Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with 'A' Grade (CGPA : 3.18)

Continuous Assessment for Laboratory / Assignment sessions

Department: Computer Engineering

Academic Year 2022-23

Name: Kautik Jolapara SAP ID: 60004200107

Course: Information Security Course Code: **DJ19CEL603**

Year: **T.Y. B.Tech.** Sem: **VI** Batch: B1

Performance Indicators (Any no. of Indicators) (Maximum 5 marks per indicator)	1	2	3	4	5	6	7	8	9	10	11	Σ	A vg	A 1	A 2	Σ	A vg
Course Outcome	1	1	2	2	4	3	4	4	5	4	6						
1. Knowledge (Factual/Conceptual/Procedural/ Metacognitive)	5	5	5	5	5	5	5	5	5	5							
2. Describe (Factual/Conceptual/Procedural/ Metacognitive)	5	5	5	5	5	5	5	4	5	5							
3. Demonstration (Factual/Conceptual/Procedural/ Metacognitive)	4	5	4	5	5	5	4	4	5	4							
4. Strategy (Analyse & / or Evaluate) (Factual/Conceptual/ Procedural/Metacognitive)	4	4	4	4	5	4	4	4	4	4							
5. Interpret/ Develop (Factual/Conceptual/ Procedural/Metacognitive)	–	–	–	–	–	–	–	–	–					–	–		
6. Attitude towards learning (receiving, attending, responding, valuing, organizing, characterization by value)	4	4	4	4	4	4	4	4	4	4							
7. Non-verbal communication skills/ Behaviour or Behavioural skills (motor skills, hand-eye coordination, gross body movements, finely coordinated body movements speech behaviours)	–	–	–	–	–	–	–	–	–					–	–		
Total	22	23	22	23	24	23	22	21	23	22							
Signature of the faculty member																	

Outstanding (5), Excellent (4), Good (3), Fair (2), Needs Improvement (1)

Laboratory marks Σ Avg. =	Assignment marks Σ Avg. =	Total Term-work (25) =
Laboratory Scaled to (15) =	Assignment Scaled to (10) =	Sign of the Student: Kautik

Signature of the Faculty member: Name of the Faculty member:

Signature of Head of the Department
Date:

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Experiment 1

Date of Performance : 20-02-2023 **Date of Submission: 26-02-2023**

SAP Id: 60004200107 **Name :** Kartik Jolapara

Div: B **Batch :** B1

Aim of Experiment

Design and Implement Encryption and Decryption Algorithm for Caesar cipher cryptographic algorithm by considering letter [A..Z] and digits [0..9]. Create two functions Encrypt() and Decrypt(). Apply Brute Force Attack to reveal secret. Create Function BruteForce().

(CO1)

Theory / Algorithm / Conceptual Description

The Caesar cipher works by first choosing a shift value, which is an integer between 1 and 25. This shift value is then used to encode or decode a message. To encode a message, each letter in the message is replaced by the letter that is a certain number of positions down the alphabet. For example, if the shift value is 3, the letter 'A' would be replaced by the letter 'D', 'B' would be replaced by 'E', and so on. To decode a message, the process is simply reversed, by shifting each letter back by the same number of positions.

The algorithm for the Caesar cipher can be summarized as follows:

- Choose a shift value between 1 and 25.
- For each letter in the message:
  - If the letter is uppercase, shift it down the alphabet by the shift value and replace it with the corresponding letter.
  - If the letter is lowercase, shift it down the alphabet by the shift value and replace it with the corresponding letter.
  - If the letter is not a letter (such as a number or symbol), leave it unchanged.
- The resulting message is the encoded message.

To decode a message, the same process is followed, but in reverse, by shifting each letter back up the alphabet by the same number of positions.

Program

```
 def encrypt(message, shift):
    ciphertext = ''
    for char in message:
        # Check if the character is an uppercase or lowercase letter
        if char.isupper():
            ciphertext += chr((ord(char) + shift - 65) % 26 + 65)
        elif char.islower():
            ciphertext += chr((ord(char) + shift - 97) % 26 + 97)
        else:
            ciphertext += char
    return ciphertext

def decrypt(ciphertext, shift):
    message = ''
    for char in ciphertext:
        # Check if the character is an uppercase or lowercase letter
        if char.isupper():
            message += chr((ord(char) - shift - 65) % 26 + 65)
        elif char.islower():
            message += chr((ord(char) - shift - 97) % 26 + 97)
        else:
            message += char
    return message

def brute_force_attack(ciphertext):
    for shift in range(1, 26):
        message = decrypt(ciphertext, shift)
        print(f'Shift = {shift:2d}: {message}')


# Example usage
message = 'This is a secret message'
shift = 5

print("PLAIN TEXT:", message)
print()

ciphertext = encrypt(message, shift)
print("CIPHER TEXT:", ciphertext)

decrypted_message = decrypt(ciphertext, shift)
print("DECRYPTED TEXT:", decrypted_message)
print()

brute_force_attack(ciphertext)
```

Input

```
● ➜  Practicals git:(master) ✗ python3 -u "/media/codingmickey/Kartik/
  PLAIN TEXT: This is a secret message
```

Output

```
CIPHER TEXT: Ymnx nx f xjhwjy rjxxflj
DECRYPTED TEXT: This is a secret message

Shift =  1: Xlmw mw e wigvix qiwweki
Shift =  2: Wklv lv d vhfuhw phvvdjh
Shift =  3: Vjku ku c ugetgv oguucig
Shift =  4: Uijt jt b tfdsfu nfttbhf
Shift =  5: This is a secret message
Shift =  6: Sghr hr z rdbqds ldrrzfd
Shift =  7: Rfgq gq y qcapcr kcqqyec
Shift =  8: Qefp fp x pbzobq jbppxdb
Shift =  9: Pdeo eo w oaynap iaoowca
Shift = 10: Ocdn dn v nzxmzo hznnvbz
Shift = 11: Nbcm cm u mywlyn gymmuay
Shift = 12: Mabl bl t lxvkxm fxlltzx
Shift = 13: Lzak ak s kwujwl ewkksyw
Shift = 14: Kyzj zj r jvtivk dvjjrxv
Shift = 15: Jxyi yi q iushuj cuiiqwu
Shift = 16: Iwxh xh p htrgti bthhpvt
Shift = 17: Hvwg wg o gsqfsh asggous
Shift = 18: Guvf vf n frperg zrffntr
Shift = 19: Ftue ue m eqodqf yqeemsq
Shift = 20: Estd td l dpncpe xpddlrp
Shift = 21: Drsc sc k combod wocckqo
Shift = 22: Cqrb rb j bnlanc vnbbjpn
Shift = 23: Bpqa qa i amkzmb umaaiom
Shift = 24: Aopz pz h zljyla tlzzhnl
Shift = 25: Znoy oy g ykixkz skyygmk
○ ➜  Practicals git:(master) ✗
```

Experiment 2

Date of Performance : 20-02-2023 Date of Submission: 20-02-2023

SAP Id: 60004200107 Name : Kartik Jolapara

Div: B Batch : B1

Aim of Experiment

Design and Implement Playfair Cipher. Create two function Encyrpt() and Decyrpt().

(CO1)

Theory / Algorithm / Conceptual Description

The Playfair cipher uses a 5 by 5 table containing a key word or phrase. Memorization of the keyword and 4 simple rules was all that was required to create the 5 by 5 table and use the cipher.

To generate the key table, one would first fill in the spaces in the table (a modified Polybius square) with the letters of the keyword (dropping any duplicate letters), then fill the remaining spaces with the rest of the letters of the alphabet in order (usually omitting "J" or "Q" to reduce the alphabet to fit; other versions put both "I" and "J" in the same space). The key can be written in the top rows of the table, from left to right, or in some other pattern, such as a spiral beginning in the upper-left-hand corner and ending in the center. The keyword together with the conventions for filling in the 5 by 5 table constitute the cipher key.

To encrypt a message, one would break the message into digrams (groups of 2 letters) such that, for example, "HelloWorld" becomes "HE LL OW OR LD". These digrams will be substituted using the key table. Since encryption requires pairs of letters, messages with an odd number of characters usually append an uncommon letter, such as "X", to complete the final digram. The two letters of the digram are considered opposite corners of a rectangle in the key table. To perform the substitution, apply the following 4 rules, in order, to each pair of letters in the plaintext:

1. If both letters are the same (or only one letter is left), add an "X" after the first letter. Encrypt the new pair and continue. Some variants of Playfair use "Q" instead of "X", but any letter, itself uncommon as a repeated pair, will do.
2. If the letters appear on the same row of your table, replace them with the letters to their immediate right respectively (wrapping around to the left side of the row if a letter in the original pair was on the right side of the row).
3. If the letters appear on the same column of your table, replace them with the letters immediately below respectively (wrapping around to the top side of the column if a letter in the original pair was on the bottom side of the column).

4. If the letters are not on the same row or column, replace them with the letters on the same row respectively but at the other pair of corners of the rectangle defined by the original pair. The order is important – the first letter of the encrypted pair is the one that lies on the same row as the first letter of the plaintext pair.

