



Experiment 1

Date of Performance :

Date of Submission:

SAP Id: 60004190057

Name : Junaid Altaf Girkar

Div: A

Batch : A4

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(). Demonstrate the use of these functions on any paragraph.

Affine Cipher. Your Program Must Input Image in Gray Scale. Choose keys according to Gray Scale Intensity level. Create two functions Encrypt() and Decrypt(). Make sure to have Multiplicative Inverse Exists for one of the Key in selected Key pair of Affine Cipher.

(CO1)

Theory / Algorithm / Conceptual Description

CAESAR CIPHER

The Caesar Cipher technique is one of the earliest and simplest method of encryption technique. It's simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter some 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.

Thus to cipher a given text we need an integer value, known as shift which indicates the number of position each letter of the text has been moved down.

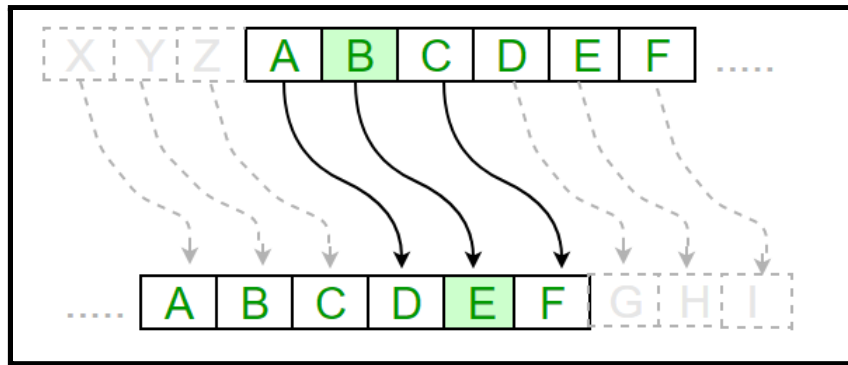
The encryption can be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, A = 0, B = 1,..., Z = 25. Encryption of a letter by a shift n can be described mathematically as.

$$E_n(x) = (x+n) \bmod 26$$

(Encryption Phase with shift n)

$$D_n(x) = (x-n) \bmod 26$$

(Decryption Phase with shift n)



HILL CIPHER

Hill cipher is a polygraphic substitution cipher based on linear algebra. Each letter is represented by a number modulo 26. Often the simple scheme $A = 0, B = 1, \dots, Z = 25$ 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 $n \times n$ 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 $n \times n$ matrices

Encryption

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix} = \begin{bmatrix} 67 \\ 222 \\ 319 \end{bmatrix} \equiv \begin{bmatrix} 15 \\ 14 \\ 7 \end{bmatrix} \pmod{26}$$

Decryption

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix}^{-1} \equiv \begin{bmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{bmatrix} \pmod{26}$$

Program

A)

```
# Encryption and decryption of a message using a caesar cipher

def encrypt(message, key):
```

```

    encrypted_message = ""
    for letter in message:
        if letter.isupper():
            encrypted_message += chr((ord(letter) + key - 64) % 26 +
65)
        else:
            encrypted_message += chr((ord(letter) + key - 96) % 26 +
97)
    return encrypted_message

plain_text = "UkraineIsACountryInEasternEurope"
key = 5

print("PLain text: ", plain_text)
print("Key: ", key)
print("Cipher Text : " + encrypt(plain_text, key))

def decrypt(cipher_text, key):
    decrypted_message = ""
    for letter in cipher_text:
        if letter.isupper():
            decrypted_message += chr((ord(letter) + key - 65) % 26 +
65)
        else:
            decrypted_message += chr((ord(letter) + key - 97) % 26 +
97)
    return decrypted_message

def brute_force_decrypt(cipher_text):
    for i in range(26):
        print("Key: ", abs(25 - i))
        print("Decrypted Text: " + decrypt(cipher_text, i))

brute_force_decrypt(encrypt(plain_text, key))

```

Output

```

Plain text: UkrainelsACountryInEasternEurope
Key: 5
Cipher Text : AqxgotkOyGluatzxeOtKgyzkxtKaxuvk
Key: 25
Decrypted Text: AqxgotkOyGluatzxeOtKgyzkxtKaxuvk

```

Key: 24
Decrypted Text: BryhpulPzHJvbuayfPuLhzalyuLbyvwl
Key: 23
Decrypted Text: CszivmQaIKwcvbzgQvMiabmzvMczwxm
Key: 22
Decrypted Text: DtajrwnRbJLxdwcahRwNjbcnawNdaxyn
Key: 21
Decrypted Text: EubksxoScKMyexdbiSxOkcdobxOebyzo
Key: 20
Decrypted Text: FvcltypTdLNzfyecjTyPldepcyPfczap
Key: 19
Decrypted Text: GwdmuzqUeMOagzfdkUzQmefqdzQgdabq
Key: 18
Decrypted Text: HxenvarVfNPbhagelVaRnfgreaRhebcr
Key: 17
Decrypted Text: lyfowbsWgOQcibhfmWbSoghsfbSifcds
Key: 16
Decrypted Text: JzgpuxtXhPRdjignXcTphitgcTjgdet
Key: 15
Decrypted Text: KahqyduYiQSekdjhoYdUqijuhdUkhefu
Key: 14
Decrypted Text: LbirzevZjRTflekipZeVrjkvieVlifgv
Key: 13
Decrypted Text: McjsafwAkSUGmfliqAfWsklwjfwMjghw
Key: 12
Decrypted Text: NdktbgxBITVhngmkrBgXtlmxkgXnkhix
Key: 11
Decrypted Text: OeluchyCmUWiohnlSChYumnylhYolijy
Key: 10
Decrypted Text: PfmvdizDnVXjpiomtDiZvnozmiZpmjkz
Key: 9
Decrypted Text: QgnwejaEoWYkqjpnuEjAwopanJAqnkla
Key: 8
Decrypted Text: RhoxfkbFpXZlrqovFkBxpbokBrolmb
Key: 7
Decrypted Text: SipyglcGqYAmslrpwGICyqrcplCspmnc
Key: 6
Decrypted Text: TjqzhmdHrZBntmsqxHmDzrsdqmDtqnod
Key: 5
Decrypted Text: UkrainelsACountryInEasternEurope
Key: 4
Decrypted Text: VlsbjofJtBDpvouszJoFbtufsoFvspqf
Key: 3
Decrypted Text: WmtckpgKuCEqwpvtaKpGcuvgtgGwtqrg
Key: 2
Decrypted Text: XnudlqhLvDFrxqwubLqHdvwhuqHxursh
Key: 1
Decrypted Text: YovemriMwEGsyrxvcMrlewxivrvlysti
Key: 0
Decrypted Text: ZpwfnsjNxFHtzsywdNsJfxyjwsJzwtuj

Program

B)

```
from scipy import misc
import imageio
import numpy as np
import matplotlib.pyplot as plt
import os.path
import pickle
from numpy.linalg import inv, det
import sys
import scipy.misc

# IMAGE SECTION
def read_image(image_path):
    """ Read an image and return a one hot vector of the image"""
    img = imageio.imread(image_path)
    reshape_value = 1

    for i in img.shape:
        reshape_value *= i

    return img.reshape((1, reshape_value)), img.shape

def show_image(image):
    """ Show a single image"""
    plt.imshow(image)
    plt.show()

def show_images(a, b):
    """ Show two images side by side"""
    plot_image = np.concatenate((a, b), axis=1)
    plt.imshow(plot_image)
    plt.show()

# HILL CLIMB SECTION

class HillClimb:
    def __init__(self, data, file_name, key_path=None):

        self.data = data
```

```

# Computet the chunk
self.chunk = self.computer_chunk()

if key_path:
    # Load the key if it exist in the current dir
    self._key = pickle.load(open( key_path, "rb" ))
    print('Usigng the args -k ' + key_path)
else:
    file_name = file_name + '.key'

    if os.path.isfile(file_name):
        # Load the key if it exist in the current dir
        self._key = pickle.load(open( file_name, "rb" ))
        print('Usigng the ' + file_name)
    else:
        # Generate a random key
        self._key = np.random.random_integers(0, 100,
(self.chunk, self.chunk))

        # If determinat is equal to zero regenrate another
key
        if det(self._key) == 0:
            self._key = np.random.random_integers(0, 100,
(self.chunk, self.chunk))

        # Save the key in a pickle
        pickle.dump( self._key, open( file_name, "wb" ) )

print(self._key.dtype)
print(self._key.shape)
print(self._key)

# Get the inverse of the key
self.reversed_key = np.matrix(self._key).I.A

print(self.reversed_key.dtype)
print(self.reversed_key.shape)
print(self.reversed_key)

def computer_chunk(self):
    max_chunk = 100
    data_shape = self.data.shape[1]
    print(data_shape)

```

```

    for i in range(max_chunk, 0, -1):
        if data_shape % i == 0:
            return i

@property
def key(self):
    return self._key

def encode(self, data):
    """ Encode function """
    crypted = []
    chunk = self.chunk
    key = self._key

    for i in range(0, len(data), chunk):

        temp = list(np.dot(key, data[i:i + chunk]))
        crypted.append(temp)

    crypted = (np.array(crypted)).reshape((1, len(data)))
    return crypted[0]

def decode(self, data):
    """ Decode function """
    uncryptd = []
    chunk = self.chunk
    reversed_key = self.reversed_key

    for i in range(0, len(data), chunk):
        temp = list(np.dot(reversed_key, data[i:i + chunk]))
        uncryptd.append(temp)

    uncryptd = (np.array(uncryptd)).reshape((1, len(data)))

    return uncryptd[0]

```

```

import pickle
from numpy.linalg import inv, det
import sys
import scipy.misc
from HillClimb import HillClimb
from HillClimb import *
import imageio

def transform(np_array, shape):
    return np_array.reshape(shape).astype('uint8')

if __name__ == '__main__':
    if len(sys.argv) > 1:
        image_file_name = sys.argv[1]
    else:
        raise Exception('Missing image file name')

    img, original_shape = read_image(image_file_name)
    hill = HillClimb(data=img, file_name=image_file_name)

    ### Testing zone
    print(img.shape)

    # ----- Encoding -----

    # Get the encoded vector image
    encoded_image_vector = hill.encode(img[0])

    # Reshape to the original shape of the image
    encoded_image = encoded_image_vector.reshape(original_shape)

    # Show the decoded image
    # show_image(encoded_image.astype('uint8'))

    # Setup the encoded file name to be used when saving the encoded
    image
    img_name = image_file_name.split('.')[0]
    img_extension = image_file_name.split('.')[1]
    encoded_img_name = '{0}-encoded.{1}'.format(img_name,
img_extension)

```



```

    # Convert to uint8
    encoded_image = encoded_image.astype('uint8')

    # Save the image
    imageio.imsave(encoded_img_name, encoded_image)

    # Save the image as a pickle model
    pickle.dump(encoded_image_vector, open( encoded_img_name + '.pk',
"wb" ))

# # ----- Decoding -----

img_vector = pickle.load(open(encoded_img_name + '.pk', 'rb'))

# Get the decoded vector image
decoded_image_vector = hill.decode(img_vector)

# Reshape to the original shape of the image
decoded_image = decoded_image_vector.reshape(original_shape)

decoded_img_name = '{0}-decoded.{1}'.format(img_name,
img_extension)

# Save the image
imageio.imsave(decoded_img_name, decoded_image)

```

Output

```

722775
Using the Hello.jpg.key
int32
(75, 75)
[[ 2 91 29 ... 12 27 46]
 [ 8 57 87 ... 75 84 63]
 [50 18 84 ...  2 25 86]
...
```

```

[ 77 91 41 ... 100 75 22]
[ 17 54 53 ... 80 69 62]
[ 91 66 16 ... 35 28 83]]
float64
(75, 75)
[[-0.00260477 0.00198561 0.00334513 ... 0.00091603 0.00228508
  0.00486007]
 [ 0.00694458 -0.00346724 0.00111209 ... 0.00046972 -0.00580259
 -0.00113174]
 [ 0.01007937 -0.01406634 -0.00100486 ... -0.00049209 -0.01050542
  0.0045447 ]
 ...
 [-0.00092501 0.0058364 0.00134957 ... 0.00548127 0.00062253
  0.00192339]
 [-0.00492509 0.00583449 -0.00210589 ... -0.00322735 0.00416567
 -0.00583711]
 [-0.00516961 0.0075935 0.00729953 ... -0.00127429 0.00345992
 -0.0035384 ]]
(1, 722775)

```



CONCLUSION

With the increasing amount of data being generated, it is very important that confidential information does not get leaked and is read by the intended recipient. We learnt about the different encryption techniques and different ciphers. We then wrote a python program which implemented Caesar Cipher and Hill Cipher.



Experiment 2

Date of Performance :

Date of Submission:

SAP Id: 60004190057

Name : Junaid Altaf Girkar

Div: A

Batch : A4

Aim of Experiment

Design and Implement Encryption and Decryption Algorithm for Columnar Transposition Cipher.

