1. Write a C program that contains a string (char pointer) with a value \Hello World'. The program should XOR each character in this string with 0 and display the result.

PROGRAM:

```
#include<stdlib.h>
#include<string.h>
main()
{
char str[]="010010";
char str1[11];
int i,len;
len=strlen(str);
for(i=0;i<len;i++)
{
str1[i]=str[i]^0;
printf("%c",str1[i]);
}
printf("\n");
}
```

Output:

010010
010010

2. Analyze the packets using Wireshark and sniff the User credentials by capturing the TCP traffic using suitable commands and options in the Wireshark.

Execution Steps:

● In Wireshark, select the Ethernet interface to capture the packets and select any one of these packets, right-click and hover on conversation filter and select TCP.

● Once done analyze the TCP Packets.

2. Capturing TCP Packets with

browser : Steps:

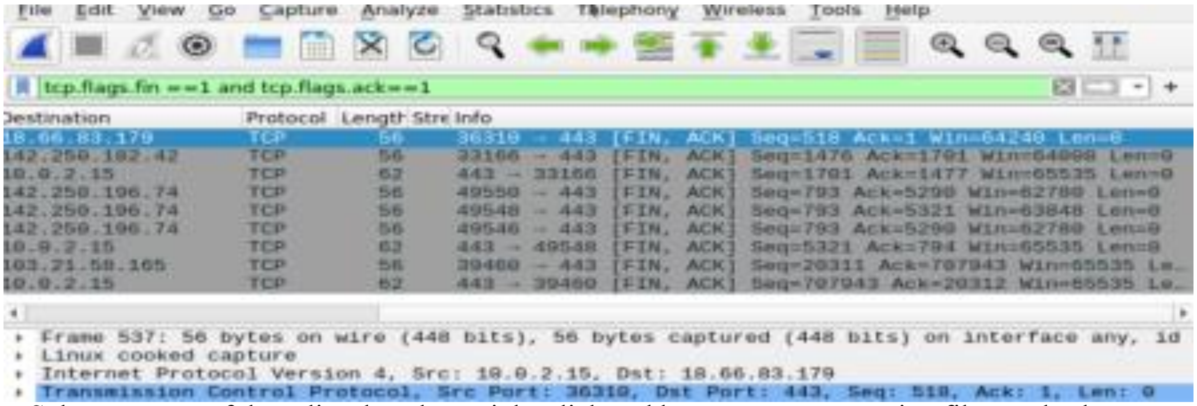
● Open Wireshark and double-click on **any-interface** to start the packet capture process.

● Open the browser and enter any website's fully qualified domain name in the browser address bar and hit enter.

● After the site is fully loaded, stop the capturing process, in Wireshark.

● Type the following in, apply a filter column and hit-enter :

tcp.flags.fin==1 and tcp.flags.ack ==1



- Select any one of these listed packets, right-click and hover on conversation filter and select TCP.
- Once done analyze the TCP Packets.
- Now to find the packet which contains the User credential we use the following commands in the display filter: **tcp.flags.push==1** and also look for **POST method in the HTTP request**.

3. Download and install nmap. Use it to perform Version detection, TCP connect scan, do a ping scan etc.

Theory:

Nmap (Network Mapper) is a security scanner originally written by Gordon Lyon (also known by his pseudonym Fyodor Vaskovich) used to discover hosts and services on a computer network, thus creating a "map" of the network. To accomplish its goal, Nmap sends specially crafted packets to the target host and then analyzes the responses. Unlike many simple port scanners that just send packets at some predefined constant rate, Nmap accounts for the network conditions (latency fluctuations, network congestion, the target interference with the scan) during the run. Also, owing to the large and active user community providing feedback and contributing to its features, Nmap has been able to extend its discovery capabilities beyond simply figuring out whether a host is up or down and which ports are open and closed; it can determine the operating system of the target, names and versions of the listening services, estimated uptime, type of device, and presence of a firewall. Nmap features include:

Host Discovery – Identifying hosts on a network. For example, listing the hosts which respond to pings or have a particular port open.

Port Scanning – Enumerating the open ports on one or more target hosts.

Version Detection – Interrogating listening network services listening on remote devices to determine the application name and version number.

OS Detection – Remotely determining the operating system and some hardware characteristics of network devices.