Program

```
def create_matrix(key):     key =
key.upper()
    matrix = [[0 for i in range (5)] for j in range(5)]     letters_added = []     row = 0
col = 0
     for letter in key:         if letter not in letters_added:
matrix[row][col] = letter
letters_added.append(letter)         else:
            continue         if (col==4):
col = 0             row += 1         else:
            col += 1
     for letter in range(65,91):         if letter==74:                 continue
if chr(letter) not in letters_added:
letters_added.append(chr(letter))
                 index = 0     for i in
range(5):         for j in range(5):
            matrix[i][j] = letters_added[index]             index+=1     return
matrix
```

```
def separate_same_letters(message):
    index = 0     while (index<len(message)):
        l1 = message[index]        if index == len(message)-1:            message = message + 'X'
index += 2            continue        l2 = message[index+1]        if l1==l2:            message =
message[:index+1] + "X" + message[index+1:]        index +=2        return message
 def indexOf(letter,matrix):
    for i in range (5):
        try:
            index = matrix[i].index(letter)            return (i,index)
except:
            continue
 def playfair(key, message, encrypt=True):
    inc = 1     if encrypt==False:
        inc = -1
    matrix = create_matrix(key)     message = message.upper()     message =
message.replace(' ','')        message = separate_same_letters(message)
cipher_text=''     for (l1, l2) in zip(message[0::2], message[1::2]):
        row1,col1 = indexOf(l1,matrix)        row2,col2 =
indexOf(l2,matrix)        if row1==row2:
            cipher_text += matrix[row1][(col1+inc)%5] + matrix[row2][(co l2+inc)%5]        elif col1==col2:
            cipher_text += matrix[(row1+inc)%5][col1] + matrix[(row2+inc
)%5][col2]        else:
            cipher_text += matrix[row1][col2] + matrix[row2][col1]     print(matrix)     return
cipher_text
```

```
option = 1 while(option == 1 or option == 2):
  option = int(input("Select an option\n 1 for Encyrption\n 2 for decyrp tion\n 3 for exit"))   if( option == 1):
    plainText = input("Enter the Plain text\n")     key = input("Enter
Key\n")     print ('Encripting')     print ( playfair(key, plainText))   if(
option == 2):
    cipherText = input("Enter the cipher text\n")     key = input("Enter
Key\n")     print ('Decrypting')     print ( playfair(key, cipherText, False))
if (option == 3):     break
```

Input

```
Select an option
 1 for Encyrption
 2 for decyrption
 3 for exit1
Enter the Plain text
meet me tomorrow
Enter Key
krish
Encripting
[['K', 'R', 'I', 'S', 'H'], ['A', 'B', 'C', 'D', 'E'], ['F', 'G', 'L', 'M', 'N'], ['O', 'P', 'Q', 'T', 'U'], ['V', 'W', 'X', 'Y', 'Z']]
NDDUNDUPFTIWKPXY
Select an option
 1 for Encyrption
 2 for decyrption
 3 for exit2
Enter the cipher text
NDDUNDUPFTIWKPXY
Enter Key
krish
Decrypting
[['K', 'R', 'I', 'S', 'H'], ['A', 'B', 'C', 'D', 'E'], ['F', 'G', 'L', 'M', 'N'], ['O', 'P', 'Q', 'T', 'U'], ['V', 'W', 'X', 'Y', 'Z']]
MEETMETOMORXROWX
```

Output

```
Select an option
 1 for Encyrption
 2 for decyrption
 3 for exit1
Enter the Plain text
meet me tomorrow
Enter Key
krish
Encripting
[['K', 'R', 'I', 'S', 'H'], ['A', 'B', 'C', 'D', 'E'], ['F', 'G', 'L', 'M', 'N'], ['O', 'P', 'Q', 'T', 'U'], ['V', 'W', 'X', 'Y', 'Z']]
NDDUNDUPFTIWKPXY
Select an option
 1 for Encyrption
 2 for decyrption
 3 for exit2
Enter the cipher text
NDDUNDUPFTIWKPXY
Enter Key
krish
Decrypting
[['K', 'R', 'I', 'S', 'H'], ['A', 'B', 'C', 'D', 'E'], ['F', 'G', 'L', 'M', 'N'], ['O', 'P', 'Q', 'T', 'U'], ['V', 'W', 'X', 'Y', 'Z']]
MEETMETOMORXROWX
```

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Experiment 3

Date of Performance : 27-02-2023 **Date of Submission: 27-02-2023**

SAP Id: 60004200107 **Name : Kartik Jolapara**

Div: B **Batch : B1**

Aim of Experiment

Implement simple columnar transposition technique. The columnar transposition rearranges the plaintext letters, based on a matrix filled with letters in the order determined by the secret keyword.

Theory / Algorithm / Conceptual Description

The Columnar Transposition Cipher is a form of transposition cipher just like the Rail Fence Cipher. Columnar Transposition involves writing the plaintext out in rows and then reading the ciphertext off in columns one by one.

Encryption:

In a transposition cipher, the order of the alphabet is re-arranged to obtain the cipher text:

1. The message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order.
2. Width of the rows and the permutation of the columns are usually defined by a keyword.
3. For example, the word HACK is of length 4 (so the rows are of length 4), and the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be "3 1 2 4".
4. Any spare spaces are filled with nulls or left blank or placed by a character (Example: _).
5. Finally, the message is read off in columns, in the order specified by the keyword.

Decryption:

1. To decipher it, the recipient has to work out the column lengths by dividing the message length by the key length.
2. Then, write the message out in columns again, then re-order the columns by reforming the keyword.

Encryption

Given text = Geeks for Geeks

Keyword = HACK **Length of Keyword** = 4 (no of rows) **Order of Alphabets in HACK** = 3124

H	A	C	K
3	1	2	4
G	e	e	k
s	_	f	o
r	_	G	e
e	k	s	_

Print Characters of column 1,2,3,4

Encrypted Text = e kefGsGsrekoe_

Program

```
import java.util.*;

class ColumnCipher {
    String rank(String key) {
        StringBuilder sb = new StringBuilder();
        char[] ch = key.toLowerCase().toCharArray();
        Arrays.sort(ch);

        for(int i = 0; i < key.length(); i ++) {
            for(int j = 0; j < key.length(); j ++) {
                if(key.toLowerCase().charAt(i) == ch[j]) {
                    ch[j] = '~';
                    sb.append(j + 1);
                    break;
                }
            }
        }

        return sb.toString();
    }

    ArrayList<ArrayList<Character>> encryptMatrix(String text, int n) {
        ArrayList<ArrayList<Character>> arrayList = new ArrayList<>();
        ArrayList<Character> array;
        int row = (int) Math.ceil((double) text.length() / n);
        for(int i = 0; i < row; i ++) {
            array = new ArrayList<>();
            for(int j = 0; j < n; j ++) {
                if(i * n + j < text.length()) {
                    array.add(text.charAt(i * n + j));
                }
            }
            arrayList.add(array);
        }

        while(arrayList.get(arrayList.size() - 1).size() < n) {
            arrayList.get(arrayList.size() - 1).add('~');
        }

        return arrayList;
    }

    String encrypt(String rank, String text) {
```

```
        Map<Integer, Integer> map = new HashMap<>();
        ArrayList<ArrayList<Character>> arrayList = encryptMatrix(text.toLowerCase(),
rank.length());
        StringBuilder sb = new StringBuilder();
        for(int i = 0; i < rank.length(); i ++) {
            map.put(Integer.parseInt(String.valueOf(rank.charAt(i))), i);
        }

        for(int i = 0; i < rank.length(); i ++) {
            for(int j = 0; j < (int) Math.ceil((double) text.length() / rank.length()); j ++) {
                sb.append(arrayList.get(j).get(map.get(i + 1)));
            }
        }

        return sb.toString();
    }

    char[][] decryptMatrix(String cipher, Map<Integer, Integer> map, int n) {
        int row = (int) Math.ceil((double) cipher.length() / n);
        char[][] ch = new char[row][n];

        for(int i = 0; i < n; i ++) {
            for(int j = 0; j < row; j ++) {
                ch[j][map.get(i + 1)] = cipher.charAt(i * row + j);
            }
        }

        return ch;
    }

    String decrypt(String rank, String cipher) {
        Map<Integer, Integer> map = new HashMap<>();
        StringBuilder sb = new StringBuilder();
        for(int i = 0; i < rank.length(); i ++) {
            map.put(Integer.parseInt(String.valueOf(rank.charAt(i))), i);
        }

        char[][] ch = decryptMatrix(cipher.toLowerCase(), map, rank.length());

        for (char[] chars : ch) {
            for (char c : chars) {
                sb.append(c);
            }
        }

        return sb.toString().replace("~", "");
```

```
    }
}

public class ColumnarTranspositionalCipher {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        ColumnCipher columnCipher = new ColumnCipher();
        System.out.println("Enter the key: ");
        String key = in.nextLine();
        System.out.println("Enter the text to be ciphered: ");
        String text = in.nextLine();
        String rank = columnCipher.rank(key.toLowerCase());
        String encryptedText = columnCipher.encrypt(rank, text);
        String decryptedText = columnCipher.decrypt(rank, encryptedText);

        System.out.println("The rank of the key" + key + " is: " + rank);
        System.out.println("The encrypted text is: " + encryptedText);
        System.out.println("The decrypted text is: " + decryptedText);
    }
}
```

Input

```
PS C:\Users\HP\VSC> cd "c:\Users\HP\VSC\Informatoin and Network Security\" ; if ($?) { javac ColumnarTranspositionalCipher.java } ; if
($?) { java ColumnarTranspositionalCipher }
Enter the key:
heaven
Enter the text to be ciphered:
Hello, welcome to CS
```

Output

```
The rank of the keyheaven is: 421635
The encrypted text is: le ~ewesoco~h mc,o ~llt~
The decrypted text is: hello, welcome to cs
```

