Roll No: 2103163
Batch: C32
Name: Om Shete

Experiment No 1

Aim: Implementation of Extended Euclidean algorithm.

Description:

The Euclidean algorithm is a way to find the greatest common divisor of two positive integers. GCD of two numbers is the largest number that divides both of them. A simple way to find GCD is to factorize both numbers and multiply common prime factors.

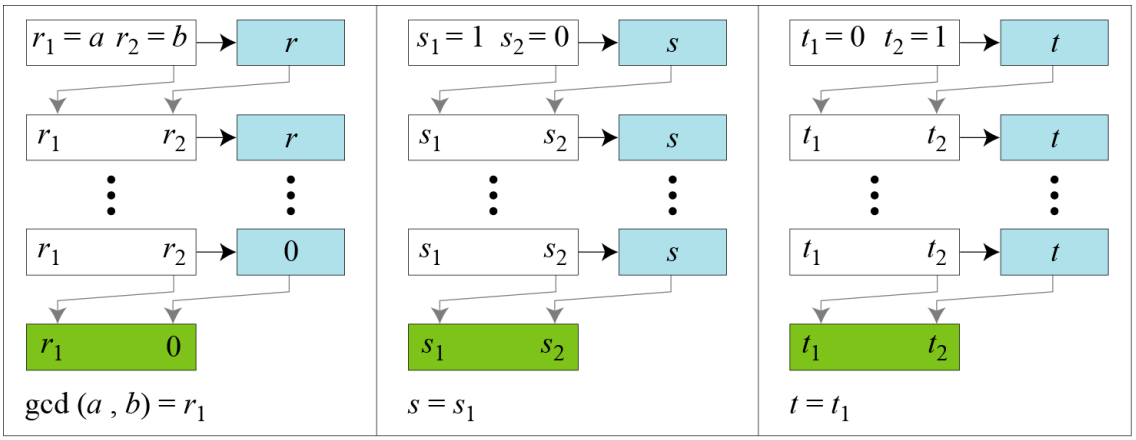
36 = 2 x 2 x 3 x 3
60 = 2 x 2 x 3 x 5

GCD = Multiplication of common factors
= 2 x 2 x 3
= 12

Given two integers a and b, we often need to find other two integers, s and t, such that -

$$s \times a + t \times b = \gcd(a, b)$$

The extended Euclidean algorithm can calculate the gcd (a, b) and at the same time calculate the value of s and t.



a. Process

Roll No: 2103163
Batch: C32
Name: Om Shete

$r_1 \leftarrow a; \quad r_2 \leftarrow b;$
$s_1 \leftarrow 1; \quad s_2 \leftarrow 0;$
$t_1 \leftarrow 0; \quad t_2 \leftarrow 1;$ (Initialization)

while $(r_2 > 0)$
{
$q \leftarrow r_1 / r_2;$

$r \leftarrow r_1 - q \times r_2;$
$r_1 \leftarrow r_2; \; r_2 \leftarrow r;$ (Updating r's)

$s \leftarrow s_1 - q \times s_2;$
$s_1 \leftarrow s_2; \; s_2 \leftarrow s;$ (Updating s's)

$t \leftarrow t_1 - q \times t_2;$
$t_1 \leftarrow t_2; \; t_2 \leftarrow$ t; (Updating t's)

}
gcd $(a, b) \leftarrow r_1; \; s \leftarrow s_1; \; t \leftarrow t_1$

b. Algorithm

Given a = 161 and b = 28, find gcd (a, b) and the values of s and t.

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
5	161	28	21	1	0	1	0	1	−5
1	28	21	7	0	1	−1	1	−5	6
3	21	7	0	1	−1	4	−5	6	−23
	7	0		**−1**	4		**6**	−23	

We get gcd (161, 28) = 7, s = −1 and t = 6.

The extended Euclidean algorithm finds the multiplicative inverses of b in Z_n when n and b are given and gcd (n, b) = 1.

The multiplicative inverse of b is the value of t after being mapped to Z_n.

Roll No: 2103163
Batch: C32
Name: Om Shete

Code:

```
#include <iostream>
using namespace std;

int main() {
  int a, b;
  cout << "Enter the first number: ";
  cin >> a;
  cout << "Enter the second number: ";
  cin >> b;

  int r1 = max(a, b);
  int r2 = min(a, b);
  a = r1;
  b = r2;

  int s1 = 1;
  int s2 = 0;
  int t1 = 0;
  int t2 = 1;

  cout << "Q r1 r2 r s1 s2 s t1 t2 t" << endl;
  cout << "-----------------------------------" << endl;

  while (r2 > 0) {
    int q = r1 / r2;
    int rem = r1 % r2;
    int s = s1 - (q * s2);
    int t = t1 - (q * t2);

    cout << q << " " << r1 << " " << r2 << " " << rem << " " << s1 << " " << s2
         << " " << s << " " << t1 << " " << t2 << " " << t << endl;

    r1 = r2;
    r2 = rem;
    s1 = s2;
    s2 = s;
    t1 = t2;
    t2 = t;
  }
```

Roll No: 2103163
Batch: C32
Name: Om Shete

```
  cout << endl;
  cout << "The Euclidean Eq is ax + by = GCD: " << endl;
  cout << "Proof: " << endl;
  cout << "a is : " << a << endl;
  cout << "b is : " << b << endl;
  cout << "x is : " << s1 << endl;
  cout << "y is : " << t1 << endl;
  cout << "GCD is : ax + by i.e " << a << "(" << s1 << ") + " << b << "(" << t1
       << ") = " << (a * s1 + b * t1) << endl;

  return 0;
}
```

Results:

```
Enter the first number: 161
Enter the second number: 28
Q r1 r2 r s1 s2 s t1 t2 t
-----------------------------------
5 161 28 21 1 0 1 0 1 -5
1 28 21 7 0 1 -1 1 -5 6
3 21 7 0 1 -1 4 -5 6 -23

The Euclidean Eq is ax + by = GCD:
Proof:
a is : 161
b is : 28
x is : -1
y is : 6
GCD is : ax + by i.e 161(-1) + 28(6) = 7
```

Roll No: 2103163
Batch: C32
Name: Om Shete

Experiment No 2

Aim: Implementation of Ceaser Cipher.

Description:

- 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 named after Julius Caesar, who 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:

1. Write down the plaintext message: HELLO
2. Choose a shift value. In this case, we will use a shift of 3.
3. Replace each letter in the plaintext message with the letter that is three positions to the right in the alphabet.

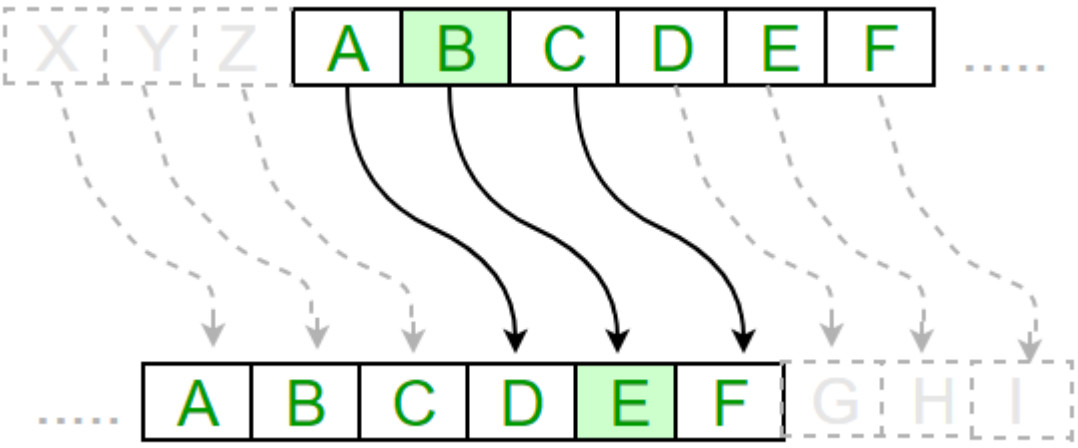
H becomes K (shift 3 from H)
E becomes H (shift 3 from E)
L becomes O (shift 3 from L)
L becomes O (shift 3 from L)
O becomes R (shift 3 from O)

4. The encrypted message is now “KHOOR”.

Roll No: 2103163
Batch: C32
Name: Om Shete
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”.

E_n(x)=(x+n)mod\ 26
(Encryption Phase with shift n)

D_n(x)=(x-n)mod\ 26
(Decryption Phase with shift n)



Text: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Shift: 23
Cipher: XYZABCDEFGHIJKLMNOPQRSTUVW

Advantages:

1. Easy to implement and use thus, making suitable for beginners to learn about encryption.
2. Can be physically implemented, such as with a set of rotating disks or a set of cards, known as a scytale, which can be useful in certain situations.
3. Requires only a small set of pre-shared information.
4. Can be modified easily to create a more secure variant, such as by using multiple shift values or keywords.

Disadvantages:

1. It is not secure against modern decryption methods.
2. Vulnerable to known-plaintext attacks, where an attacker has access to both the encrypted and unencrypted versions of the same messages.

Roll No: 2103163
Batch: C32
Name: Om Shete

3. The small number of possible keys means that an attacker can easily try all possible keys until the correct one is found, making it vulnerable to a brute-force attack.

Code:

```
#include <cctype>
#include <iostream>
using namespace std;

string encrypt(const string &text, int shift) {
  string result = "";

  for (char ch : text) {
    if (isalpha(ch)) {
      char base = isupper(ch) ? 'A' : 'a';
      result += static_cast<char>((ch - base + shift) % 26 + base);
    } else {
      result += ch;
    }
  }
  return result;
}

int main() {
  string plaintext;
  int shift;

  cout << "Enter the text to encrypt: ";
  getline(cin, plaintext);

  cout << "Enter the shift value: ";
  cin >> shift;

  string ciphertext = encrypt(plaintext, shift);

  cout << "Encrypted text: " << ciphertext << endl;

  return 0;
}
```

Roll No: 2103163
Batch: C32
Name: Om Shete

Results:

```
Enter the text to encrypt: strength
Enter the shift value: 5
Encrypted text: xywjslym
```

Roll No: 2103163
Batch: C32
Name: Om Shete

Experiment No 3

Aim: Implementation of Playfair cipher

Description:

The Playfair cipher was the first practical digraph substitution cipher. The scheme was invented in 1854 by Charles Wheatstone but was named after Lord Playfair who promoted the use of the cipher. In Playfair cipher unlike traditional cipher, we encrypt a pair of alphabets(digraphs) instead of a single alphabet.
It was used for tactical purposes by British forces in the Second Boer War and in World War I and for the same purpose by the Australians during World War II. This was because Playfair is reasonably fast to use and requires no special equipment.

Encryption Technique

For the encryption process let us consider the following example:

Key: monarchy
Plaintext: instruments

Encryption Technique

The Playfair Cipher Encryption Algorithm:
The Algorithm consists of 2 steps:

Generate the key Square(5×5):
The key square is a 5×5 grid of alphabets that acts as the key for encrypting the plaintext. Each of the 25 alphabets must be unique and one letter of the alphabet (usually J) is omitted from the table (as the table can hold only 25 alphabets). If the plaintext contains J, then it is replaced by I.

The initial alphabets in the key square are the unique alphabets of the key in the order in which they appear followed by the remaining letters of the alphabet in order.

Roll No: 2103163
Batch: C32
Name: Om Shete
Algorithm to encrypt the plain text: The plaintext is split into pairs of two letters (digraphs). If there is an odd number of letters, a Z is added to the last letter.
For example

PlainText: "instruments"
After Split: 'in' 'st' 'ru' 'me' 'nt' 'sz'

1. A pair cannot be made with the same letter. Break the letter in a single and add a bogus letter to the previous letter.

Plain Text: “hello”
After Split: ‘he’ ‘lx’ ‘lo’
Here ‘x’ is the bogus letter.

2. If the letter is standing alone in the process of pairing, then add an extra bogus letter with the alone letter

Plain Text: “helloe”
AfterSplit: ‘he’ ‘lx’ ‘lo’ ‘ez’
Here ‘z’ is the bogus letter.

Rules for Encryption:

If both the letters are in the same column: Take the letter below each one (going back to the top if at the bottom).

For example
Diagraph: "me"
Encrypted Text: cl
Encryption:
m -> c
e -> l

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

Roll No: 2103163
Batch: C32
Name: Om Shete

Rules for Encryption 1

If both the letters are in the same row: Take the letter to the right of each one (going back to the leftmost if at the rightmost position).

For example
Diagraph: "st"
Encrypted Text: tl
Encryption:
s -> t
t -> l

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

Rules for Encryption 2

If neither of the above rules is true: Form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.

For example
Diagraph: "nt"
Encrypted Text: rq
Encryption:
n -> r
t -> q

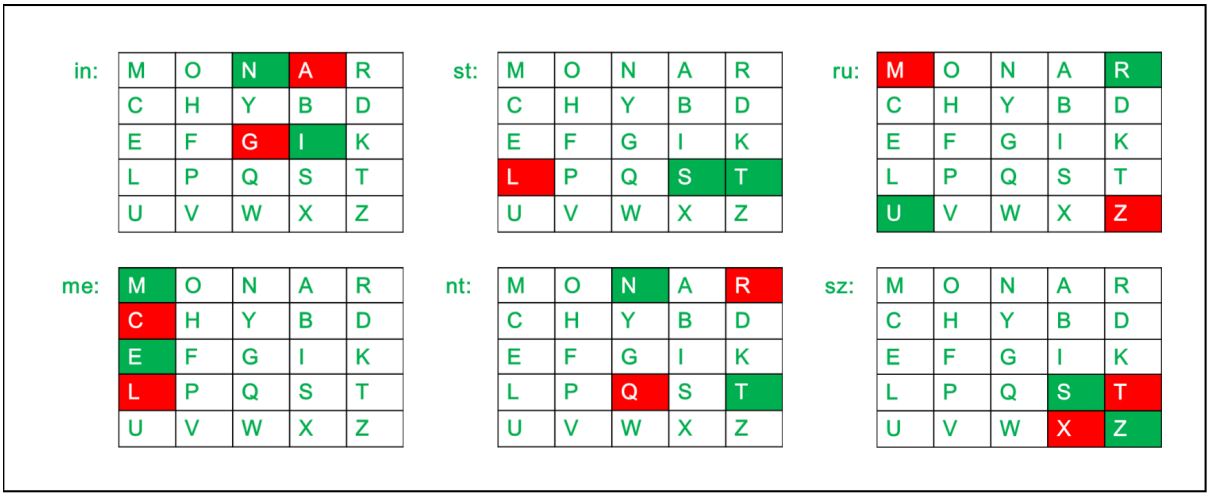
Roll No: 2103163
Batch: C32
Name: Om Shete

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

Rules for Encryption 3

For example

Plain Text: "instrumentsz"
Encrypted Text: gatlmzclrqtx
Encryption:
i -> g
n -> a
s -> t
t -> l
r -> m
u -> z
m -> c
e -> l
n -> r
t -> q
s -> t
z -> x



Roll No: 2103163
Batch: C32
Name: Om Shete

Code:

```
#include <algorithm>
#include <cctype>
#include <cstring>
#include <iostream>
using namespace std;

char mat[5][5];
const char PADD_CHAR = 'x';
bool is_odd = false;

void generate_the_matrix(string key) {
  int checkSmall[26] = {0};
  int r = 0, c = 0;

  for (int i = 0; i < key.length(); i++) {
    if (key[i] == 'j') {
      key = key.substr(0, i) + 'i' + key.substr(i + 1);
    }

    if (checkSmall[key[i] - 'a'] == 0) {
      mat[r][c++] = key[i];
      checkSmall[key[i] - 'a'] = 1;
    }

    if (c == 5) {
      c = 0;
      r++;
    }
  }

  for (char ch = 'a'; ch <= 'z'; ch++) {
    if (ch == 'j') {
      continue;
    }

    if (checkSmall[ch - 'a'] == 0) {
      checkSmall[ch - 'a'] = 1;
      mat[r][c++] = ch;
    }
```

