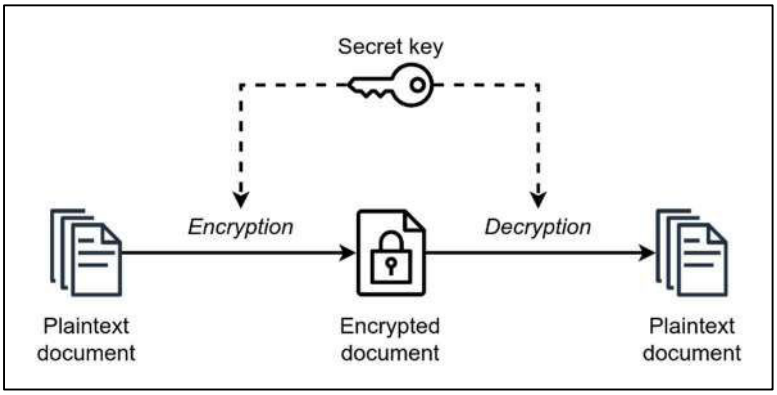


EXPERIMENT NO: 2

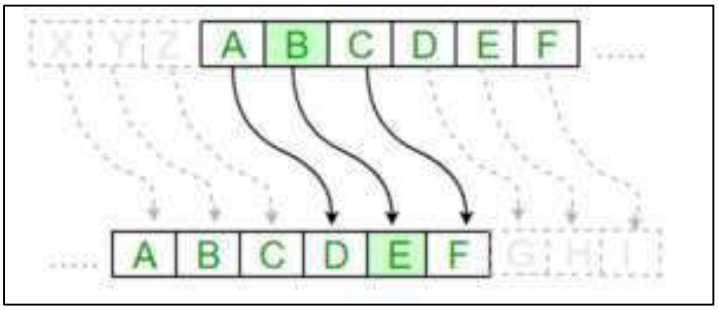
AIM: Implementation of Ceaser Cipher

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 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:

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

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.

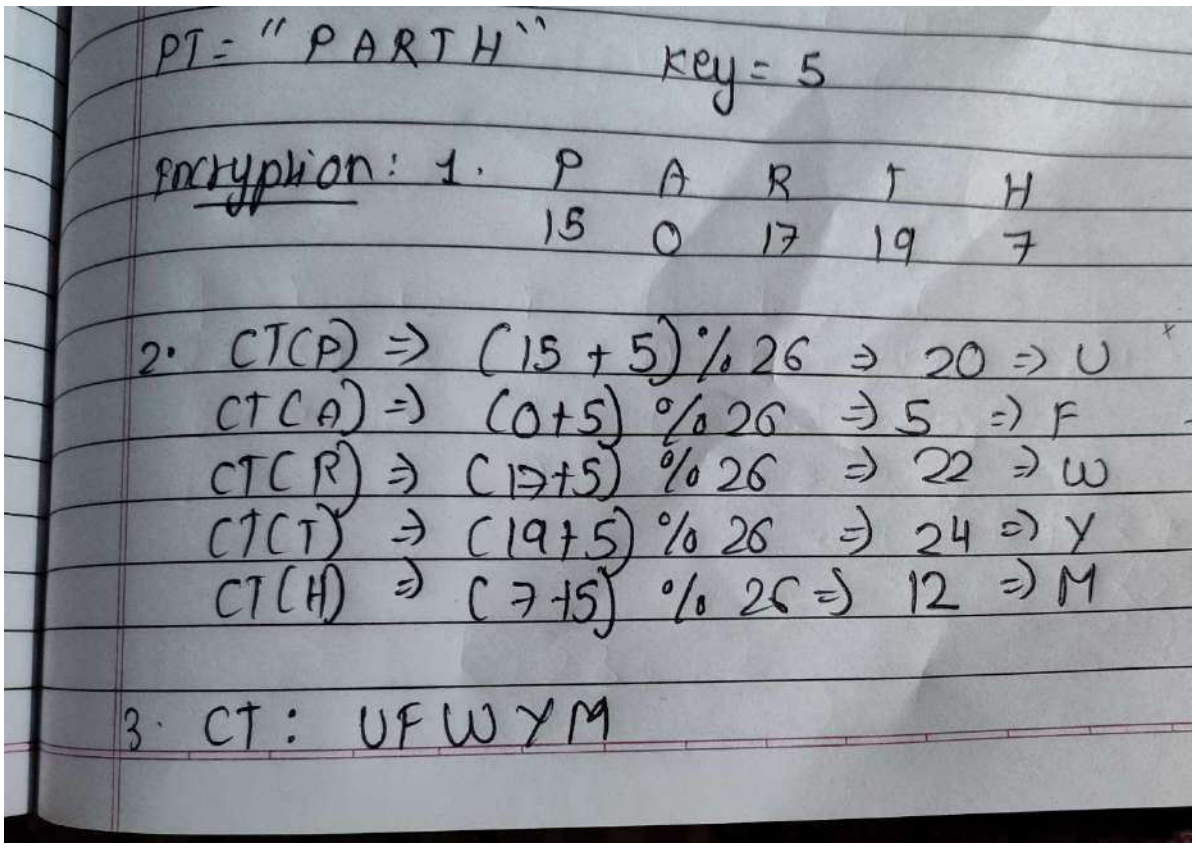
Decryption in Ceaser Cipher is done using the following steps:

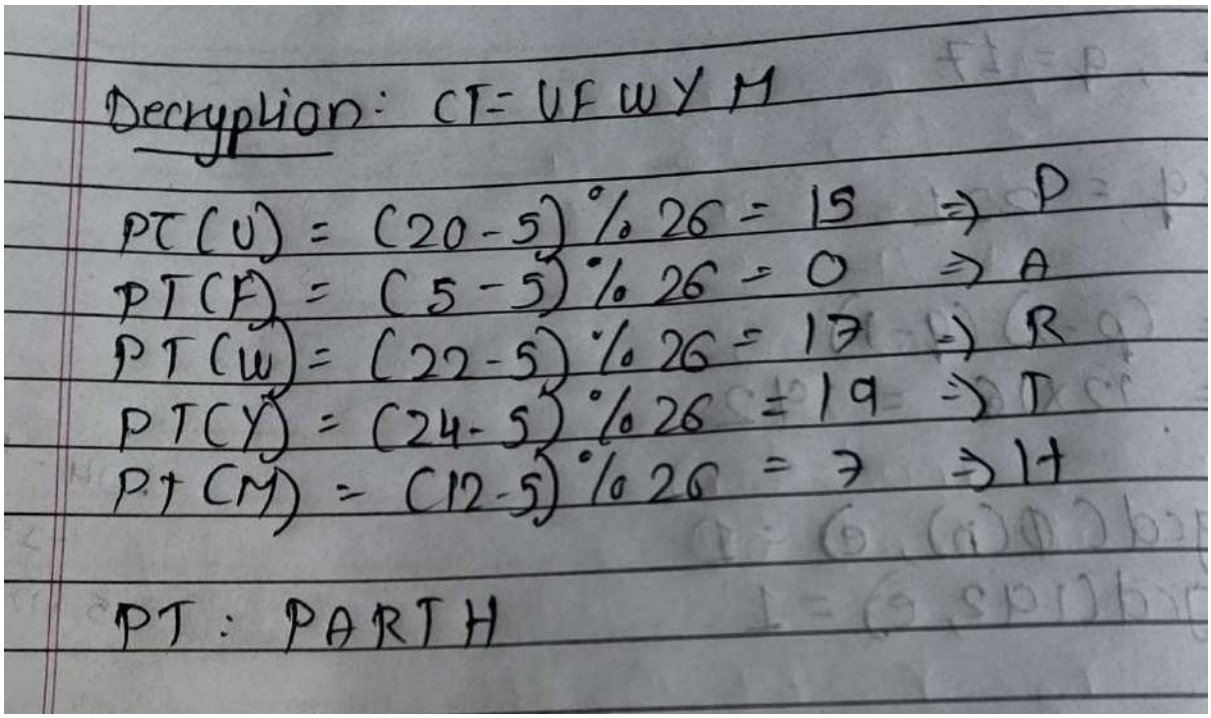
1. Convert the Cipher text characters into corresponding indices using the above table.
2. After getting the indices use the following formula:

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

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

EXAMPLE:

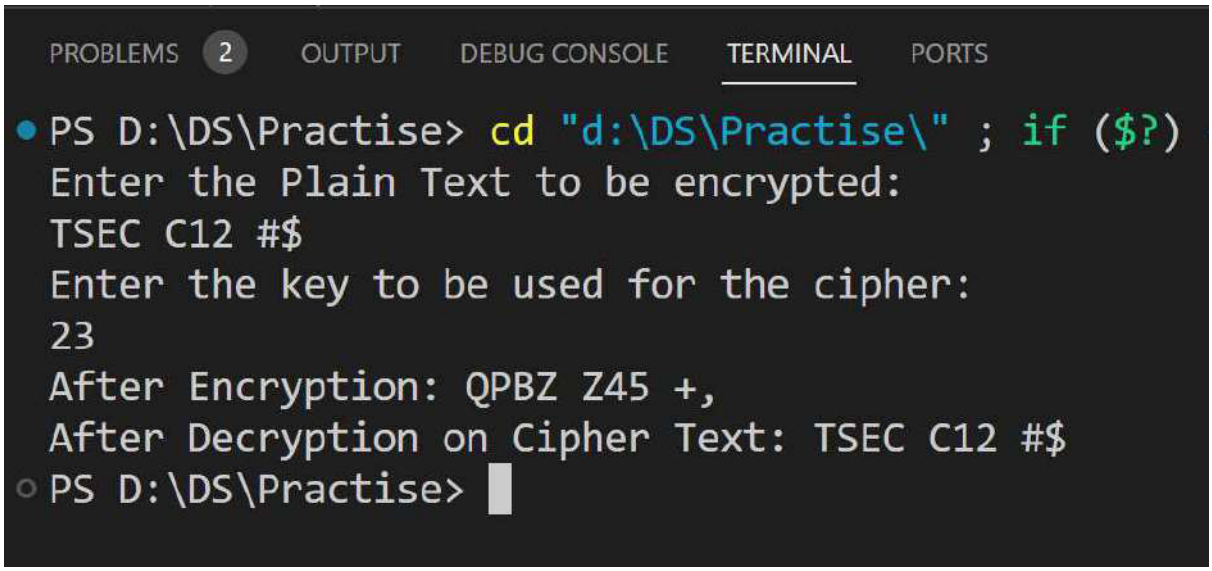

PT = "PARTH" key = 5
Encryption: 1.
P A R T H
15 0 17 19 7
2. CT(P) ⇒ (15 + 5) % 26 ⇒ 20 ⇒ U
CT(A) ⇒ (0+5) % 26 ⇒ 5 ⇒ F
CT(R) ⇒ (17+5) % 26 ⇒ 22 ⇒ W
CT(T) ⇒ (19+5) % 26 ⇒ 24 ⇒ Y
CT(H) ⇒ (7+5) % 26 ⇒ 12 ⇒ M
3. CT: UFWYM


Decryption: CT = UFWYM
PT(U) = (20-5) % 26 = 15 ⇒ P
PT(F) = (5-5) % 26 = 0 ⇒ A
PT(W) = (22-5) % 26 = 17 ⇒ R
PT(Y) = (24-5) % 26 = 19 ⇒ T
PT(M) = (12-5) % 26 = 7 ⇒ H
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);
   }
}
```


PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
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 it's letters must not repeat in the matrix. For example if the key text is "SECURITY" then the matrix will be filled as:

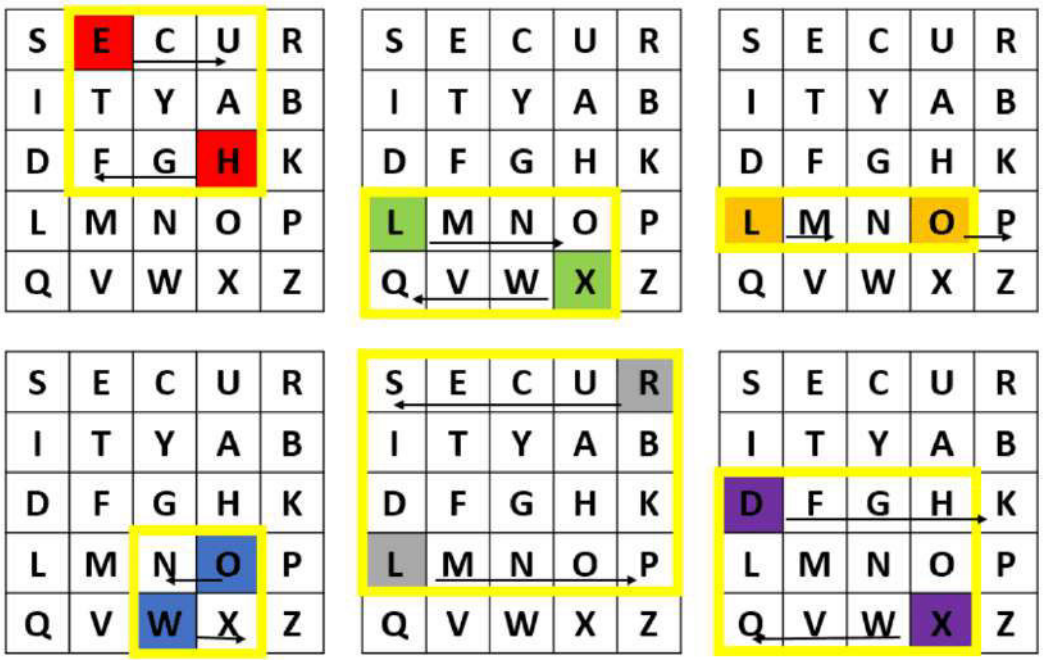
S	E	C	U	R
I	T	Y		

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.

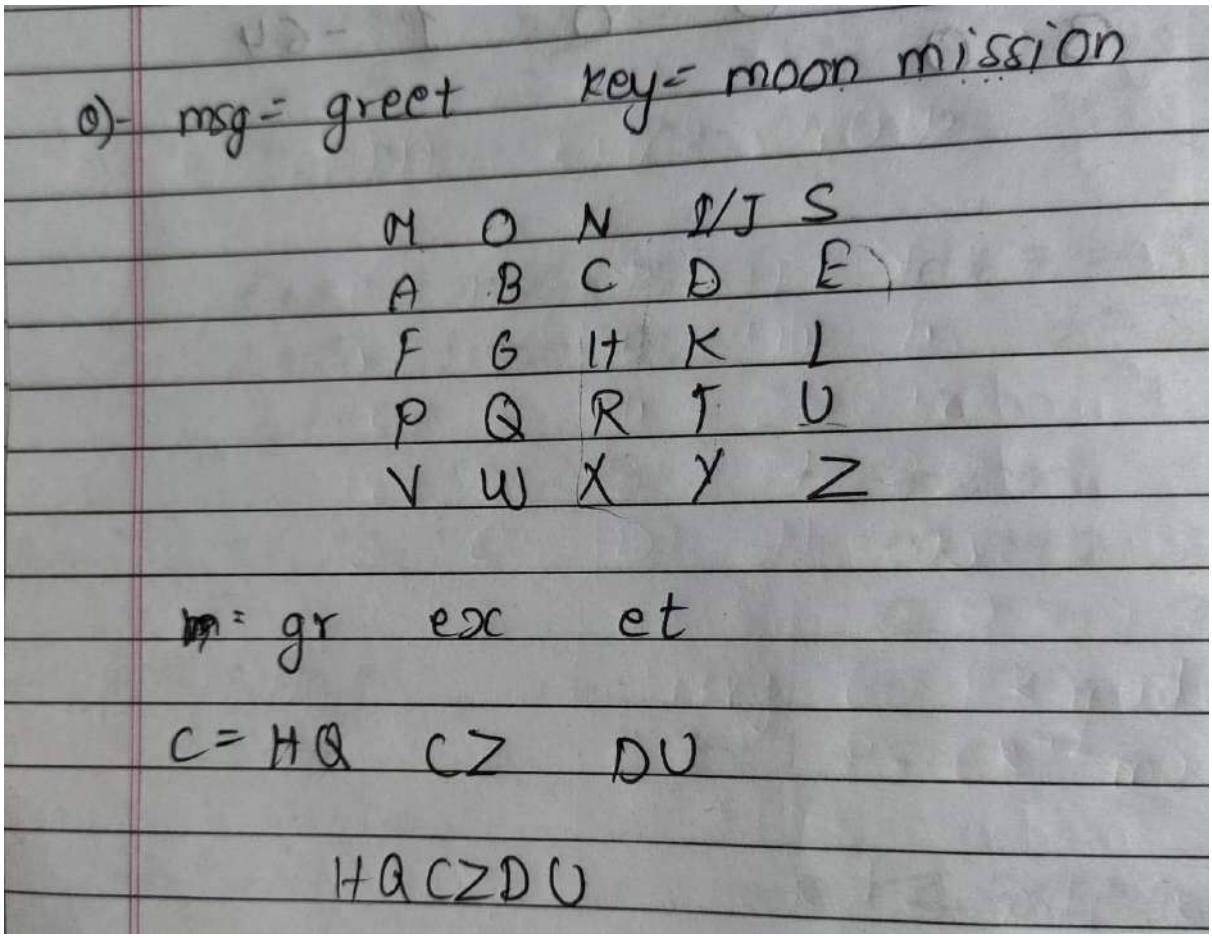
3. Since the there are only 25 cells in the matrix, only 25 alphabets can be filled in the matrix. Therefore, I and J are placed in the same cell and can be interchanged during encryption.

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:
   a. If the Letter in the pair is in the same row, then consider the immediate next letter.
   b. If the Letter in the pair is in the same column, then consider the immediate next letter.
   c. Make a rectangle in the matrix and then consider the extremes in the rectangle.



Example:


Q)- msg = greet key = moon mission
M O N I/J S
A B C D E
F G H K L
P Q R T U
V W X Y Z
m = gr ex et
C = HQ CZ DU
HQCZDU

```
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 encrpytPlayFair(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 = encrpytPlayFair(plainText, keyArr);
        System.out.println("String after Encyption is: "+encString);
    }
}
```

OUTPUT:

```
PS D:\DS\Practise>  & 'C:\Program Files\Java\jdk-
s\parth\AppData\Roaming\Code\User\workspaceStorag
'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 Encyption is: IBBWEHZAIZ
PS D:\DS\Practise>
```

EXPERIMENT NO: 4

AIM: Implementation of Euler's Totient Function.

THEORY: Euler's Totient function, ϕ(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 ϕ(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 ϕ(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, ϕ(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:

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

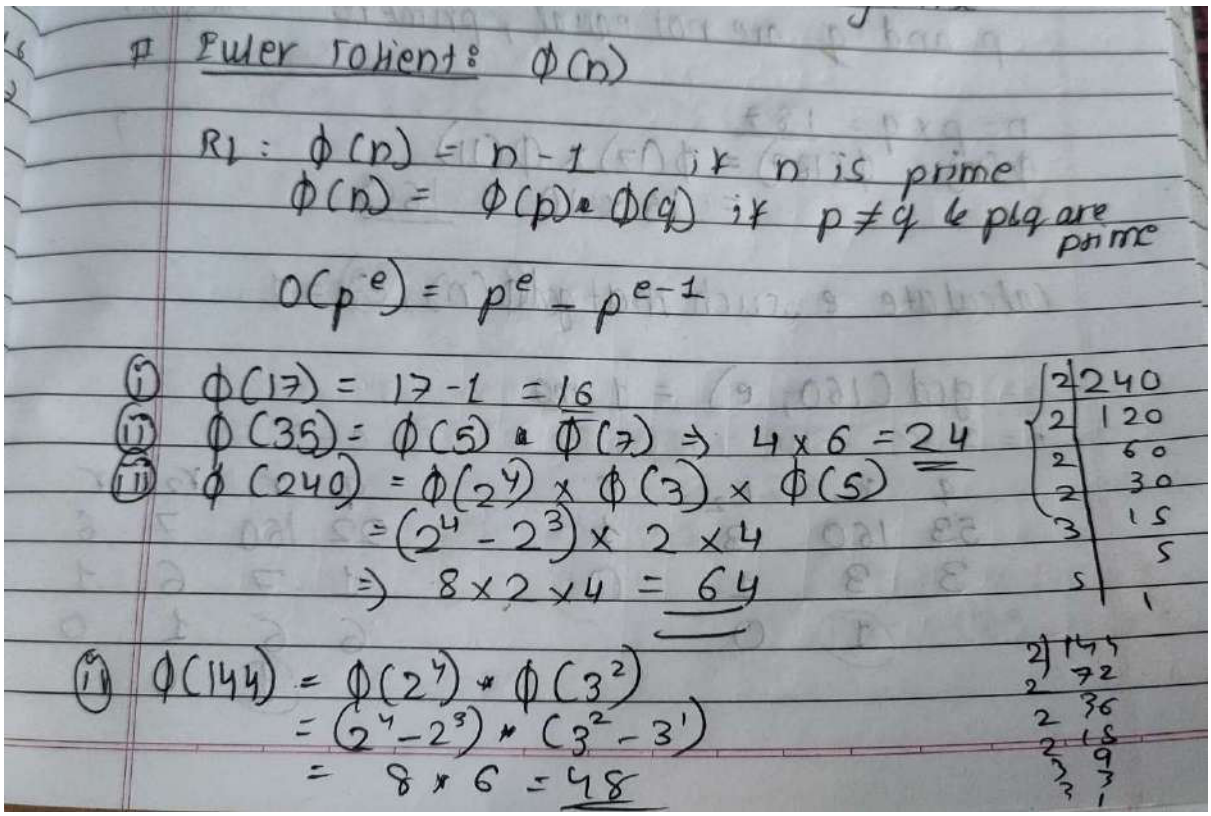
2. ***If the number is not prime but factorizable:***
   If the number is not a prime number, factorize the number into its prime factors. Suppose for a given number the factors are p and q then Euler's Totient is given as:

$$\phi(n) = \phi(p*q) = (p-1)*(q-1)$$
$$p \text{ must not be equal to } q$$

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:



```
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.