Experiment 4

Date of Performance : 27-02-2023 **Date of Submission: 04-03-2023**

SAP Id: 60004200107 **Name : Kartik Jolapara**

Div: B **Batch : B1**

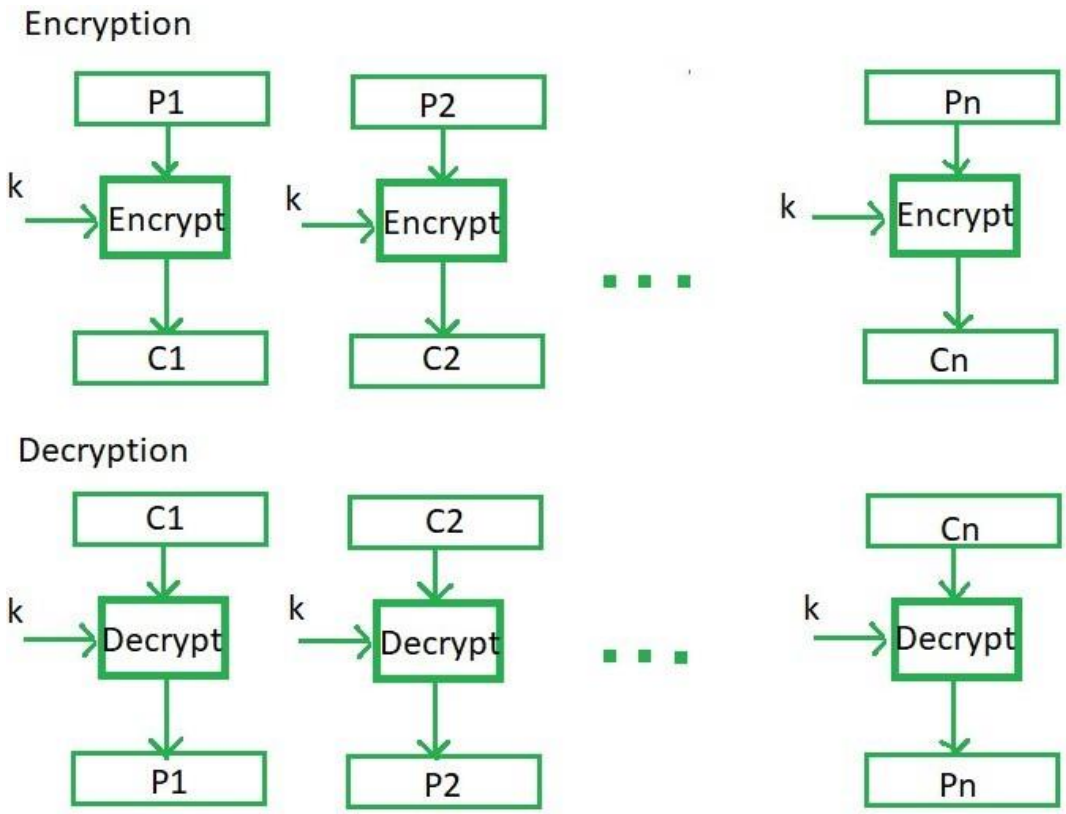
Aim of Experiment

Design and Implement Electronic Code Book(ECB) algorithmic mode. Plaintext is given as a paragraph. First, convert the given paragraph into ASCII values and then Binary. Use 128 bits of a block as input to ECB and encrypt using "t" bits shifter(Left/Right). Display encrypted paragraph.

Theory / Algorithm / Conceptual Description

Encryption algorithms are divided into two categories based on the input type, as block cipher and stream cipher. Block cipher is an encryption algorithm that takes a fixed size of input say b bits and produces a ciphertext of b bits again. If the input is larger than b bits it can be divided further. For different applications and uses, there are several modes of operation for a block cipher.

Electronic Code Book (ECB):
An electronic code book is the easiest block cipher mode of functioning. It is easier because of the direct encryption of each block of input plaintext and output is in form of blocks of encrypted ciphertext. Generally, if a message is larger than b bits in size, it can be broken down into a bunch of blocks and the procedure is repeated.



Advantages of using ECB:

- Parallel encryption of blocks of bits is possible, thus it is a faster way of encryption.
- Simple way of the block cipher.

Disadvantages of using ECB:

- Prone to cryptanalysis since there is a direct relationship between plaintext and ciphertext.

Program

```
import java.math.BigInteger;
import java.util.Scanner;

class ECB {
    String ascii(String paragraph) {
        StringBuilder sb = new StringBuilder();
        for(char i: paragraph.toUpperCase().toCharArray()) {
            sb.append(Integer.valueOf(i));
        }

        return sb.toString();
    }

    String binary(String ascii) {
        return new BigInteger(ascii).toString(2);
    }

    String doLeftShift(String block, int t) {
        return block.substring(t) + block.substring(0, t);
    }

    String encrypt(String binary) {
        int count = 1;
        StringBuilder sb = new StringBuilder(), encryptedParagraph = new StringBuilder();
        for(int i = 0; i < binary.length(); i ++, count ++) {
            sb.append(binary.charAt(i));
            if(count % 128 == 0) {
                encryptedParagraph.append(doLeftShift(sb.toString(), 3));
                sb.setLength(0);
            }
        }
        if(sb.length() != 0) {
            encryptedParagraph.append(doLeftShift(sb.toString(), 3));
        }

        return encryptedParagraph.toString();
    }

    String doRightShift(String block, int t) {
        return block.substring(block.length() - t) + block.substring(0, block.length() - t);
    }

    String decrypt(String encryptedParagraph) {
```

```
        int count = 1;
        StringBuilder sb = new StringBuilder(), binary = new StringBuilder(),
decryptedParagraph = new StringBuilder();
        for(int i = 0; i < encryptedParagraph.length(); i ++, count ++) {
            sb.append(encryptedParagraph.charAt(i));
            if(count % 128 == 0) {
                binary.append(doRightShift(sb.toString(), 3));
                sb.setLength(0);
            }
        }
        if(sb.length() != 0) {
            binary.append(doRightShift(sb.toString(), 3));
        }

        String ascii = new BigInteger(binary.toString(), 2).toString(10);

        for(int i = 0; i < ascii.length(); i+= 2) {
            decryptedParagraph.append((char) Integer.valueOf(ascii.substring(i, i + 2)).intValue());
        }

        return decryptedParagraph.toString();
    }
}

public class ElectronicCodeBookCipher {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        ECB ecb = new ECB();

        System.out.println("Please enter a paragraph: ");
        String paragraph = in.nextLine();
        String ascii = ecb.ascii(paragraph);
        String binary = ecb.binary(ascii);
        String encryptedParagraph = ecb.encrypt(binary);

        System.out.println("The ASCII conversion of the paragraph is: " + ascii);
        System.out.println("The Binary value of the ASCII  value is: " + binary);
        System.out.println("The encrypted paragraph is: " + encryptedParagraph);
        System.out.println("The decrypted paragraph is: " + ecb.decrypt(encryptedParagraph));
    }
}
```

Input

```
PS C:\Users\HP\VSC\Informatoin and Network Security> cd "c:\Users\HP\VSC\Informatoin and Network Security\" ; if ($?) { javac Electroni
cCodeBookCipher.java } ; if ($?) { java ElectronicCodeBookCipher }
Please enter a paragraph:
Hello, how have you been?
```

Output

```
The ASCII conversion of the paragraph is: 7269767679443272798732726586693289798532666697863
The Binary value of the ASCII value is: 1100011011110111100100100100100011111001100110100000110000010101110100011100011000011100101111 0
01100100111101111111010001101001101000001101101111111001000100110100 0111
The encrypted paragraph is: 001101111011110010010010010001111100110011010000011000001010111010001110001100001110010111100110010011110 11
1111010001101001101100001101101111111001000100110100 0111100
The decrypted paragraph is: HELLO, HOW HAVE YOU BEEN?
```

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Experiment 5

Date of Performance : 06-03-2023 **Date of Submission: 07-03-2023**

SAP Id: 60004200107 **Name :** Kartik Jolapara

Div: B **Batch :** B1

Aim of Experiment

mplement Hill Cipher. Create two functions Encrypt() and Decrypt(). Demonstrate these ciphers using Color Images / Gray Scale Images

Theory / Algorithm / Conceptual Description

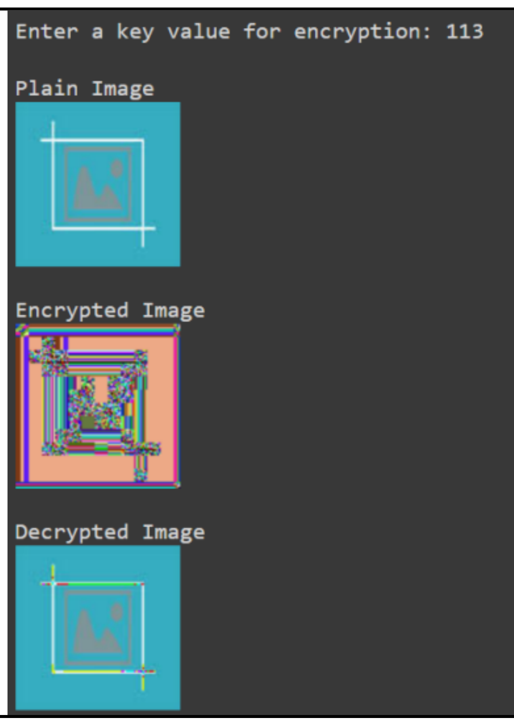
Hill cipher is a polygraphic substitution cipher based on linear algebra. Each letter is represented by a number modulo 26. Often the simple scheme A = 0, B = 1, …, Z = 25 is used, but this is not an essential feature of the cipher. To encrypt a message, each block of n letters (considered as an n-component vector) is multiplied by an invertible n × n matrix, against modulus 26. To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption. The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible n × n matrices (modulo 26).

Program

```
from PIL import Image
from numpy import array
import numpy as np
def modInverse(A, M):
for X in range(1, M):
if (((A % M) * (X % M)) % M == 1):
return X
return -1
im = Image.open(r"1.jpg")
ar = array(im)
list(ar)
k = int(input("Enter a key value for encryption: "))
key = np.zeros((100,100,3))
for i in range(100):
 for j in range(100):
```

```
 key[i][j] = k
result = np.zeros((100,100,3))
for i in range(100):
 for j in range(100):
 result[i][j] = (key[i][j] * ar[i][j]) % 255
print("\nPlain Image")
r = Image.fromarray(np.uint8(ar))
r.show()
print("\nEncrypted Image")
r = Image.fromarray(np.uint8(result))
r.show()
mod = modInverse(k, 255)
key_inv = np.zeros((100,100,3))
for i in range(100):
 for j in range(100):
 key_inv[i][j] = mod
result_dec = np.zeros((100,100,3))
for i in range(100):
 for j in range(100):
 result_dec[i][j] = (key_inv[i][j] * result[i][j]) % 255
print("\nDecrypted Image")
r = Image.fromarray(np.uint8(result_dec))
r.show()
```

Input/Output



CONCLUSION

Thus, we have successfully implemented Hill cipher

Experiment 6

Date of Performance : 13-03-2023
Date of Submission : 20-03-2023
SAP Id: 60004200107 **Name :** Kartik Jolapara
Div: B **Batch :** B1

AIM

Implement RSA cryptosystem. Demonstrate the application of RSA cryptosystem using multimedia data.