Theory / Algorithm / Conceptual Description

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

Encryption

1. In a transposition cipher, the order of the alphabets is rearranged to obtain the cipher-text.
2. 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.
3. Width of the rows and the permutation of the columns are usually defined by a keyword.
4. 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".
5. Any spare spaces are filled with nulls or left blank or placed by a character (Example: _).
6. Finally, the message is read off in columns, in the order specified by the keyword.
7. columnar-transposition-cipher

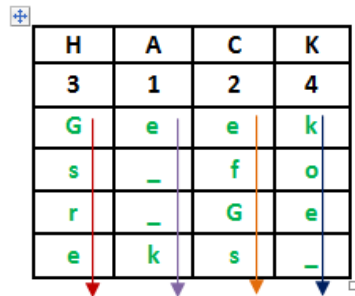
Encryption

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_

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 reorder the columns by reforming the key word.

CODE:

```
import math

key = "DJSCE"

# Encryption
def encryptMessage(msg):
    cipher = ""
    key_index = 0

    msg_len = float(len(msg))
    msg_lst = list(msg)
    key_lst = sorted(list(key))

    col = len(key)

    row = int(math.ceil(msg_len / col))
```

```

fill_null = int((row * col) - msg_len)
msg_lst.extend('_' * fill_null)

matrix = [msg_lst[i: i + col]
           for i in range(0, len(msg_lst), col)]

for _ in range(col):
    curr_idx = key.index(key_lst[key_index])
    cipher += ".join([row[curr_idx]
                      for row in matrix])

    key_index += 1

return cipher

```

Decryption

```

def decryptMessage(cipher):
    decrypted_message = ""

    key_index = 0
    msg_indx = 0
    msg_len = float(len(cipher))
    msg_lst = list(cipher)

    col = len(key)

    row = int(math.ceil(msg_len / col))
    key_lst = sorted(list(key))

    deciphered_cipher_message = []
    for _ in range(row):
        deciphered_cipher_message += [[None] * col]

    for _ in range(col):
        curr_idx = key.index(key_lst[key_index])

        for j in range(row):
            deciphered_cipher_message[j][curr_idx] = msg_lst[msg_indx]
            msg_indx += 1
            key_index += 1

    try:

```

```

        decrypted_message = ".join(sum(deciphered_cipher_message, []))
except TypeError:
    raise TypeError("This program cannot",
                    "handle repeating words.")

null_count = decrypted_message.count('_')

if null_count > 0:
    return decrypted_message[: -null_count]

return decrypted_message

msg = "Junaid Girkar"
print("\nPlaintext Message:", msg)

cipher = encryptMessage(msg)
print("\nCiphertext Message: {}".format(cipher))

decrypted_message = decryptMessage(cipher)
print("\nDecryped Message: {}\n".format(decrypted_message))

```

OUTPUT:

```

Plaintext Message: Junaid Girkar

Ciphertext Message: ai_Jdkir_u anGr

Decryped Message: Junaid Girkar

```

CODE:

```

import cv2
from google.colab.patches import cv2_imshow
import numpy as np

plaintext = list('harrydidyoutputyournameinthegobletoffire')
key = 'junaidabcd'

def encryption(message, key):

```

```

ciphertext = []
# message_list = list(message)
message_length = len(message)
sorted_key = sorted(list(key))
column = len(key)
row = int(ceil(message_length/column))
dummy_characters = (row*column) - message_length
message.extend('*'*dummy_characters)
matrix = list()
counter = 0
for i in range(row):
    temp = list()
    for j in range(column):
        temp.append(message[counter])
        counter += 1
    matrix.append(temp)
counter = 0
for j in range(column):
    index = key.index(sorted_key[counter])
    # ciphertext += ''.join([row[index] for row in matrix])
    for row in matrix:
        ciphertext.append(row[index])
    counter += 1
return ciphertext
ciphertext = encryption(plaintext, key)
print('For given plaintext : ' + ''.join(plaintext) + ' the
corresponding ciphertext is : ' + ''.join(ciphertext))

def decryption(ciphertext, key):
    plaintext = []
    message_length = len(ciphertext)
    # message = list(ciphertext)
    sorted_key = sorted(list(key))
    column = len(key)
    row = int(message_length/column)
    counter = 0
    matrix = list()
    for j in range(row):
        matrix.append(['']*column)
    for j in range(column):
        index = key.index(sorted_key[j])
        for i in range(row):

```



```

        matrix[i][index] = ciphertext[counter]
        counter+=1
    for i in range(row):
        for j in range(column):
            if matrix[i][j] != '*':
                plaintext.append(matrix[i][j])
    return plaintext
plaintext = decryption(ciphertext,key)
print('For given ciphertext : ' + ''.join(ciphertext) + ' the
corresponding plaintext is : ' + ''.join(plaintext))

key = "authorized"
image = cv2.imread("Brocolli.jpg",0)
image = np.array(image,dtype=np.int64)
cv2_imshow(image)

shape = image.shape
flat_image = list(image.flatten())

encrypted = encryption(flat_image,key)
encrypted = np.array(encrypted,dtype=np.int64)
encrypted = encrypted.reshape(shape)
cv2_imshow(encrypted)

encrypted_flat_image = list(encrypted.flatten())

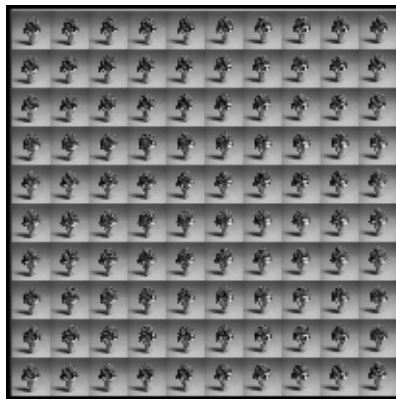
decrypted = decryption(encrypted_flat_image,key)
decrypted = np.array(decrypted,dtype=np.int64)
decrypted = decrypted.reshape(shape)
cv2_imshow(decrypted)

```

OUTPUT:



Original Image



Encrypted Image



Decrypted Image

CONCLUSION

With the increasing amount of data being generated, it is very important that confidential information does not get leaked and is read by the intended recipient. We learnt about the Columnar Transposition Cipher algorithm and we then wrote a python program to implement it.



Experiment 3

Date of Performance :

SAP Id: 60004190057

Div: A

Aim of Experiment

Design and Implement Encryption and Decryption Algorithm for S-DES

Date of Submission:

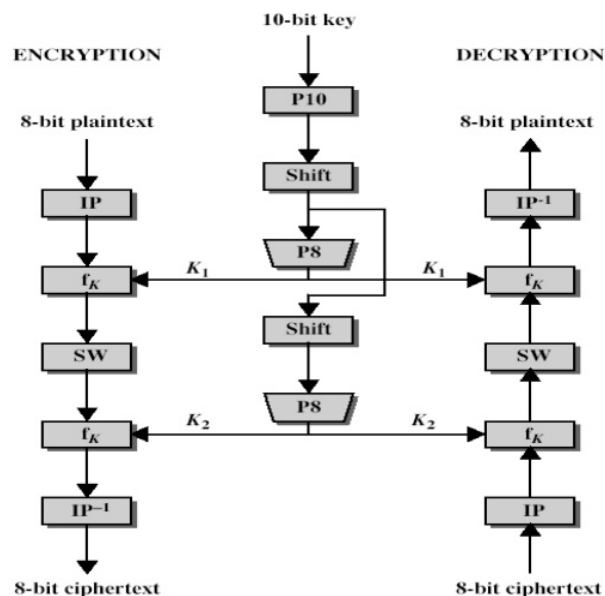
Name : Junaid Altaf Girkar

Batch : A4

Theory / Algorithm / Conceptual Description

Simplified Data Encryption Standard is a simple version of Data Encryption Standard having a 10-bit key and 8-bit plain text. It is much smaller than the DES algorithm as it takes only 8-bit plain text whereas DES takes 64-bit plain text. It is a block cipher algorithm and uses a symmetric key for its algorithm i.e. they use the same key for both encryption and decryption. It has 2 rounds for encryption which use two different keys.

The S-DES encryption algorithm takes an 8-bit block of plaintext (example: 10111101) and a 10-bit key as input and produces an 8-bit block of ciphertext as output. The S-DES decryption algorithm takes an 8-bit block of ciphertext and the same 10-bit key used to produce that ciphertext as input and produces the original 8-bit block of plaintext.



The encryption algorithm involves five functions:

An initial permutation (IP)

A complex function labeled f_k , which involves both permutation and substitution operations and depends on a key input

A simple permutation function that switches (SW) the two halves of the data the function f_k again

A permutation function that is the inverse of the initial permutation

The function f_k takes as input not only the data passing through the encryption algorithm, but also an 8-bit key. Here a 10-bit key is used from which two 8-bit subkeys are generated. The key is first subjected to a permutation (P10). Then a shift operation is performed. The output of the shift operation then passes through a permutation function that produces an 8-bit output (P8) for the first subkey (K1). The output of the shift operation also feeds into another shift and another instance of P8 to produce the second subkey (K2).

The encryption algorithm can be expressed as a composition of functions: $IP^{-1} \circ f_{K2} \circ SW \circ f_{K1} \circ IP$

Which can also be written as

```
Ciphertext = IP-1 (fK2 (SW (fK1 (IP (plaintext)))))
```

Where

$K1 = P8 (\text{Shift} (P10 (\text{Key})))$

$K2 = P8 (\text{Shift} (\text{shift} (P10 (\text{Key}))))$

Decryption can be shown as

```
Plaintext = IP-1 (fK1 (SW (fK2 (IP (ciphertext)))))
```

CODE:

```
def apply_table(inp, table):  
    res = ""  
    for i in table:  
        res += inp[i - 1]  
    return res
```

```

def left_shift(data):
    return data[1:] + data[0]

def XOR(a, b):
    res = ""
    for i in range(len(a)):
        if a[i] == b[i]:
            res += "0"
        else:
            res += "1"
    return res

def apply_sbox(s, data):
    row = int("0b" + data[0] + data[-1], 2)
    col = int("0b" + data[1:3], 2)
    return bin(s[row][col])[2:]

def function(expansion, s0, s1, key, message):
    left = message[:4]
    right = message[4:]
    temp = apply_table(right, expansion)
    temp = XOR(temp, key)
    l = apply_sbox(s0, temp[:4])
    r = apply_sbox(s1, temp[4:])
    l = "0" * (2 - len(l)) + l
    r = "0" * (2 - len(r)) + r
    temp = apply_table(l + r, p4_table)
    temp = XOR(left, temp)
    return temp + right

def key_generation_1(key, table):
    k = table_shift(key, table)
    key_merge = split_and_merge(k)
    return table_shift(key_merge, table)

def key_generation_2(key, table): return split_and_merge(key)

```

```

if __name__ == "__main__":

    key = key = str('0001101101')#input("Enter 10 bit key: ")
    message = "10101010"#input("Enter 8 bit message: ")
    print("Plain text before decryption is : " + str(message))

    p8_table = [6, 3, 7, 4, 8, 5, 10, 9]
    p10_table = [3, 5, 2, 7, 4, 10, 1, 9, 8, 6]
    p4_table = [2, 4, 3, 1]
    IP = [2, 6, 3, 1, 4, 8, 5, 7]
    IP_inv = [4, 1, 3, 5, 7, 2, 8, 6]
    expansion = [4, 1, 2, 3, 2, 3, 4, 1]
    s0 = [[1, 0, 3, 2], [3, 2, 1, 0], [0, 2, 1, 3], [3, 1, 3, 2]]
    s1 = [[0, 1, 2, 3], [2, 0, 1, 3], [3, 0, 1, 0], [2, 1, 0, 3]]

    # key generation
    temp = apply_table(key, p10_table)
    left = temp[:5]
    right = temp[5:]
    left = left_shift(left)
    right = left_shift(right)
    key1 = apply_table(left + right, p8_table)
    left = left_shift(left)
    right = left_shift(right)
    left = left_shift(left)
    right = left_shift(right)
    key2 = apply_table(left + right, p8_table)

    # encryption
    temp = apply_table(message, IP)
    temp = function(expansion, s0, s1, key1, temp)
    temp = temp[4:] + temp[:4]
    temp = function(expansion, s0, s1, key2, temp)
    CT = apply_table(temp, IP_inv)
    print("Cipher text is:", CT)

    # decryption
    temp = apply_table(CT, IP)
    temp = function(expansion, s0, s1, key2, temp)
    temp = temp[4:] + temp[:4]

```

```
temp = function(expansion, s0, s1, key1, temp)
PT = apply_table(temp, IP_inv)
print("Plain text after decrypting is:", PT)
```

OUTPUT:

```
Plain text before decryption is : 10101010
Cipher text is: 00011111
Plain text after decrypting is: 10101010
```

CONCLUSION

With the increasing amount of data being generated, it is very important that confidential information does not get leaked and is read by the intended recipient. We learnt about the Simplified DES algorithm and we then wrote a python program to implement it.