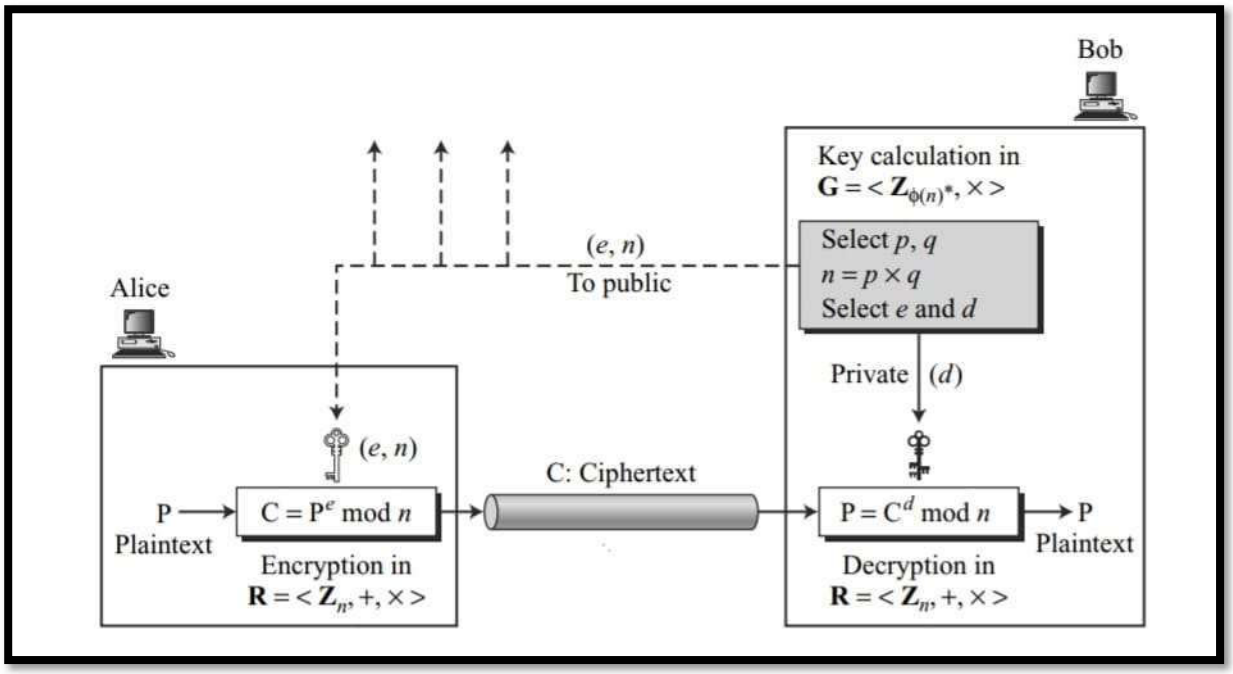
2. Integrity:

   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:

   a. Choose distinct prime numbers p and q.

   b. Compute n = p * q.

   c. Compute ϕ(n) = (p-1) * (q-1).

   d. Choose an integer e such that 1 < e < ϕ(n) and gcd(e, ϕ(n)) = 1.

   e. Compute the modular multiplicative inverse d of e modulo ϕ(n), such that d * e ≡ 1 (mod ϕ(n)).

2. Encryption:

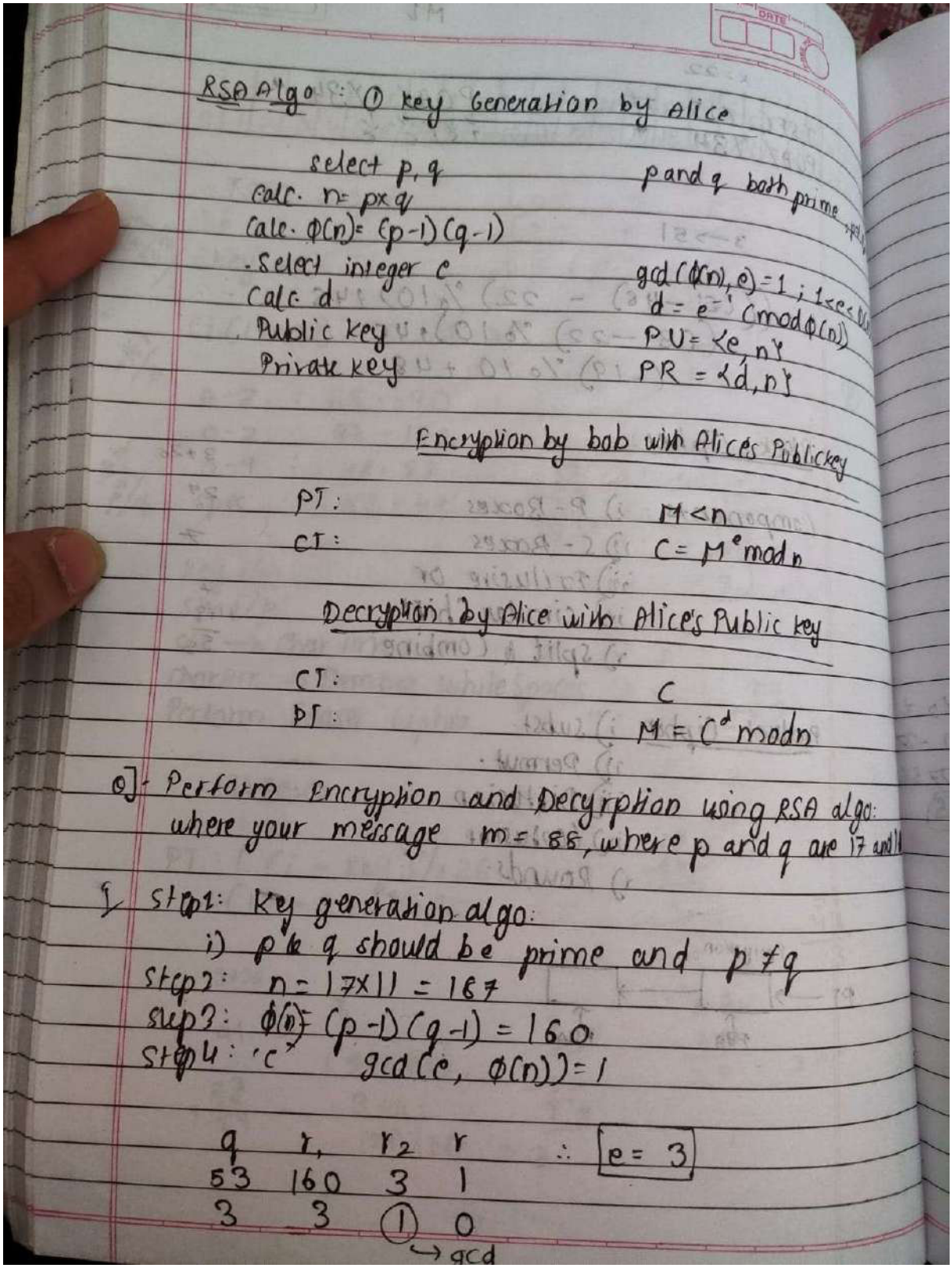
   a. Convert the plaintext message M into an integer m such that 0 ≤ m < n.

   b. Compute the ciphertext c ≡ m^e (mod n).

3. Decryption:

   a. Decrypt the ciphertext c using the private key d to recover the plaintext m ≡ c^d (mod n).

EXAMPLE:



RSA Algo : ① Key Generation by Alice

select p, q — p and q 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 — $d = e^{-1} \pmod{\phi(n)}$
Public key — $PU = \{e, n\}$
Private key — $PR = \{d, n\}$

Encryption by bob with Alices Publickey

PT: — $M < n$
CT: — $C = M^e \bmod n$

Decryption by Alice with Alices Public key

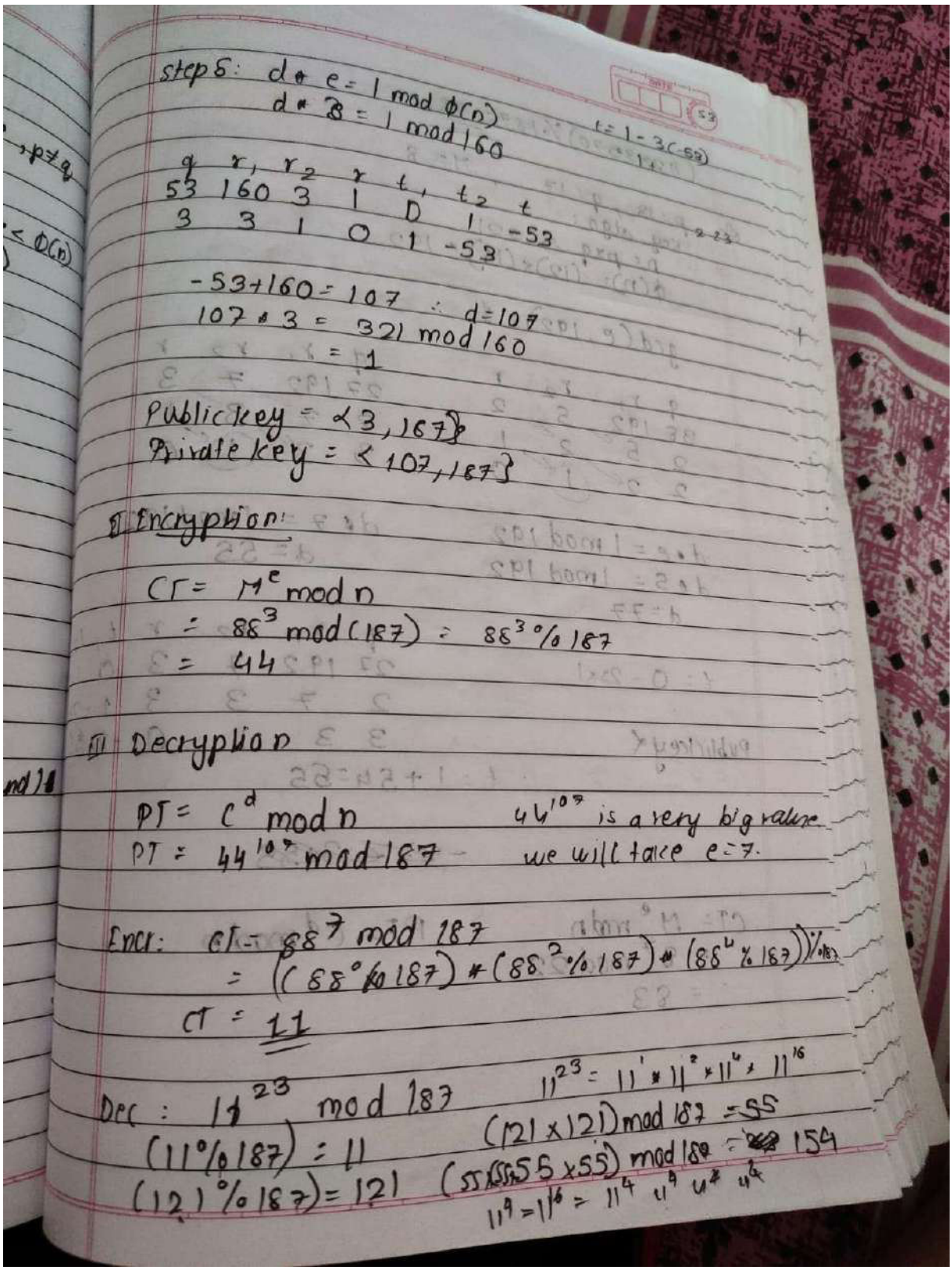
CT: — C
PT: — $M = C^d \bmod n$

Q]- Perform Encryption and Decryption using RSA algo: where your message m = 88, where p and q are 17 and 11

1. Step1: Key generation algo:
   i) p & q should be prime and $p \neq q$
   Step2: $n = 17 \times 11 = 187$
   Step3: $\phi(n) = (p-1)(q-1) = 160$
   Step4: 'e' — $\gcd(e, \phi(n)) = 1$

q	r_1	r_2	r
53	160	3	1
3	3	(1)	0
		→ gcd	

∴ e = 3



step 5: $d * e = 1 \bmod \phi(n)$

$d * 3 = 1 \bmod 160$

q	r_1	r_2	r	t_1	t_2	t
53	160	3	1	0	1	-53
3	3	1	0	1	-53	

$-53 + 160 = 107 \therefore d = 107$

$107 * 3 = 321 \bmod 160$

$= 1$

Public key = $\{3, 187\}$

Private key = $\{107, 187\}$

I Encryption:

$CT = M^e \bmod n$

$= 88^3 \bmod (187) = 88^3 \% 187$

$= 44$

II Decryption

$PT = C^d \bmod n$

$PT = 44^{107} \bmod 187$

44^{107} is a very big value we will take e = 7.

Encr: $CT = 88^7 \bmod 187$

$= ((88^2 \% 187) * (88^2 \% 187) * (88^3 \% 187)) \% 187$

$CT = 11$

Dec: $11^{23} \bmod 187$

$(11 \% 187) = 11$

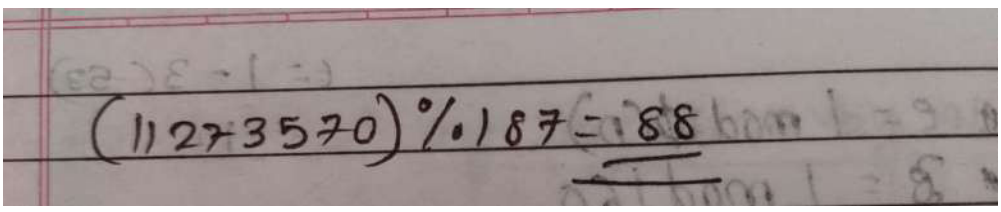
$(121) \% 187 = 121$

$11^{23} = 11^1 * 11^2 * 11^4 * 11^{16}$

$(121 \times 121) \bmod 187 = 55$

$(55 \times 55) \bmod 187 = 154$

$11^8 = 11^{16} = 11^4 \; 11^2 \; 11^2 \; 11^4$



$(11273570) \% 187 = 88$

```
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\tt1\tt2\tt");
    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 Decyrption: "+pt);
    }
  }
}
```

```
PROBLEMS 3   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS D:\DS\Practise>  d:; cd 'd:\DS\Practise'; & 'C:\Prog
ilsInExceptionMessages' '-cp' 'C:\Users\parth\AppData\R
e27fd647\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 Decyrption: 2
PS D:\DS\Practise>
```

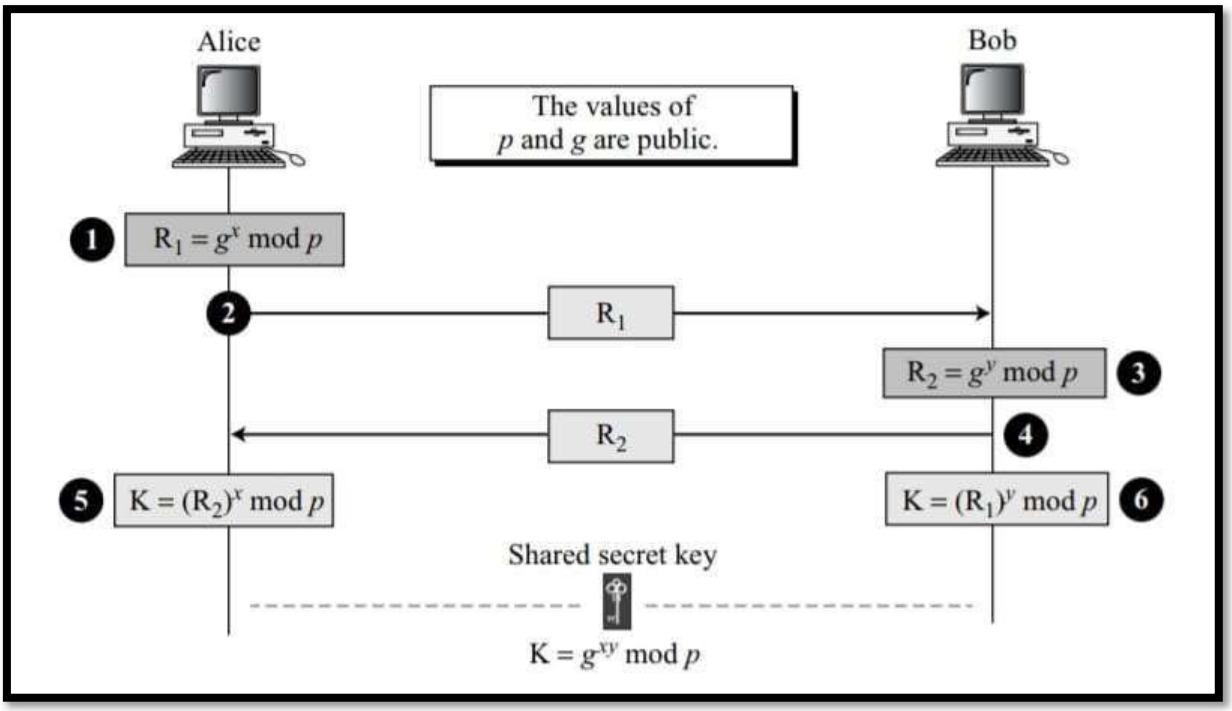
EXPERIMENT NO: 6

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 <Zp*, ×>. These two (group and generator) do not need to be confidential. They can be sent through the Internet; they can be public.



Diagramatic Representation of Diffie Hellman Key Exchange Algorithm

ALGORITHM:

The steps are as follows:

1. Alice chooses a large random number x such that **0 ≤ x ≤ p – 1** and calculates

$R1 = g^x \bmod p.$

2. Bob chooses another large random number y such that **0 ≤ y ≤ p – 1** and calculates

$R2 = g^y \bmod p.$

3. Alice sends R1 to Bob. Note that Alice does not send the value of x; she sends only R1.

4. Bob sends R2 to Alice. Again, note that Bob does not send the value of y, he sends only R2.

5. Alice calculates **$K = (R2)^x \bmod p.$**

6. Bob also calculates **$K = (R1)^y \bmod p.$**

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

2^1 mod 5 = 2

2^2 mod 5 = 4

2^3 mod 5 = 3

2^4 mod 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

$R_1 = 3^2 \bmod 7 = 2$
$R_2 = 3^3 \bmod 7 = 6$

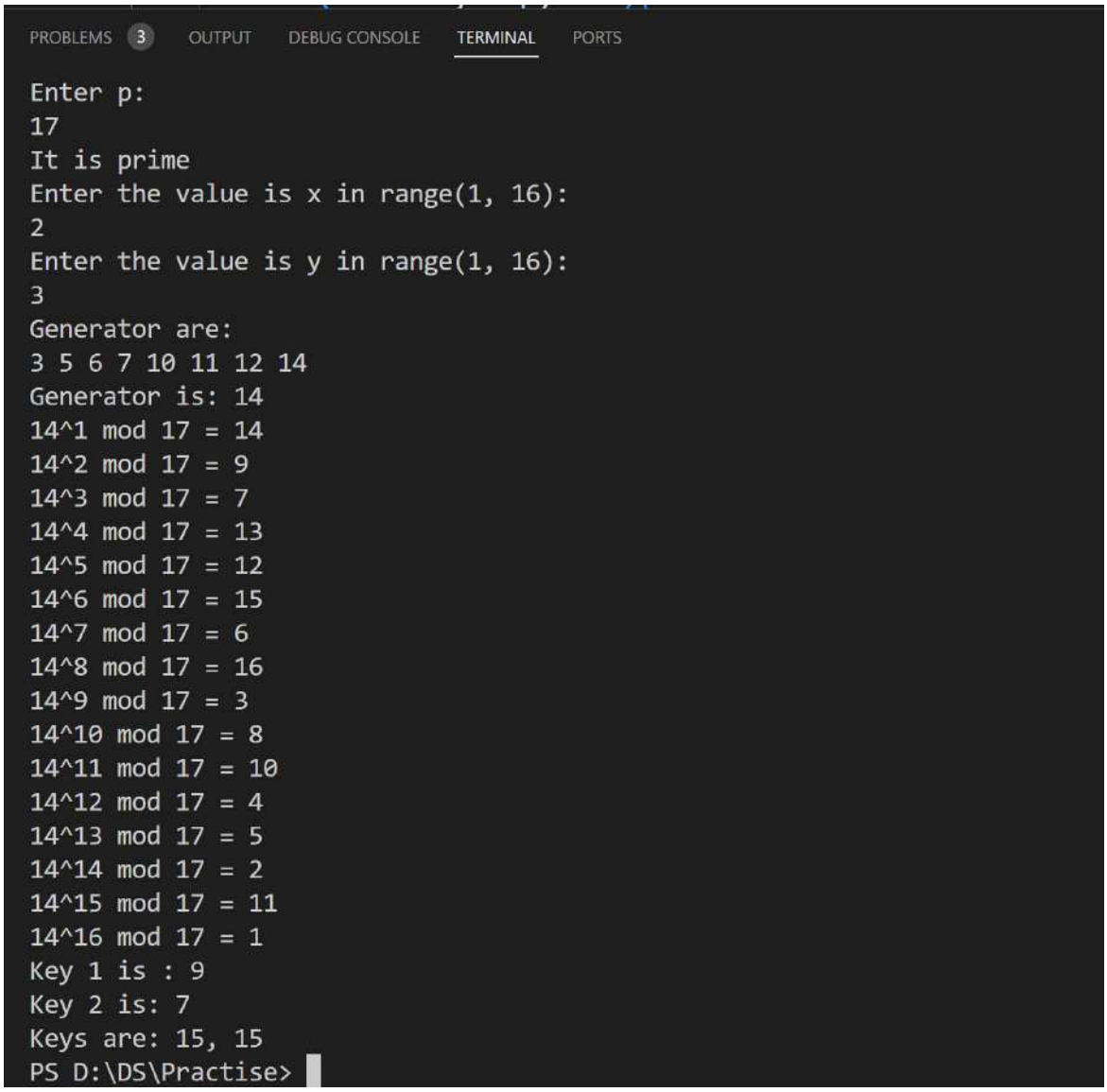
key for A $= 6^2 \bmod 7 = 1$
key for B $= 2^3 \bmod 7 = 1$

```
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;

    }
}
```


PROBLEMS 3
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
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 NO: 7

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

Steps:

1. Open ubuntu terminal.

2. Get root access by typing “sudo su root”. Put the pc password.

3. Install the tool using the following command

#apt-get install whois

#apt-get install dig

#apt-get install traceroute

#apt-get install nslookup

1. **WHOIS:**
   The WHOIS command in Linux is a network utility that allows you to retrieve information about domain names, IP addresses, and autonomous system numbers. When you run the WHOIS command followed by a domain name or IP address, it queries the WHOIS database maintained by registrars and registries to provide details such as the owner of the domain, registration and expiration dates, contact information, and more.

   Command: whois tsec.edu

   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.

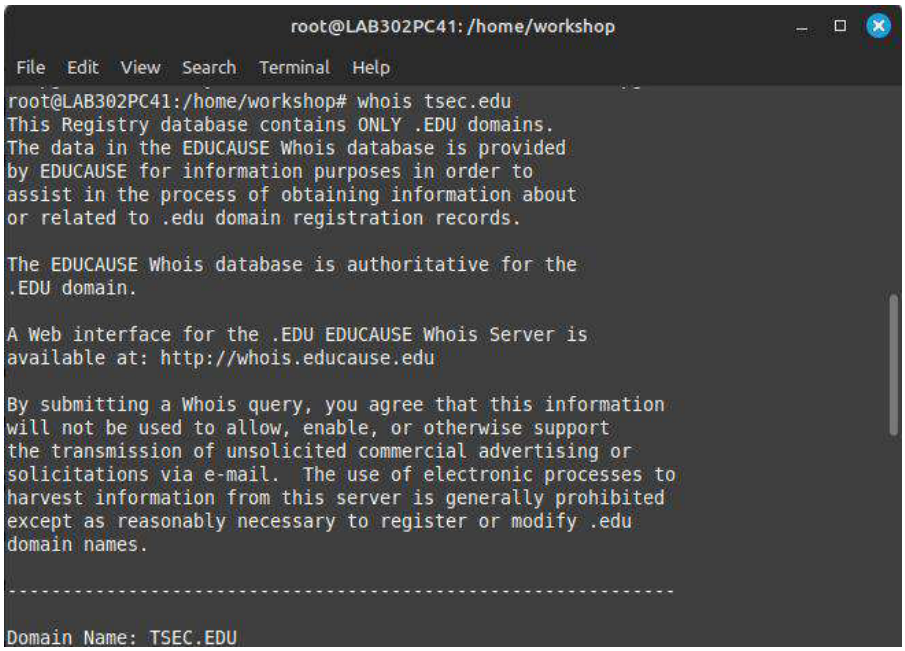
   3. Registrant Information: This may include the name, address, email, and phone number of the person or organization that registered the domain.