THEORY

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 that the Public Key is given to everyone and the Private key is kept private.
The idea of RSA is because 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 soon. But till now it seems to be an infeasible task.

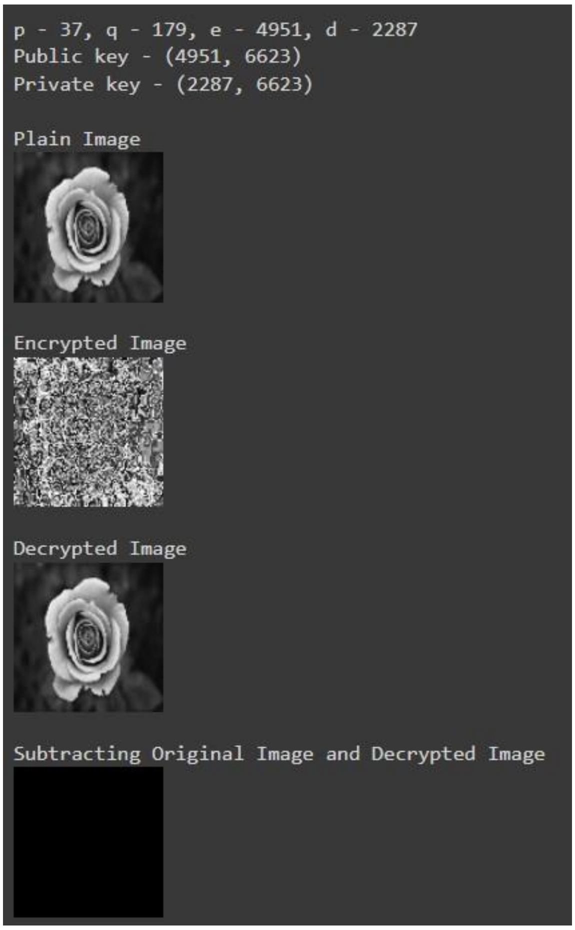
PROGRAM

```
import random as r from
PIL import Image from
numpy import array
import numpy as np

def modInverse(e, phin):        for d in range(1, phin):
        if (((e % phin) * (d % phin)) % phin == 1):
                    return d
        return -1

def prime(a):
count = 0   for i in
range(2, int(a/2)):
if a % i == 0:
count += 1   if
count == 0:
```

```
return True   else:
return False

cond = 1 count
= 0 while
cond:
p          = r.randint(2,
255)   if prime(p):
break

cond = 1
while cond:
q          = r.randint(2,
255)   if prime(q) and
q != p:
   break

n = (p * q) phin =
(p - 1)*(q - 1)

cond = 1
while cond:
  e = r.randint(2, phin)   if
prime(e) and e != p and e != q:
   break

d = modInverse(e, phin)

while d == -1:
cond = 1
  temp = e
while cond:
e = r.randint(2, phin)     if prime(e) and e != p and e != q and e != temp:
     break   d =
modInverse(e, phin)
print(f"p - {p}, q - {q}, e
- {e}, d - {d}")

publicKey = (e, n)
privateKey = (d, n)

print(f"Public key - {publicKey}")
print(f"Private key - {privateKey}")
```

```
im = Image.open(r"/content/download (1).jpg")
ar = array(im) list(ar)

result = np.zeros((100,100,3))
for i in range(100):   for j in
range(100):     for k in
range(3):
      result[i][j][k] = (int(ar[i][j][k])**e)%n

print("\nPlain Image") im =
Image.fromarray(np.uint8(ar))
im.show() print("\nEncrypted Image")
im = Image.fromarray(np.uint8(result))
im.show()

result_dec = np.zeros((100,100,3))
for i in range(100):   for j in
range(100):     for k in range(3):
      result_dec[i][j][k] = (int(result[i][j][k])**d)%n

print("\nDecrypted Image") im =
Image.fromarray(np.uint8(result_dec))
im.show()

check =
np.zeros((100,100,3)) for i in
range(100):   for j in
range(100):     for k in
range(3):
      check[i][j][k] = int(ar[i][j][k])-int(result_dec[i][j][k]) print("\nSubtracting Original Image
and Decrypted Image") r = Image.fromarray(np.uint8(check)) r.show()
```

INPUT AND OUTPUT



CONCLUSION

Thus, we have successfully implemented RSA cryptosystem.

Experiment 7

Date of Performance : 20-03-23

Date of Submission : 30-03-23

SAP Id: 60004200107 **Name :** Kartik Jolapara

Div: B **Batch :** B1

AIM

Implement Merkle Root creation with the help of SHA-1. Your program will have input as paragraph. Paragraph can be converted to suitable blocks for which hash values can be computed. Finally generate Merkle root based on these computed hash values.

THEORY

A hash tree is also known as Merkle Tree. It is a tree in which each leaf node is labeled with the hash value of a data block and each non-leaf node is labeled with the hash value of its child nodes labels.

In a Merkle tree, transactions are grouped into pairs. The hash is computed for each pair and this is stored in the parent node. Now the parent nodes are grouped into pairs and their hash is stored one level up in the tree. This continues till the root of the tree. The different types of nodes in a Merkle tree are:

- Root node: The root of the Merkle tree is known as the Merkle root and this Merkle root is stored in the header of the block.
- Leaf node: The leaf nodes contain the hash values of transaction data. Each transaction in the block has its data hashed and then this hash value (also known as transaction ID) is stored in leaf nodes.
- Non-leaf node: The non-leaf nodes contain the hash value of their respective children. These are also called intermediate nodes because they contain the intermediate hash values and the hash process continues till the root of the tree.

PROGRAM

```
from hashlib import sha256
```

```
def  hash(x):      ans  =  sha256(x.encode("utf-
8")).hexdigest()   return ans

def hash_value(h):   h1 = []   if
len(h) % 2 == 0:     for i in
range(0, len(h), 2):       text =
h[i] + h[i + 1]
h1.append(hash(text))   else:
for i in range(0, len(h) - 1, 2):
      text  =  h[i]  +  h[i  +  1]
h1.append(hash(text))
h1.append(h[len(h) - 1])

  return h1

para = input("Enter para (use '.' to seperate lines): ")

l = para.split('.') count
= len(l)

if count % 8 != 0 :
temp = int(count / 8)
for i in range(0, (temp
+ 1) * 8 - count):
    l.append(l[count - 1])
```

```
h = list(map(hash, l))
length = len(h)

while length > 1:   h
=      hash_value(h)
length = len(h)

print("\n\nMerkle root - ", h[0])
```

INPUT AND OUTPUT

```
Enter para (use '.' to seperate lines): hello.goodbye.blue.crystal.puppy.sandalwood.lamp.fineprint.myrtle


Merkle root -  653198460235112299a813791127838c8e0de563
```

CONCLUSION

Thus, we have successfully implemented Merkle root creation using SHA-1.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Experiment 8

Date of Performance : 10-04-2023
Date of Submission : 17-04-2023
SAP Id: 60004200107 **Name :** Kartik Jolapara
Div: B **Batch :** B1

AIM

To implement Diffie Hellman Key exchange protocol. Demonstrate man in middle attack.

THEORY

Diffie Hellman Key exchange

Diffie-Hellman Key Exchange is a cryptographic protocol that allows two parties to establish a shared secret key over an insecure communication channel. The shared key can then be used for encryption, decryption, or other cryptographic operations. The protocol was invented by Whitfield Diffie and Martin Hellman in 1976 and is widely used in modern cryptography.
The basic idea behind the Diffie-Hellman protocol is that both parties agree on a large prime number and a generator, which is a smaller number that generates a cyclic group of numbers modulo the prime. Each party then selects a private key, which is a randomly chosen number, and computes a public key by raising the generator to the power of the private key modulo the prime. The parties exchange their public keys over the insecure channel, and then use them to compute a shared secret key.
The Diffie-Hellman protocol is used in many cryptographic applications, such as secure communication over the Internet (e.g., in the TLS/SSL protocol), key exchange in symmetric encryption schemes (e.g., in the SSH protocol), and digital signatures.

Man in the Middle

A man-in-the-middle (MITM) attack is a type of cyber-attack where an attacker intercepts and alters communications between two parties who believe they are communicating directly with each other. The attacker can eavesdrop on the communication, modify the content of the messages, and even impersonate one or both of the parties. In order to carry out a MITM attack, the attacker must be able to intercept the communication between the two parties. This can be done in several ways, such as by compromising a network device (e.g., a router or switch), by using a rogue access point to intercept wireless communications, or by using malware to intercept communications on a compromised computer.

Once the attacker has intercepted the communication, they can then modify the content of the messages. For example, they may insert malicious code or malware into a download, or modify a financial transaction to redirect funds to their own account.

