C23-109

Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

CERTIFICATE

Certify that Mr./Miss Aznav Malvia

of Computer Department, Semester VI with Roll No. 2103109 has completed a course of the necessary experiments in the subject CSS under my supervision in the **Thadomal Shahani Engineering College** Laboratory in the year 2023 - 2024

Teacher In-Charge

Head of the Department

Date 10/4/24

Principal

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

CONTENTS

10/4/24

Name: Arnav Malvia

Roll No.: 2103109

Batch: C23

EXPERIMENT 1

Aim: Implementation of Extended Eucledian Algorithm

Theory:

Extended Euclidean Algorithm is an extension of the Euclidean Algorithm that computes the greatest common divisor (GCD) of integers a and b. GCD is the largest integer that divides both a and b without any remainder.

In addition to computing GCD, Extended Euclidean Algorithm also finds integers s and t such that as+bt=gcd(a,b).

Bézout's Identity guarantees the existence of s and t.

Extended Euclidean Algorithm finds s and t by using back substitutions to recursively rewrite the division algorithm equation until we end up with the equation that is a linear combination of our initial numbers.

Consider Example:

a = 161 and b = 28. Calculate gcd(a, b) and s and t.

a	b	q	r	s1	s2	s3	t1	t2	t3
161	28	5	21	1	0	1	0	1	-5
28	21	1	7	0	1	-1	1	-5	6
21	7	3	0	1	-1	4	-5	6	-23

Results

Now, remember that we're calculating the value of s, t and gcd(a, b) in this equation:
$s \times a + t \times b = gcd(a, b)$
Since we have reached r=0, we are done and we can now find the answers in the last row of the table:

- gcd(a, b) = [b on the last row]
- s = [s2 on the last row]
- t = [t2 on the last row]

So in this case, we have found the following:

- gcd(161, 28) = 7
- s = -1
- t = 6

Code:

```
import java.util.*;
class ExtendedEucledianDistance{
    public static void main(String args[]){
        int q,r,s,t;
        int s1=1;
        int s2=0;
        int t1=0;
        int t2=1;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter number 1\n");
        int r1=sc.nextInt();
        System.out.println("Enter number 2\n");
        int r2=sc.nextInt();
        r=r1%r2;
        while(r!=0){
            q=r1/r2;
            r=r1%r2;
            s=s1-(q*s2);
            t=t1-(q*t2);
            System.out.println("q"+" "+"r1"+" "+"r2"+" "+"r"+" "+"t1"+" "+"t2"+"
"+"t"+"\n");
            System.out.println(q+" "+r1+" "+r2+" "+r+" "+t1+" "+t2+" "+t+"\n");
            r1=r2;
            r2=r;
```

```
            t1=t2;
            t2=t;
        }
        if(r1==1){
            System.out.println("\nInverse exists at");
            if(t1<0){
                System.out.println(t1+t2);
            }
            else{
                System.out.println(t1);
            }
        }
        else{
            System.out.println("Inverse does not exist");
        }
    }
}
```

Output:

```
Enter number 1

161
Enter number 2

28
q r1 r2 r t1 t2 t

5 161 28 21 0 1 -5

q r1 r2 r t1 t2 t

1 28 21 7 1 -5 6

q r1 r2 r t1 t2 t

3 21 7 0 -5 6 -23

Inverse does not exist
```

Conclusion:

Hence we have understood the implementation of Extended Eucledian Algorithm in Java.

Name: Arnav Malvia

Roll No.: 2103109

Batch: C23

EXPERIMENT 2

Aim: Implementation of Caesar Cipher

Theory:

The Caesar cipher is a simple encryption technique that was used by Julius Caesar to send secret messages to his allies. It works by shifting the letters in the plaintext message by a certain number of positions, known as the “shift” or “key”.

The Caesar Cipher technique is one of the earliest and simplest methods of encryption technique. It’s simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter with a fixed number of positions down the alphabet. For example with a shift of 1, A would be replaced by B, B would become C, and so on. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.

Thus to cipher a given text we need an integer value, known as a shift which indicates the number of positions each letter of the text has been moved down. The encryption can be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, A = 0, B = 1,…, Z = 25. Encryption of a letter by a shift n can be described mathematically as.

For example, if the shift is 3, then the letter A would be replaced by the letter D, B would become E, C would become F, and so on. The alphabet is wrapped around so that after Z, it starts back at A.

Here is an example of how to use the Caesar cipher to encrypt the message “HELLO” with a shift of 3:

Write down the plaintext message: HELLO

Choose a shift value. In this case, we will use a shift of 3.

Replace each letter in the plaintext message with the letter that is three positions to the right in the alphabet.

H becomes K (shift 3 from H)

E becomes H (shift 3 from E)

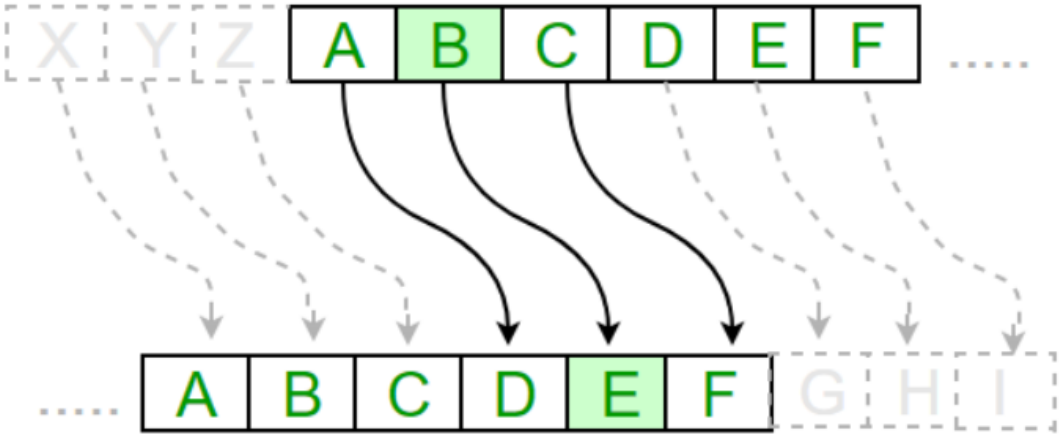
L becomes O (shift 3 from L)

L becomes O (shift 3 from L)

O becomes R (shift 3 from O)

The encrypted message is now “KHOOR”.

To decrypt the message, you simply need to shift each letter back by the same number of positions. In this case, you would shift each letter in “KHOOR” back by 3 positions to get the original message, “HELLO”.



Code:

```
import java.util.Scanner;

public class CaesarCipherExample

{

   // ALPHABET string denotes alphabet from a-z

   public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyz";


   // create encryptData() method for encrypting user input string with given shift key

   public static String encryptData(String inputStr, int shiftKey)

   {

      // convert inputStr into lower case

      inputStr = inputStr.toLowerCase();
```

```
        // encryptStr to store encrypted data
        String encryptStr = "";

        // use for loop for traversing each character of the input string
        for (int i = 0; i < inputStr.length(); i++)
        {
            // get position of each character of inputStr in ALPHABET
            int pos = ALPHABET.indexOf(inputStr.charAt(i));

            // get encrypted char for each char of inputStr
            int encryptPos = (shiftKey + pos) % 26;
            char encryptChar = ALPHABET.charAt(encryptPos);

            // add encrypted char to encrypted string
            encryptStr += encryptChar;
        }

        // return encrypted string
        return encryptStr;
    }

    // create decryptData() method for decrypting user input string with given
shift key
    public static String decryptData(String inputStr, int shiftKey)
    {
        // convert inputStr into lower case
        inputStr = inputStr.toLowerCase();
```

```
    // decryptStr to store decrypted data
    String decryptStr = "";

    // use for loop for traversing each character of the input string
    for (int i = 0; i < inputStr.length(); i++)
    {

        // get position of each character of inputStr in ALPHABET
        int pos = ALPHABET.indexOf(inputStr.charAt(i));

        // get decrypted char for each char of inputStr
        int decryptPos = (pos - shiftKey) % 26;

        // if decryptPos is negative
        if (decryptPos < 0){
            decryptPos = ALPHABET.length() + decryptPos;
        }
        char decryptChar = ALPHABET.charAt(decryptPos);

        // add decrypted char to decrypted string
        decryptStr += decryptChar;
    }
    // return decrypted string
    return decryptStr;
}

// main() method start
```

```
    public static void main(String[] args)
    {
        // create an instance of Scanner class
        Scanner sc = new Scanner(System.in);

        // take input from the user
        System.out.println("Enter a string for encryption using Caesar Cipher: ");
        String inputStr = sc.nextLine();

        System.out.println("Enter the value by which each character in the
plaintext message gets shifted: ");
        int shiftKey = Integer.valueOf(sc.nextLine());

        System.out.println("Encrypted Data ===> "+encryptData(inputStr,
shiftKey));
        System.out.println("Decrypted Data ===>
"+decryptData(encryptData(inputStr, shiftKey), shiftKey));

        // close Scanner class object
        sc.close();
    }
}
```

Output:

```
Enter a string for encryption using Caesar Cipher:
sanjana
Enter the value by which each character in the plaintext message gets shifted:
3
Encrypted Data ===> vdqmdqd
Decrypted Data ===> sanjana
```

Conclusion: Hence we have understood the implementation of Caeser Cipher.

Name: Arnav Malvia

Roll No.: 2103109

Batch: C23

EXPERIMENT 3

Aim: Implementation of Playfair Cipher

Theory:

Playfair cipher is an encryption algorithm to encrypt or encode a message. It is the same as a traditional cipher. The only difference is that it encrypts a digraph (a pair of two letters) instead of a single letter.

It initially creates a key-table of 5*5 matrix. The matrix contains alphabets that act as the key for encryption of the plaintext. Note that any alphabet should not be repeated. Another point to note that there are 26 alphabets and we have only 25 blocks to put a letter inside it. Therefore, one letter is excess so, a letter will be omitted (usually J) from the matrix. Nevertheless, the plaintext contains J, then J is replaced by I. It means treat I and J as the same letter, accordingly.

Since Playfair cipher encrypts the message digraph by digraph. Therefore, the Playfair cipher is an example of a digraph substitution cipher.

Rules of Playfair Cipher:

1. First, split the plaintext into digraphs (pair of two letters). If the plaintext has the odd number of letters, append the letter Z at the end of the plaintext. It makes the plaintext of even For example, the plaintext MANGO has five letters. So, it is not possible to make a digraph. Since, we will append a letter Z at the end of the plaintext, i.e. MANGOZ.

2. After that, break the plaintext into digraphs (pair of two letters). If any letter appears twice (side by side), put X at the place of the second occurrence. Suppose, the plaintext is COMMUNICATE then its digraph becomes CO MX MU NI CA TE. Similarly, the digraph for the plaintext JAZZ will be JA ZX ZX, and for plaintext GREET, the digraph will be GR EX ET.

3. To determine the cipher (encryption) text, first, build a 5*5 key-matrix or key-table and filled it with the letters of alphabets, as directed below:

Fill the first row (left to right) with the letters of the given keyword (ATHENS). If the keyword has duplicate letters (if any) avoid them. It means a letter will be

considered only once. After that, fill the remaining letters in alphabetical order. Let's create a 5*5 key-matrix for the keyword ATHENS.

Note that in the above matrix any letter is not repeated. The letters in the first row (in green color) represent the keyword and the remaining letters sets in alphabetical order.

4. There may be the following three conditions:

i) If a pair of letters (digraph) appears in the same row

In this case, replace each letter of the digraph with the letters immediately to their right. If there is no letter to the right, consider the first letter of the same row as the right letter. Suppose, **Z** is a letter whose right letter is required, in such case, **T** will be right to Z.

ii) If a pair of letters (digraph) appears in the same column

In this case, replace each letter of the digraph with the letters immediately below them. If there is no letter below, wrap around to the top of the same column. Suppose, **W** is a letter whose below letter is required, in such case, **V** will be below W.

iii) If a pair of letters (digraph) appears in a different row and different column

In this case, select a 3*3 matrix from a 5*5 matrix such that pair of letters appear in the 3*3 matrix. Since they occupy two opposite corners of a square within the matrix. The other corner will be a cipher for the given digraph.