SBD Encryption Algorithm

Kanaad Deshpande

Department of Computer Engineering D.J Sanghavi College of Engineering Mumbai, India
26kanaad@gmail.com

Junaid Girkar

Department of Computer Engineering D.J Sanghavi College of Engineering Mumbai, India
junaidgirkar@gmail.com

Dr. Ramchandra Mangrulkar *Department of Computer Engineering D.J Sanghvi College of Engineering Mumbai, India*

ramchandra.mangrulkar@djsce.ac.in

Abstract—This document is a research paper on a new type of data encryption algorithm which makes use of a Sudoku, an everyday object, as its encryption key. It can encryption multiple types of data be it a file, an image or simple plaintext. The encryption process involves multiple rounds of block cipher encryption and transposition cipher encryption based on a randomly generated Sudoku resulting in the data being encrypted better than some of the other commonly used encryption tech niques.

Index Terms—encryption, cipher, block cipher, transposition cipher, image encryption, Sudoku

I. INTRODUCTION

With the rise in data being shared over the internet, data security has been gaining more importance day by day and new algorithms are being developed daily to ensure the data is being sent securely. This paper introduces a new way of encrypting multiple types of data files using an everyday objects [?] as the key.

II. PROPOSED METHODOLOGY

A. General Process

The encryption algorithm is based on a set of recurring patterns in everyday objects. One of the most popular such item is the 9 x 9 Sudoku puzzle that is available in almost all newspapers. It has a fixed pattern sequence where all the numbers from 0 - 9 must be there in each row, in each column and in each sub-blocks of 3 x 3. This data encryption algorithm is a combination of block cipher [?] and transposition ciphers [4] where the data goes through multiple rounds of encryption.

B. Image Encryption

For image encryption [5], the algorithm takes a RGB color image [6] which it converts into a series of 9 x 9 matrices. For each matrix the algorithm then generates a random Sudoku us ing the Sudokugen python library. Using the rows and columns of the randomly generated Sudoku as the transposition key, the rows and columns of the image matrix are transpositioned multiple times to ensure a higher level of encryption.

C. Text Encryption

For text encryption, similar to the image encryption, the plain text is converted into matrices of 9 x 9. Then using the Sudokugen library, a random transposition key is generated. Looping through the key, row transposition and column trans position is applied multiple times to generate the encrypted cipher text.

D. Block Diagram



Figure 1. Block Diagram.

E. Algorithm

Algorithm 1 Add padding around image

```

0: procedure SBDPPADDING( m ge, p dd ngV e) 0: if he ght < 81 then
0: add padding 81 - height
0: else if dth < 81 then
0: add padding 81 - width
0: else if he ght mod 81! = 0 then
0: add padding (height//81+1)*81 - height 0: else if dth mod 81! = 0 then
0: add padding (width//81+1)*81 - width
=0

```

Algorithm 2 Encrypt image

```

0: procedure SBDPPADDING( m ge, key)
0: Generate random Sudoku.
0: Solve the Sudoku to generate the key.
0: For each row in Sudoku:
0: Generate random Sudoku.
0: Solve the Sudoku to generate the key
1: for e chro nS dok do
2: for e chb ockof 9b ts do
2: apply permutation to each block of 9 bits for each row
3: end for
4: end for
5: for e chco mn nS dok do
6: for e ch9 - b tb ockof co mn of m ge do 6: apply permutation to each column of image for each
column of Sudoku
7: end for
8: end for
=0

```

III. EXPERIMENTATION

It has been observed that any traditional art form including but not limited to various objects, games or mathematical models that creates a pattern can be used for encipherment. Further study shows that a Sudoku based approach of encipherment, involving pixel scattering encrypted the original data beyond recognition in the image after several iterations of applying the encryption algorithm. The number of iterations can be decided based on how much scattering is required for each image. Having a flexible number of iterations makes the algorithm computationally heavy, yet secure. Experimentation with various Sudokus followed. This involved using different keys for each iteration based on the 6 quintillion options available.

IV. COMPARISON WITH OTHER ALGORITHMS A. Comparison with S-DES

The S-DES algorithm [7] doesn't work well with symmetric images. This is because the algorithm is applied uniformly to every single pixel. The output generated hence contains pixels uniformly scattered in the plane. Sudoku-based encryption

offers a flexible number of iterations so we loop through until a completely unrecognizable image is formed.



Figure 2. Original Image

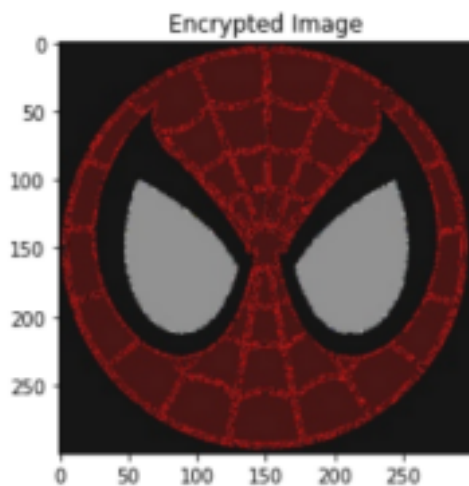


Figure 3. S-DES Encryption of Symmetric Image

$$+ b = \gamma(1)$$

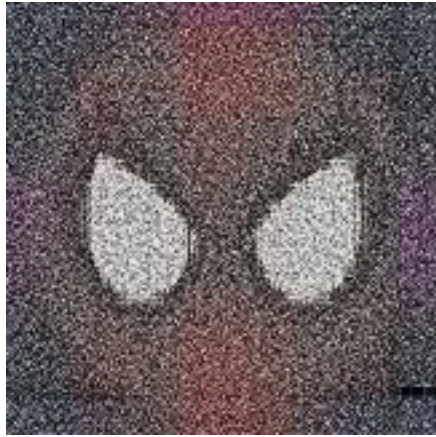


Figure 4. Sudoku-Based Encryption of Symmetric Image After 1000 Iterations

V. ATTACKING THE ALGORITHM

A. Brute Force Attack

Considering a brute force attack on the algorithm, the attacker first has to find the randomly generated 9 x 9 Sudoku key. This has a possibility of 1 in 6,670,903,752,021,072,936,960 cases. Even with a powerful device, it would take an extremely long time to try all the possible combinations in order to find the correct one as they will have to run the algorithm through a high number of randomly generated layers of decryption whose count will need to be separately brute forced.

Even if we consider the one in sextillion chance that an attacker manages to guess the randomly generated key using the brute force algorithm, the attacker would gain access to the first 324 bits of data only. For the remaining data, the attacker has to again brute force for the other randomly generated 9 x 9 key.

So brute forcing this algorithm has an infinitesimally chance of success and the return per success is also very less.

B. Results

The algorithm was tested on several iterations - 10, 50, 100, 250, 400, 500, 750, 1000 and the following results were obtained:

Table I
COMPARISON OF ITERATIONS

Number of iterations	Time Required
10	2.658229500000001
50	11.760182800000003
100	25.154778399999998
250	62.587474000000014
400	105.29135510000003
500	131.0632736
750	217.96522460000006
1000	287.7479881999998

It can be observed that the algorithm takes nearly 5 minutes to complete a thousand iterations, which shows its

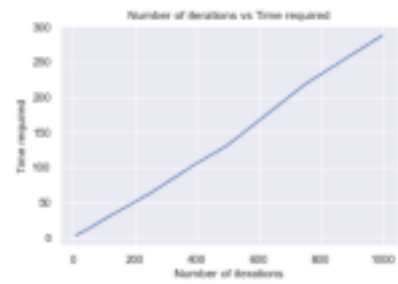


Figure 5. Time Analysis

computational inefficiency, which is a trade-off when it comes to security.

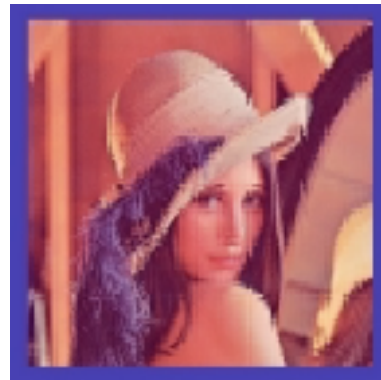


Figure 6. Image with padding after first iteration



Figure 7. Image after 10 iterations



Figure 8. Image after 100 iterations



Figure 11. Image after 500 iterations



Figure 9. Image after 250 iterations



Figure 12. Image after 750 iterations



Figure 10. Image after 400 iterations



Figure 13. Image after 1000 iterations

ACKNOWLEDGMENT

We would like to express our gratitude to Department of Computer Engineering at Dwarkadas J Sanghvi College of Engineering who motivated us to dive into research and guided us when we faced any difficulty.

REFERENCES

- [1] Mousavi, Maryam, and Babak Sadeghiyan. "A new image encryption scheme with Feistel like structure using chaotic S-box and Rubik cube based P-box." *Multimedia Tools and Applications* 80.9 (2021): 13157-13177.
- [2] Wang, Xu, et al. "Multi-level reversible data hiding for crypto imagery via a block-wise substitution-transposition cipher." *Journal of Information Security and Applications* 64 (2022): 103067.
- [3] Jose A.P. Artiles, Daniel P.B. Chaves, and Cecilio Pimentel. "2019. Image encryption using block cipher and chaotic sequences. *Image Commun.* 79, C (Nov 2019), 24–31. <https://doi.org/10.1016/j.image.2019.08.014>
- [4] Djamalilleil, As'ad Muslim, Muslim Salim, Yulita Alwi, Erick Azis, Huzain Herman,.

(2018). Modified Transposition Cipher Algorithm for Images Encryption. 1-4. 10.1109/EIConCIT.2018.8878326.

- [5] Zia, U., McCartney, M., Scotney, B. et al. Survey on image encryption techniques using chaotic maps in spatial, transform and spatiotemporal domains. *Int. J. Inf. Secur.* (2022). <https://doi.org/10.1007/s10207-022-00588-5>
- [6] Al-Roithy, Budoor Obid, and Adnan Gutub. "Remodeling randomness prioritization to boost-up security of RGB image encryption." *Multimedia Tools and Applications* 80.18 (2021): 28521-28581.
- [7] Kumar, Sanjay, and Sandeep Srivastava. "Image encryption using simplified data encryption standard (S-DES)." *International Journal of Computer Applications* 104.2 (2014).

Experiment 5

Date of Performance :

SAP Id: 60004190057

Div: A

Date of Submission:

Name : Junaid Altaf Girkar

Batch : A4

Aim of Experiment

Design and Implement Encryption and Decryption Algorithm for RSA Algorithm

Theory / Algorithm / Conceptual Description

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.

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 multiplication of two large prime numbers. And private keys are 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.

PUBLIC KEY GENERATION:

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 n .
 $1 < e < \Phi(n)$ [$\Phi(n)$ is discussed below],

Let us now consider it to be equal to 3.
Our Public Key is made of n and e

PRIVATE KEY GENERATION

We need to calculate $\Phi(n)$:
Such that $\Phi(n) = (P-1)(Q-1)$
so, $\Phi(n) = 3016$

Now calculate Private Key, d :
 $d = (k \cdot \Phi(n) + 1) / e$ for some integer k
For k = 2, value of d is 2011.

ENCRYPTION:

Convert letters to numbers : H = 8 and I = 9

Thus Encrypted Data $c = 89e \bmod n$.
Thus our Encrypted Data comes out to be 1394

DECRYPTION:

Now we will decrypt 1394 :
Decrypted Data = $cd \bmod n$.
Thus our Encrypted Data comes out to be 89
8 = H and I = 9 i.e. "HI".

CODE:

```
import numpy as np
import random
import cv2
import matplotlib.pyplot as plt

# Select 2 prime numbers
p = 19
q = 13

# First public key
n = p * q

def gcd(a, b):
    if (a == 0):
        return b
    return gcd(b % a, a)

def phi(n):
    result = 1
```



```

    for i in range(2, n):
        if (gcd(i, n) == 1):
            result+=1
    return result

# Calculating Phi(n)
phi_n = phi(n)
# e = random.randint(0, phi_n)
e = 5

public_key = (n, e)
# k = random.randint(1, 10)
k = 4

d = (k * phi_n + 1) / (e)
private_key = d

image = cv2.imread('RSA.jpg', 0)
image.shape

plt.imshow(image)

# display that image
plt.show()
def encrypt(input, e = 5, n = 247):
    cipher = pow(input, e) % n
    return cipher

shape = image.shape
pixels = image.flatten()
enc = []

for pixel in pixels:
    enc.append(encrypt(int(pixel)))

enc = np.array(enc)

encrypted_image = enc.reshape(shape)

cv2.imwrite('rsa_encryption.jpg', encrypted_image)
def decryption(encrypted, d=173, n=247):
    return pow(encrypted, int(d)) % n

#DECRYPTION:
image2 = cv2.imread('rsa_encryption.jpg', 0)
pixels_enc = encrypted_image.flatten()

```

```

image2
og = []

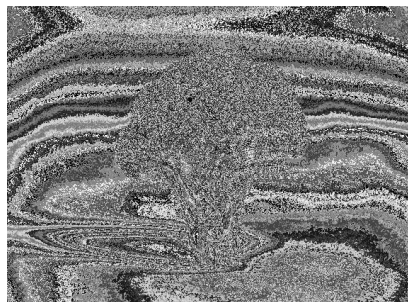
for pixel in pixels_enc:
    og.append(decryption(int(pixel)))

og = np.array(og)

original_image = og.reshape(shape)
original_image
cv2.imwrite('decrypted.jpg', original_image)
plt.imshow(original_image, cmap='gray')
# display that image
plt.show()

```

OUTPUT:



ORIGINAL

ENCRYPTED

DECRYPTED

CONCLUSION

With the increasing amount of data being generated, it is very important that confidential information does not get leaked and is read by the intended recipient. We learnt about asymmetric key ciphers and the RSA algorithm and we then wrote a python program to implement it.

