Vidya Vikas Education Trust's

Universal College of Engineering

(Permanently unaided | Approved by AICTE, DTE & Affiliated to University of Mumbai)

Department of Computer Engineering

Vision:

To be recognized globally as a department provides quality technical education that eventually caters to helping and serving the community.

Mission:

To develop human resources with sound knowledge in theory and practice of computer science and engineering. To motivate the students to solve real-world problems to help the society grow. To provide a learning ambience to enhance innovations, team spirit and leadership qualities for students.

LAB MANUAL

(CSL 602)

Cryptography & System Security (CSS)

Name: **VIVEK.SHIVAKUMAR. HOTTI**

Class: **T. E-Computer Engineering**

Semester: **VI**

Roll No.: **37**

Batch: **A2**

Faculty Name: **Mr. Chinmay Raut**

2021 – 2022

Lab Code	Lab Name	Credits
CSL602	**Computer Network Lab**	**1**

Lab Objectives & Outcomes:

Prerequisite: Computer Network	
Lab Objectives:	
1	To apply various encryption techniques
2	To study and implement various security mechanism
3	To explore the network security concept and tools
Lab Outcomes: At the end of the course, the students will be able to	
1	apply the knowledge of symmetric and asymmetric cryptography to implement simple
2	explore the different network reconnaissance tools to gather information about networks.
3	explore and use tools like sniffers, port scanners and other related tools for analysing
4	set up firewalls and intrusion detection systems using open-source technologies and to
5	explore various attacks like buffer-overflow and web application attack.

Suggested List of experiments:

Suggested List of Experiments	
Sr. No.	**Title of Experiment**
1.	Java program for Caeser Cipher
2.	Implementation and analysis of RSA crypto system.
3.	Implementation of Diffie Hellman Key exchange algorithm.
4.	Java program for DES
5.	Study of packet sniffer tools: wireshark,: 1. Download and install wireshark and capture icmp, tcp, and http packets in promiscuous mode. 2. Explore how the packets can be traced based on different filters.
6.	Case study on SQL injection attack,

Description:

Design and implementation of any case study/ applications /experiments / mini project based on departmental level courses using modern tools.

Term work:

The distribution of marks for term work shall be as follows:

Lab Performance 15 Marks

Assignments 05 Marks

Attendance (Theory & practical) 05 Marks

Vidya Vikas Education Trust's

Universal College of Engineering

(Permanently unaided | Approved by AICTE, DTE & Affiliated to University of Mumbai)

INDEX

Name: Vivek. Hotti **Roll:** 37
Class: TE- Computers **Division:** A
Semester: 6

EXPERIMENT – 01

AIM: Java Program for Caeser Cipher.

THEORY:

Caesar Ciphers:

- Caeser Cipher is simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter some fixed number of positions down the alphabet. For example with a shift of 1, A would be replaced by B, B would become C, and so on.

- Example: Text : ABCDEFGHIJKLMNOPQRSTUVWXYZ
  Shift: 23
  Cipher: XYZABCDEFGHIJKLMNOPQRSTUVW

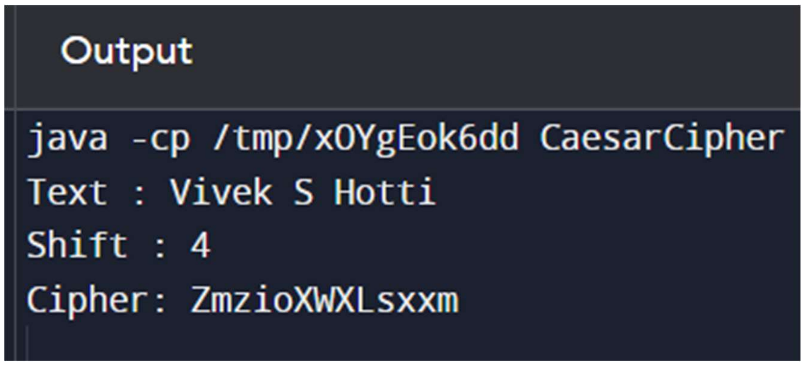
- Algorithm for Caesar Cipher:
  Input:
  1. A String of lower case letters, called Text.
  2. An Integer between 0-25 denoting the required shift. Procedure:
  • Traverse the given text one character at a time .
  • For each character, transform the given character as per the rule, depending on whether we're encrypting or
  decrypting the text.
  • Return the new string generated.

INPUT Program:

```
class CaesarCipher{
// Encrypts text using a shift od s
public static StringBuffer encrypt(String text, int s){
StringBuffer result= new StringBuffer();
for (int i=0; i<text.length(); i++){
if (Character.isUpperCase(text.charAt(i))){
char ch = (char)(((int)text.charAt(i) + s - 65) % 26 + 65);
result.append(ch);
}else{
```

```
char ch = (char)(((int)text.charAt(i) + s - 97) % 26 + 97);
result.append(ch);}}
return result;
}
public static void main(String[] args){
String text = "Vivek S Hotti";
int s = 4;
System.out.println("Text : " + text);
System.out.println("Shift : " + s);
System.out.println("Cipher: " + encrypt(text, s));
}
}
```

OUTPUT:



```
Output

java -cp /tmp/xOYgEok6dd CaesarCipher
Text : Vivek S Hotti
Shift : 4
Cipher: ZmzioXWXLsxxm
```

Conclusion:

Hence, we have implemented java Program for Caesar Cipher.

x-x-x
Experiment 01 Over

EXPERIMENT – 02

AIM: Implementation and analysis of RSA crypto system.

THEORY:

RSA or Rivest–Shamir–Adleman is an algorithm employed by modern computers to encrypt and decrypt messages. It is an asymmetric cryptographic algorithm.

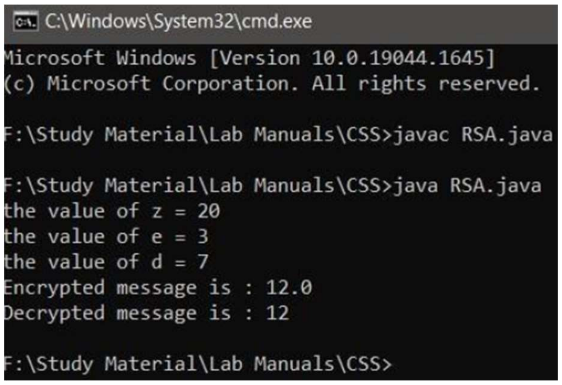
Asymmetric means that there are two different keys. This is also called public key cryptography because one among the keys are often given to anyone. The other is the private key which is kept private. The algorithm is predicated on the very fact that finding the factors of an outsized number is difficult: when the factors are prime numbers, the matter is named prime factorization. It is also a key pair (public and personal key) generator.