In other words, we can also say that intersection of H and Y will be the cipher for the first letter and

Suppose, a digraph is **HY** and we have to find a cipher for it. We observe that both H and Y are placed in different rows and different columns. In such cases, we have to select a 3*3 matrix in such a way that both H and Y appear in the 3*3 matrix (highlighted with yellow color). Now, we will consider only the selected matrix to find the cipher.

Now to find the cipher for HY, we will consider the diagonal **opposite** to HY, i.e. **LU.** Therefore, the cipher for **H** will be **L,** and the cipher for **Y** will be **U**.

Code:

```
import java.awt.Point;
import java.util.Scanner;
public class PlayfairCipher
{
//length of digraph array
private int length = 0;
//creates a matrix for Playfair cipher
private String [][] table;
//main() method to test Playfair method
public static void main(String args[])
{
PlayfairCipher pf = new PlayfairCipher();
}
//main run of the program, Playfair method
//constructor of the class
private PlayfairCipher()
{
//prompts user for the keyword to use for encoding & creates tables
System.out.print("Enter the key for playfair cipher: ");
Scanner sc = new Scanner(System.in);
String key = parseString(sc);
while(key.equals(""))
key = parseString(sc);
table = this.cipherTable(key);
//prompts user for message to be encoded
System.out.print("Enter the plaintext to be encipher: ");
```

```
//System.out.println("using the previously given keyword");
String input = parseString(sc);
while(input.equals(""))
input = parseString(sc);
//encodes and then decodes the encoded message
String output = cipher(input);
String decodedOutput = decode(output);
//output the results to user
this.keyTable(table);
this.printResults(output,decodedOutput);
}
//parses an input string to remove numbers, punctuation,
//replaces any J's with I's and makes string all caps
private String parseString(Scanner sc)
{
String parse = sc.nextLine();
//converts all the letters in upper case
parse = parse.toUpperCase();
//the string to be substituted by space for each match (A to Z)
parse = parse.replaceAll("[^A-Z]", "");
//replace the letter J by I
parse = parse.replace("J", "I");
return parse;
}
//creates the cipher table based on some input string (already parsed)
private String[][] cipherTable(String key)
{
```

```
//creates a matrix of 5*5
String[][] playfairTable = new String[5][5];
String keyString = key + "ABCDEFGHIKLMNOPQRSTUVWXYZ";
//fill string array with empty string
for(int i = 0; i < 5; i++)
for(int j = 0; j < 5; j++)
playfairTable[i][j] = "";
for(int k = 0; k < keyString.length(); k++)
{
boolean repeat = false;
boolean used = false;
for(int i = 0; i < 5; i++)
{
for(int j = 0; j < 5; j++)
{
if(playfairTable[i][j].equals("" + keyString.charAt(k)))
{
repeat = true;
}
else if(playfairTable[i][j].equals("") && !repeat && !used)
{
playfairTable[i][j] = "" + keyString.charAt(k);
used = true;
}
}
}
}
```

```
return playfairTable;
}
//cipher: takes input (all upper-case), encodes it, and returns the output
private String cipher(String in)
{
length = (int) in.length() / 2 + in.length() % 2;
//insert x between double-letter digraphs & redefines "length"

for(int i = 0; i < (length - 1); i++)
{
if(in.charAt(2 * i) == in.charAt(2 * i + 1))
{
in = new StringBuffer(in).insert(2 * i + 1, 'X').toString();
length = (int) in.length() / 2 + in.length() % 2;
}
}
//------------makes plaintext of even length--------------
//creates an array of digraphs
String[] digraph = new String[length];
//loop iterates over the plaintext
for(int j = 0; j < length ; j++)
{
//checks the plaintext is of even length or not
if(j == (length - 1) && in.length() / 2 == (length - 1))
//if not addends X at the end of the plaintext
in = in + "X";
digraph[j] = in.charAt(2 * j) +""+ in.charAt(2 * j + 1);
```

```
}
//encodes the digraphs and returns the output
String out = "";
String[] encDigraphs = new String[length];
encDigraphs = encodeDigraph(digraph);
for(int k = 0; k < length; k++)
out = out + encDigraphs[k];
return out;
}
//---------------encryption logic-----------------
//encodes the digraph input with the cipher's specifications
private String[] encodeDigraph(String di[])
{
String[] encipher = new String[length];
for(int i = 0; i < length; i++)
{
char a = di[i].charAt(0);
char b = di[i].charAt(1);
int r1 = (int) getPoint(a).getX();
int r2 = (int) getPoint(b).getX();
int c1 = (int) getPoint(a).getY();
int c2 = (int) getPoint(b).getY();
//executes if the letters of digraph appear in the same row
//in such case shift columns to right
if(r1 == r2)
{
c1 = (c1 + 1) % 5;
```

```
c2 = (c2 + 1) % 5;
}
//executes if the letters of digraph appear in the same column
//in such case shift rows down
else if(c1 == c2)
{
r1 = (r1 + 1) % 5;
r2 = (r2 + 1) % 5;
}
//executes if the letters of digraph appear in the different row and different column
//in such case swap the first column with the second column
else
{
int temp = c1;
c1 = c2;
c2 = temp;
}
//performs the table look-up and puts those values into the encoded array
encipher[i] = table[r1][c1] + "" + table[r2][c2];
}
return encipher;
}
//----------------------decryption logic---------------------
// decodes the output given from the cipher and decode methods (opp. of encoding process)
private String decode(String out)
{
```

```
String decoded = "";
for(int i = 0; i < out.length() / 2; i++)
{
char a = out.charAt(2*i);
char b = out.charAt(2*i+1);
int r1 = (int) getPoint(a).getX();
int r2 = (int) getPoint(b).getX();
int c1 = (int) getPoint(a).getY();
int c2 = (int) getPoint(b).getY();
if(r1 == r2)
{
c1 = (c1 + 4) % 5;
c2 = (c2 + 4) % 5;
}
else if(c1 == c2)
{
r1 = (r1 + 4) % 5;
r2 = (r2 + 4) % 5;
}
else
{
//swapping logic
int temp = c1;
c1 = c2;
c2 = temp;
}
decoded = decoded + table[r1][c1] + table[r2][c2];
```

```
}
//returns the decoded message
return decoded;
}
// returns a point containing the row and column of the letter
private Point getPoint(char c)
{
Point pt = new Point(0,0);
for(int i = 0; i < 5; i++)
for(int j = 0; j < 5; j++)
if(c == table[i][j].charAt(0))
pt = new Point(i,j);
return pt;
}
//function prints the key-table in matrix form for playfair cipher
private void keyTable(String[][] printTable)
{
System.out.println("Playfair Cipher Key Matrix: ");
System.out.println();
//loop iterates for rows
for(int i = 0; i < 5; i++)
{
//loop iterates for column
for(int j = 0; j < 5; j++)
{
//prints the key-table in matrix form
System.out.print(printTable[i][j]+" ");
```

```
}
System.out.println();
}
System.out.println();
}
//method that prints all the results
private void printResults(String encipher, String dec)
{
System.out.print("Encrypted Message: ");
//prints the encrypted message
System.out.println(encipher);
System.out.println();
System.out.print("Decrypted Message: ");
//prints the decryted message
System.out.println(dec);
}
}
```

Output:

```
Enter the key for playfair cipher: diamond
Enter the plaintext to be encipher: birmingham
Playfair Cipher Key Matrix:

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

Encrypted Message: HBTIDBHKMO

Decrypted Message: BIRMINGHAM
```

Conclusion: Hence we have understood the implementation of Playfair Cipher.

Name: Arnav Malvia

Roll No.: 2103109

Batch: C23

EXPERIMENT 4

Aim: Implementation of Euler's Totient Function

Theory:

Euler's totient function, also known as phi-function, counts the number of integers between 1 and n inclusive, which are coprime to n. Two numbers are coprime if their greatest common divisor equals 1 (1 is considered to be coprime to any number).

Here are values of phi-n for the first few positive integers:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
$\phi(n)$	1	1	2	2	4	2	6	4	6	4	10	4	12	6	8	8	16	6	18	8	12

Properties:

The following properties of Euler totient function are sufficient to calculate it for any number:

1. If p is a prime number, then gcd(p,q) = 1 for all 1<=q<p. Therefore we have: phi(n)=p-1
2. If p is a prime number and k>=1, then there are exactly p^k/p numbers between 1 and p^k that are divisible by p. Which gives us: phi(p^k)=p^k=p^k-1
3. If a and b are relatively prime, then: phi(ab)=phi(a).phi(b)
4. In general, for not coprime a and b, the equation: phi(ab)=phi(a).phi(b).d/phi(d) with d=gcd(a,b) holds.

Thus, using the first three properties, we can compute phi(n) through the factorization of n. If n = p1^a1.p2^a2…….pk^ak where pi are prime factors of n,

$$\phi(n) = \phi({p_1}^{a_1}) \cdot \phi({p_2}^{a_2}) \cdots \phi({p_k}^{a_k})$$

$$= ({p_1}^{a_1} - {p_1}^{a_1-1}) \cdot ({p_2}^{a_2} - {p_2}^{a_2-1}) \cdots ({p_k}^{a_k} - {p_k}^{a_k-1})$$

$$= p_1^{a_1} \cdot \left(1 - \frac{1}{p_1}\right) \cdot p_2^{a_2} \cdot \left(1 - \frac{1}{p_2}\right) \cdots p_k^{a_k} \cdot \left(1 - \frac{1}{p_k}\right)$$

$$= n \cdot \left(1 - \frac{1}{p_1}\right) \cdot \left(1 - \frac{1}{p_2}\right) \cdots \left(1 - \frac{1}{p_k}\right)$$

Code:

```
import java.util.*;
class euler {

    // Function to return GCD of a and b
    static int gcd(int a, int b)
    {
        if (a == 0)
            return b;
        return gcd(b % a, a);
    }

    // A simple method to evaluate
    // Euler Totient Function
    static int phi(int n)
    {
        int result = 1;
        for (int i = 2; i < n; i++)
            if (gcd(i, n) == 1)
```

```
                result++;
        return result;
    }

    // Driver code
    public static void main(String[] args)
    {
        int n;
         Scanner sc = new Scanner(System.in);

        System.out.println("Enter n: ");
        n = sc.nextInt();
            System.out.println("phi(" + n + ") = " + phi(n));
    }
}
```

Output:

```
Enter n:
10
phi(10) = 4
```

Conclusion: Hence we have understood the implementation of Euler Totient Functions in Java.

Name: Arnav Malvia

Roll No.: 2103109

Batch: C23

EXPERIMENT 5

Aim: Implementation of RSA cryptosystem

Theory:

The RSA algorithm is a public-key signature algorithm developed by Ron Rivest, Adi Shamir, and Leonard Adleman. Their paper was first published in 1977, and the algorithm uses logarithmic functions to keep the working complex enough to withstand brute force and streamlined enough to be fast post-deployment. The image below shows it verifies the digital signatures using RSA methodology.

RSA can also encrypt and decrypt general information to securely exchange data along with handling digital signature verification. The image above shows the entire procedure of the RSA algorithm. You will understand more about it in the next section.

RSA in Data Encryption

When using RSA for encryption and decryption of general data, it reverses the key set usage. Unlike signature verification, it uses the receiver's public key to encrypt the data, and it uses the receiver's private key in decrypting the data. Thus, there is no need to exchange any keys in this scenario.

There are two broad components when it comes to RSA cryptography, they are:

Key Generation: Generating the keys to be used for encrypting and decrypting the data to be exchanged.

Encryption/Decryption Function: The steps that need to be run when scrambling and recovering the data.

Steps in RSA Algorithm

Keeping the image above in mind, go ahead and see how the entire process works, starting from creating the key pair, to encrypting and decrypting the information.

Key Generation

You need to generate public and private keys before running the functions to generate your ciphertext and plaintext. They use certain variables and parameters, all of which are explained below:

Choose two large prime numbers (p and q)

Calculate n = p*q and z = (p-1)(q-1)

Choose a number e where 1 < e < z

Calculate d = e-1mod(p-1)(q-1)

You can bundle private key pair as (n,d)

You can bundle public key pair as (n,e)

Encryption/Decryption Function

Once you generate the keys, you pass the parameters to the functions that calculate your ciphertext and plaintext using the respective key.

If the plaintext is m, ciphertext = me mod n.

If the ciphertext is c, plaintext = cd mod n

To understand the above steps better, you can take an example where p = 17 and q=13. Value of e can be 5 as it satisfies the condition 1 < e < (p-1)(q-1).

