|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ex. No** | **Date** | **Title** | **Page No** | **Signature** |
| 1 |  | Implement substitution and translation techniques |  |  |
| 2 |  | Implement the Hashing Algorithms |  |  |
| 3 |  | Write a program to implement a set of values to combine the control of Bell lapadula with the integrating controls of the Biba model |  |  |
| 4 |  | Installation of rootkits and study about various options |  |  |
| 5 |  | Implement hacking windows – windows login and password |  |  |
| 6 |  | Implement hacking windows – accessing restricted drivers |  |  |
| 7 |  | Implement cross site scripting and implement XSS |  |  |
| 8 |  | Implement SQL injection attack |  |  |
| 9 |  | Implement Buffer overflow attack |  |  |
| 10 |  | Understanding malware working principles, detection and prevention |  |  |
| 11 |  | Setup honeypot and monitor the honeypot network |  |  |
| 12 |  | Demonstrate intrusion detection system using any tool |  |  |

**Ex.No. 1**

**Implement the following substitution and translation techniques of the following Caesar Cipher, Playfair Cipher, Hill Cipher, ABCD Vigenere Cipher, Railfence Row and Column transformation**

**CAESAR CIPHER**

**Aim:**

To implement Caesar Substitution Cipher.

**Algorithm:**

Caesar Cipher is one of the earliest and the simplest method of substitution technique. Each letter of a given text is replaced by a letter some fixed number of positions down the alphabet.

**Input:**

1. A string of lower case letters, called Plain Text.
2. An integer denoting the required shift, called Key.

**Steps:**

1. Key = key % 26.
2. Traverse the given text one character at a time.
3. For each character, transform the given character as per the rule, depending on whether encryption or decryption is done.
   1. **Rule for encryption**

Cipher text = (Plain text + Key) % 26.

* 1. **Rule for decryption**

Cipher text = (Plain text + Key) % 26.

1. Return the generated string.

**Program:**

class caesar:

def encrypt(self, text, key):

newText = ""

key = key % 26

for i in text:

if i.isupper():

ch = ( ( ord(i) % 65 + key ) % 26 ) + 65

else:

ch = ( ( ord(i) % 97 + key ) % 26 ) + 97

newText += chr(ch)

print(newText)

def decrypt(self, text, key):

self.encrypt(text,26-key)

if \_\_name\_\_=='\_\_main\_\_':

pf = caesar()

print("Original text : Vikraman\nKey : 3")

print("Encrypted text", pf.encrypt("Vikraman",3))

print("Decrypted text", pf.decrypt("Ylnudpdq",3),"\n")

**Output:**

vikramans-MacBook-Pro:Desktop vikikkdi$ py 1.py

Original text : Vikraman

Key : 3

Encrypted text Ylnudpdq

Decrypted text Vikraman

**PLAYFAIR CIPHER**

**Aim:**

To implement Playfair Cipher.

**Algorithm:**

1. The key for playfair cipher is generally a word.
2. Any sequence of 25 letters can be used as a key.
3. Remove any punctuations/characters not part of grid.
4. Split the plain text into pairs.
5. Identify any double letter in the plain text and insert ‘z’
6. If there is an odd number of letters add ‘z’.
7. Break the plain text into pairs of letters.
8. Now apply the following rules.
   1. If both the letters are in the same column take letter below each one.
   2. If both the letters are in the same row take letter right to each one.
   3. If neither of the rules apply, form a rectangle with two letters and take letters of horizontal opposite.

**Program:**

class playfair:

def formKey(self, key):

k = []

di = {chr(i):False for i in range(ord('a'),ord('z')+1)}

for t in key:

if not di[t]:

k.append(t)

di[t] = True

ch = ord('a')

while len(k) != 25:

if chr(ch) == 'q':

ch += 1

continue

if not di[chr(ch)]:

k.append(chr(ch))

di[chr(ch)] = True

ch += 1

keys = [k[0:5],k[5:10],k[10:15],k[15:20],k[20:25]]

return keys

def formText(self, text):

if len(text) % 2 == 1:

text += 'z'

te = []

while len(text) != 0:

te.append(text[:2])

text = text[2:]

return te

def getIndex(self, key):

di = {chr(i):[0,0] for i in range(ord('a'),ord('z')+1)}

for i in range(5):

for j in range(5):

di[key[i][j]] = [i,j]

return di

def encryptUtil(self, newText, newKey):

index = self.getIndex(newKey)

cipher = ""

for i in newText:

a = i[0]

b = i[1]

indA = index[a]

indB = index[b]

if indA[0] == indB[0]:

x = indA[0]

yA = (indA[1] + 1) % 5

yB = (indB[1] + 1) % 5

cipher += newKey[x][yA]

cipher += newKey[x][yB]

elif indA[1] == indB[1]:

y = indA[1]

xA = (indA[0] + 1) % 5

xB = (indB[0] + 1) % 5

cipher += newKey[xA][y]

cipher += newKey[xB][y]

else:

xA, yA = indA[0], indB[1]

xB, yB = indB[0], indA[1]

cipher += newKey[xA][yA]

cipher += newKey[xB][yB]

print(cipher)

def decryptUtil(self, newText, newKey):

index = self.getIndex(newKey)

cipher = ""

for i in newText:

a = i[0]

b = i[1]

indA = index[a]

indB = index[b]

if indA[0] == indB[0]:

x = indA[0]

yA = (indA[1] - 1 + 5) % 5

yB = (indB[1] - 1 + 5) % 5

cipher += newKey[x][yA]

cipher += newKey[x][yB]

elif indA[1] == indB[1]:

y = indA[1]

xA = (indA[0] - 1 + 5) % 5

xB = (indB[0] - 1 + 5) % 5

cipher += newKey[xA][y]

cipher += newKey[xB][y]

else:

xA, yA = indA[0], indB[1]

xB, yB = indB[0], indA[1]

cipher += newKey[xA][yA]

cipher += newKey[xB][yB]

print(cipher)

def encrypt(self, text, key):

key = key.lower()

key = ''.join(key.split())

text = text.lower()

text = ''.join(text.split())

newKey = self.formKey(key)

newText = self.formText(text)

cipher = self.encryptUtil(newText, newKey)

def decrypt(self, text, key):

key = key.lower()

key = ''.join(key.split())

text = text.lower()

text = ''.join(text.split())

newKey = self.formKey(key)

newText = self.formText(text)

decipher = self.decryptUtil(newText, newKey)

if \_\_name\_\_=='\_\_main\_\_':

pf = playfair()

print("Original text : vikraman\nKey : macbook")

print("Encrypted text", pf.encrypt("Vikraman","macbook"))

print("Decrypted text", pf.decrypt("Ylnudpdq","macbook"),"\n")

**Output:**

vikramans-MacBook-Pro:Desktop vikikkdi$ py 1.py

Original text : vikraman

Key : macbook

Encrypted text whdpcaoi

Decrypted text vikraman

**HILL CIPHER**

**Aim:**

To implement Hill cipher.

**Algorithm:**

1. Generate the key matrix, where each character is represented by its equivalent number.
2. Turn the plain text into digraphs and generate the column vector.
3. Perform matrix multiplication modulo 26.
4. These letters are then converted back to produce cipher text.