INPUT PROGRAM

```
import java.math.*;
import java.util.*;
class Main {
public static void main(String args[]){
int p, q, n, z, d = 0, e, i;
// The number to be encrypted and decrypted
int msg = 12;
double c;
BigInteger msgback;
// 1st prime number p
p = 3;
// 2nd prime number q
q = 11;
n = p * q;
z = (p - 1) * (q - 1);
System.out.println("the value of z = " + z);
for (e = 2; e < z; e++) {
// e is for public key exponent
if (gcd(e, z) == 1) {
break;
}}
System.out.println("the value of e = " + e);
for (i = 0; i <= 9; i++) {
```

```
int x = 1 + (i * z);
// d is for private key exponent
if (x % e == 0) {
d = x / e;
break;
}}
System.out.println("the value of d = " + d);
c = (Math.pow(msg, e)) % n;
System.out.println("Encrypted message is : " + c);
// converting int value of n to BigInteger
BigInteger N = BigInteger.valueOf(n);
// converting float value of c to BigInteger
BigInteger C = BigDecimal.valueOf(c).toBigInteger();
msgback = (C.pow(d)).mod(N);
System.out.println("Decrypted message is : " + msgback);
}
static int gcd(int e, int z){
if (e == 0)
return z;
else
return gcd(z % e, e);
}}
```

RECEIVED OUTPUT:



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

F:\Study Material\Lab Manuals\CSS>javac RSA.java

F:\Study Material\Lab Manuals\CSS>java RSA.java
the value of z = 20
the value of e = 3
the value of d = 7
Encrypted message is : 12.0
Decrypted message is : 12

F:\Study Material\Lab Manuals\CSS>
```

CONCLUSION:

Hence, we have implemented java Program for Rivest Shamir Adleman (RSA) Algorithm.

x-x-x

Experiment 02 Over

EXPERIMENT – 03

AIM: Implementation of Diffie Hellman Key exchange algorithm.

THEORY:

What is Diffie Hellman Key ExchangeAlgorithm ? :

- Diffie Hellman key exchange is a cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel.
- This key can then be used to encrypt subsequent communications using a symmetric key cipher.
- The Diffie–Hellman key exchange algorithm solves the following dilemma. Alice and Bob want to share a secret key for use in a symmetric cipher, but their only means of communication is insecure.
- Every piece of information that they exchange is observed by their adversary Eve. How is it possible for Alice and Bob to share a key without making it available to Eve? At first glance it appears that Alice and Bob face an impossible task. It was a brilliant insight of Diffie and Hellman that the difficulty of the discrete logarithm problem for F*p provides a possible solution.
- The simplest, and original, implementation of the protocol uses the Multiplicative group of integers modulo p, where p is prime and g is primitive root mod p.
- In the original description, the Diffie Hellman exchange by itself does not provide authentication of the communicating parties and is thus vulnerable to a man in the middle attack. A person in the middle may establish two distinct Diffie Hellman key exchanges, one with Alice and the other with Bob, effectively masquerading as Alice to Bob, and vice versa, allowing the attacker to decrypt (and read or store) then re encrypt the messages passed between them. A method to authenticate the

communicating parties to each other is generally needed to prevent this type of attack.

Algorithm ?:

▪ Alice and Bob, two users who wish to establish secure communications.
▪ We can assume that Alice and Bob know nothing about each other but are in contact.
▪ 1. Communicating in the clear, Alice and Bob agree on two large positive integers, p and g, where p is a prime number and g is a primitive root mod p.
▪ 2. Alice randomly chooses another large positive integer, XA, which is smaller than p. XA will serve as Alice's private key.
▪ 3. Bob similarly chooses his own private key, XB.
▪ 4. Alice computes her public key, YA, using the formula YA = (g^XA) mod p.
▪ 5. Bob similarly computes his public key, YB, using the formula YB = (g^XB) mod p.
▪ 6. Alice and Bob exchange public keys over the insecure circuit.
▪ 7. Alice computes the shared secret key, k, using the formula k = (YB^XA) mod p.
▪ 8. Bob computes the same shared secret key, k, using the formula k = (YA^XB) mod p.
▪ 9. Alice and Bob communicate using the symmetric algorithm of their choice and the shared secret key, k, which was never transmitted over the insecure circuit.

INPUT CODE TO COMPUTE first():

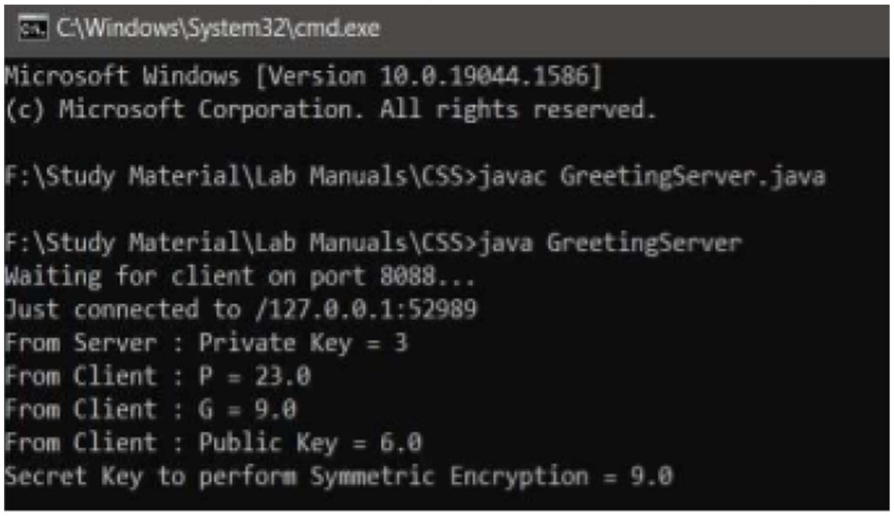
(server side code)