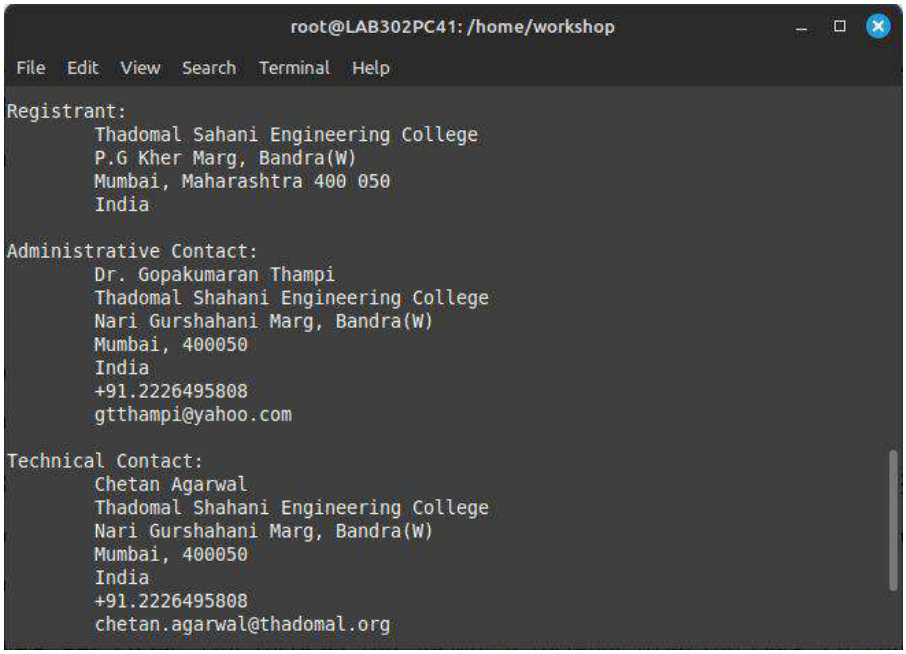
   4. Name Servers: These are the authoritative servers responsible for resolving the

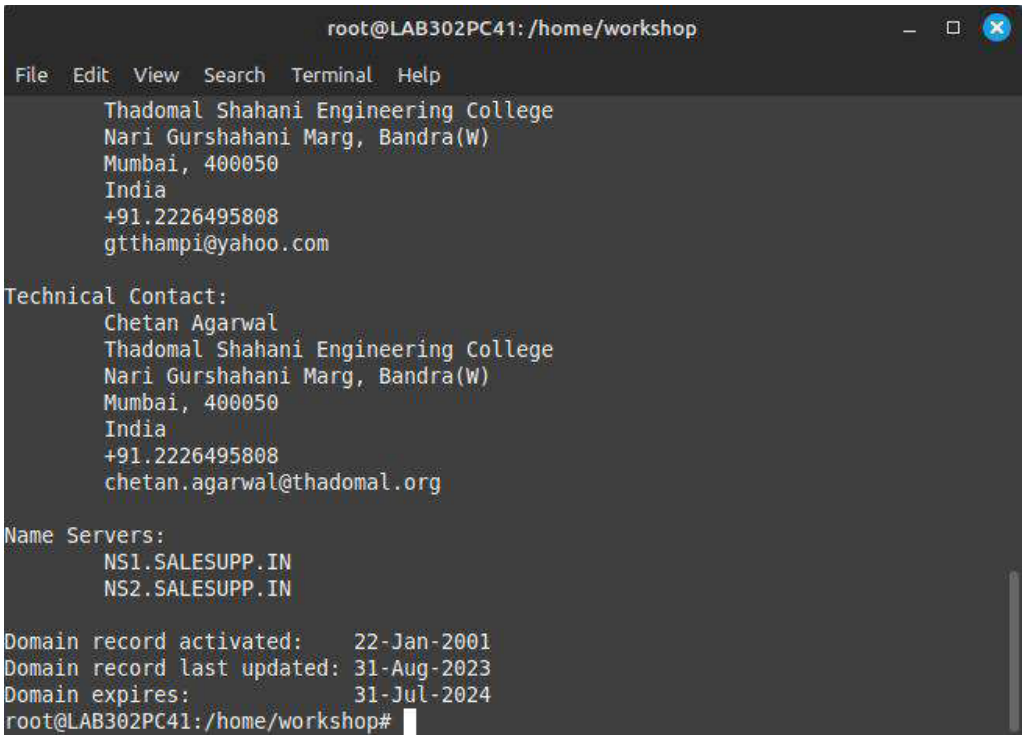
domain to its associated IP address.

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.







2. **DIG:**

Dig (domain information groper) is a network administration command-line tool for querying Domain Name System (DNS) name servers. Dig is useful for network troubleshooting and for educational purposes.
When you pass a domain name to the dig command, by default it displays the A record (the ip-address of the site that is queried).

Command: dig www.geeksforgeeks.com

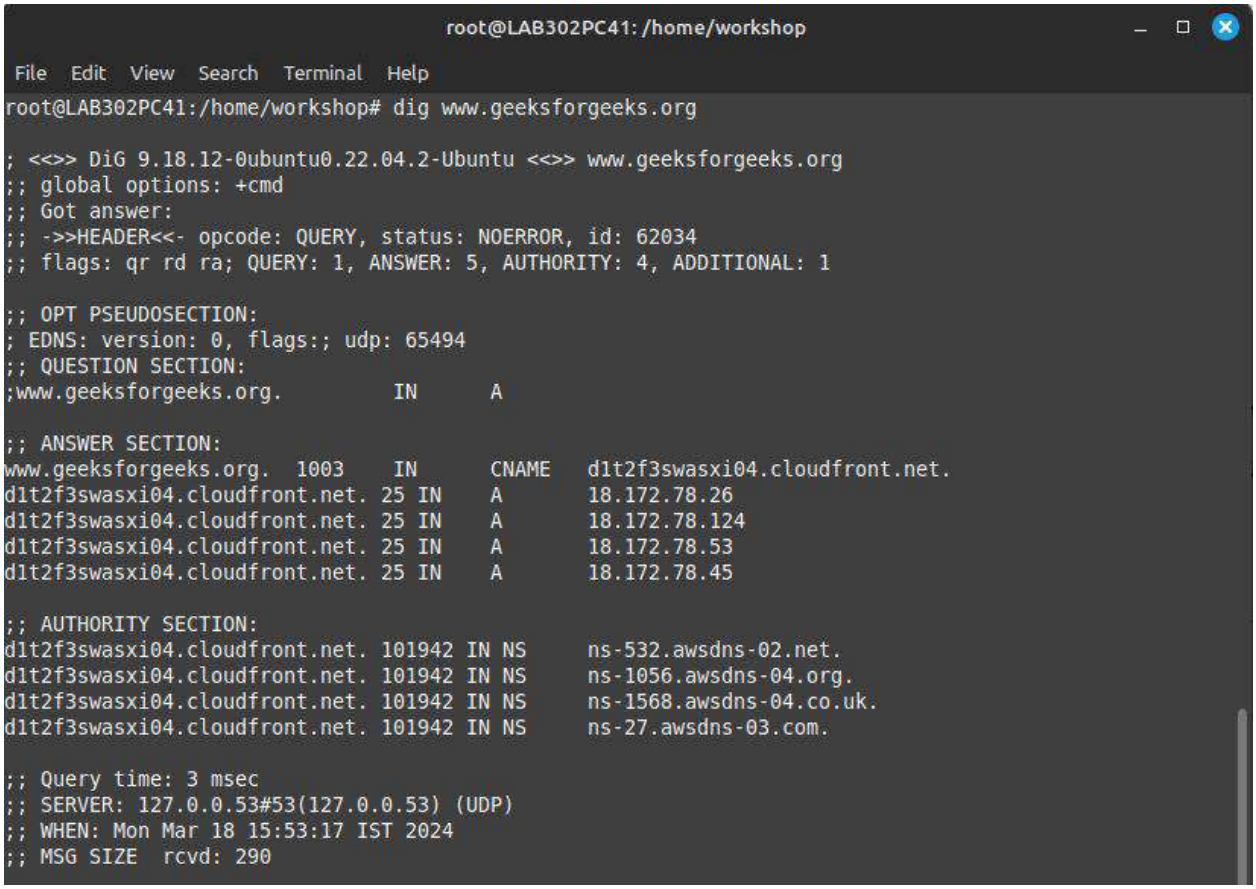
Sure, let's break down each section:

1. Question Section: This section of a DNS response contains the question that was asked to the DNS server. It includes the domain name that was queried and the type of record being requested (such as A, AAAA, MX, etc.). Essentially, it shows what information the client is asking for.

2. Answer Section: The answer section of a DNS response contains the actual answer to the query. It includes the resource records that match the query in the question section. For example, if the question asked for the IPv4 address (A record) of a domain name, the answer section would contain the IP address(es) associated with that domain name.

3. Authority Section: This section provides information about the authoritative DNS servers for the queried domain. It includes the domain name of the authoritative DNS server(s) and may also include additional information such as the TTL (time to live) for the records in the answer section. Essentially, it tells you which DNS servers are responsible for providing information about the domain.

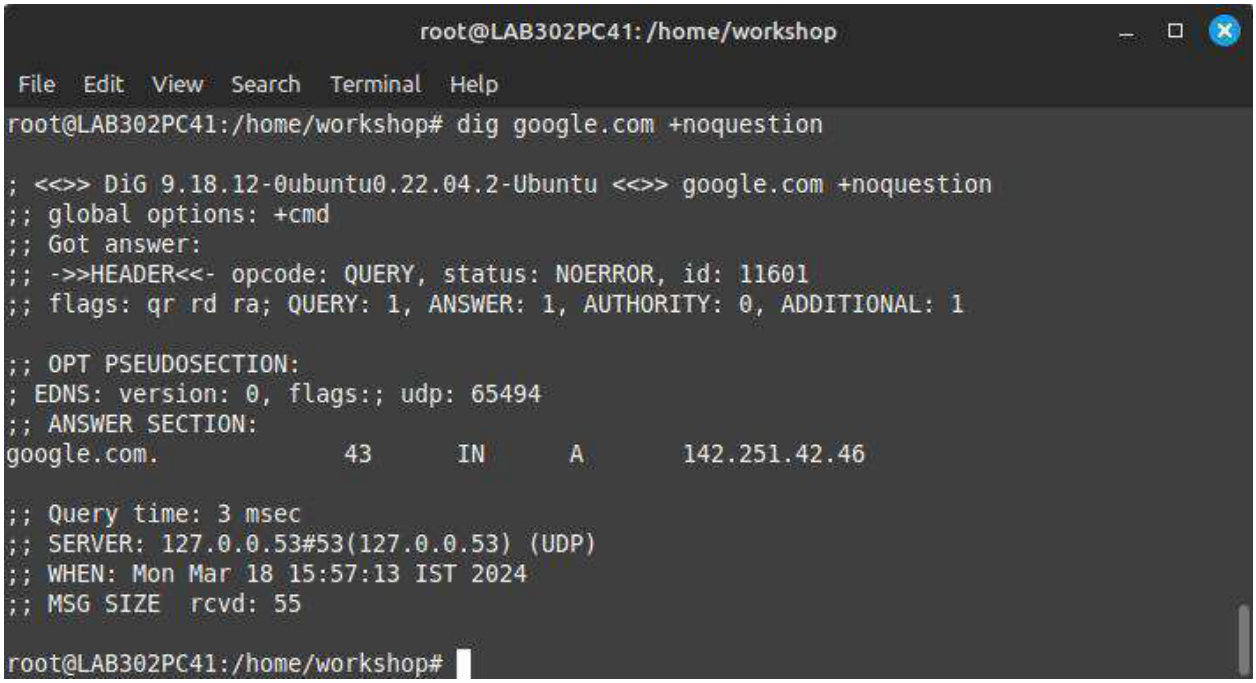
4. Additional Section: The additional section contains any extra information that might be helpful but is not strictly necessary to answer the query. This can include additional resource records that are related to the queried domain, such as glue records (IP addresses of authoritative name servers), or other related information. It helps in providing additional context or data that might be relevant to the query.



Command: dig google.com +noquestion

The command `dig google.com +noquestion` is used to query the DNS information for the domain "google.com" without displaying the question being asked.

In simpler terms, it's like asking a DNS server, "Hey, what's the information about google.com?" but not showing exactly what you asked. It's useful if you want to see the DNS response without clutter from the question you asked.



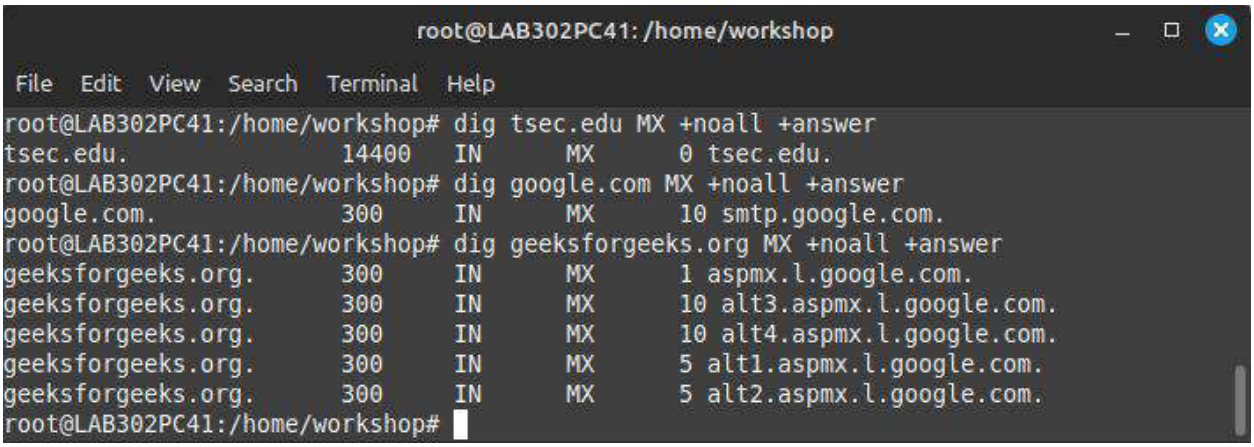
Command: dig tsec.edu MX +noall +answer

The command `dig tsec.edu MX +noall +answer` is used to query the DNS server for the domain "tsec.edu" specifically for its mail exchange (MX) records and then display only the answer section of the response.

In simpler terms, it's like asking the DNS server, "Hey, can you tell me the mail exchange information for tsec.edu?" Then, it shows only the answer without any additional details.

MX (Mail Exchange): MX records are DNS records that specify the mail servers responsible for receiving email messages on behalf of a domain. When you send an email to an address like "user@example.com," your email server checks the MX records for the

"example.com" domain to find out which server is responsible for handling emails for that domain. The MX record points to the hostname of the mail server and its priority (which determines the order in which email servers are contacted).

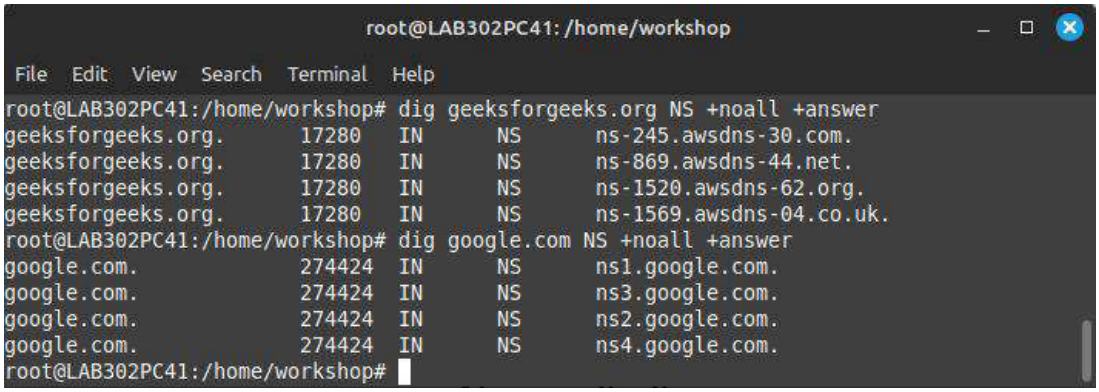


Command: dig geeksforgeeks.org NS +noall +answer

The command `dig geeksforgeeks.org NS +noall +answer` is used to query the DNS server for the domain “geeksforgeeks.org” specifically for its name server (NS) records and then display only the answer section of the response.

In simpler terms, it’s like asking the DNS server, “Hey, can you tell me the name servers for geeksforgeeks.org?” Then, it shows only the answer without any additional details.

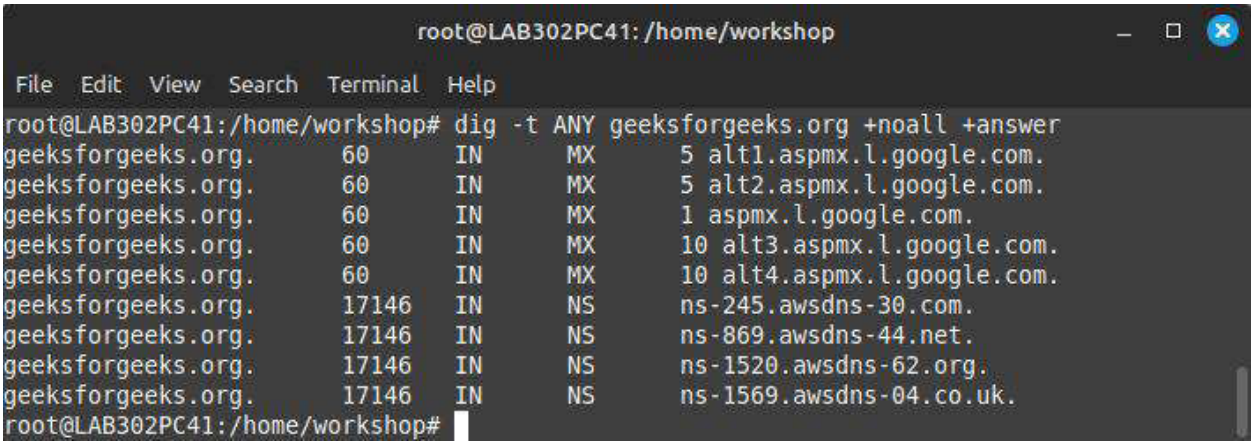
NS stands for "Name Server." NS records are DNS resource records that specify which name servers are authoritative for a particular domain. These name servers are responsible for providing DNS resolution for that domain, meaning they hold information about the domain's DNS records, such as A (IPv4 address), AAAA (IPv6 address), MX (mail exchange), and so on.



Command: dig -t ANY geeksforgeeks.org +noall +answer

The command `dig -t ANY geeksforgeeks.com +noall +answer` is used to query the DNS server for the domain "geeksforgeeks.com" for all types of DNS records and then display only the answer section of the response, excluding any additional information.

In simpler terms, it's like asking the DNS server, "Can you give me all the information you have about geeksforgeeks.com?" Then, it shows only the answer without any extra details.



Command: dig geeksforgeeks.org +short

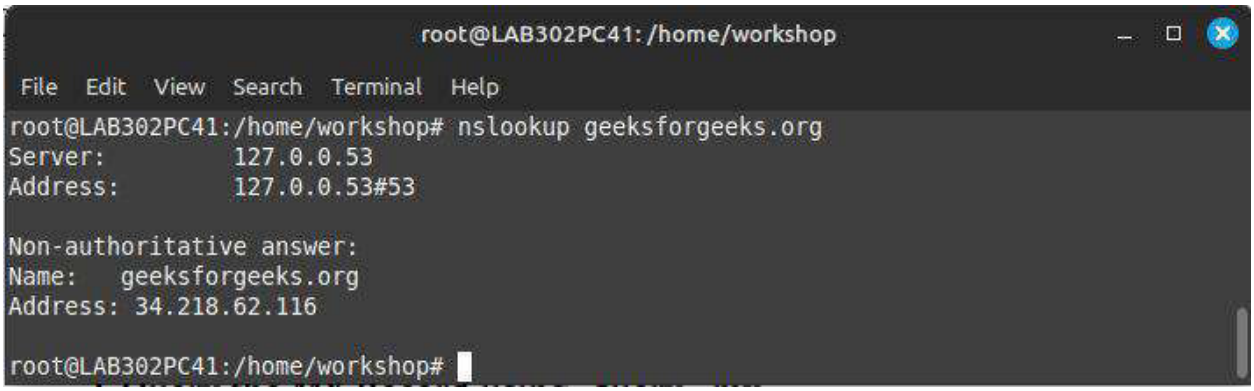
The command `dig geeksforgeeks.org +short` retrieves and displays the IP addresses associated with the domain "geeksforgeeks.org" in a concise format, without additional information such as DNS record types or the DNS server's response time.

3. **NSLOOKUP:**

The nslookup command is used to query internet name servers interactively for information. Nslookup, which stands for "name server lookup". It is a useful tool for finding out information about a named domain. By default, nslookup will translate a domain name to an IP address (or vice versa). Nslookup has two modes: interactive and non-interactive.
Interactive mode allows the user to query name servers for information about various hosts and domains or to print a list of hosts in a domain. Non-interactive mode is used to print just the name and requested information for a host or domain.

Command: nslookup geeksforgeeks.org

The `nslookup geeksforgeeks.org` command is used to query the DNS (Domain Name System) server for information about the domain "geeksforgeeks.org". It typically returns the IP address(es) associated with the domain, as well as other DNS-related details such as the authoritative DNS server for the domain.



Command: nslookup -type = soa geeksforgeeks.com

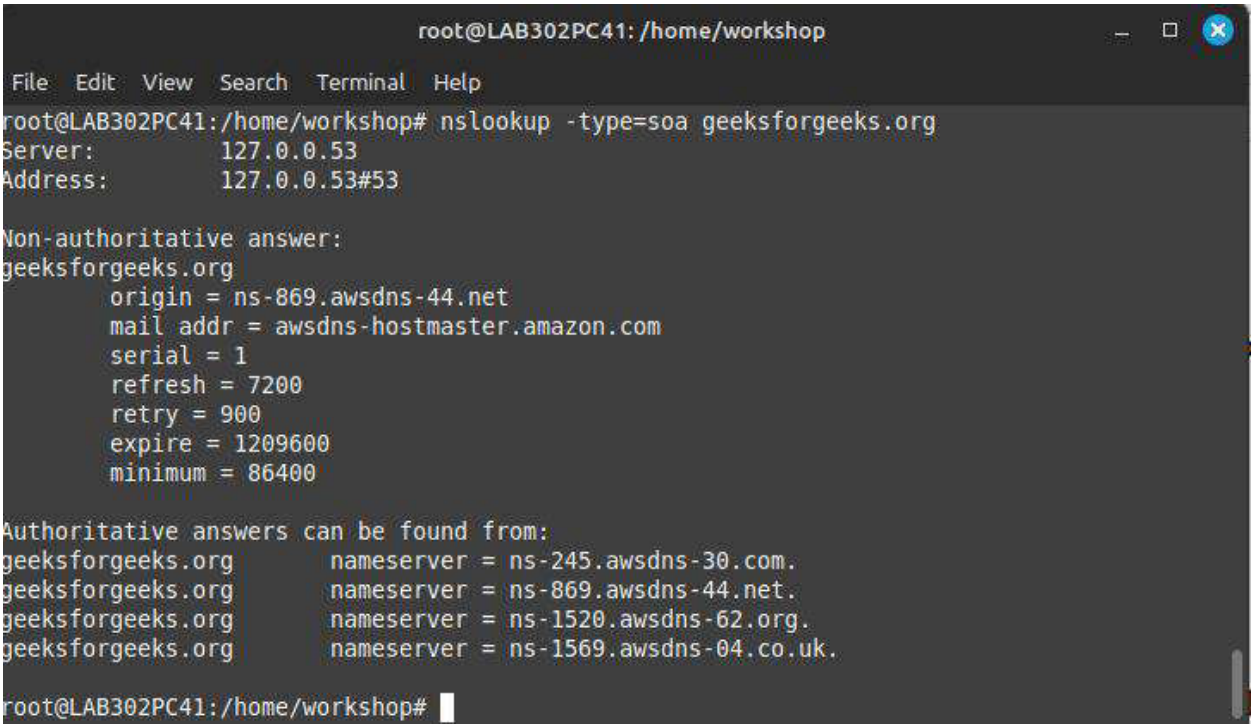
The command `nslookup -type=soa google.com` is used to perform a DNS query specifically for the Start of Authority (SOA) record of the domain "google.com".