**Program:**

import re

import random

import numpy as np

import math

def multiply(t,key):

l=[]

for i in range(0,len(t)):

l.append([])

l[i].append(ord(t[i])-65)

a = np.matrix(key)

b = np.matrix(l)

c = a\*b

ct=""

for i in c:

ct += chr(65+(i%26))

return ct

def encrypt(text,key):

cipher\_text=""

a = key

b =[]

i=0

while i<len(text):

cipher\_text+=multiply(text[i:i+len(key)],key)

i+=len(key)

return cipher\_text

def inv\_det(d):

for i in range(0,26):

if (i\*d)%26==1: return i

def divide(t,key):

l=[]

for i in range(0,len(t)):

l.append([])

l[i].append(ord(t[i])-65)

a = np.matrix(key)

b = np.matrix(l)

c = a\*b

c = c%26

ct=""

l=[]

l=c.tolist()

for i in range(len(l)):

ch = l[i]

ch = round(ch[0],0)

ch = int(ch)

l[i]=ch%26

for i in range(len(l)):

ch = l[i]

ct += chr(65+ch)

return ct

def decrypt(cipher,key):

k=np.matrix(key)

d = int(np.linalg.det(k))

inv=k.I

for i in range(0,len(inv)):

inv[i]\*=d

id = inv\_det(d)

inv[i]=(inv[i]\*id)%26

text=""

i=0

while i < len(cipher):

text += divide(cipher[i:i+len(key)],inv)

i+=len(key)

return text

def find\_key\_matrix(key):

l = int(math.sqrt(len(key)))

mat = []

for i in range(0,len(key),l):

li = []

for j in range(0,l):

li.append(ord(key[i+j])-ord('A'))

mat.append(li)

return mat

if \_\_name\_\_ == "\_\_main\_\_":

text = "VIK"

Key = "GYBNQKURP"

key = find\_key\_matrix(Key)

print("Text : " + text)

print("Key : ",key)

cipher = encrypt(text,key)

print("Cipher: " + cipher)

print("Decrpyt:" + decrypt(cipher,key))

**Output:**

vikramans-MacBook-Pro:Desktop vikikkdi$ py hill.py

Text : VIK

Key : [[6, 24, 1], [13, 16, 10], [20, 17, 15]]

Cipher: QHE

Decrpyt:VIK

**VIGENERE CIPHER**

**Aim:**

To implement ABCD Vigenere Cipher.

**Algorithm:**

1. Construct a 26 x 26 matrix containing all 26 alphabets.
2. Multiply the keyword to match the length of plain text.
3. Pair the text and the keyword and find the character in the matrix.

**Program:**

import re

def remove\_space(text):

text = re.sub(" ","",text)

return text

def format\_text(text):

text = text.upper()

text = remove\_space(text)

return text

def encrypt(text,key):

text = format\_text(text)

key = format\_text(key)

cipher\_text=""

k=0

for i in text:

val1 = ord(i)-65

val2 = ord(key[k])-65

val = (val1+val2)%26

c=chr(65+val)

cipher\_text+=c

k=(k+1)%(len(key))

return cipher\_text

def decrypt(cipher,key):

cipher = format\_text(cipher)

key = format\_text(key)

text=""

k=0

for i in cipher:

val1 = ord(i)-65

val2 = ord(key[k])-65

val = (val1-val2)%26

c=chr(65+val)

text+=c

k=(k+1)%(len(key))

return text

if \_\_name\_\_ == "\_\_main\_\_":

text = "Vikraman"

key = "mac"

print("Text : " + text)

print("Key : "+key)

cipher = encrypt(text,key)

print("Cipher: " + cipher)

print("Decrpyt:" + decrypt(cipher,key))

**Output:**

vikramans-MacBook-Pro:Desktop vikikkdi$ py vigenere.py

Text : Vikraman

Key : mac

Cipher: HIMDAOMN

Decrpyt:VIKRAMAN

**RAILFENCE CIPHER**

**Aim:**

To implement Railfence row and column transformation

**Algorithm:**

1. The plain text is written downward diagonally on successive rails of a fence.
2. On reaching bottom we traverse upwards diagonally.
3. After building the fence the cipher is formed by writing the text row wise.

**Program:**

import re

def remove\_space(text):

text = re.sub(" ","",text)

return text

def format\_text(text):

text = text.upper()

text = remove\_space(text)

return text

def encrypt(text,key):

text = format\_text(text)

cipher\_text=""

l=[]

for i in range(0,key):

l.append( [0]\*len(text) )

i=0

j=0

v=1

for c in text:

l[i][j]=c

i+=v

j+=1

if i >= key:

i-=2

v=-1

elif i <0:

i+=2

v=1

for i in range(0,key):

for j in range(0,len(text)):

if l[i][j]!=0:

cipher\_text+=l[i][j]

for i in l:

print(i)

return cipher\_text

def decrypt(cipher,key):

cipher = format\_text(cipher)

l=[]

for i in range(0,key):

l.append( [0]\*len(cipher) )

k=0

text=""

i=0

j=0

v=1

for c in cipher:

l[i][j]=1

i+=v

j+=1

if i >= key:

i-=2

v=-1

elif i <0:

i+=2

v=1

k=0

for i in range(0,key):

for j in range(0,len(cipher)):

if l[i][j] == 1:

l[i][j] = cipher[k]

k+=1

i=0

j=0

v=1

for c in cipher:

text+=l[i][j]

i+=v

j+=1

if i >= key:

i-=2

v=-1

elif i <0:

i+=2

v=1

return text

if \_\_name\_\_ == "\_\_main\_\_":

text = "Vikraman sathiyanarayanan"

key = 3

print("Text : " + text)

print("Key : ",key)

cipher = encrypt(text,key)

print("Cipher: " + cipher)

print("Decrpyt:" + decrypt(cipher,key))

**Output:**

vikramans-MacBook-Pro:Desktop vikikkdi$ py rail.py

Text : Vikraman sathiyanarayanan

Key : 3

['V', 0, 0, 0, 'A', 0, 0, 0, 'S', 0, 0, 0, 'I', 0, 0, 0, 'A', 0, 0, 0, 'A', 0, 0, 0]

[0, 'I', 0, 'R', 0, 'M', 0, 'N', 0, 'A', 0, 'H', 0, 'Y', 0, 'N', 0, 'R', 0, 'Y', 0, 'N', 0, 'N']

[0, 0, 'K', 0, 0, 0, 'A', 0, 0, 0, 'T', 0, 0, 0, 'A', 0, 0, 0, 'A', 0, 0, 0, 'A', 0]

Cipher: VASIAAIRMNAHYNRYNNKATAAA

Decrpyt:VIKRAMANSATHIYANARAYANAN

**Result:**

Thus translation and substitution ciphers have been implemented successfully.

**Ex.No. 2**

**Implement the following algorithms DES, RSA, Diffie, MD5, SHA1**

**DATA ENCRYPTION STANDARD**

**Aim:**

To implement Data Encryption Standard algorithm.

**Algorithm:**

The Data Encryption Standard (DES) is a symmetric-key block cipher.

**DES Encryption:**

1. Plaintext is broken into blocks of length 64 bits. Encryption is block wise.
2. A message block is first gone through an initial permutation IP, then divided into two parts L0,where L0 is the left part of 32 bits and R0 is the right part of the 32 bits
3. Round i has input Li-1,Ri-1 and output Li, Ri.

        Li = Ri-1, Ri = Li-1 ⊕ f(Ri-1,Ki)

        and Ki is the subkey for the 'i'th where 1 ≤ i ≤ 16

      L1 = R0,    R1 = L0 ⊕ f(R0,K1)

      L2 = R1,    R2 = L1 ⊕ f(R1,K2)

      L3 = R2,    R3 = L2 ⊕ f(R2,K3)

      ................         ..........................

      ................         ..........................

      ................         ..........................

      L16 = R15,    R16 = L15 ⊕ f(R15,K16)

1. After round 16, L16 and R16 are swapped, so that the decryption algorithm has the same structure as the encryption algorithm.
2. Finally, the block is gone through the inverse the permutation IP-1 and then output

**DES Decryption:**

1. In encryption, we have

      Li = Ri-1, Ri = Li-1 ⊕ f(Ri-1, Ki)

1. Ki is the subkey for the 'i'th round. Hence

    Ri-1 = Li, Li-1 = Ri ⊕ f(Li,Ki) for each 'i'

1. Due to swap operation after the 16th round encryption, the output of encryption is IP-1(R16,L16)
2. Equation(1) as follows:

      R15 = L16,    L15 = R16 ⊕ f(L16,K16)

      R14 = L15,    L14 = R15 ⊕ f(L15,K15)

      R13 = L14,    L13 = R14 ⊕ f(L14,K14)

      ................         ..........................