```
import java.net.*;
import java.io.*;
public class GreetingServer { public static void main(String[] args) throws
IOException{
try {
int port = 8088;
// Server Key
int b = 3;
// Client p, g, and key
double clientP, clientG, clientA, B, Bdash;
String Bstr;
// Established the Connection
ServerSocket serverSocket = new ServerSocket(port);
System.out.println("Waiting for client on port " + serverSocket.getLocalPort() +
"...");
Socket server = serverSocket.accept();
System.out.println("Just connected to " + server.getRemoteSocketAddress());
// Server's Private Key
System.out.println("From Server : Private Key = " + b);
// Accepts the data from client
DataInputStream in = new DataInputStream(server.getInputStream());
clientP = Integer.parseInt(in.readUTF()); // to accept p
System.out.println("From Client : P = " + clientP);
clientG = Integer.parseInt(in.readUTF()); // to accept g
System.out.println("From Client : G = " + clientG);
clientA = Double.parseDouble(in.readUTF()); // to accept
System.out.println("From Client : Public Key = " +clientA);
B = ((Math.pow(clientG, b)) % clientP); // calculation of B
Bstr = Double.toString(B);
// Sends data to client
// Value of B
OutputStream outToclient = server.getOutputStream();
DataOutputStream out = new DataOutputStream(outToclient);
out.writeUTF(Bstr); // Sending B
Bdash = ((Math.pow(clientA, b)) % clientP); // calculation
System.out.println("Secret Key to perform Symmetric Encryption = "+ Bdash);
server.close();
}
catch (SocketTimeoutException s) {
System.out.println(s);
} catch (
IOException e) {
System.out.println(e);
}
}}
```

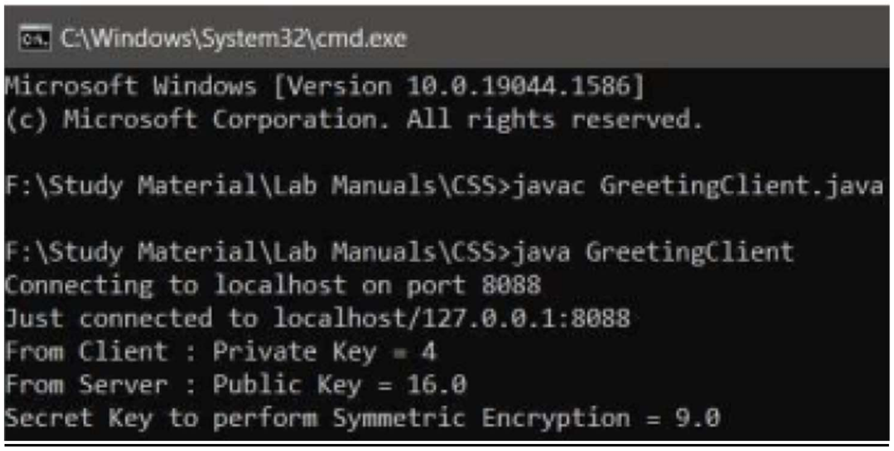
(client side code)

```
import java.net.*;
import java.io.*;
public class GreetingClient { public static
void main(String[] args){
try {
String pstr, gstr, Astr;
String serverName = "localhost";
int port = 8088;
// Declare p, g, and Key of client
int p = 23;
int g = 9;
int a = 4;
double Adash, serverB;
// Established the connection
System.out.println("Connecting to " + serverName + " on port " + port);
Socket client = new Socket(serverName, port);
System.out.println("Just connected to " + client.getRemoteSocketAddress());
// Sends the data to client
OutputStream outToServer = client.getOutputStream();
DataOutputStream out = new DataOutputStream(outToServer);
pstr = Integer.toString(p); out.writeUTF(pstr); // Sending p
gstr = Integer.toString(g); out.writeUTF(gstr); // Sending g
double A = ((Math.pow(g, a)) % p); // calculation of A
Astr = Double.toString(A);
out.writeUTF(Astr); // Sending A
// Client's Private Key
System.out.println("From Client : Private Key = " + a);
// Accepts the data
DataInputStream in = new DataInputStream(client.getInputStream());
serverB = Double.parseDouble(in.readUTF());
System.out.println("From Server : Public Key = " + serverB);
Adash = ((Math.pow(serverB, a)) % p); // calculation of
System.out.println("Secret Key to perform Symmetric Encryption = " + Adash);
client.close();
} catch (
Exception e) {
e.printStackTrace();
}
}}
```

OUTPUT



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.1586]
(c) Microsoft Corporation. All rights reserved.

F:\Study Material\Lab Manuals\CSS>javac GreetingServer.java

F:\Study Material\Lab Manuals\CSS>java GreetingServer
Waiting for client on port 8088...
Just connected to /127.0.0.1:52989
From Server : Private Key = 3
From Client : P = 23.0
From Client : G = 9.0
From Client : Public Key = 6.0
Secret Key to perform Symmetric Encryption = 9.0
```



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.1586]
(c) Microsoft Corporation. All rights reserved.

