Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

EXPERIMENT NO. 1

AIM: Implementation of Extended Euclidean algorithm.

Description:

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

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

- 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.

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

IMPLEMENTATION/CODE:

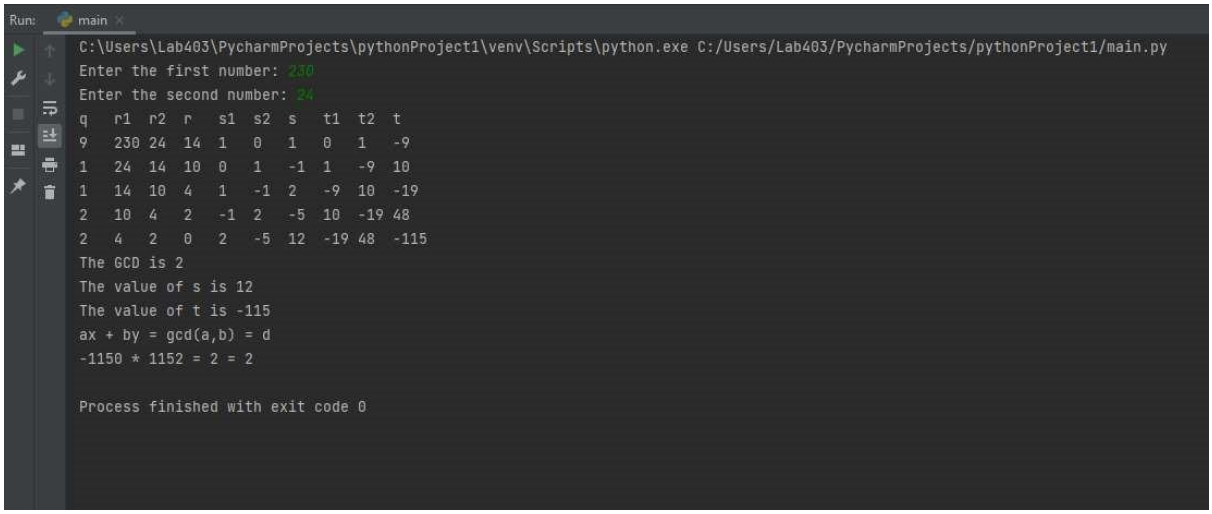
```
a = int(input("Enter the first number: ")) b = int(input("Enter the
second number: ")) num1 = a num2 = b


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)
```

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

RESULTS:



Roll No. : 2103110

Batch : C23

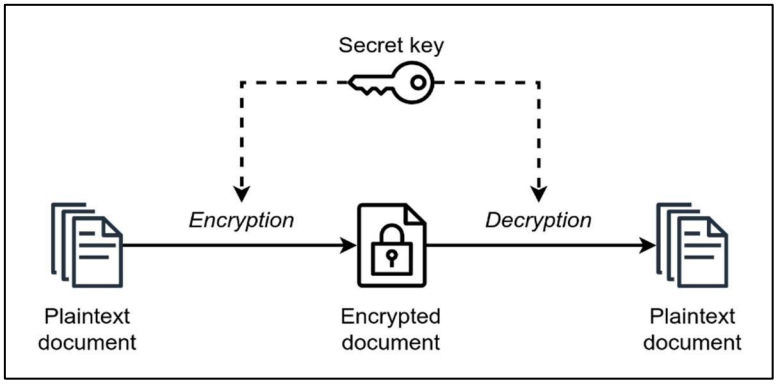
Name : Rishab Mandal

EXPERIMENT NO. 2

AIM: Implementation of Caesar Cipher

Description/Algorithm:

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 par es 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 encrypt on, in comparison to public-key encrypt on.

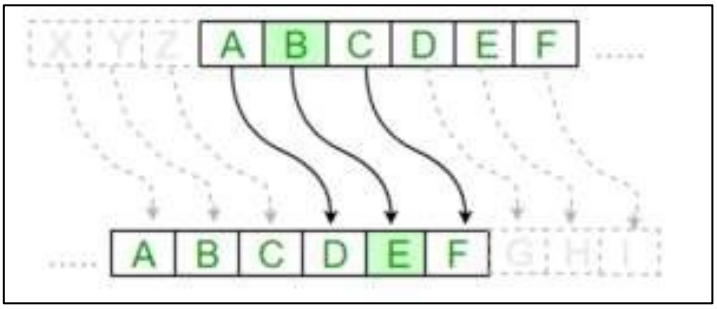


In cryptography, a substitution cipher is a method of encrypting in which units of plaintext are replaced with the ciphertext, in a defined manner, with the help of a key; the "units" may be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth.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

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal



The Caesar cipher is one of the simplest and most widely known encryption techniques. It is a type of subset on cipher where each letter in the plaintext is shifted a certain number of places down or up the alphabet. For example, with a shi of 3, 'A' would be replaced by 'D', 'B' would become 'E', and so on. The method is named a er 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 Caesar 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".

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

2. A er ge ng the indices of every character, use the formula given below: CT(i) = (PT(i) + (key)mod 26) mod 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 Caesar Cipher is done using the following steps:

1. Convert the Cipher text characters into corresponding indices using the above table.
2. Arranging 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.

Implementation/Code :

```
import   java.io.*;  public
class CeaserCipher
{
   sta c String Encryp on(String PlainText, int Key)
   {
      int len = PlainText.length();
    char[] pt,ct;        pt = new
char[len];         ct = new
```

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

```
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] = ' ';
         }
      }
```

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

```
      String CipherText = new String(ct);
return CipherText;
   }
   sta c String Decryp on(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] <= '/')
         {
```

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

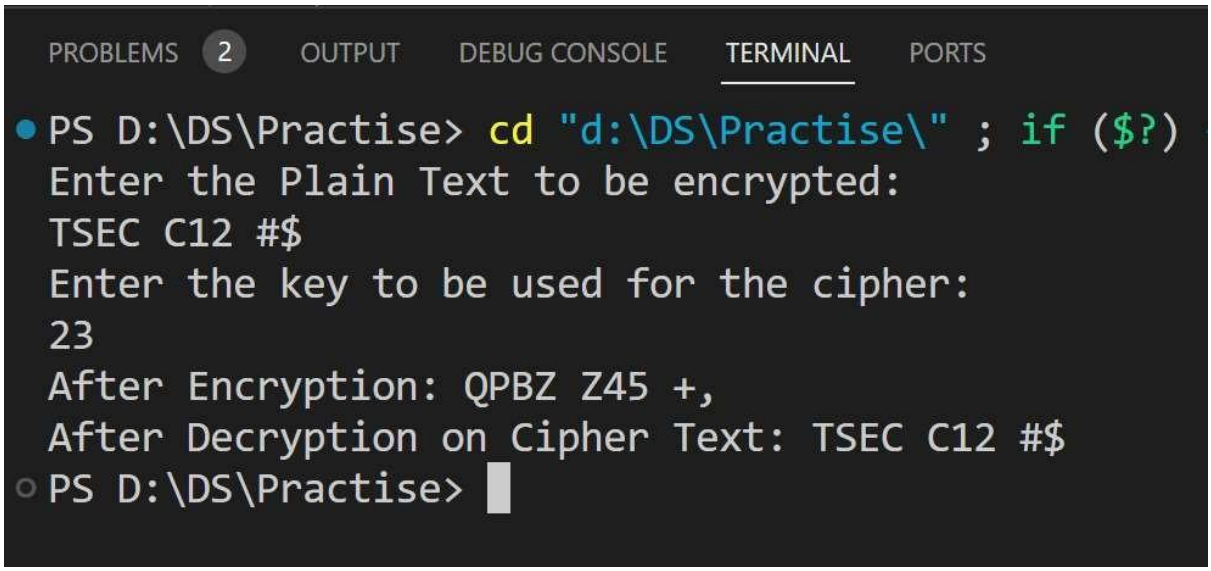
```
            pt[i] = (char)( ( ( ( (int)ct[i] + 30 - 33 ) - (Key%26)) % 15 ) + 33 );
        }
        if(ct[i] == ' ')
        {
pt[i] = ' ';
        }
      }
      String PlainText = new String(pt);
return PlainText;
   }
   public sta c void main(String[] args) throws IOExcep on
   {
      BufferedReader 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 = Encryp on(PlainText, Key);
      System.out.println("A er Encryp on: "+CipherText);

      String newPlainText = Decryp on(CipherText, Key);
      System.out.println("A er Decryp on on Cipher Text: "+newPlainText);
   }
}
```

Results/Screenshots :

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal



Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

EXPERIMENT NO: 3

AIM: Implementation of Playfair Cipher.

DESCRIPTION/ALGORITHM:

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 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.

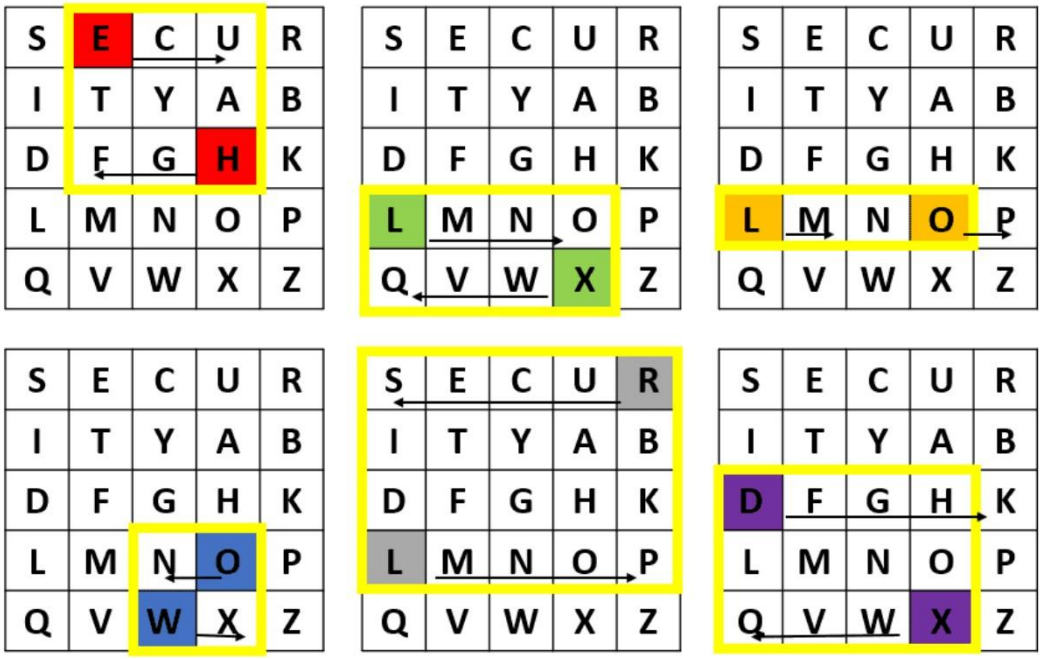
S	E	C	U	R
I	T	Y	A	B
D	F	G	H	K
L	M	N	O	P
Q	V	W	X	z

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.

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal



Implementation/Code :

```
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;
```

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

```
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)
{
```

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

```
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;
```

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

```
}
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[f
irstPos[0]][(firstPos[1] + 1)
% 5]);
cipherText.append(matrix[s
econdPos[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[f
irstPos[0]][secondPos[1]]);
```

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

```
cipherText.append(matrix[s
econdPos[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);
}
}
```

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

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>
```

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

EXPERIMENT NO. 4

AIM:

Implementation of Euler's Totient Function.

DESCRIPTION/ALGORITHM:

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 func on 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 func on 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 func on 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. 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.

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

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 To ent Func on can be given as:

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

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 To ent is given as:

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

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

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

IMPLEMENTATION/CODE :

```
import java.io.*;
public class EulersTo  ent {
public sta  c int eulerTo  ent(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;
```

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

```
} }
if (n > 1)
result -= result / n;
return result; }
public sta  c int gcd(int a, int b) { if (b == 0)
return a;
return gcd(b, a % b);
}
public sta  c void main(String[] args) throws IOExcep  on {
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter the number whose Euler's To  ent is to be found out: "); int num =
Integer.parseInt(br.readLine());
int result = eulerTo  ent(num);
System.out.println("The Euler's To  ent is: "+ result);
System.out.println("Rela  vely Prime numbers to "+num+" are: "); for(int i = 0 ; i < num ;
i++)
{
if(gcd(num, i) == 1) {
System.out.print(i+" "); }
} }
}
```

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

RESULTS/SCREENSHOTS:

```
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>
```

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

EXPERIMENT NO. 5

AIM:

Implement RSA cryptosystems.

DESCRIPTION/ALGORITHM:

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 nonrepudiation 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:

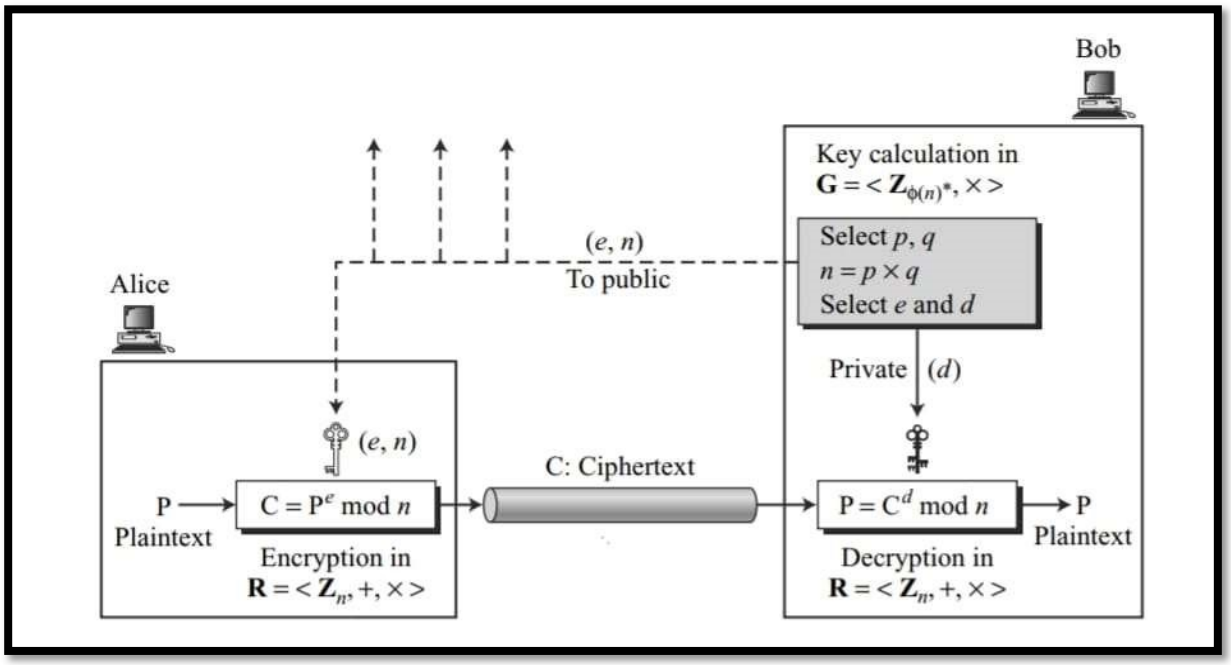
Non-repudiation on ensures that a sender cannot deny sending a message and that a recipient cannot deny receiving it.

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

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal



ALGORITHM:

1. Key Genera on:

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).

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

3. Decryption:

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

IMPLEMENTATION/CODE :

```
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; }
sta  c int gcd(int a, int b) {
if(b == 0) return a;
return gcd(b, a%b); }
sta  c 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); }
sta  c 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\  1\  2\  "); 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;
```

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

```
} System.out.println(q+"\t"+r1+"\t"+r2+"\t"+r+"\t"+t1+"\t"+t2+"\t"+t); if(t1 < 0)
{
return (t1 + phi);
}
return t1; }
sta  c int power(int base, int exponent) {
int result = 1;
for(int i = 0 ; i < exponent; i++) {
result = result*base; }
return result; }
sta  c int RSAencyrpt(int m, int e, int n) {
int temp = power(m, e); int result = temp % n; return result;
}
sta  c int RSAdecrypt(int c, int d, int n) {
int temp = power(c,d); int result = temp%n; return result;
}
public sta  c 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); inte=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("A  er Encryption: "+cipher);
System.out.println("After Decryption: "+pt);
```

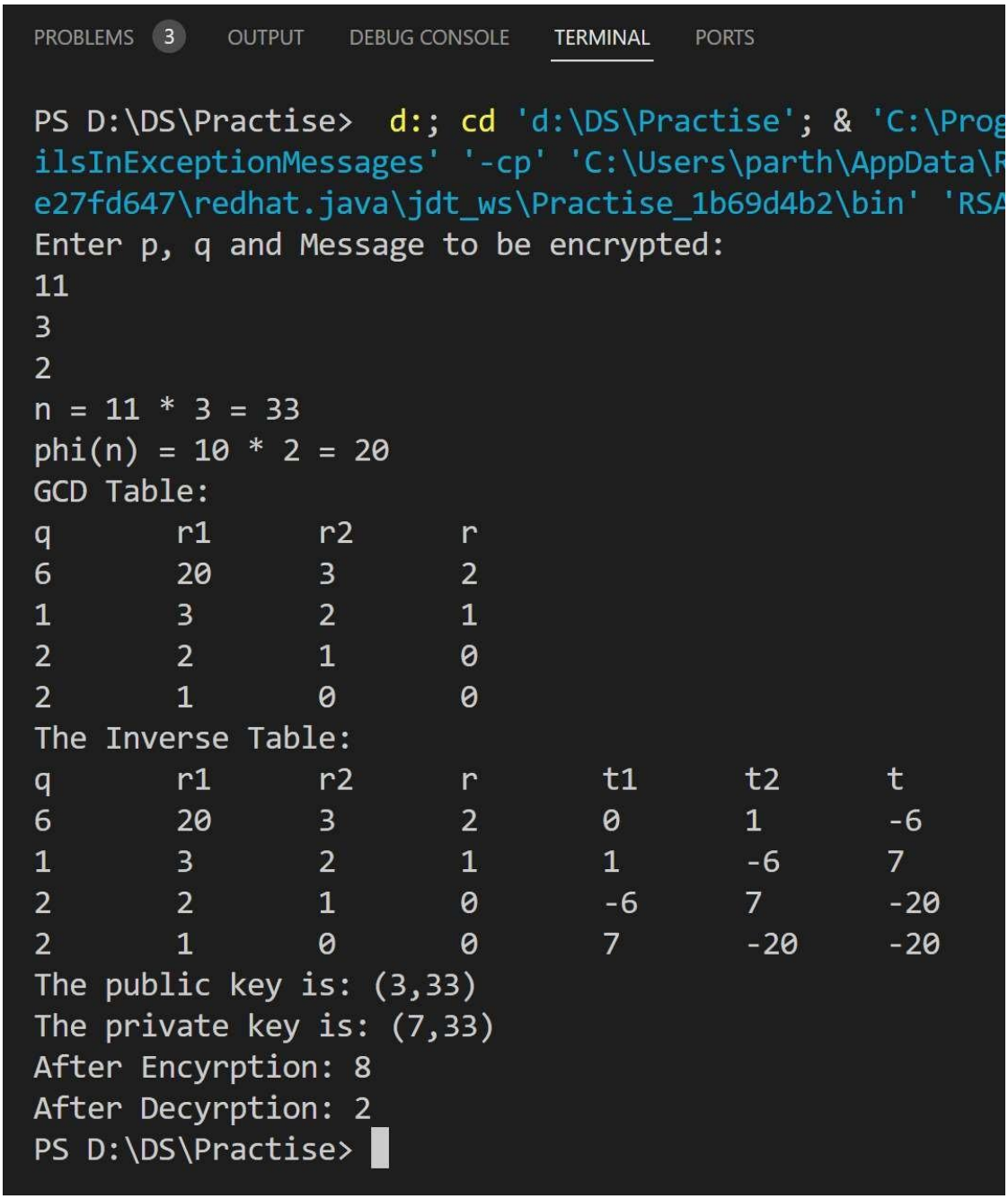
Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

```
}}
}
```

RESULT/SCREENSHOTS:



Roll No. : 2103110

Batch : C23

Name: Rishab Mandal

EXPERIMENT NO: 6

AIM :

Implement Diffie Hellman Key Exchange Algorithm.

DESCRIPTION/ALGORITHM :

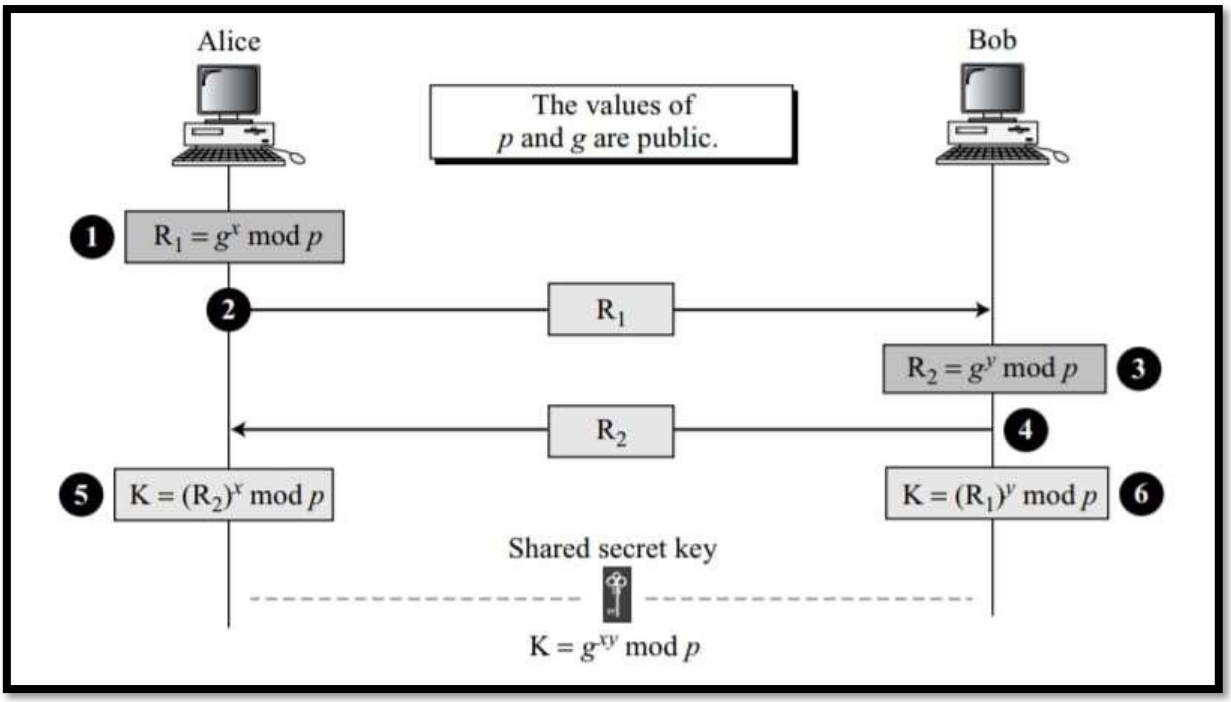
The Diffie-Hellman key exchange algorithm is a fundamental method used in cryptography to securely exchange cryptographic keys over a public channel. It was developed independently by Whitfield Diffie and Hellman in 1976 and is one of the first public-key protocols. The beauty of the Diffie-Hellman algorithm 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 pairs 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.

Roll No. : 2103110

Batch : C23

Name: Rishab Mandal



Diagrammatic Representation of Diffie Hellman Key Exchange Algorithm

ALGORITHM:

The steps are as follows:

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

$R1 = g^x \bmod p.$

2. Bob chooses another large random number y such that $0 \leq y \leq p - 1$ 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 $K = (R1)^y \bmod p.$

Roll No. : 2103110

Batch : C23

Name: Rishab Mandal

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

IMPLEMENTATION/CODE:

```
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);
```

Roll No. : 2103110

Batch : C23

Name: Rishab Mandal

```
System.out.println("Keys are: "+final_key1+", "+final_key2); }
public sta  c int power(int a, int b, int mod) { int result = 1;
int count = 0;
intx=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;
ints=s1-q*s2; intt=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("");
```

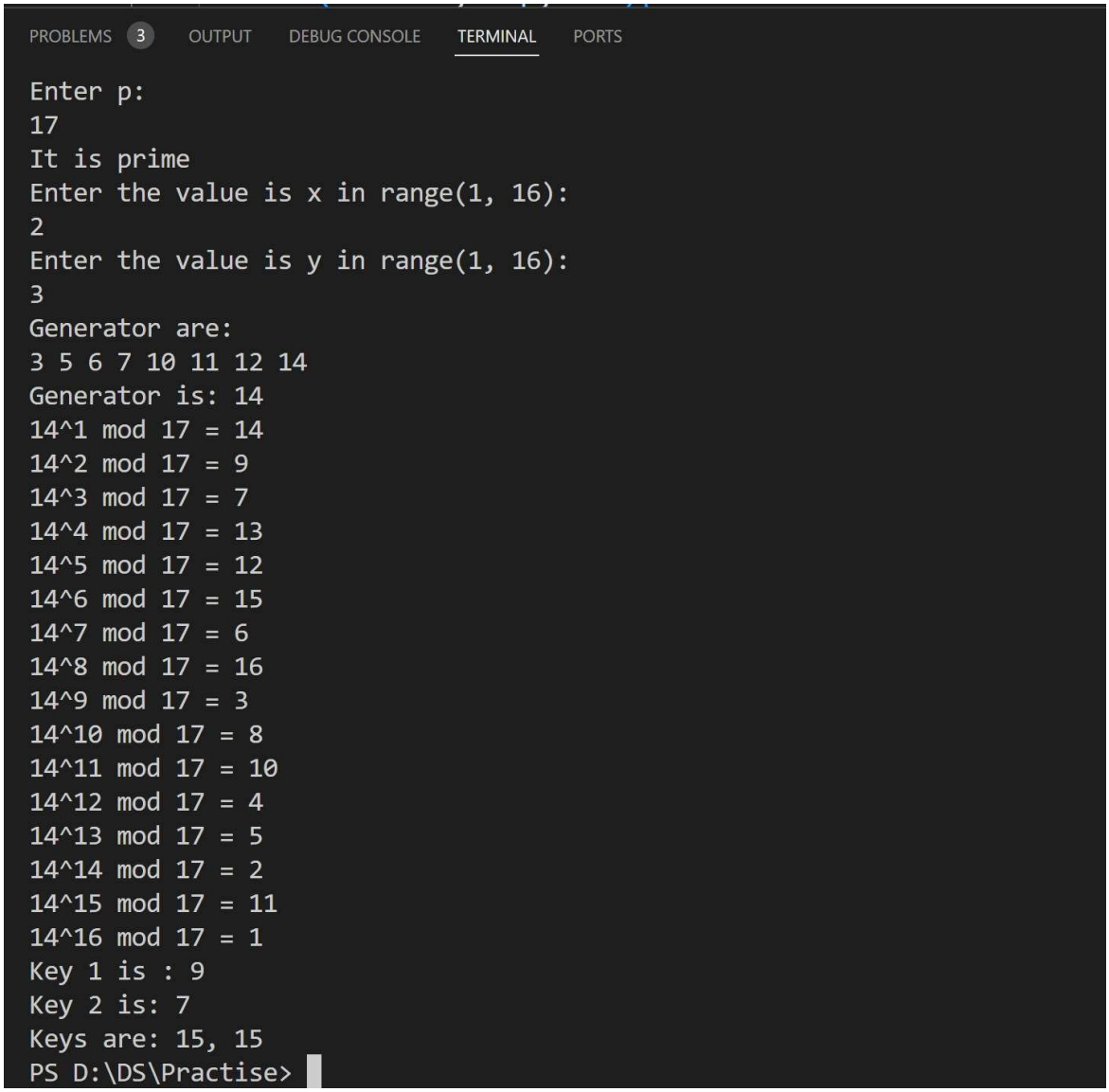
Roll No. : 2103110

Batch : C23

Name: Rishab Mandal