      ................         ..........................

      ................         ..........................

      R1 = L2,    L1 = R2 ⊕ f(L2,K2)

1. If we give IP-1(R16,L16) as the input for the same algorithm with round sub keys(K16,K15,......K1),then the output is IP-1(L0,R0),the original message block
2. Decryption is performed using the same algorithm, except the K16 is used as the first round,K15 in the second, and so on, with K1 used in the 16th round.

**Program:**

PI = [58, 50, 42, 34, 26, 18, 10, 2,

60, 52, 44, 36, 28, 20, 12, 4,

62, 54, 46, 38, 30, 22, 14, 6,

64, 56, 48, 40, 32, 24, 16, 8,

57, 49, 41, 33, 25, 17, 9, 1,

59, 51, 43, 35, 27, 19, 11, 3,

61, 53, 45, 37, 29, 21, 13, 5,

63, 55, 47, 39, 31, 23, 15, 7]

#Initial permut made on the key

CP\_1 = [57, 49, 41, 33, 25, 17, 9,

1, 58, 50, 42, 34, 26, 18,

10, 2, 59, 51, 43, 35, 27,

19, 11, 3, 60, 52, 44, 36,

63, 55, 47, 39, 31, 23, 15,

7, 62, 54, 46, 38, 30, 22,

14, 6, 61, 53, 45, 37, 29,

21, 13, 5, 28, 20, 12, 4]

#Permut applied on shifted key to get Ki+1

CP\_2 = [14, 17, 11, 24, 1, 5, 3, 28,

15, 6, 21, 10, 23, 19, 12, 4,

26, 8, 16, 7, 27, 20, 13, 2,

41, 52, 31, 37, 47, 55, 30, 40,

51, 45, 33, 48, 44, 49, 39, 56,

34, 53, 46, 42, 50, 36, 29, 32]

#Expand matrix to get a 48bits matrix of datas to apply the xor with Ki

E = [32, 1, 2, 3, 4, 5,

4, 5, 6, 7, 8, 9,

8, 9, 10, 11, 12, 13,

12, 13, 14, 15, 16, 17,

16, 17, 18, 19, 20, 21,

20, 21, 22, 23, 24, 25,

24, 25, 26, 27, 28, 29,

28, 29, 30, 31, 32, 1]

#SBOX

S\_BOX = [

[[14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7],

[0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8],

[4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0],

[15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13],

],

[[15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10],

[3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5],

[0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15],

[13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9],

],

[[10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8],

[13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1],

[13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7],

[1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12],

],

[[7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15],

[13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9],

[10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4],

[3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14],

],

[[2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9],

[14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6],

[4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14],

[11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3],

],

[[12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11],

[10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8],

[9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6],

[4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13],

],

[[4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1],

[13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6],

[1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2],

[6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12],

],

[[13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7],

[1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2],

[7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8],

[2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11],

]

]

#Permut made after each SBox substitution for each round

P = [16, 7, 20, 21, 29, 12, 28, 17,

1, 15, 23, 26, 5, 18, 31, 10,

2, 8, 24, 14, 32, 27, 3, 9,

19, 13, 30, 6, 22, 11, 4, 25]

#Final permut for datas after the 16 rounds

PI\_1 = [40, 8, 48, 16, 56, 24, 64, 32,

39, 7, 47, 15, 55, 23, 63, 31,

38, 6, 46, 14, 54, 22, 62, 30,

37, 5, 45, 13, 53, 21, 61, 29,

36, 4, 44, 12, 52, 20, 60, 28,

35, 3, 43, 11, 51, 19, 59, 27,

34, 2, 42, 10, 50, 18, 58, 26,

33, 1, 41, 9, 49, 17, 57, 25]

#Matrix that determine the shift for each round of keys

SHIFT = [1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1]

def string\_to\_bit\_array(text):

array = list()

for char in text:

binval = binvalue(char, 8)

array.extend([int(x) for x in list(binval)])

return array

def bit\_array\_to\_string(array):

res = ''.join([chr(int(y,2)) for y in [''.join([str(x) for x in bytes]) for bytes in nsplit(array,8)]])

return res

def binvalue(val, bitsize):

binval = bin(val)[2:] if isinstance(val, int) else bin(ord(val))[2:]

if len(binval) > bitsize:

raise "binary value larger than the expected size"

while len(binval) < bitsize:

binval = "0"+binval

return binval

def nsplit(s, n):

return [s[k:k+n] for k in range(0, len(s), n)]

ENCRYPT=1

DECRYPT=0

class des():

def \_\_init\_\_(self):

self.password = None

self.text = None

self.keys = list()

def run(self, key, text, action=ENCRYPT, padding=False):

if len(key) < 8:

raise "Key Should be 8 bytes long"

elif len(key) > 8:

key = key[:8]

self.password = key

self.text = text

if padding and action==ENCRYPT:

self.addPadding()

elif len(self.text) % 8 != 0:

raise "Data size should be multiple of 8"

self.generatekeys()

text\_blocks = nsplit(self.text, 8)

result = list()

for block in text\_blocks:

block = string\_to\_bit\_array(block)

block = self.permut(block,PI)

g, d = nsplit(block, 32)

tmp = None

for i in range(16):

d\_e = self.expand(d, E)

if action == ENCRYPT:

tmp = self.xor(self.keys[i], d\_e)

else:

tmp = self.xor(self.keys[15-i], d\_e)

tmp = self.substitute(tmp)

tmp = self.permut(tmp, P)

tmp = self.xor(g, tmp)

g = d

d = tmp

result += self.permut(d+g, PI\_1)

final\_res = bit\_array\_to\_string(result)

if padding and action==DECRYPT:

return self.removePadding(final\_res)

else:

return final\_res

def substitute(self, d\_e):

subblocks = nsplit(d\_e, 6)

result = list()

for i in range(len(subblocks)):

block = subblocks[i]

row = int(str(block[0])+str(block[5]),2)

column = int(''.join([str(x) for x in block[1:][:-1]]),2)

val = S\_BOX[i][row][column]

bin = binvalue(val, 4)

result += [int(x) for x in bin]

return result

def permut(self, block, table):

return [block[x-1] for x in table]

def expand(self, block, table):

return [block[x-1] for x in table]

def xor(self, t1, t2):

return [x^y for x,y in zip(t1,t2)]

def generatekeys(self):

self.keys = []

key = string\_to\_bit\_array(self.password)

key = self.permut(key, CP\_1)

g, d = nsplit(key, 28)

for i in range(16):

g, d = self.shift(g, d, SHIFT[i])

tmp = g + d

self.keys.append(self.permut(tmp, CP\_2))

def shift(self, g, d, n):

return g[n:] + g[:n], d[n:] + d[:n]

def addPadding(self):

pad\_len = 8 - (len(self.text) % 8)

self.text += pad\_len \* chr(pad\_len)

def removePadding(self, data):

pad\_len = ord(data[-1])

return data[:-pad\_len]

def encrypt(self, key, text, padding=False):

return self.run(key, text, ENCRYPT, padding)

def decrypt(self, key, text, padding=False):

return self.run(key, text, DECRYPT, padding)

if \_\_name\_\_ == '\_\_main\_\_':

key = "macboook"

text= "Vikraman"

d = des()

r = d.encrypt(key,text)

r2 = d.decrypt(key,r)

print("Text :",text)

print("Key :",key)

print ("Ciphered: %r" % r)

print ("Deciphered: ", r2)

**Output:**

vikramans-MacBook-Pro:Desktop vikikkdi$ py des.py

Text : Vikraman

Key : macboook

Ciphered: '9\x88ax¿\x15\x10\x12'

Deciphered: Vikraman

**RSA ALGORITHM**

**Aim:**

To implement RSA asymmetric cryptographic algorithm.