Experiment 6

Date of Performance :

SAP Id: 60004190057

Div: A

Date of Submission:

Name : Junaid Altaf Girkar

Batch : A4

Aim of Experiment

Design and Implement Diffie Hellman Key Exchange Algorithm

Theory:

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

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

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

Implementation 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: $x = (9^4 \bmod 23) = (6561 \bmod 23) = 6$
Bob: $y = (9^3 \bmod 23) = (729 \bmod 23) = 16$

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: $k_a = y^a \bmod p = 65536 \bmod 23 = 9$
Bob: $k_b = x^b \bmod p = 216 \bmod 23 = 9$

Step 7: 9 is the shared secret.

CODE:

```
from random import randint

# Both the persons will be agreed upon the
# public keys Q and P
# A prime number P is taken
P = 23

# A primitive root for P, Q is taken
Q = 9

print('The Value of P is :%d'%(P))
print('The Value of Q is :%d'%(Q))

# Alice will choose the private key a
a = 4
print('The Private Key a for Alice is :%d'%(a))

# gets the generated key
x = int(pow(Q,a,P))

# Bob will choose the private key b
b = 3
print('The Private Key b for Bob is :%d'%(b))

# gets the generated key
y = int(pow(Q,b,P))

# Secret key for Alice
ka = int(pow(y,a,P))

# Secret key for Bob
kb = int(pow(x,b,P))

print('Secret key for the Alice is : %d'%(ka))
print('Secret Key for the Bob is : %d'%(kb))
```

OUTPUT:

```
The Value of P is :23
The Value of Q is :9
The Private Key a for Alice is :4
The Private Key b for Bob is :3
Secret key for the Alice is : 9
Secret Key for the Bob is : 9
The Value of P is :23
The Value of Q is :9
The Private Key a for Alice is :4
The Private Key b for Bob is :3
Secret key for the Alice is : 9
Secret Key for the Bob is : 9
```

CONCLUSION

Even while using ciphers for encryption, it is crucial that the key for encryption and decryption is secure and yet available to the sender and receiver. Diffie Hellman key exchange algorithm is an algorithm in which even the sender and receiver are unaware of others private key. We learnt about this algorithm and implemented it in Python

Experiment 7

Date of Performance :

SAP Id: 60004190057

Div: A

Date of Submission:

Name : Junaid Altaf Girkar

Batch : A4

Aim of Experiment

Study the use of network reconnaissance tools like WHOIS, dig, traceroute, nslookup to gather information about networks and domain registrars.

Theory:

WHOIS:

WHOIS is a TCP-based query and response protocol that is commonly used to provide information services to Internet users. It returns information about the registered Domain Names, an IP address block, Name Servers and a much wider range of information services.

```
Whois v1.21 - Domain information lookup
Copyright (C) 2005-2019 Mark Russinovich
Sysinternals - www.sysinternals.com
```

```
Connecting to COM.whois-servers.net...
```

```
WHOIS Server: whois.markmonitor.com
Registrar URL: http://www.markmonitor.com
Updated Date: 2019-09-09T15:39:04Z
Creation Date: 1997-09-15T04:00:00Z
Registry Expiry Date: 2028-09-14T04:00:00Z
Registrar: MarkMonitor Inc.
Registrar IANA ID: 292
Registrar Abuse Contact Email: abusecomplaints@markmonitor.com
Registrar Abuse Contact Phone: +1.2086851750
Domain Status: clientDeleteProhibited
https://icann.org/epp#clientDeleteProhibited
Domain Status: clientTransferProhibited
https://icann.org/epp#clientTransferProhibited
Domain Status: clientUpdateProhibited
https://icann.org/epp#clientUpdateProhibited
Domain Status: serverDeleteProhibited
https://icann.org/epp#serverDeleteProhibited
Domain Status: serverTransferProhibited
https://icann.org/epp#serverTransferProhibited
Domain Status: serverUpdateProhibited
```

<https://icann.org/epp#serverUpdateProhibited>

Name Server: NS1.GOOGLE.COM

Name Server: NS2.GOOGLE.COM

Name Server: NS3.GOOGLE.COM

Name Server: NS4.GOOGLE.COM

DNSSEC: unsigned

URL of the ICANN Whois Inaccuracy Complaint Form:

<https://www.icann.org/wicf/>

>>> Last update of whois database: 2022-06-02T15:50:49Z <<<

For more information on Whois status codes, please visit
<https://icann.org/epp>

NOTICE: The expiration date displayed in this record is the date the registrar's sponsorship of the domain name registration in the registry is currently set to expire. This date does not necessarily reflect the expiration date of the domain name registrant's agreement with the sponsoring registrar. Users may consult the sponsoring registrar's Whois database to view the registrar's reported date of expiration for this registration.

TERMS OF USE: You are not authorized to access or query our Whois database through the use of electronic processes that are high-volume and automated except as reasonably necessary to register domain names or modify existing registrations; the Data in VeriSign Global Registry Services' ("VeriSign") Whois database is provided by VeriSign for information purposes only, and to assist persons in obtaining information about or related to a domain name registration record. VeriSign does not guarantee its accuracy. By submitting a Whois query, you agree to abide by the following terms of use: You agree that you may 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 of
mass
unsolicited, commercial advertising or solicitations via e-mail,
telephone,
or facsimile; or (2) enable high volume, automated, electronic
processes
that apply to VeriSign (or its computer systems). The compilation,
repackaging, dissemination or other use of this Data is expressly
prohibited without the prior written consent of VeriSign. You
agree not to
use electronic processes that are automated and high-volume to
access or
query the Whois database except as reasonably necessary to
register
domain names or modify existing registrations. VeriSign reserves
the right
to restrict your access to the Whois database in its sole
discretion to ensure
operational stability. VeriSign may restrict or terminate your
access to the
Whois database for failure to abide by these terms of use.
VeriSign
reserves the right to modify these terms at any time.

The Registry database contains ONLY .COM, .NET, .EDU domains and
Registrars.

Connecting to whois.markmonitor.com...

WHOIS Server: whois.markmonitor.com
Registrar URL: <http://www.markmonitor.com>
Updated Date: 2019-09-09T15:39:04+0000
Creation Date: 1997-09-15T07:00:00+0000
Registrar Registration Expiration Date: 2028-09-13T07:00:00+0000
Registrar: MarkMonitor, Inc.
Registrar IANA ID: 292
Registrar Abuse Contact Email: abusecomplaints@markmonitor.com
Registrar Abuse Contact Phone: +1.2083895770
Domain Status: clientUpdateProhibited
(<https://www.icann.org/epp#clientUpdateProhibited>)
Domain Status: clientTransferProhibited

(<https://www.icann.org/epp#clientTransferProhibited>)
Domain Status: clientDeleteProhibited
(<https://www.icann.org/epp#clientDeleteProhibited>)
Domain Status: serverUpdateProhibited
(<https://www.icann.org/epp#serverUpdateProhibited>)
Domain Status: serverTransferProhibited
(<https://www.icann.org/epp#serverTransferProhibited>)
Domain Status: serverDeleteProhibited
(<https://www.icann.org/epp#serverDeleteProhibited>)
Registrant Organization: Google LLC
Registrant State/Province: CA
Registrant Country: US
Registrant Email: Select Request Email Form at
<https://domains.markmonitor.com/whois/google.com>
Admin Organization: Google LLC
Admin State/Province: CA
Admin Country: US
Admin Email: Select Request Email Form at
<https://domains.markmonitor.com/whois/google.com>
Tech Organization: Google LLC
Tech State/Province: CA
Tech Country: US
Tech Email: Select Request Email Form at
<https://domains.markmonitor.com/whois/google.com>
Name Server: ns1.google.com
Name Server: ns4.google.com
Name Server: ns3.google.com
Name Server: ns2.google.com
DNSSEC: unsigned
URL of the ICANN WHOIS Data Problem Reporting System:
<http://wdprs.internic.net/>
>>> Last update of WHOIS database: 2022-06-02T15:43:07+0000 <<<

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>

Contact us at +1.8007459229

In Europe, at +44.02032062220

--

Domain Name: google.com

Registry Domain ID: 2138514_DOMAIN_COM-VRSN

Registrar WHOIS Server: whois.markmonitor.com

Registrar URL: <http://www.markmonitor.com>

Updated Date: 2019-09-09T15:39:04+0000

Creation Date: 1997-09-15T07:00:00+0000

Registrar Registration Expiration Date: 2028-09-13T07:00:00+0000

Registrar: MarkMonitor, Inc.

Registrar IANA ID: 292

Registrar Abuse Contact Email: abusecomplaints@markmonitor.com

Registrar Abuse Contact Phone: +1.2083895770

Domain Status: clientUpdateProhibited

(<https://www.icann.org/epp#clientUpdateProhibited>)

Domain Status: clientTransferProhibited

(<https://www.icann.org/epp#clientTransferProhibited>)

Domain Status: clientDeleteProhibited

(<https://www.icann.org/epp#clientDeleteProhibited>)

Domain Status: serverUpdateProhibited

(<https://www.icann.org/epp#serverUpdateProhibited>)

Domain Status: serverTransferProhibited

(<https://www.icann.org/epp#serverTransferProhibited>)

Domain Status: serverDeleteProhibited

(<https://www.icann.org/epp#serverDeleteProhibited>)

Registrant Organization: Google LLC

Registrant State/Province: CA

Registrant Country: US

Registrant Email: Select Request Email Form at

<https://domains.markmonitor.com/whois/google.com>

Admin Organization: Google LLC

Admin State/Province: CA

Admin Country: US

Admin Email: Select Request Email Form at

<https://domains.markmonitor.com/whois/google.com>
Tech Organization: Google LLC
Tech State/Province: CA
Tech Country: US
Tech Email: Select Request Email Form at
<https://domains.markmonitor.com/whois/google.com>
Name Server: ns1.google.com
Name Server: ns4.google.com
Name Server: ns3.google.com
Name Server: ns2.google.com
DNSSEC: unsigned
URL of the ICANN WHOIS Data Problem Reporting System:
<http://wdprs.internic.net/>
>>> Last update of WHOIS database: 2022-06-02T15:43:07+0000 <<<

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>

Contact us at +1.8007459229

In Europe, at +44.02032062220

--

DIG:

Dig (Domain Information Groper) is a powerful command-line tool for querying DNS
name servers. The dig command, allows you to query information about various DNS
records, including host addresses, mail exchanges, and name servers. It is the most
commonly used tool among system administrators for troubleshooting DNS
problems
because of its flexibility and ease of use

```
C:\Users\DELL>dig google.com
```

```
; <>> DiG 9.16.29 <>> google.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 62757
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL:
9

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;google.com.                IN      A

;; ANSWER SECTION:
google.com.                 157     IN      A      142.251.42.14

;; AUTHORITY SECTION:
google.com.                 35249   IN      NS      ns4.google.com.
google.com.                 35249   IN      NS      ns2.google.com.
google.com.                 35249   IN      NS      ns1.google.com.
google.com.                 35249   IN      NS      ns3.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.             206783  IN      A      216.239.32.10
ns1.google.com.             293783  IN      AAAA
2001:4860:4802:32::a
ns4.google.com.             31667   IN      A      216.239.38.10
ns4.google.com.             31667   IN      AAAA
2001:4860:4802:38::a
ns3.google.com.             31667   IN      A      216.239.36.10
ns3.google.com.             31667   IN      AAAA
2001:4860:4802:36::a
ns2.google.com.             204464  IN      A      216.239.34.10
ns2.google.com.             204464  IN      AAAA
2001:4860:4802:34::a

;; Query time: 6 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Thu Jun 02 21:30:08 India Standard Time 2022
;; MSG SIZE rcvd: 303
```

```
C:\Users\DELL>dig 142.251.42.14

; <<>> DiG 9.16.29 <<>> 142.251.42.14
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 38521
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL:
1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;142.251.42.14.                IN      A

;; AUTHORITY SECTION:
.                10800   IN      SOA
a.root-servers.net. nstld.verisign-grs.com. 2022060200 1800 900
604800 86400

;; Query time: 0 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Thu Jun 02 21:35:01 India Standard Time 2022
;; MSG SIZE rcvd: 117
```