N = p * q = 221

D = e-1mod(p-1)(q-1) = 29

Public Key pair = (221,5)

Private Key pair = (221,29)

If the plaintext(m) value is 10, you can encrypt it using the formula me mod n = 82.

To decrypt this ciphertext(c) back to original data, you must use the formula cd mod n = 29.

Code:

```
import java.util.Scanner;


public class RSA {


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


        int p, q, n, z, d = 0, e = 0, i;


        // Take input for p
        System.out.print("Enter the value of p: ");
        p = scanner.nextInt();


        // Take input for q
        System.out.print("Enter the value of q: ");
        q = scanner.nextInt();


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


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

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

        // d is for private key exponent
        if (x % e == 0) {
            d = x / e;
            break;
        }
    }
    System.out.println("the value of d = " + d);

    // Perform encryption and decryption for a sample message (e.g., 12)
    int msg = 12;
    int c = power(msg, e, n);
    System.out.println("Encrypted message is : " + c);

    int msgback = power(c, d, n);
    System.out.println("Decrypted message is : " + msgback);
}
```

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

    static int power(int x, int y, int p) {
        int res = 1; // Initialize result
        x = x % p; // Update x if it is more than or
        // equal to p
        while (y > 0) {
            // If y is odd, multiply x with result
            if (y % 2 == 1)
                res = (res * x) % p;

            // y must be even now
            y = y >> 1; // y = y/2
            x = (x * x) % p;
        }
        return res;
    }
}
```

Output:

```
Enter the value of p: 11
Enter the value of q: 7
the value of z = 60
the value of e = 7
the value of d = 43
Encrypted message is : 12
Decrypted message is : 12
```

Conclusion: Hence we have understood the implementation of RSA in cryptosystem using Java.

Name: Arnav Malvia

Roll No.: 2103109

Batch: C23

Experiment 6

Aim: Implementation of Diffie Hellman Key exchange algorithm

Theory:

The Diffie-Hellman algorithm is being used to establish a shared secret that can be used for secret communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.

For the sake of simplicity and practical implementation of the algorithm, we will consider only 4 variables, one prime P and G (a primitive root of P) and two private values a and b.

P and G are both publicly available numbers. Users (say Alice and Bob) pick private values a and b and they generate a key and exchange it publicly. The opposite person receives the key and that generates a secret key, after which they have the same secret key to encrypt.

```
Step 1: Alice and Bob get public numbers P = 23, G = 9

Step 2: Alice selected a private key a = 4 and
        Bob selected a private key b = 3

Step 3: Alice and Bob compute public values
Alice:    x =(9^4 mod 23) = (6561 mod 23) = 6
        Bob:    y = (9^3 mod 23) = (729 mod 23)  = 16

Step 4: Alice and Bob exchange public numbers

Step 5: Alice receives public key y =16 and
        Bob receives public key x = 6

Step 6: Alice and Bob compute symmetric keys
        Alice:  ka = y^a mod p = 65536 mod 23 = 9
        Bob:    kb = x^b mod p = 216 mod 23 = 9

Step 7: 9 is the shared secret.
```

Code:

```
class DiffieHellman {

    // Power function to return value of a ^ b mod P
    private static long power(long a, long b, long p)
    {
        if (b == 1)
            return a;
        else
            return (((long)Math.pow(a, b)) % p);
    }
```

```
// Driver code
public static void main(String[] args)
{
        long P, G, x, a, y, b, ka, kb;

        // Both the persons will be agreed upon the
        // public keys G and P

        // A prime number P is taken
        P = 23;
        System.out.println("The value of P:" + P);

        // A primitive root for P, G is taken
        G = 9;
        System.out.println("The value of G:" + G);

        // Alice will choose the private key a
        // a is the chosen private key
        a = 4;
        System.out.println("The private key a for Alice:"
                                        + a);

        // Gets the generated key
        x = power(G, a, P);

        // Bob will choose the private key b
        // b is the chosen private key
```

```
        b = 3;
        System.out.println("The private key b for Bob:"
                                        + b);

        // Gets the generated key
        y = power(G, b, P);

        // Generating the secret key after the exchange
        // of keys
        ka = power(y, a, P); // Secret key for Alice
        kb = power(x, b, P); // Secret key for Bob

        System.out.println("Secret key for the Alice is:"
                                        + ka);
        System.out.println("Secret key for the Bob is:"
                                        + kb);
    }
}
```

Output:

```
The value of P:23
The value of G:9
The private key a for Alice:4
The private key b for Bob:3
Secret key for the Alice is:9
Secret key for the Bob is:9
```

Conclusion: Hence we have understood the implementation of Diffie Hellman Key exchange algorithm.

Name: Arnav Malvia

Roll No.: 2103109

Batch: C23

Experiment 7

Aim: Study the use of network Reconnaissance Tools and apply the following: WHOIS, dig, traceroute, nslookup

Theory:

Whois

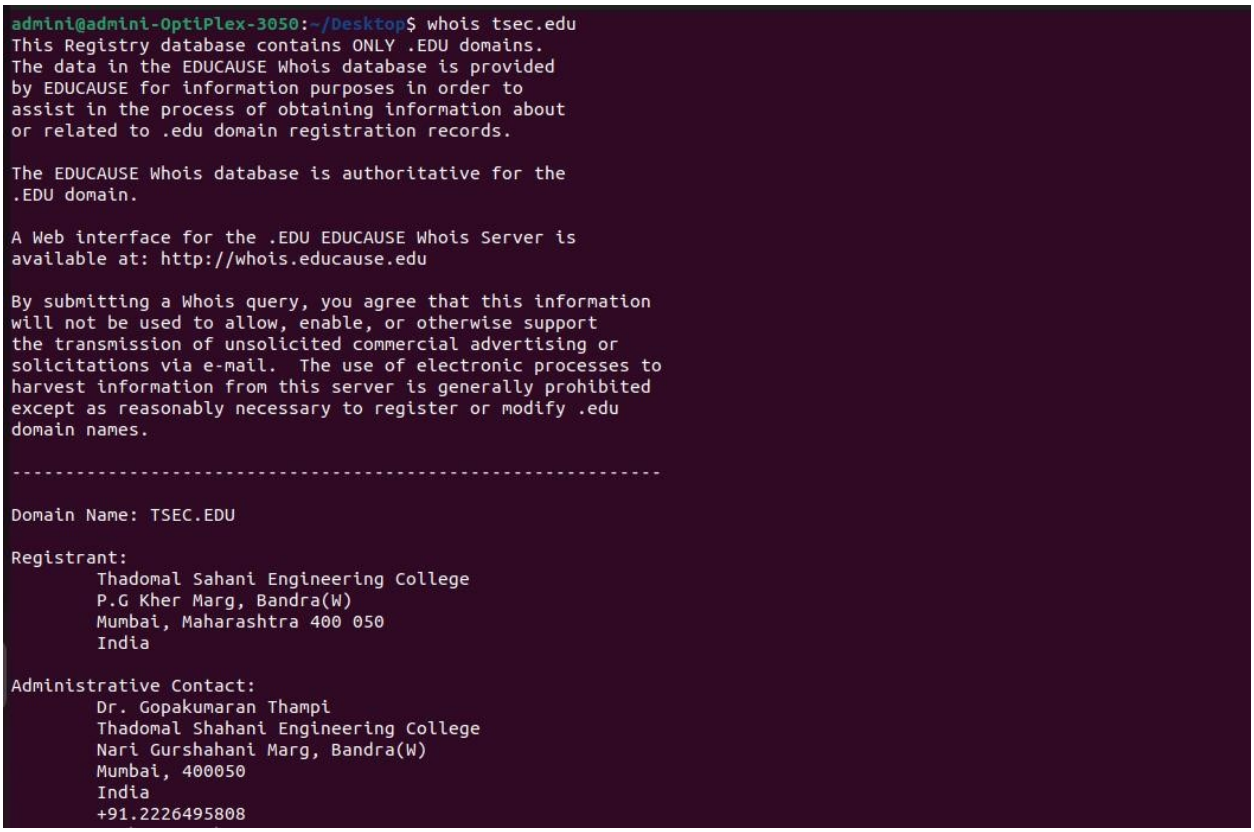
Command: whois tsec.edu

Explanation:

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

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

Screenshot:



Dig

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

Command: dig www.google.com

Explanation:

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

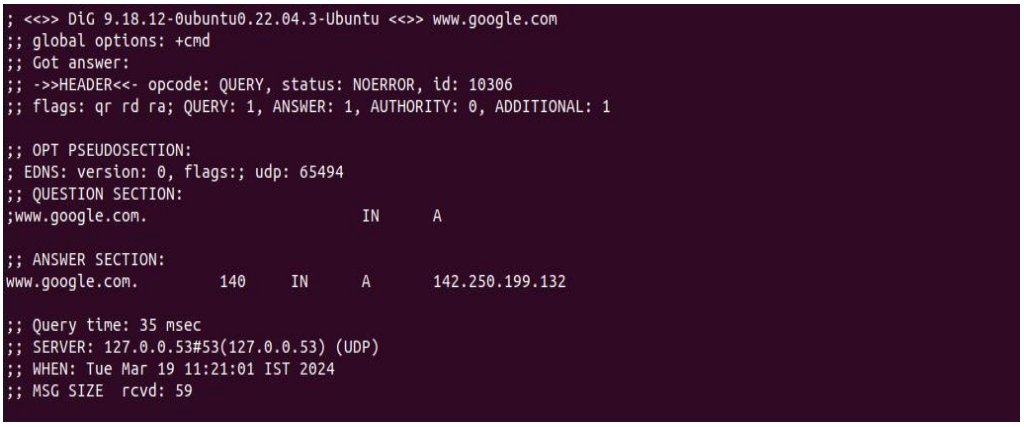
2. Name Servers: DNS servers store information about domain names and their corresponding IP addresses. When you run `dig`, it also provides information about the authoritative name servers responsible for the domain. These name servers hold the official records for the domain and can provide additional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

3. Additional Records: Apart from the IP address, `dig` can also retrieve other DNS records associated with the domain, such as MX (Mail Exchange) records for email

servers, TXT records for text information like SPF (Sender Policy Framework) records used for email authentication, and more. These additional records can be crucial for understanding how a domain is configured and used.

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

Screenshot:



Command: dig google.com +noquestion

Explanation:

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

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

Screenshot:

```
; <<>> DiG 9.18.18-0ubuntu0.22.04.2-Ubuntu <<>> google.com +noquestion
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6635
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; ANSWER SECTION:
google.com.             262     IN      A       142.251.42.46

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Wed Mar 20 13:31:06 IST 2024
;; MSG SIZE  rcvd: 55
```

Command: dig google.com +nocomments

Explanation:

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

Screenshot:

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

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

Command:

dig google.com +noauthority

Explanation:

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

Screenshot:

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

; <<>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <<>> google.com +noauthority
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20274
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

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

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

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

Command: dig google.com +noadditional

Explanation:

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

Screenshot:

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

; <<>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <<>> google.com +noadditional
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64277
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

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

;; ANSWER SECTION:
google.com.             172     IN      A       172.217.167.174

;; Query time: 96 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Tue Mar 19 11:36:44 IST 2024
;; MSG SIZE  rcvd: 55
```

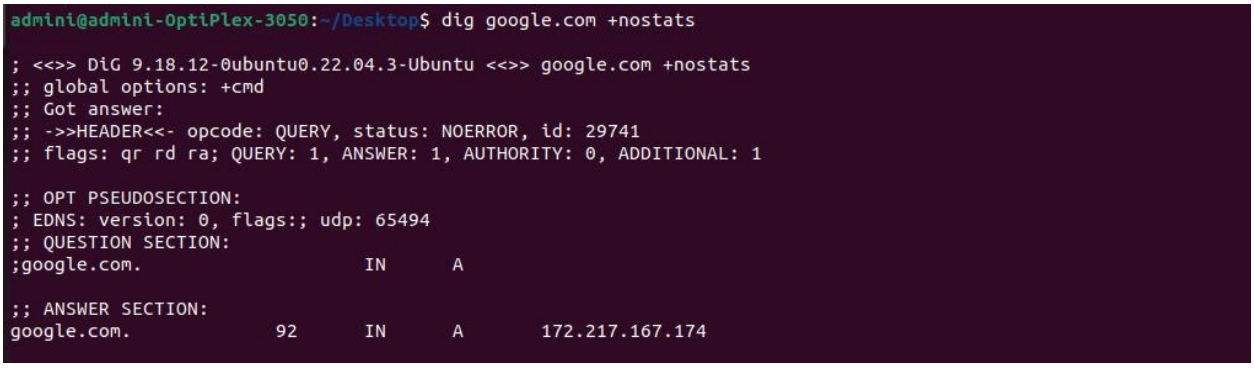
Command:

dig google.com +nostats

Explanation:

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

Screenshot:



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:

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:

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:

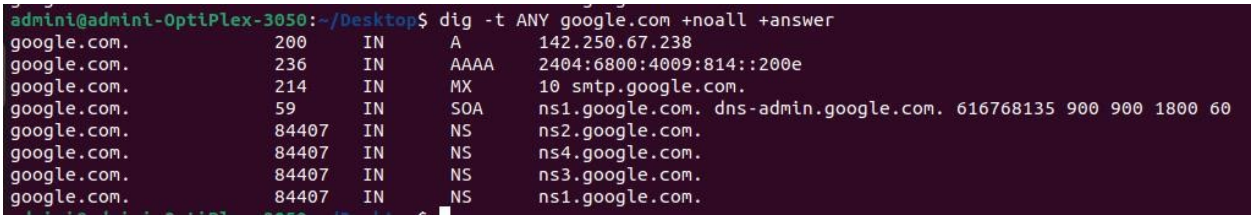
dig -t ANY google.com +noall +answer

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.