Roll No: 2103163
Batch: C32
Name: Om Shete

```
    if (c == 5) {
      c = 0;
      r++;
    }
  }
}

string partition(string text) {
  for (int i = 0; i < text.length(); i += 2) {
    if (text[i] == text[i + 1]) {
      text.insert(i + 1, 1, PADD_CHAR);
    }
  }

  if (text.length() % 2 == 1) {
    text += PADD_CHAR;
    is_odd = true;
  }

  return text;
}

string encryption(string text, int flag) {
  string result;
  cout << "Encryption Process:\n";
  for (int i = 0; i < text.length(); i += 2) {
    int p1[2], p2[2];
    p1[0] = -1;
    p1[1] = -1;
    p2[0] = -1;
    p2[1] = -1;

    for (int j = 0; j < 5; j++) {
      for (int k = 0; k < 5; k++) {
        if (mat[j][k] == text[i]) {
          p1[0] = j;
          p1[1] = k;
        }

        if (mat[j][k] == text[i + 1]) {
          p2[0] = j;
          p2[1] = k;
        }
```

Roll No: 2103163
Batch: C32
Name: Om Shete

```
        if (p1[0] != -1 && p2[0] != -1) {
          break;
        }
      }

      if (p1[0] != -1 && p2[0] != -1) {
        break;
      }
    }

    char ch1, ch2;
    if (p1[0] == p2[0]) {
      ch1 = mat[p1[0]][(p1[1] + flag) % 5];
      ch2 = mat[p2[0]][(p2[1] + flag) % 5];
    } else if (p1[1] == p2[1]) {
      ch1 = mat[(p1[0] + flag) % 5][p1[1]];
      ch2 = mat[(p2[0] + flag) % 5][p2[1]];
    } else {
      ch1 = mat[p1[0]][p2[1]];
      ch2 = mat[p2[0]][p1[1]];
    }

    result.push_back(ch1);
    result.push_back(ch2);

    cout << text[i] << text[i + 1] << " ---- " << ch1 << ch2 << "\n";
  }

  return result;
}

int main() {
  string text, key;

  cout << "Enter the text: ";
  getline(cin, text);

  cout << "Enter the Key: ";
  getline(cin, key);

  text.erase(remove_if(text.begin(), text.end(), ::isspace), text.end());
  transform(text.begin(), text.end(), text.begin(), ::tolower);
```

```
  key.erase(remove_if(key.begin(), key.end(), ::isspace), key.end());
  transform(key.begin(), key.end(), key.begin(), ::tolower);

  generate_the_matrix(key);

  cout << "The Playfair Matrix: " << endl;
  for (int i = 0; i < 5; i++) {
    for (int j = 0; j < 5; j++) {
      cout << mat[i][j] << " ";
    }
    cout << endl;
  }

  string f_msg = partition(text);
  cout << "\nFormatted msg: " << f_msg << "\n" << endl;

  string cipher = encryption(f_msg, 1);
  cout << "\nEncrypted msg: " << cipher << "\n" << endl;

  cout << "Decryption Process:\n";
  string decryptedMsg = encryption(cipher, 4);
  cout << "\nDecrypted msg: " << decryptedMsg << endl;

  return 0;
}
```

Roll No: 2103163
Batch: C32
Name: Om Shete

Results:

```
Enter the text: instruments
Enter the Key: monarchy
The Playfair Matrix:
m o n a r
c h y b d
e f g i k
l p q s t
u v w x z

Formatted msg: instrumentsx

Encryption Process:
in ---- ga
st ---- tl
ru ---- mz
me ---- cl
nt ---- rq
sx ---- xa

Encrypted msg: gatlmzclrqxa

Decryption Process:
Encryption Process:
ga ---- in
tl ---- st
mz ---- ru
cl ---- me
rq ---- nt
xa ---- sx

Decrypted msg: instrumentsx
```

Roll No: 2103163
Batch: C32
Name: Om Shete

Experiment No 4

Aim: Implementation of Euler's Totient Function.

Description:

Euler's Totient function? (n) for an input n is the count of numbers in {1, 2, 3, …, n-1} that are relatively prime to n, i.e., the numbers whose GCD (Greatest Common Divisor) with n is 1.

Example:

?(1) = 1
gcd(1, 1) is 1

?(2) = 1
gcd(1, 2) is 1, but gcd(2, 2) is 2.

?(3) = 2
gcd(1, 3) is 1 and gcd(2, 3) is 1

?(4) = 2
gcd(1, 4) is 1 and gcd(3, 4) is 1

?(5) = 4
gcd(1, 5) is 1, gcd(2, 5) is 1,
gcd(3, 5) is 1 and gcd(4, 5) is 1

?(6) = 2
gcd(1, 6) is 1 and gcd(5, 6) is 1,

Euler's totient function one may use to know how many prime numbers are coming up to the given integer 'n.' It is also called an arithmetic function. Two things are important for an application or use of Euler's totient function. One is that the gcd formed from the given integer 'n' should be multiplicative. The other is that the numbers of gcd should be the prime numbers only. The integer 'n' in this case should be more than 1. Calculating the Euler's totient function from a negative integer is impossible. The principle, in this case, is that for ϕ(n), the multiplicators called m and n should be greater than 1. Hence, denoted by 1<m<n and gcd (m, n) = 1. Sign ϕ is the sign used to denote the totient function.

Roll No: 2103163
Batch: C32
Name: Om Shete

Properties of Euler's Totient Function

There are some different properties. Some of the properties of Euler's totient function are as under:

- Φ is the symbol used to denote the function.
- The function deals with the prime numbers theory.
- The function is applicable only in the case of positive integers.
- For ϕ (n), one can find two multiplicative prime numbers to calculate the function.
- The function is a mathematical function and useful in many ways.
- If integer 'n' is a prime number, then gcd (m, n) = 1.
- The function works on the formula 1< m< n, where m and n are the prime and multiplicative numbers.
- In general, the equation is:

Φ (mn) = ϕ (m) * ϕ (n) (1- 1 / m) (1 – 1/ n)

Euler's totient function is useful in many ways. One may use it in the RSA encryption system for security purposes. The function deals with the prime number theory, and it is useful in the calculation of large calculations also. One may also use this function in algebraic calculations and elementary numbers. The symbol used to denote the function is ϕ, also called a phi function. The function consists of more theoretical use rather than practical use. The practical use of the function is limited. One can understand the function through various practical examples rather than theoretical explanations. There are various rules for calculating the Euler's totient function, and different rules apply to different numbers. The function was first introduced in 1763. Due to some issues, it got recognition in 1784, and they modified the name in 1879. The function is universal and can be applied everywhere.

Roll No: 2103163
Batch: C32
Name: Om Shete

Code:

```
#include <bits/stdc++.h>
using namespace std;

unordered_map<int, int> primeFactors(int n) {
  unordered_map<int, int> factors;
  for (int i = 2; i * i <= n; ++i) {
    while (n % i == 0) {
      factors[i] = factors[i] + 1;
      n /= i;
    }
  }
  if (n > 1) {
    factors[n] = factors[n] + 1;
  }
  return factors;
}

int gcd(int a, int b) { return b == 0 ? a : gcd(b, a % b); }

void printCoprimes(int n) {
  for (int i = 1; i < n; ++i) {
    if (gcd(i, n) == 1) {
      cout << i << " ";
    }
  }
  cout << endl;
}

void eulerTotient(int n) {
  unordered_map<int, int> factors = primeFactors(n);
  if (factors.size() == 1 && factors[n] == 1) {
    cout << "Euler Totient Function: " << (n - 1) << endl;

    cout << "Coprimes: ";
    printCoprimes(n);
  } else {
    int size = factors.size();
    if (size == 2 && factors.begin()->second == 1 &&
        next(factors.begin())->second == 1) {
```

Roll No: 2103163
Batch: C32
Name: Om Shete

```
      int p = factors.begin()->first;
      int q = next(factors.begin())->first;
      int totient = (p - 1) * (q - 1);
      cout << "Euler Totient Function: " << totient << endl;
      cout << "Coprimes: ";
      printCoprimes(n);
    } else {
      int totient = 1;
      for (const auto &entry : factors) {
        int p = entry.first;
        int k = entry.second;
        totient *= (pow(p, k) - pow(p, k - 1));
      }
      cout << "Euler Totient Function: " << totient << endl;
      cout << "Coprimes: ";
      printCoprimes(n);
    }
  }
}

int main() {
  for (int i = 1; i <= 3; i++) {
    cout << "\nEnter the number for calculating Euler Totient: ";
    int n;
    cin >> n;
    cout << endl;
    eulerTotient(n);
  }
  return 0;
}
```

Roll No: 2103163
Batch: C32
Name: Om Shete

Results:

```
Enter the number for calculating Euler Totient: 19

Euler Totient Function: 18
Coprimes: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

Enter the number for calculating Euler Totient: 60

Euler Totient Function: 16
Coprimes: 1 7 11 13 17 19 23 29 31 37 41 43 47 49 53 59

Enter the number for calculating Euler Totient: 244

Euler Totient Function: 120
Coprimes: 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 5
7 59 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99 101 103 105 107 109 111 113
115 117 119 121 123 125 127 129 131 133 135 137 139 141 143 145 147 149 151 153 155 157 15
9 161 163 165 167 169 171 173 175 177 179 181 185 187 189 191 193 195 197 199 201 203 205
207 209 211 213 215 217 219 221 223 225 227 229 231 233 235 237 239 241 243
```

Roll No: 2103163
Batch: C32
Name: Om Shete

Experiment No 5

Aim: Implementation of RSA cryptosystem.

Description:

The RSA algorithm is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key. As the name describes the Public Key is given to everyone and the Private key is kept private.

An example of asymmetric cryptography:

A client (for example browser) sends its public key to the server and requests some data.
The server encrypts the data using the client's public key and sends the encrypted data.
The client receives this data and decrypts it.
Since this is asymmetric, nobody else except the browser can decrypt the data even if a third party has the public key of the browser.

The idea! The idea of RSA is based on the fact that it is difficult to factorize a large integer. The public key consists of two numbers where one number is a multiplication of two large prime numbers. And private key is also derived from the same two prime numbers. So if somebody can factorize the large number, the private key is compromised. Therefore encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially. RSA keys can be typically 1024 or 2048 bits long, but experts believe that 1024-bit keys could be broken in the near future. But till now it seems to be an infeasible task.

Let us learn the mechanism behind the RSA algorithm : >> Generating Public Key:

Select two prime no's. Suppose P = 53 and Q = 59.
Now First part of the Public key : n = P*Q = 3127.
We also need a small exponent say e :
But e Must be
An integer.
Not be a factor of Φ(n).
1 < e < Φ(n) [Φ(n) is discussed below],
Let us now consider it to be equal to 3.
Our Public Key is made of n and e

>> Generating Private Key:

Roll No: 2103163
Batch: C32
Name: Om Shete

We need to calculate Φ(n) :
Such that Φ(n) = (P-1)(Q-1)
so, Φ(n) = 3016
Now calculate Private Key, d :
d = (k*Φ(n) + 1) / e for some integer k
For k = 2, value of d is 2011.

Now we are ready with our – Public Key (n = 3127 and e = 3) and Private Key(d = 2011) Now we will encrypt “HI”:

Convert letters to numbers : H = 8 and I = 9
Thus Encrypted Data c = (89e)mod n
Thus our Encrypted Data comes out to be 1394
Now we will decrypt 1394 :
Decrypted Data = (cd)mod n
Thus our Encrypted Data comes out to be 89
8 = H and I = 9 i.e. "HI".

Code:

```
#include <iostream>
using namespace std;

int power(int a, int b, int mod) {
  int x = a;
  int result = 1;
  int count = 0;
  cout << "-------------------------------------------------" << endl;
  while (b > 0) {
    if ((b & 1) == 1)
      result = (result * a) % mod;
    if ((b & 1) == 1)
      cout << "Binary Expo: " << x << "^" << (1 << count) << endl;
    a = (a * a) % mod;
    b >>= 1;
    count++;
  }
  cout << "-------------------------------------------------" << endl;
  return result;
}
```

Roll No: 2103163
Batch: C32
Name: Om Shete

```
int encryption(int e, int m, int n) {
  if (m < n) {
    int cipher = power(m, e, n);
    return cipher;
  } else {
    return -1;
  }
}
int gcd(int a, int b) {
  while (b != 0) {
    int temp = b;
    b = a % b;
    a = temp;
  }
  return a;
}
int decryption(int d, int c, int n) {
  int message = power(c, d, n);
  return message;
}

int extendedEuclidean(int r1, int r2) {
  int t1 = 0;
  int t2 = 1;
  while (r2 > 0) {
    int q = r1 / r2;
    int rem = r1 % r2;
    int t = t1 - (q * t2);
    r1 = r2;
    r2 = rem;
    t1 = t2;
    t2 = t;
  }
  return t1;
}

int calculateD(int e, int phi) {
  int d = extendedEuclidean(phi, e);
  if (d < 0) {
    d += phi;
  }
  return d;
}
```

Roll No: 2103163
Batch: C32
Name: Om Shete

```
int calculateE(int phi) {
  for (int i = 2; i < phi; i++) {
    if (gcd(phi, i) == 1) {
      return i;
    }
  }
  return -1;
}

bool isPrime(int num) {
  if (num <= 1) {
    return false;
  }
  for (int i = 2; i * i <= num; i++) {
    if (num % i == 0) {
      return false;
    }
  }
  return true;
}

int main() {
  cout << "Enter the value of p: ";
  int p;
  cin >> p;

  cout << "Enter the value of q: ";
  int q;
  cin >> q;

  if (isPrime(p) && isPrime(q)) {
    int n = p * q;
    int phi = (p - 1) * (q - 1);
    int e = calculateE(phi);
    int d = calculateD(e, phi);

    cout << "phi(n): " << phi << endl;
    cout << "e: " << e << endl;
    cout << "d: " << d << endl;

    int message;
    cout << "Enter the Message: ";
```

Roll No: 2103163
Batch: C32
Name: Om Shete

```
    cin >> message;

    int cipher = encryption(e, message, n);
    if (cipher != -1) {
      cout << "Encrypted Text: " << cipher << endl;
      int decryptedMessage = decryption(d, cipher, n);
      cout << "Decrypted Text: " << decryptedMessage << endl;
    } else {
      cout << "Encryption not possible, choose big p and q." << endl;
    }
  } else {
    cout << "Both p and q should be prime." << endl;
  }

  return 0;
}
```

Results:

```
Enter the value of p: 13
Enter the value of q: 17
phi(n): 192
e: 5
d: 77
Enter the Message: 8
--------------------------------------------------
Binary Expo: 8^1
Binary Expo: 8^4
--------------------------------------------------
Encrypted Text: 60
--------------------------------------------------
Binary Expo: 60^1
Binary Expo: 60^4
Binary Expo: 60^8
Binary Expo: 60^64
--------------------------------------------------
Decrypted Text: 8
```

Roll No: 2103163
Batch: C32
Name: Om Shete

Experiment No 6

Aim: Implementation of Diffie Hellman Key exchange algorithm.