NSLOOKUP:

Nslookup (stands for "Name Server Lookup") is a useful command for getting information from 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 followed by the domain name will display the "A Record" (IP Address) of the domain. Use this command to find the address record for a domain. It queries to domain name servers and get the details.

```
C:\Users\junai>nslookup amazon.com
Server: UnKnown
Address: 192.168.0.1
```

```
Non-authoritative answer:
Name:      amazon.com
Addresses:  176.32.103.205
             54.239.28.85
             205.251.242.103
```

SOA record (start of authority), provides the authoritative information about the domain, the e-mail address of the domain admin, the domain serial number, etc

```
C:\Users\junai>nslookup -type=soa amazon.com
Server:  UnKnown
Address:  192.168.0.1
```

```
Non-authoritative answer:
amazon.com
```

```
    primary name server = dns-external-master.amazon.com
    responsible mail addr = root.amazon.com
    serial      = 2010158906
    refresh     = 180 (3 mins)
    retry       = 60 (1 min)
    expire      = 3024000 (35 days)
    default TTL = 60 (1 min)
```

```
amazon.com      nameserver = pdns6.ultradns.co.uk
amazon.com      nameserver = pdns1.ultradns.net
amazon.com      nameserver = ns1.p31.dynect.net
amazon.com      nameserver = ns2.p31.dynect.net
amazon.com      nameserver = ns3.p31.dynect.net
amazon.com      nameserver = ns4.p31.dynect.net
ns1.p31.dynect.net  internet address = 108.59.161.31
ns2.p31.dynect.net  internet address = 108.59.162.31
ns3.p31.dynect.net  internet address = 108.59.163.31
ns4.p31.dynect.net  internet address = 108.59.164.31
pdns1.ultradns.net  internet address = 204.74.108.1
pdns6.ultradns.co.uk internet address = 204.74.115.1
ns1.p31.dynect.net  AAAA IPv6 address = 2600:2000:2210::31
ns2.p31.dynect.net  AAAA IPv6 address = 2600:2000:2220::31
ns3.p31.dynect.net  AAAA IPv6 address = 2600:2000:2230::31
```



```
ns4.p31.dynect.net      AAAA IPv6 address = 2600:2000:2240::31
pdns1.ultradns.net      AAAA IPv6 address = 2001:502:f3ff::1
pdns6.ultradns.co.uk    AAAA IPv6 address = 2610:a1:1017::1
```

NS (Name Server) record maps a domain name to a list of DNS servers authoritative for that domain. It will output the name serves which are associated with the given domain.

```
C:\Users\junai>nslookup -type=ns amazon.com
```

```
Server:  UnKnown
```

```
Address: 192.168.0.1
```

```
Non-authoritative answer:
```

```
amazon.com      nameserver = pdns1.ultradns.net
amazon.com      nameserver = ns4.p31.dynect.net
amazon.com      nameserver = pdns6.ultradns.co.uk
amazon.com      nameserver = ns1.p31.dynect.net
amazon.com      nameserver = ns2.p31.dynect.net
amazon.com      nameserver = ns3.p31.dynect.net
```

```
ns1.p31.dynect.net      internet address = 108.59.161.31
ns2.p31.dynect.net      internet address = 108.59.162.31
ns3.p31.dynect.net      internet address = 108.59.163.31
ns4.p31.dynect.net      internet address = 108.59.164.31
pdns1.ultradns.net      internet address = 204.74.108.1
pdns6.ultradns.co.uk    internet address = 204.74.115.1
ns1.p31.dynect.net      AAAA IPv6 address = 2600:2000:2210::31
ns2.p31.dynect.net      AAAA IPv6 address = 2600:2000:2220::31
ns3.p31.dynect.net      AAAA IPv6 address = 2600:2000:2230::31
ns4.p31.dynect.net      AAAA IPv6 address = 2600:2000:2240::31
pdns1.ultradns.net      AAAA IPv6 address = 2001:502:f3ff::1
pdns6.ultradns.co.uk    AAAA IPv6 address = 2610:a1:1017::1
```

TRACEROUTE:

Traceroute command in Linux prints the route that a packet takes to reach the host. This command is useful when you want to know about the route and about all the hops that a packet takes. The first column corresponds to the hop count. The second

column represents the address of that hop and after that, you see three space-separated time in milliseconds. the traceroute command sends three packets to the hop and each of the time refers to the time taken by the packet to reach the hop. In windows, alternative for traceroute command is tracert.

```
C:\Users\junai>tracert amazon.com
```

```
Tracing route to amazon.com [205.251.242.103]
over a maximum of 30 hops:
```

```
  1      3 ms      3 ms      2 ms  192.168.0.1
  2      2 ms      1 ms      3 ms  42-200.59.103.n4uspl.net
[103.59.200.42]
  3      3 ms      2 ms      3 ms  41-200.59.103.n4uspl.net
[103.59.200.41]
  4      *          *          3 ms  254-200.59.103.n4uspl.net
[103.59.200.254]
  5      6 ms      5 ms      6 ms  123.252.244.1
  6      7 ms      4 ms      4 ms  10.0.10.209
  7      6 ms      4 ms      5 ms  10.124.253.101
  8      *          *          8 ms  10.118.143.9
  9      7 ms      5 ms      5 ms
115.113.165.21.static-mumbai.vsnl.net.in [115.113.165.21]
 10      9 ms      7 ms      5 ms  172.23.78.237
 11      5 ms      4 ms      5 ms
ix-ae-0-100.tcore1.mlv-mumbai.as6453.net [180.87.38.5]
 12    251 ms    201 ms    200 ms
if-ae-2-2.tcore2.mlv-mumbai.as6453.net [180.87.38.2]
 13      *          *          *    Request timed out.
 14      *      401 ms      *
if-ae-66-8.tcore3.nto-newyork.as6453.net [80.231.130.195]
 15    207 ms      *      202 ms
if-be-2-2.ecore1.n75-newyork.as6453.net [66.110.96.62]
 16    211 ms    217 ms    223 ms
if-ae-57-2.tcore1.n75-newyork.as6453.net [66.110.96.58]
 17    200 ms    200 ms    347 ms  66.110.96.157
 18      *          *          *    Request timed out.
 19      *          *          *    Request timed out.
 20      *          *          *    Request timed out.
 21      *          *          *    Request timed out.
 22      *          *          *    Request timed out.
 23      *          *          *    Request timed out.
```

```
24      *      *      *      Request timed out.  
25      *      *      *      Request timed out.  
26      *      *      *      Request timed out.  
27      *      *      *      Request timed out.  
28      *      *      *      Request timed out.  
29      *      *      *      Request timed out.  
30      *      *      *      Request timed out.
```

```
Trace complete.
```

CONCLUSION

Thus, we have successfully implemented and studied the use of network reconnaissance tools like WHOIS, dig, traceroute, nslookup to gather information about networks and domain registrars.

Experiment 8

Date of Performance :

SAP Id: 60004190057

Div: A

Date of Submission:

Name : Junaid Altaf Girkar

Batch : A4

Aim of Experiment

Study of packet sniffer tools : wireshark, :

- a. Download and install wireshark and capture icmp, tcp, and http packets in promiscuous mode.
- b. Explore how the packets can be traced based on different filters. (CO5)

Theory:

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\junai>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 from Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

icmp

No.	icmp	icmpv6	Source	Destination	Protocol	Length	Info
1149	17.810458	172.67.7.42	192.168.0.175	TCP	1494	443 → 53989 [ACK] Seq=877485 Ack=618 Win=246 Len=1440 [TCP segment of a reassembled PDU]	
1150	17.810458	172.67.7.42	192.168.0.175	TCP	1494	443 → 53989 [ACK] Seq=878925 Ack=618 Win=246 Len=1440 [TCP segment of a reassembled PDU]	
1151	17.810458	172.67.7.42	192.168.0.175	TLSv1.2	506	Application Data, Application Data	
1152	17.810659	192.168.0.175	172.67.7.42	TCP	54	53989 → 443 [ACK] Seq=618 Ack=882257 Win=2056 Len=0	
1153	17.828026	192.168.0.175	172.67.7.42	TLSv1.2	310	Application Data	
1154	17.832340	172.67.7.42	192.168.0.175	TCP	54	443 → 53989 [ACK] Seq=882257 Ack=874 Win=248 Len=0	
1155	17.833106	192.168.0.171	224.0.0.251	IGMPv2	46	Membership Report group 224.0.0.251	
1156	17.947858	192.168.0.171	224.0.0.251	MDNS	171	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QU" question PTR _companion-link._tcp.local, "QU" question PTR _homekit._tcp.lo...	
1157	17.948987	fe80::842:3734:857a::f	ff02::fb	MDNS	191	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QU" question PTR _companion-link._tcp.local, "QU" question PTR _homekit._tcp.lo...	
1158	18.148313	192.168.0.175	3.108.46.16	TLSv1.2	108	Application Data	
1159	18.157597	3.108.46.16	192.168.0.175	TCP	54	443 → 56718 [ACK] Seq=1241 Ack=165 Win=10 Len=0	
1160	18.157597	3.108.46.16	192.168.0.175	TLSv1.2	110	Application Data	
1161	18.209250	192.168.0.175	3.108.46.16	TCP	54	56718 → 443 [ACK] Seq=165 Ack=1297 Win=512 Len=0	
1162	18.301024	192.168.0.171	224.0.0.251	MDNS	215	Standard query 0x0000 ANY Rahat Altaf Girkar._rdlink._tcp.local, "QU" question ANY Rahat-Altaf-Girkar.local, "QU" question SRV 0 _	
1163	18.302058	fe80::842:3734:857a::f	ff02::fb	MDNS	235	Standard query 0x0000 ANY Rahat Altaf Girkar._rdlink._tcp.local, "QU" question ANY Rahat-Altaf-Girkar.local, "QU" question SRV 0 _	
1164	18.557830	192.168.0.171	224.0.0.251	MDNS	215	Standard query 0x0000 ANY Rahat Altaf Girkar._rdlink._tcp.local, "QM" question ANY Rahat-Altaf-Girkar.local, "QM" question SRV 0 _	
1165	18.557830	fe80::842:3734:857a::f	ff02::fb	MDNS	235	Standard query 0x0000 ANY Rahat Altaf Girkar._rdlink._tcp.local, "QM" question ANY Rahat-Altaf-Girkar.local, "QM" question SRV 0 _	
1166	18.810123	192.168.0.171	224.0.0.251	MDNS	215	Standard query 0x0000 ANY Rahat Altaf Girkar._rdlink._tcp.local, "QM" question ANY Rahat-Altaf-Girkar.local, "QM" question SRV 0 _	

> Frame 1: 1494 bytes on wire (11952 bits), 1494 bytes captured (11952 bits) on interface \Device\NPF_{90485CC5-6194-4E36-A1C7-32FEBE1728C0}, id 0

> Ethernet II, Src: Tp-LinkT_89:e7:a8 (d8:07:b6:89:e7:a8), Dst: IntelCor_d6:31:6b (40:74:e0:d6:31:6b)

> Internet Protocol Version 4, Src: 172.67.7.42, Dst: 192.168.0.175

> Transmission Control Protocol, Src Port: 443, Dst Port: 53989, Seq: 1, Ack: 1, Len: 1440

0000 40 74 e0 d6 31 6b d8 07 b6 89 e7 a8 08 00 45 00 @t-1k-...E-

0010 05 c8 e9 7f 40 00 35 06 e1 eb ac 43 07 2a c0 a8 ...@S...C*..

0020 00 af 01 bb d2 e5 9c 19 9e 18 09 8c 44 7c 50 10D|P.....

0030 00 f5 5d 0f 00 00 17 03 03 20 1a 21 cc 08 9e f2 ..].....I....

0040 9c e0 a6 0c 81 c1 2e 79 6c 64 86 7c 86 aa 1d b5y Id|....

0050 2f 9b 66 ec 3c 18 00 60 91 04 28 0e d4 04 7b 5c /-f<...(\

0060 2b 32 91 8b d6 82 87 aa 62 64 08 5c af a3 fc 1f +2.....bd\...

0070 1d 52 40 4a 28 67 38 4f b4 0f 99 46 87 45 3c 5d -R@!(g80 F-E<]

0080 94 78 2e 95 10 74 e2 ad 6b a0 ce c0 92 24 f3 32 -x...t...k...\$-2

0090 f4 78 b1 d0 d3 ea 26 aa 2d d6 82 47 9b a0 a2 84 -x...&...G....

00a0 b8 fe 2c ff e3 23 00 d6 51 59 be 34 64 ed 09 f3 ,...#...QY-4d...

00b0 da 6d 1d 8d 13 3e 83 2e 58 9c 4c 23 21 cb 33 28 m...>...X L!#1-3(

00c0 85 ee 6f b5 c0 63 04 d8 12 1c 3d da 54 e2 29 -o-h-c...-T-)

00d0 8a d8 43 ff a0 67 0a 58 63 72 cc 79 8e 12 0a 13 -C-g-X cr-y....