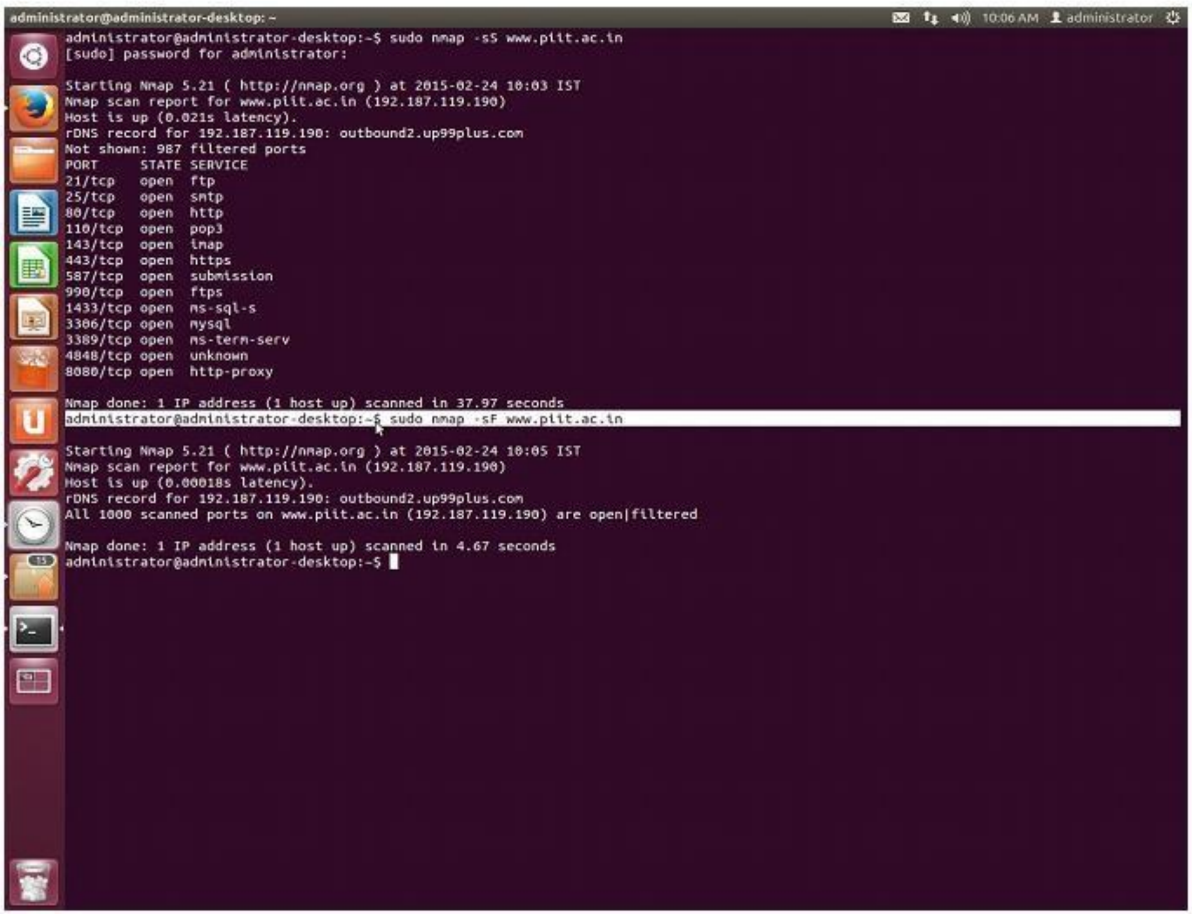
Basic commands working in Nmap For target specifications:

nmap <target‘s URL or IP with spaces between them>

For OS detection: nmap -O <target-host's URL or IP>
For version detection: nmap -sV <target-host's URL or IP>
After the installation of nmap:> sudo apt-get install nmap

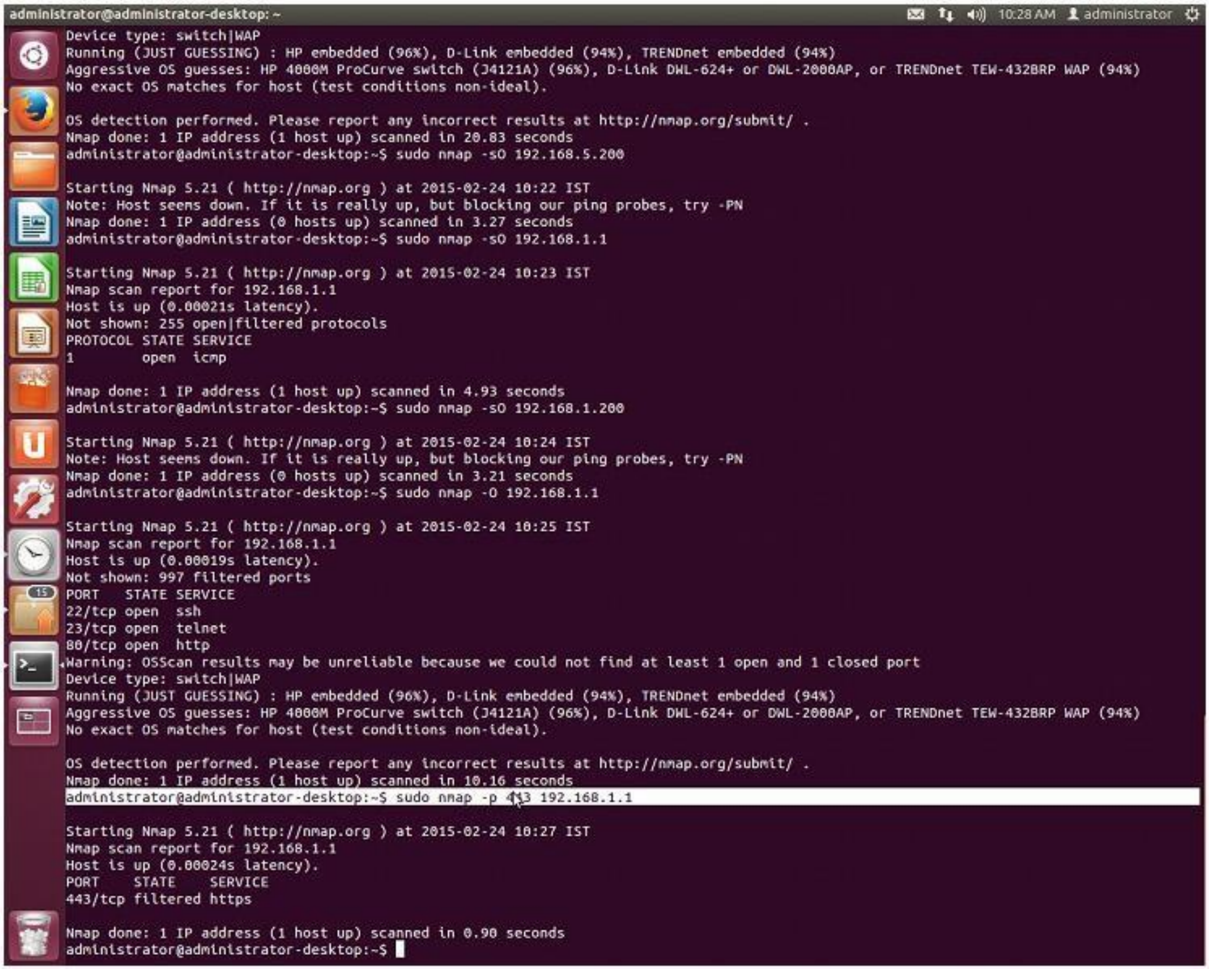
SYN scan is the default and most popular scan option for good reasons. It can be performed quickly, scanning thousands of ports per second on a fast network not hampered by restrictive firewalls. It is also relatively unobtrusive and stealthy since it never completes TCP connections.