PROGRAM

a) Diffie Hellman Key exchange Alice Program import socket import random as r

```
def alice():
    host = socket.gethostname()
port = 5000      s =
socket.socket()
    s.bind((host, port))
    s.listen(2)     conn, address = s.accept()
print("Connection from: " + str(address))

    p = int(input("Enter p = "))     g
= int(input("Enter g = "))
conn.send(str(p).encode('ascii'))
conn.send(str(g).encode('ascii'))

    a = r.randint(3, 1000)     Xa =
int(pow(g, a, p))     print("Xa
computed = ", Xa)
conn.send(str(Xa).encode('ascii'))

    Xb = int(conn.recv(1024).decode('ascii'))
print("Xb from Bob = ", Xb)

    Ak = int(pow(Xb, a, p))     print('Secret
key for Alice is = %d' % (Ak))
conn.close()

alice()
```

Bob Program import socket import random as r def bob():

```
    host =
socket.gethostname(
)      port = 5000
s = socket.socket()
    s.connect((host, port))
```

p =
int(s.recv(1024).decode('ascii'))
print("p = ", p)   g =
int(s.recv(1024).decode('ascii'))
print("g = ", g)
Xa = int(s.recv(1024).decode('ascii'))
print("Xa from Man in the middle = ", Xa)

b = r.randint(3, 1000)   Xb
= int(pow(g, b, p))
print("Xb computed = ", Xb)
s.send(str(Xb).encode('ascii'))

Bk = int(pow(Xa, b, p))   print('Secret
key for Bob is = %d' % (Bk))   s.close()

bob()

b) Man in the Middle
Attack <u>Man in the Middle</u>
<u>Program</u> import socket import
random as r

def mitm():   host =
socket.gethostname()   port =
5000   s = socket.socket()
s.bind((host, port))
s.listen(10)   alice, address1 = s.accept()
bob, address2 = s.accept()
print("Connection from: " + str(address1))
print("Connection from: " + str(address2))

p = int(alice.recv(1024).decode('ascii'))   print("p = ", p)   g =
int(alice.recv(1024).decode('ascii'))   print("g = ", g)

bob.send(str(p).encode('ascii'))
bob.send(str(g).encode('ascii'))

Xa =
int(alice.recv(1024).decode('ascii'))
print("Xa from Alice = ", Xa)   e =
r.randint(3, 1000)   Xe = int(pow(g, e,
p))   bob.send(str(Xe).encode('ascii'))

```
    Xb =
int(bob.recv(1024).decode('ascii'))
print("Xb from Bob = ", Xb)     f =
r.randint(3, 1000)     Xf = int(pow(g, f,
p))     alice.send(str(Xf).encode('ascii'))

    Ak = int(pow(Xa, f, p))     Bk =
int(pow(Xb, e, p))     print("Key
generated by Alice = ", Ak)
print("Key generated by Bob = ", Bk)

mitm()
```

Alice Program

```
import socket import
random as r

def alice():     host =
socket.gethostname()      port =
5000      s = socket.socket()
    s.connect((host, port))

    p = int(input("Enter p = "))
g = int(input("Enter g = "))
    s.send(str(p).encode('ascii'))
    s.send(str(g).encode('ascii'))

    a = r.randint(3, 1000)     Xa
= int(pow(g, a, p))
print("Xa computed = ", Xa)
    s.send(str(Xa).encode('ascii'))

    Xb = int(s.recv(1024).decode('ascii'))
print("Xb from Man in the middle = ", Xb)

    Ak = int(pow(Xb, a, p))     print('Secret
key for Alice is = %d' % (Ak))     s.close()

alice()
```

Bob Program import
socket import
random as r

```
def bob():     host =
socket.gethostname()      port =
5000      s = socket.socket()
    s.connect((host, port))

    p =
int(s.recv(1024).decode('ascii'))
print("p = ", p)      g =
int(s.recv(1024).decode('ascii'))
print("g = ", g)
    Xa = int(s.recv(1024).decode('ascii'))
print("Xa from Man in the middle = ", Xa)

    b = r.randint(3, 1000)      Xb
= int(pow(g, b, p))
print("Xb computed = ", Xb)
    s.send(str(Xb).encode('ascii'))

    Bk = int(pow(Xa, b, p))      print('Secret
key for Bob is = %d' % (Bk))      s.close()

bob()
```

INPUT AND OUTPUT

a) Diffie Hellman Key exchange

```
PS D:\Riya\DJ Sanghvi\Information and Network System\Experiments\Experiment8> python alice.py
Connection from: ('192.168.29.186', 54094)
Enter p = 23
Enter g = 9
Xa computed =  18
Xb from Bob =  12
Secret key for Alice is = 2
```

```
PS D:\Riya\DJ Sanghvi\Information and Network System\Experiments\Experiment8> python bob.py
p =  23
g =  9
Xa from Man in the middle =  18
Xb computed =  12
Secret key for Bob is = 2
```

b) Man in the Middle Attack

```
PS D:\Riya\DJ Sanghvi\Information and Network System\Experiments\Experiment8> python mitm.py
Connection from: ('192.168.29.186', 53986)
Connection from: ('192.168.29.186', 53987)
p =  23
g =  9
Xa from Alice =  2
Xb from Bob =  16
Key generated by Alice =  3
Key generated by Bob =  12
```

```
PS D:\Riya\DJ Sanghvi\Information and Network System\Experiments\Experiment8> python alice.py
Enter p = 23
Enter g = 9
Xa computed =  2
Xb from Man in the middle =  13
Secret key for Alice is = 3
```

```
PS D:\Riya\DJ Sanghvi\Information and Network System\Experiments\Experiment8> python bob.py
p =  23
g =  9
Xa from Man in the middle =  13
Xb computed =  16
Secret key for Bob is = 12
```

CONCLUSION

Thus, we have successfully implemented Diffie Hellman Key exchange protocol and demonstrated man in middle attack.

Experiment 9

Date of Performance : 17-04-2023 **Date of Submission : 07-05-2023**

SAP Id: 60004200107 **Name :** Kartik Jolapara

Div: B **Batch :** B1

Aim of Experiment

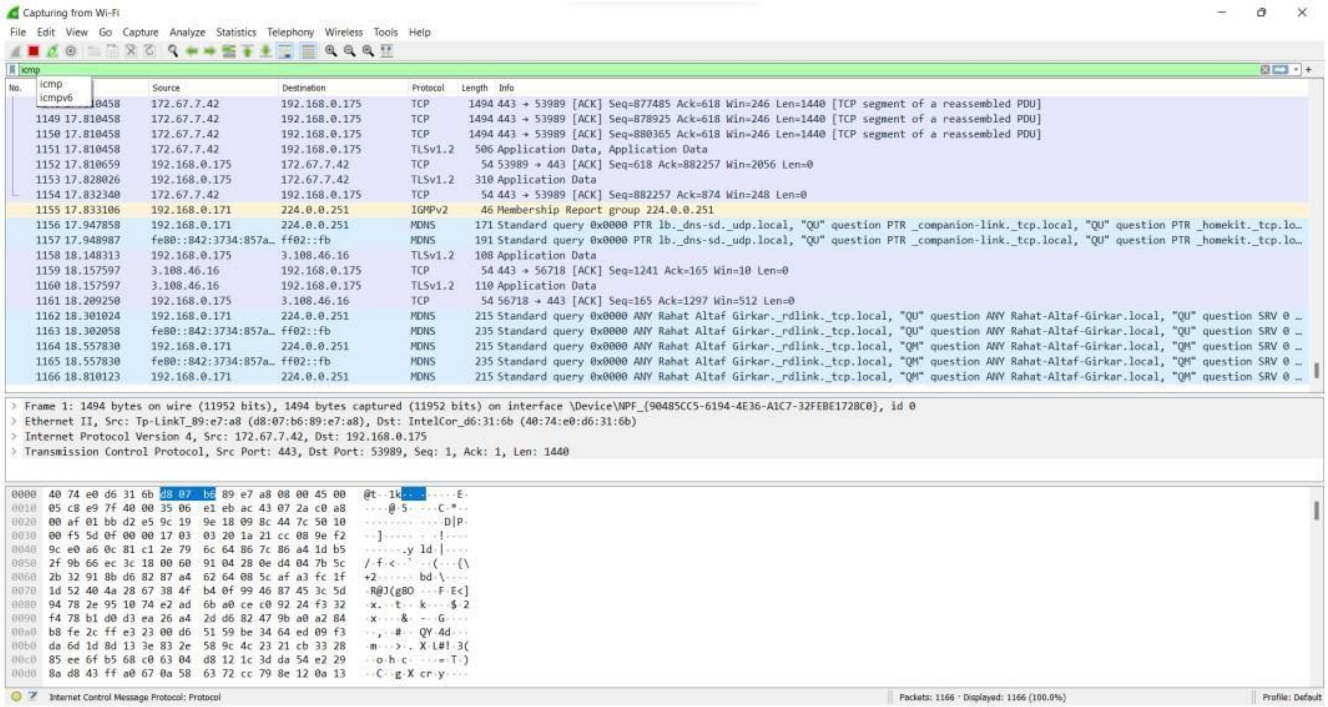
Study of packet sniffer tools: WireShark Download and install wireshark and captue icmp, tcp and http packets in promiscuous mode Explore how the packets can be traced based on different filters.