Description:

Diffie-Hellman algorithm:

The Diffie-Hellman algorithm is one of the most important algorithms used for establishing a shared secret. At the time of exchanging data over a public network, we can use the shared secret for secret communication. We use an elliptic curve for generating points and getting a secret key using the parameters.

1. We will take four variables, i.e., **P (prime), G (the primitive root of P),** and **a and b (private values)**.
2. The variables **P** and **G** both are publicly available. The sender selects a private value, either a or b, for generating a key to exchange publicly. The receiver receives the key, and that generates a secret key, after which the sender and receiver both have the same secret key to encrypt.

Step-by-Step explanation is as follows:

Alice	**Bob**
Public Keys available = P, G	Public Keys available = P, G
Private Key Selected = a	Private Key Selected = b
Key generated => x = G^a mod P	Key generated => y = G^b mod P
The exchange of generated keys takes place	
Key received = y	key received = x
Generated Secret Key => k_a = y^a mod P	Generated Secret Key => k_b = x^b mod P
Algebraically, it can be shown that- k_a = k_b	
Users now have a symmetric secret key to encrypt	

Roll No: 2103163
Batch: C32
Name: Om Shete

Example:

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.

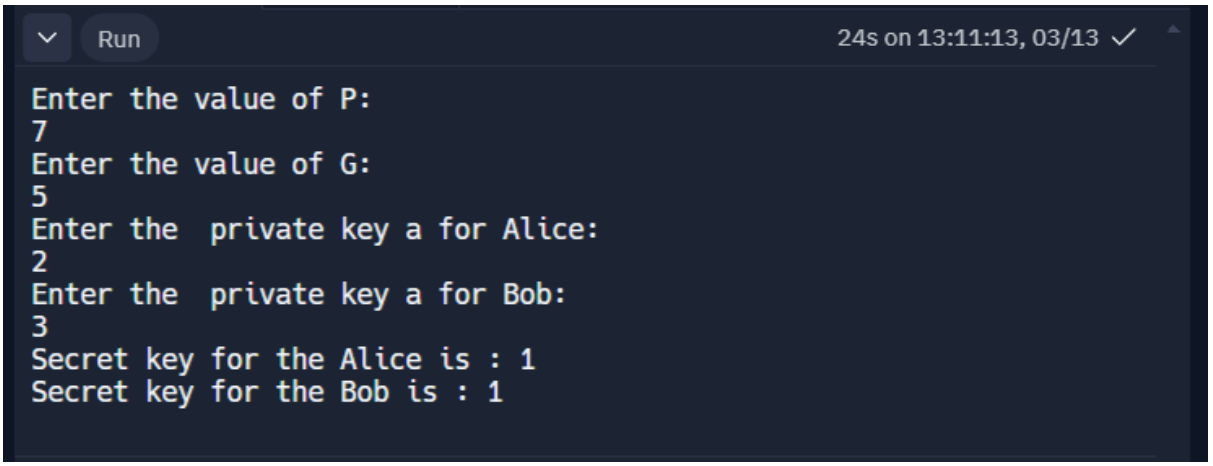
Code:

```
#include <cmath>
#include <iostream>
using namespace std;

long long int power(long long int a, long long int b, long long int P) {
  if (b == 1)
    return a;

  else
    return (((long long int)pow(a, b)) % P);
}

int main() {
  long long int P, G, x, a, y, b, ka, kb;

  cout << "Enter the value of P: " << endl;
  cin >> P;
```

Roll No: 2103163
Batch: C32
Name: Om Shete

```
  cout << "Enter the value of G: " << endl;
  cin >> G;

  cout << "Enter the  private key a for Alice: " << endl;
  cin >> a;

  x = power(G, a, P);

  cout << "Enter the  private key a for Bob: " << endl;
  cin >> b;

  y = power(G, b, P);

  ka = power(y, a, P);
  kb = power(x, b, P);
  cout << "Secret key for the Alice is : " << ka << endl;

  cout << "Secret key for the Bob is : " << kb << endl;

  return 0;
}
```

Results:



Roll No: 2103163
Batch: C32
Name: Om Shete

Roll No: 2103163
Batch: C32
Name: Om Shete

Experiment No. 7

Aim: Study the use of network reconnaissance tools and apply the following: WHOIS, dig, traceroute, nslookup

Description:

Reconnaissance (or simply Recon) is the initial phase in the Pen Testing process. The goal of recon is to gather as much information about the target as you can. More the information, the more beneficial it will be for further phases of pen testing. Most new learners underestimate this phase and ignore it but recon is the most important phase of pen testing. Your point of view for the digital world changes if you completely understand this process. Learning to successfully conduct the recon process is a valuable skill for anyone. There are two strategies of recon i.e., Active and Passive reconnaissance.

- Active Recon**:** It means interacting directly with a target to gather information. This is not recommended because it violates the rule of "hiding traces" in pen testing.
- Passive Recon**:** It means gathering information about the target using vast information present on the internet. In it, we aren't interacting directly with the target so there is no fear of recording or logging of our activity by target.

WHOIS:

Whois is a command-line utility used in Linux systems to retrieve information about domain names, IP addresses, and network devices registered with the Internet Corporation for Assigned Names and Numbers (ICANN). The data received by Whois consists of the name and contact information of the domain or IP address owner, the registration and expiration date, the domain registrar, and the server information. Whois command can be very useful for network administrators, web developers, and security professionals for achieving various tasks like checking network connectivity or troubleshooting. In this article, we will go through the usage of the Whois command on Linux (Ubuntu system).

The whois command is a useful tool for obtaining information about domain names, IP Addresses, and network devices registered with ICANN. Whois command is a simple and powerful tool that can be useful for network administration, web development, and security tasks, and every Linux user should be familiar with its usage. In this article, we have gone through the installation of the whois command and its usage in the form of examples.

Dig:

'dig' command stands for Domain Information Groper. It is used for retrieving information about DNS name servers. It is basically used by network administrators. It is used for verifying and troubleshooting DNS problems and to perform DNS lookups. Dig command replaces older tools such as nslookup and the host.

In case of Debian/Ubuntu

```
$sudo apt-get install dnsutils
```

In case of CentOS/RedHat

```
$sudo yum install bind-utils
```

Nslookup:

Nslookup (stands for “Name Server Lookup”) is a useful command for getting information from the DNS server. It is a network administration tool for querying the Domain Name System (DNS) to obtain domain name or IP address mapping or any other specific DNS record. It is also used to troubleshoot DNS-related problems.

```
nslookup [option] [hosts]
```

Traceroute:

A traceroute provides a map of how data on the internet travels from its source to its destination. When you connect with a website, the data you get must travel across multiple devices and networks along the way, particularly routers.

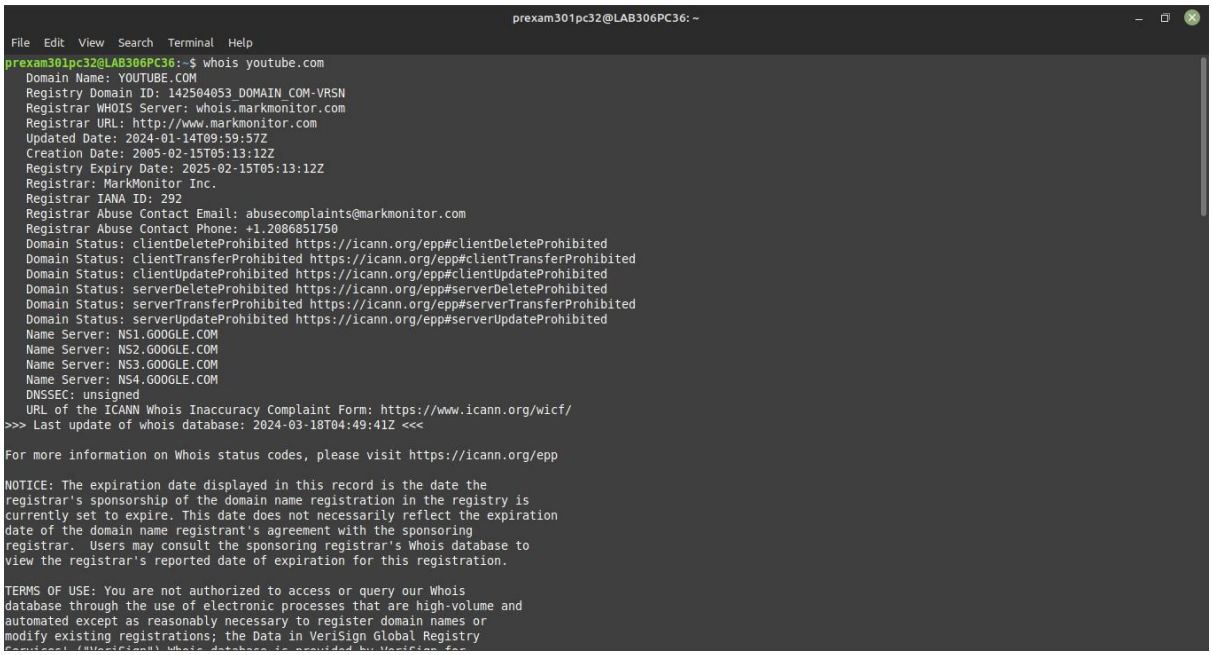
A traceroute plays a different role than other diagnostic tools, such as packet capture, which analyzes data. Traceroute differs in that it examines how the data moves through the internet. Similarly, you can use Domain Name System time to live (DNS TTL) for tracerouting, but DNS TTL addresses the time needed to cache a query and does not follow the data path between routers.

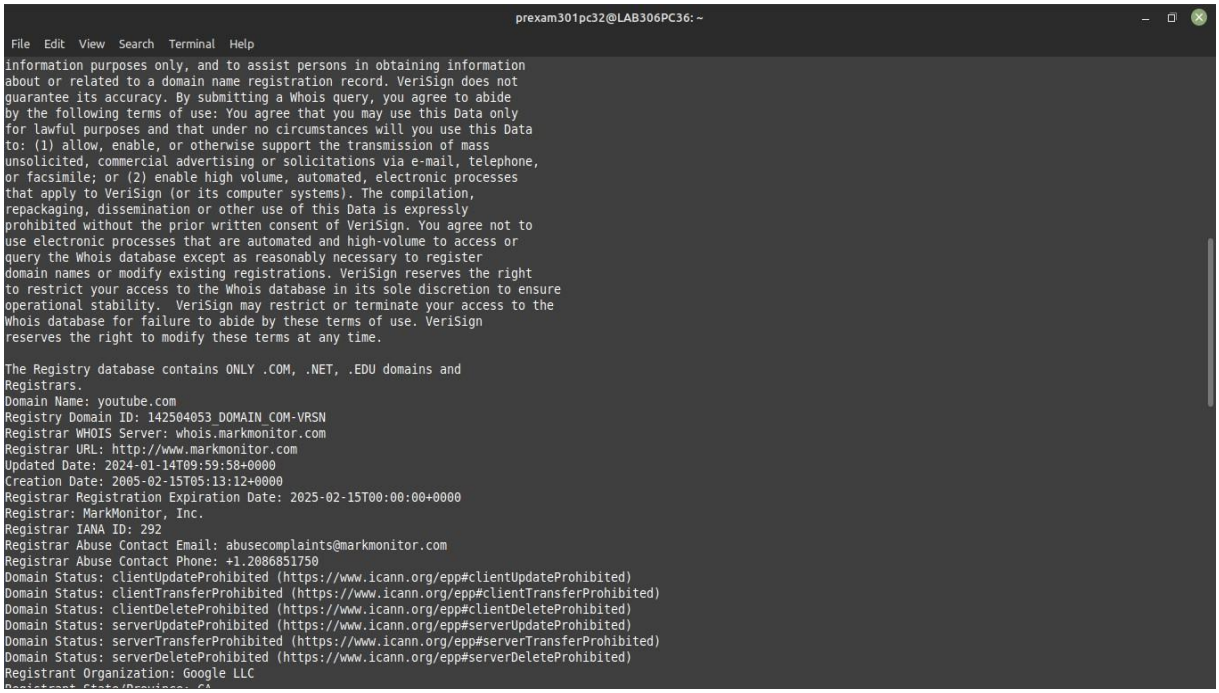
```
traceroute [options]  host_Address [pathlength]
```

Roll No: 2103163
Batch: C32
Name: Om Shete

Results:

1.Whois:





Roll No: 2103163
Batch: C32
Name: Om Shete

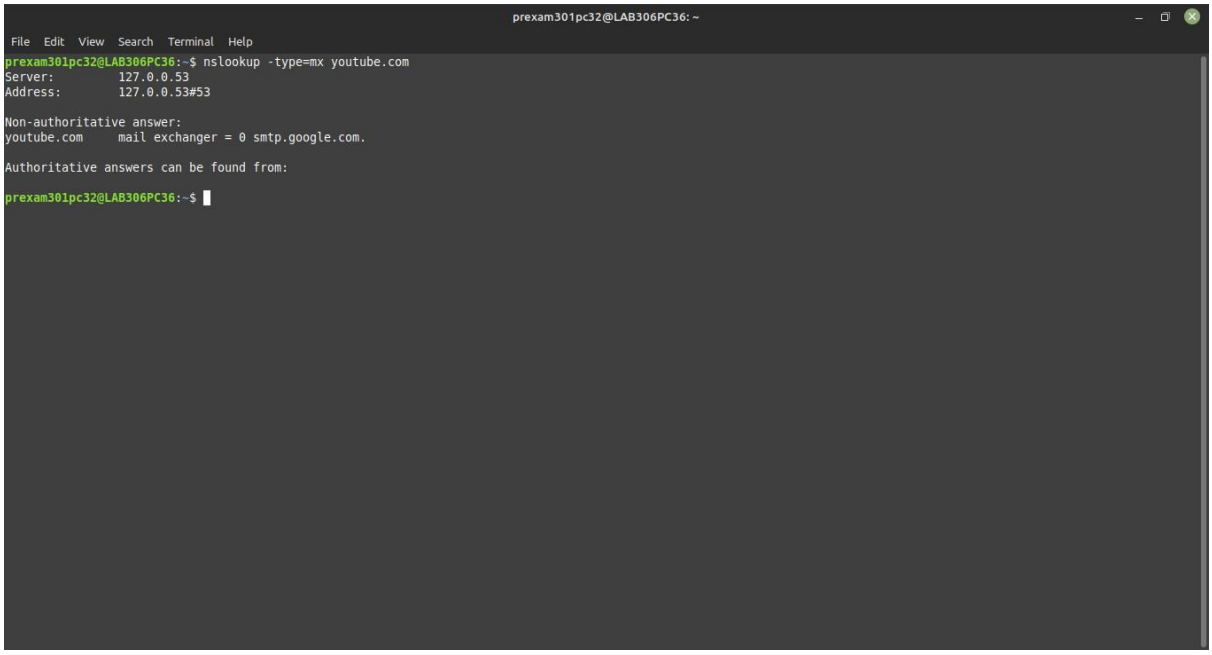
```
prexam301pc32@LAB306PC36: ~
File  Edit  View  Search  Terminal  Help
For more information on WHOIS status codes, please visit:
  https://www.icann.org/resources/pages/epp-status-codes

If you wish to contact this domain's Registrant, Administrative, or Technical
contact, and such email address is not visible above, you may do so via our web
form, pursuant to ICANN's Temporary Specification. To verify that you are not a
robot, please enter your email address to receive a link to a page that
facilitates email communication with the relevant contact(s).

Web-based WHOIS:
  https://domains.markmonitor.com/whois

If you have a legitimate interest in viewing the non-public WHOIS details, send
your request and the reasons for your request to whoisrequest@markmonitor.com
and specify the domain name in the subject line. We will review that request and
may ask for supporting documentation and explanation.

The data in MarkMonitor's WHOIS database is provided for information purposes,
and to assist persons in obtaining information about or related to a domain
name's registration record. While MarkMonitor believes the data to be accurate,
the data is provided "as is" with no guarantee or warranties regarding its
accuracy.

By submitting a WHOIS query, you agree that you will use this data only for
lawful purposes and that, under no circumstances will you use this data to:
  (1) allow, enable, or otherwise support the transmission by email, telephone,
or facsimile of mass, unsolicited, commercial advertising, or spam; or
  (2) enable high volume, automated, or electronic processes that send queries,
data, or email to MarkMonitor (or its systems) or the domain name contacts (or
its systems).

MarkMonitor reserves the right to modify these terms at any time.

By submitting this query, you agree to abide by this policy.

MarkMonitor Domain Management(TM)
Protecting companies and consumers in a digital world.

Visit MarkMonitor at https://www.markmonitor.com
```

dig:

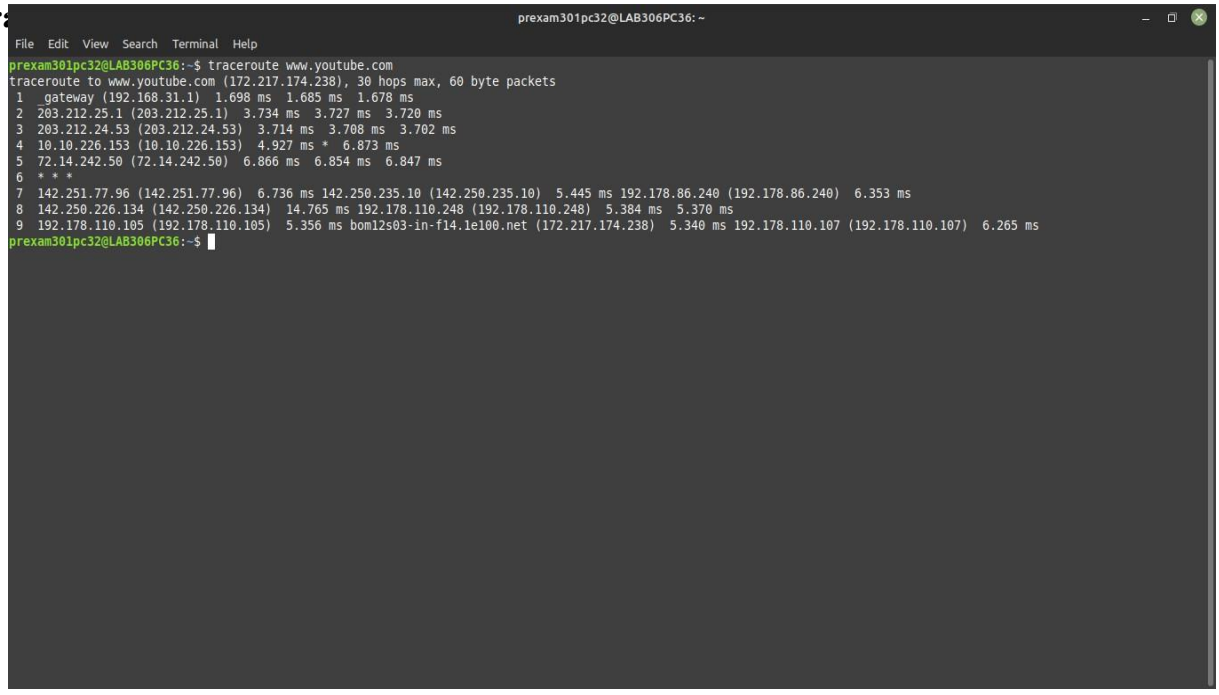
```
prexam301pc32@LAB306PC36: ~
File  Edit  View  Search  Terminal  Help
prexam301pc32@LAB306PC36:~$ dig www.youtube.com

; <<>> DiG 9.18.12-0ubuntu0.22.04.1-Ubuntu <<>> www.youtube.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59330
;; flags: qr rd ra; QUERY: 1, ANSWER: 17, AUTHORITY: 0, ADDITIONAL: 1

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

;; ANSWER SECTION:
www.youtube.com.        238     IN      CNAME   youtube-ui.l.google.com.
youtube-ui.l.google.com. 238    IN      A       142.250.199.142
youtube-ui.l.google.com. 238    IN      A       142.250.199.174
youtube-ui.l.google.com. 238    IN      A       142.251.42.110
youtube-ui.l.google.com. 238    IN      A       142.250.70.46
youtube-ui.l.google.com. 238    IN      A       142.250.70.78
youtube-ui.l.google.com. 238    IN      A       142.250.70.110
youtube-ui.l.google.com. 238    IN      A       142.250.71.110
youtube-ui.l.google.com. 238    IN      A       172.217.167.174
youtube-ui.l.google.com. 238    IN      A       172.217.174.238
youtube-ui.l.google.com. 238    IN      A       216.58.203.14
youtube-ui.l.google.com. 238    IN      A       172.217.160.174
youtube-ui.l.google.com. 238    IN      A       142.250.183.14
youtube-ui.l.google.com. 238    IN      A       142.250.183.142
youtube-ui.l.google.com. 238    IN      A       142.250.183.174
youtube-ui.l.google.com. 238    IN      A       142.250.183.206
youtube-ui.l.google.com. 238    IN      A       142.250.66.14

;; Query time: 8 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Mon Mar 18 10:22:40 IST 2024
;; MSG SIZE  rcvd: 334

prexam301pc32@LAB306PC36:~$
```

Roll No: 2103163
Batch: C32
Name: Om Shete

nslookup:



tr



Roll No: 2103163
Batch: C32
Name: Om Shete

Experiment No 8

Aim: Study and implementation of packet sniffer tool: wireshark

Description:

What is Wireshark?

Wireshark is a widely used, open source network analyzer that can capture and display real-time details of network traffic. It is particularly useful for troubleshooting network issues, analyzing network protocols and ensuring network security.

Networks must be monitored to ensure smooth operations and security. Popular with academic institutions, government agencies, corporations and nonprofits, Wireshark is one such tool that can offer an in-depth view into network activities, diagnose network performance issues or identify potential security threats.

Key features of Wireshark

Wireshark seeks to simplify and enhance the process of network traffic analysis. Each function is designed to offer unique insights and control over network activities. Here are some of its core features:

- **Packet capture (PCAP).** Converts network traffic into a human-readable format, making it easier to understand and diagnose concerns.
- **Real-time analysis.** Provides a live view of network traffic, offering immediate insights into ongoing network activities.
- **Filtering capabilities.** Enables users to focus on specific types of network traffic, making analysis more efficient and targeted.
- **Graphical user interface (GUI).** Designed for ease of use, ensures that both beginners and experts can navigate and analyze data effectively

Common uses for Wireshark

Wireshark can be used to examine the details of traffic at a variety of levels, ranging from connection-level information to the bits constituting a single packet.

Roll No: 2103163
Batch: C32
Name: Om Shete

PCAP can provide a network administrator with information about individual packets, including transmit time, source, destination, protocol type and header data. This information can be useful for evaluating security events and troubleshooting network security device issues.

Wireshark's capabilities extend beyond just monitoring to address other network administration tasks:

- **Network troubleshooting.** Pinpoints and resolves network issues with the comprehensive data Wireshark provides.
- **Security analysis.** Detects and analyzes potential security threats in the network.
- **Performance analysis.** Monitors and optimizes network performance to ensure smooth operations.
- **Protocol analysis.** Gains insights into the behavior of individual protocols within the network.

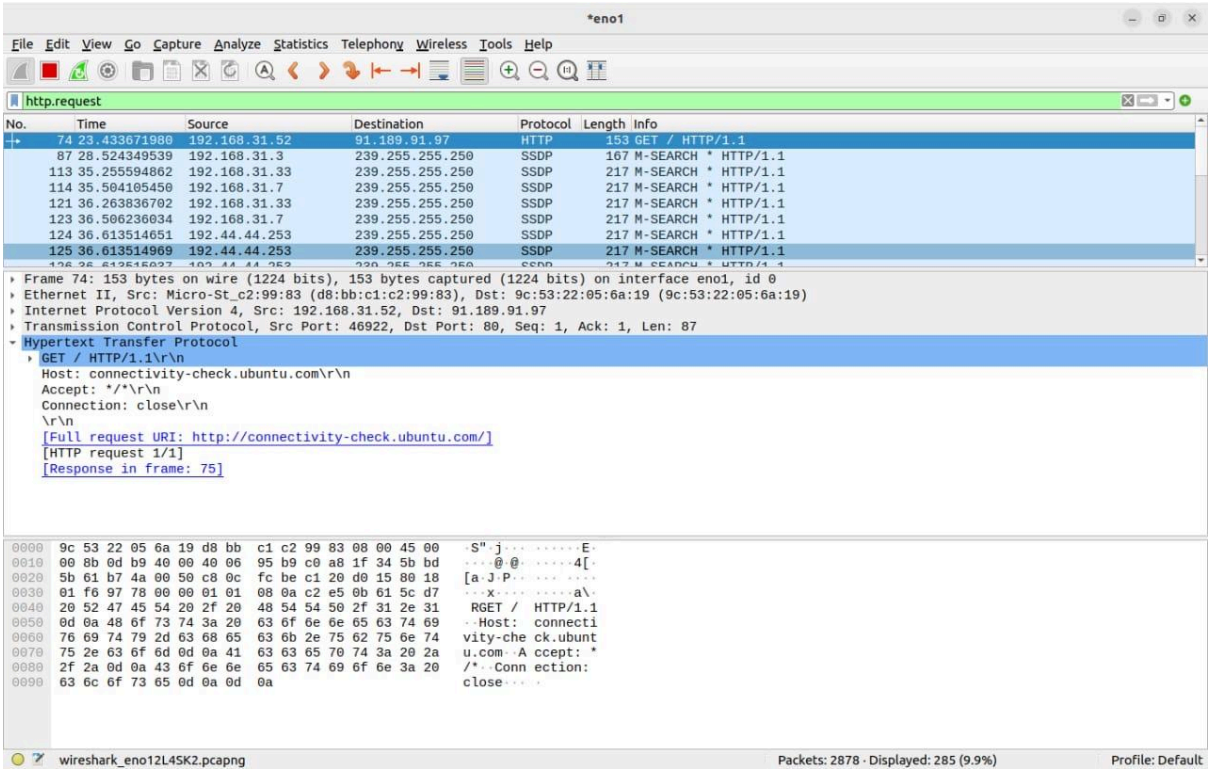
Supported file formats in Wireshark

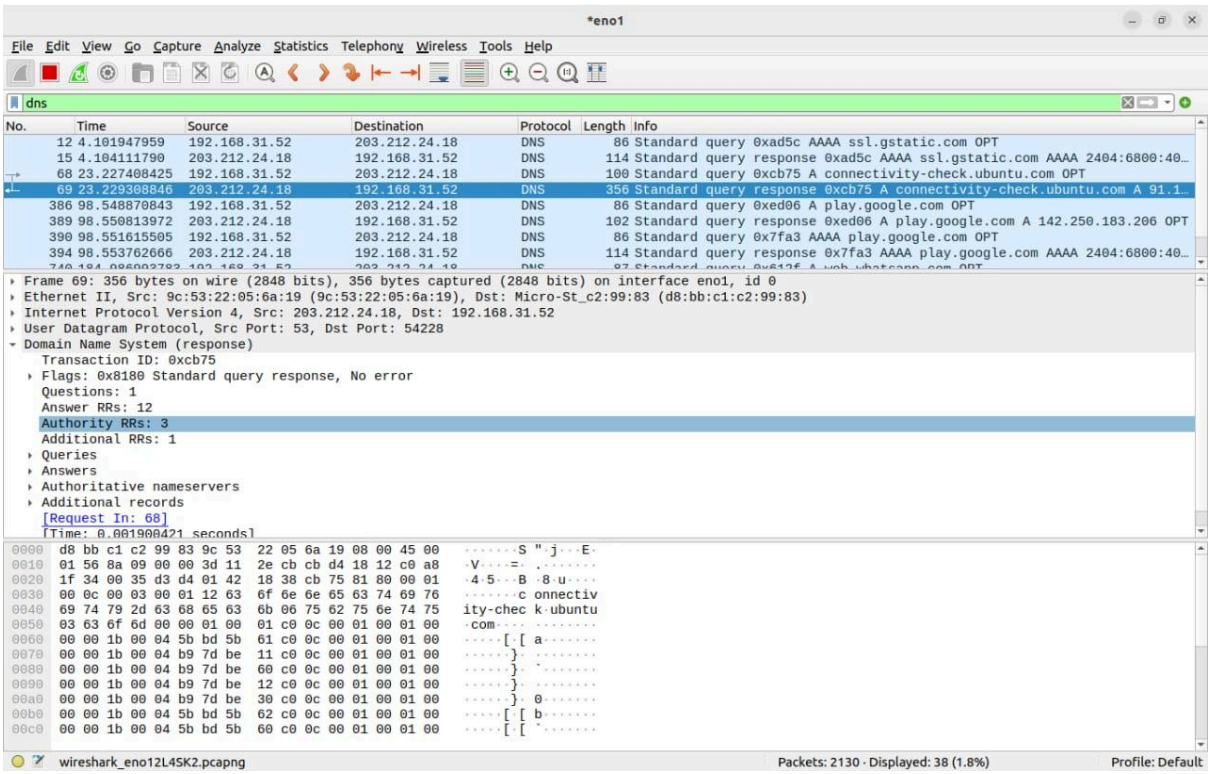
Wireshark is known for its versatility and the wide array of file formats it supports. The primary file format used by Wireshark to save PCAPs is PcapNG, which stands for Packet Capture Next Generation. This format is recognized for its flexibility in capturing and storing packet data.

To support interoperability with third-party protocol analyzers, Wireshark also has the ability to read and save packet data in other file formats, including CAP and PCAP.