F:\Study Material\Lab Manuals\CSS>javac GreetingClient.java

F:\Study Material\Lab Manuals\CSS>java GreetingClient
Connecting to localhost on port 8088
Just connected to localhost/127.0.0.1:8088
From Client : Private Key = 4
From Server : Public Key = 16.0
Secret Key to perform Symmetric Encryption = 9.0
```

CONCLUSION:

Hence, we have implemented Diffie Hellman key exchange to ensure secure key exchange between sender and receiver.

x-x-x

Experiment 03 Over

EXPERIMENT – 04

AIM: Java Program for DES.

THEORY:

Data Encryption Standard?

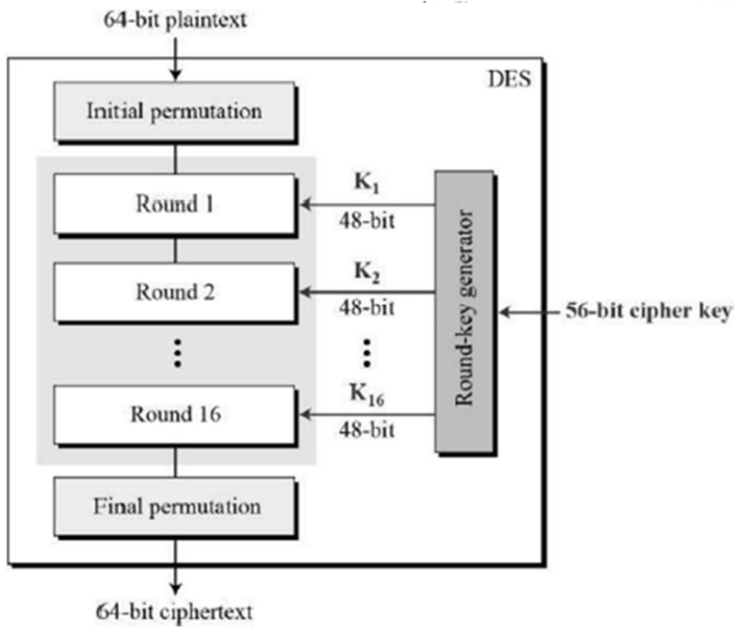
• DES is a symmetric key block cipher published by NIST (National institute of Standards & Technologies)
• DES is an implementation of a Fiestal cipher.
• It has a 64-bit block size, a 64-bit key length & uses 16 rounds.
• DES has an effective key length of 56 bits, since 8 of the 64 bits are not used by the encryption algorithm (they are check bits)
• Since fiestal cipher blueprint is used for DES, all that is required is to fully specify DES is – o. The round function: which is based on taking groups of input bits & replacing (substituting) them according to some rules based on tables known as S-boxes.
The key structure: which identifies which bits of the key are used to form the subkeys for any given round. o Any additional processing steps: for eg. DES conducts an initial permutation of all i/p bits before the first encryption round begins & then performs the inverse of this permutation to all the Output bits immediately after the last encryption round has been completed.

The DES consists of:

- General Structure of DES
- Initial and final Permutations
- Rounds
- DES Function
- Key generation

1.General Structure of DES

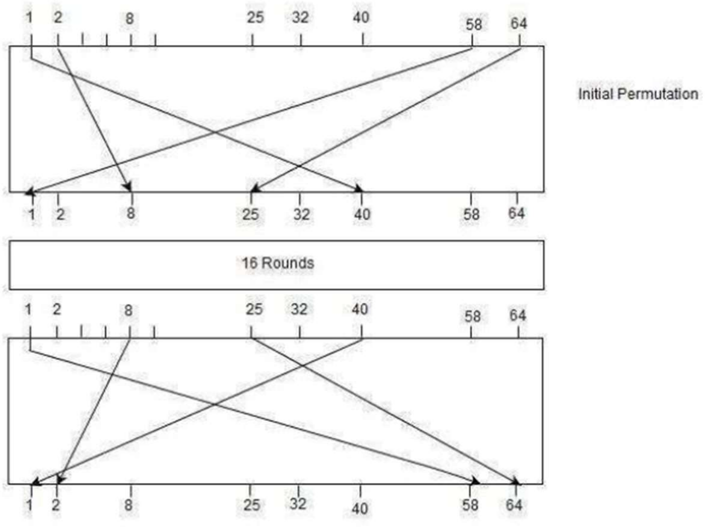
□ The encryption process is made of two permutations (P-boxes), which are called Initial and final permutations and six Fiestal rounds. Each round uses a different 48-bit round key generated from the cipher key according to a predefined algorithm. The figure below the elements of DES cipher at the encryption site.



2. Initial and Final permutation:

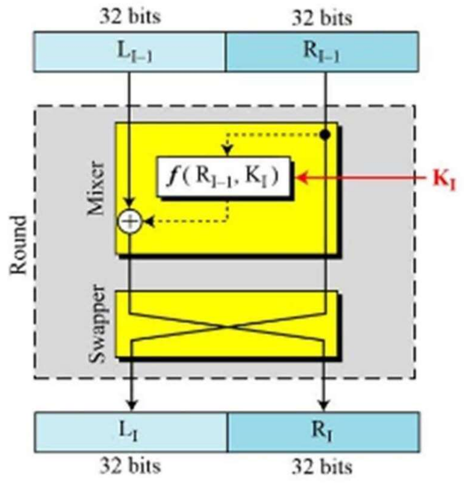
• These are straight permutation boxes (p-boxes) that are inverse of each other. They have no cryptography significance in DES. The initial & final permutation is as shown below.

• Each of these permutations takes a 64-bit i/p and permutes them according to a predefined rule.



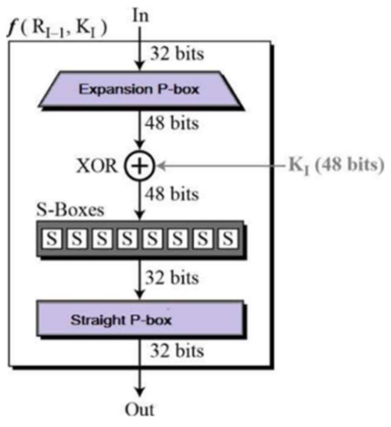
3. Rounds:

• The DES uses 16 rounds.
• Each round of DES is a Feistal Cipher
• The round takes LI−1LI−1 and RI−1RI−1 from the previous round(or the initial permutation box if that is the first round) and creates LILI and RIRI, which go to next round(or final permutation box if that is the last round). Each round has two cipher elements (mixer and swapper). Each of these elements is invertible. The swapper swaps the left half of the text with the right half. The mixer is invertible because of the XOR operation.



4.DES function

DES function is the most important part of DES. It applies a 48-bit key to the rightmost 32
bits (RI−1RI−1) to produce a 32-bit Output.
This function is comprised of four parts as shown in the figure below
• An expansion box
• An XOR function
• A group of S-boxes