Theory / Algorithm / Conceptual Description

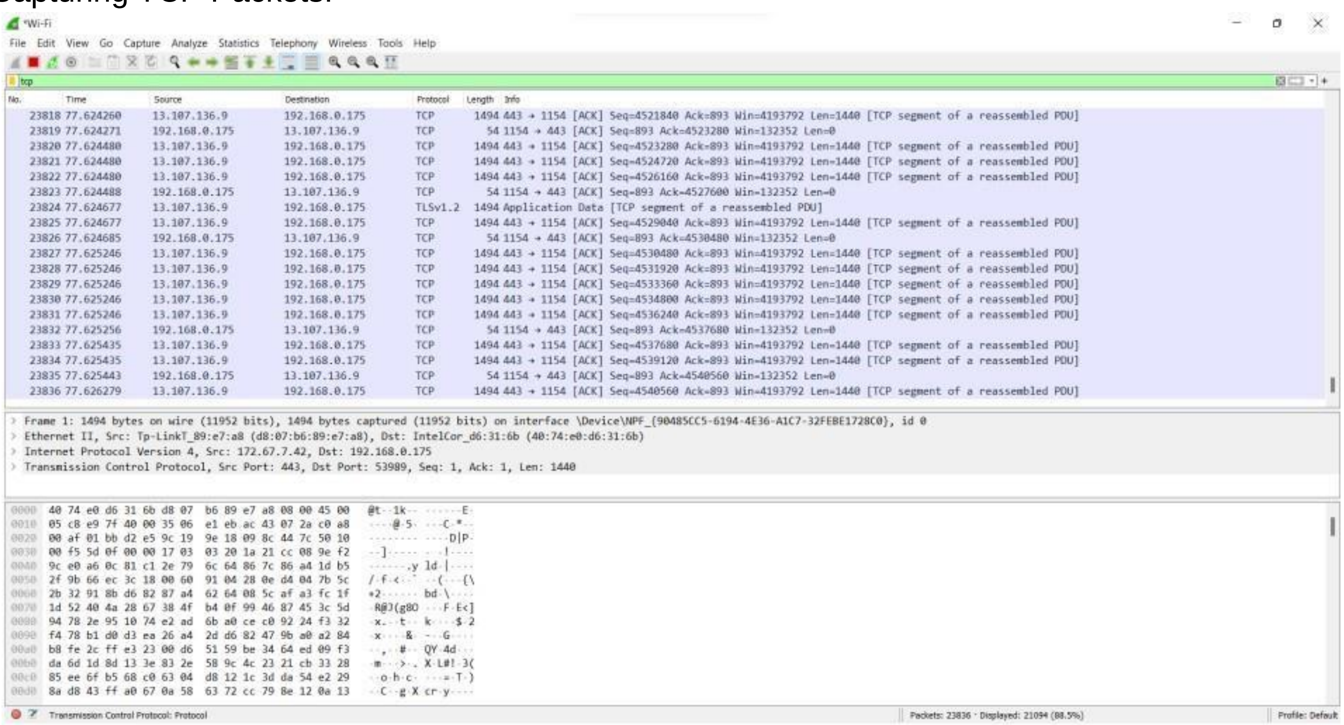
Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Wireshark lets the user put network interface controllers into promiscuous mode (if supported by the network interface controller), so they can see all the traffic visible on that interface including unicast traffic not sent to that network interface controller's MAC address. However, when capturing with a packet analyzer in promiscuous mode on a port on a network switch, not all traffic through the switch is necessarily sent to the port where the capture is done, so capturing in promiscuous mode is not necessarily sufficient to see all network traffic. Port mirroring or various network taps extend capture to any point on the network. Simple passive taps are extremely resistant to tampering.

Capturing ICMP Packets:

C:\Users\Marwin Shroff>ping 8.8.8.8 Pinging
8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=5ms TTL=119
Reply from 8.8.8.8: bytes=32 time=6ms TTL=119
Reply from 8.8.8.8: bytes=32 time=2ms TTL=119
Reply from 8.8.8.8: bytes=32 time=3ms TTL=119 Ping
statistics for 8.8.8.8:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 6ms, Average = 4ms



Capturing TCP Packets:



Capturing FTP Packets:

C:\Users\Marwin Shroff>ftp ftp.cdc.gov Connected to ftp.cdc.gov.

220 Microsoft FTP Service

200 OPTS UTF8 command successful - UTF8 encoding now ON.

User (ftp.cdc.gov:(none)): anonymous

331 Anonymous access allowed, send identity (e-mail name) as password.

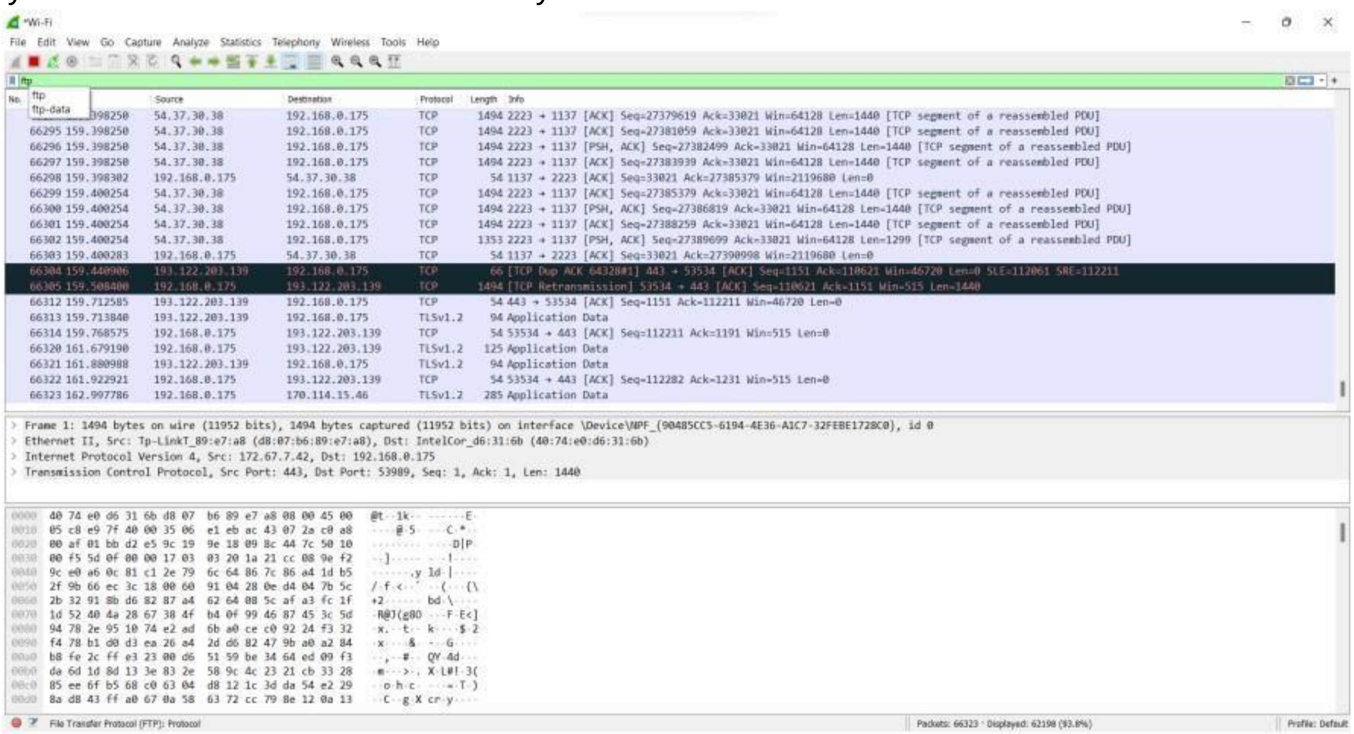
Password: 230 User
logged in.

ftp> ls

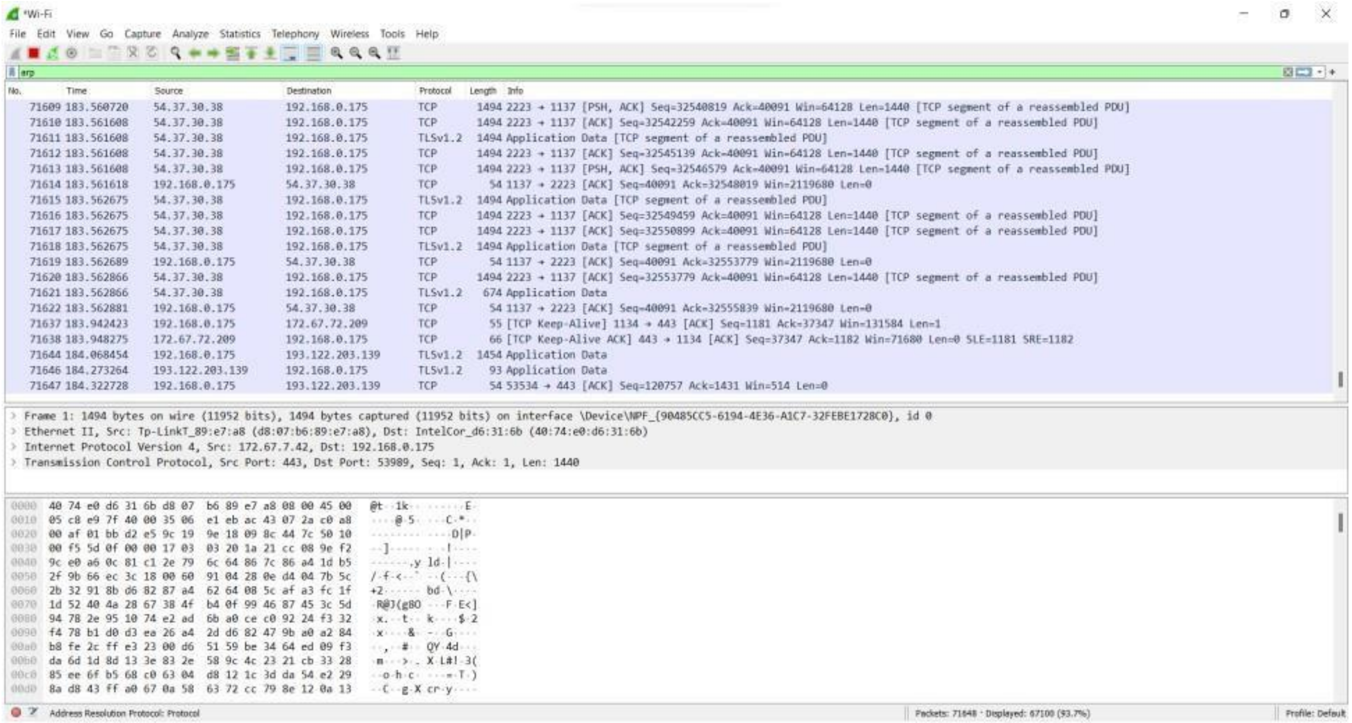
200 PORT command successful.

150 Opening ASCII mode data connection.

.change.dir
.message
pub Readme
Siteinfo w3c welcome.msg 226 Transfer
complete. ftp: 67
bytes received in 0.03Seconds 2.03Kbytes/sec.