**Algorithm:**

1. **Generate public key:**
   1. Select two prime numbers, p and q.
   2. Public key is composed of n and e, where e is an integer, n = p \* q;
   3. e is not a factor of n, 1 < e < theta(n).
2. **Generate private key:**
   1. Theta(n) = (p-1) \* (q-1)
   2. Private key d = (k \* theta(n) + 1) / e, k is an integer.
3. **Encryption and Decryption:**
   1. CT = pow (PT, e) mod n.
   2. PT = pow (CT, d) mod n.

**Program:**

import re

import random

import math

def remove\_space(text):

text = re.sub(" ","",text)

return text

def format\_text(text):

text = text.upper()

text = remove\_space(text)

return text

def is\_prime(x):

if x == 2:

return True

else:

for number in range (3,x):

if x % number == 0 or x % 2 == 0:

#print number

return False

return True

def get\_prime(no):

i = 2

count = 1

if no == 1:

return i

while True:

if is\_prime(i)==True:

count+=1

if count==no:

return i

i+=1

def gcd(x, y):

while(y):

x, y = y, x % y

return x

def get\_public\_key1():

p = get\_prime(random.randint(1,50))

q = get\_prime(random.randint(1,50))

return p\*q

def phi(n):

result = 1

for i in range(2, n):

if (gcd(i, n) == 1):

result+=1

return result

def get\_public\_key2(n):

totient = phi(n)

i =n-1

while i >0:

if gcd(i,totient)==1:

return i

i-=1

raise "Co-prime not found exception"

def to\_cipher(m,e,n):

c = ord(m)-65

v = c\*\*e

v = v%n

return int(v)

def encrypt(text,key):

text = format\_text(text)

e,n = key[0],key[1]

cipher\_text =[]

for i in text:

cipher\_text.append(to\_cipher(i,e,n))

return cipher\_text

def inv\_mod(a,b):

for i in range(1,b):

if (a\*i)%b == 1:

return i

raise "Imverse not found"

def find\_d(e,n):

totient = phi(n)

return inv\_mod(e,totient)

def to\_normal\_text(val,d,n):

val = val\*\*d

val = val % n

return chr(65+val)

def decrypt(cipher,key):

text =""

e,n = key[0],key[1]

d = find\_d(e,n)

for i in cipher:

val = i#ord(i)-65

text += to\_normal\_text(val,d,n)

return text

if \_\_name\_\_ == "\_\_main\_\_":

text = "Vikraman sathiyanarayanan"

n = get\_public\_key1()

e = get\_public\_key2(n)

key = (e,n)

print("Text :",text)

print("Key :",key)

cipher = encrypt(text,key)

print("Encrypted text :",cipher)

print("Decrypted text :",decrypt(cipher,key))

**Output:**

vikramans-MacBook-Pro:Desktop vikikkdi$ py rsa.py

Text : Vikraman sathiyanarayanan

Key : (12365, 12367)

Encrypted text : [8857, 9708, 7436, 4891, 0, 606, 0, 10134, 4578, 0, 7627, 3745, 9708, 1551, 0, 10134, 0, 4891, 0, 1551, 0, 10134, 0, 10134]

Decrypted text : VIKRAMANSATHIYANARAYANAN

**DIFFIE HELLMAN ALGORITHM**

**Aim:**

To implement Diffie Hellman Algorithm.

**Algorithm:**

1. **Sender side**
   1. Public key = P, G
   2. Private key = a
   3. Key generated, x = pow(G, a) mod P
   4. Exchange generated keys.
   5. Key received = y
   6. Generated secret key, ka = pow(y, a) mod P
2. **Receiver side**
   1. Public keys = P, G
   2. Private key = b
   3. Key generated, x = pow(G, b) mod P
   4. Exchange generated keys.
   5. Key received = x
   6. Generated secret key, kb = pow(x, b) mod P

**Program:**

import math

import random

def is\_prime(x):

if x == 2:

return True

else:

for number in range (3,x):

if x % number == 0 or x % 2 == 0:

return False

return True

def get\_prime(no):

i = 2

count = 1

if no == 1:

return i

while True:

if is\_prime(i)==True:

count+=1

if count==no:

return i

i+=1

def calculate\_y(x,g,p):

v = g\*\*x

return int(v % p)

def calculate\_z(y,x,p):

v =y\*\*x

return int(v%p)

def differ\_hillman(p,g):

Xa = random.randint(1,p)

Xb = random.randint(1,p)

Ya = calculate\_y(Xa,g,p)

Yb = calculate\_y(Xb,g,p)

Za = calculate\_z(Yb,Xa,p)

Zb = calculate\_z(Ya,Xb,p)

print p,g

print Xa,Ya

print Xb,Yb

if Za == Zb:

print "Sucessful key\_exchange between a and b"

else:

print "key\_exchange failed"

if \_\_name\_\_== "\_\_main\_\_":

p = get\_prime(random.randint(1,100))

g = random.randint(1,100)

differ\_hillman(p,g)

**Output:**

vikramans-MacBook-Pro:Desktop vikikkdi$ py diffie.py

Prime numbers : 509 95

Private and public key of A 128 91

Private and punlic key of B 419 276

Sucessful key\_exchange between a and b

**MD5 ALGORITHM**

**Aim:**

To implement MD5 algorithm.

**Algorithm:**

1. Divide the input text into blocks of 512 bits each.
2. 64 bits are inserted at the end of last block.
3. These 64 bits record the length of original input.
4. MD5 helper function
   1. **The Buffer**

MD5 uses a buffer that is made up of 4 words that are 32 bits long.

* 1. **The Table**

MD5 uses a table k that has 64 elements. Each number i is indicated as Ki

Ki = abs (sin (i + 1)) \* 2^32

* 1. **Four auxiliary function**

F(X, Y, Z) = (X && Y) || (!(X) && Z)

G(X, Y, Z) = (X && Y) || (Y && !(Z))

H(X, Y, Z) = (X ^ Y ^ Z)

I(X, Y, Z) = Y ^ (X && !(Z))

**Program:**

def to\_bin(val):

b = bin(val)

b = b[2:len(b)-2]

return b

def get\_bits(t):

l = []

for i in t:

val = ord(i)

b = to\_bin(val)

b = "0"\*(8-len(b))+b

for bit in b:

l.append(bit)

return l

def padding(text):

l = get\_bits(text)

rem = len(l)%512

pad = 448-rem

t = [0]\*(pad-1)

t = [1]+t

l +=t

rem = len(get\_bits(text))%(2\*\*64)

b = to\_bin(rem)

b = "0"\*(64-len(b))+b

for bit in b:

l.append(bit)

i = 0

x=[]

while i < len(l):

x.append(l[i:i+32])

i+=32

return l

def hexa(val):

h = hex(val)

return h[2:len(h)-2]

def f1(x,y,z):

return (x & y) | (~x & z)

def f2(x,y,z):

return (x & z) | (y & ~z)

def f3(x,y,z):

return x^y^z

def f4(x,y,z):

return y ^ (x | ~z)

def md5(text):

l = padding(text)

a = 2\*\*10+100

b = 2\*\*9+89

c = 2\*\*4+3789

d = 2\*\*10+900

f = (f1,f2,f3,f4)

hash\_value = ""

words=[]

i = 0

while i < len(l):

words.append(l[i:i+512])

i+=512

hl = []

for word in words:

k = 0

i =0

x=[]

while i < len(word):

sl = word[i:i+8]

s=""

for ch in sl:

s+=str(ch)

x.append(int(s,2))

i+=8

for i in range(4):

fun = f[i]

for j in range(16):

val = fun(b,c,d)

a = b + a + val+x[k]

a = a%2\*\*32

t = a

a = b

b = c

c = d

d = t

k+=1

s = hexa(a)+hexa(b)+hexa(c)+hexa(d)

return s

if \_\_name\_\_=="\_\_main\_\_":

text = "md5 is a hash algorithm"

hash\_text =md5(text)

print hash\_text

**Output:**

vikramans-MacBook-Pro:Desktop vikikkdi$ py md5.py

Text : Vikraman sathiyanarayanan is a good boy

Hashed text : 18a2374b1aa06ba42fc61624

**SHA1 ALGORITHM**

**Aim:**

To implement SHA1 algorithm.