```
return ans;
} }
```

RESULT/SCREENSHOTS:



Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

EXPERIMENT NO. 7

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

Description/Algorithm:

Whois

Command: whois tsec.edu

Explanation:

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

1. Registrar Information: This includes the name of the organization that registered the domain, their contact information, and possibly their website.

2. Domain Registration Details: This includes the date when the domain was registered and the date when it will expire.

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

Roll No. : 2103110

Batch : C23

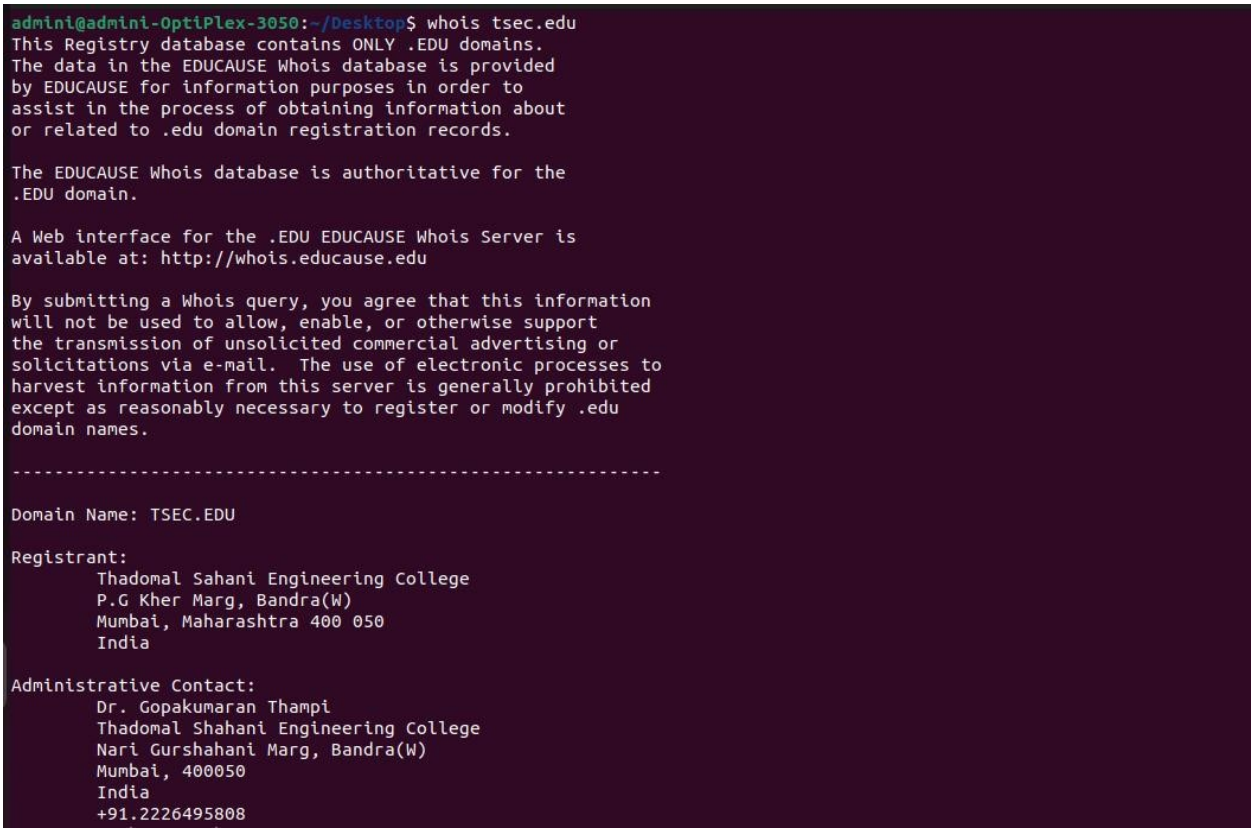
Name : Rishab Mandal

4. Name Servers: These are the authoritative servers responsible for resolving the domain to its associated IP address.

5. Domain Status: This indicates whether the domain is active, on hold, pending transfer, or has expired.

6. DNS Records: Some WHOIS servers provide information about the DNS records associated with the domain, such as A records, MX records, and NS records.

Screenshot:



Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

Dig

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

Command: dig www.google.com

Explanation:

1. DNS Resolution: The primary purpose of using `dig` is to resolve domain names to IP addresses. For example, when you type `www.google.com` in a web browser, your computer needs to know the corresponding IP address to establish a connection to Google's servers. `dig` helps you understand this process by showing you the IP address associated with the domain.

2. Name Servers: DNS servers store information about domain names and their corresponding IP addresses. When you run `dig`, it also provides information about the authoritative name servers responsible for the domain. These name servers hold the official records for the domain and can provide additional details if needed.

3. Additional Records: Apart from the IP address, `dig` can also retrieve other DNS records associated with the domain, such as MX (Mail Exchange) records for email servers, TXT records for text information like SPF (Sender Policy Framework) records used for email authentication, and more. These additional records can be crucial for understanding how a domain is configured and used.

4. Debugging and Troubleshooting: `dig` is a valuable tool for network administrators and developers for debugging and troubleshooting DNS-related issues. By inspecting the output of `dig`, you can identify DNS misconfigurations, check DNS propagation status, and diagnose connectivity problems.

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

Screenshot:

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

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

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

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

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

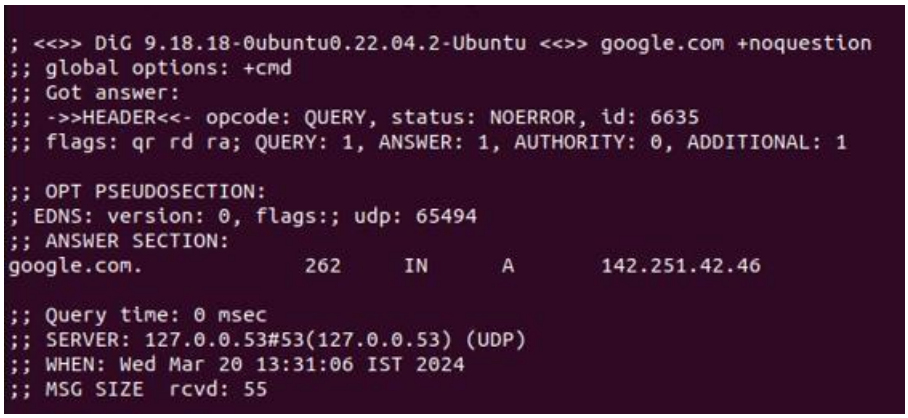
Command: dig google.com +noquestion

Explanation:

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

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

Screenshot:



Command: dig google.com +nocomments

Explanation:

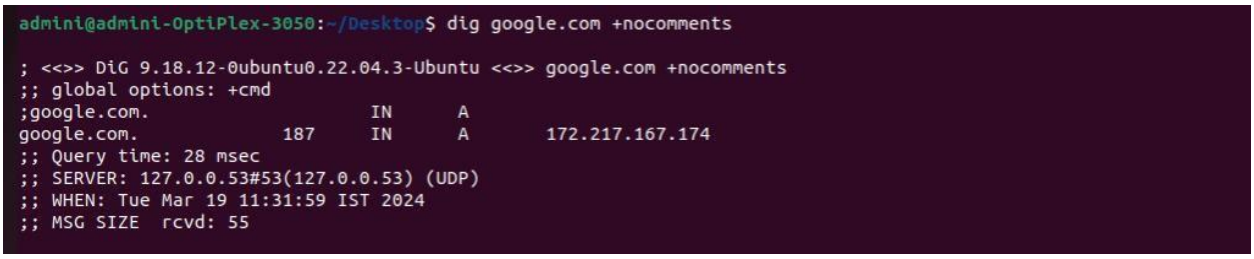
In this command, google.com is the domain name being queried. By using the +nocomments option, the output will not display any comments that might be included in the DNS server's response.

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

Screenshot:



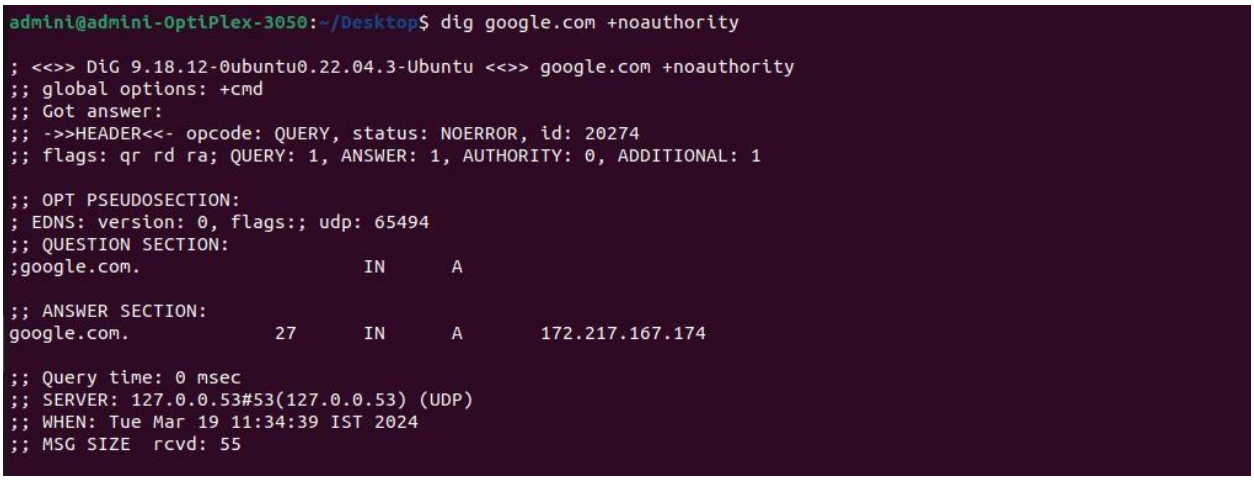
Command:

dig google.com +noauthority

Explanation:

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

Screenshot:



Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

Command: dig google.com +noadditional

Explanation:

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

Screenshot:

Command:

dig google.com +nostats

Explanation:

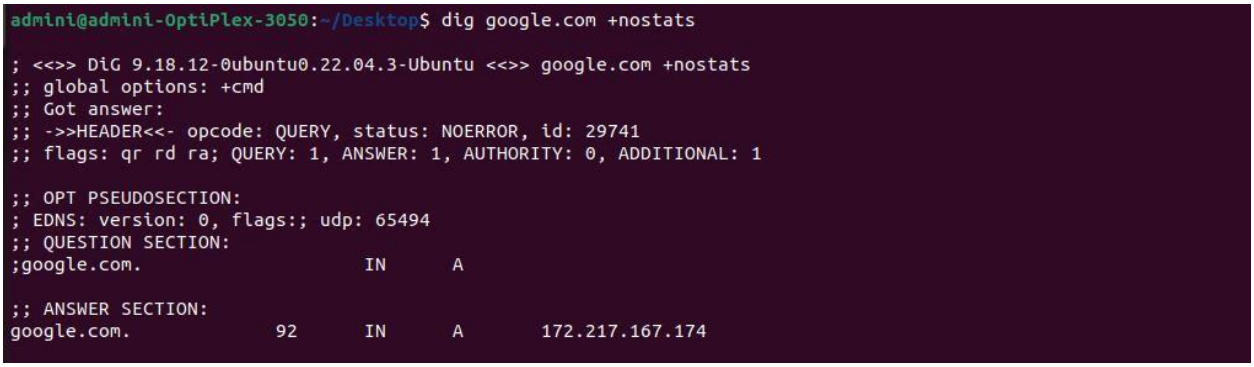
By utilizing the +nostats option, the output will not include any statistical information regarding the query execution.

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

Screenshot:



Command:

Dig google.com +noanswer

Explanation:

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

Screenshot:

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

Command:

Dig google.com MX +noall +answer

Explanation:

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

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

Screenshot:

Command:

Dig google.com NS +noall +answer

Explanation:

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

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

Screenshot:

Command:

dig -t ANY google.com +noall +answer

Explanation:

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

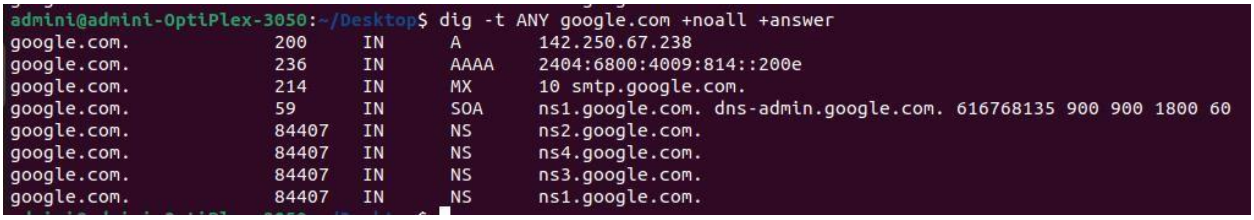
When you execute this command, it sends a DNS query to a DNS server to retrieve all types of DNS records associated with the domain google.com, including A, AAAA, MX, NS, and others. By using the +noall and +answer options, the output will only include the DNS records found in the answer section, excluding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

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

Screenshot:



Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

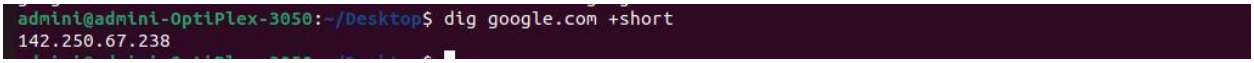
Command:

dig google.com +short

Explanation:

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

Screenshot:



Command:

Dig -x 209.132.183.81

Explanation:

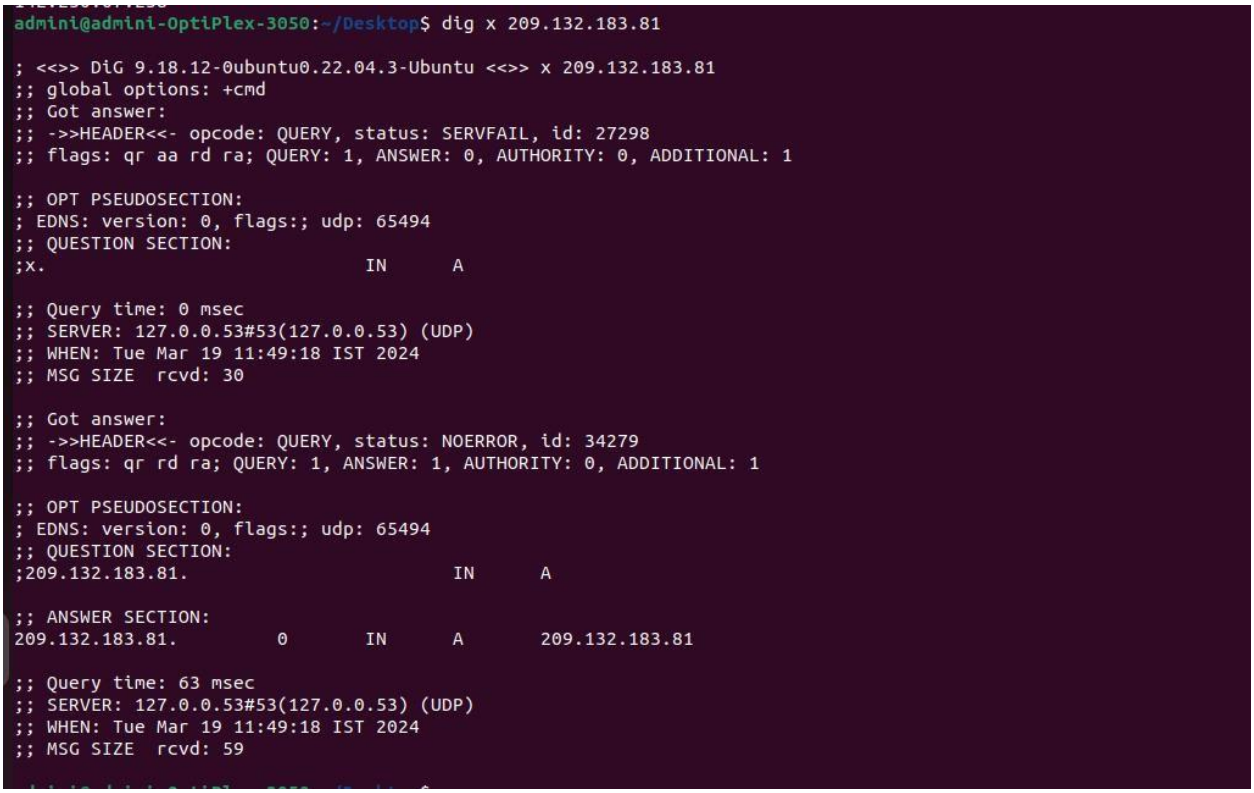
The command uses the -x option to perform a reverse DNS lookup, also known as a PTR (Pointer) query.

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

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal



Traceroute

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

1. **Sending Packets:** When you execute the **traceroute** command followed by a destination (e.g., **google.com**), your computer sends out a series of packets towards that destination.
2. **Incrementing TTL:** Each packet has a Time To Live (TTL) value set initially to 1. When the first packet is sent, it reaches the first router along the path, where the

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

TTL is decremented to 0. The router then discards the packet and sends back an ICMP (Internet Control Message Protocol) Time Exceeded message to your computer.

3. **Determining Hops:** Your computer receives the ICMP Time Exceeded message, indicating the first hop along the route to the destination. It then sends another packet with a TTL of 2, which reaches the second router along the path. This process continues, incrementing the TTL for each subsequent packet, until the destination is reached or the maximum number of hops is reached.
4. **Displaying Results: traceroute** displays the list of routers (or "hops") along the path to the destination. It shows the IP addresses of the routers, along with the round-trip time (RTT) for each hop, indicating how long it took for the packets to reach that router and receive a response.
5. **Identifying Network Issues:** Traceroute is commonly used for diagnosing network connectivity issues. By examining the list of routers and RTT values, network administrators can identify potential bottlenecks, routing problems, or points of failure along the path to the destination.
6. **Different Protocols:** Depending on the operating system and version of **traceroute** being used, it may employ different protocols such as ICMP, UDP, or TCP to send the packets. Each protocol has its advantages and limitations, and may be chosen based on network configurations and security policies.

Command:

traceroute google.com

Explanation:

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

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

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

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

NsLookup

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

1. **Querying DNS Servers:** When you run **nslookup** followed by a domain name or an IP address, the command queries the DNS servers configured on your system to resolve the domain name or perform a reverse DNS lookup for the IP address.
2. **Obtaining DNS Records: nslookup** can retrieve various types of DNS records associated with a domain name, including A (IPv4 address), AAAA (IPv6 address), MX (mail exchange), NS (name server), CNAME (canonical name), and TXT (text) records, among others.
3. **Interactive Mode: nslookup** can also be used in interactive mode, where you can enter multiple queries without having to invoke the command multiple times.

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

In interactive mode, you can specify the type of record you want to query and choose the DNS server to query.

4. **Diagnosing DNS Issues: nslookup** is a valuable tool for diagnosing DNSrelated issues such as DNS resolution failures, misconfigurations, and DNS server connectivity problems. By querying different DNS servers and examining the responses, network administrators can troubleshoot and resolve DNS issues efficiently.

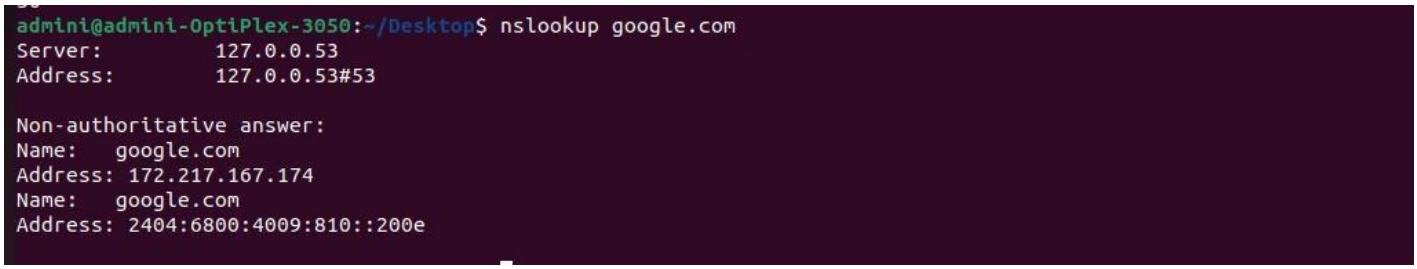
Command:

nslookup google.com

Explanation:

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

Screenshot:



Command:

nslookup -query = mx google.com

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

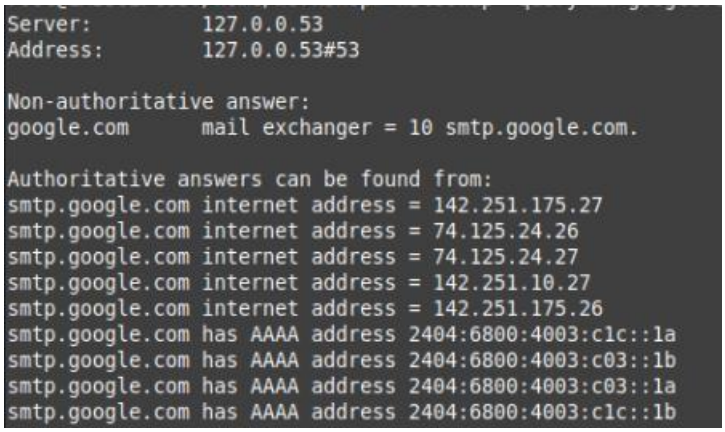
Explanation:

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

In this command:

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

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



Command:

nslookup -type = ns google.com

Explanation:

Roll No. : 2103110

Batch : C23

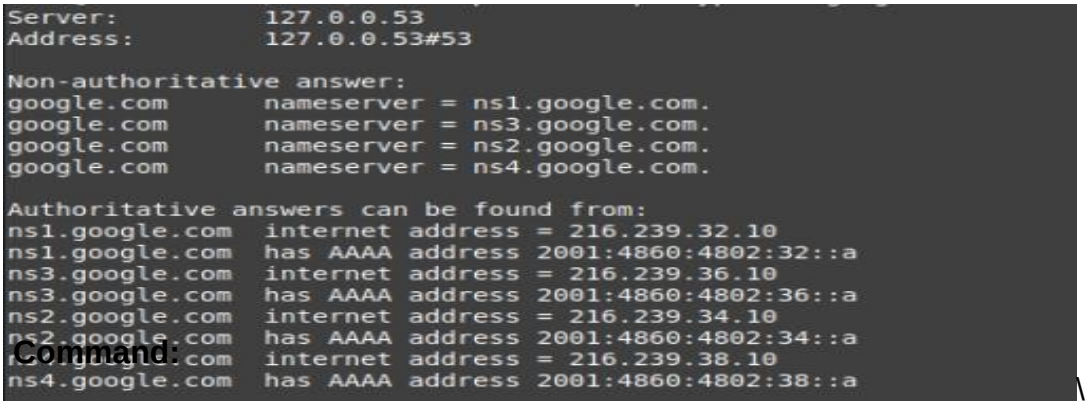
Name : Rishab Mandal

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

In this command:

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

Screenshot:



Command:

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

#nslookup -type = soa google.com

Screenshot :

Explanation:

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

Screenshot:

```
Server:         127.0.0.53
Address:        127.0.0.53#53

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

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

Command:

nslookup -type = any google.com

Explanation:

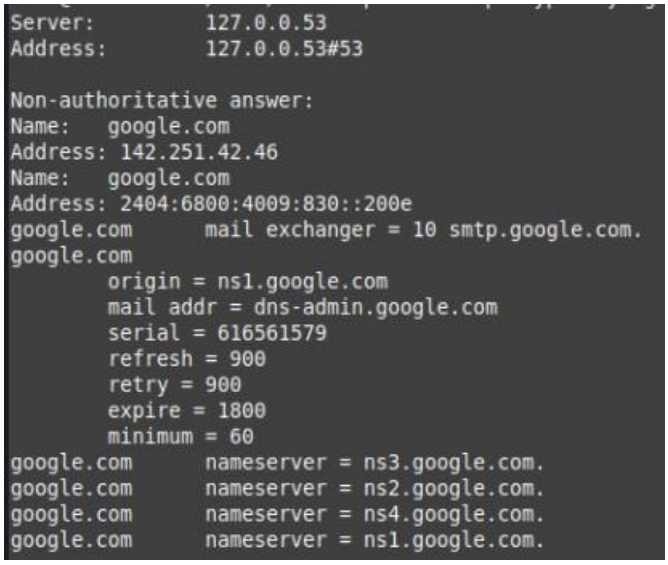
Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

The nslookup command with the -type=any option is used to query all available DNS records (ANY) for the domain google.com. When you execute this command, your system sends a DNS query to the default DNS server configured on your system (or the one specified in your network settings) to retrieve all DNS records associated with the domain google.com. The DNS server then responds with information about all available DNS record types, including A (IPv4 address), AAAA (IPv6 address), MX (mail exchange), NS (name server), SOA (start of authority), and others.