Version detection:



-sV (Version detection) :Enables version detection, as discussed above. Alternatively, we can use -A, which enables version detection among other things.

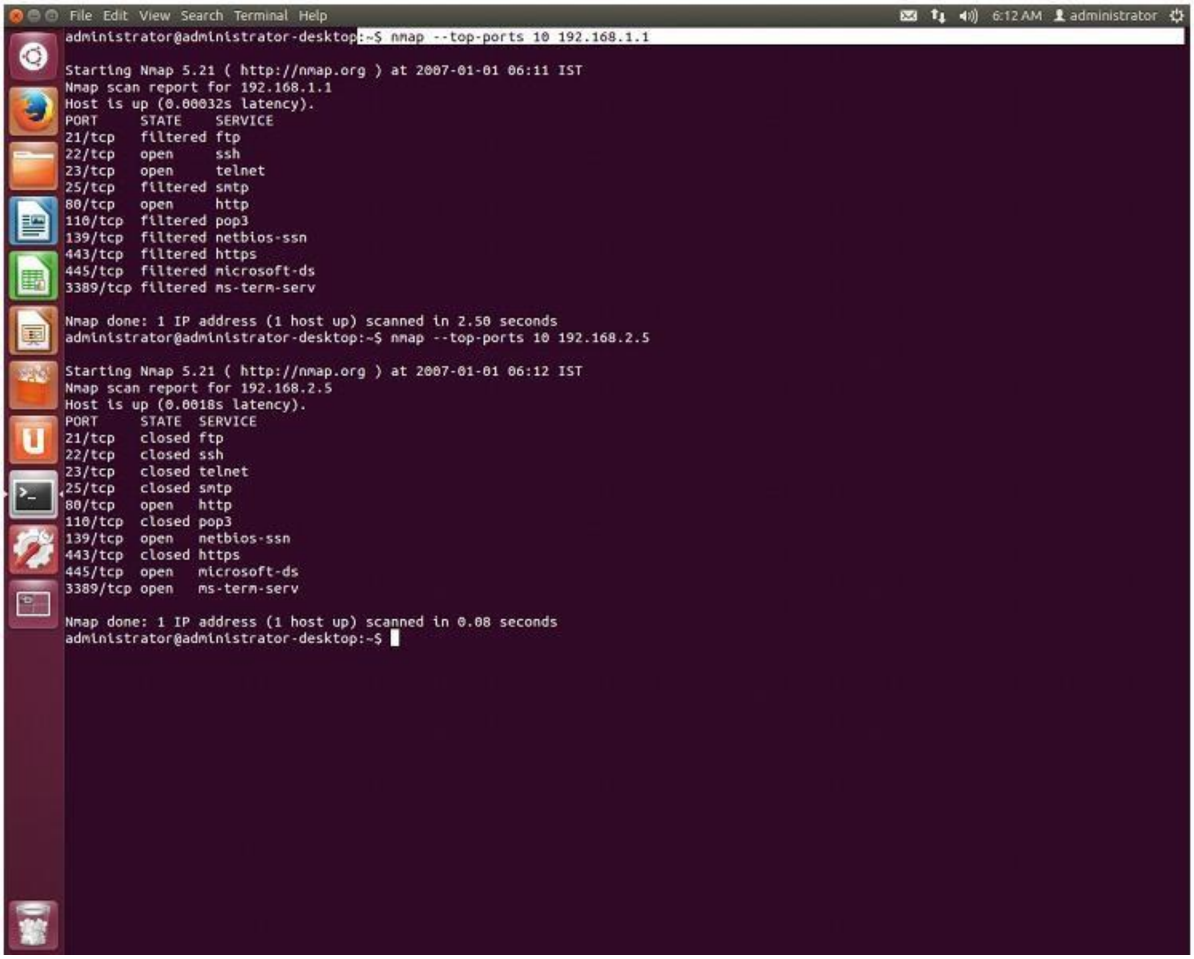
TCP Connect Scan:



-sT (TCP connect scan)

TCP connect scan is the default TCP scan type when SYN scan is not an option. This is the case when a user does not have raw packet privileges or is scanning IPv6 networks. Insteadof writing raw packets as most other scan types do, Nmap asks the underlying operating system to establish a connection with the target machine and port by issuing the connect system call. Along with spoofing.