**Algorithm:**

**Input:** Any input whose length is less than 2^64 bits.

**Output:** Output 160 bits

1. Padding of bits.
2. Append length.
3. Divide the input into blocks of 512 bits.
4. Initialize chaining variables.

A = 01 23 45 67

B = 89 AB CD EF

C = FE DC BA 98

D = 76 53 32 98

E = C3 D2 E1 F0

1. Process Blocks
   1. Copy chaining variables into a-e variables.
   2. Divide current 512 bit block into 16 sub-blocks of 32 bits.
   3. SHA has 4 rounds, each 20 steps, 3 inputs, 512 bit block, register abcde.

|  |  |
| --- | --- |
| **ROUND** | **VALUE OF t** |
| 1 | 1 and 19 |
| 2 | 20 and 39 |
| 3 | 40 and 59 |
| 4 | 60 and 79 |

* 1. abcde = (e + process p + s^5 (a) + w[t], k[t]), a, s^30(b), c, d.
  2. S^t = circular shift left of 32 bit block by t bit.

|  |  |
| --- | --- |
| **ROUND** | **PROCESS p** |
| 1 | (b & c) | (~b & d) |
| 2 | b ^ c ^ d |
| 3 | (b & c) | (b & d) | (c & d) |
| 4 | b ^ c ^d |

**Program:**

def to\_bin(val):

b = bin(val)

b = b[2:len(b)-2]

return b

def hexa(val):

h = hex(val)

return h[2:len(h)-2]

def get\_bits(t):

l = []

for i in t:

val = ord(i)

b = to\_bin(val)

b = "0"\*(8-len(b))+b

for bit in b:

l.append(bit)

return l

def padding(text):

l = get\_bits(text)

rem = len(l)%512

pad = 448-rem

t = [0]\*(pad-1)

t = [1]+t

l +=t

rem = len(get\_bits(text))%(2\*\*64)

b = to\_bin(rem)

b = "0"\*(64-len(b))+b

for bit in b:

l.append(bit)

return l

def circularshift(n,b):

s = bin(n)

s = s[2:]

b = b%len(s)

s = s[b:len(s)]+s[0:b]

return int(s,2)

def hexa(val):

h = hex(val)

return h[2:len(h)-2]

def f(x,y,z,i):

if i<20:

return (x & y) | (~y & z)

elif i <40:

return x^y^z

elif i<60:

return (x&y)|(y&z)|(x&z)

elif i<80:

return x^y^z

def k(i):

s1 = 2\*\*10+100

s2 = 2\*\*9+89

s3 = 2\*\*4+3789

s4 = 2\*\*10+900

if i<20:

return s1

elif i <40:

return s2

elif i<60:

return s3

elif i<80:

return s4

def sha1(text):

l = padding(text)

h1 = 2\*\*8+98

h2 = 2\*\*7+76

h3 = 2\*\*22+332

h4 = 2\*\*6+5

h5 = 2\*\*7+21

words=[]

i = 0

while i < len(l):

words.append(l[i:i+512])

i+=512

for word in words:

w=[]

i=0

while i<len(word):

s=""

for c in word[i:i+32]:

s+=str(c)

w.append(int(s,2))

i+=32

for i in range(16,80):

val = w[i-3]^w[i-8]^w[i-14]^w[i-16]

val = circularshift(val,1)

w.append(val)

a=h1

b=h2

c=h3

d=h4

e=h5

for t in range(0,80):

temp = circularshift(a,5)+f(b,c,d,i)+w[i]+k(t)

e = d

d = c

c = circularshift(b,30)

b = a

a = temp

h1 = h1+a

h2 = h2+b

h3 = h3+c

h4 = h4+d

h5 = h5+e

return hexa(h1)+hexa(h2)+hexa(h3)+hexa(h4)+hexa(h5)

if \_\_name\_\_=="\_\_main\_\_":

text = "sha1 is a hash algorithm"

hash\_text =sha1(text)

print hash\_text

**Output:**

vikramans-MacBook-Pro:Desktop vikikkdi$ py sha1.py

Text : Vikraman sathiyanarayanan is a good boy

Hashed text:906f0b6db2b7b812754947c36c2a00bae7322413ff259fbe371d31b25ed36b7c6

**Result:**

This DES, RSA, Diffie Hellman, MD5 and SHA1 algorithms have been implemented successfully.

**Ex.No. 3**

**Write a program to implement a set of values to combine the control of Bell lapadula with the integrating controls of the Biba model**

**Aim:**

To write a program to implement a set of values to combine the control of Bell lapadula with the integrating controls of the Biba model.

**Algorithm:**

**Bell lapadula**

1. State machine model that describes a set of access control rules which use security labels on objects and clear access for subjects.
2. Security labels.
   1. Top secret.
   2. Secret
   3. Confidentiality.
3. Properties.
   1. No read up
   2. No write down
   3. Discretionary security property

**Biba model**

1. Biba model has a lattice structure.
2. Defined on mathematical basis that allows security level dedicated by Bella lapdula.
3. Biba policy uses three defining properties to protect objects from illegitimately modified.
   1. Simply integrity.
   2. Star integrity.
   3. Invocation.

**Program:**

stringToLevel = {

'topsecret': 0,

'secret': 1,

'confidential': 2,

'unclassified':3

}

levelToString = {v:k for k,v in stringToLevel.items()}

class sFile:

def \_\_init\_\_(self, name, level):

self.name = name

self.level = level

self.content='’

class sUser:

def \_\_init\_\_(self, name, password, level):

self.name = name

self.password = password

self.level = level

class sFileSystem:

def \_\_init\_\_(self):

self.files={}

def createFile(self, filename, level):

self.files[filename] = sFile(filename, level)

def openFile(self, filename, user, mode):

reqFile = self.files[filename]

if mode == 'r':

if user.level <= reqFile.level:

return reqFile

else:

print('Cannot read up')

elif mode == 'w':

if user.level >= reqFile.level:

return reqFile

else:

print('Cannot write down')

def showFiles(self):

print('File\t\tLevel')

for key, value in self.files.items():

print('{}\t{}'.format(key, levelToString[value.level]))

class sUserSystem:

def \_\_init\_\_(self):

self.users = {}

def createUser(self, name, password, level):

self.users[name] = sUser(name, password, level)

def loginUser(self, name, password):

user = self.users[name]

if user.password == password:

return user

else:

print("Wrong Password")

class sComputer:

def \_\_init\_\_(self):

self.userSystem = sUserSystem()

self.userSystem.createUser('vikraman', 'password', 0)

self.userSystem.createUser('akshay', 'pass1234', 1)

self.userSystem.createUser('timothy', '12345678', 2)

self.userSystem.createUser('gowtham', 'passpass', 3)

self.fileSystem = sFileSystem()

self.fileSystem.createFile('LVL0.txt', 0)

self.fileSystem.createFile('LVL1.txt', 1)

self.fileSystem.createFile('LVL2.txt', 2)

self.fileSystem.createFile('LVL3.txt', 3)

def start(self):

user = None

while True:

if user is None:

choice = int(input('Menu:\n1.Create User\n2.Login\n3.Shutdown\nChoice: '))

if choice == 1:

print('Enter name, password, level(topsecret, secret, confidential, unclassified):')

name, password, level = input().split()

self.userSystem.createUser(name, password, stringToLevel[level]);

elif choice == 2:

print('Enter name, password:')

name, password = input().split()

user = self.userSystem.loginUser(name, password)

else:

break

elif user is not None:

choice = int(input('Logged in as {} with {} access\nMenu:\n1.Create File\n2.Read File\n3.Write File\n4.Show Files\n5.Logout\nChoice: '.format(user.name, levelToString[user.level])))

if choice == 1:

print('Enter filename, level(topsecret, secret, confidential, unclassified):')

filename, level = input().split()

self.fileSystem.createFile(name, stringToLevel[level], user)

elif choice == 2:

filename = input('Enter filename:\n')

opFile = self.fileSystem.openFile(filename, user, 'r')

if opFile:

print(opFile.content)

elif choice == 3:

filename = input('Enter filename:\n')

opFile = self.fileSystem.openFile(filename, user, 'w')

if opFile:

opFile.content += input('Enter content:\n')

elif choice == 4:

self.fileSystem.showFiles()

else:

user = None

computer = sComputer()

computer.start()