Screenshot:



Extra Commands:

Command:

ipconfig

Explanation:

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

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

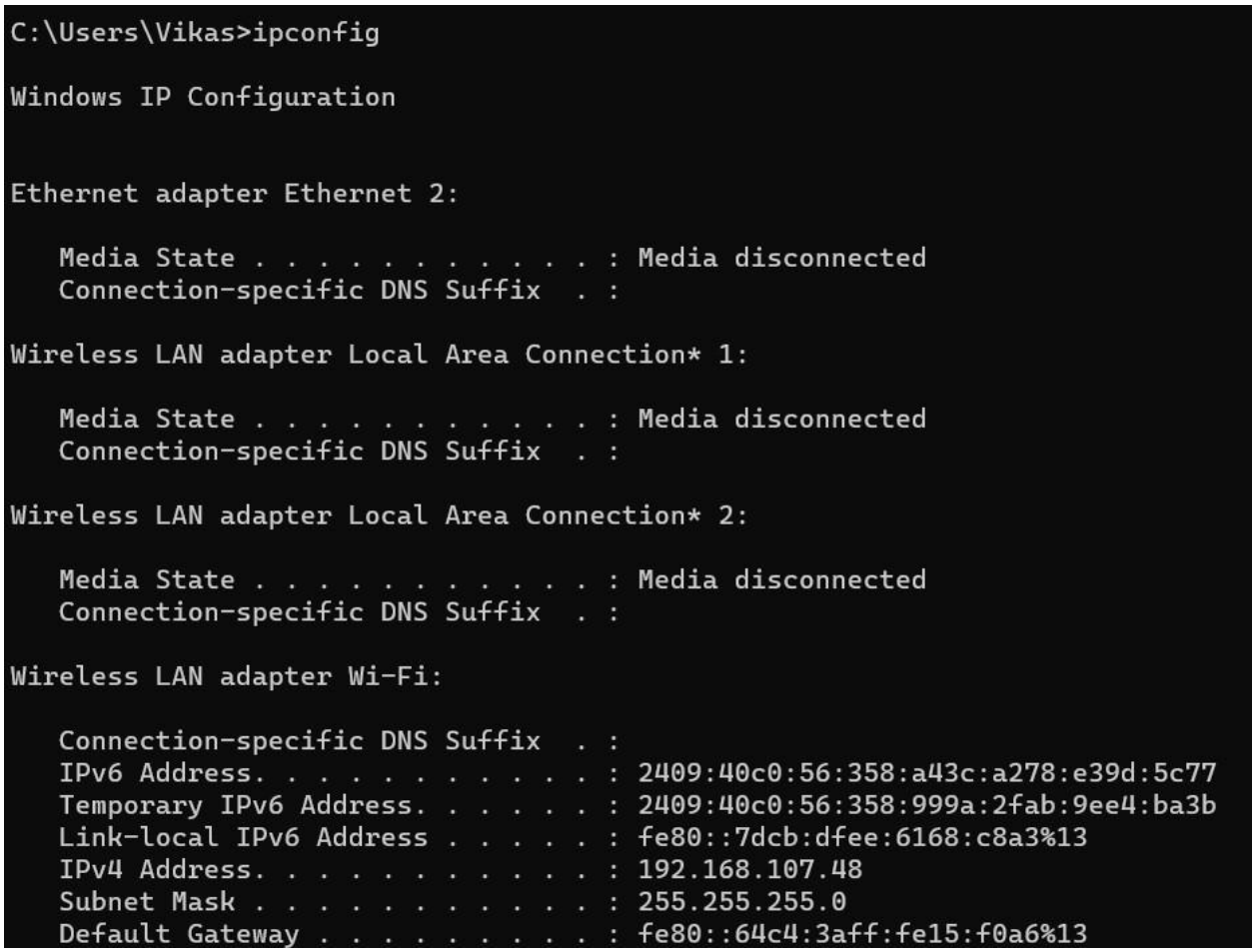
The IPConfig command also provides us with some variation in the primary command that targets specific system settings or data, which are:

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

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

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

Screenshot:



Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

Command:

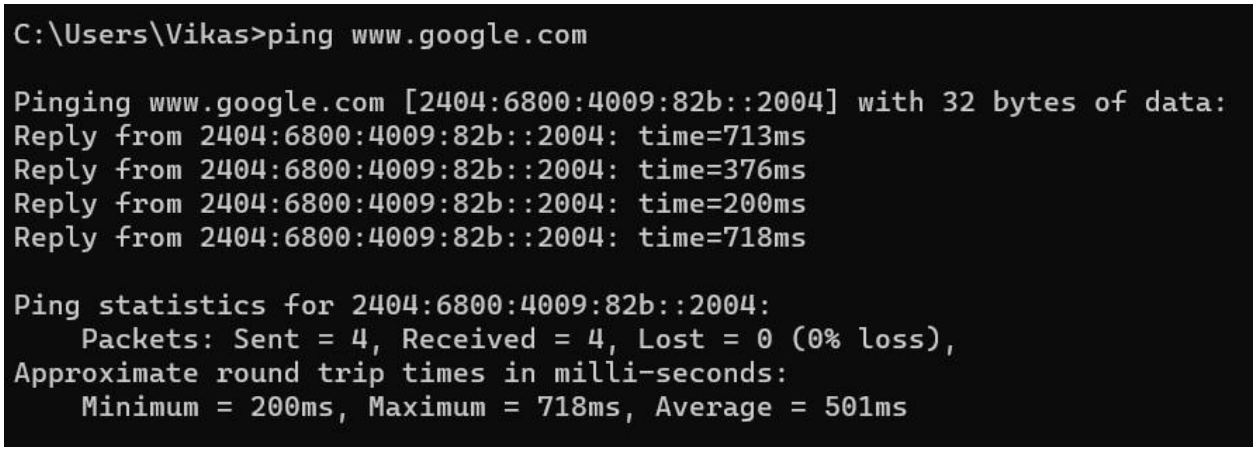
Ping www.google.com

Explanation:

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

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

Screenshot:



Command:

Tracert www.google.com

Roll No. : 2103110

Batch : C23

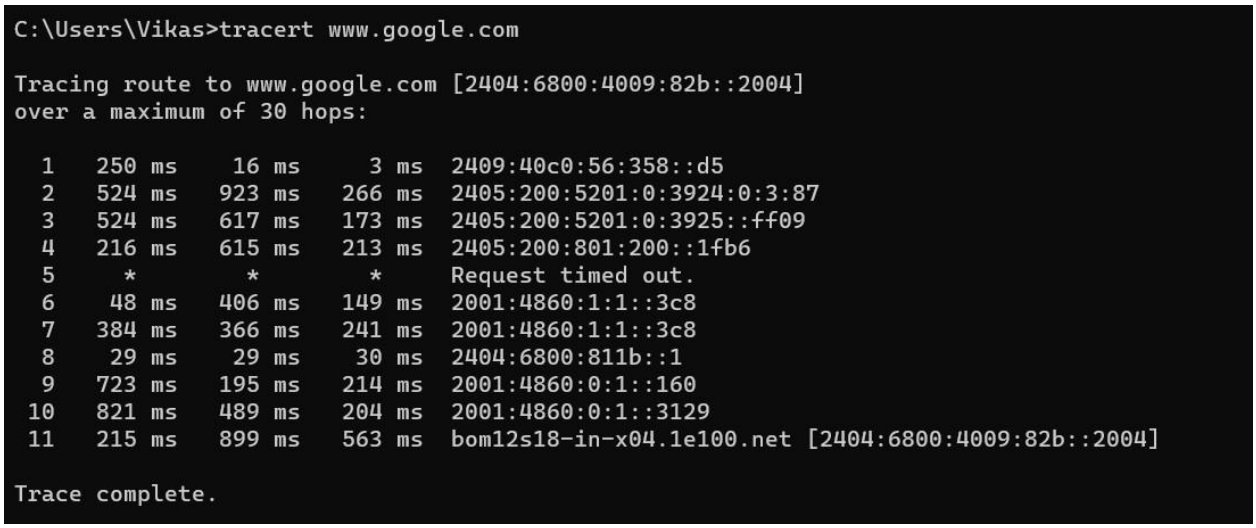
Name : Rishab Mandal

Explanation:

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

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

Screenshot:



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

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

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

Trace complete.
```

Command:

netstat

Explanation:

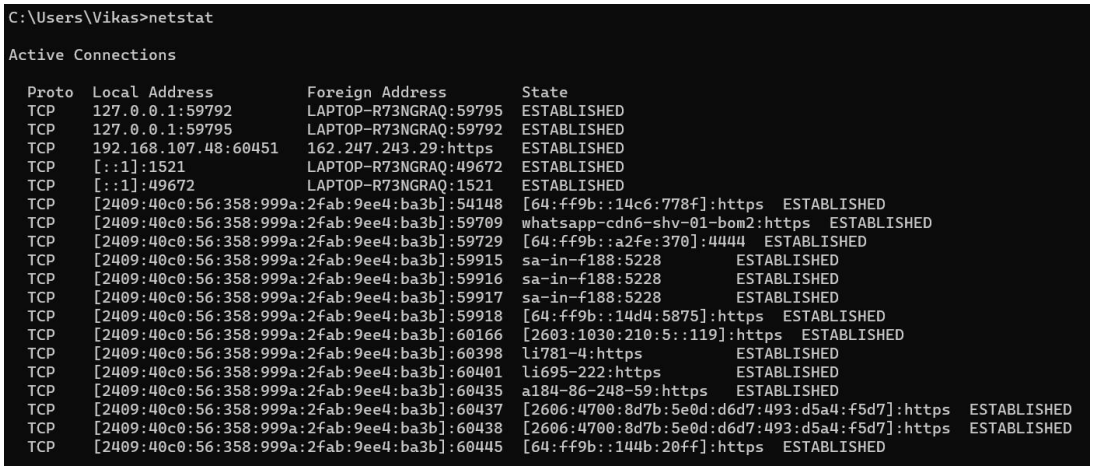
The Netstat command as the name suggests displays an overview of all the network connections in the device. The table shows detail about the connection protocol, address, and the current state of the network.

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

Screenshot:



```
C:\Users\Vikas>netstat

Active Connections

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

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

EXPERIMENT NO. 8

Aim: Examine the use of a packet sniffer tool: Wireshark.

DESCRIPTION/ALGORITHM:

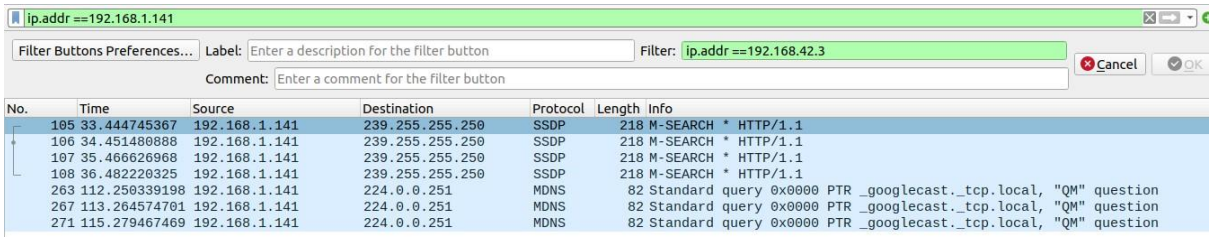
Command:

ip.addr == 192.168.42.3

Explanation:

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

Screenshot:



Command: ip.addr == 10.0.5.119 &&

ip.addr == 91.189.94.25

Explanation:

This filter will capture network traffic where both the source and destination IP addresses match the specified values simultaneously. This filter can be useful for troubleshooting network communication between two specific hosts. It

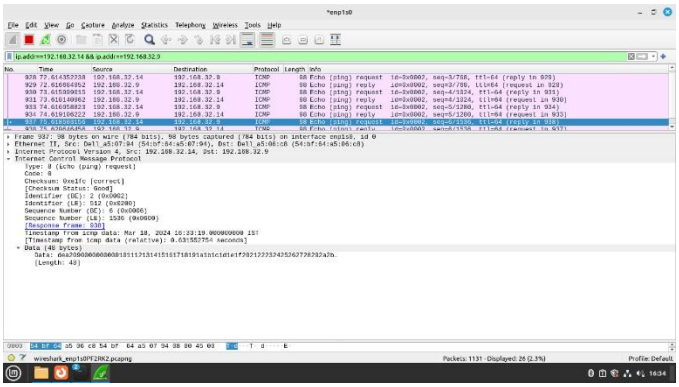
Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

allows you to focus on traffic exchanged exclusively between these two IP addresses, helping to diagnose any issues or monitor communication between them.