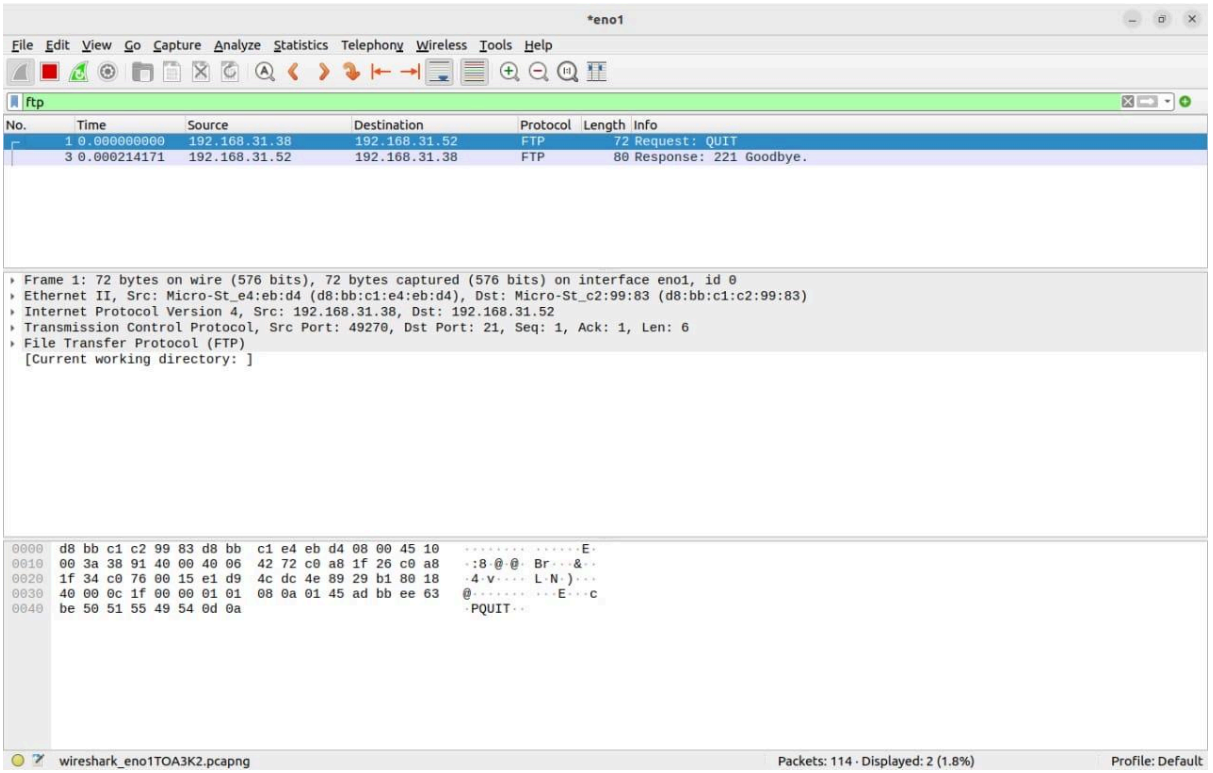
Roll No: 2103163
Batch: C32
Name: Om Shete

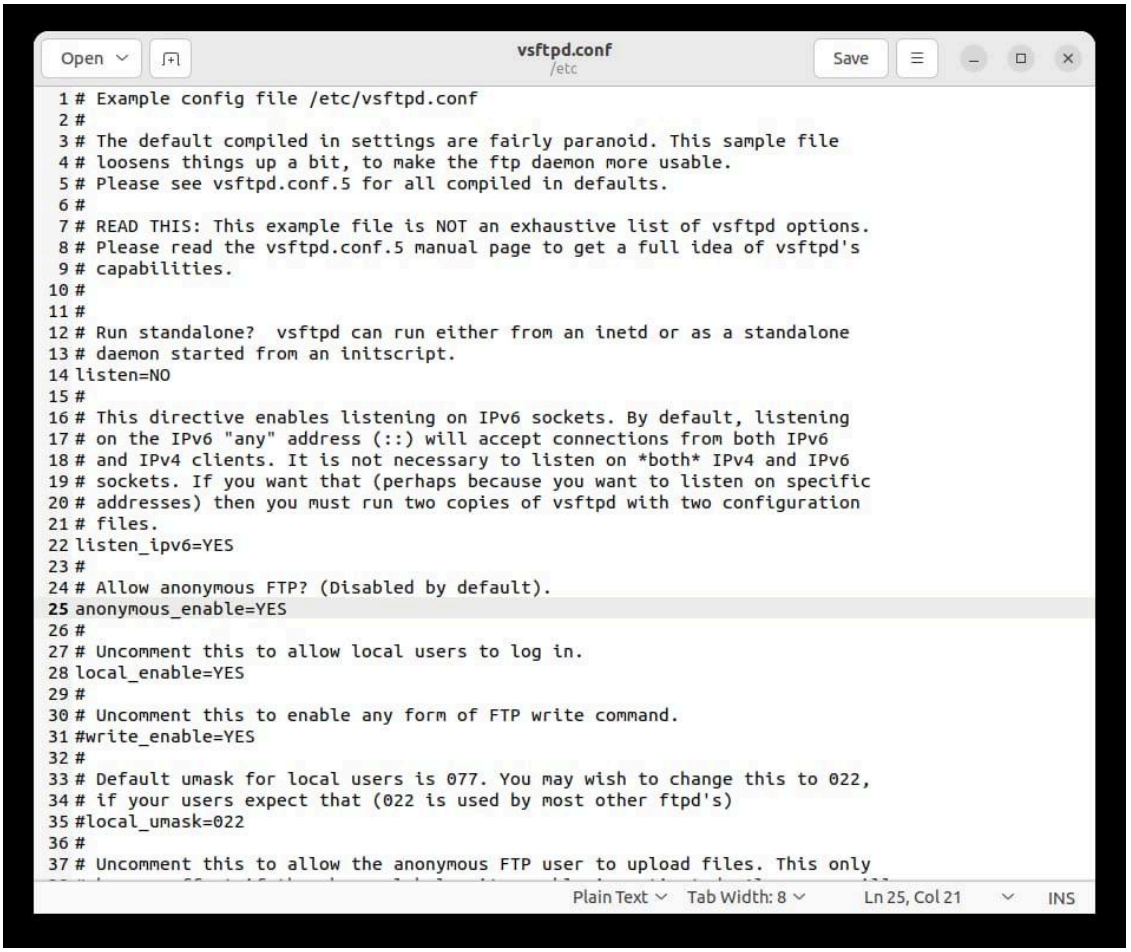
Results:



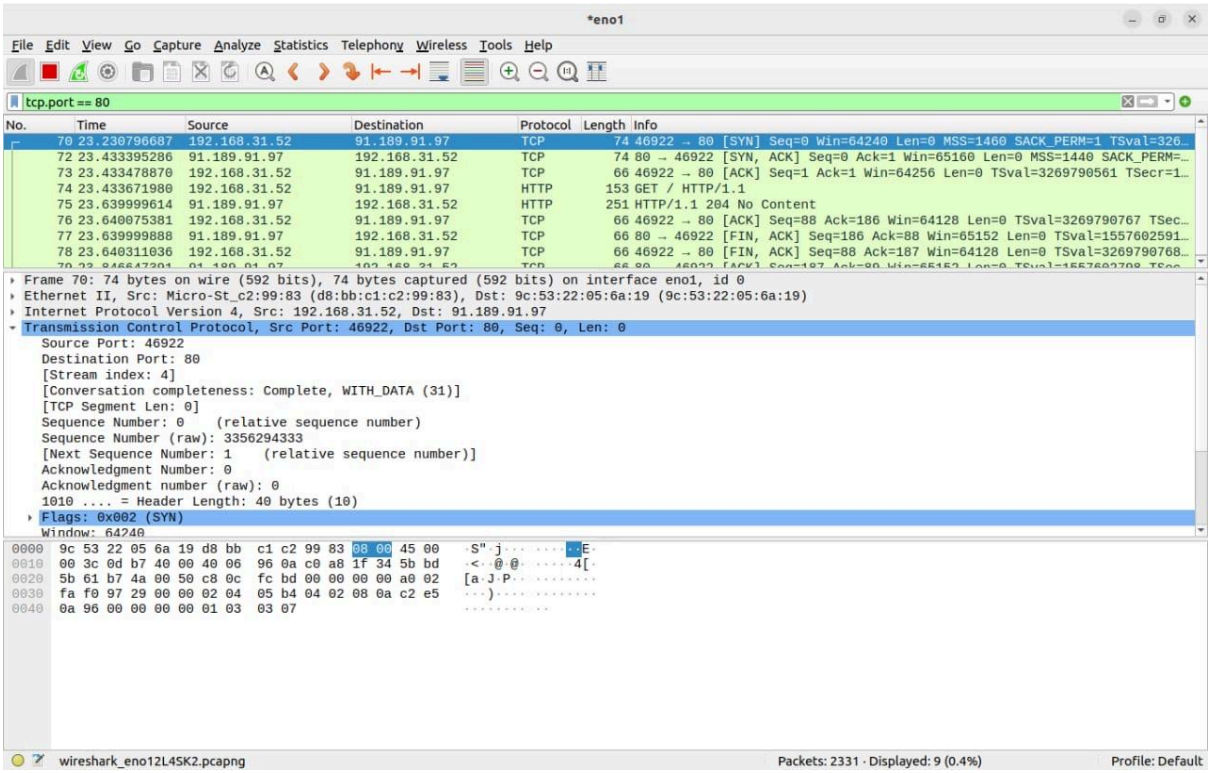


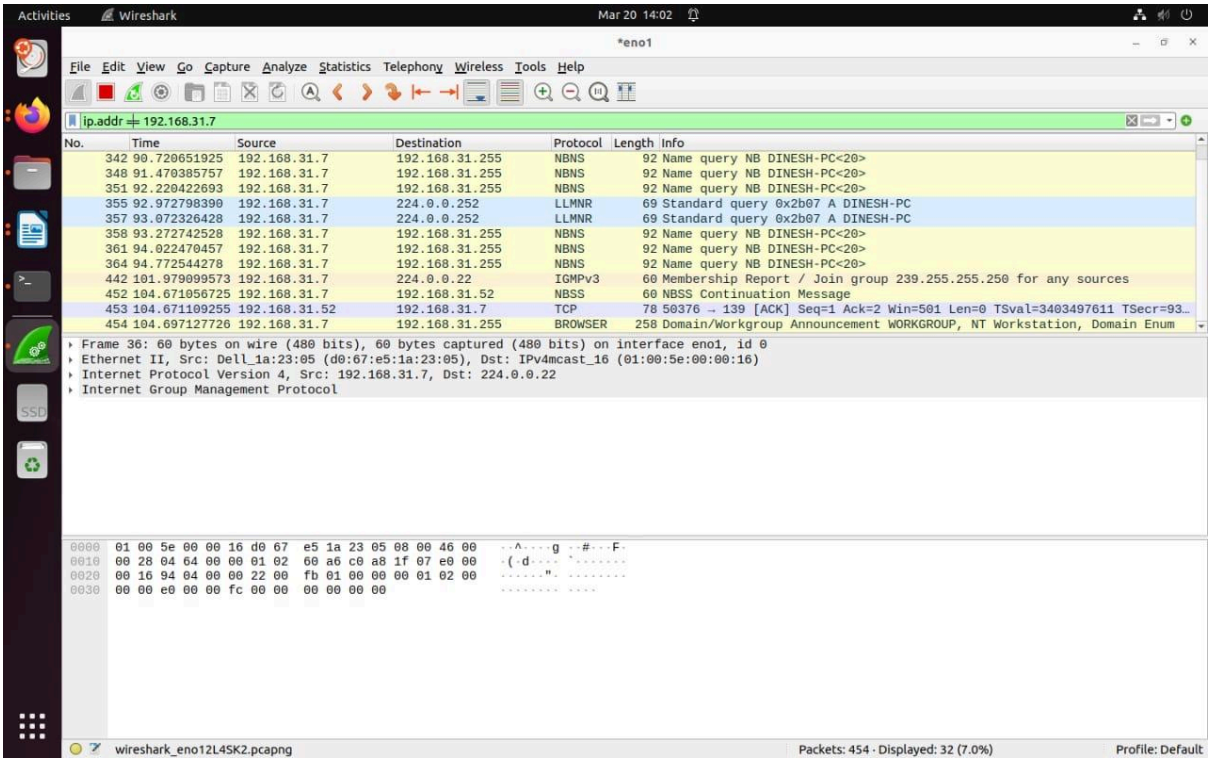
Roll No: 2103163
Batch: C32
Name: Om Shete



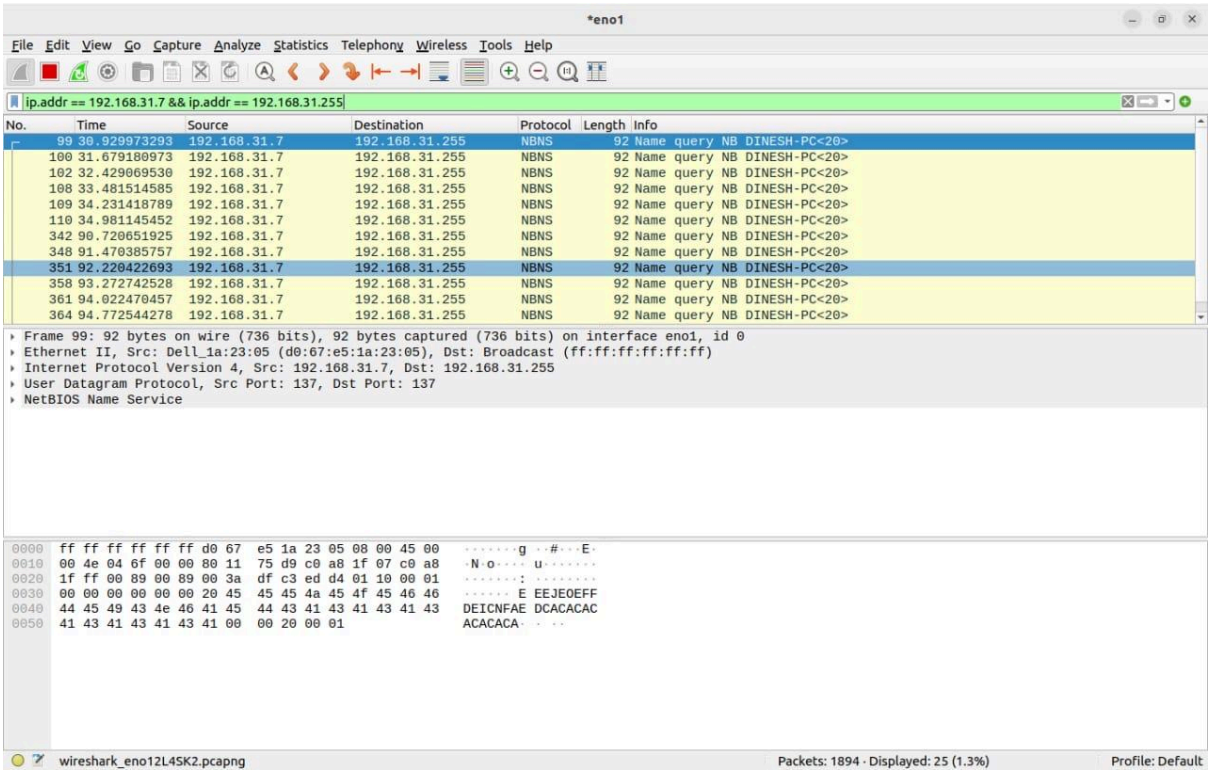


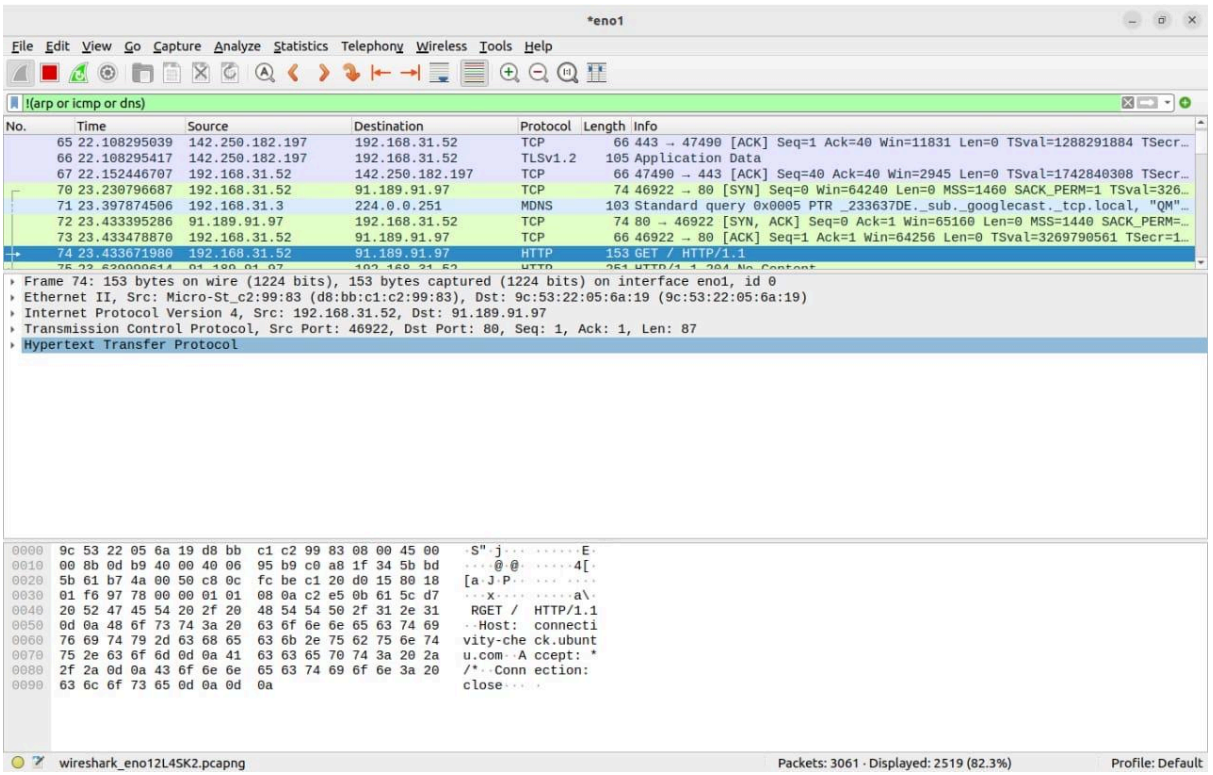
Roll No: 2103163
Batch: C32
Name: Om Shete