TCP-SYN PING scan:



-PS port list (TCP SYN Ping) .

This option sends an empty TCP packet with the SYN flag set. The default destination port is 80 (configurable at compile time by changing DEFAULT_TCP_PROBE_PORT_SPEC in nmap.h). Alternate ports can be specified as a parameter. The syntax is the same as for the -p except that port type specifiers like T: are not allowed.

4. Implement Caesar cipher using suitable programming and show the successful decryption of Ciphertext and verify the same with “Cryptool”.

THEORY:

The Caesar cipher is one of the earliest known and simplest ciphers. It is a type of substitution cipher in which each letter in the plaintext is 'shifted' a certain number of places down the alphabet. For example, with a shift of 1, A would be replaced by B, B would become C, and so on. The method is named after Julius Caesar, who apparently used it to communicate with his generals. The Caesar cipher involves replacing each letter of the alphabet with the letter standing three places further down the alphabet.

For example,

```
plain: meet me after the toga party
cipher: PHHW PH DIWHU WKH WRJD SDUWB
```

Note that the alphabet is wrapped around, so that the letter following Z is A. We can define the transformation by listing all possibilities, as follows:

```
plain: a b c d e f g h i j k l m n o p q r s t u v w x y z
cipher: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

Let us assign a numerical equivalent to each letter:

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

Then the algorithm can be expressed as follows. For each plaintext letter, substitute the cipher text letter C2:

$C = \mathrm{E}(3, p) = (p + 3) \bmod 26$

A shift may be of any amount, so that the general Caesar algorithm is

$C = \mathrm{E}(k, p) = (p + k) \bmod 26$

Where takes on a value in the range 1 to 25.The decryption algorithm is simply

$p = \mathrm{D}(k, C) = (C - k) \bmod 26$

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

```
void main(){
char pt[50], ct[50], dt[50];
int key, len, i;
printf("Enter message to be encrypted : ");
scanf("%s", pt);
len = strlen(pt);
printf("\nEnter Key : ");
scanf("%d", &key);
if(key > 26){
printf("\nInvalid key. Please enter valid key.\n");
}
for(i = 0; i < len; i++){
ct[i] = pt[i] + key;
if(ct[i] > 122)
ct[i] = ct[i] - 26;
}
ct[i] = '\0';
printf("\nEncrypted message is, %s", ct);
for(i = 0; i < len; i++) {
dt[i] = ct[i] - key;
if(dt[i] < 97)
dt[i] = dt[i] + 26;
}
dt[i] = '\0';
printf("\n\nDecrypted message is, %s", dt);
getch();
}
```

```
Enter message to be encrypted : after

Enter Key : 3

Encrypted message is, diwhu

Decrypted message is, after

...Program finished with exit code 0
Press ENTER to exit console.
```

5. To write a C program to implement the hill cipher substitution technique, show encryption and decryption process for the Plaintext.

Description:

Each letter is represented by a number modulo 26. Often the simple scheme A = 0, B= 1... Z = 25, is used, but this is not an essential feature of the cipher. To encrypt a message, each block of n letters is multiplied by an invertible n × n matrix, against modulus 26. To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption. The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible n × n matrices (modulo 26).

$$
\begin{aligned}
\begin{pmatrix} B & A & C \\ K & U & P \\ A & B & C \end{pmatrix}\begin{pmatrix} r \\ e \\ t \end{pmatrix} = \begin{pmatrix} 1 & 0 & 2 \\ 10 & 20 & 15 \\ 0 & 1 & 2 \end{pmatrix}\begin{pmatrix} 17 \\ 4 \\ 19 \end{pmatrix} \\
= \begin{pmatrix} 1 \times 17 + 0 \times 4 + 2 \times 19 \\ 10 \times 17 + 20 \times 4 + 15 \times 19 \\ 0 \times 17 + 1 \times 4 + 2 \times 19 \end{pmatrix} \\
= \begin{pmatrix} 55 \\ 535 \\ 42 \end{pmatrix} \\
= \begin{pmatrix} 3 \\ 15 \\ 16 \end{pmatrix} mod\ 26 \\
= \begin{pmatrix} D \\ P \\ Q \end{pmatrix}
\end{aligned}
$$

Program:

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

int main()

{

unsigned int a[3][3]={{6,24,1},{13,16,10},{20,17,15}};

unsigned int b[3][3]={{8,5,10},{21,8,21},{21,12,8}};

int i,j, t=0;

unsigned int c[20],d[20]; char msg[20];

printf("Enter plain text\n ");

scanf("%s",msg);

for(i=0;i<strlen(msg);i++)

{
```

```
   c[i]=msg[i]-65;
printf("%d ",c[i]);
}
for(i=0;i<3;i++)
{
   t=0;
for(j=0;j<3;j++)
{
t=t+(a[i][j]*c[j]);
}
d[i]=t%26;
}
printf("\nEncrypted Cipher Text:");
for(i=0;i<3;i++)
printf(" %c",d[i]+65);
for(i=0;i<3;i++)
{
t=0;
for(j=0;j<3;j++)
{
t=t+(b[i][j]*d[j]);
}
c[i]=t%26;
}
printf("\nDecrypted Cipher Text:");
for(i=0;i<3;i++)
printf(" %c",c[i]+65);
getch();
return 0;
```