Screenshot:



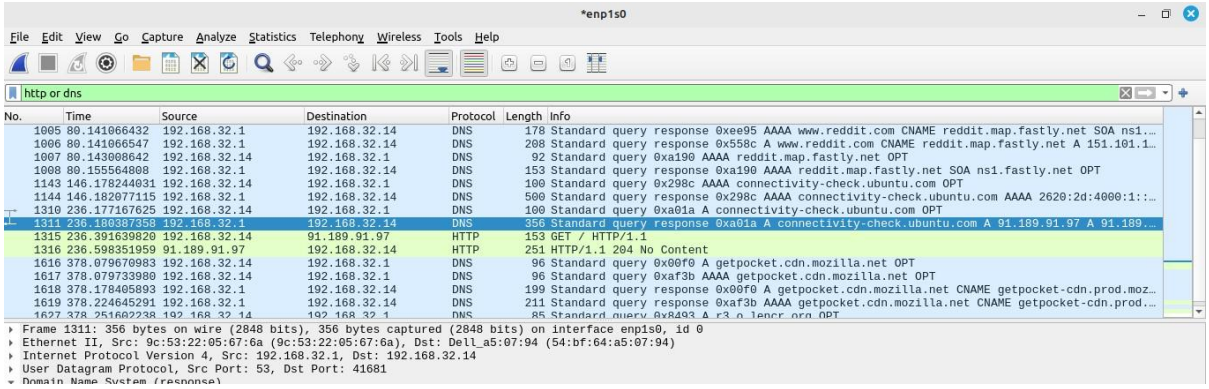
Command:

http or dns

Explanation:

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

Screenshot:



Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

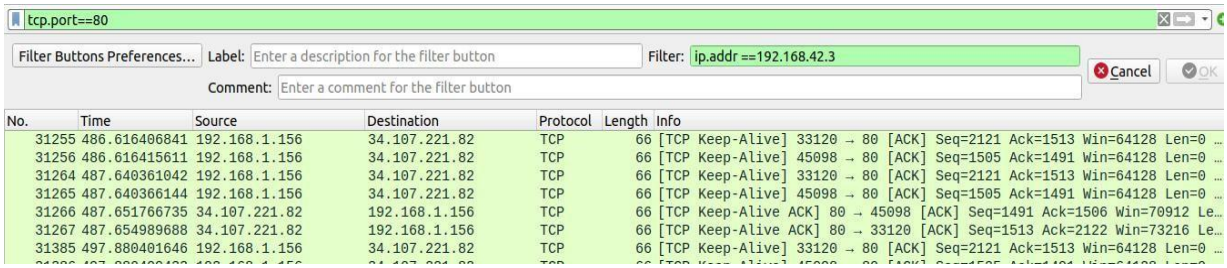
Command:

tcp.port==4000

Explanation:

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

Screenshot:



Command:

tcp.flags.reset== 0

Explanation:

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

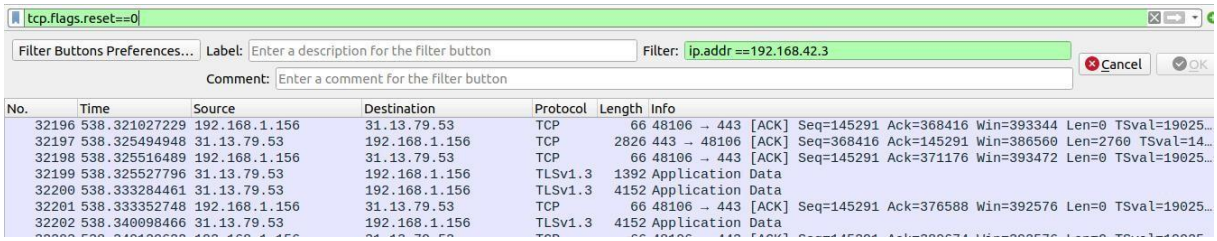
By filtering packets where the Reset (RST) flag is not set (i.e., tcp.flags.reset==0), you can capture TCP traffic that is not associated with connection termination or error conditions. This may include normal data transmission, connection establishment, or connection teardown without using the Reset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

Screenshot:

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal



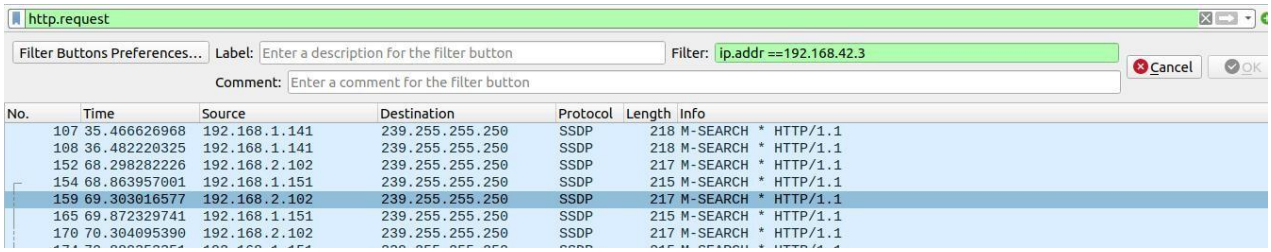
Command:

http.request

Explanation:

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

Screenshot:



Command:

!(arp or icmp or

dns)

Explanation:

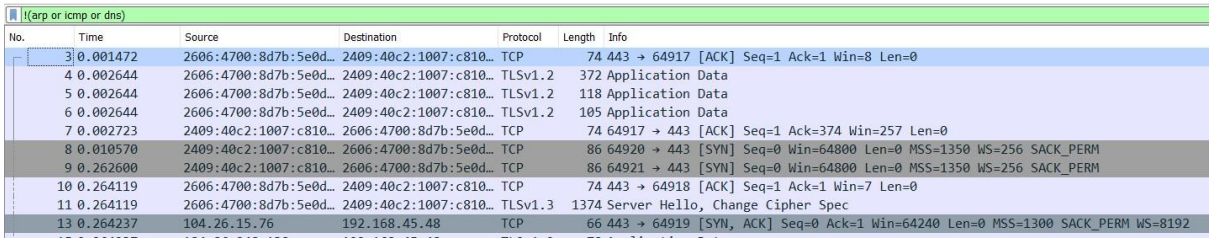
This Wireshark filter is used to capture packets that are not ARP (Address Resolution Protocol), ICMP (Internet Control Message Protocol), or DNS (Domain Name System) traffic. It allows you to analyze actual data communication between devices on the network without being distracted by protocolspecific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

Screenshot:

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal



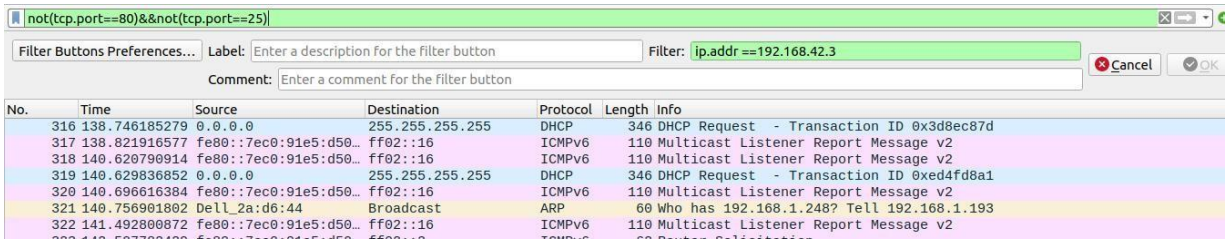
Command: not (tcp.port == 80)

and not (tcp port == 25)

Explanation:

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

Screenshot:



Command:

ftp

Explanation:

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

Screenshot:

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

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

ftp

ftp
ftp-data

	Source	Destination	Protocol	Length	Info
	192.168.45.48	192.168.45.48	FTP	72	Request: PORT 192,168,45,48,200,101
9 20…	192.168.45.48	192.168.45.48	FTP	74	Response: 200 PORT command successful.
11 20…	192.168.45.48	192.168.45.48	FTP	50	Request: NLST
17 20…	192.168.45.48	192.168.45.48	FTP	73	Response: 150 Starting data transfer.
76 20…	192.168.45.48	192.168.45.48	FTP	70	Response: 226 Operation successful
84 20…	192.168.45.48	192.168.45.48	FTP	50	Request: QUIT
88 20…	192.168.45.48	192.168.45.48	FTP	58	Response: 221 Goodbye.
101 20…	192.168.45.48	192.168.45.48	FTP	121	Response: 220-FileZilla Server 1.8.1
103 20…	192.168.45.48	192.168.45.48	FTP	58	Request: OPTS UTF8 ON
107 20…	192.168.45.48	192.168.45.48	FTP	107	Response: 202 UTF8 mode is always enabled. No need to send this command
109 20…	192.168.45.48	192.168.45.48	FTP	55	Request: USER user
115 20…	192.168.45.48	192.168.45.48	FTP	79	Response: 331 Please, specify the password.
117 20…	192.168.45.48	192.168.45.48	FTP	54	Request: PASS 123
123 20…	192.168.45.48	192.168.45.48	FTP	67	Response: 230 Login successful.

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

EXPERIMENT NO. 9

Aim: Setting up a personal Firewall using ip-tables.

Description/Algorithm:

Description

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

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

Targets

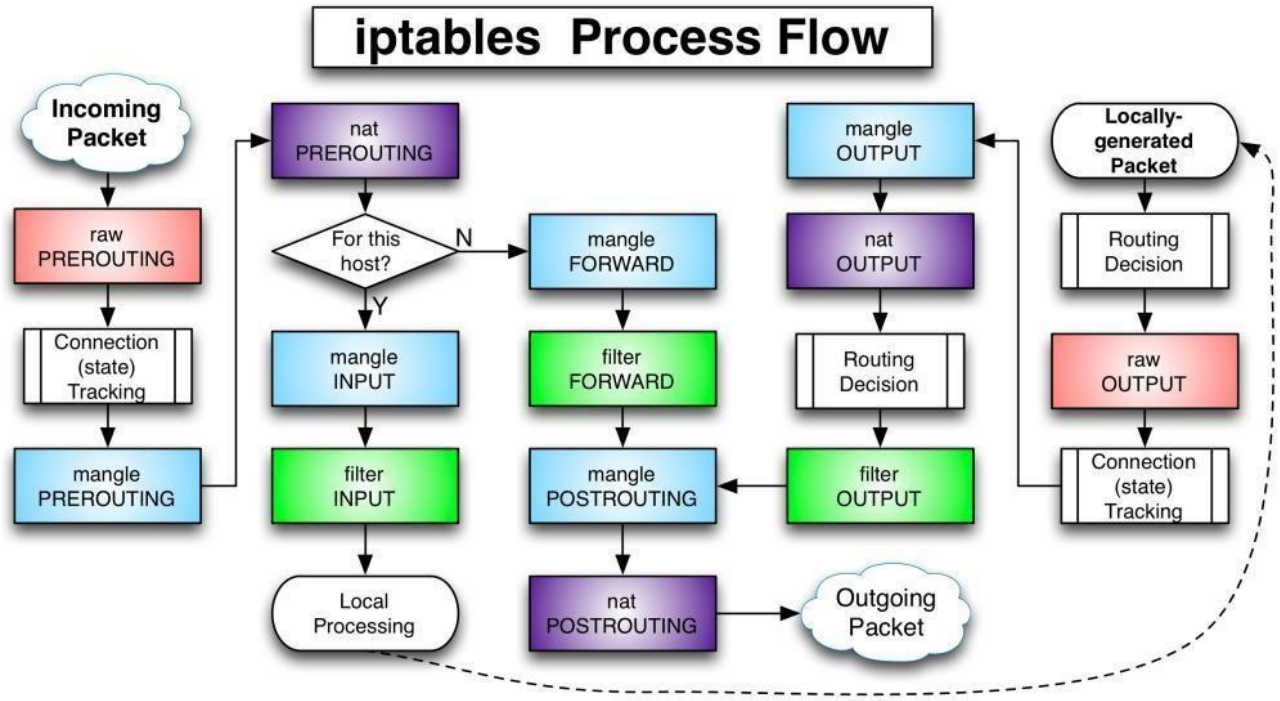
A firewall rule specifies criteria for a packet, and a target. If the packet does not match, the next rule in the chain is the examined; if it does match, then the next rule is specified by the value of the target, which can be the name of a user-defined chain or one of the special values ACCEPT, DROP,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

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

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal



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.

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

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. Manageable table has the following built-in chains.

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

4. Raw Table

Iptables' 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

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

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 behavior for incoming connections. For example, if a user attempts to SSH into your PC/server, ip-tables 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 deciding 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

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

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

1. List iptables rules: `iptables -L` - Displays the current list of iptables rules, initially empty.

2. Block outgoing traffic: `iptables -I OUTPUT -s <your ip> -d <neighbour ip> -p <protocol> -j <action>` - Blocks outgoing traffic to a specific destination for a specific protocol from a machine.