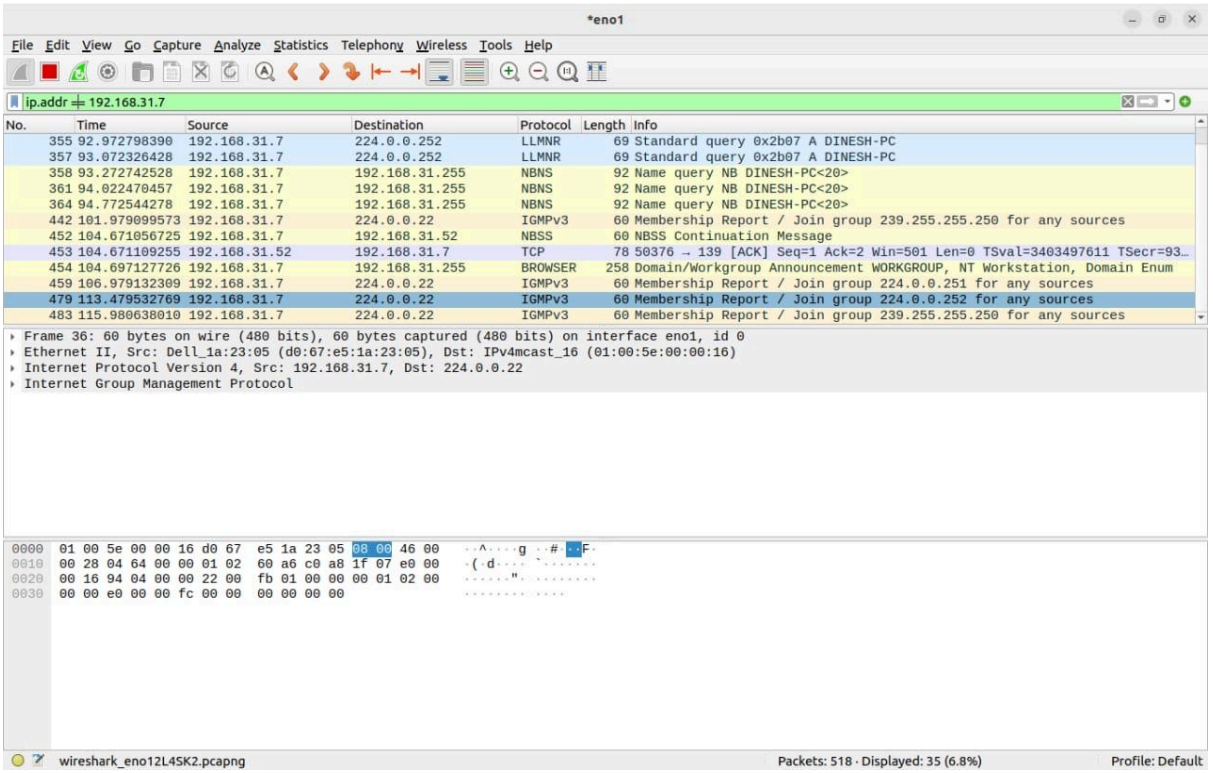


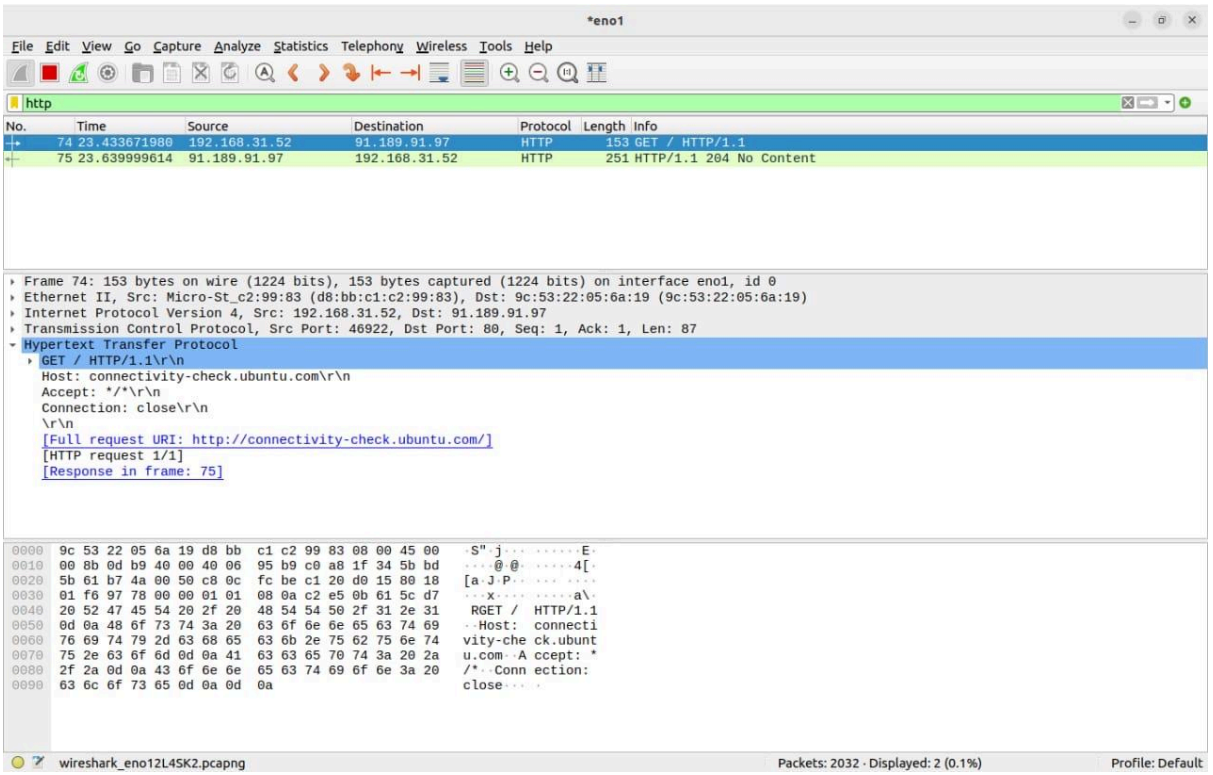
Roll No: 2103163
Batch: C32
Name: Om Shete





Roll No: 2103163
Batch: C32
Name: Om Shete





Roll No: 2103163
Batch: C32
Name: Om Shete

Experiment No 9

Aim: Design of personal firewall using iptables.

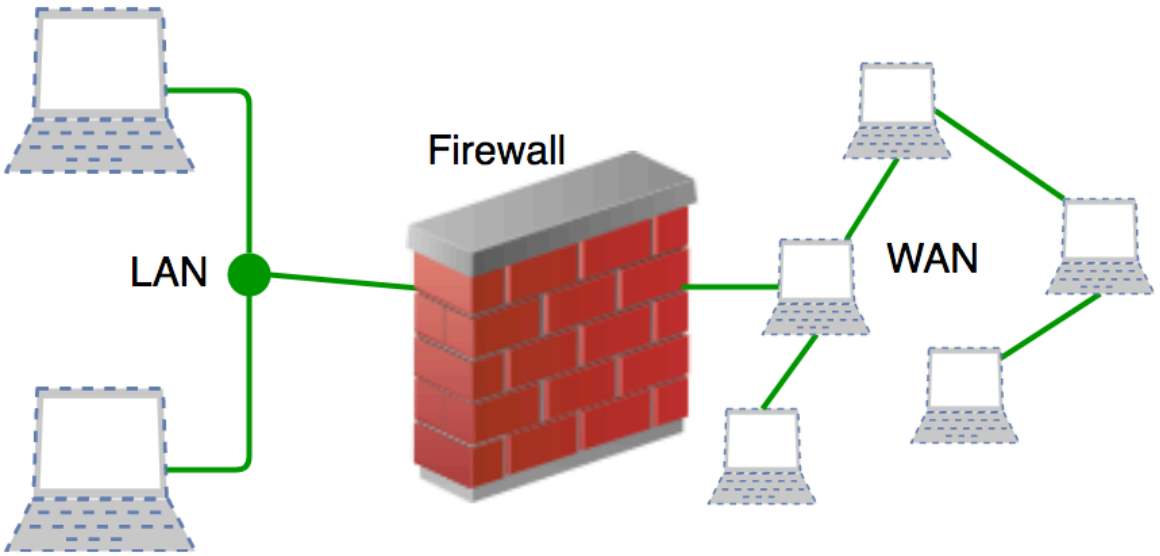
Description:

A firewall is a network security device that prevents unauthorized access to a network. It monitors both incoming and outgoing traffic using a predefined set of security to detect and prevent threats.

What is Firewall?
A firewall is a network security device, either hardware or software-based, which monitors all incoming and outgoing traffic and based on a defined set of security rules accepts, rejects, or drops that specific traffic.

Accept: allow the traffic
Reject: block the traffic but reply with an “unreachable error”
Drop: block the traffic with no reply

A firewall is a type of network security device that filters incoming and outgoing network traffic with security policies that have previously been set up inside an organization. A firewall is essentially the wall that separates a private internal network from the open Internet at its very basic level.



Designing a personal firewall using iptables involves creating rules to control incoming and outgoing traffic on a Linux system. Below is a basic design outline for a personal firewall using iptables:

Roll No: 2103163
Batch: C32
Name: Om Shete

Define Default Policies: Set default policies for INPUT, OUTPUT, and FORWARD chains to DROP or REJECT to ensure that traffic is only allowed if explicitly permitted.

sudo iptables -P INPUT DROP
sudo iptables -P FORWARD DROP
sudo iptables -P OUTPUT ACCEPT

Allow Established Connections: Allow traffic related to established connections. This ensures that responses to outgoing connections are allowed back in.

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

Allow Loopback Interface: Allow traffic on the loopback interface, which is essential for internal communication.

sudo iptables -A INPUT -i lo -j ACCEPT
sudo iptables -A OUTPUT -o lo -j ACCEPT

Allow Specific Incoming Connections: Allow incoming connections for specific services or ports you want to make accessible from outside.

sudo iptables -A INPUT -p tcp --dport <port_number> -j ACCEPT
sudo iptables -A INPUT -p udp --dport <port_number> -j ACCEPT

Allow Outgoing Connections: Allow outgoing connections from your system.

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

Logging: Optionally, you can log dropped packets to track potential security issues.

sudo iptables -A INPUT -j LOG --log-prefix "iptables: INPUT DROP: "
sudo iptables -A OUTPUT -j LOG --log-prefix "iptables: OUTPUT DROP: "

Save and Apply Rules: Once you have configured the rules, save them and ensure they are applied on system reboot.
sudo iptables-save > /etc/iptables/rules.v4
sudo systemctl enable netfilter-persistent
sudo systemctl start netfilter-persistent

Roll No: 2103163
Batch: C32
Name: Om Shete

Results:

```
  libip4tc2 libip6tc2 libxtables12
Suggested packages:
  firewalld
The following packages will be upgraded:
  iptables libip4tc2 libip6tc2 libxtables12
4 upgraded, 0 newly installed, 0 to remove and 74 not upgraded.
Need to get 527 kB of archives.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 iptables amd64 1.8.7-1ubuntu5
Get:2 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libxtables12 amd64 1.8.7-1ubu
Get:3 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libip6tc2 amd64 1.8.7-1ubuntu
Get:4 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libip4tc2 amd64 1.8.7-1ubuntu
Fetched 527 kB in 2s (276 kB/s)
(Reading database ... 310344 files and directories currently installed.)
Preparing to unpack .../iptables_1.8.7-1ubuntu5.2_amd64.deb ...
Unpacking iptables (1.8.7-1ubuntu5.2) over (1.8.7-1ubuntu5.1) ...
Preparing to unpack .../libxtables12_1.8.7-1ubuntu5.2_amd64.deb ...
Unpacking libxtables12:amd64 (1.8.7-1ubuntu5.2) over (1.8.7-1ubuntu5.1) ...
Preparing to unpack .../libip6tc2_1.8.7-1ubuntu5.2_amd64.deb ...
Unpacking libip6tc2:amd64 (1.8.7-1ubuntu5.2) over (1.8.7-1ubuntu5.1) ...
Preparing to unpack .../libip4tc2_1.8.7-1ubuntu5.2_amd64.deb ...
Unpacking libip4tc2:amd64 (1.8.7-1ubuntu5.2) over (1.8.7-1ubuntu5.1) ...
Setting up libip4tc2:amd64 (1.8.7-1ubuntu5.2) ...
Setting up libip6tc2:amd64 (1.8.7-1ubuntu5.2) ...
Setting up libxtables12:amd64 (1.8.7-1ubuntu5.2) ...
Setting up iptables (1.8.7-1ubuntu5.2) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.6) ...
root@admini-OptiPlex-3050:/home/admini# ipconfig
Command 'ipconfig' not found, did you mean:
  command 'ifconfig' from deb net-tools (1.60+git20181103.0eebece-1ubuntu5)
  command 'iwconfig' from deb wireless-tools (30~pre9-13.1ubuntu4)
  command 'iconfig' from deb ipmiutil (3.1.8-1)
Try: apt install <deb name>
root@admini-OptiPlex-3050:/home/admini# ifconfig
enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.161  netmask 255.255.252.0  broadcast 192.168.3.255
```

```
enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.161  netmask 255.255.252.0  broadcast 192.168.3.255
        inet6 fe80::d60c:de4c:a36f:8c3b  prefixlen 64  scopeid 0x20<link>
        ether d8:9e:f3:2b:0d:92  txqueuelen 1000  (Ethernet)
        RX packets 4563  bytes 5598144 (5.5 MB)
        RX errors 0  dropped 1  overruns 0  frame 0
        TX packets 2097  bytes 217870 (217.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 339  bytes 45851 (45.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 339  bytes 45851 (45.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp3s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 00:21:6b:fe:69:24  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@admini-OptiPlex-3050:/home/admini# ipaddress
Command 'ipaddress' not found, did you mean:
  command 'ip-address' from snap ip-address (1.0.0)
See 'snap info <snapname>' for additional versions.
root@admini-OptiPlex-3050:/home/admini# ip-address
Command 'ip-address' not found, but can be installed with:
snap install ip-address
root@admini-OptiPlex-3050:/home/admini# sudo-apt install ip-address
sudo-apt: command not found
root@admini-OptiPlex-3050:/home/admini# snap install ip-address
ip-address 1.0.0 from Daniel Dewberry (dwd-daniel) installed
root@admini-OptiPlex-3050:/home/admini# ip-address
ip          : 203.212.24.36
```