Here's what `-type=soa` means:

- `-type=soa`: This option specifies the type of DNS record to query for, in this case, the Start of Authority (SOA) record. The SOA record is a fundamental part of the DNS system and contains essential information about the zone, including the primary authoritative

name server for the domain, the email address of the responsible party, and various timing parameters for the zone (such as refresh, retry, expire, and minimum TTL).

So, when you run `nslookup -type=soa google.com`, you're specifically asking for the SOA record of the domain "google.com". This command will return details about the primary authoritative name server and other important parameters related to the DNS zone for the domain.



Command: nslookup -type = mx geeksforgeeks.com

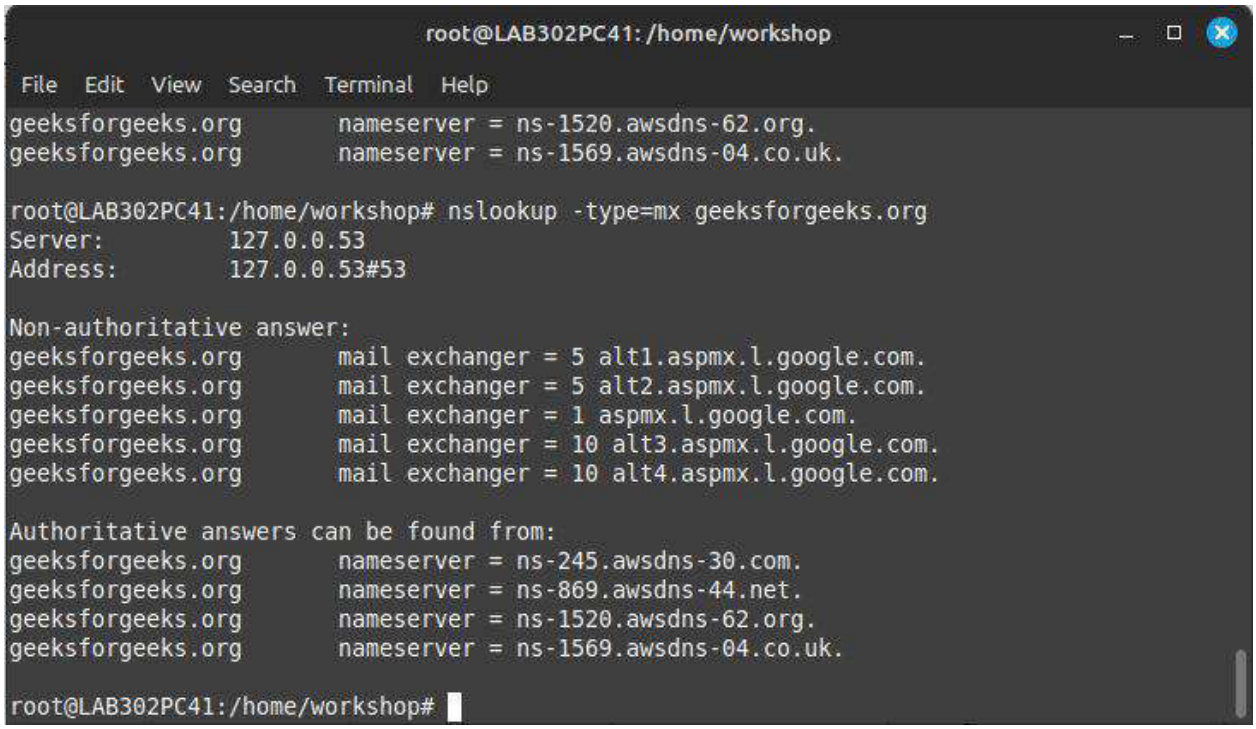
The command `nslookup -type=mx google.com` is used to perform a DNS query specifically for the Mail Exchange (MX) records of the domain "google.com".

Here's what `-type=mx` means:

- `-type=mx`: This option specifies the type of DNS record to query for, in this case, the Mail Exchange (MX) records. MX records are DNS resource records that specify the mail servers responsible for receiving email messages on behalf of a domain. When you send an email to an address like "user@example.com," your email server checks the MX records for the "example.com" domain to find out which server is responsible for handling

emails for that domain. The MX record points to the hostname of the mail server and its priority (which determines the order in which email servers are contacted).

So, when you run `nslookup -type=mx google.com`, you're specifically asking for the MX records of the domain "google.com". This command will return information about the mail servers configured to handle email for the domain.



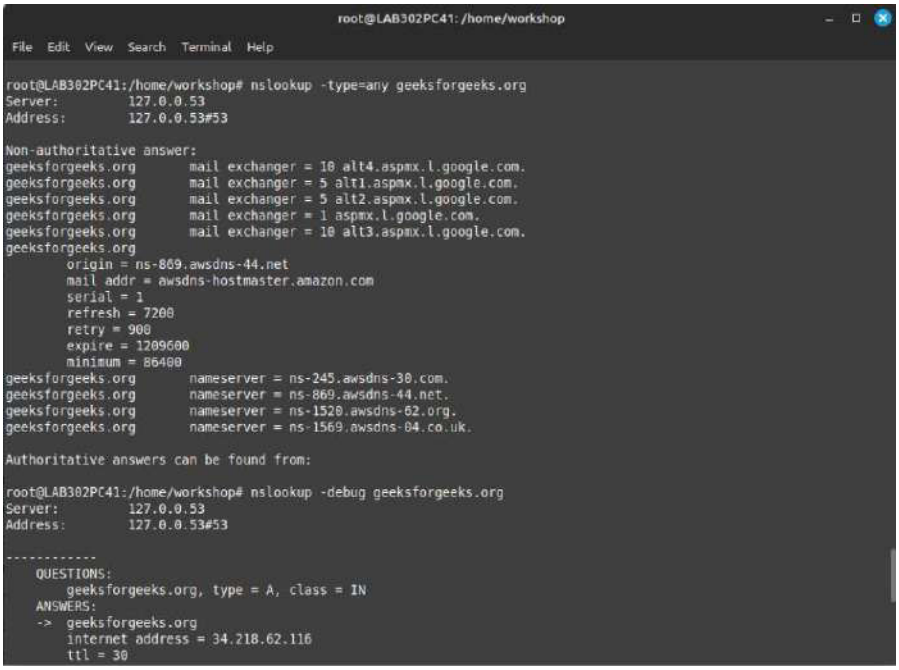
Command: nslookup -type = any geeksforgeeks.com

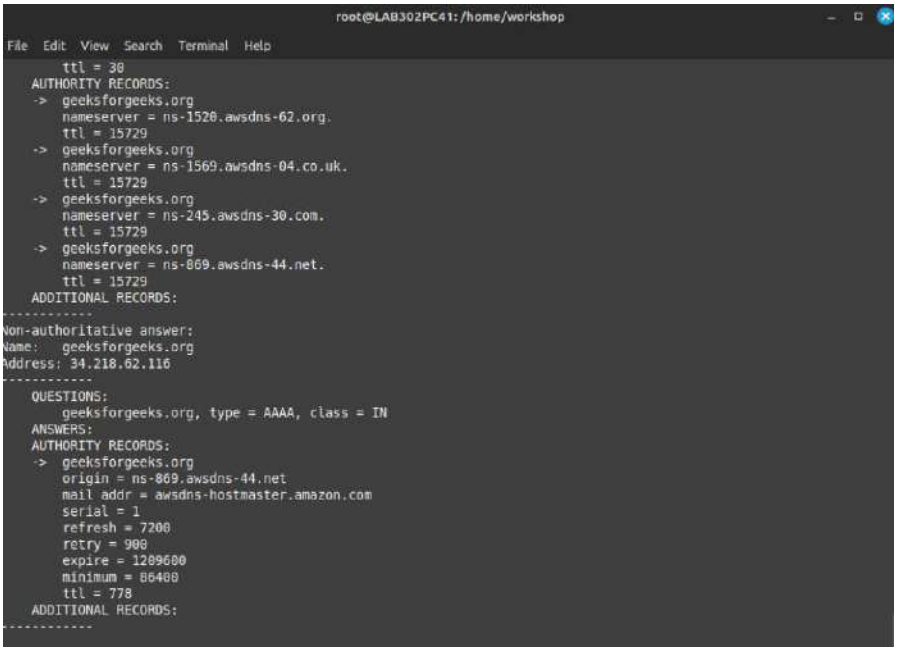
The command `nslookup -type=any geeksforgeeks.com` is used to perform a DNS query for all available DNS record types for the domain "geeksforgeeks.com".

Here's what `-type=any` means:

- `-type=any`: This option specifies the type of DNS record to query for. When set to "any," it requests all available types of DNS records associated with the domain. This includes A (IPv4 address), AAAA (IPv6 address), MX (mail exchange), NS (name server), SOA (start of authority), and any other record types present for the domain.

So, when you run `nslookup -type=any geeksforgeeks.com`, you're asking for all possible types of DNS records associated with the domain "geeksforgeeks.com". This command will return comprehensive information about the domain's DNS configuration, including its IP address(es), mail exchange servers, authoritative name servers, and other DNS-related details.





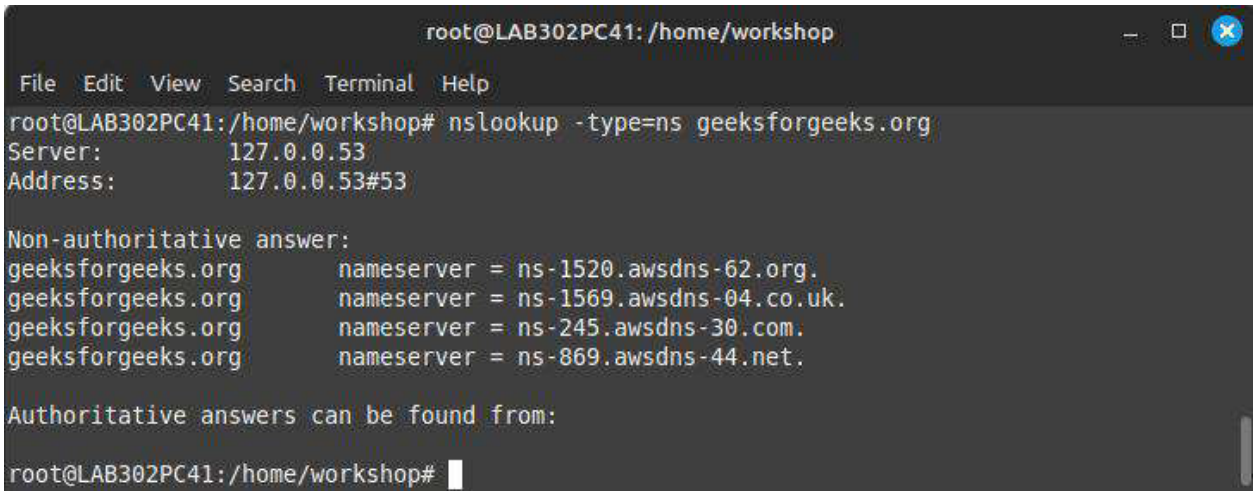
Command: nslookup -type = ns geeksforgeeks.com

The command `nslookup -type=ns geeksforgeeks.com` is used to perform a DNS query specifically for the Name Server (NS) records of the domain "geeksforgeeks.com".

Here's what `-type=ns` means:

- `-type=ns`: This option specifies the type of DNS record to query for, in this case, the Name Server (NS) records. NS records are DNS resource records that specify which name servers are authoritative for a particular domain. These name servers are responsible for providing DNS resolution for that domain, meaning they hold information about the domain's DNS records, such as A (IPv4 address), AAAA (IPv6 address), MX (mail exchange), and so on. When you query a domain's NS records, you get the list of name servers that can provide authoritative information about that domain.

So, when you run `nslookup -type=ns geeksforgeeks.com`, you're specifically asking for the NS records of the domain "geeksforgeeks.com". This command will return the list of name servers responsible for providing DNS information for the domain.



4. **TRACEROUTE:**

The `traceroute` command in Linux is a network diagnostic tool used to trace the route taken by packets from your computer to a destination host. Here's how it works:

1. Sending Packets: When you run the `traceroute` command followed by a destination host (such as a domain name or IP address), your computer starts sending a series of packets towards that destination.

2. TTL Field: Each packet sent has a Time-to-Live (TTL) field set to 1 initially. When the first router receives the packet, it decrements the TTL by 1. If the TTL reaches 0, the router discards the packet and sends back an ICMP "Time Exceeded" message to your computer.

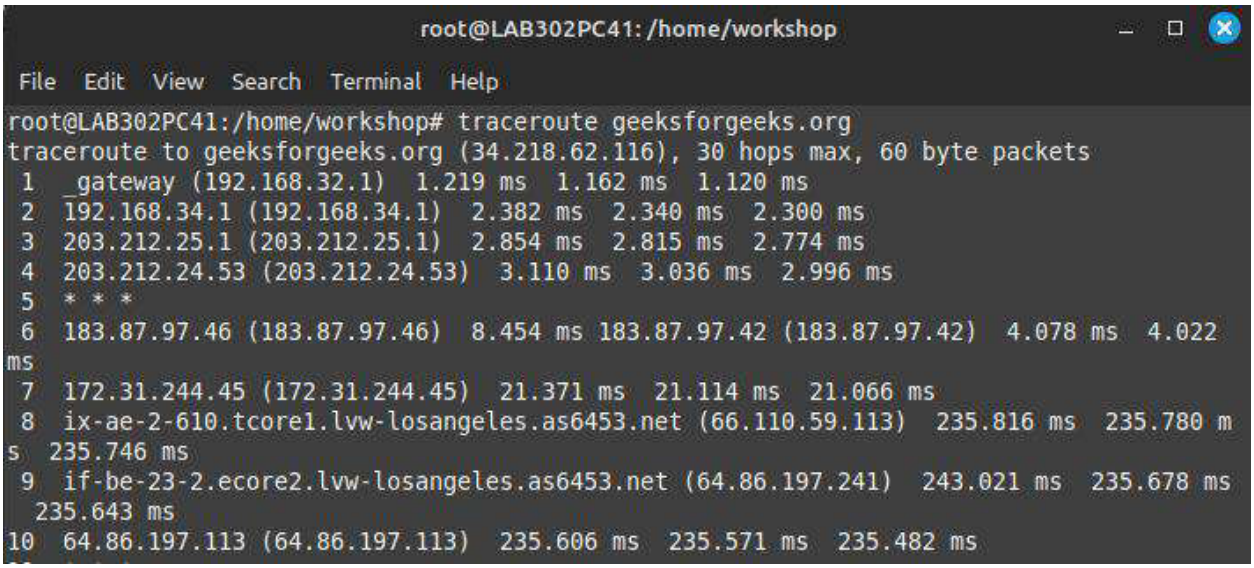
3. Tracing the Route: The `traceroute` command then sends subsequent packets with increasing TTL values (starting from 1) so that they travel farther along the route. By doing this repeatedly, it can trace the path taken by the packets as they hop through different routers on the internet.

4. Displaying Results: As each packet reaches a router, `traceroute` records the IP address of the router and the round-trip time (RTT) it took for the packet to reach that router. This information is displayed to the user in a list format, showing the sequence of routers traversed and the corresponding RTT for each hop.

5. Completion: `traceroute` continues sending packets until it reaches the destination host or until a maximum number of hops is reached (which is typically set to 30 by default). Once the destination is reached or the maximum number of hops is reached, `traceroute` completes and provides the final results.

Overall, `traceroute` is a valuable tool for diagnosing network connectivity issues, identifying bottlenecks, and understanding the network path taken by packets between your computer and a destination host.

Command: traceroute geeksforgeeks.org



Parth Dabholkar C12-32

Experiment No. 8

Aim: Examine the use of packet sniffer tool: Wireshark
a) Download and install wireshark and capture different packets like icmp, tcp and http packets
b) Explore how the packets can be traced based on different filters
c) Capture packets of FTP and retrieve login ID and Password

Theory:

Wireshark is a widely-used network protocol analyzer that allows users to capture and inspect network traffic in real-time or from stored data. It offers a user-friendly interface with powerful features such as packet filtering, protocol analysis, and detailed packet inspection. Wireshark is invaluable for network troubleshooting, protocol development, and network security analysis, providing deep insights into network behavior and aiding in the detection of anomalies and security threats. With its extensive capabilities and cross-platform support, Wireshark is an essential tool for network administrators, security analysts, developers, and anyone involved in managing or securing network infrastructure.

Features of Wireshark:

1. Live Packet Capture: Wireshark can capture live network traffic from various interfaces, allowing real-time analysis of data packets as they flow through the network.

2. Deep Protocol Analysis: It provides detailed insights into network protocols by allowing users to inspect packet headers, payloads, and other data fields. This depth of analysis helps in understanding protocol behavior and identifying issues.

3. Flexible Filtering: Wireshark offers powerful filtering capabilities to focus on specific types of network traffic based on various criteria such as IP addresses, port numbers, protocols, and packet contents. This helps in isolating and analyzing relevant packets efficiently.

4. Customizable Display: Users can customize the display of captured packets to suit their preferences and requirements. This includes options to configure packet views, colorization, and protocol hierarchy, enhancing readability and analysis efficiency.

5. Comprehensive Statistics: Wireshark provides detailed statistics on captured network traffic, including traffic volume, protocol usage, packet timing, and error rates. These statistics offer valuable insights into network performance and behavior.

6. Cross-Platform Support: Wireshark is available for multiple operating systems, including Windows, macOS, and Linux, ensuring broad compatibility across different environments. This allows network administrators and analysts to use the tool across various platforms seamlessly.

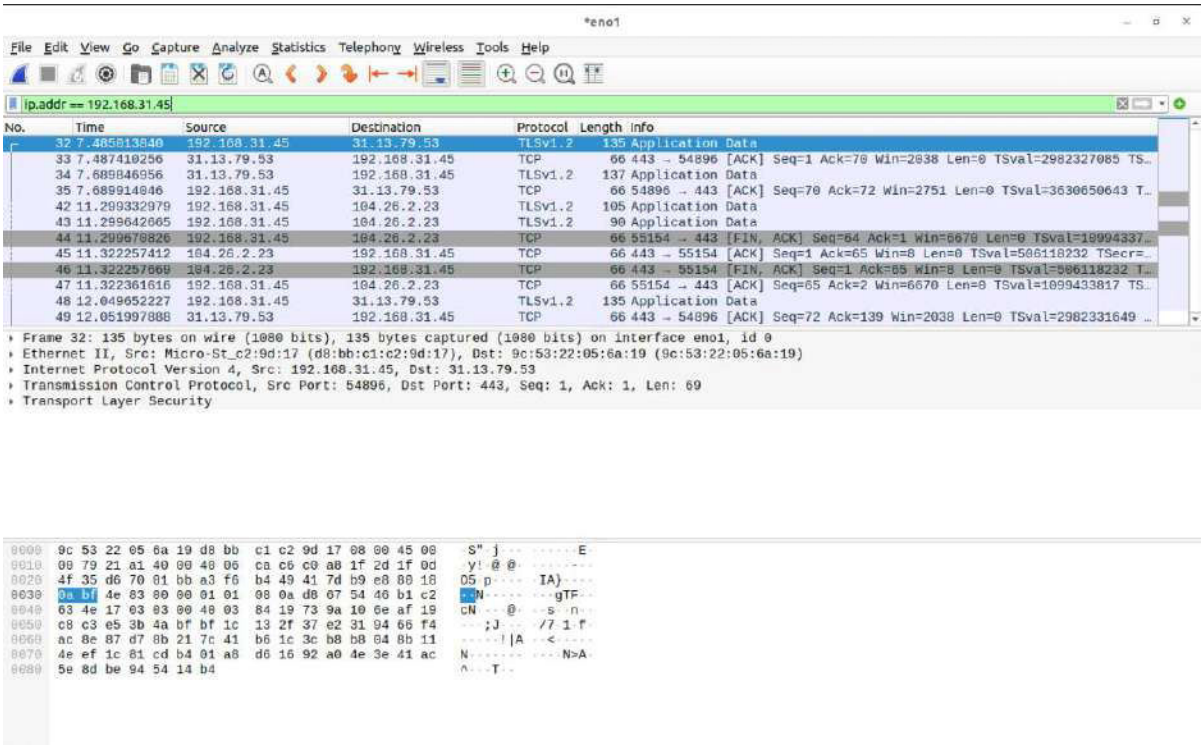
Steps to download wireshark:

1. Open ubuntu terminal
2. Install wireshark
apt-get install wireshark
3. To know the name of your Ethernet interface: (Mostly it is “etht0”)
#ifconfig
4. Start wireshark
#sudo wireshark
5. Once wireshark window opens, select the interface and click on star

Capturing Packets:

After downloading and installing wireshark, you can launch it and click the name of an interface under Interface List to start capturing packets on that interface.
For example, if you want to capture traffic on the wireless network, click your wireless interface.You can configure advanced features by clicking Capture Options.
As soon as you click the interface‘s name, you‘ll see the packets start to appear in real time. Wireshark captures each packet sent to or from your system.
Click the stop capture button near the top left corner of the window when you want to stop capturing traffic Wireshark uses colors to help you identify the types of traffic at a glance. By default, green is TCP traffic, dark blue is DNS traffic, light blue is UDP traffic, and black identifies TCP packets with problems — for example, they could have been delivered out-of-order. Wireshark can record the capturing information in the file with extension .pcap (packet capture). This file can be reopened for analysis in offline mode.
There is no need to remember filtering commands. Filters can be applied by putting predefined strings in Wireshark.

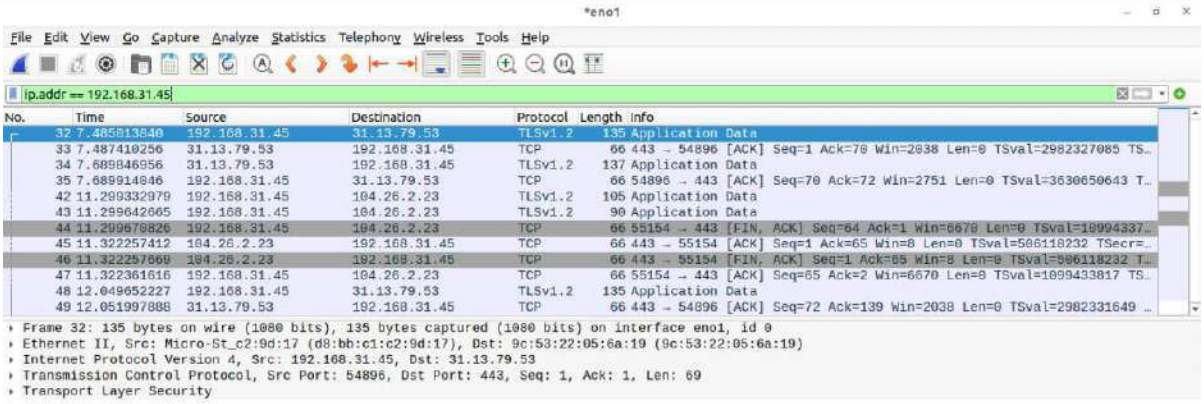
Commands:-
1. Capturing packets of a particular host
ip.addr = = 192.163.31.45
Sets a filter for any packet with 192.163.31.45, as either the source or destination.