Screenshot:



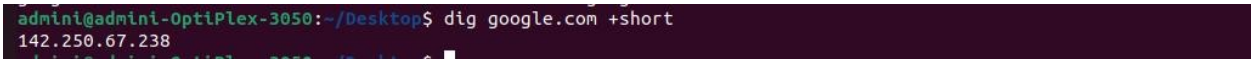
Command:

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.

Screenshot:

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

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

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

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

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

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

;; ANSWER SECTION:
209.132.183.81.         0       IN      A       209.132.183.81

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

Traceroute

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

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

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

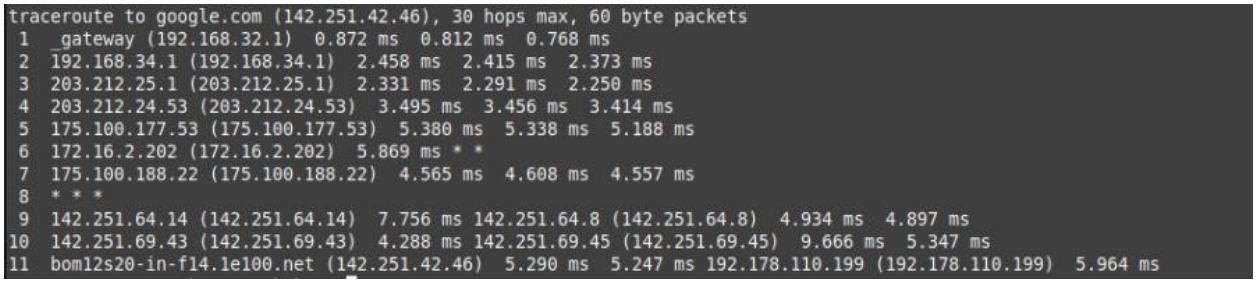
Command:

traceroute google.com

Explanation:

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

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



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. 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.

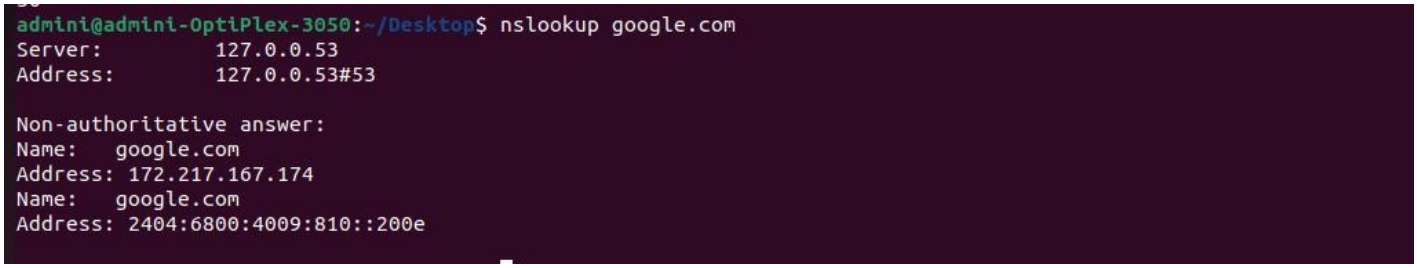
Command:

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

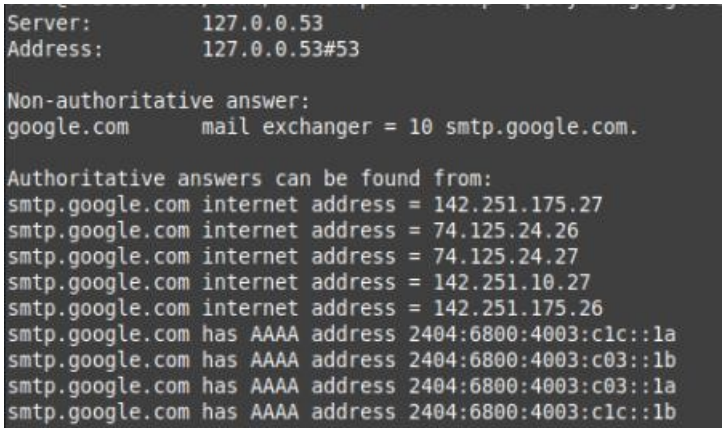
Explanation:

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

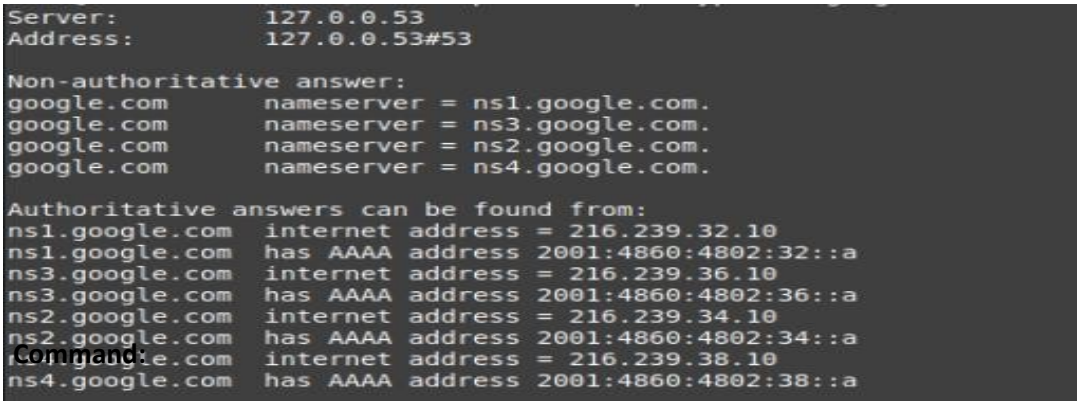
Explanation:

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:



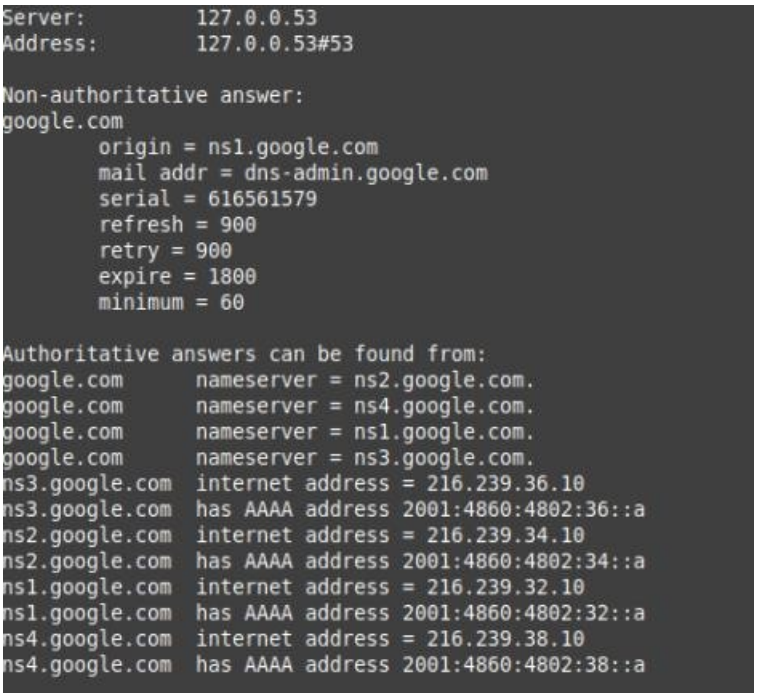
Command:

#nslookup -type = soa google.com

Explanation:

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

Screenshot:



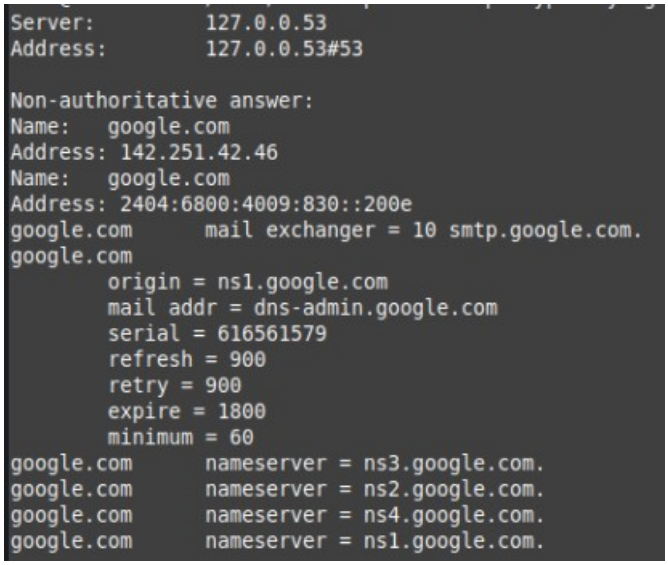
Command:

nslookup -type = any google.com

Explanation:

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

Screenshot:



Extra Commands:

Command:

ipconfig

Explanation:

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

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

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

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

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

Screenshot:

```
C:\Users\Vikas>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet 2:

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

Wireless LAN adapter Local Area Connection* 1:

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

Wireless LAN adapter Local Area Connection* 2:

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

Wireless LAN adapter Wi-Fi:

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

Command:

Ping www.google.com

Explanation:

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

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

Screenshot:

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

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

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

Command:

Tracert www.google.com

Explanation:

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

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

Screenshot:

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

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

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

Trace complete.
```

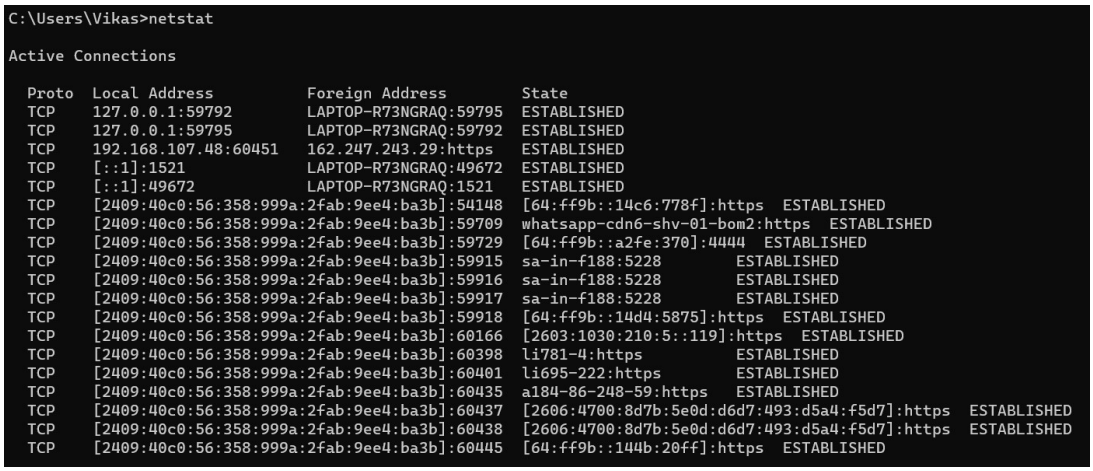
Command:

netstat

Explanation:

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

Screenshot:



```
C:\Users\Vikas>netstat

Active Connections

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

Conclusion: Hence we have understood the concept of network Reconnaissance Tools.

Name: Arnav Malvia

Roll No.: 2103109

Batch: C23

Experiment 8

Aim: Study and Implementation of packet sniffer tool: wireshark

Theory:

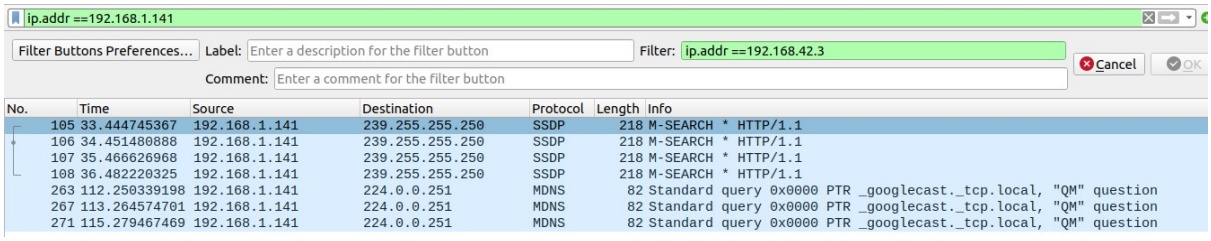
Command:

ip.addr == 192.168.42.3

Explanation:

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

Screenshot:



Command: ip.addr ==

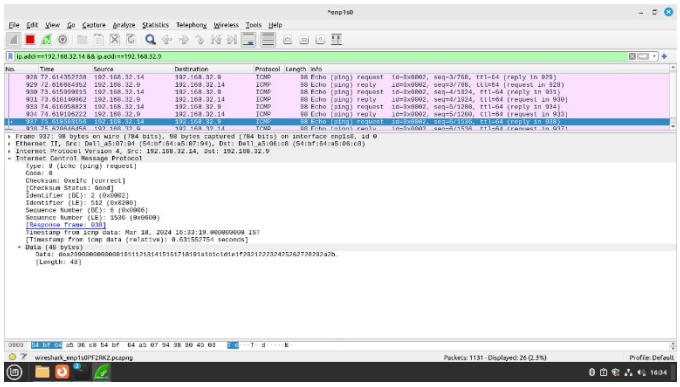
10.0.5.119 && ip.addr ==

91.189.94.25

Explanation:

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

Screenshot:



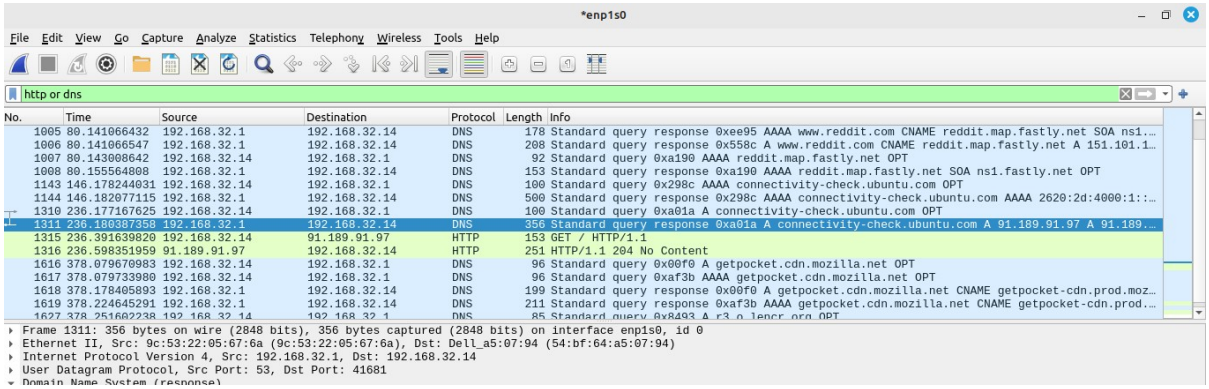
Command:

http or dns

Explanation:

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

Screenshot:



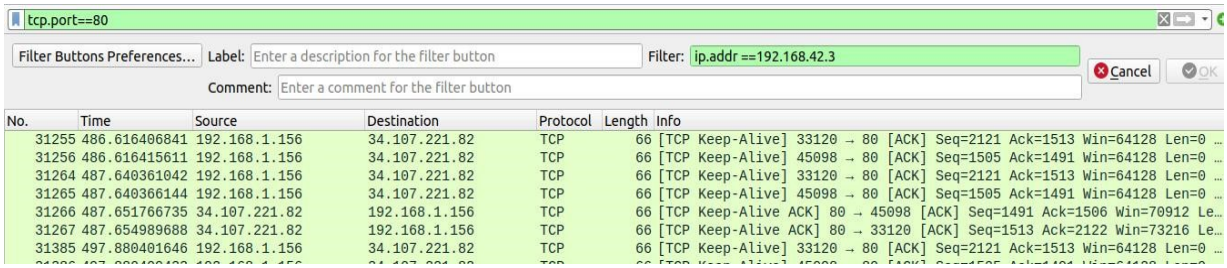
Command:

tcp.port==4000

Explanation:

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

Screenshot:



Command:

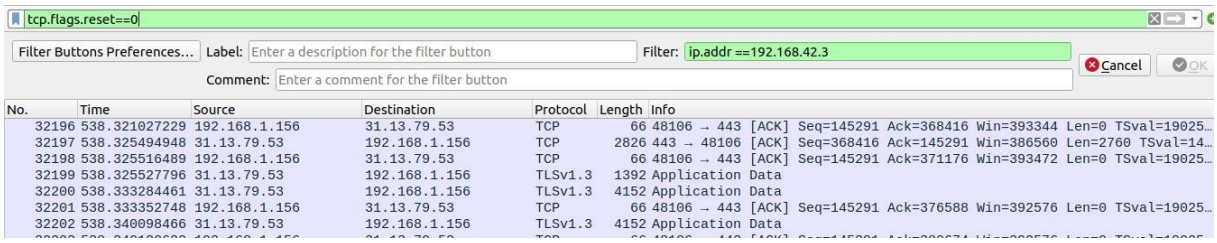
tcp.flags.reset== 0

Explanation:

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

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

Screenshot:



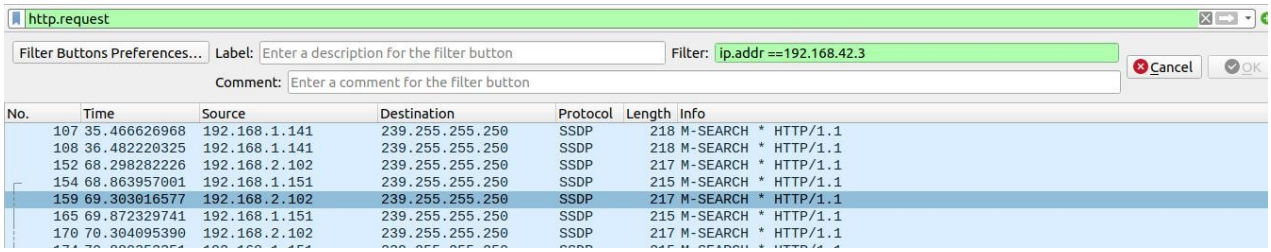
Command:

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

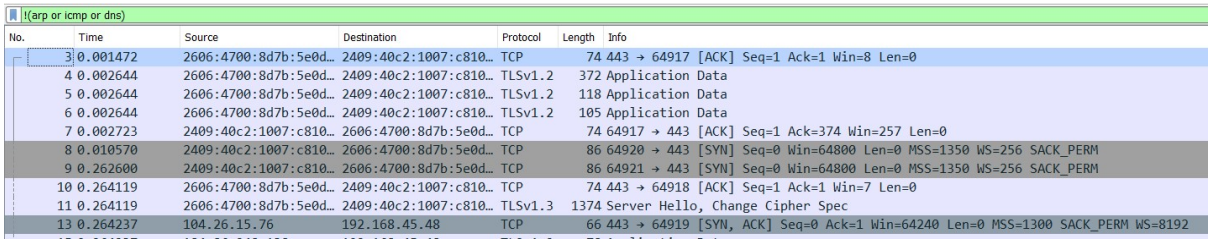
icmp or

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:



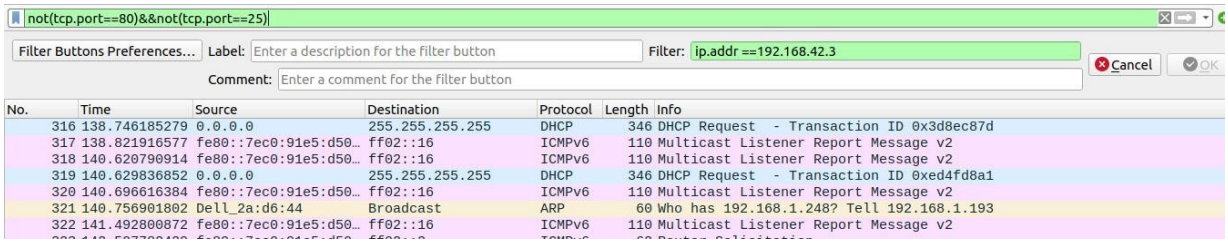
Command: not (tcp.port ==

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:



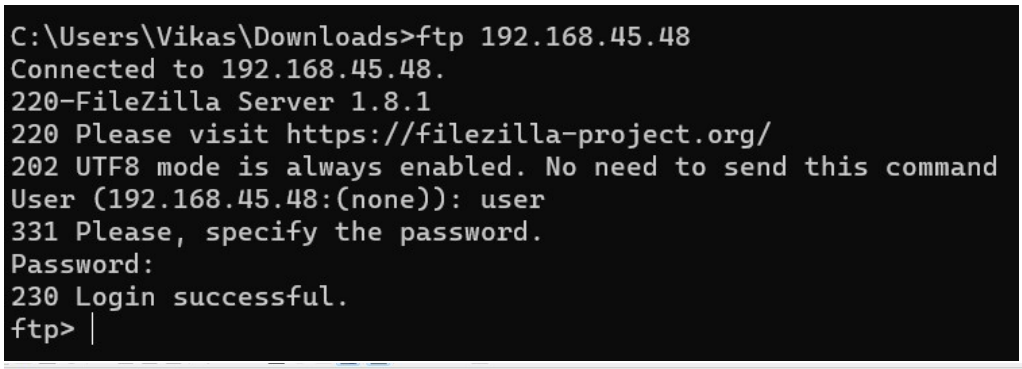
Command:

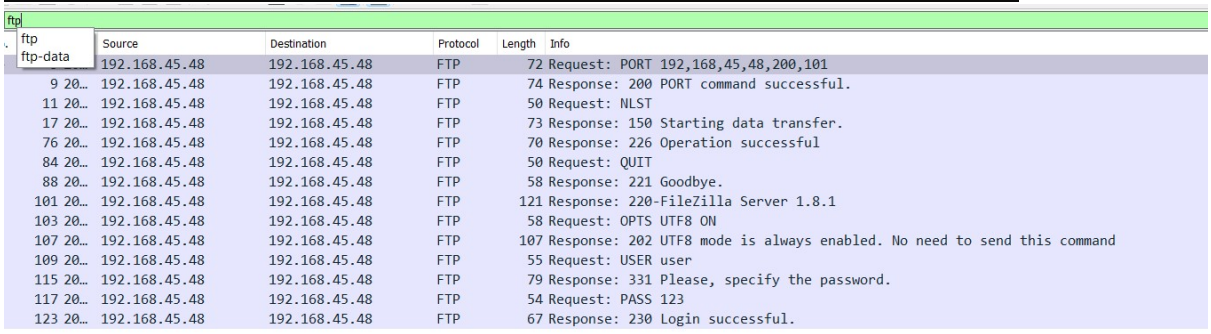
ftp

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:





Conclusion: Hence we have understood the implementation of packet sniffer tool in wireshark.

Name: Arnav Malvia
Roll No.: 2103109
Batch: C23

Experiment No. 9

Aim: **Setting up personal Firewall using iptables**

Theory:-

Description

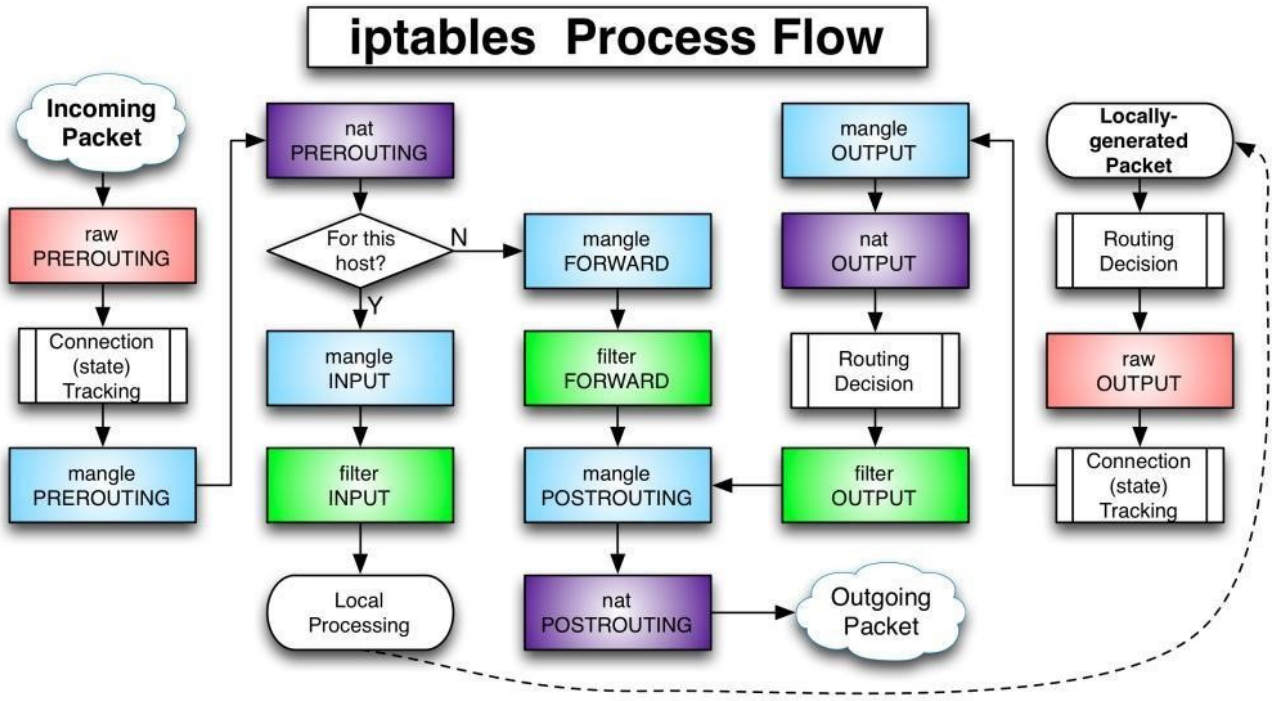
Iptables is used to set up, maintain, and inspect the tables of IP packet filter rules in the Linux kernel. Several different tables may be defined. Each table contains a number of built-in chains and may also contain user-defined chains.

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

Targets

A firewall rule specifies criteria for a packet, and a target. If the packet does not match, the next rule in the chain is 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.



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

1. Filter Table

Filter is default table for iptables.

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

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

2. NAT Table