Packets: 1166 · Displayed: 1166 (100.0%) Profile: Default

Capturing TCP Packets:

Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp

No.	Time	Source	Destination	Protocol	Length	Info
23818	77.624260	13.107.136.9	192.168.0.175	TCP	1494	443 → 1154 [ACK] Seq=4521840 Ack=893 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]
23819	77.624271	192.168.0.175	13.107.136.9	TCP	54	1154 → 443 [ACK] Seq=893 Ack=4523280 Win=132352 Len=0
23820	77.624480	13.107.136.9	192.168.0.175	TCP	1494	443 → 1154 [ACK] Seq=4523280 Ack=893 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]
23821	77.624480	13.107.136.9	192.168.0.175	TCP	1494	443 → 1154 [ACK] Seq=4524720 Ack=893 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]
23822	77.624480	13.107.136.9	192.168.0.175	TCP	1494	443 → 1154 [ACK] Seq=4526160 Ack=893 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]
23823	77.624488	192.168.0.175	13.107.136.9	TCP	54	1154 → 443 [ACK] Seq=893 Ack=4527600 Win=132352 Len=0
23824	77.624677	13.107.136.9	192.168.0.175	TLSv1.2	1494	Application Data [TCP segment of a reassembled PDU]
23825	77.624677	13.107.136.9	192.168.0.175	TCP	1494	443 → 1154 [ACK] Seq=4529840 Ack=893 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]
23826	77.624685	192.168.0.175	13.107.136.9	TCP	54	1154 → 443 [ACK] Seq=893 Ack=4530480 Win=132352 Len=0
23827	77.625246	13.107.136.9	192.168.0.175	TCP	1404	443 → 1154 [ACK] Seq=4530480 Ack=893 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]
23828	77.625246	13.107.136.9	192.168.0.175	TCP	1494	443 → 1154 [ACK] Seq=4531920 Ack=893 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]
23829	77.625246	13.107.136.9	192.168.0.175	TCP	1494	443 → 1154 [ACK] Seq=4533360 Ack=893 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]
23830	77.625246	13.107.136.9	192.168.0.175	TCP	1494	443 → 1154 [ACK] Seq=4534800 Ack=893 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]
23831	77.625246	13.107.136.9	192.168.0.175	TCP	1494	443 → 1154 [ACK] Seq=4536240 Ack=893 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]
23832	77.625256	192.168.0.175	13.107.136.9	TCP	54	1154 → 443 [ACK] Seq=893 Ack=4537680 Win=132352 Len=0
23833	77.625435	13.107.136.9	192.168.0.175	TCP	1494	443 → 1154 [ACK] Seq=4537680 Ack=893 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]
23834	77.625435	13.107.136.9	192.168.0.175	TCP	1494	443 → 1154 [ACK] Seq=4539120 Ack=893 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]
23835	77.625443	192.168.0.175	13.107.136.9	TCP	54	1154 → 443 [ACK] Seq=893 Ack=4540560 Win=132352 Len=0
23836	77.626279	13.107.136.9	192.168.0.175	TCP	1494	443 → 1154 [ACK] Seq=4540560 Ack=893 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]

> Frame 1: 1494 bytes on wire (11952 bits), 1494 bytes captured (11952 bits) on interface \Device\NPF_{90485CC5-6194-4E36-A1C7-32FEBE1728C0}, id 0

> Ethernet II, Src: Tp-LinkT_89:e7:a8 (d8:07:b6:89:e7:a8), Dst: IntelCor_d6:31:6b (40:74:e0:d6:31:6b)

> Internet Protocol Version 4, Src: 172.67.7.42, Dst: 192.168.0.175

> Transmission Control Protocol, Src Port: 443, Dst Port: 53989, Seq: 1, Ack: 1, Len: 1440

0000 40 74 e0 d6 31 6b d8 07 b6 89 e7 a8 08 00 45 00 @t-1k-...E-

0010 05 c8 e9 7f 40 00 35 06 e1 eb ac 43 07 2a c0 a8 ...@S...C*..

0020 00 af 01 bb d2 e5 9c 19 9e 18 09 8c 44 7c 50 10D|P.....

0030 00 f5 5d 0f 00 00 17 03 03 20 1a 21 cc 08 9e f2 ..].....I....

0040 9c e0 a6 0c 81 c1 2e 79 6c 64 86 7c 86 aa 1d b5y Id|....

0050 2f 9b 66 ec 3c 18 00 60 91 04 28 0e d4 04 7b 5c /-f<...(\

0060 2b 32 91 8b d6 82 87 aa 62 64 08 5c af a3 fc 1f +2.....bd\...

0070 1d 52 40 4a 28 67 38 4f b4 0f 99 46 87 45 3c 5d -R@!(g80 F-E<]

0080 94 78 2e 95 10 74 e2 ad 6b a0 ce c0 92 24 f3 32 -x...t...k...\$-2

0090 f4 78 b1 d0 d3 ea 26 aa 2d d6 82 47 9b a0 a2 84 -x...&...G....

00a0 b8 fe 2c ff e3 23 00 d6 51 59 be 34 64 ed 09 f3 ,...#...QY-4d...

00b0 da 6d 1d 8d 13 3e 83 2e 58 9c 4c 23 21 cb 33 28 m...>...X L!#1-3(

00c0 85 ee 6f b5 c0 63 04 d8 12 1c 3d da 54 e2 29 -o-h-c...-T-)

00d0 8a d8 43 ff a0 67 0a 58 63 72 cc 79 8e 12 0a 13 -C-g-X cr-y....

Packets: 23836 · Displayed: 21894 (88.5%) Profile: Default

Capturing FTP Packets:

```
C:\Users\junai>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.
```

The image shows a Wi-Fi network analyzer interface, likely Wireshark, displaying a packet capture of an FTP session. The interface includes a menu bar (File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help) and a toolbar. The main display area is divided into three panes:

- Packet List:** Shows a list of captured packets. The selected packet is 66323, which is a TCP segment of a reassembled PDU.
- Packet Details:** Shows the details of the selected packet. It includes the Ethernet II header, Internet Protocol Version 4 header, and Transmission Control Protocol header.
- Packet Bytes:** Shows the raw bytes of the selected packet in hexadecimal and ASCII format.

The packet list shows the following details for the selected packet (66323):

No.	Time	Source	Destination	Protocol	Length	Info
66323	162.997786	192.168.0.175	170.114.15.46	TLSv1.2	285	Application Data

The packet details pane shows the following information:

- Ethernet II:** Src: Tp-LinkT_89:e7:a8 (08:07:b6:89:e7:a8), Dst: IntelCor_d6:31:6b (40:74:e0:d6:31:6b)
- Internet Protocol Version 4:** Src: 172.67.7.42, Dst: 192.168.0.175
- Transmission Control Protocol:** Src Port: 443, Dst Port: 53989, Seq: 1, Ack: 1, Len: 1440

The packet bytes pane shows the raw bytes of the packet in hexadecimal and ASCII format.

Capturing ARP Packets:

Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp

No.	Time	Source	Destination	Protocol	Length	Info
71609	183.568720	54.37.30.38	192.168.0.175	TCP	1494	2223 → 1137 [PSH, ACK] Seq=32540819 Ack=40091 Win=64128 Len=1440 [TCP segment of a reassembled PDU]
71610	183.561608	54.37.30.38	192.168.0.175	TCP	1494	2223 → 1137 [ACK] Seq=32542259 Ack=40091 Win=64128 Len=1440 [TCP segment of a reassembled PDU]
71611	183.561608	54.37.30.38	192.168.0.175	TLSv1.2	1494	Application Data [TCP segment of a reassembled PDU]
71612	183.561608	54.37.30.38	192.168.0.175	TCP	1494	2223 → 1137 [ACK] Seq=32545139 Ack=40091 Win=64128 Len=1440 [TCP segment of a reassembled PDU]
71613	183.561608	54.37.30.38	192.168.0.175	TCP	1494	2223 → 1137 [PSH, ACK] Seq=32546579 Ack=40091 Win=64128 Len=1440 [TCP segment of a reassembled PDU]
71614	183.561618	192.168.0.175	54.37.30.38	TCP	54	1137 → 2223 [ACK] Seq=40091 Ack=32548019 Win=2119680 Len=0
71615	183.562675	54.37.30.38	192.168.0.175	TLSv1.2	1494	Application Data [TCP segment of a reassembled PDU]
71616	183.562675	54.37.30.38	192.168.0.175	TCP	1494	2223 → 1137 [ACK] Seq=32549459 Ack=40091 Win=64128 Len=1440 [TCP segment of a reassembled PDU]
71617	183.562675	54.37.30.38	192.168.0.175	TCP	1494	2223 → 1137 [ACK] Seq=32550899 Ack=40091 Win=64128 Len=1440 [TCP segment of a reassembled PDU]
71618	183.562675	54.37.30.38	192.168.0.175	TLSv1.2	1494	Application Data [TCP segment of a reassembled PDU]
71619	183.562689	192.168.0.175	54.37.30.38	TCP	54	1137 → 2223 [ACK] Seq=40091 Ack=32553779 Win=2119680 Len=0
71620	183.562866	54.37.30.38	192.168.0.175	TCP	1494	2223 → 1137 [ACK] Seq=32553779 Ack=40091 Win=64128 Len=1440 [TCP segment of a reassembled PDU]
71621	183.562866	54.37.30.38	192.168.0.175	TLSv1.2	674	Application Data
71622	183.562881	192.168.0.175	54.37.30.38	TCP	54	1137 → 2223 [ACK] Seq=40091 Ack=32555839 Win=2119680 Len=0
71637	183.942423	192.168.0.175	172.67.72.209	TCP	55	[TCP Keep-Alive] 1134 → 443 [ACK] Seq=1181 Ack=37347 Win=131584 Len=1
71638	183.948275	172.67.72.209	192.168.0.175	TCP	66	[TCP Keep-Alive ACK] 443 → 1134 [ACK] Seq=37347 Ack=1182 Win=1680 Len=0 SLE=1181 SRE=1182
71644	184.068454	192.168.0.175	193.122.203.139	TLSv1.2	1454	Application Data
71646	184.273264	193.122.203.139	192.168.0.175	TLSv1.2	93	Application Data
71647	184.322728	192.168.0.175	193.122.203.139	TCP	54	53534 → 443 [ACK] Seq=120757 Ack=1431 Win=514 Len=0

> Frame 1: 1494 bytes on wire (11952 bits), 1494 bytes captured (11952 bits) on interface \Device\NPF_{90485CC5-6194-4E36-A1C7-32FEBE1728C0}, id 0

> Ethernet II, Src: Tp-LinkT_89:e7:a8 (d8:07:b6:89:e7:a8), Dst: IntelCor_d6:31:6b (40:74:e0:d6:31:6b)

> Internet Protocol Version 4, Src: 172.67.7.42, Dst: 192.168.0.175

> Transmission Control Protocol, Src Port: 443, Dst Port: 53989, Seq: 1, Ack: 1, Len: 1440

0000 40 74 e0 d6 31 6b d8 07 b6 89 e7 a8 08 00 45 00 @t..1k.....E-
0010 05 c8 e9 7f 40 00 35 06 e1 eb ac 43 07 2a c0 a8 ...@ 5-...C.*..
0020 00 af 01 bb d2 e5 9c 19 9e 18 09 8c 44 7c 50 10D|P..
0030 00 f5 5d 0f 00 00 17 03 03 20 1a 21 cc 08 9e f2 ...].....|.....
0040 9c e0 a6 0c 81 c1 2e 79 6c 64 06 7c 86 a0 1d b5 ...yld|.....
0050 2f 9b 66 ec 3c 18 00 60 91 04 28 0e d4 04 7b 5c /-f<...{...[\
0060 2b 32 91 8b d6 82 87 a4 62 64 08 5c af a3 fc 1f +2.....bd\.....
0070 1d 52 40 4a 28 67 38 4f b4 0f 99 46 87 45 3c 5d +R@)(g80...F<|
0080 94 78 2e 95 10 74 e2 ad b0 a0 ce c0 92 24 f3 32 -x...t...k...\$ 2
0090 44 78 b1 a0 d3 ea 26 a4 2d d6 82 47 9b a0 a2 84 -x...&...-G...
00a0 b8 fe 2c ff e3 23 00 d6 51 59 be 34 64 e0 09 f3 ...#...QYAd...
00b0 da 6d 1d 8d 13 3e 83 2e 58 9c 4c 23 21 cb 33 28 -m...>...X.L@!3(
00c0 85 ee 6f b5 68 c0 63 04 d8 12 1c 3d da 54 e2 29 -o.h.c...-T.)
00d0 8a d8 43 ff a0 67 0a 58 63 72 cc 79 8e 12 0a 13 --C.g.X cr-y.....

Address Resolution Protocol: Protocol

Packets: 71648 · Displayed: 67100 (93.7%)

Profile: Default

B) Tracing Packets based on filters:

1) Filter Results by Port:

Traces all packets related to Port 80.

Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.port == 80

No.	Time	Source	Destination	Protocol	Length	Info
10208	55.181214	192.168.0.175	23.47.229.231	TCP	66	1139 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
10209	55.183774	23.47.229.231	192.168.0.175	TCP	66	80 → 1139 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1440 SACK_PERM=1 WS=128
10209	55.183835	192.168.0.175	23.47.229.231	TCP	54	1139 → 80 [ACK] Seq=1 Ack=1 Win=132352 Len=0
10210	55.183880	192.168.0.175	23.47.229.231	HTTP	450	GET /MFewtZBNPEswTA3BgUrdgKcGUABRRz2bwARTMteY9aspRAZg5QFhaqQQUgrrrMPZF0nB9x6J13rX2F2ztUk1VB8CEHvXVsvNVTdixp9wSlab8X3D HTTP/1.1
10211	55.187221	23.47.229.231	192.168.0.175	HTTP	54	80 → 1139 [ACK] Seq=1 Ack=397 Win=64128 Len=0
10217	55.189187	23.47.229.231	192.168.0.175	HTTP	413	HTTP/1.1 304 Not Modified
10220	55.199463	192.168.0.175	118.214.137.233	TCP	66	1140 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
10221	55.202036	118.214.137.233	192.168.0.175	TCP	66	80 → 1140 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1440 SACK_PERM=1 WS=128
10222	55.202089	192.168.0.175	118.214.137.233	TCP	54	1140 → 80 [ACK] Seq=1 Ack=1 Win=132352 Len=0
10223	55.202194	192.168.0.175	118.214.137.233	HTTP	281	GET / HTTP/1.1
10224	55.205180	118.214.137.233	192.168.0.175	TCP	54	80 → 1140 [ACK] Seq=1 Ack=228 Win=64128 Len=0
10225	55.205189	118.214.137.233	192.168.0.175	HTTP	317	HTTP/1.1 304 Not Modified
10234	55.238964	192.168.0.175	23.47.229.231	TCP	54	1139 → 80 [ACK] Seq=397 Ack=360 Win=132096 Len=0
10245	55.254326	192.168.0.175	118.214.137.233	TCP	54	1140 → 80 [ACK] Seq=228 Ack=264 Win=132096 Len=0
10256	55.284987	192.168.0.175	183.87.86.186	TCP	66	1141 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
10260	55.289086	183.87.86.186	192.168.0.175	TCP	66	80 → 1141 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1440 SACK_PERM=1 WS=128
10261	55.289242	192.168.0.175	183.87.86.186	TCP	54	1141 → 80 [ACK] Seq=1 Ack=1 Win=132352 Len=0
10262	55.289470	192.168.0.175	183.87.86.186	HTTP	263	GET /ctnca.cr1 HTTP/1.1
10263	55.293143	183.87.86.186	192.168.0.175	TCP	54	80 → 1141 [ACK] Seq=1 Ack=210 Win=64128 Len=0
10264	55.304230	183.87.86.186	192.168.0.175	HTTP	263	HTTP/1.1 304 Not Modified

> Frame 10205: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{90485CC5-6194-4E36-A1C7-32FEBE1728C0}, id 0

> Ethernet II, Src: IntelCor_d6:31:6b (40:74:e0:d6:31:6b), Dst: Tp-LinkT_89:e7:a8 (d8:07:b6:89:e7:a8)

> Internet Protocol Version 4, Src: 192.168.0.175, Dst: 23.47.229.231

> Transmission Control Protocol, Src Port: 1139, Dst Port: 80, Seq: 0, Len: 0

0000 d8 07 b6 89 e7 a8 74 e0 d6 31 6b 08 00 45 00@t..1k...E-
0010 00 34 4e ac 40 00 06 00 00 c0 a8 00 af 17 2f -AN@.....-/
0020 e5 e7 04 73 00 50 7c 5c 0c 00 00 00 00 00 02 ...s-Pe(n.....
0030 fa f0 be 94 00 00 02 04 05 b4 01 03 03 08 01 01
0040 04 02 ..

Bytes 38-41: Sequence Number (raw) (tcp.seq.raw)

Packets: 77778 · Displayed: 57 (0.1%)

Profile: Default

2) Filter by Delta Time :

Displays tcp packets with delta time of greater than 0.500 sec

No.	Time	Source	Destination	Protocol	Length	Info
82035	252.268312	192.168.0.175	31.13.79.12	TLSv1.2	83	Application Data
82048	252.769902	192.168.0.175	54.37.30.38	TLSv1.2	1064	Application Data
82683	253.229608	192.168.0.175	170.114.15.46	TLSv1.2	285	Application Data
82687	254.132869	192.168.0.175	193.122.203.139	TLSv1.2	617	Application Data
82692	256.524622	192.168.0.175	31.13.79.12	TLSv1.2	355	Application Data
82694	256.589775	192.168.0.175	192.168.0.175	TLSv1.2	1309	Application Data
82697	256.679614	192.168.0.175	54.37.30.38	TLSv1.2	1064	Application Data
82702	256.796090	192.168.0.175	23.98.104.193	TLSv1.2	123	Application Data
83073	258.142808	192.168.0.175	3.108.46.16	TLSv1.2	108	Application Data
83076	258.187524	192.168.0.175	20.198.162.76	TLSv1.2	98	Application Data
83080	259.676895	192.168.0.175	74.125.68.188	TCP	55	[TCP Keep-Alive] 53990 → 5228 [ACK] Seq=1 Ack=1 Win=511 Len=1
83081	259.693361	192.168.0.175	193.122.203.139	TLSv1.2	150	Application Data
83083	259.771018	192.168.0.175	54.37.30.38	TLSv1.2	1064	Application Data
83508	260.069823	192.168.0.175	192.168.0.175	TLSv1.2	253	Application Data
83552	261.157340	192.168.0.175	20.198.162.76	TLSv1.2	98	Application Data
83553	261.222144	192.168.0.175	192.168.0.175	TLSv1.2	253	Application Data
83559	262.180441	192.168.0.175	54.37.30.38	TLSv1.2	1064	Application Data
84621	262.985171	192.168.0.175	192.168.0.175	TLSv1.2	173	Application Data
84623	264.134671	192.168.0.175	193.122.203.139	TLSv1.2	575	Application Data

> Frame 9930: 1130 bytes on wire (9040 bits), 1130 bytes captured (9040 bits) on interface \Device\NPF_{00485CC5-6194-4E36-A1C7-32FEBE1728C0}, id 0

> Ethernet II, Src: IntelCor_d6:31:6b (40:74:e0:d6:31:6b), Dst: Tp-LinkT_89:e7:a8 (d8:07:b6:89:e7:a8)

> Internet Protocol Version 4, Src: 192.168.0.175, Dst: 54.37.30.38

> Transmission Control Protocol, Src Port: 2223, Dst Port: 2223, Seq: 569, Ack: 153, Len: 1876

> Transport Layer Security

```

0000  d8 07 b6 89 e7 a8 40 74 e0 d6 31 6b 08 00 45 00  ....@t..1k..E
0010  04 5c b4 d9 40 00 80 06 00 00 c0 a8 00 af 36 25  \..@.....6%
0020  1e 26 04 71 00 af 23 ec fb 54 6c 12 43 6a 50 19  &q.0..TL..gP
0030  02 04 19 f1 00 00 17 03 03 04 2f 00 00 00 00 00  .....f.....
0040  00 00 01 8a cc 34 bf 9f 59 6c 2d 9a 6a a9 7b 79  ....4...Y!..j.y
0050  b2 49 ca 1b 27 2b 5f 07 2c c7 bd a9 b9 6e e4 d8  I..k.....n..
0060  46 b0 a5 e5 9f 84 70 61 8e 85 af 6e 97 5a 7d fc  F....va..n.Z..
0070  bc 0c 3d 74 f5 71 f9 89 e8 58 98 51 a3 e6 26 90  ...t.q...X.Q.&
0080  b6 fb 55 32 b6 d2 8f ff fb 7f 49 c5 24 68 82 a0  ..U2...I.Sh...
0090  da 6e fe 19 35 30 a8 29 c3 b0 43 69 00 43 93 b8  n..80..c.C.C...
00a0  a4 ad 43 af 8b 3c 45 c3 79 b8 ce 90 f6 a8 ed 94  ..C.C.E.y.....
00b0  de a3 e8 8e 48 f0 95 9a e9 d2 78 d9 94 54 f0 3b  ..H....x..j..
00c0  b9 4f 81 62 0b 00 b4 92 21 51 06 19 ee 97 c8 9b  ..b....lq.....
00d0  ba 78 82 0d 7f 8e 65 02 5d bb 03 6c 7c 59 e2 bd  x...e..]..nY..

```

wireShark_Wi-FiE730M1.pcapng Packets: 84628 Displayed: 402 (0.5%) Profile: Default

3) Filter by Byte Sequence:

Displays packets which contain a particular byte sequence.

No.	Time	Source	Destination	Protocol	Length	Info
7156	52.085070	54.37.30.38	192.168.0.175	TLSv1.2	1494	Ignored Unknown Record
19893	77.219988	13.107.136.9	192.168.0.175	TCP	1494	443 → 1154 [ACK] Seq=291504 Ack=893 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]
22950	77.523361	13.107.136.9	192.168.0.175	TCP	1494	443 → 1154 [ACK] Seq=3551408 Ack=893 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]
23905	77.632501	13.107.136.9	192.168.0.175	TCP	1494	443 → 1154 [ACK] Seq=4619760 Ack=893 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]
23911	77.633545	13.107.136.9	192.168.0.175	TLSv1.2	1494	Application Data [TCP segment of a reassembled PDU]
37292	79.148930	192.168.0.175	13.107.136.9	TLSv1.2	31734	Application Data, Application Data
66383	163.715506	54.37.30.38	192.168.0.175	TLSv1.2	1494	Ignored Unknown Record
66667	166.508943	54.37.30.38	192.168.0.175	TLSv1.2	1494	Ignored Unknown Record

> Frame 7156: 1494 bytes on wire (11952 bits), 1494 bytes captured (11952 bits) on interface \Device\NPF_{00485CC5-6194-4E36-A1C7-32FEBE1728C0}, id 0

> Ethernet II, Src: Tp-LinkT_89:e7:a8 (d8:07:b6:89:e7:a8), Dst: IntelCor_d6:31:6b (40:74:e0:d6:31:6b)

> Internet Protocol Version 4, Src: 54.37.30.38, Dst: 192.168.0.175

> Transmission Control Protocol, Src Port: 2223, Dst Port: 2223, Seq: 1839324, Ack: 4601, Len: 1440

> Transport Layer Security

```

0000  40 74 e0 d6 31 6b d8 07 b6 89 e7 a8 08 00 45 00  @t..1k.....E
0010  05 c8 49 df 40 00 2d 06 e8 ae 36 25 1e 26 c0 a8  ..I@.....6%&..
0020  00 af 08 af 04 72 44 c1 47 ba 94 0a 8d 8b 50 10  ....rD...G....P
0030  01 f5 ab f6 00 01 40 d6 0f e4 7c ed d9 cd 50  ....@.....P
0040  97 2a 22 d2 55 59 a1 bc 75 3a 08 35 12 5e a6 6a  **UV...u:5^..j
0050  e9 bb 8c 42 74 06 b5 6c ae ab e2 ab 79 85 f9 bf  ...Bt..l....y...
0060  fd 43 2e 4b 24 5b a4 50 7c 2f ff 28 d7 6f 8c 3c  .C.KS[P ]/.(o<
0070  f1 f9 a8 ae 1c bb 84 5e e1 4b 39 c4 f1 01 4a f1  ....^..K9...J
0080  30 28 29 60 e1 93 df 26 87 64 b1 67 7f 9d cd 9c  0()...&..d....
0090  f9 1a 60 e1 e5 af 80 7a 54 62 72 7a 33 36 cf 10  ....z.Tbrz36...
00a0  ae 06 9a 6c 2d b9 ff 83 05 ca d9 20 51 dc 6e 04  ...l....Q..n...
00b0  32 84 82 10 98 84 0b 53 13 d9 74 56 76 80 d5 bd  2.....S...TVv...
00c0  29 62 f0 59 68 5d ed 05 27 dd 1c ef 7e 71 84 6d  )b.Yh].....q..m
00d0  e2 78 b1 79 2e d3 80 08 e4 95 f3 57 2c 2d 2a b0  x.y.....w,~*..