**Output:**

Menu:

1.Create User

2.Login

3.Shutdown

Choice: 2

Enter name, password:

Vikraman 2345

Logged in as Vikraman with confidential access

Menu:

1.Create File

2.Read File

3.Write File

4.Show Files

5.Logout

Choice: 4

File Level

LVL3.txt unclassified

LVL1.txt secret

LVL0.txt topsecret

LVL2.txt confidential

Logged in as Vikraman with confidential access

Menu:

1.Create File

2.Read File

3.Write File

4.Show Files

5.Logout

Choice: 2

Enter filename:

LVL0.txt

Cannot read up

Logged in as Vikraman with confidential access

Menu:

1.Create File

2.Read File

3.Write File

4.Show Files

5.Logout

Choice: 3

Enter filename:

LVL3.txt

Cannot write down

Logged in as Vikraman with confidential access

Menu:

1.Create File

2.Read File

3.Write File

4.Show Files

5.Logout

Choice: 5

Menu:

1.Create User

2.Login

3.Shutdown

Choice: 3

**Result:**

Thus a program to implement a set of values to combine the control of Bell laPadula with the integrating controls of the Biba models has been implemented successfully.

**Ex.No. 4**

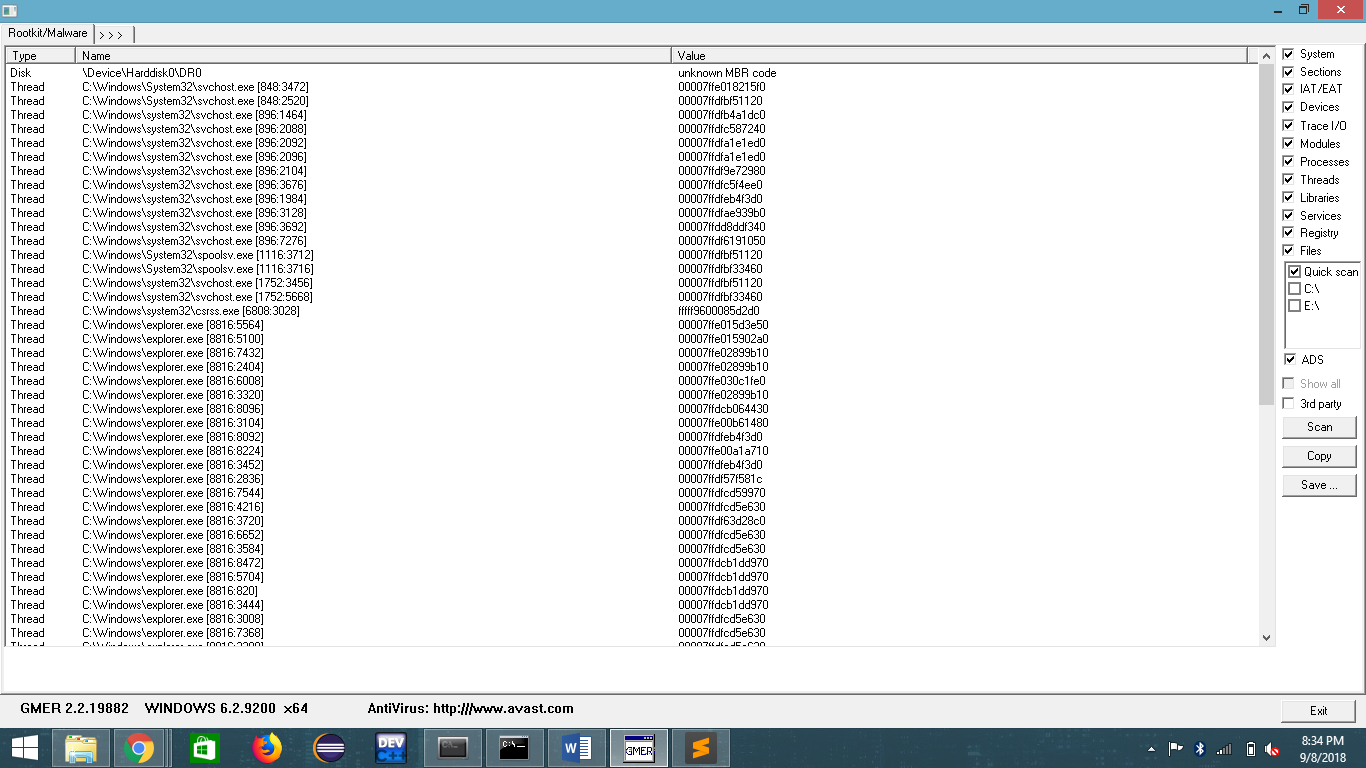
**Installation of rootkits and the study about various options**

**Aim:**

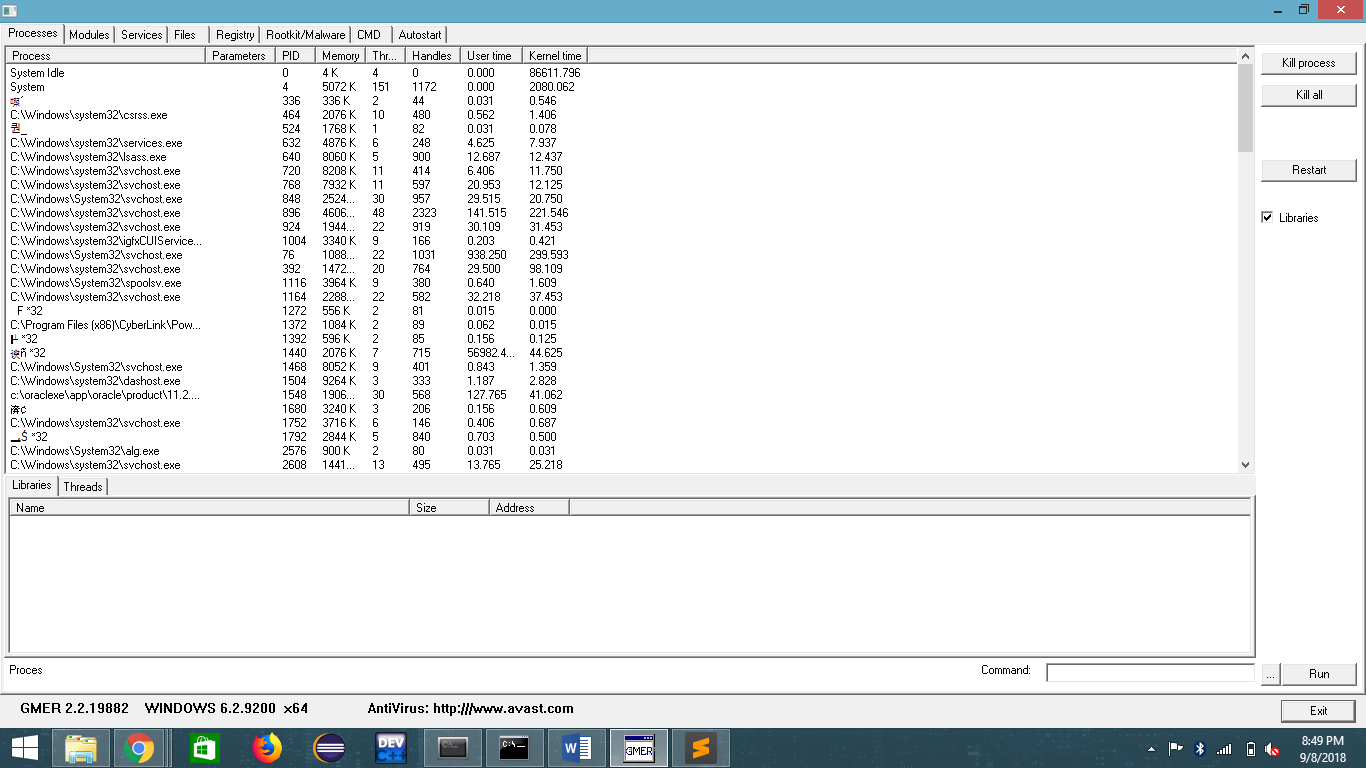
To install rootkits and study the various options available.