This table is consulted when a packet that creates a new connection is encountered. Iptable's NAT table has the following built-in chains.

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

3. Mangle Table

Iptables's Mangle table is for specialized packet alteration. This alters QOS bits in the TCP header.

Mangle table has the following built-in chains.

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

4. Raw Table

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

- PREROUTING chain
- OUTPUT chain

5. Security Table

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

Chains

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

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

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

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

Targets:

ACCEPT: Allow packet to pass through the firewall.

DROP: Deny access by the packet.

REJECT: Deny access and notify the server.

QUEUE: Send packets to user space.

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

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

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

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

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

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

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

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

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

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

EXTRA:

9. Block all incoming and outgoing traffic by default:

- `iptables -P INPUT DROP`

- `iptables -P OUTPUT DROP`

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

10. Allow established connections:

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

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

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

Steps:-

1. Get root access: $ sudo su root

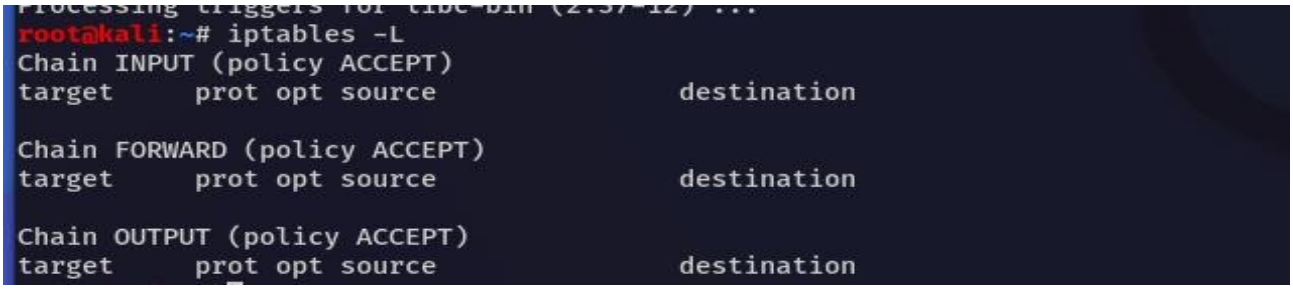
2. # apt-get install iptables

Commands:-

1. To see the list of iptables rules

iptables -L

. Initially it is empty



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

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

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

#ping 192.168.208.6

Open another terminal and run the iptables command

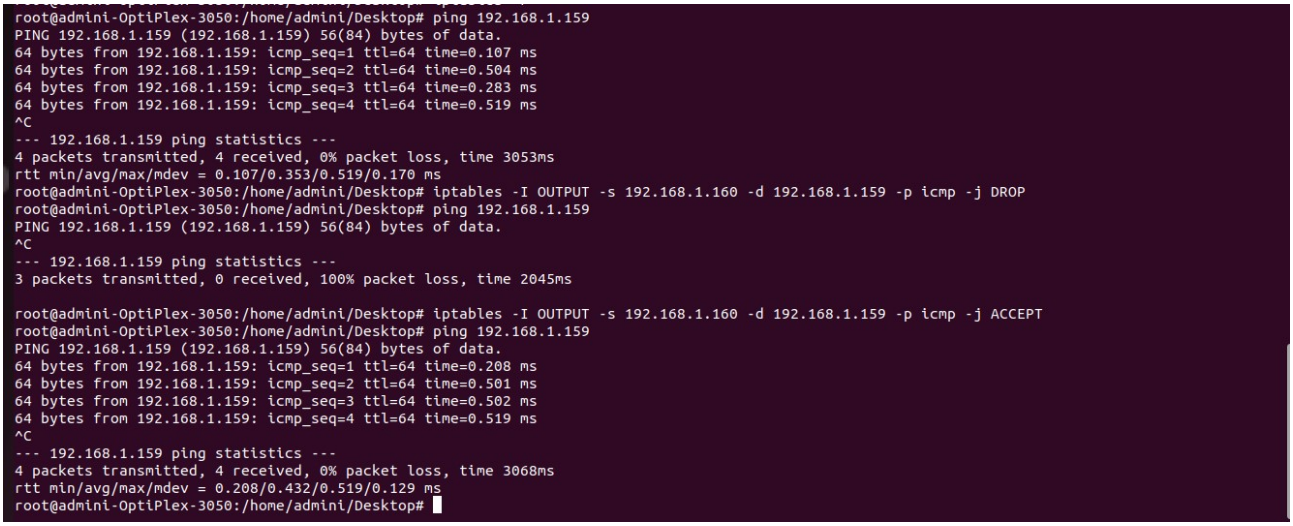
iptables -I OUTPUT -s 192.168.208.18 -d 192.168.208.6 -p icmp -j DROP

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

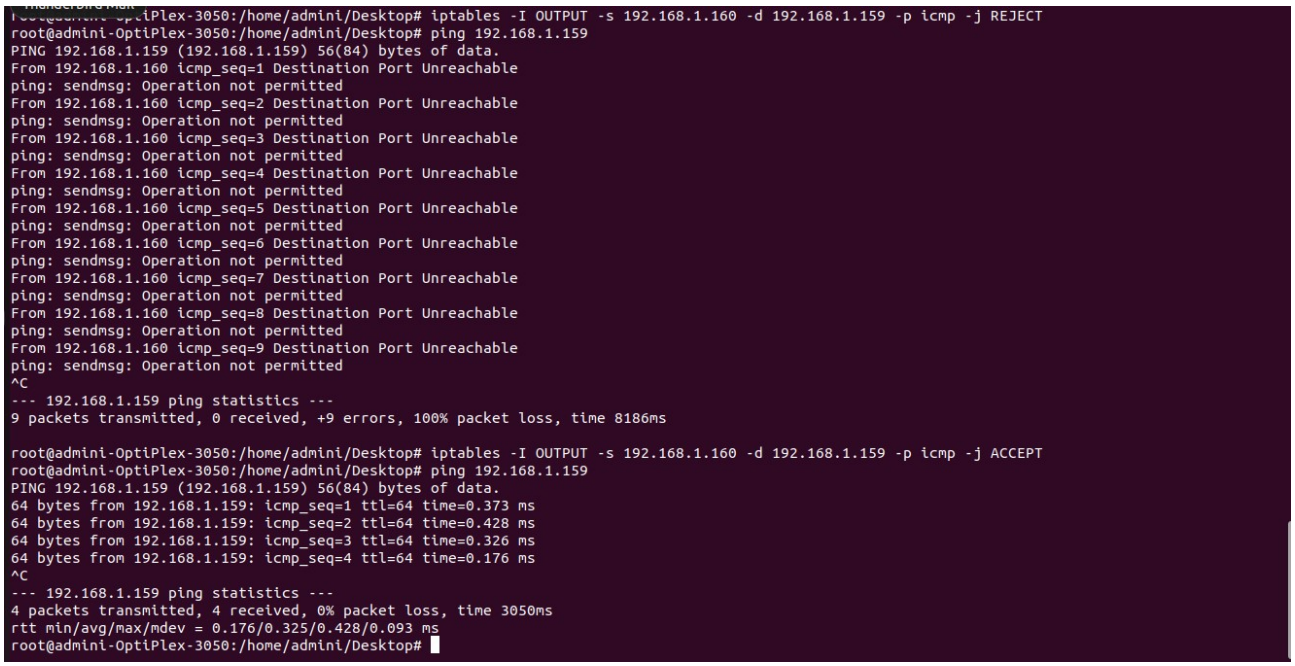
iptables -I OUTPUT -s 192.168.208.18 -d 192.168.208.6 -p icmp -j ACCEPT

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

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



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>

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:~$
```