• A straight D-box

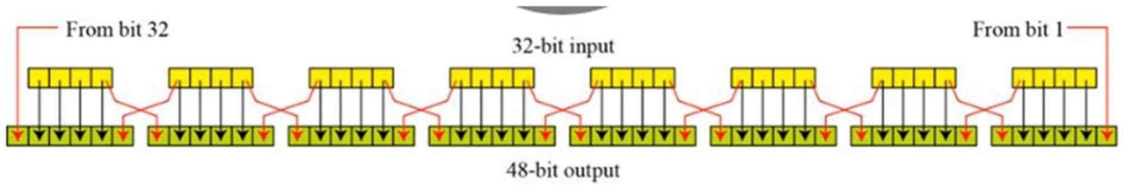


Expansion D-box:

• As i/p RI−1RI−1 is 32 bit & round key is of 48-bit, we need to expand the right input to 48 bits.

• RI−1RI−1 is divided into 8 4-bit sections.

• Each 4-bit section is expanded to 6-bit. It follows the predetermined rules. For each section, input bits 1,2,3 and 4 are copied to Output bits 2,3,4 and 5. Output bit 1 comes from the 4th bit of the previous section, Output bit 6 comes from the 1st bit of the next section. If sections 1 and 8 can be considered adjacent to each other the same rules can be applied to bits 1 and bit 32 as shown in fig below.

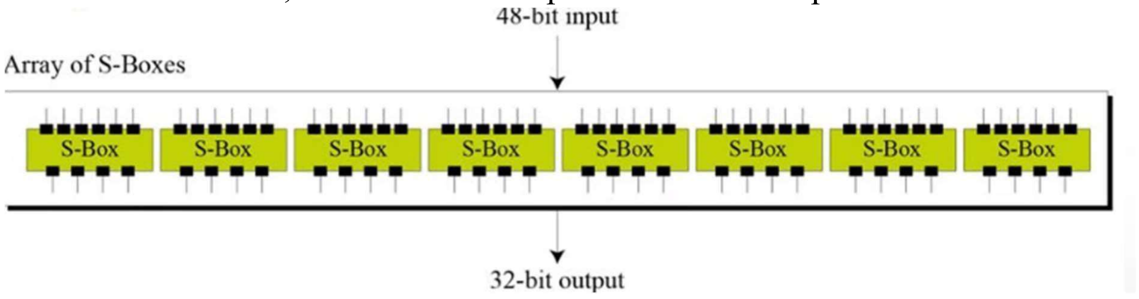


XOR Operation:

After expansion permutation, DES does XOR operation on the expanded right section & the
round key. The round key is used only in this operation. Substitution boxes:

• The S-boxes carry out the real mixing

• DES uses S-boxes, each with 6-bit input and a 4-bit Output.



There are total 8 S-box tables. The o/p of all eight S-boxes is then combined in to a 32-bit section.

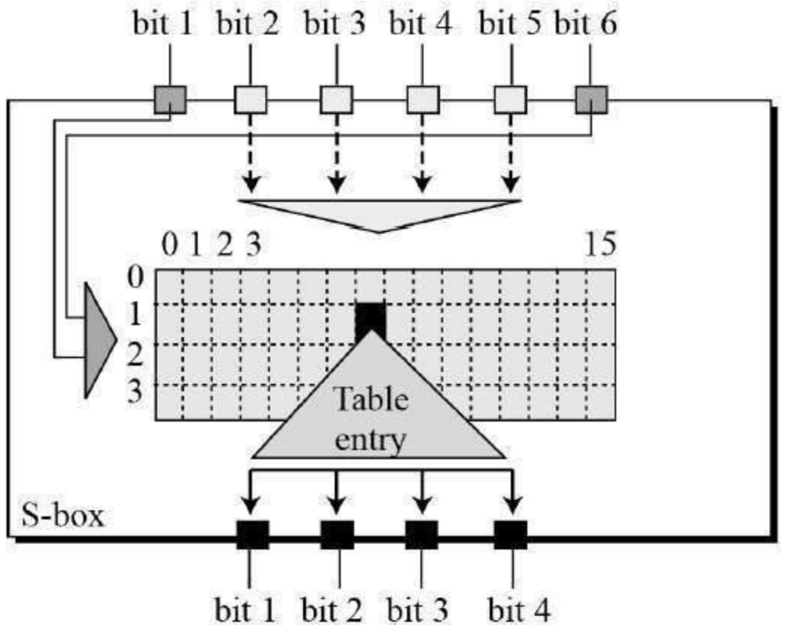
The 48-bit data from the XOR operation is divided into 8 6-bit sections and each section is given to one S-box.

The result of each S-box is 4-bit section when these results are combined final 32-bit text is obtained.

The substitution in each box follows the predetermined rule based on a 4-row and 16column table.

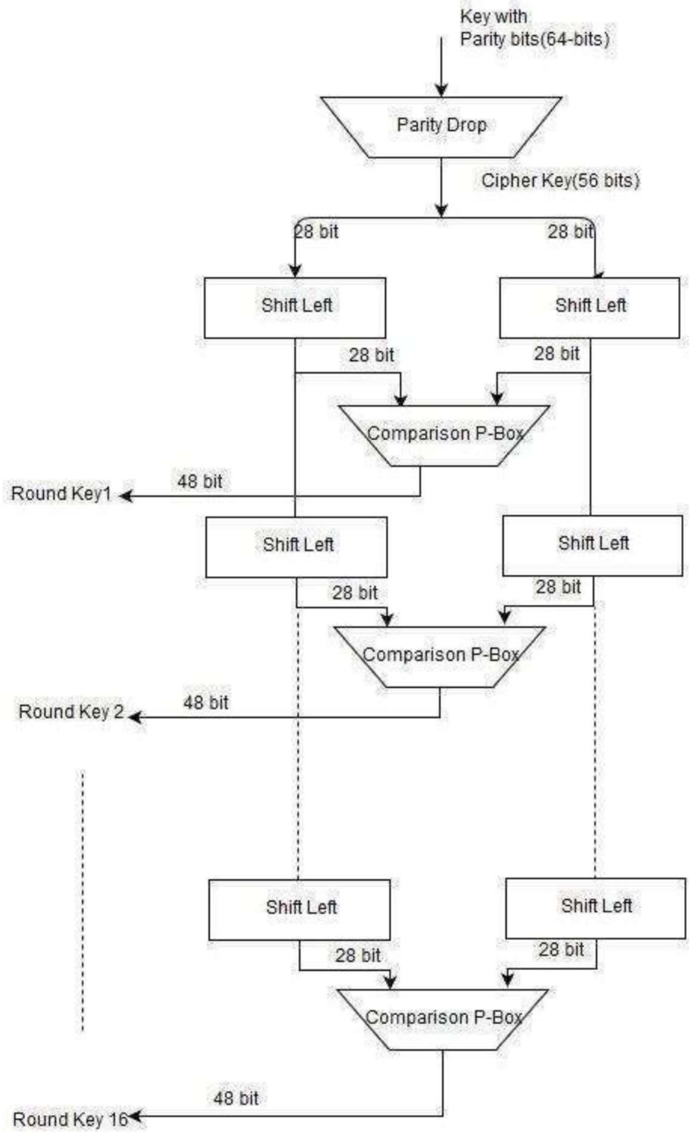
The combination of bits 1 and 6 of the input defines one of the four rows, the combination of bits 2,3,4 and 5 defines one of the 16 columns as shown in the figure below.

Each S-box has its own table, so total of 8 tables for 8 S-boxes. The values in these tables are stored in decimal numbers.



example:
Input to the S- box1 is 100011
1st and 6th bit are combined together which forms "11" which in 4 bit binary is 3 in decimal, so we look for the value in row 3 of S-Box 1 table Now take the remaining bits "0001" whose equivalent binary is 1 in decimal, so we look for the value in column 1 of S-Box 1 table.
So, then the result is again a decimal number calculate its binary. Suppose the S-box 1 table has entry 12 in decimal in position (3,1) so Output will be 4-bit "1100" for 6-bit "100011".
Straight D box: The 32-bit o/p of the S-boxes then subjected to the straight permutation which transposes the bits.

5. Key Generation:
The round key generation creates 48-bit keys out of 56-bit cipher Key. Normally given as a 64-bit key in which 8 extra bits are the parity bits,which are dropped before the actual key-generation process. Process of key generation is as follows

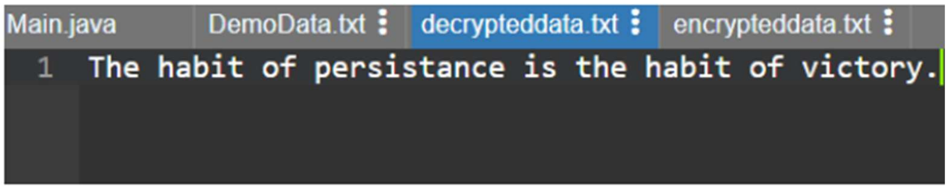


Parity Drop: It drops the parity bits (8,16,24,32,40,48,56,64) from the 64-bit key and permutes the rest of the bits according to the predefined table. Shift Left:
• After the straight permutation, the key is divided into 2 28-bit parts
each part is shifted left (circular shift)one or two bits.
• In round 1,2,9,16 shifting is one bit and in other rounds, it is 2 bits
The two parts are then combined to form a 56- bit part.
Compression D Box: This box changes the 56-bits to 48 bits, which are used as a key for a round.

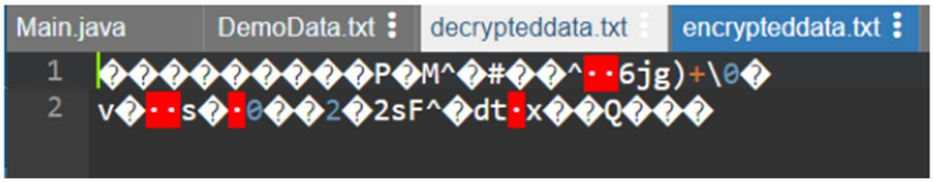
INPUT to eliminate left recursion:

```
//Java classes that are mandatory to import for encryption and decryption process
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.AlgorithmParameterSpec;
import javax.crypto.Cipher;
import javax.crypto.CipherInputStream;
import javax.crypto.CipherOutputStream;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
public class Main{
//creating an instance of the Cipher class for encryption
private static Cipher encrypt;
//creating an instance of the Cipher class for decryption
private static Cipher decrypt;
//initializing vector
private static final byte[] initialization_vector =
{ 22, 33, 11, 44, 55, 99, 66, 77 };
//main() method
public static void main (String[]args){
//path of the file that we want to encrypt
String textFile = "DemoData.txt";
//path of the encrypted file that we get as Output
String encryptedData = "encrypteddata.txt";
String decryptedData = "decrypteddata.txt";
try{
//generating keys by using the KeyGenerator class
SecretKey scrtkey = KeyGenerator.getInstance ("DES").generateKey ();
AlgorithmParameterSpec aps = new IvParameterSpec (initialization_vector);
//setting encryption mode
encrypt = Cipher.getInstance ("DES/CBC/PKCS5Padding");
encrypt.init (Cipher.ENCRYPT_MODE, scrtkey, aps);
//setting decryption mode
decrypt = Cipher.getInstance ("DES/CBC/PKCS5Padding");
decrypt.init (Cipher.DECRYPT_MODE, scrtkey, aps);
//calling encrypt() method to encrypt the file
encryption (new FileInputStream (textFile),
new FileOutputStream (encryptedData));
//calling decrypt() method to decrypt the file
decryption (new FileInputStream (encryptedData),
new FileOutputStream (decryptedData));
//prints the stetment if the Program runs successfully
System.out.println("The encrypted and decrypted files have been created
successfully.");
} //catching multiple exceptions by using the |
(
or) operator in a single catch block
catch (NoSuchAlgorithmException | NoSuchPaddingException |
InvalidKeyException |
InvalidAlgorithmParameterException | IOException e){
//prints the message (if any) related to exceptions
e.printStackTrace ();
}
} //method for encryption
private static void encryption(InputStream input, OutputStream Output) throws
IOException
{
Output = new CipherOutputStream (Output, encrypt);
//calling the writeBytes() method to write the encrypted bytes to the file
writeBytes (input, Output);
} //method for decryption
private static void decryption (InputStream input, OutputStream Output) throws
IOException{
input = new CipherInputStream (input, decrypt);
//calling the writeBytes() method to write the decrypted bytes to the file
writeBytes (input, Output);
}
//method for writting bytes to the files
private static void writeBytes (InputStream input, OutputStream Output) throws
IOException{
byte[]writeBuffer = new byte[512];
int readBytes = 0;
while ((readBytes = input.read (writeBuffer)) >= 0){
Output.write (writeBuffer, 0, readBytes);
} //closing the Output stream
Output.close ();
//closing the input stream
input.close ();
}
}
```

OUPUT To Above Program