Roll No: 2103163
Batch: C32
Name: Om Shete

```
admini@admini-OptiPlex-3050:~$ sudo su root
[sudo] password for admini:
root@admini-OptiPlex-3050:/home/admini# apt-get update
Hit:1 https://packages.microsoft.com/repos/code stable InRelease
Ign:2 https://cloud.r-project.org/bin/linux/ubuntu jammy InRelease
Hit:3 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:4 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Hit:5 https://ppa.launchpadcontent.net/wireshark-dev/stable/ubuntu jammy InRelease
Hit:6 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
Err:7 https://cloud.r-project.org/bin/linux/ubuntu jammy Release
  404  Not Found [IP: 108.158.61.26 443]
Hit:8 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
E: The repository 'https://cloud.r-project.org/bin/linux/ubuntu jammy Release' does not have a R
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
root@admini-OptiPlex-3050:/home/admini# apt-get iptables
E: Invalid operation iptables
root@admini-OptiPlex-3050:/home/admini# apt-get install iptables
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi i965-va-driver
  intel-media-va-driver libaacs0 libaom3 libass9 libavcodec58 libavformat58
  libavutil56 libbdplus0 libbluray2 libbs2b0 libchromaprint1 libcodec2-1.0
  libdav1d5 libflashrom1 libflite1 libftdi1-2 libgme0 libgsm1
  libgstreamer-plugins-bad1.0-0 libigdgmm12 liblilv-0-0 libllvm13 libmfx1
  libmysofa1 libnorm1 libopenmpt0 libpgm-5.3-0 libpostproc55 librabbitmq4
  librubberband2 libserd-0-0 libshine3 libsord-0-0 libsratom-0-0
  libsrt1.4-gnutls libswresample3 libswscale5 libudfread0 libva-drm2
  libva-wayland2 libva-x11-2 libva2 libvdpau1 libvidstab1.1 libx265-199
  libxvidcore4 libzimg2 libzmq5 libzvbi-common libzvbi0 mesa-va-drivers
  mesa-vdpau-drivers pocketsphinx-en-us va-driver-all vdpau-driver-all
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libip4tc2 libip6tc2 libxtables12
Suggested packages:
```

Roll No: 2103163
Batch: C32
Name: Om Shete

```
root@admini-OptiPlex-3050:/home/admini# ip-address
ip         : 203.212.24.36
country    : India
country_iso: IN
country_eu : False
region_name: Maharashtra
zip_code   : 400070
city       : Mumbai
latitude   : 19.0748
longitude  : 72.8856
time_zone  : Asia/Kolkata
asn        : AS45243
asn_org    : FX Wireless Technology Solutions Pvt. Ltd.
root@admini-OptiPlex-3050:/home/admini# ^C
root@admini-OptiPlex-3050:/home/admini# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@admini-OptiPlex-3050:/home/admini# iptables -I OUTPUT -s 203.212.24.36
root@admini-OptiPlex-3050:/home/admini# iptables -I OUTPUT -s 203.212.24.36 -d 203.212.24
root@admini-OptiPlex-3050:/home/admini# ifconfig
enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.161  netmask 255.255.252.0  broadcast 192.168.3.255
        inet6 fe80::d60c:de4c:a36f:8c3b  prefixlen 64  scopeid 0x20<link>
        ether d8:9e:f3:2b:0d:92  txqueuelen 1000  (Ethernet)
        RX packets 8657  bytes 11518308 (11.5 MB)
        RX errors 0  dropped 1  overruns 0  frame 0
        TX packets 4196  bytes 406880 (406.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
```

```
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 359  bytes 48217 (48.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp3s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 00:21:6b:fe:69:24  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@admini-OptiPlex-3050:/home/admini# ^C
root@admini-OptiPlex-3050:/home/admini# iptables -I OUTPUT -s 192.168.1.161 -d 192.168.1.193
root@admini-OptiPlex-3050:/home/admini# iptables -I OUTPUT -s 192.168.1.161 -d 192.168.1.193 -p icmp -j DROP
root@admini-OptiPlex-3050:/home/admini# ping 192.168.1.193
PING 192.168.1.193 (192.168.1.193) 56(84) bytes of data.
^C
--- 192.168.1.193 ping statistics ---
121 packets transmitted, 0 received, 100% packet loss, time 122887ms

root@admini-OptiPlex-3050:/home/admini# iptables -I OUTPUT -s 192.168.1.161 -d 192.168.1.193 -p icmp -j ACCEPT
root@admini-OptiPlex-3050:/home/admini# iptables -I OUTPUT -s 192.168.1.161 -d 192.168.1.193 -p icmp -j REJECT
root@admini-OptiPlex-3050:/home/admini# iptables -I OUTPUT -s 192.168.1.161 -d 192.168.1.193 -p icmp -j ACCEPT
root@admini-OptiPlex-3050:/home/admini# iptables -F
root@admini-OptiPlex-3050:/home/admini# iptables -t filter -I OUTPUT -m string --string whatsapp.com
iptables v1.8.7 (nf_tables): string: option "--algo" must be specified

Try `iptables -h' or 'iptables --help' for more information.
root@admini-OptiPlex-3050:/home/admini# iptables -t filter -I OUTPUT -m string --string whatsapp.com -j REJECT
iptables v1.8.7 (nf_tables): string: option "--algo" must be specified

Try `iptables -h' or 'iptables --help' for more information.
root@admini-OptiPlex-3050:/home/admini# iptables -t filter -I OUTPUT -m string --string google.com -j REJECT --algo kmp
root@admini-OptiPlex-3050:/home/admini# iptables -t filter -I OUTPUT -m string --string google.com -j ACCEPT --algo kmp
root@admini-OptiPlex-3050:/home/admini# iptables -I OUTPUT -s 192.168.1.161 -d 192.168.1.193 -p icmp -j ACCEPT
```

Roll No: 2103163
Batch: C32
Name: Om Shete

Experiment No 10

Aim: Simulate Buffer overflow attack using Splint

Description:

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

There are main two types

- Stack based attacks
- Heap based attacks

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

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

Problems detected by Splint include:

- Dereferencing a possibly null pointer
- Using possibly undefined storage or returning storage that is not properly defined
- Type mismatches, with greater precision and flexibility than provided by C compilers
- Violations of information hiding
- Memory management errors including uses of dangling references and memory leaks

• Dangerous aliasing

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

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

• Buffer overflow vulnerabilities

• Dangerous macro implementations or invocations

• Violations of customized naming conventions

Steps :

1. Installation

$ sudo apt-get install splint

2. Checking Vulnerability

$ splint program1.c

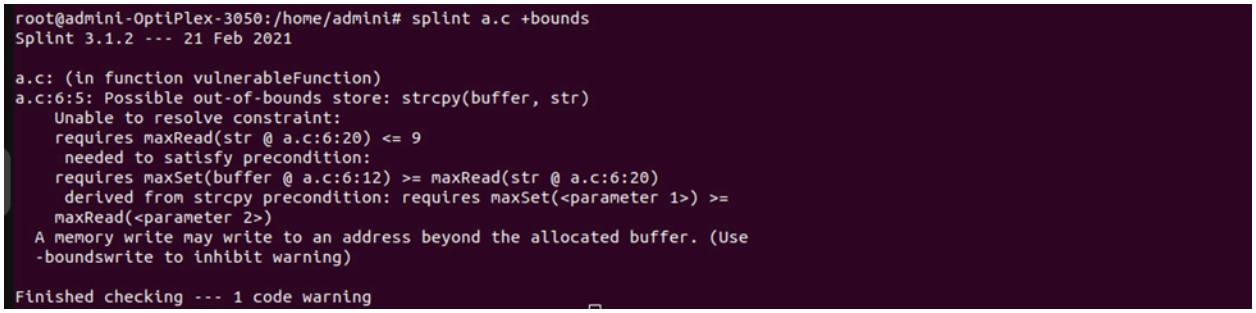
Implementation:

Program 1:

```
#include <stdio.h>

#include <string.h>


void vulnerableFunction(char *str) {

        char buffer[10];
```

```
strcpy(buffer, str); // Copies the input str to buffer without checking its length
}

int main() {

    char largeString[] = "This string is way too long for the buffer";
    vulnerableFunction(largeString);
    printf("Returned normally from vulnerableFunction\n");
    return 0;
}
```

Output:



Program 2:

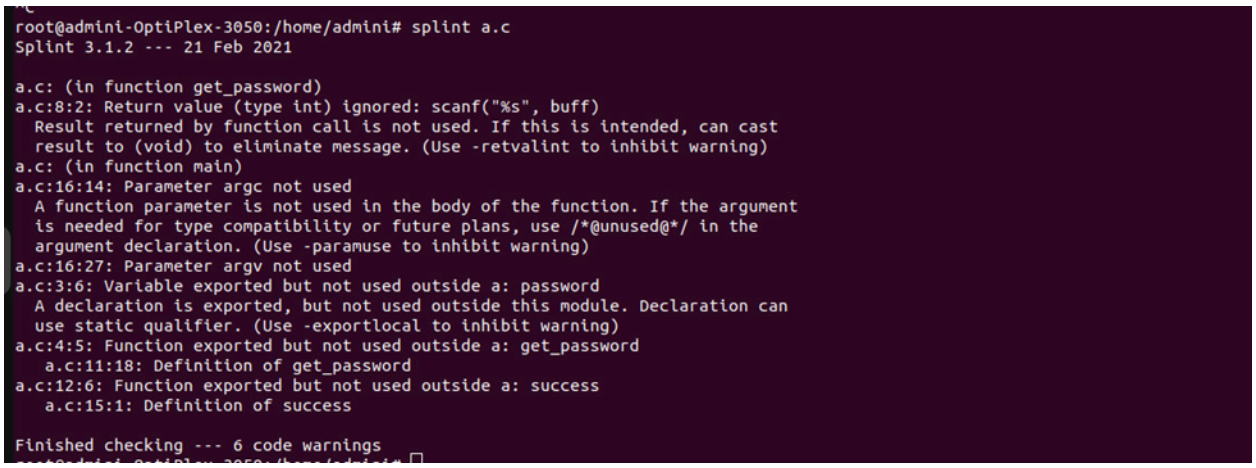
```
#include <stdio.h>
#include <string.h>
char password[] = "password";
int get_password()
```

Roll No: 2103163
Batch: C32
Name: Om Shete

```
{
        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:



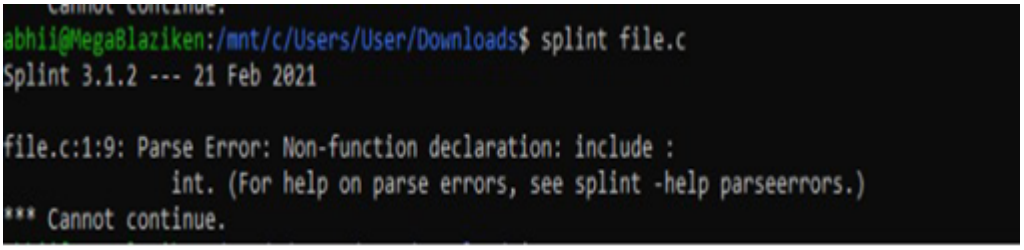
Program 3:

```
#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%p\n");

}

Output:



THADOMAL SHAHANI TSEC ENGINEERING COLLEGE
Om Shete
TE C32
2103163

Assignment No : 1

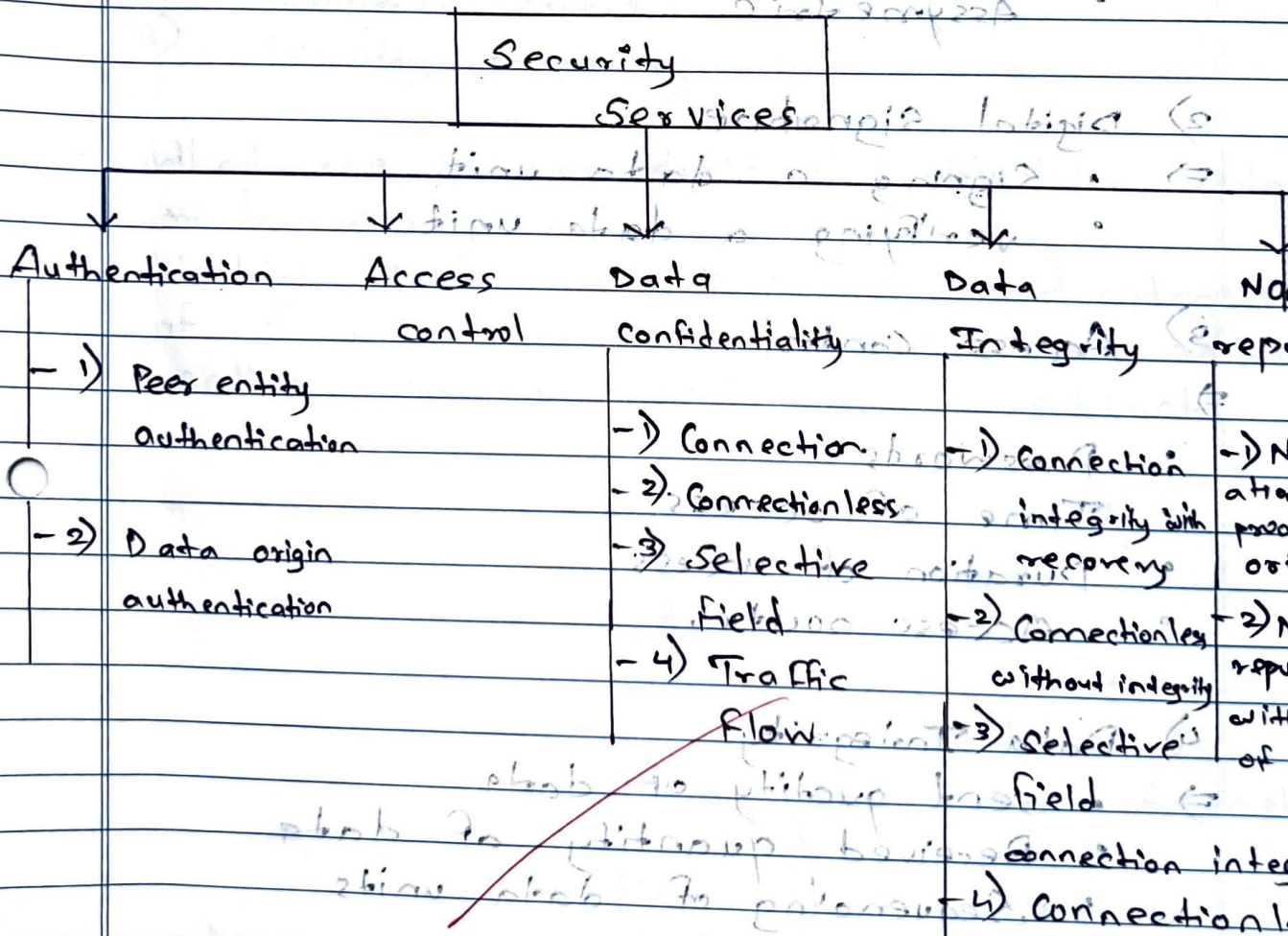
Q.1 Explain different services and mechanisms of security.

⇒

- **Security services**

⇒ Security services are safegaurd controls recommended to be placed at their various OSI layers.