**Procedure:**

1. Download the rootkit tool from the GMER website.
2. This tool displays the following options, Processes, Modules, Services, Files, Registry, Rootkit/Malware, Auto start, cmd of localhost.
3. Select Processes menu and kill any unwanted processes.
4. Modules menu display the various system files like .sys, .dll.
5. Services menu display the complete services running with Autostart, Enable, Disable, System, Boot.
6. Files menu display full files on hard-disk volumes.
7. Rootkits/Malware scans the local drivers selected.
8. Autostart displays the registry base Autostart applications.
9. CMD allows the user to interact wth command line utilities or registry.



**Output:**



**Result:**

Thus implementation of rootkits and the study about various experiments has been implemented successfully.

## Ex. No. 5

## Implement Hacking windows - Windows login password

**Aim:**

To login to windows 7 by hacking the administrator account password.

**Procedure:**

* Turn on the UPS and the CPU, tap F8 continuously on the boot screen to get some windows start-up options.
* Choose "**Start windows normally**" option and turn the UPS off immediately.
* Then turn on the PC again, let it load.
* After that you will be prompted with two options in the boot screen (again), select the first option - "**Launch Start-up Repair(recommended)**"



* Let it load and Scan for issues.
* After few min, It will ask to restore defaults, select "**Cancel**" option.
* After few minutes, an error report screen will pop-up, asking to send information or not.
* Click on **"View Problem Details"** arrow, scroll down to the end of the report, then click a link stating **X**:\windows\something\something (the link starts with an**"X"**)
* Another Window will pop-up, and will look like a notepad
* Click File on the Menu-Bar, then select Open, and another window will pop-up
* Navigate to C: drive (or the drive on which windows is installed), click Windows, then System32, after that click on the arrow beside the "**File Type**" option and select "**all files**"
* Then search for a file named **"sethc"**(this is the shortcut to sticky keys), rename it to something else (Eg: sethc.bak)
* Search for cmd, create a copy and rename the copy as **"sethc"**
* Close everything, restart the PC, go to the log-in screen, press shift 5 times, until a cmd (command prompt) pops-up.



* Type in **"net user administrator /active:yes"**, and this will activate the default administrator account of the PC.
* Change/delete/manage/reset passwords from there.
* We can directly change passwords from cmd by typing "**net user (admin/any admin account's name) \*”**
* For Example: “**net user student \***”
* Once the new password is set, login normally in the login screen using the new password

**Result:**

Hence, access to administrator account was hacked by setting a new password and we can login to the system.