3. Allow outgoing traffic: `iptables -I OUTPUT -s <your ip> -d <neighbour ip> -p <protocol> -j ACCEPT` - Allows outgoing traffic to a specific destination for a specific protocol from a machine.

4. Block outgoing traffic temporarily: `iptables -I OUTPUT -s <your ip> -d <neighbour ip> -p <protocol> -j REJECT` - Temporarily blocks outgoing traffic to a specific destination for a specific protocol from a machine.

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

5. Block incoming traffic: `iptables -I INPUT -s <neighbour ip> -d <firewall ip> -p <protocol> -j DROP` - Blocks incoming traffic from a particular destination for a specific protocol to the machine.

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

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

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

EXTRA:

9. Block all incoming and outgoing traffic by default:

- `iptables -P INPUT DROP`

- `iptables -P OUTPUT DROP`

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

10. Allow established connections:

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

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

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

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

Steps:-

1. Get root access: $ sudo su root

2. # apt-get install iptables

Commands:-

1. To see the list of iptables rules

iptables -L

. Initially it is empty

```
root@kali:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

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

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

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

#ping 192.168.208.6

Open another terminal and run the iptables command

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

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

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

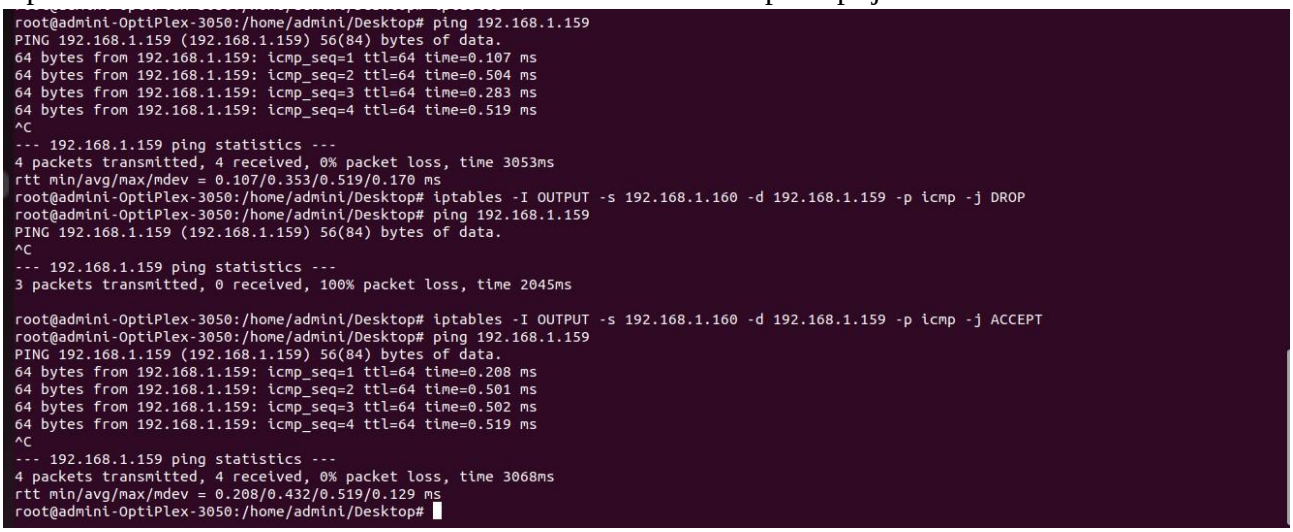
3. To block outgoing traffic to a particular destination for a specific protocol from a machine for sometime

Roll No. : 2103110

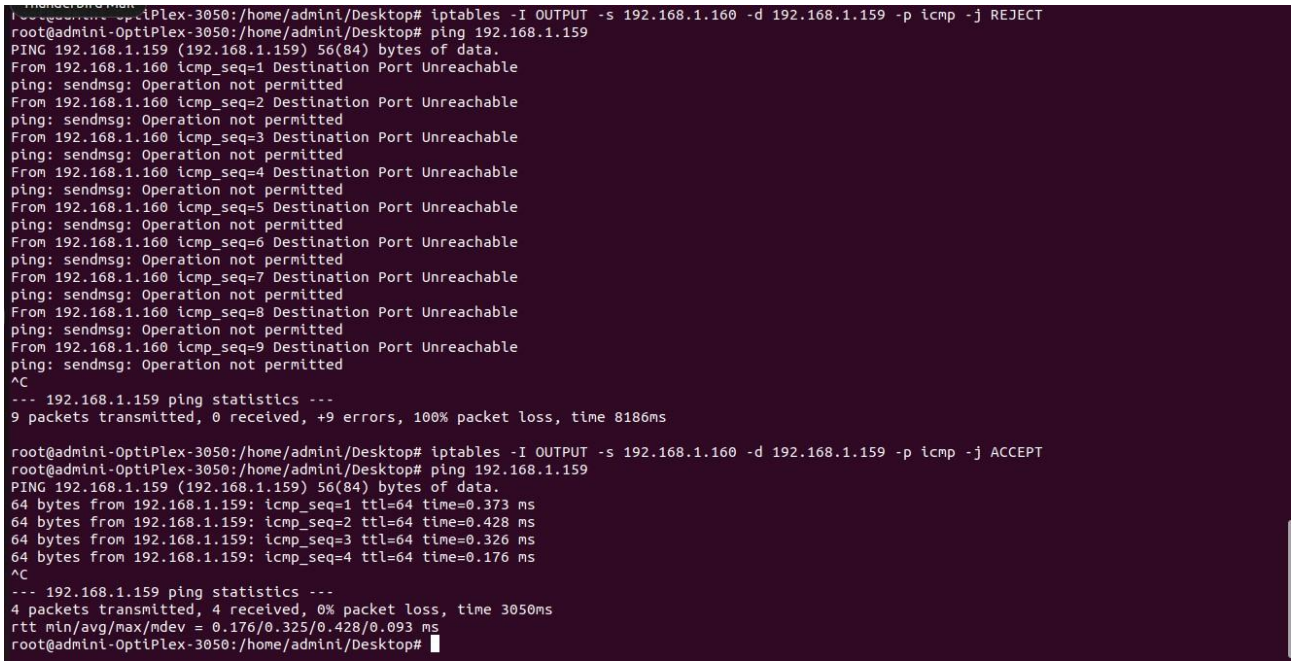
Batch : C23

Name : Rishab Mandal

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



Allow the traffic again by using ACCEPT instead of REJECT



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

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

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

Open one terminal and Ping the neighbour. Let the ping run.
#ping 192.168.208.6

Open another terminal and run the iptables command
iptables -I INPUT -s 192.168.208.6 -d 192.168.208.18 -p icmp -j DROP

5. To allow incoming traffic from particular destination for a specific protocol to machine
Syntax: iptables -I INPUT -s <neighbour ip> -d <firewall ip> -p <protocol> -j <action>

Open another terminal and run the iptables command
iptables -I INPUT -s 192.168.208.6 -d 192.168.208.18 -p icmp -j ACCEPT

Check the ping status on the other terminal

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

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

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

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

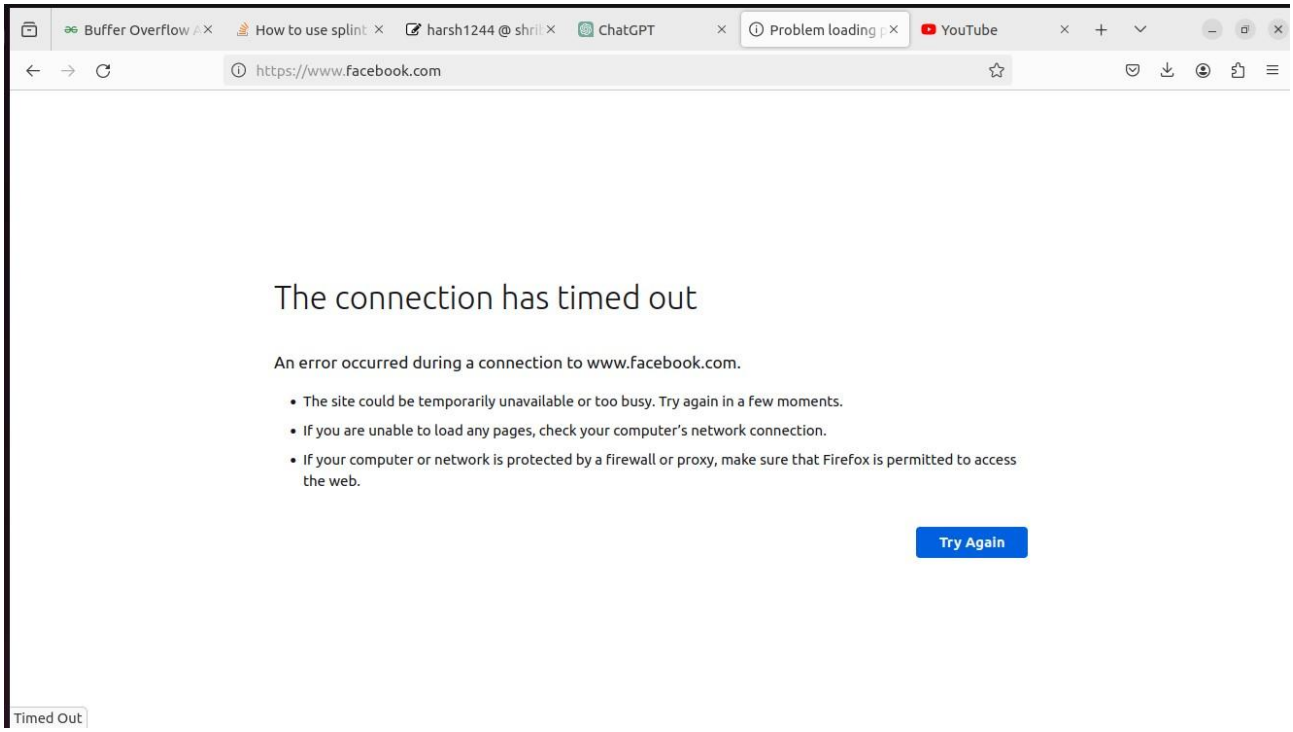
6. To clear the rules in iptables

iptables -F

7. To block specific URL from machine

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

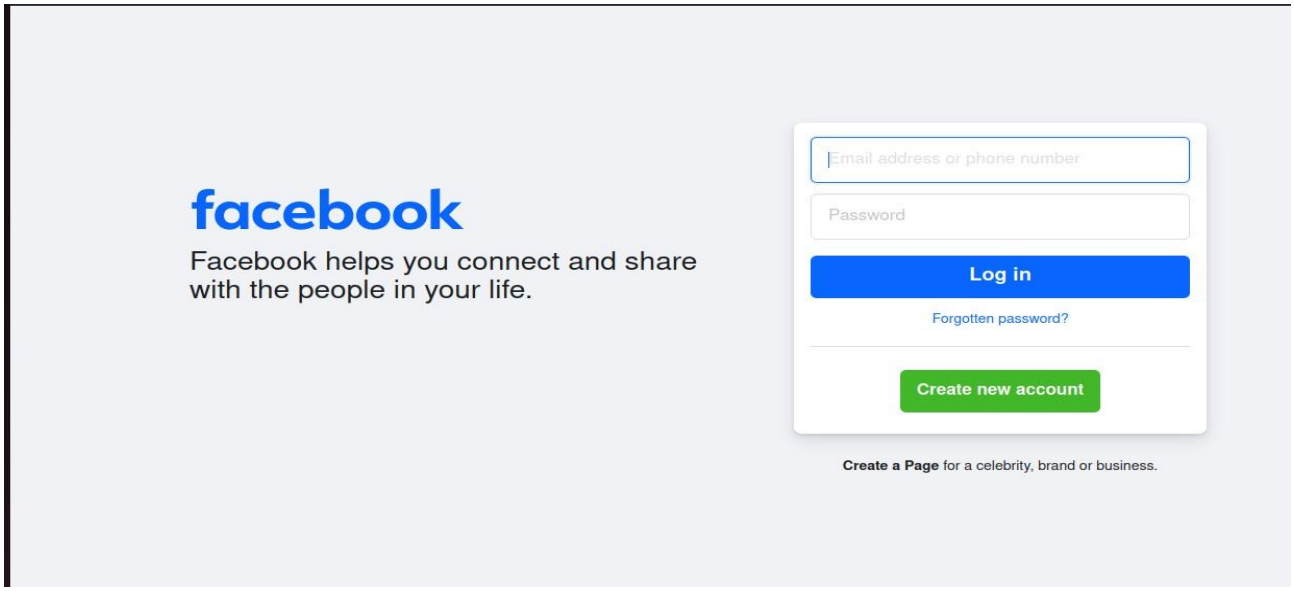
```
root@admini-OptiPlex-3050:/home/admini/Desktop# iptables -F
root@admini-OptiPlex-3050:/home/admini/Desktop# iptables -t filter -I OUTPUT -m string --string facebook.com -j REJECT --algo kmp
root@admini-OptiPlex-3050:/home/admini/Desktop# iptables -t filter -I OUTPUT -m string --string facebook.com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



Roll No. : 2103110

Batch : C23

Name : Rishab Mandal



It will block facebook.com by performing string matching. The algorithm used for string matching is KMP. If we change target *from REJECT to ACCEPT*, the site can be visited again.