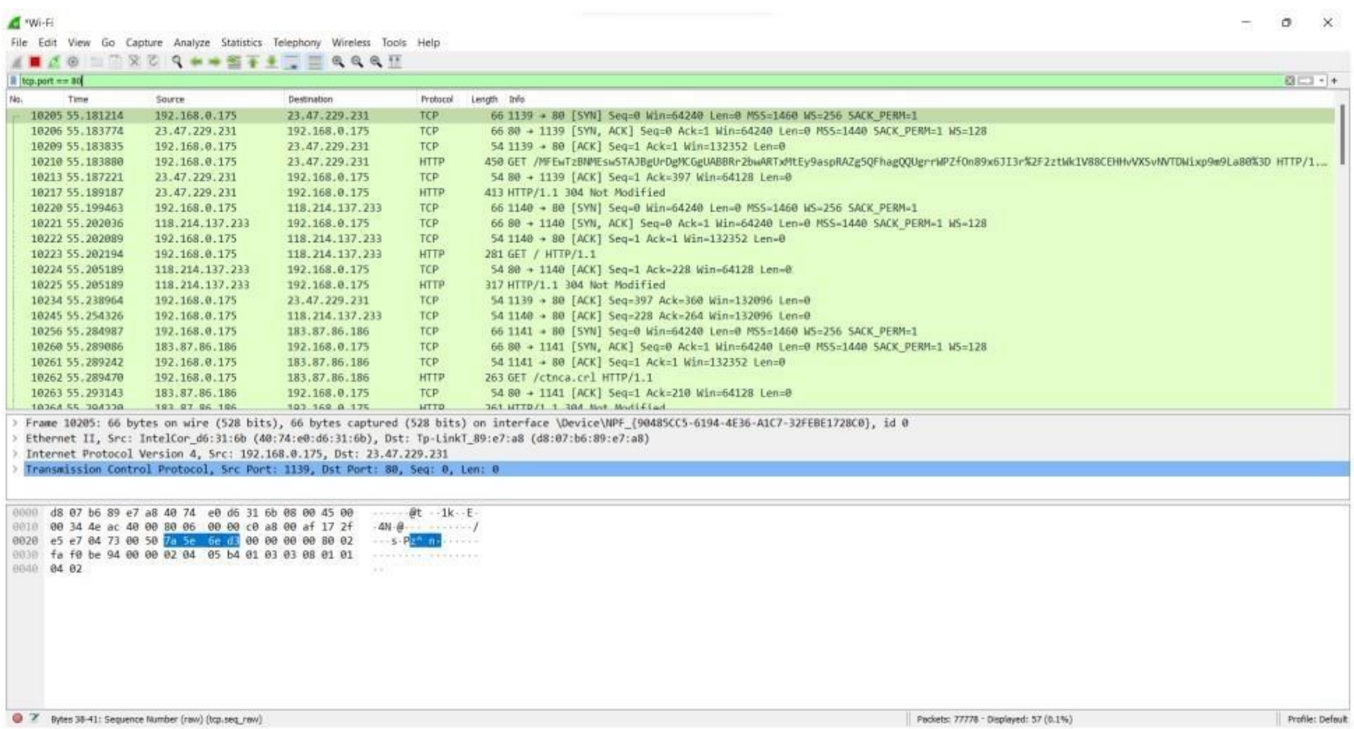
Capturing ARP Packets:



B] Tracing Packets based on filters:

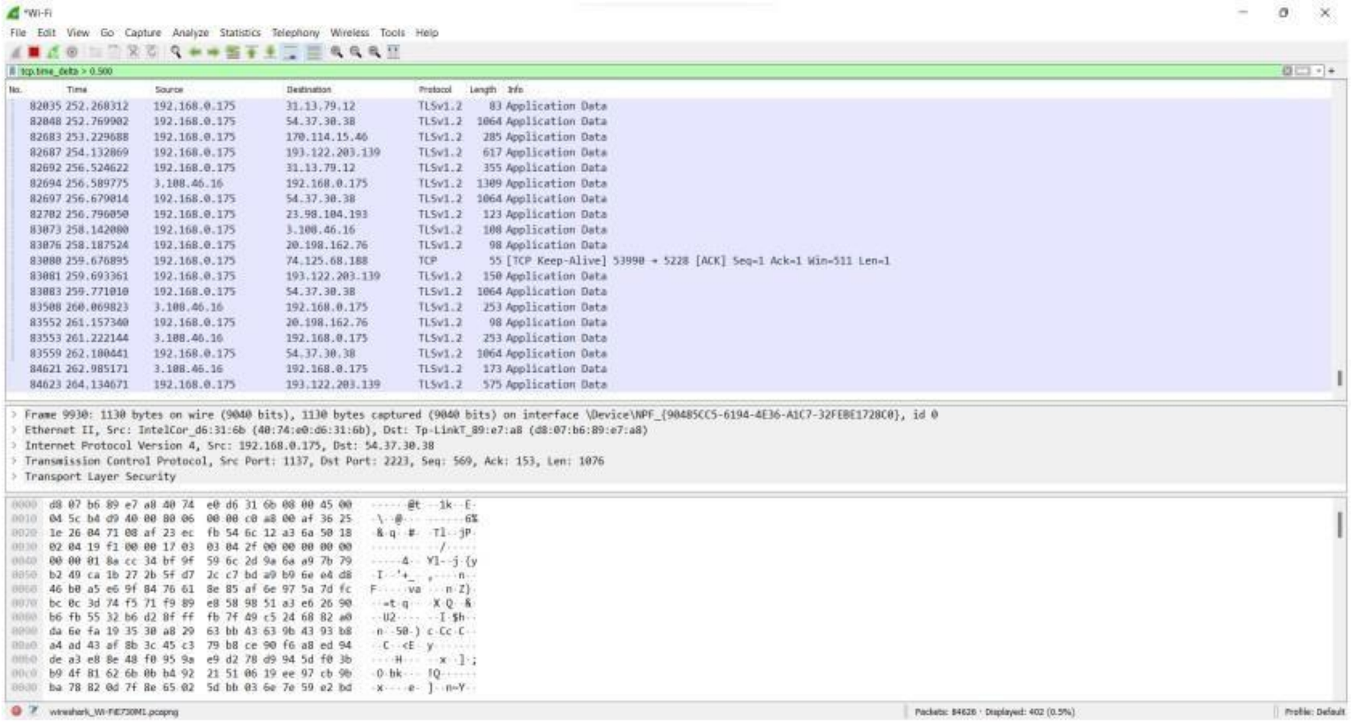
1] Filter Results by Port:

Traces all packets related to Port 80.



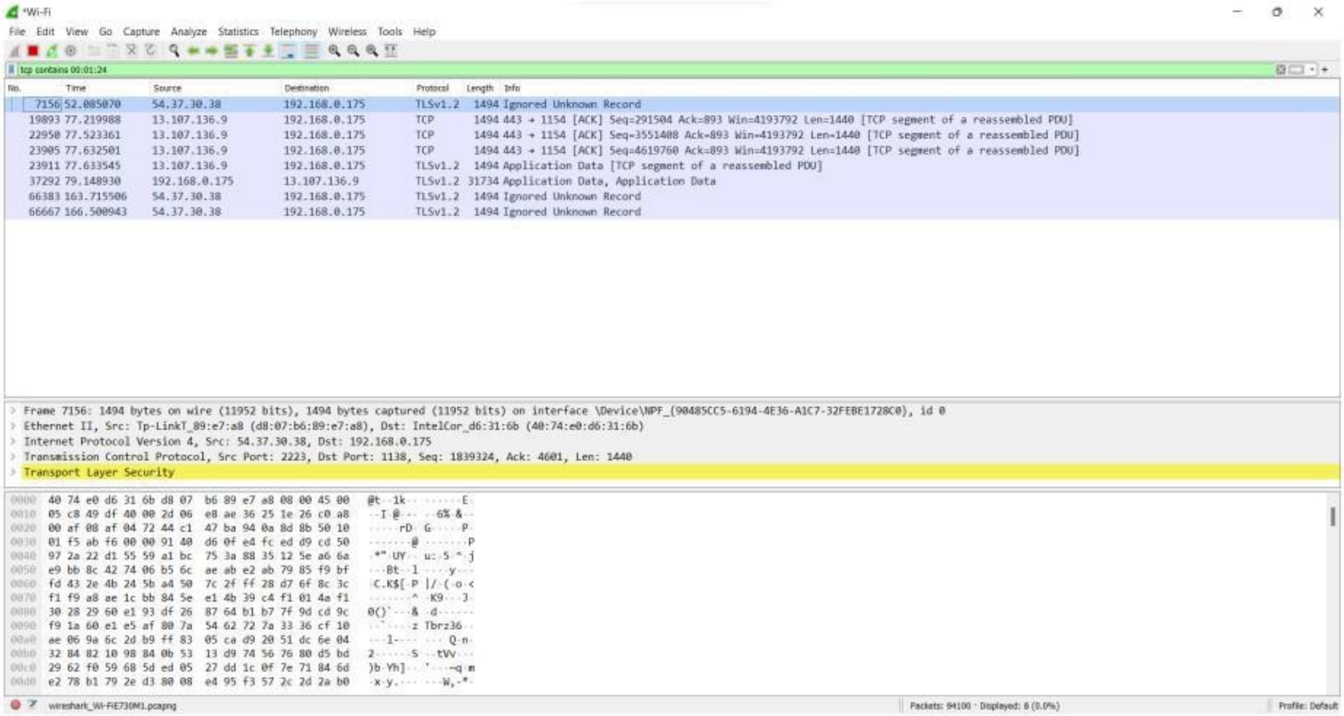
2] Filter by Delta Time :