}

Output:

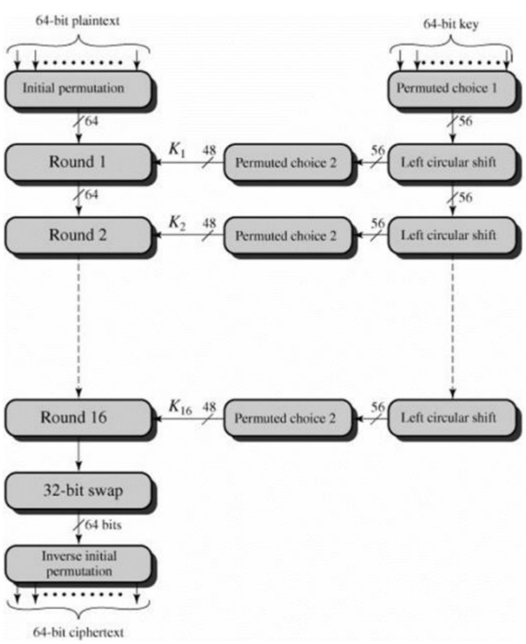
```
Enter plain text
 ABC
0 1 2
Encrypted Cipher Text: A K V
Decrypted Cipher Text: A B C

...Program finished with exit code 0
Press ENTER to exit console.
```

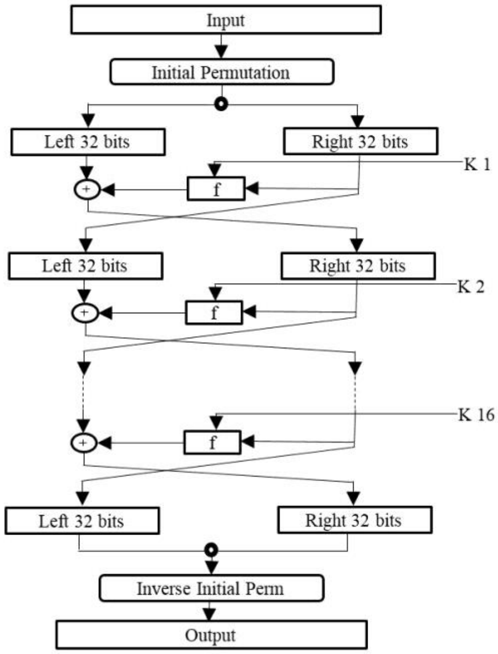
6. To write a program to implement Data Encryption Standard (DES) using any Language.

   Description:
   DES is a symmetric encryption system that uses 64-bit blocks, 8 bits of which are used for parity checks. The key therefore has a "useful" length of 56 bits, which means that only 56 bits are actually used in the algorithm. The algorithm involves carrying out combinations, substitutions and permutations between the text to be encrypted and the key, while making sure the operations can be performed in both directions. The key is ciphered on 64 bits and made of 16 blocks of 4 bits, generally denoted k1 to k16. Given that "only" 56 bits are actually used for encrypting, there can be 256 different keys.

   Data Encryption Standard diagram



Inside view of the Round function:



Program:

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import java.util.Base64;

class Main {

  Cipher ecipher;
  Cipher dcipher;

  Main(SecretKey key) throws Exception {
    ecipher = Cipher.getInstance("DES");
    dcipher = Cipher.getInstance("DES");
    ecipher.init(Cipher.ENCRYPT_MODE, key);
    dcipher.init(Cipher.DECRYPT_MODE, key);
  }

  public String encrypt(String str) throws Exception {
    // Encode the string into bytes using utf-8
    byte[] utf8 = str.getBytes("UTF8");
```

```
 // Encrypt
    byte[] enc = ecipher.doFinal(utf8);

    // Encode bytes to base64 to get a string

    return Base64.getEncoder().encodeToString(enc);

  }

  public String decrypt(String str) throws Exception {
    // Decode base64 to get bytes
    byte[] dec = Base64.getDecoder().decode(str);

    byte[] utf8 = dcipher.doFinal(dec);

    // Decode using utf-8
    return new String(utf8, "UTF8");
  }

  public static void main(String[] argv) throws Exception {
    final String secretText = "www.reva.edu.in";
    System.out.println("SecretText: " + secretText);
    SecretKey key = KeyGenerator.getInstance("DES").generateKey();
    Main encrypter = new Main(key);
    String encrypted = encrypter.encrypt(secretText);
    System.out.println("Encrypted Value: " + encrypted);
    String decrypted = encrypter.decrypt(encrypted);
    System.out.println("Decrypted: " + decrypted);

  }
}
```

Output:

```
SecretText: www.reva.edu.in
Encrypted Value: /XrGTpaKWYfGpBeotA88RQ==
Decrypted: www.reva.edu.in


...Program finished with exit code 0
Press ENTER to exit console.
```

7. Write a program for simple RSA algorithm to encrypt and decrypt the data.

Key Generation Algorithm

1. Generate two large random primes, p and q, of approximately equal size such that their product n = p*q
2. Compute n = p*q and Euler's totient function (ϕ) phi(n) = (p-1)(q-1).
3. Choose an integer e, 1 < e < phi, such that gcd(e, phi) = 1.
4. Compute the secret exponent d, 1 < d < phi, such that e*d ≡ 1 (mod phi).
5. The public key is (e, n) and the private key is (d, n). The values of p, q, and phi should also be kept secret.

Encryption

Sender A does the following:-

1. Using the public key (e,n)
2. Represents the plaintext message as a positive integer M
3. Computes the cipher text C = M^e mod n. 4. Sends the cipher text C to B (Receiver).

Decryption

Recipient B does the following:-

1. Uses his private key (d, n) to compute M = C^d mod n.
2. Extracts the plaintext from the integer representative m.

<u>Source Code:</u>