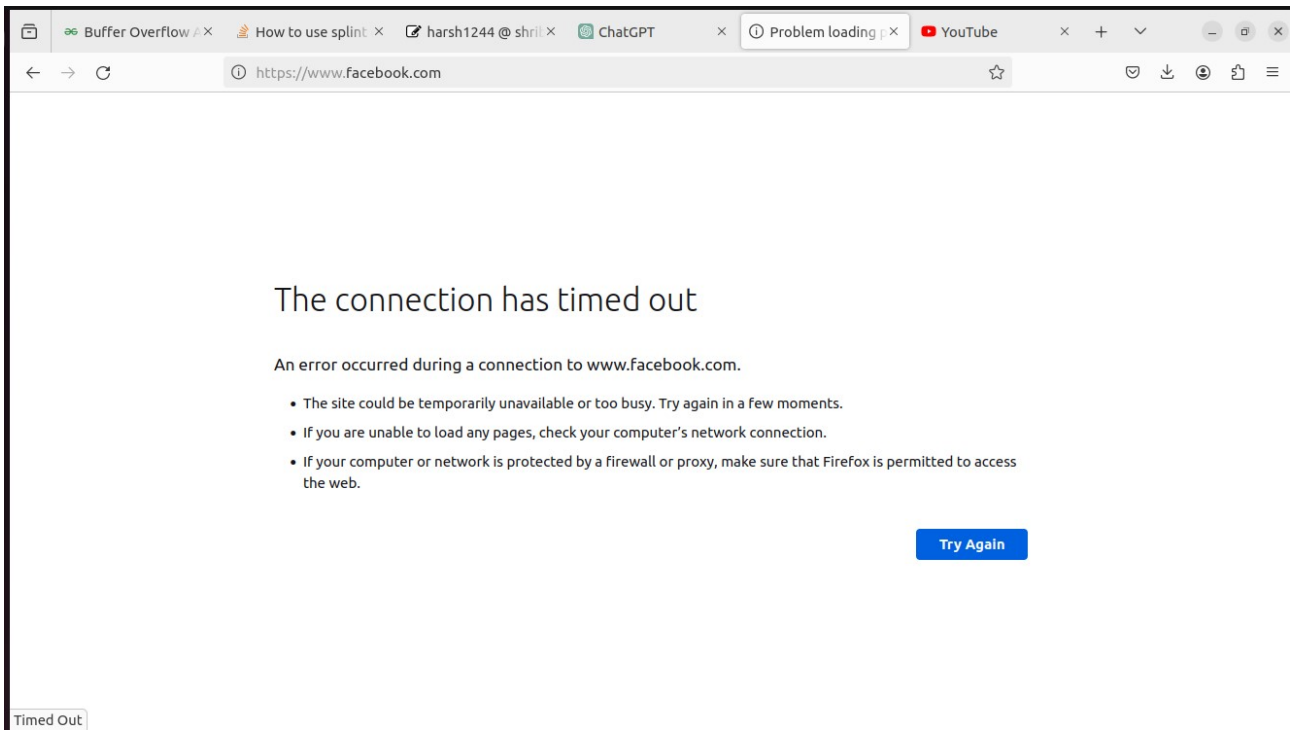
6. To clear the rules in iptables

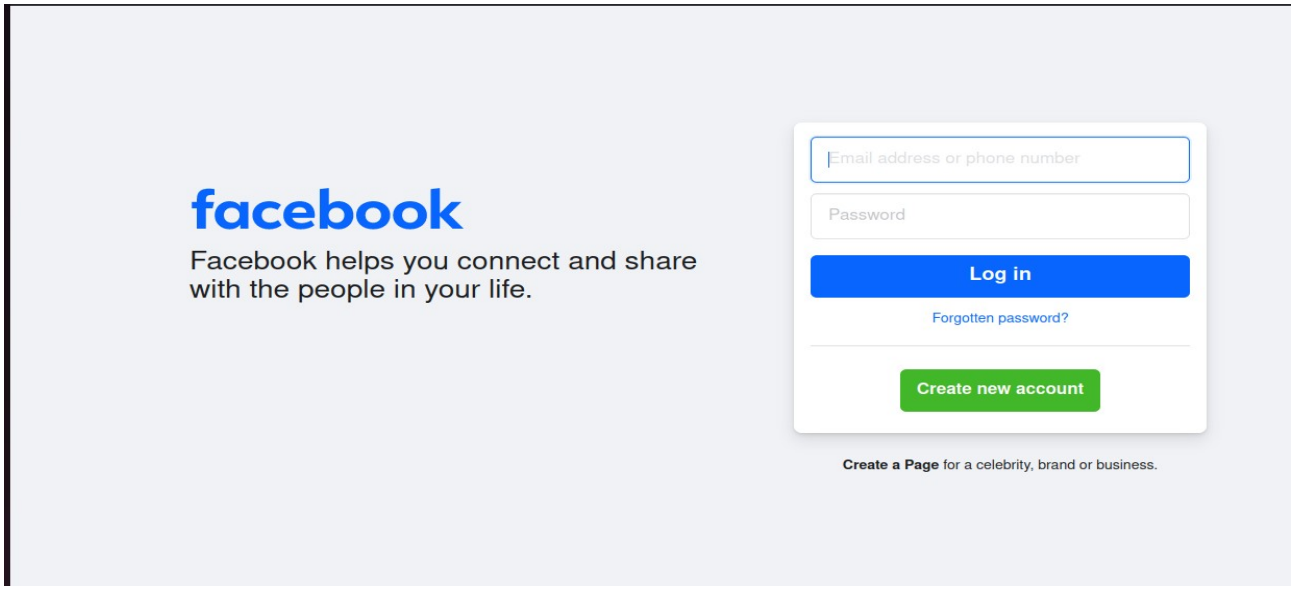
iptables -F

7. To block specific URL from machine

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

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





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

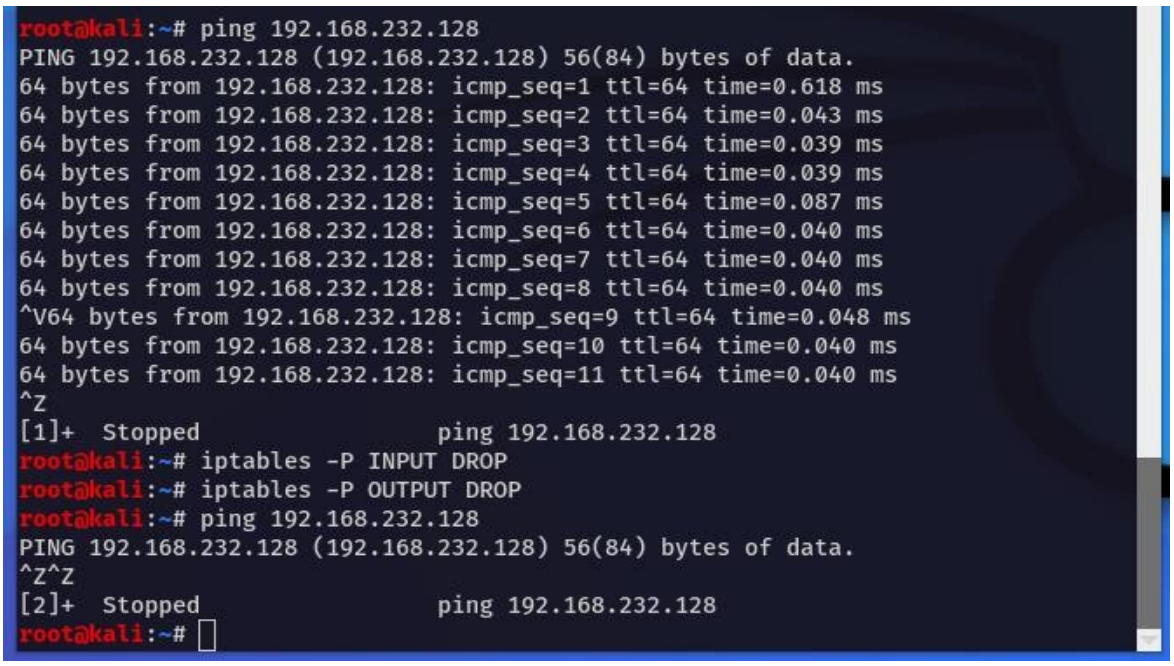
EXTRA :

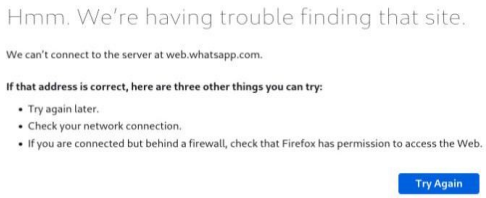
To block all incoming and outgoing traffic by default:

iptables -P INPUT DROP iptables
-P OUTPUT DROP

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

To





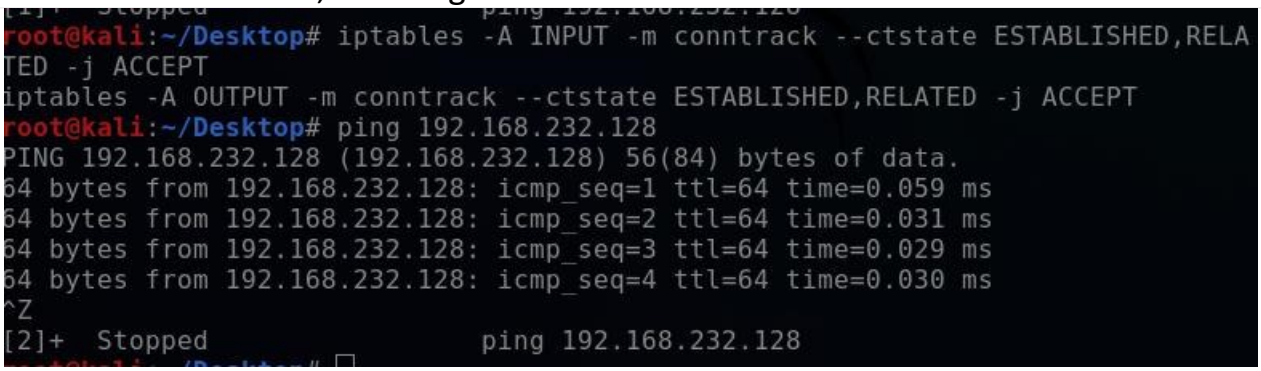
allow

established connections:

cssCopy code	

iptables NPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables - OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

ands allow incoming and outgoing traffic that is part of an established nnection, ensuring 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.

Conclusion: Hence we have understood the implementation of setting up personal Firewall using iptables.

Name: Arnav Malvia
Roll No.: 2103109
Batch: C23

Experiment 10

Aim: Simulation of Buffer Overflow Attack

Theory:

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

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

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

overflow vulnerabilities, unsafe implementations or calls of macros, and deviations from prescribed naming conventions.

Steps :

1. Installation

$ sudo apt-get install splint

2. Checking Vulnerability

$ splint program1.c

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

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

```
 {
 printf ("\n Wrong Password
\n");
 }
 else
 {
 printf ("\n Correct Password
\n");  pass = 1;}}
```

Program-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");
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");
}
```

Program 3

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

char password[] = "password"; int

get_password() {  int auth_ok = 0;  char

buff[16];  printf("Enter password: ");

scanf("%s", buff);  if(strncmp(buff,

password, sizeof(password)) == 0)

auth_ok = 1;  return auth_ok; } void

success() {  printf("Success!

\n");

}

int main(int argc, char**

argv) {  int res =

get_password();  if (res

== 0) {  printf("Failure

\n");  return 0;

 }

 success();

return 0;

 }
```

Output :

Program 1

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

 Enter the password :
lab403

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

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

Finished checking --- 4 code warnings
```

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

 Enter the password :
thegeekstuff

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

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

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

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]='abcdefghijklmnoph

sgkgfks'; |

^~~~~~~~~~~~~~~~~~~~~

~~~~~

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

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"

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"

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"

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%p\n"

program2.c:8:44: Corresponding

format code

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

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

%p\n%p\n%p\n%p\n%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%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%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%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%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%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%p\n"

program2.c:8:72: Corresponding

format code

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

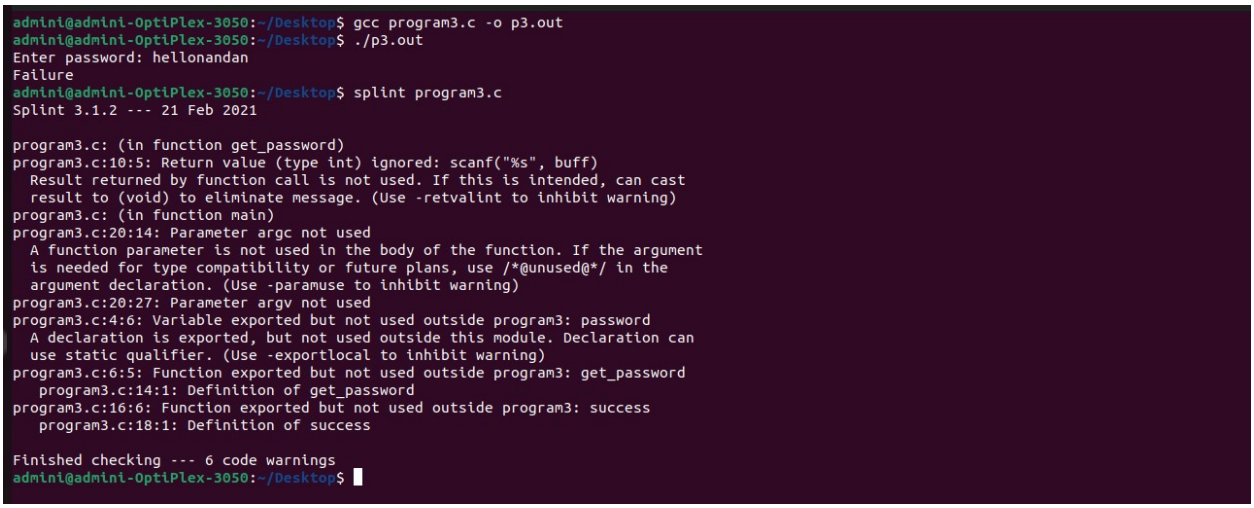
There is a path through a function declared to return a value on which there