## Ex. No. 6

## Implement Hacking windows - Accessing Restricted drives

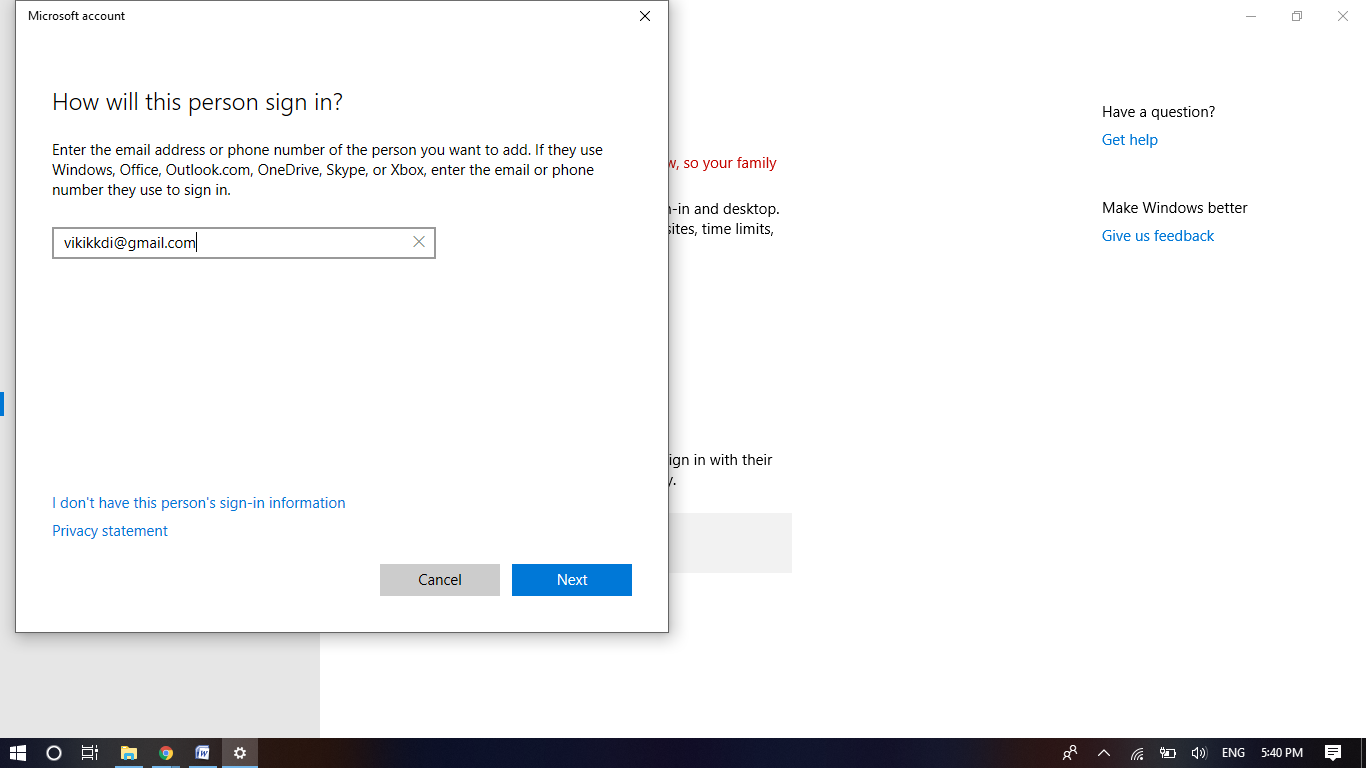
**Aim:**

To access drives that are restricted to standard users by the administrator

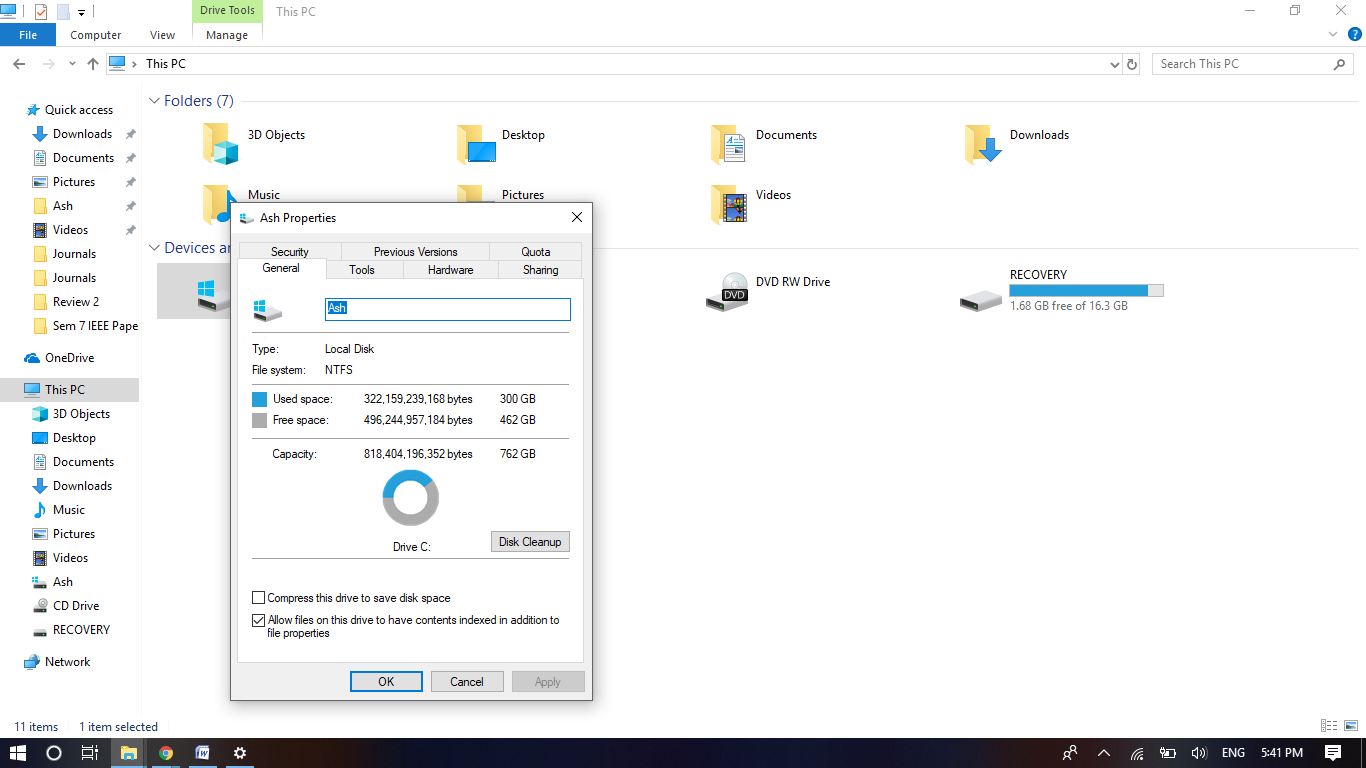
**Procedure:**

**Step 1 - Creating Restricted Drives as Administrator:**

* Log on to your computer with an account with Administrator rights. Click “Start,” type “user” (without quotes) in the automatically selected “Search programs and files” search box and click “User Accounts.” Click “Manage another account.”



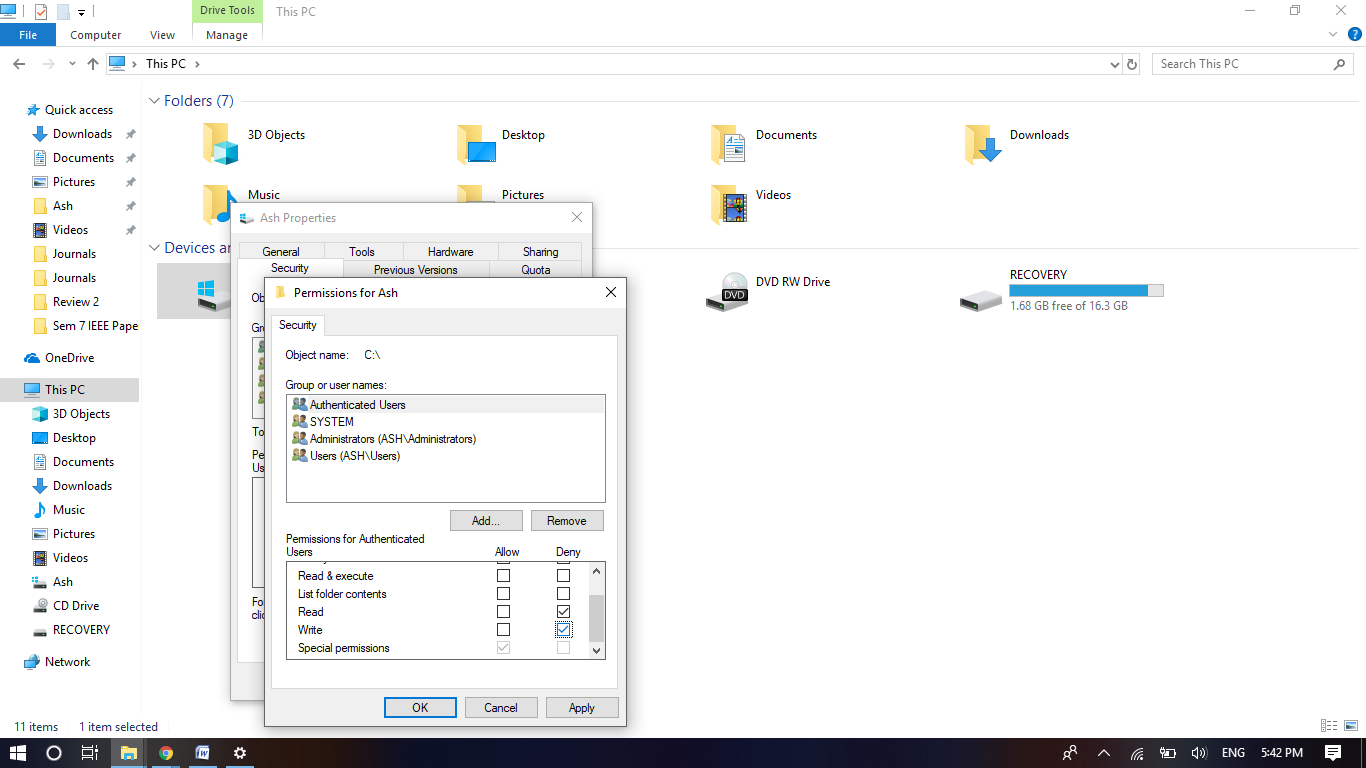
* Click “Create a new account,” if you need to create a user account for other people that will be using the computer. If you already have another account set up, go to the next step. You need to have at least your user account and another one set up to restrict access to a drive. Type a name for the user and click “Create Account.”
* Click “Start” and “Computer.” Right-click the name of the hard drive you want to restrict access to. Click “Properties.”
* Click the “Security tab” in the “Properties” window that opened. Click “Edit…” and “Add…” in the “Select Users or Groups” window that opened.
* Type the name of the other user account on your computer or you can click “Advanced” and then “Find Now”, then select your user account. Click “OK.”
* Uncheck the boxes to the left of any options that you do not want the user to have available. Check the “Deny” box for “Full control” to disable all control from the user for files on the hard drive.



* Click “OK,” “Yes” and “OK.” Close any open windows. Click “Start,” log off of your account and log on as the other user to test your settings.
* Click “Start,” “Computer” and double-click the name of the hard drive you restricted access to. A window indicating that “Access is denied” is shown. Close the window and log off the computer.

**Step 2 - Removing restrictions as a standard user:**

* Find the Administrator password or set a new password using the method described in the previous experiment.
* Click “Start” and “Computer.” Right-click the name of the hard drive for which you want to remove access restrictions. Click “Properties”.
* Click the “Security tab” in the “Properties” window that opened. Enter the new administrator password when prompted.
* Check the boxes to the left of any options that you do not want the user to have available. Check the “Allow” box for “Full control” to enable all control from the user for files on the hard drive.



* Click “OK,” “Yes” and “OK.”
* Click “Start,” “Computer” and double-click the name of the hard drive for which restrictions were removed.
* Now the contents of the drive will be displayed and the ‘Access is Denied’ window no longer occurs.

**Result:**

Hence, the restrictions on the hard drive were removed and the contents can be accessed.