EXTRA :

To block all incoming and outgoing traffic by default:

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

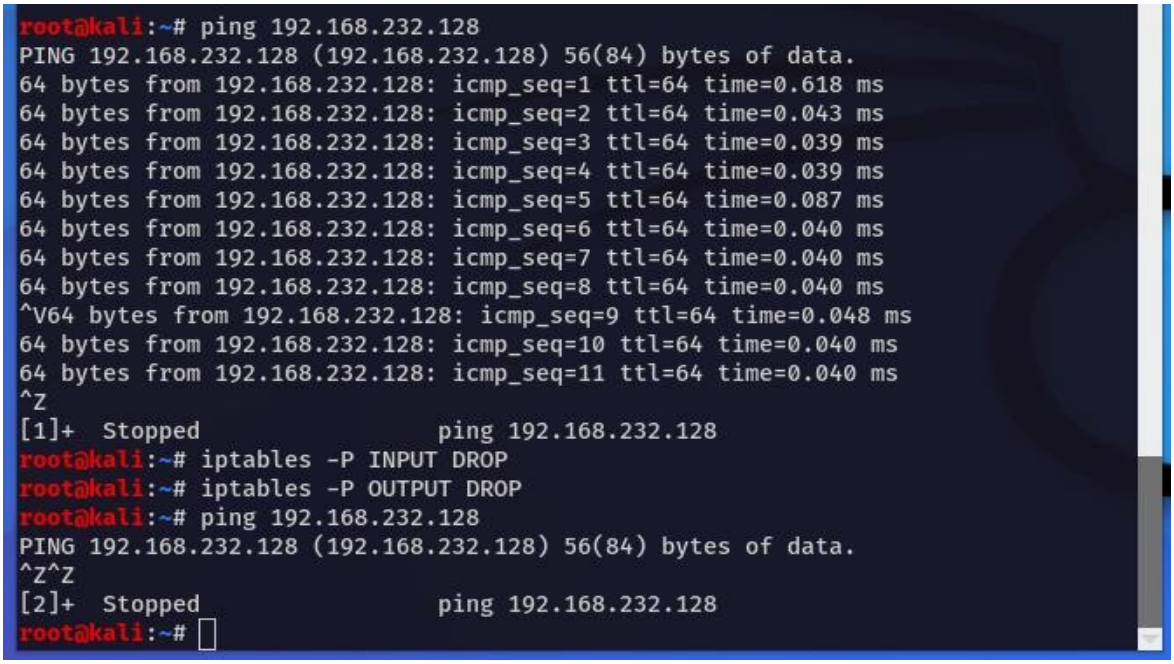
These commands set the default policy for incoming and outgoing traffic to DROP, effectively blocking all traffic unless explicitly allowed by rules.

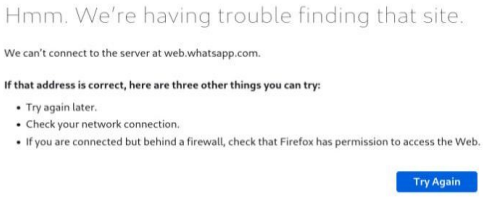
To

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal





allow

Roll No. : 2103110

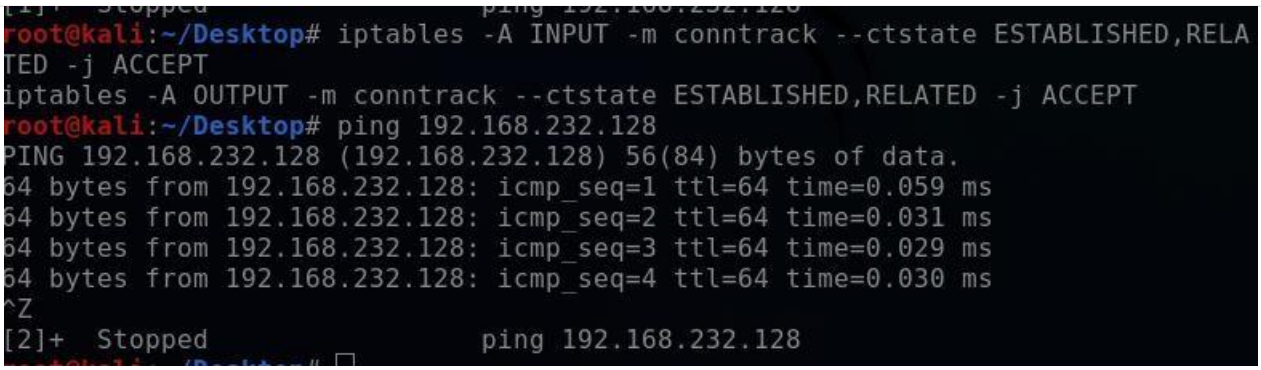
Batch : C23

Name : Rishab Mandal

established connections:

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

ds allow incoming and outgoing traffic that is part of an established or related
uring that established connections are not blocked.



Observations:

1. In case of OUTPUT chain, for DROP and REJECT chain, at source machine we get two different messages.

   For DROP – ‘Operation Not Permitted’. Here No acknowledgement is provided.

   For REJECT – ‘Destination Port Unreachable’. Here acknowledgement is given.
2. In case of INPUT chain for DROP and REJECT chain at source machine we get two different responses as follows:

   For DROP – No message. Here No acknowledgement is provided.

   For REJECT – ‘Destination Port Unreachable’. Here acknowledgement is given.

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

EXPERIMENT NO. 10

Aim: Simulate Buffer overflow attack using Splint.

Description/Algorithm:

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

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

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

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

Steps :

1. Installation

$ sudo apt-get install splint

2. Checking Vulnerability

$ splint program1.c

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

```
#include <stdio.h>

#include <string.h> int

main(void)

{

 char buff[15];  int pass = 0;

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

gets(buff);  if(strcmp(buff,

"thegeekstuff"))
```

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

```
{
 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%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%p\n%p\n");
}
```

Program 3

```
#include <stdio.h> #include <string.h> char
password[] = "password"; int get_password() {  int
auth_ok = 0;  char buff[16];  printf("Enter
```

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

```
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;
 }
```

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

Result/Screenshots :

Program 1

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

 Enter the password :
lab403

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

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

Finished checking --- 4 code warnings
```

```
admini@admini-OptiPlex-3050:~/Desktop$ ./test.out

 Enter the password :
thegeekstuff

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

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

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

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

Program 2 :

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

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

   2 | main()

     | ^~~~ program2.c: In

function ‘main’:

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

   6 | buff[5]='abcdefghijklmnophsgkgfks';

|       ^~~~~~~~~~~~~~~~~~~~~~~~~~

program2.c:6:9: warning: overflow in conversion from ‘int’

to ‘char’ changes value from ‘1734765427’ to ‘115’

[Woverflow] root@kali:~# ./p2.out My stack looks  like:

0x7fff9e6a7098

0x7fff9e6a70a8

0x5595409e6dd8

(nil)

0x7fb5769cab10

(nil)

0x7fb5769f8ab0

0x1

0x7fb5767f36ca

0x7fff9e6a7080

s

My new stack looks like:

0x559540a302a0

(nil)

(nil)

0x63

(nil)

(nil)

0x7fb5769f8ab0

0x73

0x7fb5767f36ca

0x7fff9e6a7080 root@kali:~#

splint program2.c

Splint 3.1.2 --- 21 Feb 2021


program2.c: (in function main)

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

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

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

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

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

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):

"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):

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

"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

Program 3 :

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

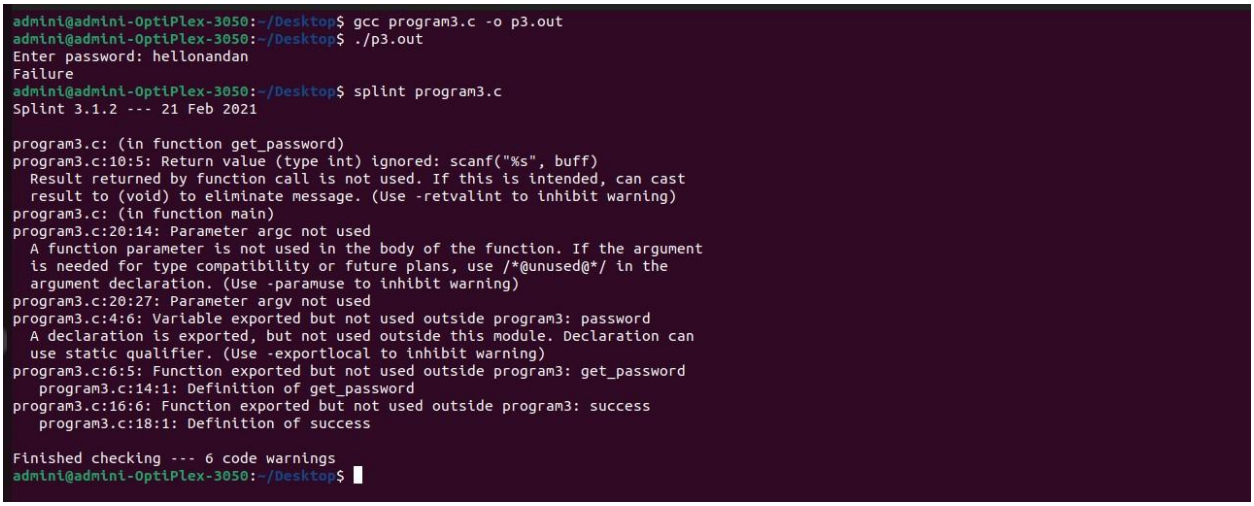
returning a meaningful result to the caller. (Use -noret to inhibit warning)

Finished checking --- 21 code warnings

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal



Extra Splint Programs and Outputs :

Buffer Overflow in Local Variables:

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

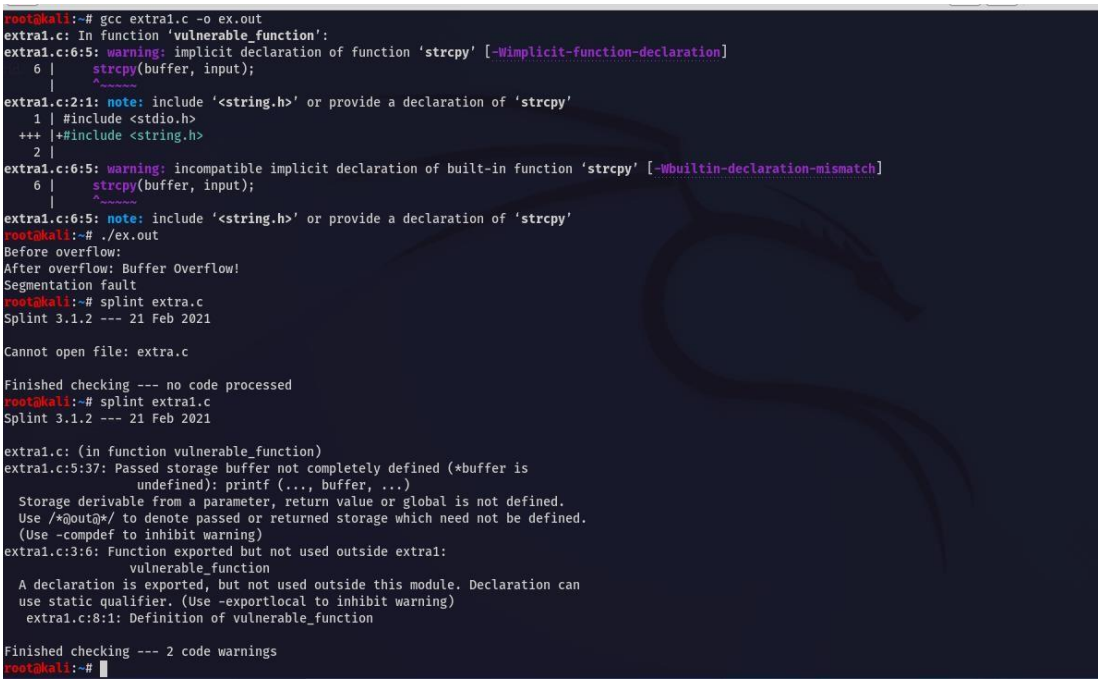
```
#include <stdio.h>


void vulnerable_function(char *input) {

char buffer[10];     printf("Before

overflow: %s\n", buffer);     strcpy(buffer,

input);     printf("After overflow: %s\n",

buffer);

}
```

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

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



Integer Overflow leading to Buffer Overflow:

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

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

Roll No. : 2103110

Batch : C23

Name : Rishab Mandal

```
#include <string.h>

void vulnerable_function(int size) {
char buffer[size];    strcpy(buffer,
"Buffer Overflow!");
printf("%s\n", buffer);
}

int main() {    int size = 10;
vulnerable_function(size);
return 0;

}
```