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



Extra Splint Programs and Outputs :

Buffer Overflow in Local Variables:

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

```
#include <stdio.h>


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

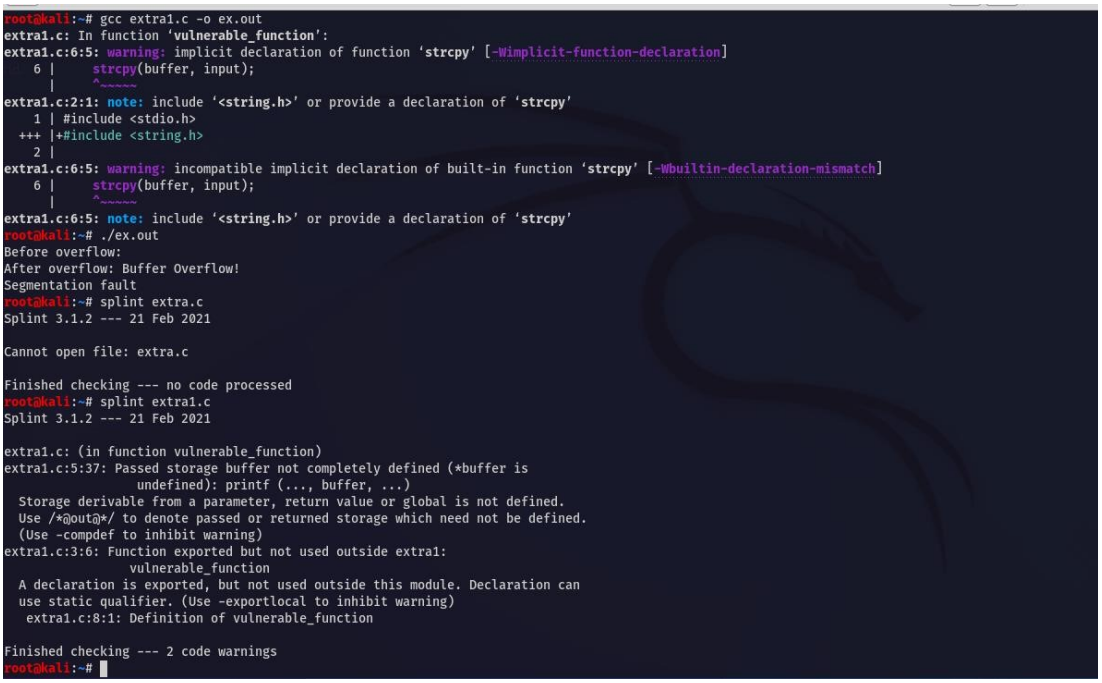
int main() { char input[20]

= "Buffer Overflow!";

vulnerable_function(input);

return 0;

}



Integer Overflow leading to Buffer Overflow:

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

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

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

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

}
```

Conclusion: Hence we have understood the simulation of Buffer Overflow Attack.

Assignment :- 1

QA Explain different services and mechanisms of security.

Ans A Security Services:

- Data confidentiality
- Data Integrity
  - Anti change
  - Anti Replay
- Authentication
  - Peer entity
  - Data origin
- Non Repudiation
  - prob of origin
  - prob of delivery
- Access control

1) Data confidentiality -
Confidentiality is the fundamental security service provided by cryptography. It is a security service that keeps the information on unauthorised person. Sometimes referred as privacy or secrecy. It prevents snooping and traffic analysis attack.

2) Data Integrity -
Designed to protect data from modification, insertion, deletion and replaying by an adversary. Its assurance that data received are exactly as sent by an authorised ~~ent~~ entity.

3) Authentication -
It provides the identification of the originator. It confirms to the receiver that the data received has been sent only by an identified and verified sender.

4) Non - Repudiation -

It is a security service that an entity cannot refuse the ownership of a previous commitement or an action. It is a property that is most desirable in situations where there are chances of a dispute over the exchange of data.

5) Access Control -

It provides protection against unauthorised access to data. It involves reading, writing, modifying, etc.

Security Mechanisms:

1) Encipherment -

Hiding or covering data can provide confidentiality. It can also be used to complement other mechanisms to provide other services.

2) Data Integrity -

Appends to the data a short check value that has been created by a specific process from the data itself.

3) Digital signature -

By this the sender can electronically sign the data, the receiver can electronically verify the signature. Sender uses its private key and receiver can verify it using the sender public key.

4) Authentication Exchange -
In this mechanism the two entities exchange some messages to prove their identity to each other.

5) Traffic Padding -
It inserts some bogus data into the data traffic to thwart the adversary attempt to use traffic analysis

6) Routing control -
It means selecting and continuously changing different available routes between the user and the receiver to prevent the opponent from eaves dropping on a particular note.

7) Notarization -
It involves use of trusted third party in communication. It acts as a mediator between sender and receiver so that if any chance of conduct is reduced. This mediator keeps record of requests made by sender to receiver for later denied.

8) Access control -
This mechanism is used to stop unintended access to data which you are sending, it can be achieved by various techniques such as applying passwords, using firewall or just by adding password to data.

Q B What are the various types of attacks? Explain with example.

Ans B Attacks Threatening Confidentiality:

a) Snooping -

Snooping is unauthorised access to data or interception of data. eg - A confidential file being transferred on the internet may be intercepted by an attacker and the confidential information can be used by him.

b) Traffic Analysis-

Refers to maintaining online traffic to obtain information. It can be used to collect requests and responses to guess the nature of transaction. eg- Attacker can find the email address of the sender or the receiver.

Attacks Threatening Integrity:

a) Modification -

After accessing the information, the attacker can modify or delete the information. eg - Customer sends a message to a bank to do some transaction. The attacker intercepts it and changes the messages to benefit him.

b) Masquerading -

When the attacker impersonates somebody else. eg- Attacker can attack the bank card and pin of a bank customer and pretend that she is the customer.

c) Replaying -
The attacker obtains a copy of message sent by a user and later tries to replay it.

d) Repudiation -
A repudiation attack happens when an application or system does not adapt controls to properly attack and log user actions thus permitting malicious manipulation.
eg- A bank customer requesting the bank to send money to a third party and later denying that they made the request.

Attacks Threatening Availability:
a) Denial of Service -
It is an attack meant ~~the~~ to shut down a machine or network making it inaccesible to its intended users. The concept is to send more traffic to a network address than the programmers have built the system to handle.

Qc List few latest viruses on internet and explain them.
Ans C A malicious program used to replicate, make modifications to other systems and enter its own code is known as an internet virus. Its replication can impact various features of computer system and then the system becomes a compromised device.
Some of the latest computer viruses around the internet are explained below:
1) 2-Worm - It is a virus capable of spreading via making its replica. It sends itself to other connected

devices via email. Moreover, it is also known as worm because it spreads itself as a network.

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

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

4) Yatron Ransomware - Yatron is a ransomware type virus that is promoted as "Ransomware as Service". After succesful infiltration, Yatron encrypts most of the data and appends filename with .yatron extension.

5) Astaroth Malware - It is an information stealer ~~machine~~ malware. It transfers confidential information from an attacked like account id and passwords, keystrokes and other data to the attacker.

A

184

Assignment :- 2

Q1 Write short note on Digital Signature and Digital Certificate.

Ans 1 Digital Signature :-

Digital Signature is a mathematical technique used to validate the authenticity and integrity of a message.

Process -

a) Key generation algorithms -

Digital signature is electronic signature which assures that the message was sent by a particular sender. While performing digital transactions authenticity and integrity should be assured otherwise that data can be altered or someone can also act as if he was under and except a reply.

b) Signing Algorithms -

To create a digital signature, signing algorithms like email programs create to one way has the electronic data which is to be signed. The signing algorithm then encrypts the hash value using the private key. The This encrypted hash along with other information like the hashing algorithm is the digital signature. This digital signature is appended with the data and sent to the visitor. The reason for encrypting the hash instead of entire message or documents is that the hash function converts arbitary input into a much shorter fixed length value. This also saves time as we do not have to sign a big message.

c) Signature verification Algorithms-
Verifier receives digital signature along with the data. It then uses verification algorithms to process the digital signature and the public key and generates some value. It also applies the same hash function on the received data and generates a hash value. If they both are equal, then the digital signature is valid, else it is invalid.

Digital Certificate :-
Digital certificate is issued by a trusted third party which proves sender's identity to the receiver and receiver's identity to the sender.
Digital certificate contains-
1) Name of certificate holder
2) Serial number which is used to uniquely identify a certificate, the individual or the entity identified by the certificate
3) Expiration Date.
4) Copy of certificate holders public key used for decrypting messages.
5) Digital signature of the certificate issuing authority

Advantages of Digital certificate:-
1) Network Security-
A complete layered strategy is required by modern cybersecurity methods, wherein many solutions cooperate to offer the highest level of protection against malevolent actors. An essential component of this puzzle is digital certificates, which offers

strong defence against manipulation and man-in the middle assoults.

2) Verification-
Digital certificates facilitate cyber security by restricting access to sensitive data which makes authentication a crucial component of cybersecurity. Thus, there is a decreased chance that hostile actors will cause chaos. At many different points certificate based ~~aut~~ authentication provides a dependable method of identity verification. Compared to other popular authentication methods like biometrics or one time passwords, certificates are more flexible.

3) Buyer Success-
Astute consumers demand complete assurance that the websites they visit are reliable because digital certificates are supported by certificate authority that users browser trust, they offer a credity identifiable indicator of reliability.

Disadvantages of Digital certificate:-
1) Phishing attacks
2) Weak encryption.
3) Misconfiguration.

Assignment :- 3

Q1 List all the software vulnerabilities. How are they exploited to launch an attack?

Ans 1 Software vulnerabilities are weakness or flaws in the software systems that can be exploited by attackers to compromise security, integrity or availability of system or data. These vulnerabilities can range from programming errors and design flaws to improper configurations and weak authentication mechanisms. Exploiting these vulnerabilities can lead to various cyber threats including unauthorised access, data breaches, system crashes and execution of malicious codes.

Following are some of the software vulnerabilities:

1) Buffer Overblow -
Occurs when an attacker modifies or writes more data to the buffer which leads to change in execution of program, system crashes and execution of malicious code.

2) SQL injection -
Exploit vulnerabilities in web applications that interact with databases allowing attackers to manipulate SQL queries to gain unauthorised access or modify data.

3) Cryptographic Failures -
Improper implementation of encryption hashing or key management mechanisms.

4) Cross side scripting -
Allows attackers to inject malicious scripts into web pages viewed by other users, comprising their security or stealing information.

5) Click jacking -
Tricks users into clicking malicious links or buttons by disguising them as legitimate elements on web page.

6) Denial of Service -
Overwhelms a system or network ~~with effect~~ with excessive traffic, rending it unavailable to legitimate users.

7) Insecure Transport Layer -
Utilises outdated or weak encryption protocols, allowing attackers to manipulate encrypted communications.

Clickjacking :-
It is an attack that tricks a user into clicking a webpage element which is invisible or disguised as another element. It can cause users to unwillingly download malware, visit malicious web-pages, provide credentials or transfer money or purchase products online. Several variations such as like jacking and cursor jacking.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

For example-
The user visit the page and clicks the "Book my Free Trip" Button.

Another examples :-
In reality the user is clicking on the invisible iframe and has clicked the "Confirm Transfer" button. Funds are transferred to attacker.

Prevention Methods :-
1) Use the "X-Frame options" HTTP response header.
2) Add a Frame killer to the website.
3) Employ a security policy to specify which domains are allowed to frame your pages.
4) Keep your system patched.
5) Employ defensive code in the user interface.

Phishing Attack :-
It involves tricking users into revealing sensitive information such as login credentials or financial data by reflecting as a legitimate entity via email, instant messages, etc.

Prevention Methods :-
1) Security Awareness - Educate users about the risk of phishing attacks and how to recognise them.
2) Two-factor Authentication - Implement 2FA to add an extra layer of security beyond passwords reducing risks of compromised credentials.

3) URL Inspection - Carefully inspecting URL's to avoid clicking on suspicious links.

4) Email Filtering - Use spam blockers and email authentication mechanisms to detect and block phishing emails.