- The various security services are listed as shown in below figure :-



Security Mechanism

⇒ Security mechanisms are various techniques recommended to provide security services at the various OSI layers.

- The various security mechanisms that can be applied are as following:

1) Encipherment

⇒
- Symmetric
- Assymetric

2) Digital Signature

⇒
- Signing a data unit
- Verifying a data unit

3) Access Control

⇒
- Passwords
- Time of access
- Duration of access
- Access route

4) Data Integrity

⇒
- Sent quality of data
- Received quantity of data
- Sequencing of data units
- Time stamping

5) Authentication
⇒ i) Handshaking
ii) Cryptographic techniques
6) Traffic padding
7) Routing control
8) Notarization
9) Pervasive security
10) Security labels
11) Event detection
12) Security audit
13) Security recovery

Q.2 What are various types of attacks? Explain with example.

⇒ At high level, there are two broad categories of security attacks carried over the network- active attacks and passive attacks

Types of Attacks

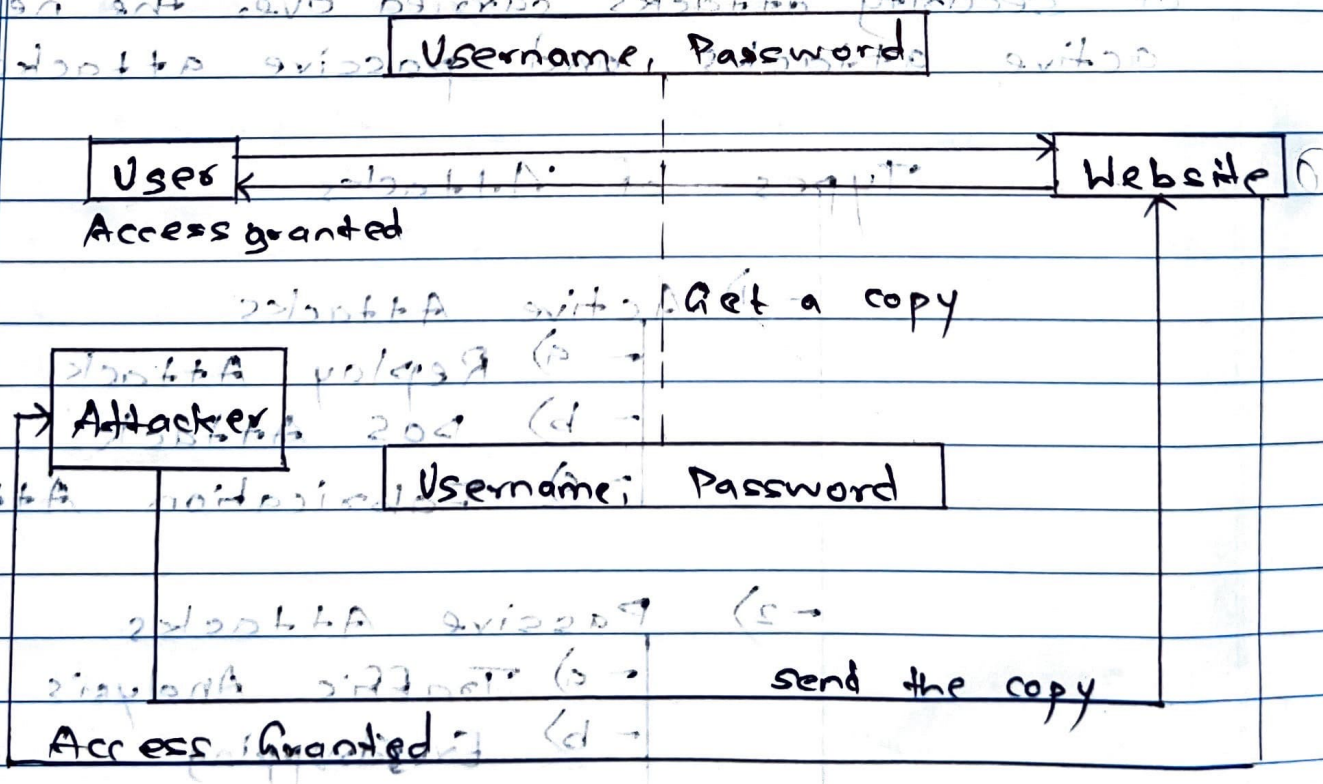
- 1) Active Attacks
  - a) Replay Attack
  - b) DOS Attack
  - c) Fabrication Attack
- 2) Passive Attacks
  - a) Traffic Analysis
  - b) Eavesdropping

1) **Active Attacks**

⇒ An active attack is defined as an attack where the attacker actively participates in the communication or the attack mechanism and disrupts the system by sending several manipulated inputs.

2) **Replay Attack**

⇒ This is like a replaying a song. The attacker captures the real and specific communication packets, stores them with herself and then sends it at a later point in time as though she is sending the information for the first time like an authentic information.



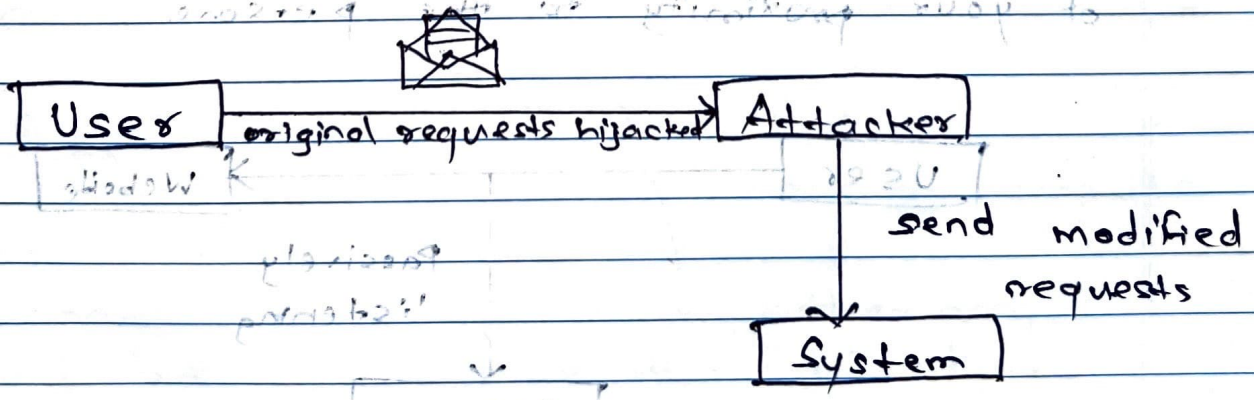
b) Denial of Service (DoS) Attack

⇒ DoS or its variation Distributed DoS refers to a category of attacks that can be aimed at various layers of network, operating system, application or other parts of information systems.

- For e.g., An application might be capable of processing a maximum of 100 requests per second. Attackers would typically send over ≥ 1000 requests per second such that the application fails to cope up with it and crashes.

c) Fabrication Attacks

⇒ Fabrication attack is again a broad category of active attacks where the attacker deliberately modifies messages, parameters, properties, etc. of information system components and try to alter the behaviour of the system often by passing security controls.



2) Passive Attacks

⇒ A passive attack is defined as an attack where the attacker does not alter the behaviour of the information system and silently performs her malicious activities.

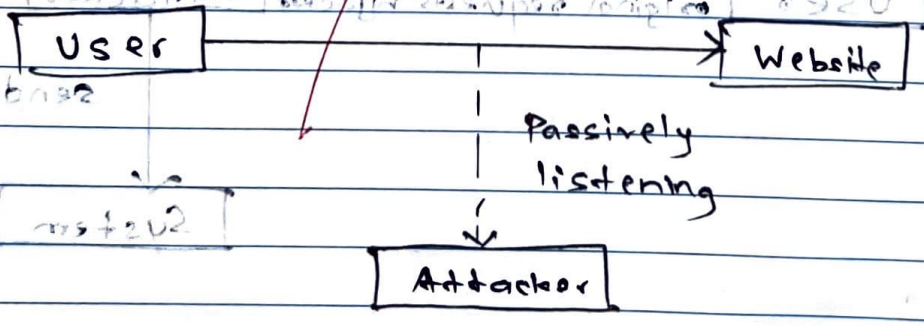
a) Traffic Analysis

⇒ In traffic analysis the network traffic and its pattern are watched out over a period of time to infer important information and guess possible activities.

- For e.g.,
Long communication : It can denote some emergency.

b) Eavesdropping

⇒ Eavesdropping is very similar to overhearing someone's telephonic conversation taking advantage of your proximity to the persons.



Q.3 List few latest viruses on net and explain.

⇒ - Latest viruses on net

⇒
1) Malware
2) Ransomware
3) Trojans
4) AoBrut
5) Jokerro

1) Malware

⇒ Malware is a broad term that includes more than computer viruses.
- Worms, Trojans, adware and even ransomware may all be considered malware.
- This is any computer code intentionally created to do damage to computer systems, gain unauthorized access to computers, or to steal information.

2) Ransomware

⇒ Ransomware disables access to computer files by encrypting data.
- Demands for payment or other requests must be met before the offending software will be unlocked to reinstate access to servers or business files.

3) Trojans

⇒ Trojans typically require the recipient to take some form of action, such as running a program or accessing a malicious website through a link passed by email.

- A common use of a Trojan attack is first to notify a computer user a virus has infected them.

4) CoBrut

⇒ CoBrut is a virus that is one of the most recent viruses to be unleashed by hackers. It is not terribly sophisticated in its technology but can wreak havoc just the same.

5) Jokeroo

⇒ Jokeroo is a serious piece of malware in the form of ransomware that is offered on underground hacking sites for proliferation by other cyberthieves.

- It can be distributed through social media sites, including Twitter and others.

Assignment No : 2

Q.1 Write short note on digital signatures and digital certificates.

⇒

- **Digital Signatures**

⇒ A digital signature is a hash value that has been encrypted with the sender's private key.
- The act of signing means encrypting the the message's hash value with a private key
- So the sender computes and encrypts the hash value with her private key.
- At the receiving end, you decrypt the hash value with sender's public key. Now, because no one knows the private key of the sender, altering the hash value and resigning with the private key of the sender is not possible.

Processing applied on message	Security Property achieved
Encryption	Confidentiality
Hashing	Integrity
Digitally signing	Integrity, authentication, non-repudiation.
Encryption and digitally signing	confidentiality, integrity, authentication, non-repudiation

- **Applications:**
1) Sending and receiving secure emails.
2) Signing documents.

- Properties of digital signature :-

1) Integrity
   ⇒ Via hash value calculation
2) Authentication
   ⇒ Via the ability to prove sender's identity by decrypting hash with the sender's known public key
3) Non-repudiation
   ⇒ Sender cannot deny sending the message because she used her private key to encrypt the hash.

- Digital certificate
  ⇒ Digital certificates is issued by a trusted third party which proves senders identity to the receiver and receiver's identity to the sender.
- A digital certificate is a certificate issued by a Certificate Authority (CA) to verify the identity of the certificate holder.
- Digital certificates is used to attach the public key with a particular individual or an identity.

- Digital Certificate Contains :-

1) Name of certificate holder
2) Serial number which used to uniquely identify a certificate
3) Expiration dates
4) Copy of certificate holder's public key.
5) Digital signature of certificate issuing authority.

163

Assignment No : 3

Q.1 List software vulnerabilities. How they are exploited to launch an attack?

⇒

- Software vulnerabilities :
  1) Buffer overflow
  2) SQL injection
  3) Cross-site scripting (XSS)
  4) Code injection
  5) Zero-day vulnerability
  6) Privilege escalation
  7) Denial of service (DOS)
  8) Man-in-the-Middle
- Today it is hard to imagine any software without any vulnerabilities. Some vulnerabilities are found during software testing and others are found in the field.
- Software vulnerabilities that are found after the software has been released are mitigated by giving out software patches or updates.
- You would have seen your browser, operating system and apps updating multiple times in its course of lifetime.
- These updates include both feature updates and also security updates.

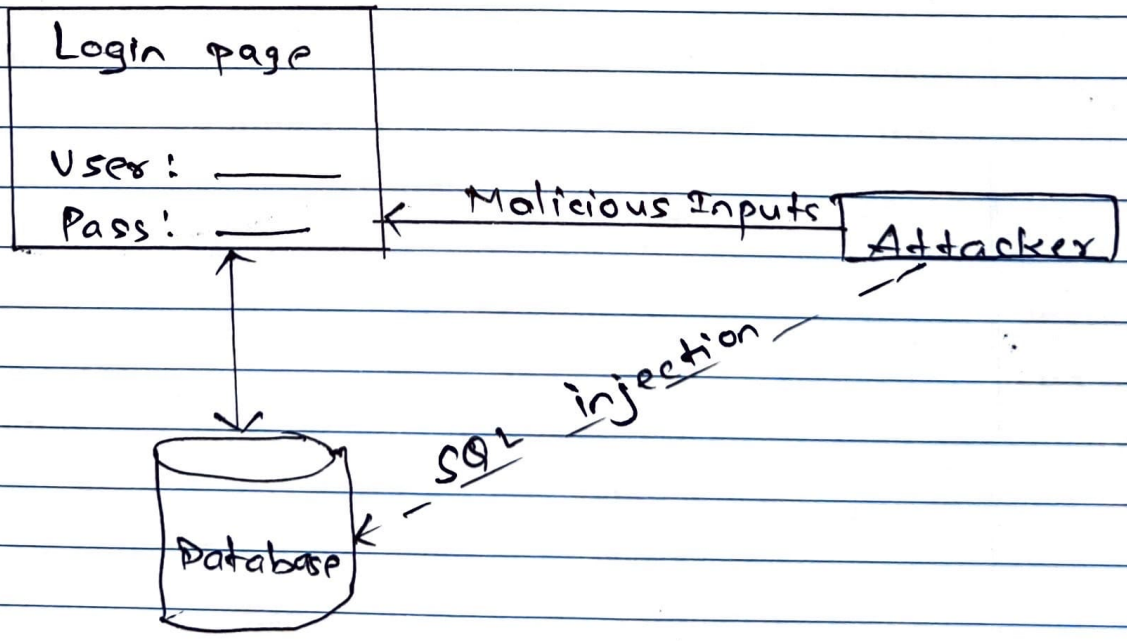
- Software developing companies have security vulnerabilities updated programs and teams that regularly investigate all reported security vulnerabilities from the field.
- As you learnt these companies also run bug bounty programs that help them find security vulnerabilities in their products and mitigate them before these vulnerabilities could be mass exploited.
- Let's see an example, depending on the type of vulnerability, it can be exploited either manually or using scripts or tools.
- Attackers has deep understanding of how the vulnerability works and what carries out the required steps in the right sequence to exploit vulnerability.

Q.2 What is SQL injection?

⇒ - SQL Injection (SQLi) is a vulnerability exploiting which an attacker can inject malicious SQL commands and cause the database server to execute them.
- Using SQLi the attacker can
1) Read sensitive and unauthorized data from the database,
2) Modify data
3) Execute admin commands on the database
4) And can also Issue commands to the operating system running the database

- Note that here the presence of SQLi attack does not mean that the SQL databases are inherently vulnerable or exploitable directly.
- This exploit only works if the application developer has programmed the application in such a way that the attack is possible.
- SQL databases just behaves as they should. So, even if the attack is called SQLi the fix for this exploit is not in the SQL databases.
- SQL is proven to be one of the most dangerous and impactful exploits on database based applications.

163



- There could be several non-malicious programming errors such as buffer overflow, SQL injection, cross-site scripting etc. that can be exploited by attacker.
- SQLi can potentially be exploited where the user input is directly used to form SQL database queries.