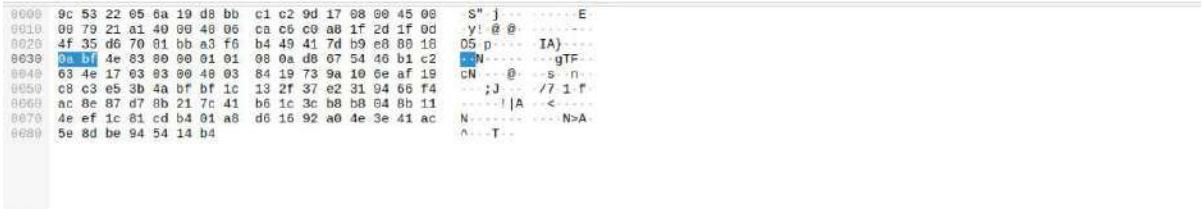


2. To capture a conversation between specified hosts

ip.addr == 192.163.31.45 && ip.addr == 104.26.2.23

Sets a conversation filter between the two defined IP addresses.



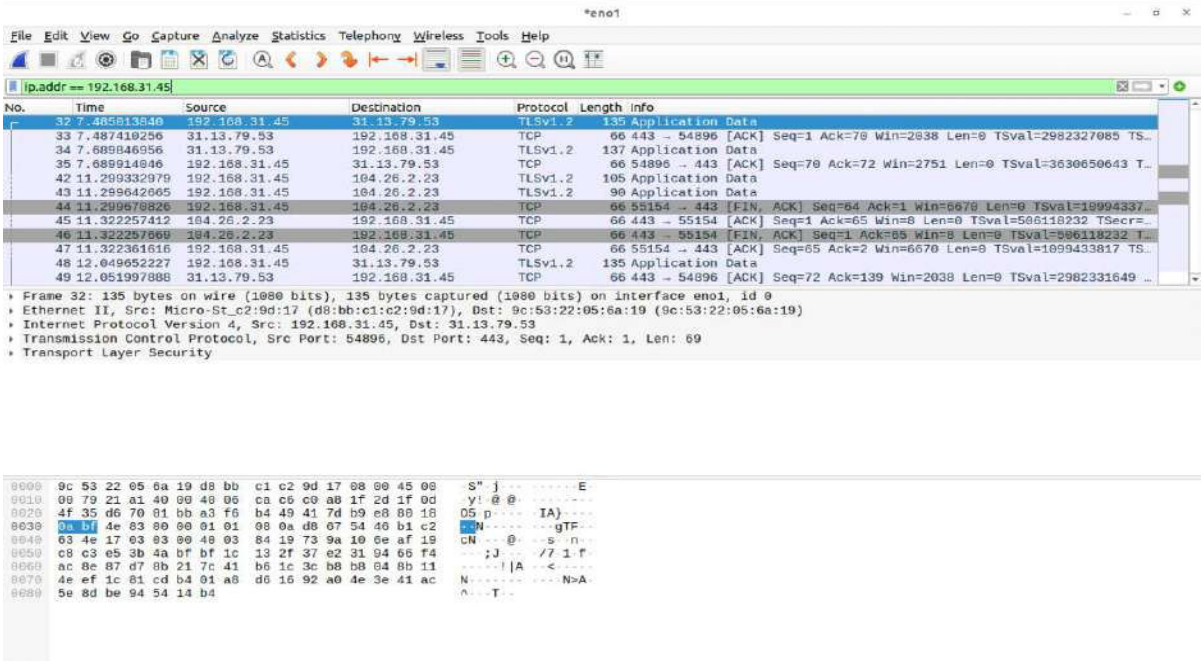


Filtering Packets

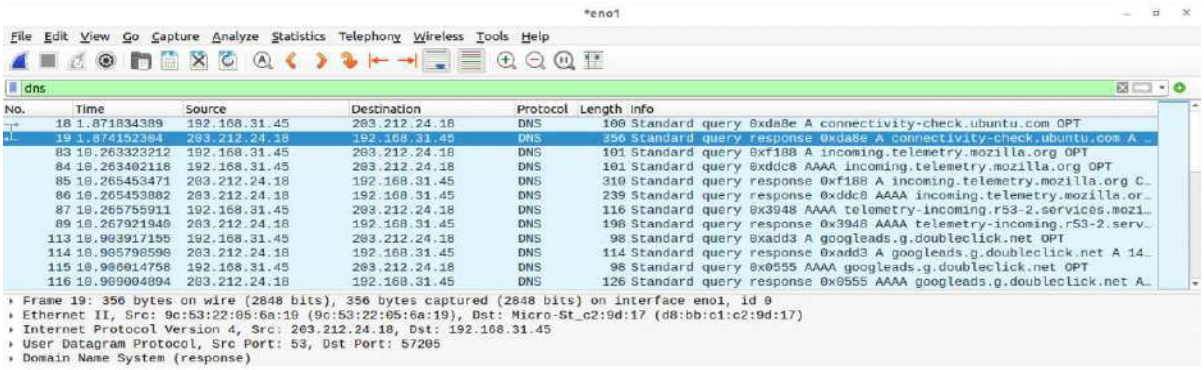
The most basic way to apply a filter is by typing it into the filter box at the top of the window and clicking Apply (or pressing Enter). For example, type —dns and you'll see only DNS packets. When you start typing, Wireshark will help you auto complete your filter.

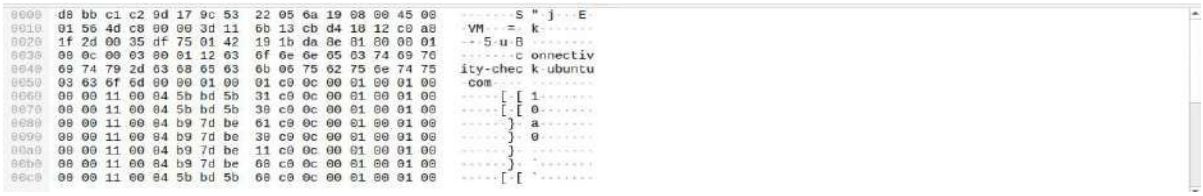
Commands:-

1. To filter packets for a specific protocol : http



Dns



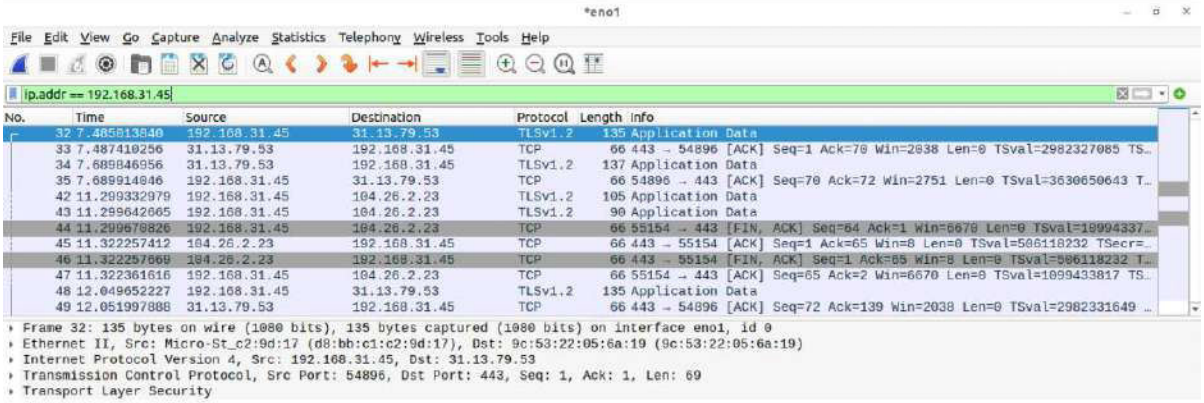


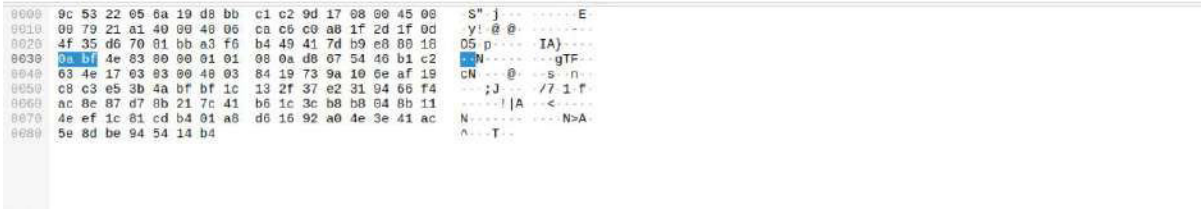
Sets a filter to display all http and dns requests.

2. To filter packets for specific port

tcp.port==80

Sets a filter for any TCP packet with 80 as a source or destination port.

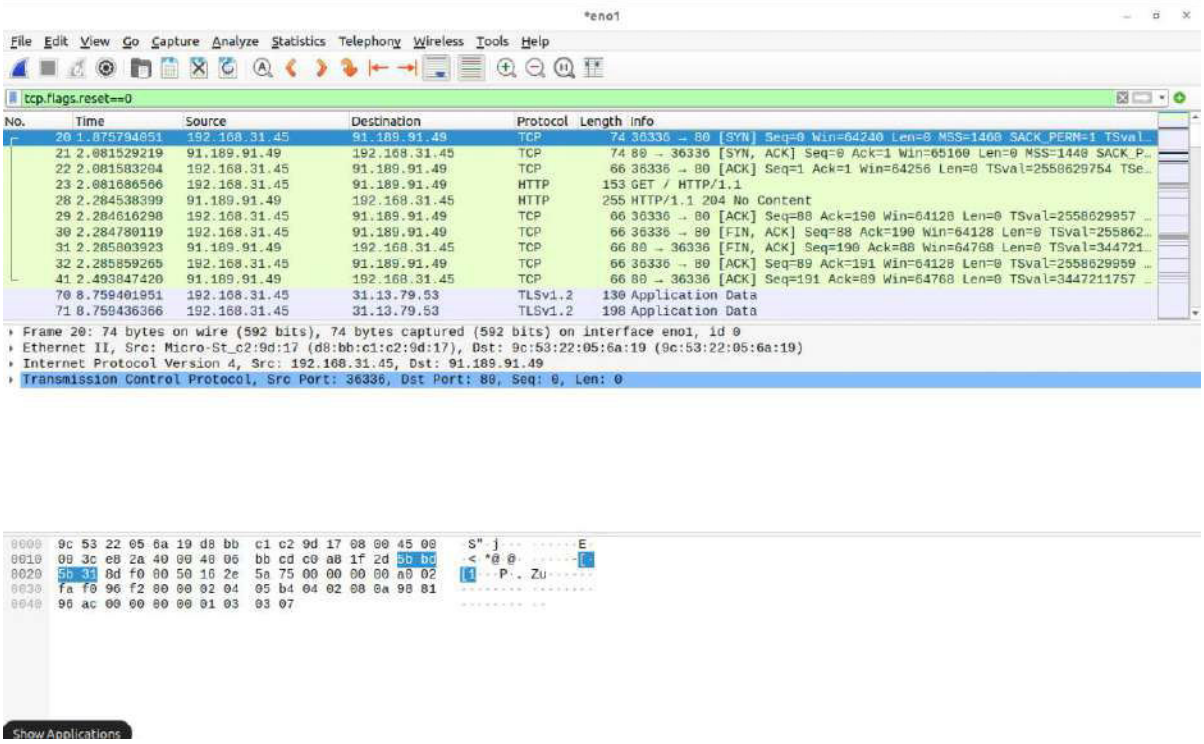




3. Filter specific packets

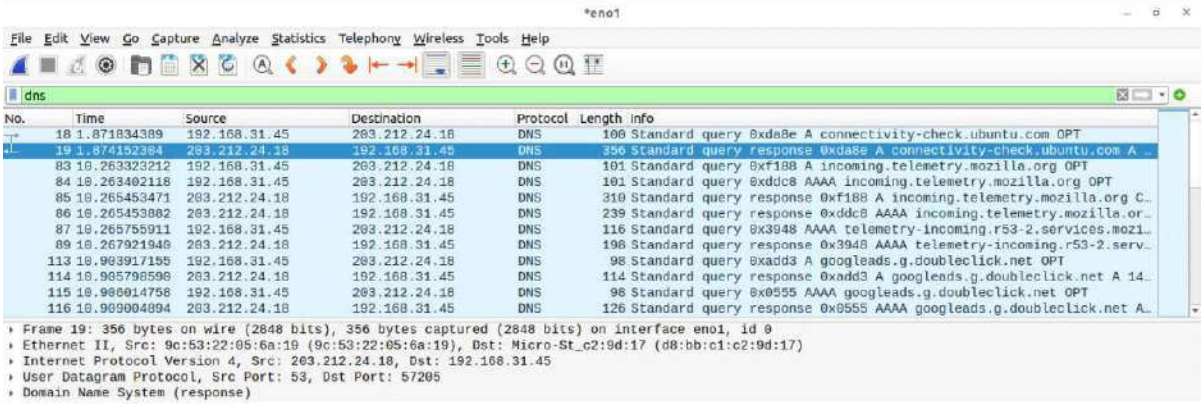
tcp.flags.reset== 0

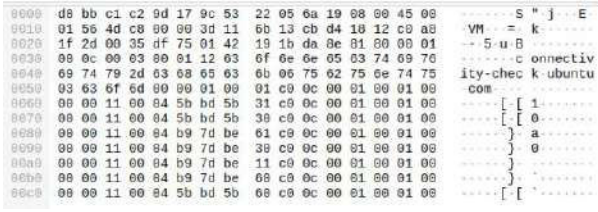
Displays all TCP resets.



4. Filter for http request packets

Http.request

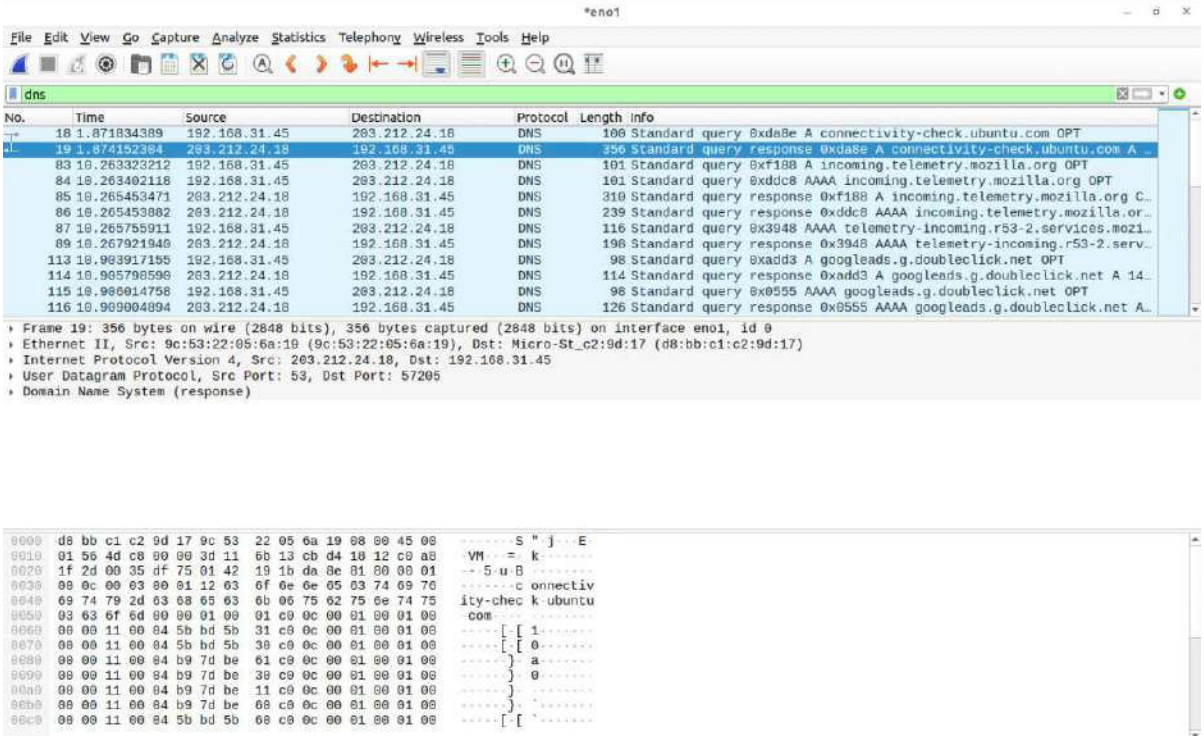




Displays all HTTP GET requests.

5. To filter traffic except given protocol packets

!(arp or icmp or dns)

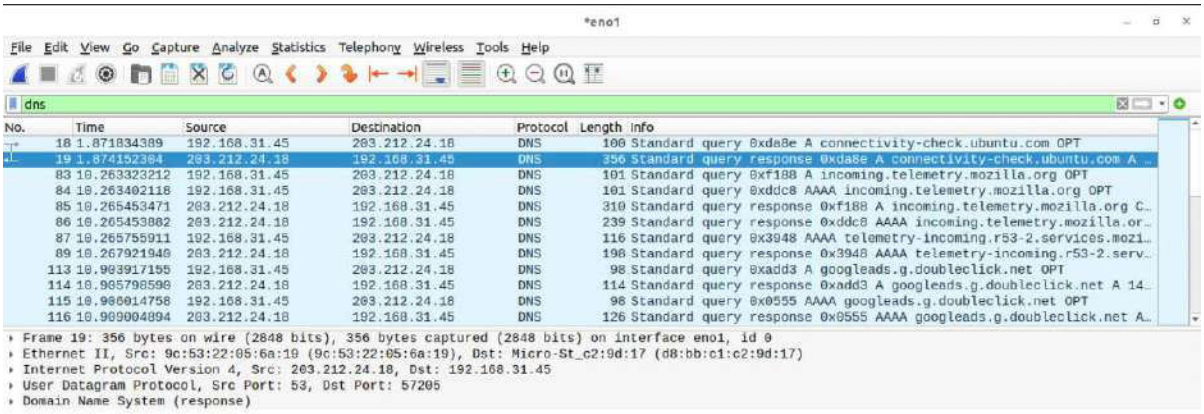


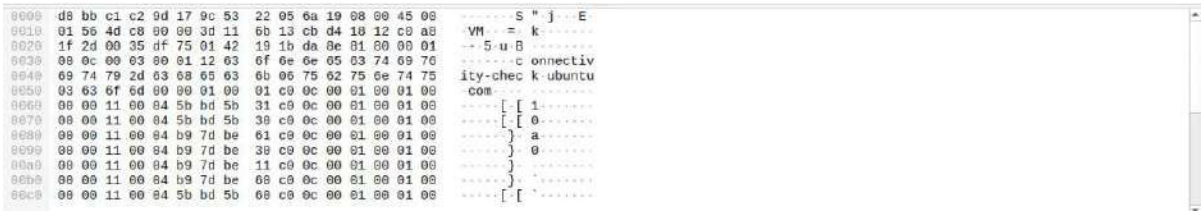
Masks out arp, icmp, dns, or whatever other protocols may be background noise, allowing you to focus on the traffic of interest.

6. Capturing packets after applying multiple filters

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

Get all packets which are not HTTP or UDP.





To stop capturing click on the “red square”

<u>To capture packets from the FTP server. (Login ID and Password)</u>

What is FTP?
FTP stands for File Transfer Protocol. As the name suggest this network protocol allows you to transfer files or directories from one host to another over the network whether it is your LAN or Internet.The package required to install FTP is known as VSFTPD (Very Secure File Transfer Protocol
Daemon)/

Steps:-
1. Get root access: $ sudo su root
2. Find your ip address: # ifconfig

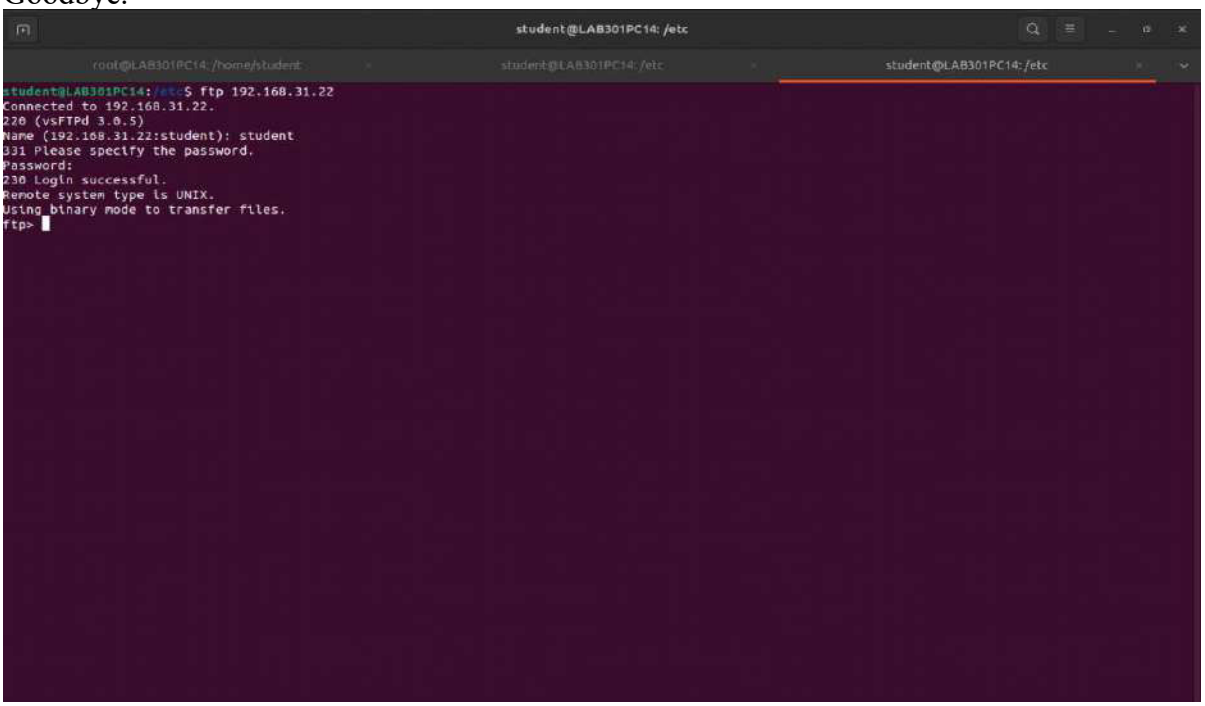
<u>Installation of FTP server in Ubuntu</u>
Name of Packages required: VSFTPD, XINETD

1. # sudo apt-get install vsftpd
2. # sudo apt-get install xinetd
The above command will install and start the xinetd superserver on your system. The chances are that you already have xinetd installed on your system. In that case you can omit the above installation command. In the next step we need to edit the FTP server's configuration file which is present in /etc/vsftpd.conf.
3. # cd /etc
4. # ls

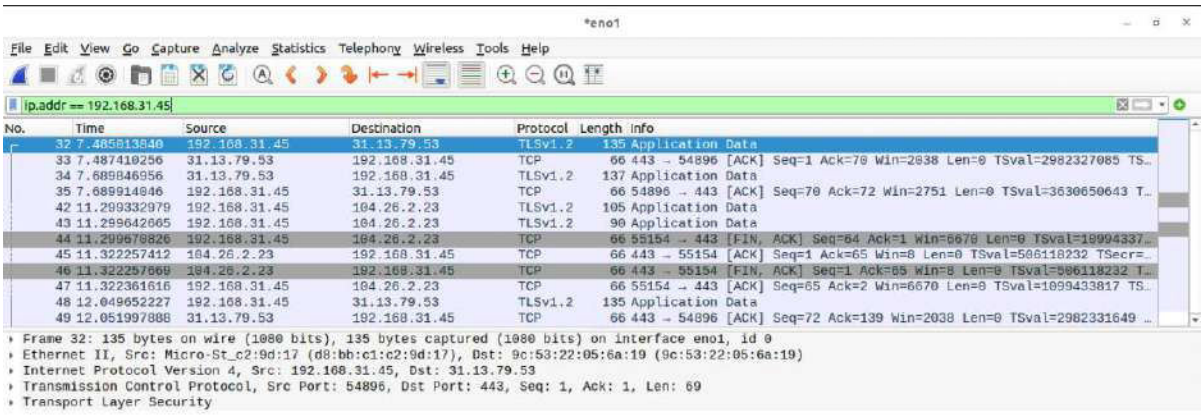
5. # gedit vsftpd.conf
Change the following line:
Anonymous_enable=NO To Anonymous_enable=YES
This will instruct the FTP server to allow connecting with an anonymous client.
6. Save and close the gedit file

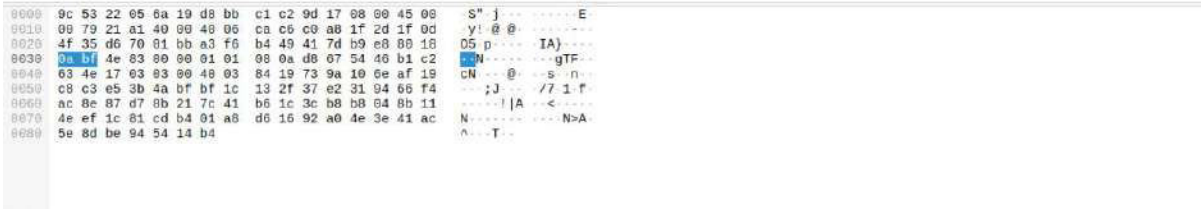
Now, that we are ready we can start the FTP server in the normal mode with:
7. # service xinetd restart
8. # service vsftpd restart OR # init.d/vsftpd restart

Connecting to a client present in other machine
$ ftp ip address of the FTP server
Name: anonymous
Please specify the password.
Password:
Login successful. (even if the login is not successful then also wireshark will capture the id and password)
ftp>
ftp> quit
Goodbye.