```
import java.util.*;
import java.io.*;
class rsa
{
	static int mult(int x,int y,int n)
	{
		int k=1;
		int j;
		for (j=1; j<=y; j++)
			k = (k * x) % n;
		return ( int) k;
	}
	public static void main (String arg[])throws Exception
	{
		Scanner s=new Scanner(System.in);
		InputStreamReader r=new InputStreamReader(System.in);
		BufferedReader br=new BufferedReader(r);
		String msg1;
		int pt[]=new int[100];
		int ct[]=new int[100];
```

```
		int a,b, n, d, e,Z, p, q, i,temp,et;
		System.out.println("Enter prime No.s p,q :");
		p=s.nextInt();
		q=s.nextInt();
		n = p*q;
		Z=(p-1)*(q-1);
		System.out.println("\nSelect e value:");
		e=s.nextInt();
		System.out.printf("Enter message : ");
		msg1=br.readLine();
		char msg[]=msg1.toCharArray();
		for(i=0;i<msg.length;i++)
			pt[i]=msg[i];
		for(d=1;d<Z;++d)
			if(((e*d)%Z)==1) break;
				System.out.println("p="+""+p+"\tq="+q+"\tn="+n+"\tz="+Z+"\te="+
			e+"\td="+d);
				System.out.println("\nCipher Text = ");
		for(i=0; i<msg.length; i++)
			ct[i] = mult(pt[i], e,n);
		for(i=0; i<msg.length; i++)
			System.out.print("\t"+ct[i]);
		System.out.println("\nPlain Text = ");
		for(i=0; i<msg.length; i++)
			pt[i] = mult(ct[i], d,n) ;
		for(i=0; i<msg.length; i++)
			System.out.print((char)pt[i]);
	}
}
```

Output:

```
lab3-20@lab320-Veriton-Series: ~/CN
lab3-20@lab320-Veriton-Series:~/CN$ javac rsa.java
Picked up JAVA_TOOL_OPTIONS: -javaagent:/usr/share/java/jayatanaag.jar
lab3-20@lab320-Veriton-Series:~/CN$ java rsa
Picked up JAVA_TOOL_OPTIONS: -javaagent:/usr/share/java/jayatanaag.jar
Enter prime No.s p,q :
13
11

Select e value:
7
Enter message : Computer Networks Laboratory
p=13    q=11    n=143   z=120   e=7     d=103