```
The habit of persistance is the habit of victory.
```



```
����������P�M^�#��^··6jg)+\0�
v�··s�·0���2�2sF^�dt·x��Q���
```

CONCLUSION:

Hence, we have written a program for DES.

x-x-x

Experiment 04 Over

EXPERIMENT – 05

AIM: Study of packet sniffer tools:Wireshark.

THEORY:

Wire shark is a network packet analyzer. A network packet analyzer presents captured packet data in as much detail as possible.

You could think of a network packet analyzer as a measuring device for examining what's happening inside a network cable, just like an electrician uses a voltmeter for examining what's happening inside an electric cable (but at a higher level, of course).

In the past, such tools were either very expensive, proprietary, or both. However, with the advent of Wireshark, that has changed. Wireshark is available for free, is open source, and is one of the best packet analysers available today.

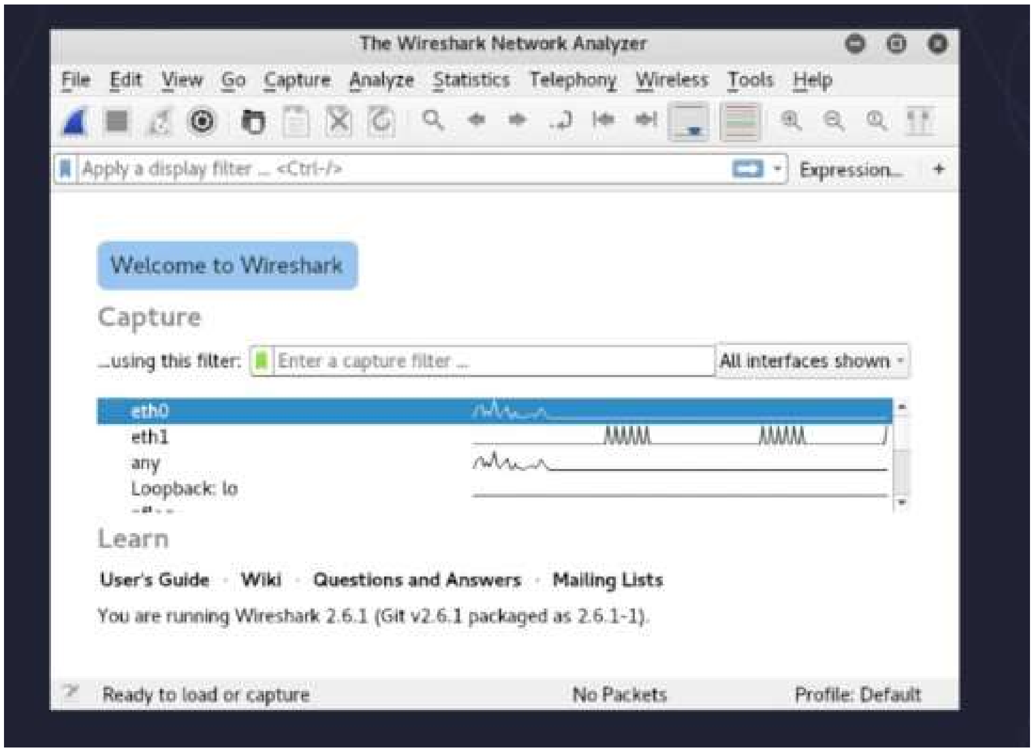
Here are some reasons people use Wire shark:
• Network administrators use it to troubleshoot network problems
• Network security engineers use it to examine security problems
• QA engineers use it to verify network applications
• Developers use it to debug protocol implementations
• People use it to learn network protocol

Internals Features:
The following are some of the many features Wire shark provides:
• Available for UNIX and Windows.
• Capture live packet data from a network interface.
• Open files containing packet data captured with tcpdump/WinDump, Wireshark, and many other packet capture Programs.
• Import packets from text files containing hex dumps of packet data.

• Display packets with very detailed protocol information.
• Save packet data captured.
• Export some or all packets in a number of capture file formats.
• Filter packets on many criteria.
• Search for packets on many criteria.
• Colorize packet display based on filters.
• Create various statistics.

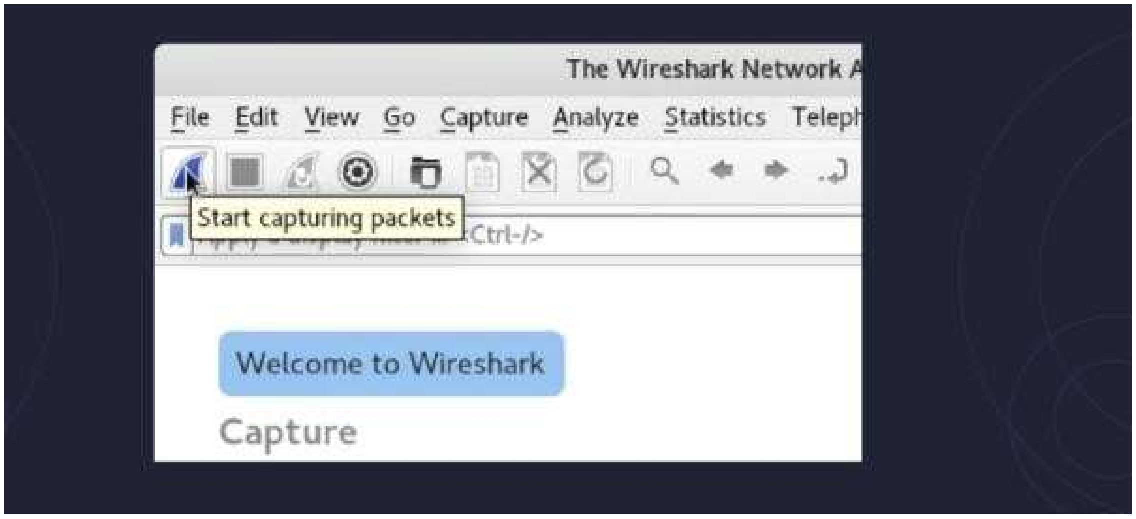
Data Packets on Wireshark:

Now that we have Wireshark installed let's go over how to enable the Wireshark packet sniffer and then analyse the network traffic. Capturing Data Packets on Wireshark.
When you open Wireshark, you see a screen that shows you a list of all of the network connections you can monitor. You also have a capture filter field, so you only capture the network traffic you want to see.

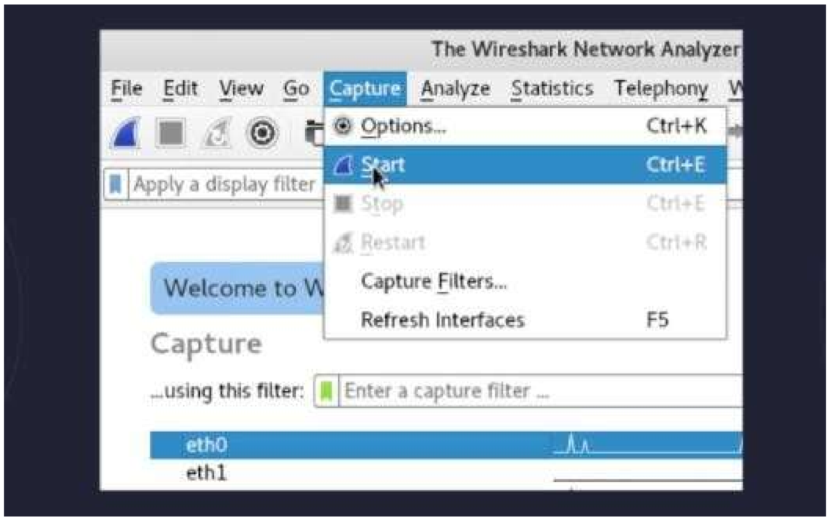

The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
eth0
eth1
any
Loopback: lo
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists
You are running Wireshark 2.6.1 (Git v2.6.1 packaged as 2.6.1-1).
Ready to load or capture
No Packets
Profile: Default

You can select one or more of the network interfaces using "shift left-click." Once you have the network interface selected, you can start the capture, and there are several ways to do that.

Click the first button on the toolbar, titled "Start Capturing Packets."

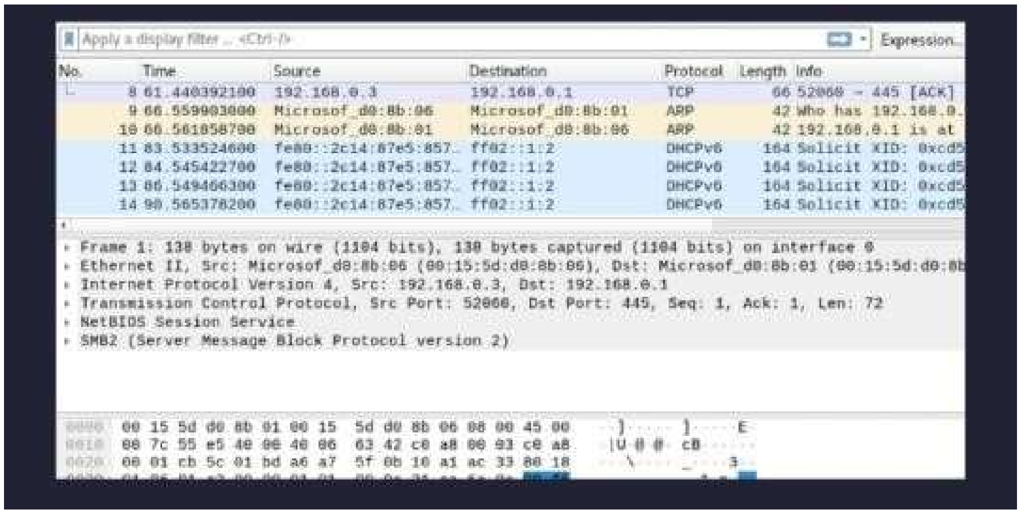


You can select the menu item Capture -> Start.



Or you could use the keystroke Control – E.

During the capture, Wire shark will show you the packets that it captures in real-time.