Displays tcp packets with delta time of greater than 0.500 sec



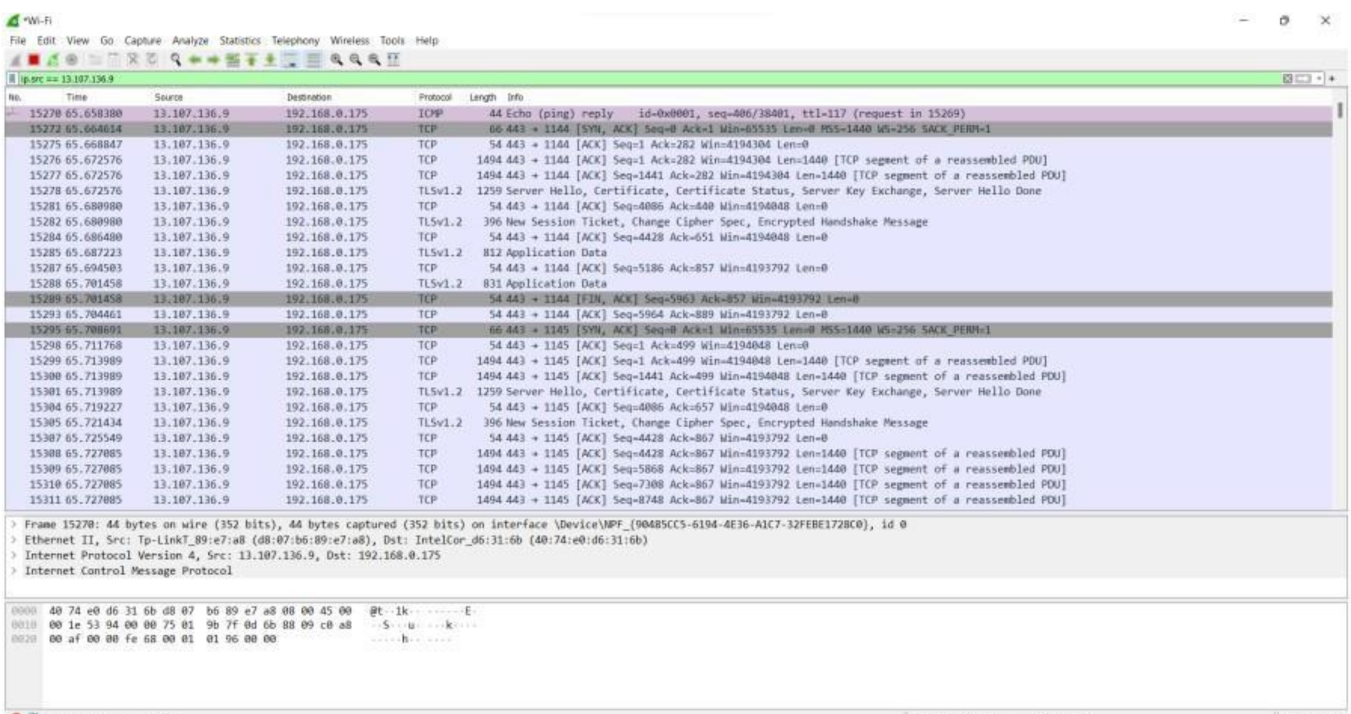
3] Filter by Byte Sequence:

Displays packets which contain a particular byte sequence.



4] Filter by Source IP Address:

Displays packets which have source IP address same as the one provided in the argument.



CONCLUSION

Thus, we have successfully studied packet sniffing tools (wireshark) and explored how packets can be traced on the basis of different filters.

Experiment 10

Date of Performance : 06-03-2023 **Date of Submission: 07-03-2023**

SAP Id: 60004200107 **Name :** Kartik Jolapara

Div: B **Batch :** B1

Aim of Experiment

Implement Buffer Overflow Attack.

Theory / Algorithm / Conceptual Description

Buffer Overflow Attack: Attackers exploit buffer overflow issues by overwriting the memory of an application. This changes the execution path of the program, triggering a response that damages files or exposes private information. For example, an attacker may introduce extra code, sending new instructions to the application to gain access to IT systems. If attackers know the memory layout of a program, they can intentionally feed input that the buffer cannot store, and overwrite areas that hold executable code, replacing it with their own code. For example, an attacker can overwrite a pointer (an object that points to another area in memory) and point it to an exploit payload, to gain control over the program. Stack-based buffer overflows are more common, and leverage stack memory that only exists during the execution time of a function. Heap-based attacks are harder to carry out and involve flooding the memory space allocated for a program beyond memory used for current runtime operations.

Ollydbg: OllyDbg (named after its author, Oleh Yuschuk) is an x86 debugger that emphasizes binary code analysis, which is useful when source code is not available. It traces registers, recognizes procedures, API calls, switches, tables, constants and strings, as well as locates routines from object files and libraries. It has a user friendly interface, and its functionality can be extended by third-party plugins. OllyDbg is often used for reverse engineering of programs. It is often used by crackers to crack software made by other developers. For cracking and reverse engineering, it is often the primary tool because of its ease of use and availability; any 32-bit executable can be used by the debugger and edited in bitcode/assembly in realtime.It is also useful for programmers to ensure that their program is running as intended, and for malware analysis purposes.

Splint: Splint is a tool for statically checking C programs for security vulnerabilities and coding mistakes. With minimal effort, Splint can be used as a better lint. If additional effort is invested adding annotations to programs, Splint can perform stronger checking than can be done by any standard lint. Splint has the ability to interpret special annotations to the source code, which gives it stronger checking than is possible just by looking at the source alone.
Splint is used by gpsd as part of an effort to design for zero defects.

Cppcheck: Cppcheck is a static code analysis tool for the C and C++ programming languages. It is a versatile tool that can check non-standard code. Cppcheck supports a wide variety of static checks that may not be covered by the compiler itself. These checks are static analysis checks that can be performed at a source code level. The program is directed towards static analysis checks that are rigorous, rather than heuristic in nature. Some of the checks that are supported include: • Automatic variable checking

• Bounds checking for array overruns

• Classes checking (e.g. unused functions, variable initialization and memory duplication)

<u>Program</u>

Code with Buffer Overflow

```
#include <stdio.h>
#include <string.h>
#define UP_MAXLEN 20
#define UP_PAIR_COUNT 3
int main() { int flag; char termBuf; char
username[UP_MAXLEN]; char
cpass[UP_MAXLEN]; char npass[UP_MAXLEN];
char keys[UP_PAIR_COUNT][2][UP_MAXLEN] =
{
{
"Admin",
"pass3693"
},
{
"Marwin",
"60004200097"
},
{
"Sally",
"Usfsmfs"
}
};
while (1) { flag = 0;
printf("Change
Password\n"); printf("Enter
Username: ");
```

```
gets(username); printf("Enter
Current Password: ");
gets(cpass);
for (int i = 0; i < UP_PAIR_COUNT; i++) { if
(strcmp(keys[i][0], username) == 0 && strcmp(keys[i][1],
cpass) == 0) { printf("Enter New Password: ");
gets(npass);
strcpy( & keys[i][1][0], npass); for (int j = 0; j <
UP_PAIR_COUNT; j++) printf("%s |
%s\n", keys[j][0], keys[j][1]);
printf("Password
Changed!\n");
printf("Continue? Y/N: "); gets(
& termBuf); if (termBuf != 'Y')
return 0;
else flag = 1;
}
}
if (flag == 1) continue; printf("Incorrect Username
and Password. Enter Y to continue.\n"); gets( &
termBuf);
if (termBuf != 'Y') return 0;
}
}
```

Output:

```
Change Password
Enter Username: Marwin
Enter Current Password: 60004200097
Enter New Password: hellothisismarwin
Admin | pass3693
Marwin | hellothisismarwin
Sally | Usfsmfs
Password Changed!
Continue? Y/N: Y
Change Password
Enter Username: Marwin
Enter Current Password: hdvghdsgf
Incorrect Username and Password. Enter Y to continue.
Y
```

<u>Program</u>

Code after fixing the Buffer Overflow Vulnerability

```
#include <stdio.h>
#include <string.h>
#define UP_MAXLEN 20
#define UP_PAIR_COUNT 3
int main() { int flag; char termBuf; char
username[UP_MAXLEN]; char
cpass[UP_MAXLEN]; char npass[UP_MAXLEN];
char keys[UP_PAIR_COUNT][2][UP_MAXLEN] =
{
{
"Admin",
"pass3693"
},
{
"Max",
"Qqkaif"
},
{
"Sally",
"Usfsmfs"
}
};
while (1) { flag = 0; printf("Change
Password\n"); printf("Enter Username:
"); fgets(username, UP_MAXLEN,
stdin); username[strcspn(username,
"\r\n")] = 0; printf("Enter Current
Password: "); fgets(cpass,
UP_MAXLEN, stdin);
```

```
cpass[strcspn(cpass, "\r\n")] = 0; for (int i = 0; i <
UP_PAIR_COUNT; i++) { if (strcmp(keys[i][0], username)
== 0 && strcmp(keys[i][1], cpass) == 0) { printf("Enter
New Password: "); fgets(npass, UP_MAXLEN, stdin);
npass[strcspn(npass, "\n")] = 0; strcpy( & keys[i][1][0],
npass); for (int j = 0; j < UP_PAIR_COUNT; j++)
printf("%s |
%s\n", keys[j][0], keys[j][1]); printf("Password
Changed!\n"); printf("Continue? Y/N: "); scanf("%c", &
termBuf); if (termBuf != 'Y') return 0; else flag = 1;
while ((termBuf = getchar()) != '\n' && termBuf !=
EOF);
}
}
if (flag == 1) continue; printf("Incorrect Username and
Password. Enter Y to continue.\n"); scanf("%c", &
termBuf); if (termBuf != 'Y') return 0; while ((termBuf =
getchar()) != '\n' && termBuf != EOF);
}
}
```

Output:

```
Change Password
Enter Username: Max
Enter Current Password: Qqkaif
Enter New Password: marwinisgod
Admin | pass3693
Max | marwinisgod
Sally | Usfsmfs
Password Changed!
```

CONCLUSION

Thus, Buffer Overflow Attack has been successfully demonstrated and prevented using the Splint programming tool