```

wireShark_Wi-FiE730M1.pcapng Packets: 94100 Displayed: 8 (0.0%) Profile: Default

4) Filter by Source IP Address:

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

Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.src==13.107.136.9

No.	Time	Source	Destination	Protocol	Length	Info
15270	65.658380	13.107.136.9	192.168.0.175	ICMP	44	Echo (ping) reply id=0x0001, seq=406/38401, ttl=117 (request in 15269)
15272	65.664614	13.107.136.9	192.168.0.175	TCP	66	443 → 1144 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM=1
15275	65.668847	13.107.136.9	192.168.0.175	TCP	54	443 → 1144 [ACK] Seq=1 Ack=282 Win=4194384 Len=0
15276	65.672576	13.107.136.9	192.168.0.175	TCP	1494	443 → 1144 [ACK] Seq=1 Ack=282 Win=4194384 Len=1440 [TCP segment of a reassembled PDU]
15277	65.672576	13.107.136.9	192.168.0.175	TCP	1494	443 → 1144 [ACK] Seq=1441 Ack=282 Win=4194384 Len=1440 [TCP segment of a reassembled PDU]
15278	65.672576	13.107.136.9	192.168.0.175	TLSv1.2	1259	Server Hello, Certificate, Certificate Status, Server Key Exchange, Server Hello Done
15281	65.680980	13.107.136.9	192.168.0.175	TCP	54	443 → 1144 [ACK] Seq=4086 Ack=440 Win=4194048 Len=0
15282	65.680980	13.107.136.9	192.168.0.175	TLSv1.2	396	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
15284	65.686480	13.107.136.9	192.168.0.175	TCP	54	443 → 1144 [ACK] Seq=4428 Ack=651 Win=4194048 Len=0
15285	65.687223	13.107.136.9	192.168.0.175	TLSv1.2	812	Application Data
15287	65.694503	13.107.136.9	192.168.0.175	TCP	54	443 → 1144 [ACK] Seq=5186 Ack=857 Win=4193792 Len=0
15288	65.701458	13.107.136.9	192.168.0.175	TLSv1.2	831	Application Data
15289	65.701458	13.107.136.9	192.168.0.175	TCP	54	443 → 1144 [FIN, ACK] Seq=5963 Ack=857 Win=4193792 Len=0
15293	65.704461	13.107.136.9	192.168.0.175	TCP	54	443 → 1144 [ACK] Seq=5964 Ack=889 Win=4193792 Len=0
15295	65.708691	13.107.136.9	192.168.0.175	TCP	66	443 → 1145 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM=1
15298	65.711768	13.107.136.9	192.168.0.175	TCP	54	443 → 1145 [ACK] Seq=1 Ack=499 Win=4194048 Len=0
15299	65.713989	13.107.136.9	192.168.0.175	TCP	1494	443 → 1145 [ACK] Seq=1 Ack=499 Win=4194048 Len=1440 [TCP segment of a reassembled PDU]
15300	65.713989	13.107.136.9	192.168.0.175	TCP	1494	443 → 1145 [ACK] Seq=1441 Ack=499 Win=4194048 Len=1440 [TCP segment of a reassembled PDU]
15301	65.713989	13.107.136.9	192.168.0.175	TLSv1.2	1259	Server Hello, Certificate, Certificate Status, Server Key Exchange, Server Hello Done
15304	65.719227	13.107.136.9	192.168.0.175	TCP	54	443 → 1145 [ACK] Seq=4086 Ack=657 Win=4194048 Len=0
15305	65.721434	13.107.136.9	192.168.0.175	TLSv1.2	396	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
15307	65.725549	13.107.136.9	192.168.0.175	TCP	54	443 → 1145 [ACK] Seq=4428 Ack=867 Win=4193792 Len=0
15308	65.727085	13.107.136.9	192.168.0.175	TCP	1494	443 → 1145 [ACK] Seq=4428 Ack=867 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]
15309	65.727085	13.107.136.9	192.168.0.175	TCP	1494	443 → 1145 [ACK] Seq=5868 Ack=867 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]
15310	65.727085	13.107.136.9	192.168.0.175	TCP	1494	443 → 1145 [ACK] Seq=7308 Ack=867 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]
15311	65.727085	13.107.136.9	192.168.0.175	TCP	1494	443 → 1145 [ACK] Seq=8748 Ack=867 Win=4193792 Len=1440 [TCP segment of a reassembled PDU]

> Frame 15270: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Device\NPF_{90485CC5-6194-4E36-A1C7-32FEBE1728C0}, id 0

> Ethernet II, Src: Tp-LinkT_89:e7:a8 (d8:07:b6:89:e7:a8), Dst: IntelCor_d6:31:6b (40:74:e0:d6:31:6b)

> Internet Protocol Version 4, Src: 13.107.136.9, Dst: 192.168.0.175

> Internet Control Message Protocol

```
0000  40 74 e0 d6 31 6b d8 07 b6 89 e7 a8 08 00 45 00  @t..1k...E-
0010  00 1e 53 94 00 00 75 01 9b 7f 0d 6b 88 09 c0 a8  --S...u...k...
0020  00 af 00 00 fe 68 00 01 01 96 00 00             ....h....
```

wireshark_Wi-FiE730ML.pcapng

Packets: 106174 • Displayed: 21265 (20.0%)

Profile: Default

CONCLUSION

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

Experiment 9

Date of Performance :

SAP Id: 60004190057

Div: A

Date of Submission:

Name : Junaid Altaf Girkar

Batch : A4

Aim of Experiment

Implementation of Network Intrusion Detection System using NMAP, SNORT and IPTABLE (CO6).

Theory:

IPTables:

iptables is a user-space utility program that allows a system administrator to configure the IP packet filter rules of the Linux kernel firewall, implemented as different Netfilter modules. The filters are organized in different tables, which contain chains of rules for how to treat network traffic packets. Different kernel modules and programs are currently used for different protocols; iptables applies to IPv4, ip6tables to IPv6, arptables to ARP, and ebtables to Ethernet frames.

NMAP:

Nmap, short for Network Mapper, is a free, open-source tool for vulnerability scanning and network discovery. Network administrators use Nmap to identify what devices are running on their systems, discovering hosts that are available and the services they offer, finding open ports and detecting security risks. Nmap can be used to monitor single hosts as well as vast networks that encompass hundreds of thousands of devices and multitudes of subnets.

```
C:\Users\junai>nmap 10.120.63.29 -O -sV -p 20-25 -Pn
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-03 00:35 India
Standard Time
Nmap scan report for 10.120.63.29
Host is up.
```

```
PORT      STATE      SERVICE    VERSION
20/tcp    filtered  ftp-data
21/tcp    filtered  ftp
22/tcp    filtered  ssh
23/tcp    filtered  telnet
24/tcp    filtered  priv-mail
25/tcp    filtered  smtp
Too many fingerprints match this host to give specific OS details
```

```
OS and Service detection performed. Please report any incorrect
results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.49 seconds
```

```
C:\Users\junai>nmap 10.120.63.29 10.120.63.28 -sL
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-03 00:36 India
Standard Time
Nmap scan report for 10.120.63.29
Nmap scan report for 10.120.63.28
Nmap done: 2 IP addresses (0 hosts up) scanned in 0.10 seconds
```

```
C:\Users\junai>nmap 10.120.63.29 -p 21,22,23,25,80 -Pn
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-03 00:37 India
Standard Time
Nmap scan report for 10.120.63.29
Host is up.

PORT      STATE      SERVICE
21/tcp    filtered  ftp
22/tcp    filtered  ssh
23/tcp    filtered  telnet
25/tcp    filtered  smtp
80/tcp    filtered  http

Nmap done: 1 IP address (1 host up) scanned in 3.39 seconds
```

SNORT:

Snort is a free and open-source network intrusion prevention and detection system. It uses a rule-based language combining signature, protocol, and anomaly inspection methods to detect malicious activity such as denial-of-service (DoS) attacks, Buffer overflows, stealth port scans, CGI attacks, SMB probes, and OS fingerprinting attempts.

It is capable of performing real-time traffic analysis and packet logging on IP networks.

IP Protocols supported by SNORT:

As we know, IP is a unique address for every computer and is used for transferring data or packets over the internet from one network to the other network. Each packet contains a message, data, source, destination address, and much more. Snort supports three IP protocols for suspicious behavior:

- Transmission Control Protocol (TCP) Connects two different hosts and exchanges data between them. Examples include HTTP, SMTP, and FTP.
- User Datagram Protocol (UDP): Broadcasts messages over the internet. Examples include DNS traffic.
- Internet Control Message Protocol (ICMP): Sends network error messages in Windows. Examples include Ping and Traceroute.

Snort Rules:

Rules are a different methodology for performing detection, which bring the advantage of 0-day detection to the table. Developing a rule requires an acute understanding of how the vulnerability actually works. Snort generates alerts according to the rules defined in the configuration file. The Snort rule language is very flexible, and creation of new rules is relatively simple. Snort rules help in differentiating between normal internet activities and malicious activities

ICMP Intrusion Detection:

```
C:\Snort\bin>snort -c C:\Snort\etc\snort.conf -l C:\Snort\log -i2 -T
Running in Test mode

      === Initializing Snort ===
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "C:\Snort\etc\snort.conf"
PortVar 'HTTP_PORTS' defined : [ 80:81 311 383 591 593 901 1220 1414
1741 1830 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988 7000:7001
7144:7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118
8123 8180:8181 8243 8280 8300 8800 8888 8899 9000 9060 9080 9090:9091
9443 9999 11371 34443:34444 41080 50002 55555 ]
PortVar 'SHELLCODE_PORTS' defined : [ 0:79 81:65535 ]
PortVar 'ORACLE_PORTS' defined : [ 1024:65535 ]
PortVar 'SSH_PORTS' defined : [ 22 ]
PortVar 'FTP_PORTS' defined : [ 21 2100 3535 ]
PortVar 'SIP_PORTS' defined : [ 5060:5061 5600 ]
PortVar 'FILE_DATA_PORTS' defined : [ 80:81 110 143 311 383 591 593
901 1220 1414 1741 1830 2301 2381 2809 3037 3128 3702 4343 4848 5250
```

```
6988 7000:7001 7144:7145 7510 7777 7779 8000 8008 8014 8028 8080 8085
8088 8090 8118 8123 8180:8181 8243 8280 8300 8800 8888 8899 9000 9060
9080 9090:9091 9443 9999 11371 34443:34444 41080 50002 55555 ]
```

```
PortVar 'GTP_PORTS' defined : [ 2123 2152 3386 ]
```

```
Detection:
```

```
Search-Method = AC-Full-Q
```

```
Split Any/Any group = enabled
```

```
Search-Method-Optimizations = enabled
```

```
Maximum pattern length = 20
```

CONCLUSION

Thus, we have successfully implemented a Network Intrusion Detection System using NMAP, SNORT and IPTables.

Experiment 10

Date of Performance :

SAP Id: 60004190057

Div: A

Aim of Experiment

Implement Buffer Overflow Attack. (CO7)

Date of Submission:

Name : Junaid Altaf Girkar

Batch : A4

Theory:

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)

CODE:

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"
        },
        {
            "Junaid",
            "60004190057"
        },
        {
            "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:

```

Enter Username: Junaid
Enter Current Password: 60004190057
Enter New Password: qwerty1387934fkehfefoev
Admin | qwerty1387934fkehfefoev
oev | Qqkaif
Sally | Usfsmfs
Password Changed!
Continue? Y/N: Y
Enter New Password: abcd34758

```



```
Admin | qwerty1387934fkehfefoev
oev | abcd34758
Sally | Usfsmfs
Password Changed!
```

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: Junaid
Enter Current Password: 60004190056
Enter New Password:
qwerty123456789dbcburnvirvrBUFFEROVERFLOWATTACKpassword
Admin | qwerty123456789dbcb
Max | Qqkaif
Sally | Usfsmfs
Password Changed!

```

Splint Output for the Vulnerable Code:

```
C:\splint-3.1.2\bin>set include=C:/mingw-64/mingw32/bin
C:\splint-3.1.2\bin>splint -type -retvalother -predboolint
"C:\Study Sem 6\IS\Practicals splint_test.c" Splint 3.1.2 --- 25
Aug 2010
Study Sem 6\IS\Practicals\splint_test.c: (in function main)
C:\Study Sem 6\IS\Practicals\splint_test.c(22,37)
Use of gets leads to a buffer overflow vulnerability. Use fgets
instead
gets Use of function that may lead to buffer overflow. (Use
-bufferoverflowhigh to inhibit warning)
C:\Study Sem 6\CSS\Practicals\splint_test.c(23,45);
Use of gets leads to a buffer overflow vulnerability. Use fgets
instead:
gets
C:\Study Sem 6\IS\Pra
Parse Error. (For help on parse errors, see splint -help
parseerrors.) *** Cannot continue.
C:\splint-3.1.2\bin>
```

Splint Output for the Vulnerable Code

```
C:\splint-3.1.2\bin>set include=C:/mingw-64/mingw32/bin
C:\splint-3.1.2\bin>splint -type -retvalother -predboolint "
C:\Study Sem 6\CSS\Practicals splint_test_fixed.c" Splint 3.1.2
--- 25 Aug 2010
C:\Study Sem 6\CSS\Practicals splint_test_fixed.c(31,16); Parse
Error. (For help on parse errors, see splint -help parseerrors.),
*** Cannot continue.
C:\splint-3.1.2\bin>
```

CONCLUSION

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