Cipher Text =
        89      45      21      18      39      129     62      49      98      78      62
129     37      45      49      68      80      98      54      59      32      45      49
59      129     45      49      121
Plain Text =
Computer Networks Laboratory
lab3-20@lab320-Veriton-Series:~/CN$
```

8. Setup a honey pot and monitor the honeypot on network (KF Sensor).

HONEY POT:

A honeypot is a computer system that is set up to act as a decoy to lure cyber attackers, and to detect, deflect or study attempts to gain unauthorized access to information systems. Generally, it consists of a computer, applications, and data that simulate the behavior of a real system that appears to be part of a network but is actually isolated and closely monitored. All communications with a honeypot are considered hostile, as there's no reason for legitimate users to access a honeypot. Viewing and logging this activity can provide an insight into the level and types of threat a network infrastructure faces while distracting attackers away from assets of real value. Honeypots can be classified based on their deployment (use/action) and based on their level of involvement.

Based on deployment, honeypots may be classified as:

1. Production honeypots
2. Research honeypots

Production honeypots are easy to use, capture only limited information, and are used primarily by companies or corporations. Production honeypots are placed inside the production network with other production servers by an organization to improve their overall state of security. Normally, production honeypots are low-interaction honeypots, which are easier to deploy. They give less information about the attacks or attackers than research honeypots.

Research honeypots are run to gather information about the motives and tactics of the Black hat community targeting different networks. These honeypots do not add direct value to a specific organization; instead, they are used to research the threats that organizations face and to learn how to better protect against those threats.

KF SENSOR:

KFSensor is a Windows based honeypot Intrusion Detection System (IDS). It acts as a honeypot to attract and detect hackers and worms by simulating vulnerable system services and trojans. By acting as a decoy server, it can divert attacks from critical systems and provide a higher level of information than can be achieved by using firewalls and NIDS alone. KFSensor is a system installed in a network in order to divert and study an attacker's behavior. This is a new technique that is very effective in detecting attacks.