Start WIRESHARK. In the FILTER field put FTP. This will filter all FTP packets





While the client is establishing a connection with the FTP server, the wireshark running in the background of the FTP server is able to capture all FTP packets. So, the Name and Password entered by the client is visible in plain text in Wireshark. Apart from that the source and the destination address is also visible. If many clients are trying to connect with the server then source address, name and password are visible for all of them.

EXPERIMENT NO: 9

AIM: Design of Personal Firewall using Iptables

THEORY:

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.

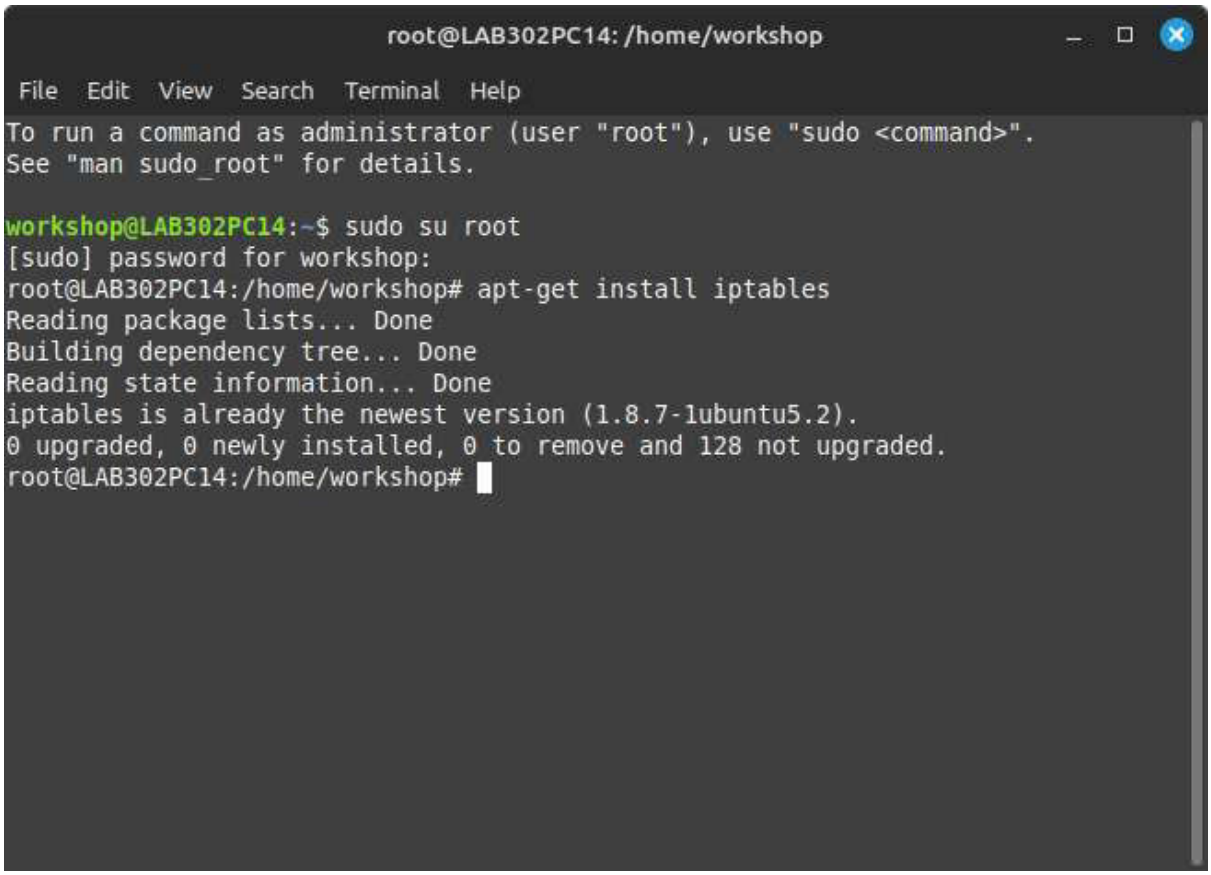
RETURN: jump to the end of the chain and let the default target process it

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

Steps to design Firewall using Iptables:

1. Get root access: $ sudo su root

2. # apt-get install iptables

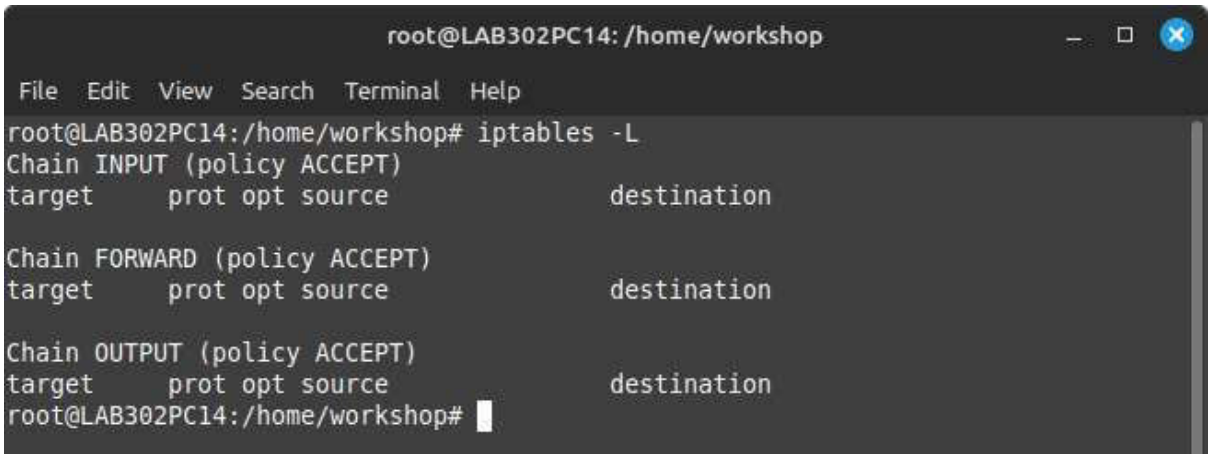


Commands:

1. To see the list of iptables rules

iptables -L

. Initially it is empty



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

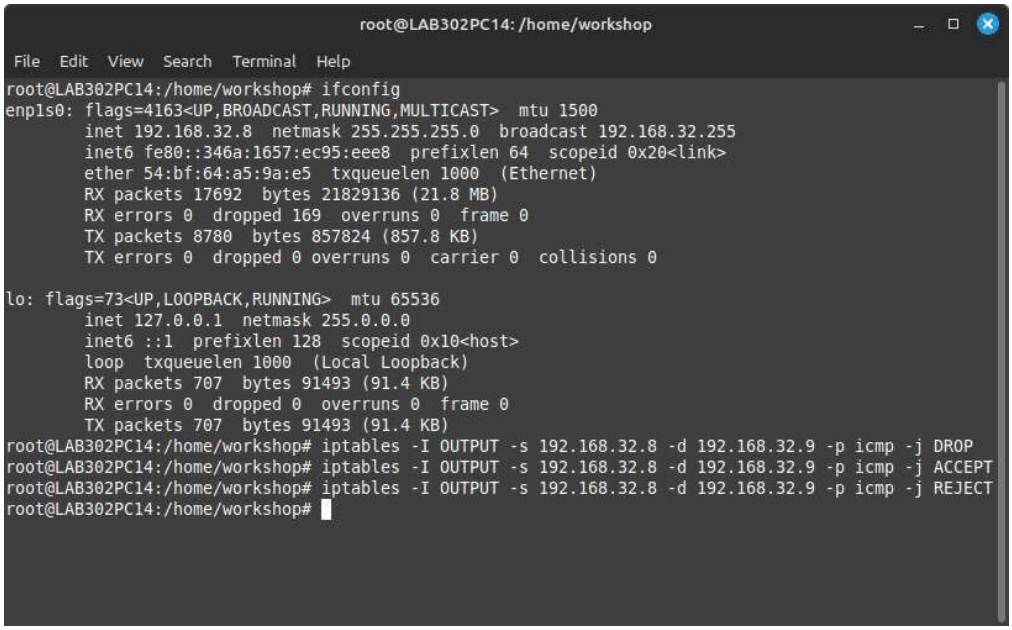
Syntax: iptables -I OUTPUT -s <your ip> -d <neighbour ip> -p <protocol> -j <action>

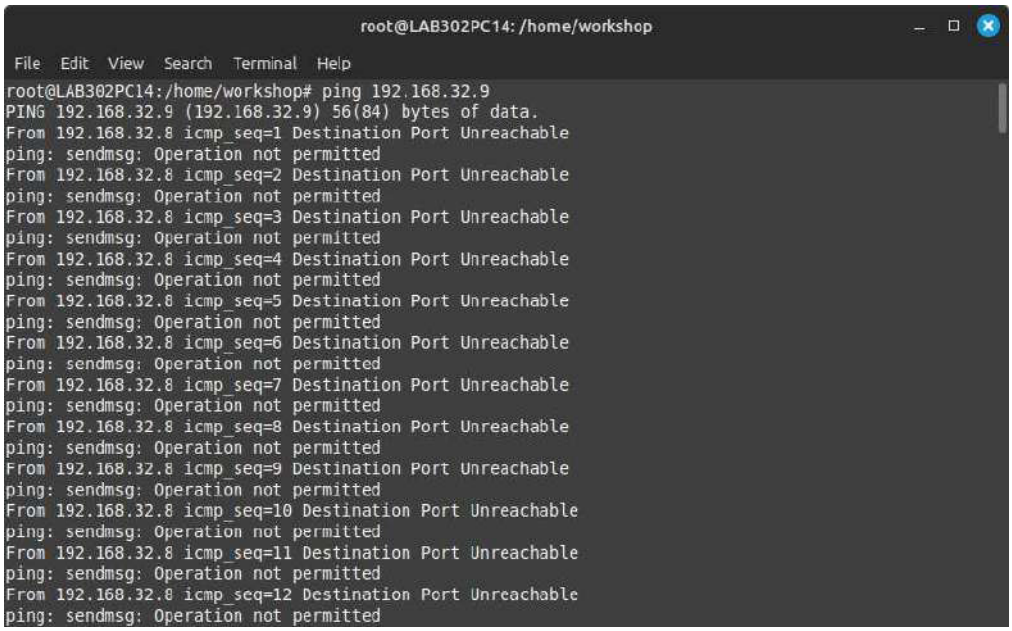
Open one terminal and Ping the neighbour. Let the ping run.

#ping 192.168.32.9

Open another terminal and run the iptables command

 # iptables -I OUTPUT -s 192.168.32.8 -d 192.168.32.9 -p icmp -j DROP





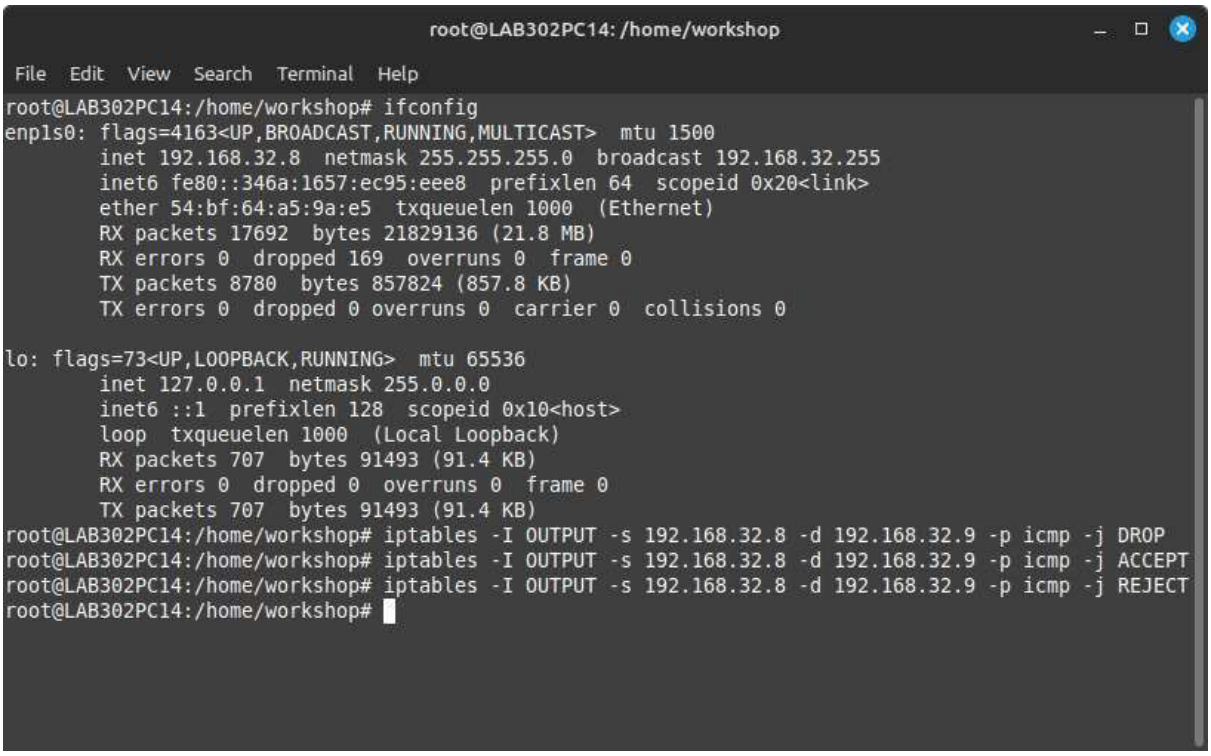
3. To allow outgoing traffic to a particular destination for a specific protocol from a machine

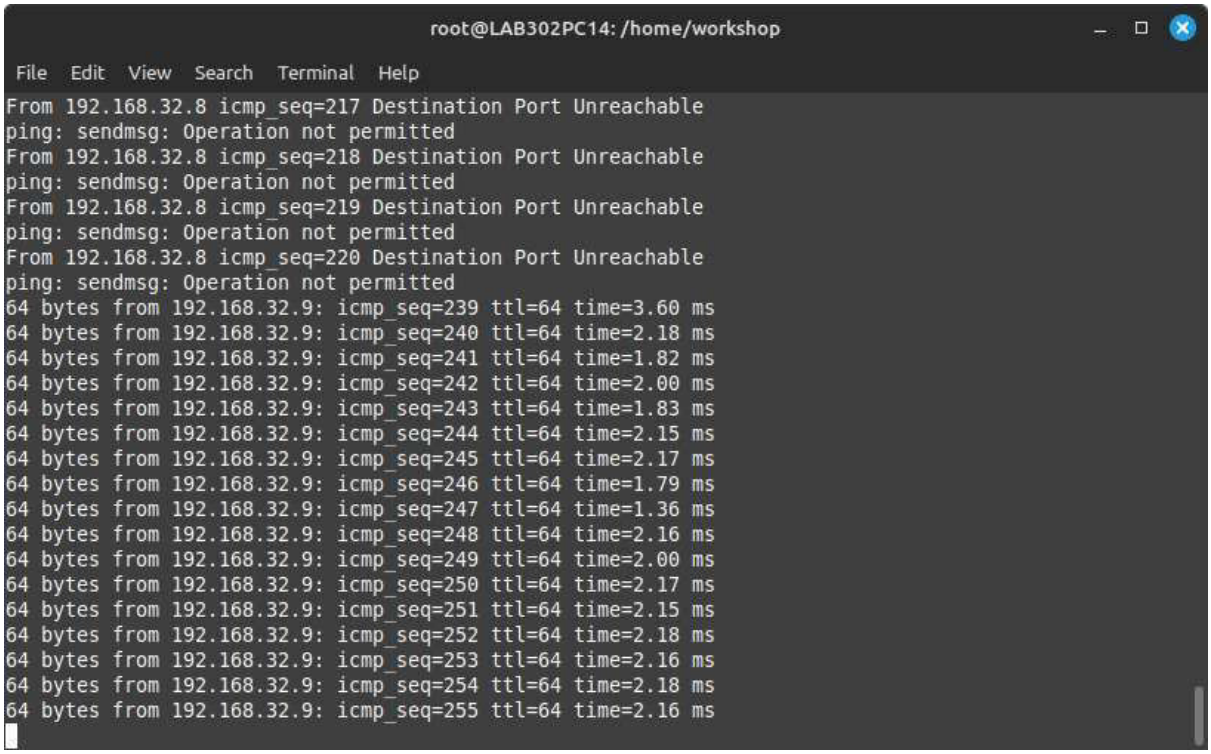
iptables -I OUTPUT -s 192.168.32.8 -d 192.168.32.9 -p icmp -j ACCEPT

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

iptables -I OUTPUT -s 192.168.32.8 -d 192.168.32.9 -p icmp -j REJECT

Allow the traffic again by using ACCEPT instead of REJECT





5. 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 192.168.32.9

Open another terminal and run the iptables command

 # iptables -I INPUT -s 192.168.32.9 -d 192.168.32.8 -p icmp -j DROP

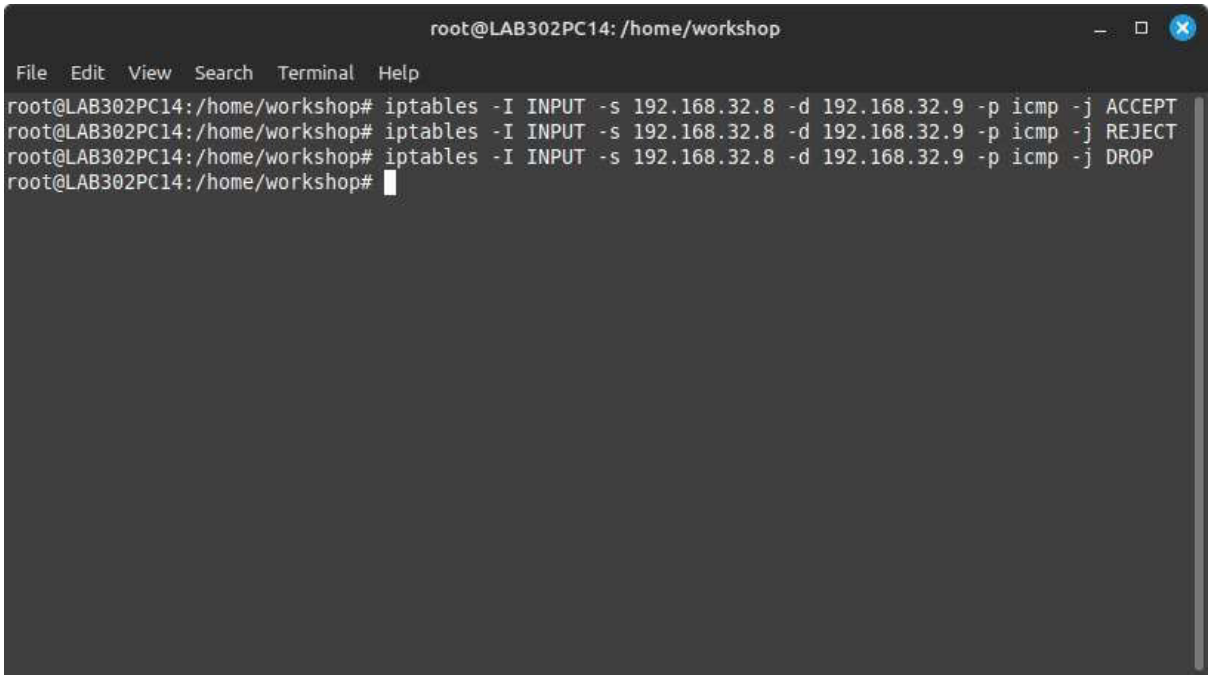
6. To allow incoming traffic from particular destination for a specific protocol to machine

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

Open another terminal and run the iptables command

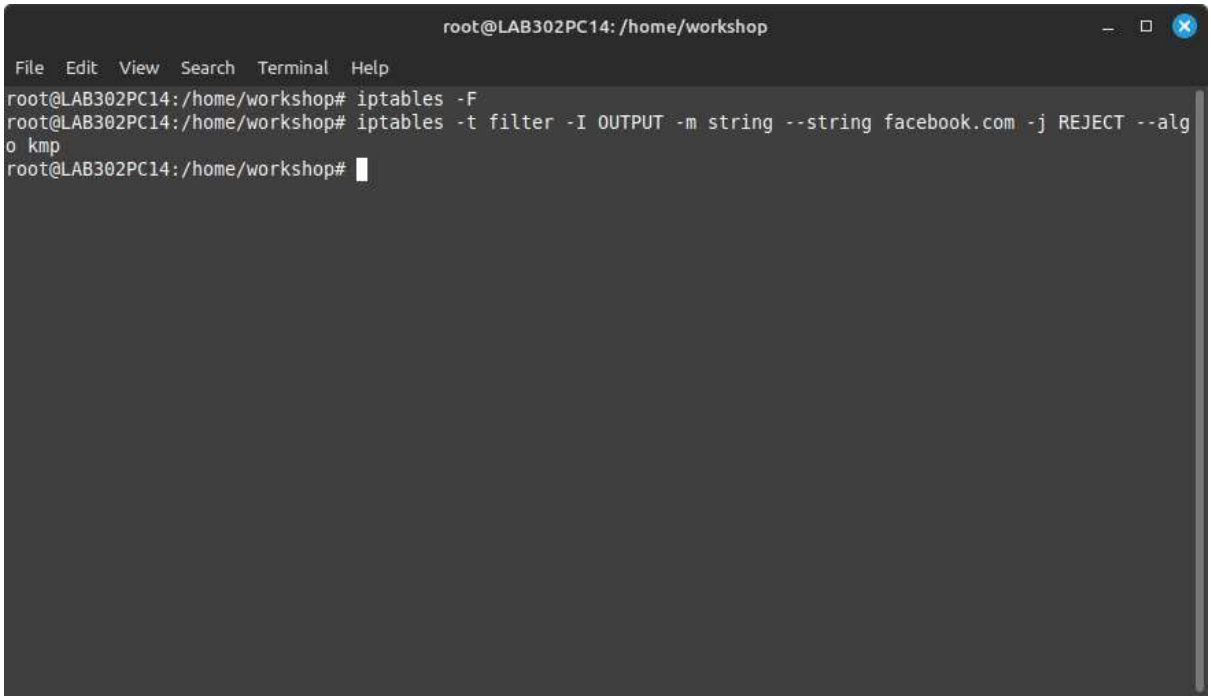
 # iptables -I INPUT -s 192.168.32.9 -d 192.168.32.8 -p icmp -j ACCEPT