The main feature of KFSensor is that every connection it receives is a suspect hence it results in very few false alerts. At the heart of KFSensor sits a powerful internet daemon service that is built to handle multiple ports and IP addresses. It is written to resist denial of service and buffer overflow attacks.

Building on this flexibility KFSensor can respond to connections in a variety of ways, from simple port listening and basic services (such as echo), to complex simulations of standard system services.

For the HTTP protocol KFSensor accurately simulates the way Microsoft's web server (IIS) responds to both valid and invalid requests. As well as being able to host a website it also handles complexities such as range requests and client-side cache negotiations. This makes it extremely difficult for an attacker to fingerprint, or identify KFSensor as a honeypot.

PROCEDURE:

STEP-1: Download KF Sensor Evaluation Setup File from KF Sensor Website.

STEP-2: Install with License Agreement and appropriate directory path.

STEP-3: Reboot the Computer now. The KF Sensor automatically starts during windows boot.

STEP-4: Click Next to setup wizard.

STEP-5: Select all port classes to include and Click Next.

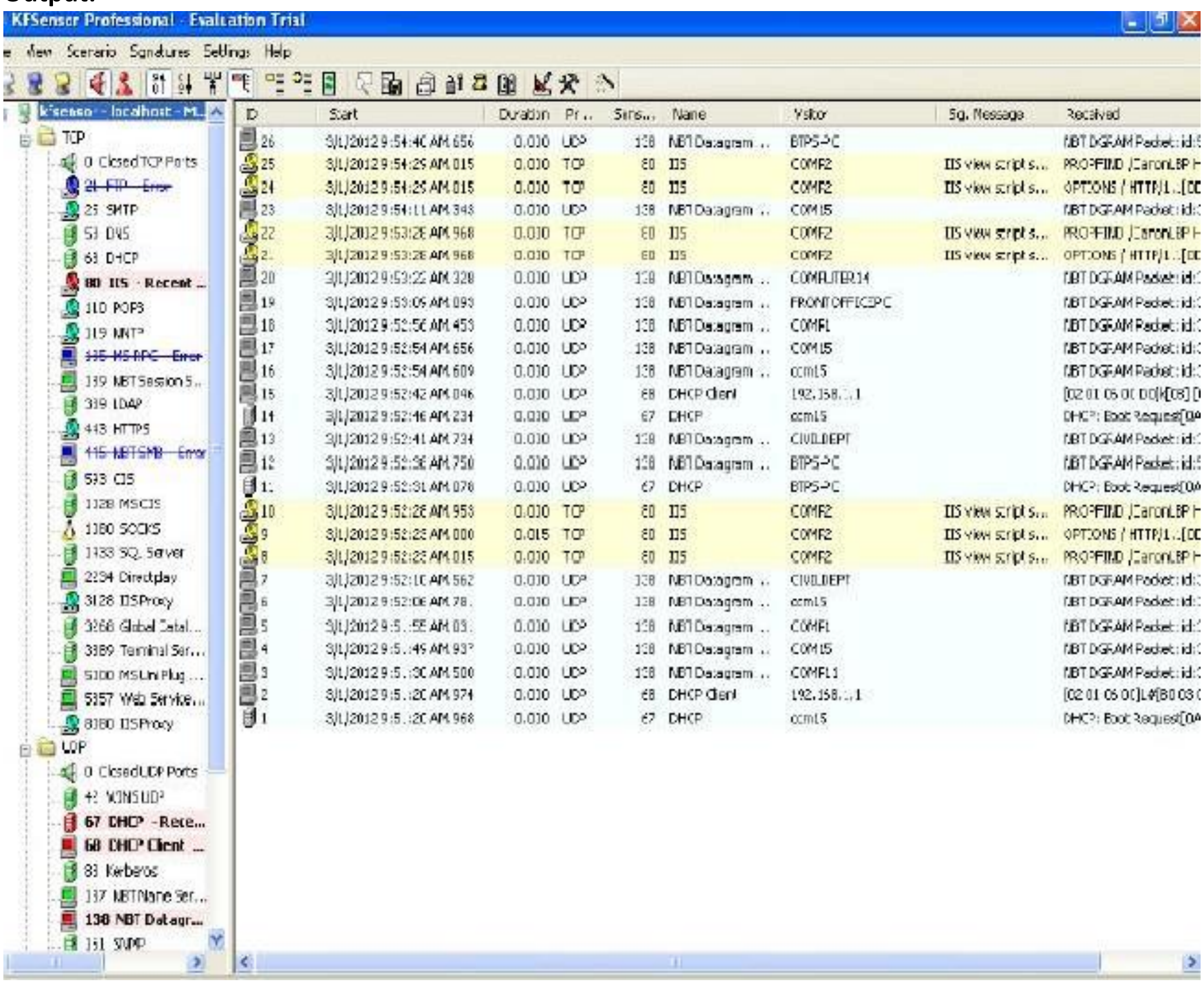
STEP-6: “Send the email and Send from email”, enter the ID and Click Next.

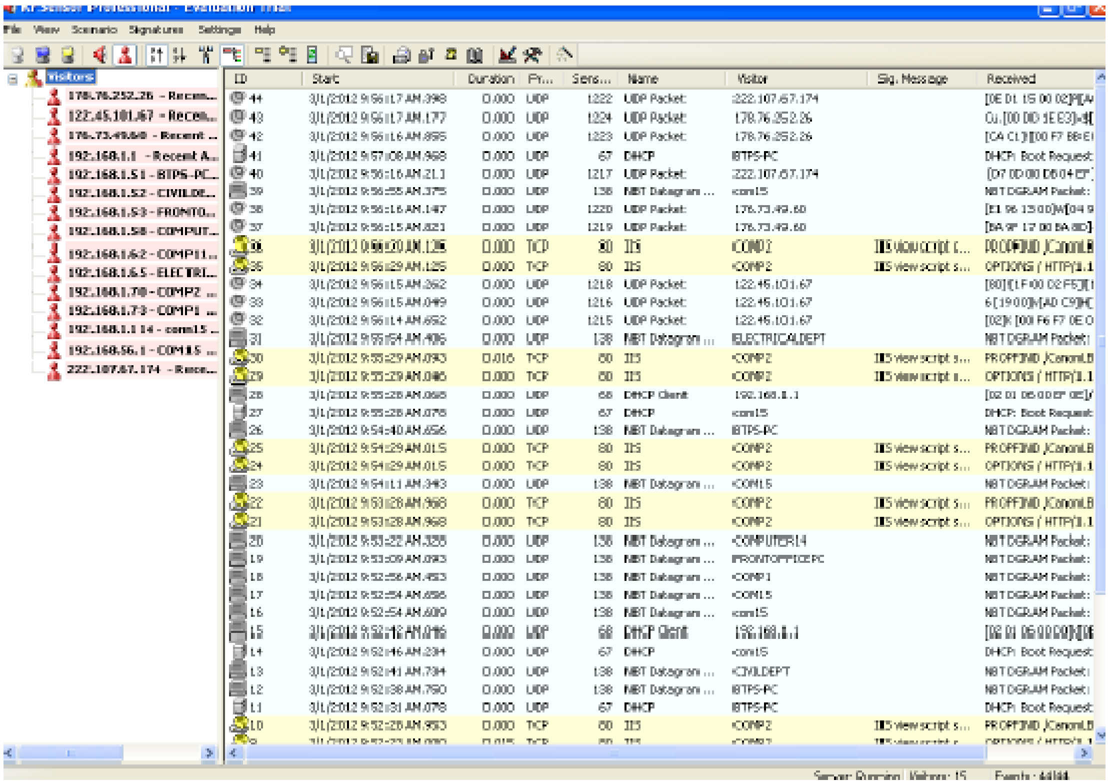
STEP-7: Select the options such as Denial of Service[DOS], Port Activity, Proxy Emulsion, Network Port
Analyzer, Click Next.

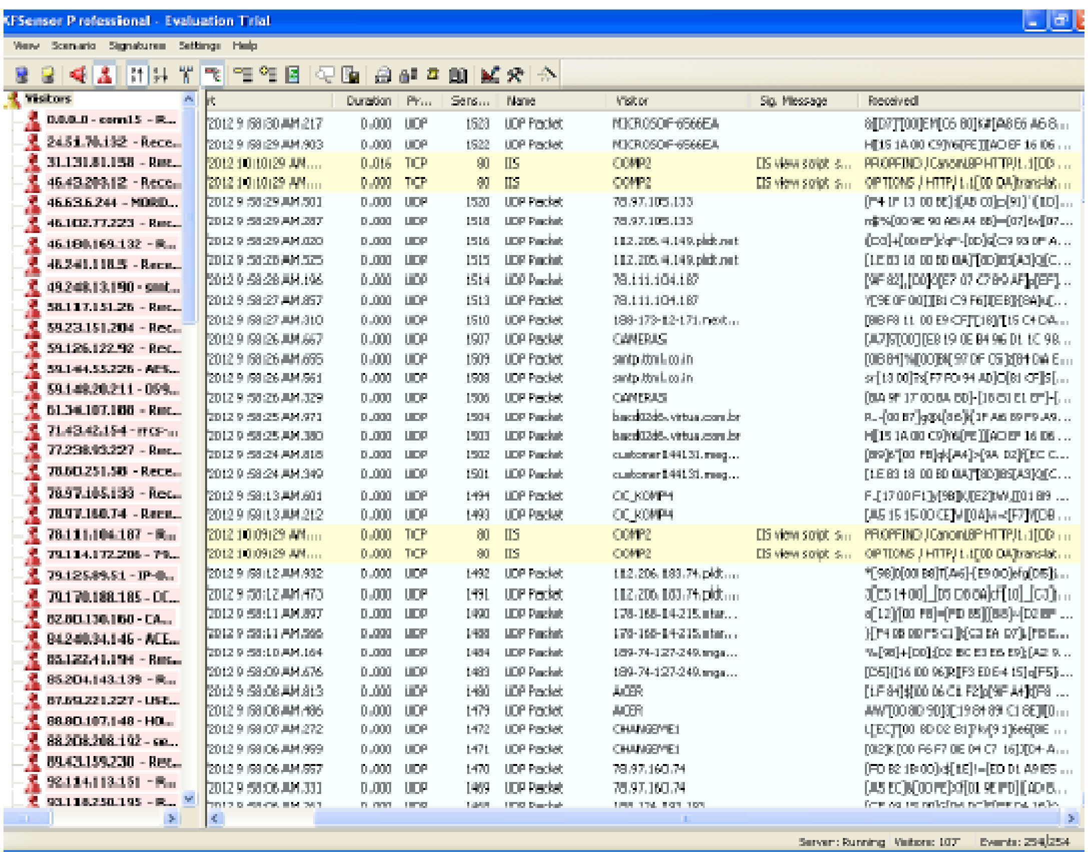
STEP-8: Select Install as System service and Click Next.

STEP-9: Click finish.

Output:




KFSensor Professional - Evaluation Trial
File View Scenario Signatures Settings Help
Visitors
ID Start Duration Pr... Sens... Name Visitor Sig. Message Received
Server: Running Visitors: 15 Events: 44144


KFSensor Professional - Evaluation Trial
View Scenario Signatures Settings Help
Visitors
Duration Pr... Sens... Name Visitor Sig. Message Received
Server: Running Visitors: 107 Events: 254|254

Results:

Thus, the study of setup a hotspot and monitor the hotspot on network has been developed successfully.