Check the ping status on the other terminal



7. To clear the rules in iptables

iptables -F

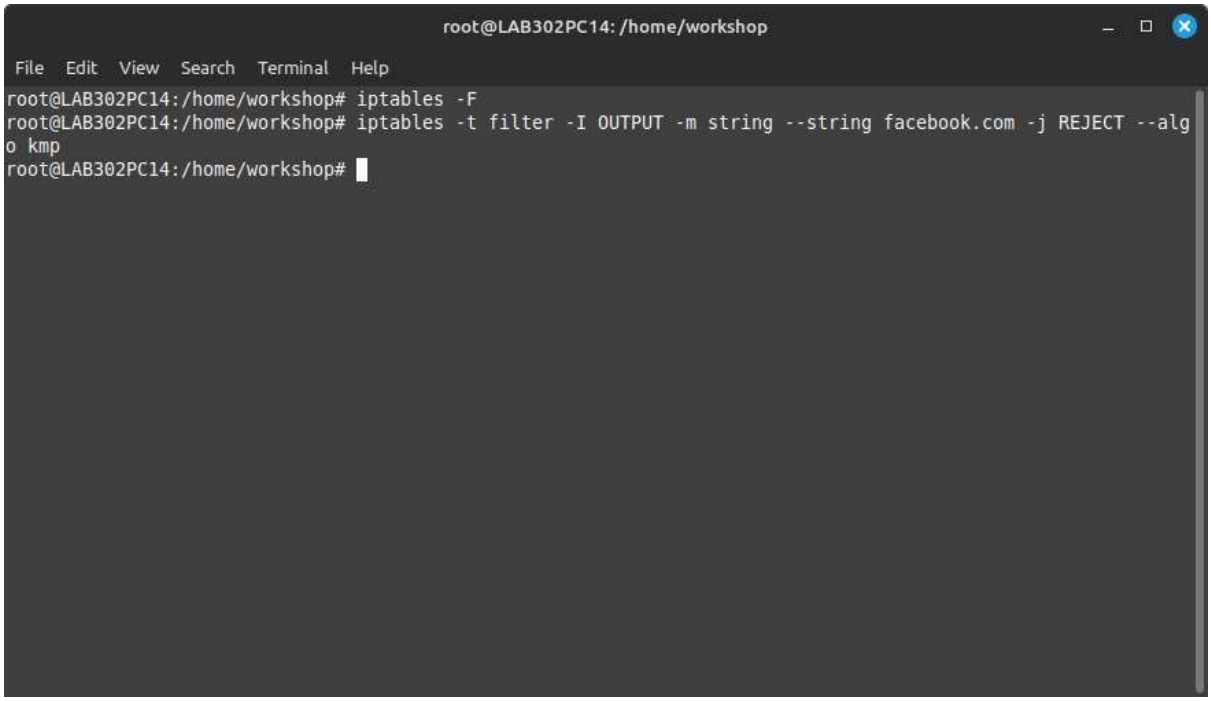


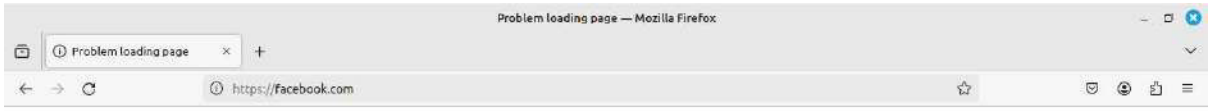
7. To block specific URL from machine

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

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.





The connection has timed out

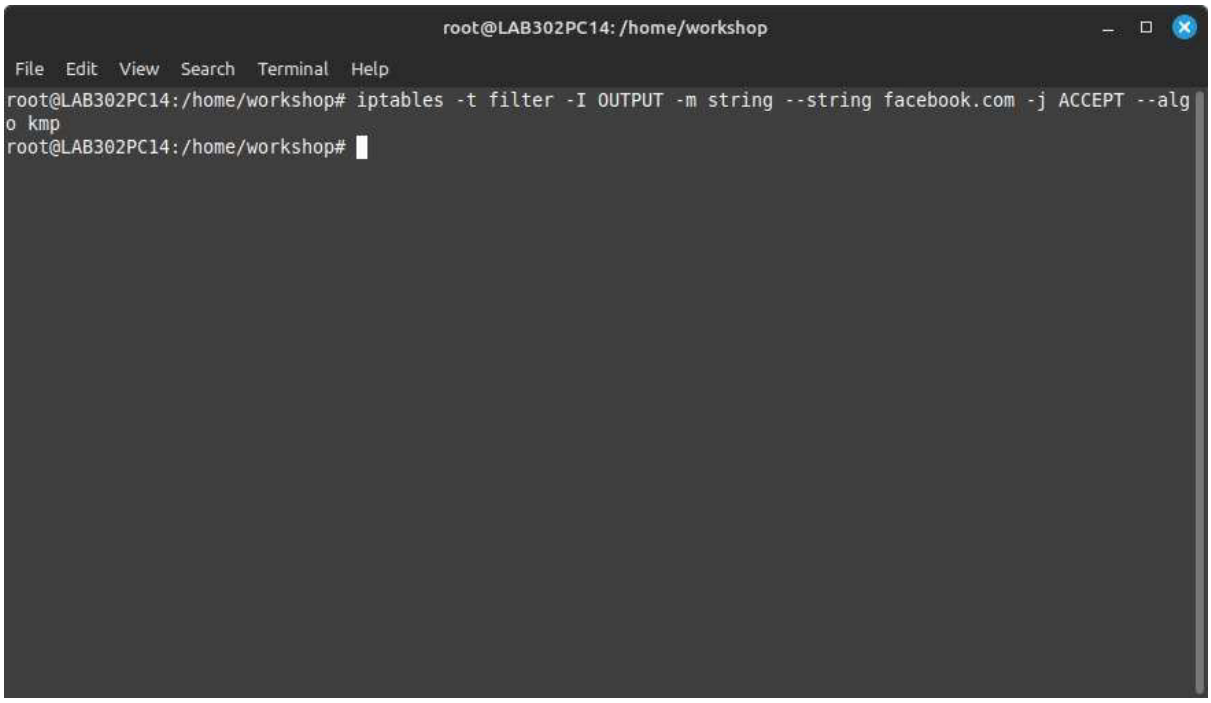
An error occurred during a connection to facebook.com.

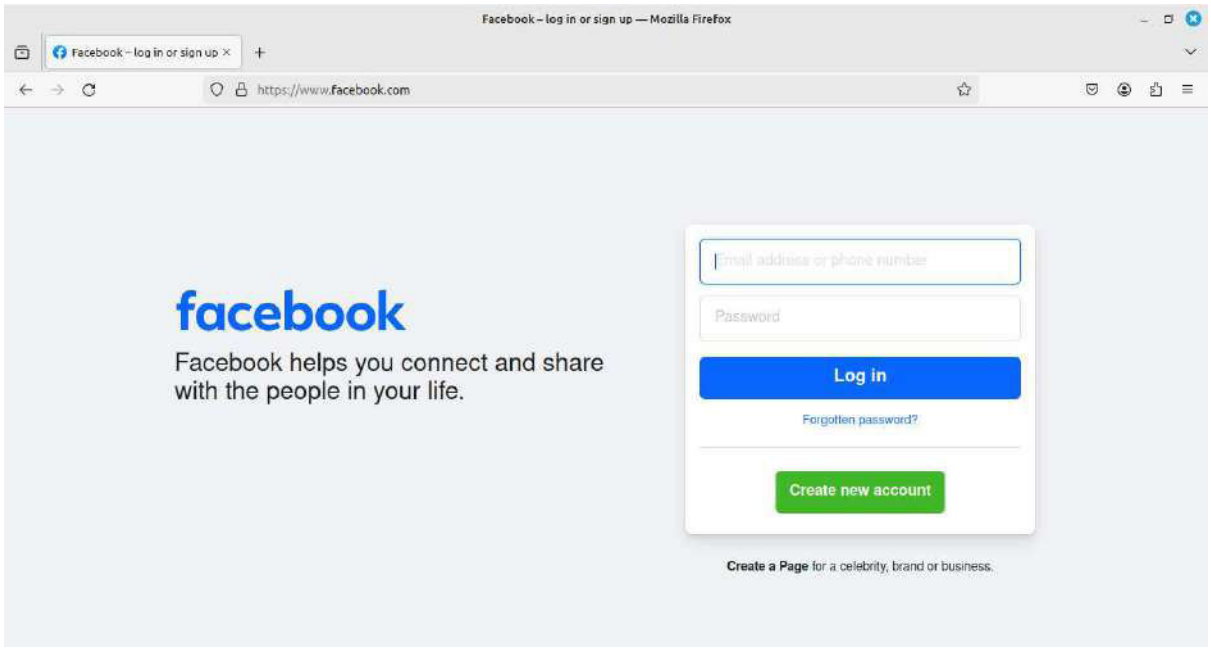
- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the web.

Try Again

Timed Out

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





EXPERIMENT NO: 10

AIM: Simulate Buffer overflow attack using Splint.

THEORY:

Buffer overflow is a mistake that exist in some C implementations. These classes of bugs are dangerous as they write past the end of a buffer or array and hence corrupt the process stack They often change the return address of a process after a function call to a secret memory location where a malicious code is planted.

There are main two types

• Stack based attacks

• Heap based attacks

Heap-based attacks flood the memory space reserved for a program, but the difficulty involved with performing such an attack makes them rare. Stack-based buffer overflows are by far the most common.

Splint is a tool for statically checking C programs for security vulnerabilities and programming mistakes. Splint does many of the traditional lint checks including unused declarations, type inconsistencies, use before definition, unreachable code, ignored return values, execution paths with no return, likely infinite loops, and fall through cases. More powerful checks are made possible by additional information given in source code annotations. Annotations are stylized comments that document assumptions about functions, variables, parameters and types. In addition to the checks specifically enabled by annotations, many of the traditional lint checks are improved by exploiting this additional information. Splint is designed to be flexible and allow programmers to select appropriate points on the effort benefit curve for particular projects. As different checks are turned on and more information is given in code annotations the number of bugs that can be detected increases dramatically.

Problems detected by Splint include:

• Dereferencing a possibly null pointer

• Using possibly undefined storage or returning storage that is not properly defined

• Type mismatches, with greater precision and flexibility than provided by C compilers

• Violations of information hiding

• Dangerous aliasing

• Modifications and global variable uses that are inconsistent with specified interfaces

• Problematic control flow such as likely infinite loops, fall through cases or incomplete switches and suspicious statements.

• Buffer overflow vulnerabilities

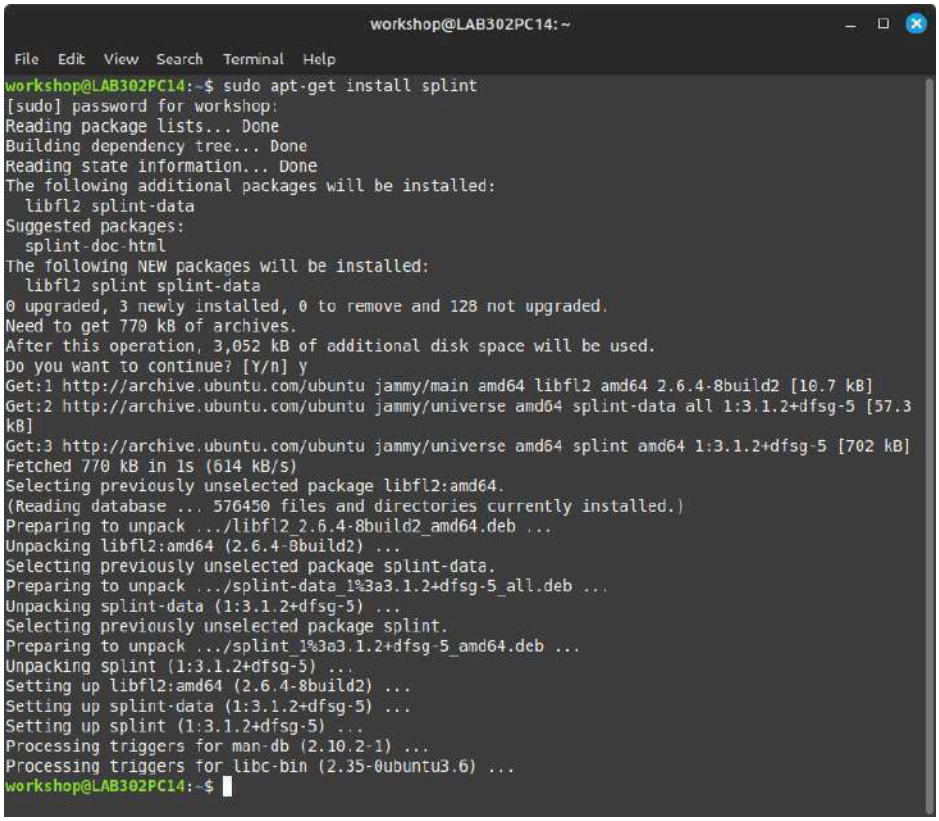
• Dangerous macro implementations or invocations

• Violations of customized naming conventions

Steps :

1. Installation

$ sudo apt-get install splint



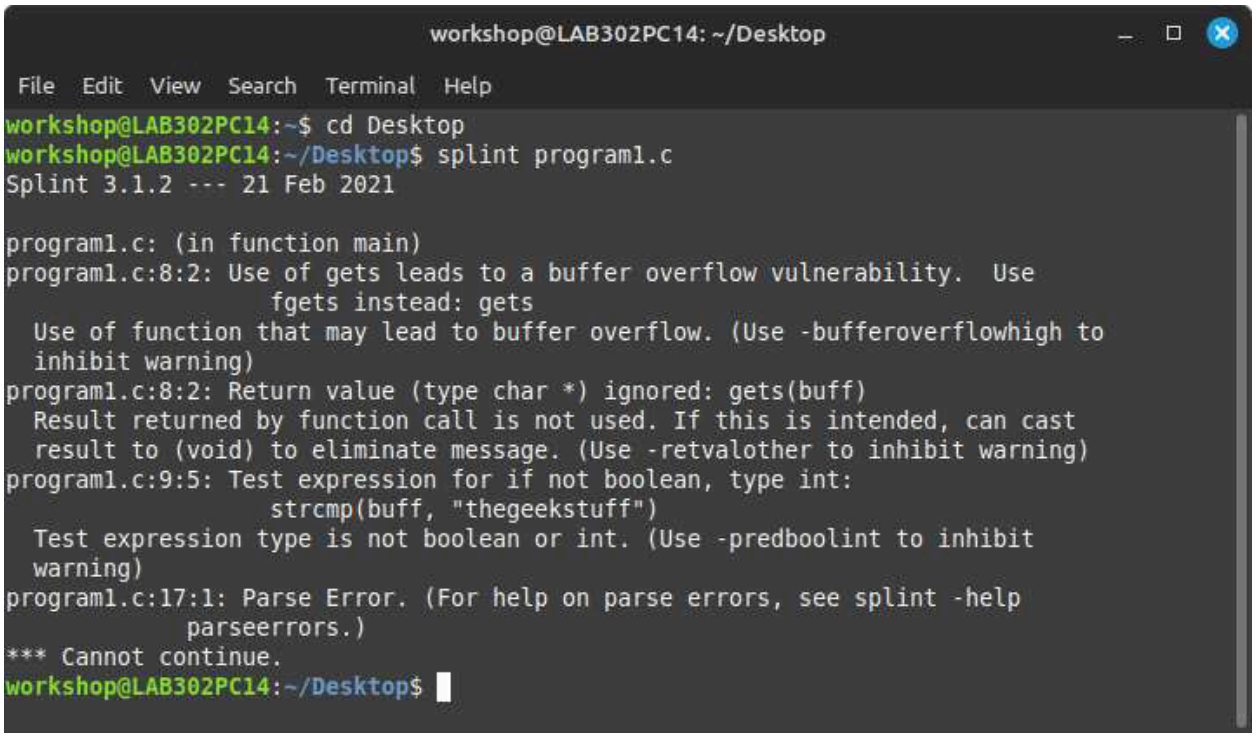
2. Checking Vulnerability

$ splint program1.c

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

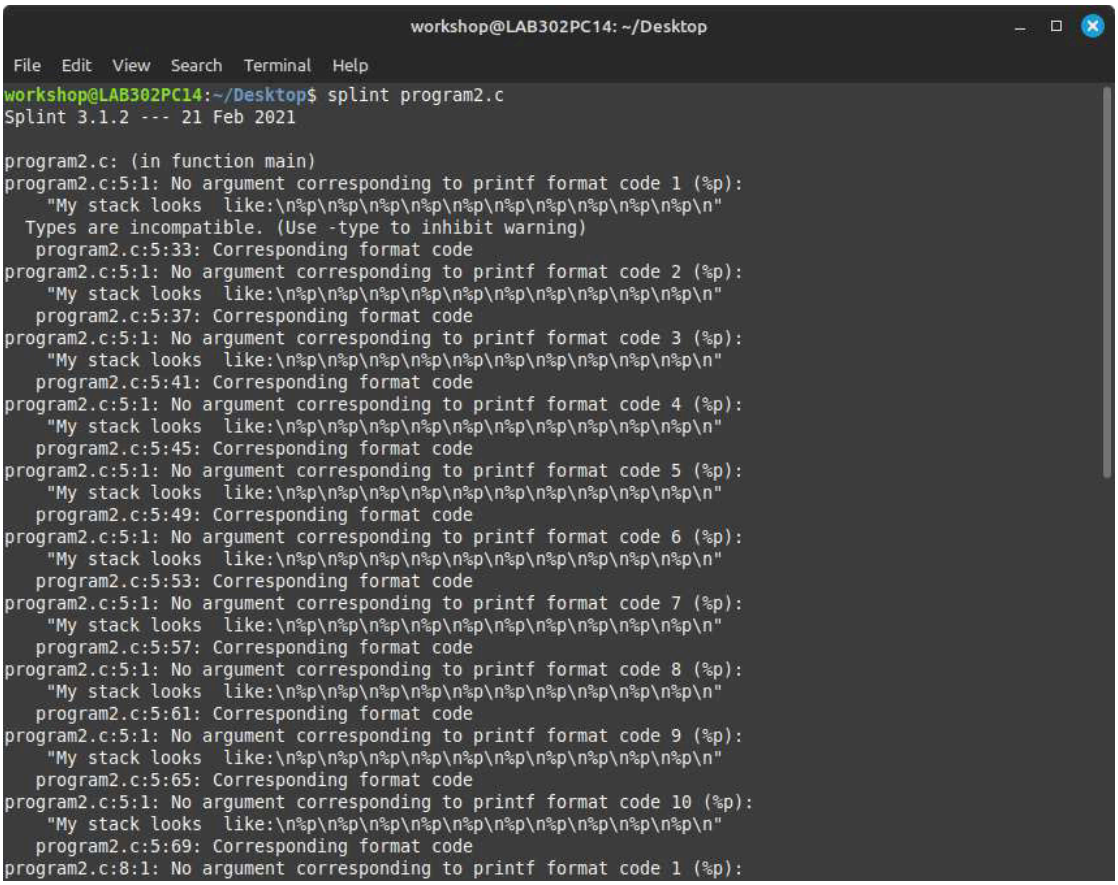
PROGRAM - 1

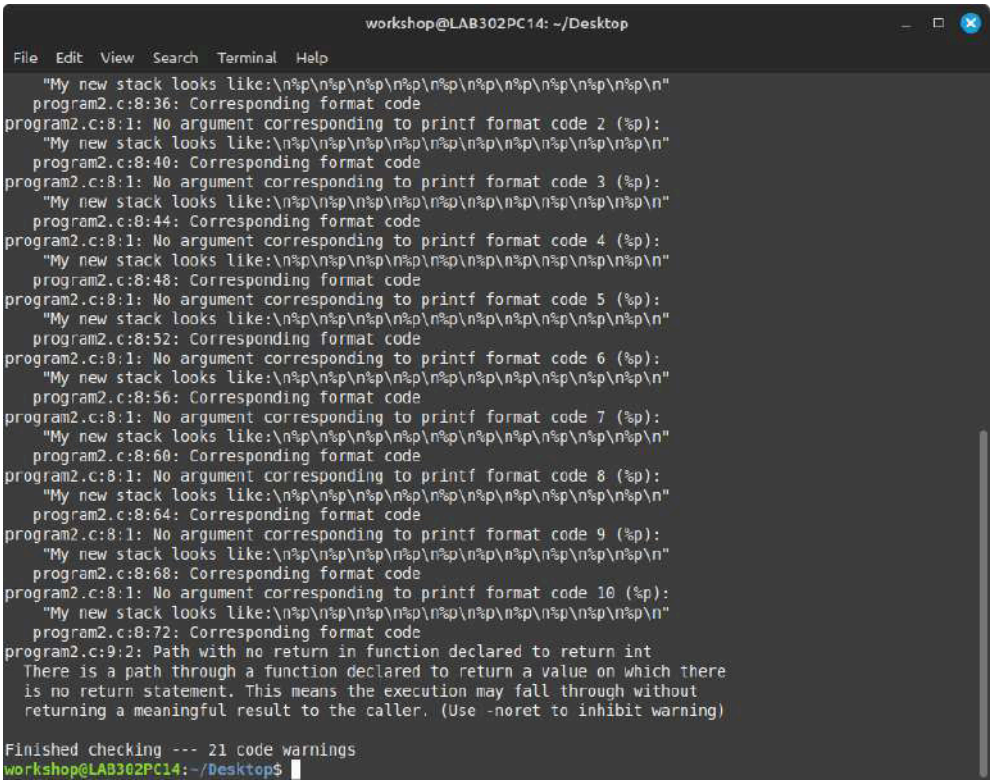
```
#include <stdio.h>
#include <string.h>
int main(void)
{
char buff[15];
int pass = 0;
printf("\n Enter the password : \n");
gets(buff);
if(strcmp(buff, "thegeekstuff"))
{
printf ("\n Wrong Password \n");
}
else
{
printf ("\n Correct Password \n");
pass = 1;
```



PROGRAM - 2

```
#include<stdio.h>

main()

{

char buff[5];

printf("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");

buff[5]='abcdefghijklmnophsgkgfks';

printf("%c\n",buff[5]);

printf("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");

}
```


workshop@LAB302PC14: ~/Desktop
File Edit View Search Terminal Help
workshop@LAB302PC14:~/Desktop$ 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"
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"
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"
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"
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%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%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%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%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%p\n"
program2.c:5:69: Corresponding format code
program2.c:8:1: No argument corresponding to printf format code 1 (%p):

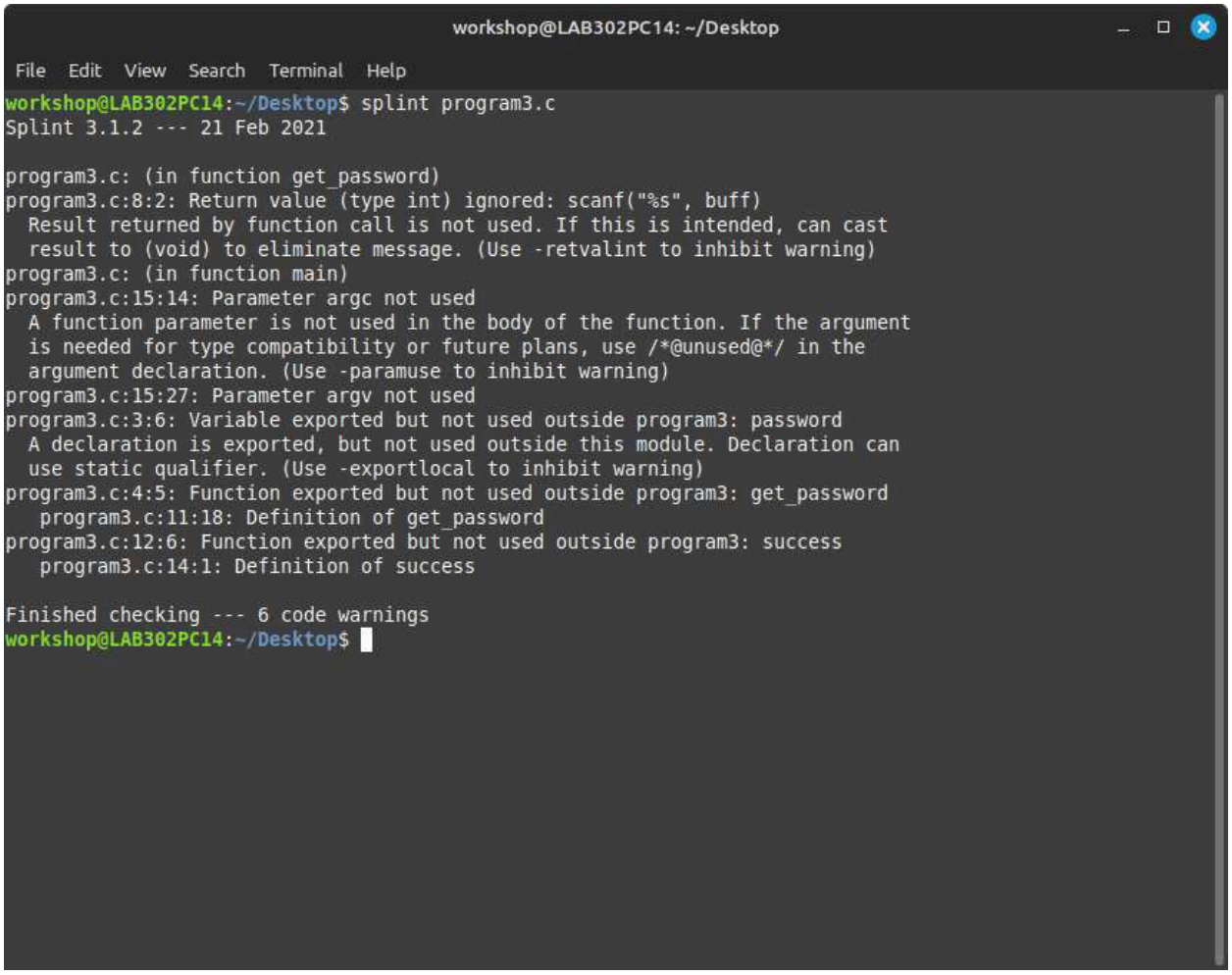

workshop@LAB302PC14: ~/Desktop
File Edit View Search Terminal Help
"My new stack looks like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"
program2.c:8:36: Corresponding format code
program2.c:8:1: No argument corresponding to printf format code 2 (%p):
"My new stack looks like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n"
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%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%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%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%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%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%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%p\n"
program2.c:8:68: Corresponding format code
program2.c:8:1: No argument corresponding to printf format code 10 (%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%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 -noret to inhibit warning)
Finished checking --- 21 code warnings
workshop@LAB302PC14:~/Desktop$

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

return 0;

}


workshop@LAB302PC14: ~/Desktop
File Edit View Search Terminal Help
workshop@LAB302PC14:~/Desktop$ splint program3.c
Splint 3.1.2 --- 21 Feb 2021
program3.c: (in function get_password)
program3.c:8:2: 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 -retvalint to inhibit warning)
program3.c: (in function main)
program3.c:15:14: 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 -paramuse to inhibit warning)
program3.c:15:27: Parameter argv not used
program3.c:3:6: Variable exported but not used outside program3: password
A declaration is exported, but not used outside this module. Declaration can
use static qualifier. (Use -exportlocal to inhibit warning)
program3.c:4:5: Function exported but not used outside program3: get_password
program3.c:11:18: Definition of get_password
program3.c:12:6: Function exported but not used outside program3: success
program3.c:14:1: Definition of success
Finished checking --- 6 code warnings
workshop@LAB302PC14:~/Desktop$

Parth D
C12
32

Kusari 18/3/24 A1

ASSIGNMENT NO: 1

Q) Explain different security services and mechanisms of security.

⇒ Security services can be defined as safeguard controls which are recommended to be placed at various OSI layers in order to strengthen the security mechanisms. There are mainly five security services provided in cryptography and system security, which are shown below:

- Data Confidentiality
- Data Integrity
  - Anti change
  - Anti Replay
- Authentication
  - Peer entity
  - Data origin
- Non Repudiation
  - Proof of origin
  - Proof of Delivery
- Access Control

i) Data Confidentiality: Confidentiality is defined as, an act of protecting information from a unauthorized disclosure to an entity. Data Confidentiality is a security service that keeps the information safe from an unauthorized person. Sometimes it is also referred to as secrecy or privacy. It prevents attacks such as snooping and traffic analysis attack.

ii) Data Integrity: Design to protect data from modification, insertion, deletion and replaying by an adversary. It can be defined as, an act of protecting information from unauthorized modification by an entity. It assures that data received are

exactly as sent by an authorized entity.

iii) Authentication: It provides the identification of the originator. It confirms to the reciever that the data recieved has been sent only by an identified and verified sender.

iv) Non-Repudiation: It is a security service that ensures that an entity cannot refuse the ownership of a previous commitment or an action. Non-repudiation is a property that is most desirable in situations where there are chances of dispute over exchange of data.

v) Access Control: It provides protection against unauthorized access to data. It involves reading, writing, modifying etc.

* Security Mechanisms:

i) Encipherment: It deals with hiding and covering of data which helps data to be confidential. It is achieved by applying mathematical calculations or algorithms which reconstructs the information into a non readable form.

ii) Data Integrity: Data integrity is used by appending value to data which is created by data itself. It is similar to sending packet of information known to both sender and reciever parties and check before and after data is recieved.

iii) Digital Signature: It is a mechanism by which the sender can electronically sign the data and the reciever can electronically verify

iii) Authentication Exchange: In authentication exchange, two entities exchange some message to provide their identity to each other. This is achieved at TCP/IP layer where two-way handshaking mechanism is used to ensure that data is sent or not.

iv) Traffic Padding: It means inserting some bogus data into data traffic to thwart the adversary attempt to use traffic analysis.

v) Routing Control: It means selecting and continuasly changing different available routes between sender and reciever to prevent the opponent from eavesdropping on a particular route.

vi) Notarization: It involves use of trusted third party in communication. It acts as a mediator between sender and reciever so that any chance of conflict is reduced.

vii) Access Control: This mechanism is used to stop unintended access to data which you are sending. It can be achieved by various techniques such as applying passwords, using firewall, or just by adding PIN to a data.

Q]. What are various types of attacks? Explain with examples.

⇒ Confidentiality, Integrity and Availability which are the security goals can be threathened by various security attacks. The security attacks can be broadly classified in two ways:

1. Attacks on Security Goals (Confidentiality, Integrity, Availability):

• Threats on Confidentiality:

1. Snooping: Snooping is unauthorized access to data or interception of data.
for eg: A confidential file being transfered on internet can be intercepted by an attacker and confidential information can be used by him.

2. Traffic Analysis: Refers to monitoring online traffic to obtain information about the user.
Eg: Attacker can find the email address of sender and the reciever.

• Threats on Integrity:

1. Modification: After accessing the information, the attacker can modify or delete the information.
for eg: An attacker can modify the email contents of Organizations in order to benefit him.

2. Masquerading: A masquerade attack is an attack that uses a fake identity, such as network identity to gain unauthorized access to personal computer information through legitimate access identification.

3. Replaying: The attackers obtain a copy of a message sent by a user and later tries to replay it.
for eg: A hacker intercepts a customer's credit card and retransmit over internet to make fraudelent purchases.

• Threats on Integrity:

1. Denial of Service: A Denial-of-service (DOS) attack is an attack meant to shut down a machine or a network, making it inaccessible to its inteded users. Buffer Overflow Attack is the most common DOS attack.
for eg: The attacker floods a server with internet traffic to prevent users from accessing connected to online services and sites.

8] Attacks based on effects on the system:

1) Passive Attack: The objective or goal of the attacker is to obtain information. This means the attacker or attack does not modify the data or harm the system.
for eg: Traffic analysis is one of the most common Passive attack used by attackers.

2) Active Attack: An active attack may change the data or harm the system. It threatens the integrity and availability.
for eg: Modification of messages, Masquerading etc.

9]. List two latest viruses on internet and explain them.

⇒ A malicious program used to replicate, make modifications to other systems and enter its own code is known as an internet virus. It's replication can impact various features of computer system and then the system becomes a compromised device.

Some of the latest computer viruses around the internet are explained below:

1) I-worm: It is a virus capable of spreading via making it's replica. It sends itself to other connected devices via email. Moreover, it is also known as worm because it spreads itself via networks.

2) Dyreza: Using this malware, adversaries steal financial credentials from anonymous targets. Usually, it is prevented via email/web pop-ups in order to fool users into installing malware.

3) Zeus Trojan: It is basically used to steal confidential financial credentials from anonymous targets. Spreading via emails, websites and other online sources.

4) Yahron Ransomware: Yahron is a ransomware type virus that is promoted as 'Ransomware as service' (RaaS). After successful infiltration, Yahron encrypts most of the data and appends filename with .yahron extension.

5) Asharoth Malware: It is an information stealer malware. It transfers confidential information from an affected victim like account id and passwords, keystrokes and other data to the attacker.

Busari 18/3/24 (At)

ASSIGNMENT NO: 2

Q] Write a short note on digital signatures and digital certificates.

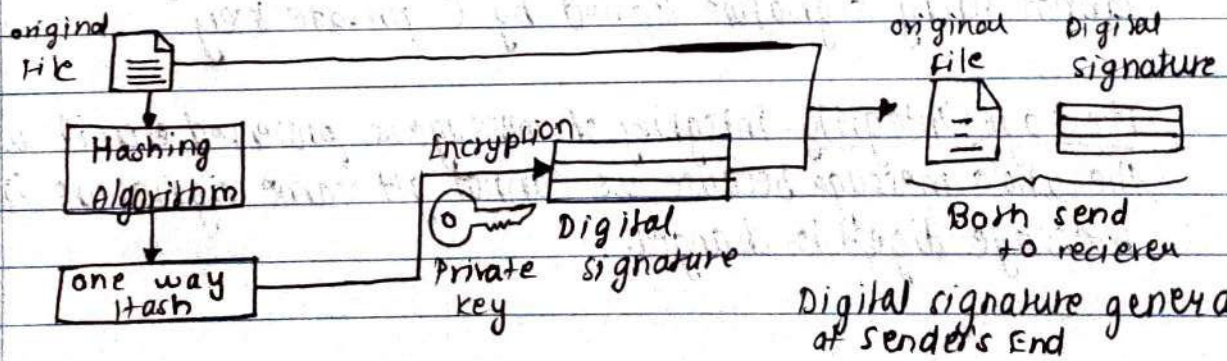
⇒ Authentication is a way to build trust in communicating parties or systems. Authentication serves as the crucial protection mechanism towards securing a communication - ensuring that the right set of parties are involved in the communication and no-one else.

Broadly, there are four ways that provide major authentication services as listed below:

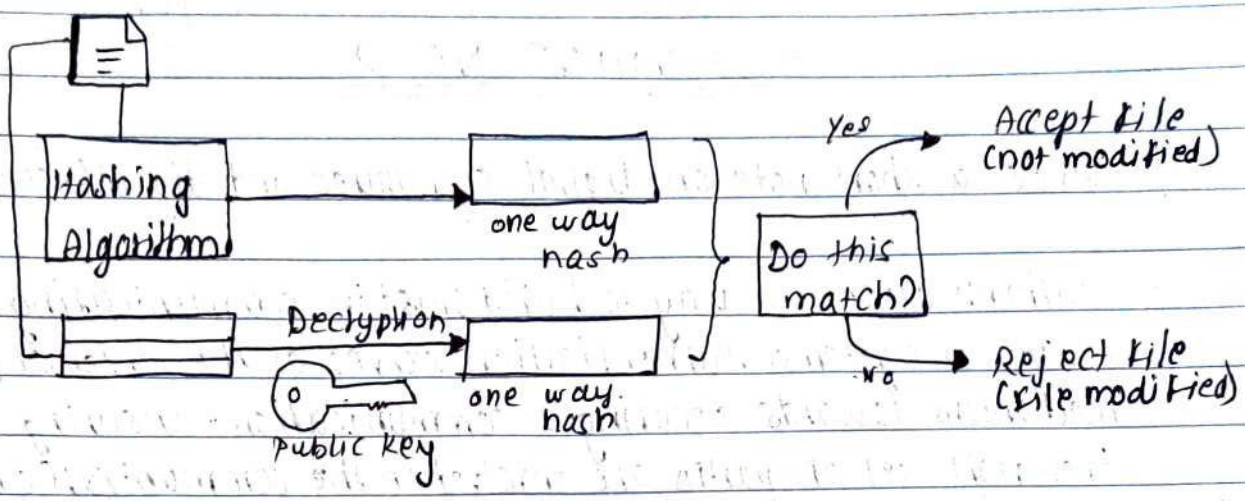
i) Hash and MAC
ii) Digital Signature
iii) x.509 Digital Certificate
iv) kerberos

We will discuss about Digital signature and Digital Certificate in detail.

* Digital Signature: A digital signature is a hash value that has been encrypted with the sender's private key. The act of signing means encrypting the message's hash value with a private key (since no one else knows the sender's private key



Digital signature generation at sender's End



Digital signature verification at reciever's end.

1. Message digest is computed by applying hash function on the message and then message and then message digest is encrypted using private key of sender to form digital signature.
2. Digital signature is then transmitted with the message.
3. Reciever Decrypts the digital signature using sender's public key.
4. The reciever now has the message digest.
5. The reciever can compute the message digest from the message.
6. The message digest computed by reciever and the decrypted message digest must be same for ensuring integrity.

Message Authentication: A secure digital signature scheme can provide message authentication. B can verify a message sent by A because A public key is used for verification. A's public key cannot verify signature signed by C private key.

Message Integrity: Integrity of message is preserved even if we sign the whole message because we cannot get same signature if the message digest is changed.

Non-Repudiation: Non-repudiation can be provided using a trusted party.

Confidentiality: Signature does not provide confidentiality. To provide confidentiality, the message digest can be encrypted using reciever's public key which can further be decrypted using reciever's private key.

* Digital Certificate: A digital certificate is a signed data structure that binds a public key to a person, computer or an organization.
The certificate uniquely identifies the owner of a public key. Certificates are issued by Certification Authorities (CA). In a secure communication, certificates are used to identify the systems and establish trust amongst the communication parties. Three versions of the X.509 certificates are evolved. The most commonly used version of X.509 is version 3. X.509 certificates are used to evolve secure communication.

The format of X.509 certificate is given below:

1) Version number: It defines the X.509 version that concerns the certificate.

2) Serial Number: It is the unique number that the certified authorities issue.

3) Signature Algorithm Identifier: This is the algorithm used for signing the certificate.

4) Issuer name: Tells about the x.509 name of the certified authority which signed and created certificates.

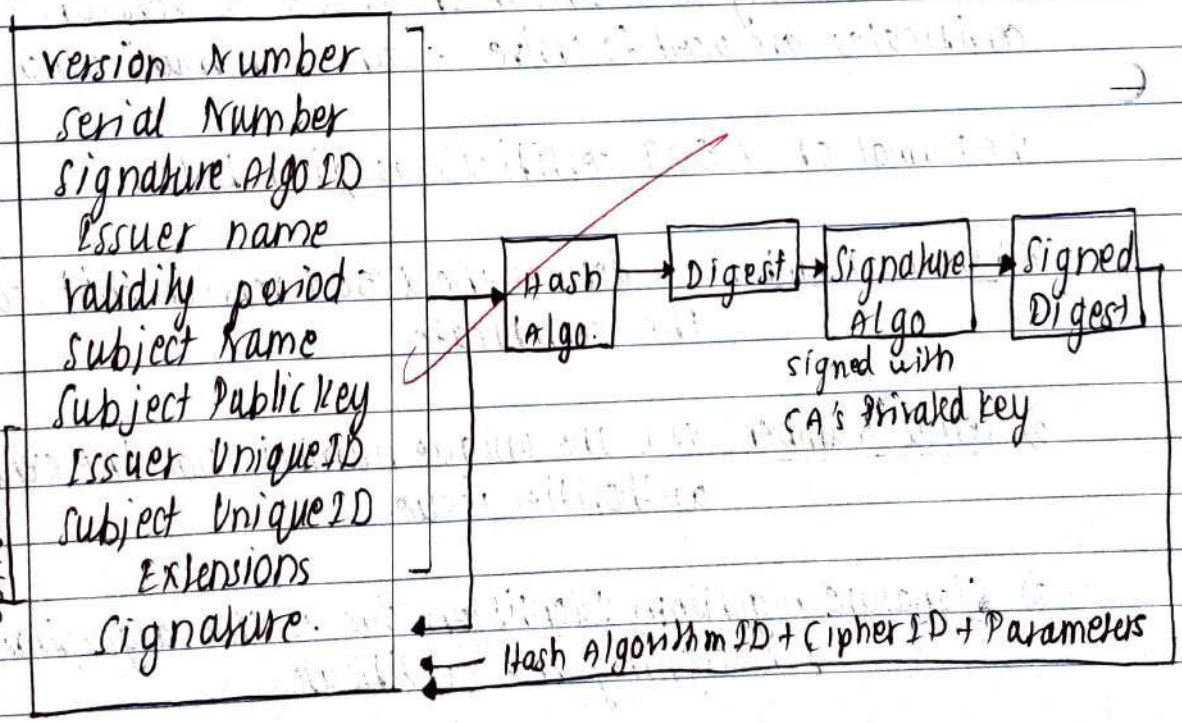
5) Period of validity: Defines the period of validity for which the certificate is valid.

6) Subject Name: Tells the name of the user to whom the certificate has been issued.

7) Subject's Publickey info: Defines the public key of subject along with identifier of the algorithm.

8) Extension Block: Field contains additional standard information.

9) Signature: Contains hash code of all other fields encrypted by certified authority private key.



Parth D. C12 32

Busari 4/3/24 (A+)

ASSIGNMENT NO: 3

Q] List all the software vulnerabilities. How they exploited to launch an attack?

→ Software vulnaribilities are weaknesses or flaws in the software systems that can be exploited by attackers to compromise security, integrity or availability of system or data. These vulnaribilities can range from programming errors and design flaws to improper configurations and weak authentication mechanisms. Exploiting those vulnaribilites can lead to various cyber threats, including unauthorized access, data breaches, system crashes, and execution of malicious code. Identifying and addressing software vulnerabilities is essential to maintain the resilience and trustworthiness of software applications and systems in the face of evolving cyber threats.

Following are some of the software vulnerabilities:

1. Butter Overflow: Occurs when an attacker modifies or writes more data to the butter which leads to change in execution of program, system crashes and execution of malicious code.

2. SQL Injection: Exploit vulnerabilities in web applications that interact with databases, allowing attackers to manipulate SQL queries to gain unauthorized access or modify data.

3. Cryptographic failures: Improper implementation of encryption, hashing or key management mechanisms, leading to data expos-ure

or integrity issues.

4. Cross-Site Scripting (XSS): Allows attackers to inject malicious scripts into web pages viewed by other users, compromising their security or stealing information.

5. Clickjacking: Tricks users into clicking on malicious links or buttons by disguising them as legitimate elements on webpage.

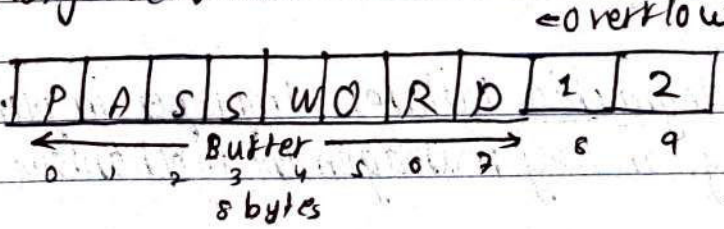
6. Denial of Service (DOS): Overwhelms a system or network with excessive traffic, rendering it unavailable to legitimate users.

7. Insecure Transport Layer (TLS): Utilizes outdated or weak encryption protocols, allowing attackers to manipulate encrypted communications.

BUFFER OVERFLOW ATTACK:

Buffers are memory storage regions that temporarily hold data while being transferred from one location to another. A buffer overflow occurs when the volume of data exceeds the storage capacity of memory buffer. As a result, the program attempting to write the data to buffer overwrites adjacent memory locations.

Eg:



• How does Butter overflow attack happen:
→ Attackers input more data than the butter can handle, thus overflowing it. For eg: If the butter is of size 8 bytes, the attacker injects data more than the size of butter, thus data gets overwritten on adjacent memory locations.
→ Malicious code injected into butter can execute arbitary commands or exploit system vulnerabilities.

• Prevention:
1. Address space Randomization: Randomly moves around the address space locations of data. Attackers need to know the locality of executable code and randomizing it makes it impossible to access.
2. Data Execution Prevention: Flags certain areas of memory as non executable, which stops attack from running code in a non-executable regions.

SQL INJECTION:

SQL injection is a type of an injection attack that makes it possible to execute malicious SQL statement. These statements control a database server behind a web application. Attackers can use SQL injection vulnerabilities to bypass application security measures. They can go around authentication and authorization of a web page or web application and retrieve the content of entire SQL database. They can also use SQL injection to add, modify, and delete records in the database.

How it happens: • Attackers input malicious SQL commands into web application forms or URL parameters.
• Improper sanitized inputs are directly concatenated into sql queries allowing attackers to manipulate them.

For eg: If the attacker wants to get administrative privilleges of a database, they can use the following query:

```
SELECT id FROM users WHERE username='username' AND
password='password' or 1=1
```

Because of 1=1, where clause returns first id which is generally the administrator or database, thus attacker gets privilleges.

Prevention: 1. Use parameterized queries or prepared statements to seperate SQL code from user input.
2. Implement strict input validation and data sanitization to ensure that input adheres to expected formats and does not contain malicious code.
3. Web Application Firewall (WAF): Deploy a web application firewall to filter and block malicious HTTP requests, including those attempting SQL Injection attack.