```
No.   Time          Source                     Destination          Protocol Length Info
    8 61.440392100  192.168.0.3                192.168.0.1          TCP          66 52068 → 445 [ACK]
    9 66.559903800  Microsof_d0:8b:06          Microsof_d0:8b:01    ARP          42 Who has 192.168.0.
   10 66.561858700  Microsof_d0:8b:01          Microsof_d0:8b:06    ARP          42 192.168.0.1 is at
   11 83.533524600  fe80::2c14:87e5:857…       ff02::1:2            DHCPv6      164 Solicit XID: 0xcd5
   12 84.545422700  fe80::2c14:87e5:857…       ff02::1:2            DHCPv6      164 Solicit XID: 0xcd5
   13 86.549466300  fe80::2c14:87e5:857…       ff02::1:2            DHCPv6      164 Solicit XID: 0xcd5
   14 90.565378200  fe80::2c14:87e5:857…       ff02::1:2            DHCPv6      164 Solicit XID: 0xcd5

▸ Frame 1: 138 bytes on wire (1104 bits), 138 bytes captured (1104 bits) on interface 0
▸ Ethernet II, Src: Microsof_d0:8b:06 (00:15:5d:d0:8b:06), Dst: Microsof_d0:8b:01 (00:15:5d:d0:8b
▸ Internet Protocol Version 4, Src: 192.168.0.3, Dst: 192.168.0.1
▸ Transmission Control Protocol, Src Port: 52068, Dst Port: 445, Seq: 1, Ack: 1, Len: 72
▸ NetBIOS Session Service
▸ SMB2 (Server Message Block Protocol version 2)

0000  00 15 5d d0 8b 01 00 15  5d d0 8b 06 08 00 45 00   ..]....]......E.
0010  00 7c 55 e5 40 00 40 06  63 42 c0 a8 00 03 c0 a8   .|U.@.@.cB......
0020  00 01 cb 5c 01 bd a6 a7  5f 0b 10 a1 ac 33 80 18   ...\...._....3..
```

CONCLUSION:

Thus, we have studied the working of Wire Shark.

x-x-x

Experiment 05 Over

EXPERIMENT – 06

AIM: Case study on Sql Injection.

THEORY:

What is SQL Injection (SQLi) and How to Prevent It
SQL Injection (SQLi) is a type of an injection attack that makes it possible to execute malicious SQL statements. These statements control a database server behind a web application. Attackers can use SQL Injection vulnerabilities to bypass application security measures. They can go around authentication and authorization of a web page or web application and retrieve the content of the entire SQL database. They can also use SQL Injection to add, modify, and delete records in the database. An SQL Injection vulnerability may affect any website or web application that uses an SQL database such as MySQL, Oracle, SQL Server, or others. Criminals may use it to gain unauthorized access to your sensitive data: customer information, personal data, trade secrets, intellectual property, and more. SQL Injection attacks are one of the oldest, most prevalent, and most dangerous web
application vulnerabilities. The OWASP organization (Open Web Application Security Project) lists injections in their OWASP Top 10 2017 document as the number one threat to web application security.

How and Why Is an SQL Injection Attack Performed
To make an SQL Injection attack, an attacker must first find vulnerable user inputs within the web page or web application. A web page or web application that has an SQL Injection vulnerability uses such user input directly in an SQL query. The attacker can create input content. Such content is often called a malicious payload and is the key part of the attack. After the attacker sends this content, malicious SQL commands are executed in the database. SQL is a query language that was designed to manage data stored in relational databases. You can use it to access, modify, and delete data. Many web applications and websites store all the data in SQL databases. In some cases, you can also use SQL commands to run operating system commands. Therefore, a successful SQL Injection attack can have very serious consequences.
Attackers can use SQL Injections to find the credentials of other users in the database. They can then impersonate these users. The impersonated user may be a database administrator with all database privileges.

SQL lets you select and output data from the database. An SQL Injection vulnerability could allow the attacker to gain complete access to all data in a database server.
SQL also lets you alter data in a database and add new data. For example, in a financial application, an attacker could use SQL Injection to alter balances, void transactions, or transfer money to their account.
You can use SQL to delete records from a database, even drop tables. Even if the administrator makes database backups, deletion of data could affect application availability until the database is restored. Also, backups may not cover the most recent data.
In some database servers, you can access the operating system using the database server. This may be intentional or accidental. In such case, an attacker could use an SQL Injection as the initial vector and then attack the internal network behind a firewall.

Simple SQL Injection Example

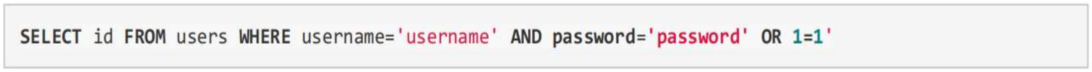
The first example is very simple. It shows, how an attacker can use an SQL Injection vulnerability to go around application security and authenticate as the administrator. The following script is pseudocode executed on a web server. It is a simple example
of authenticating with a username and a password. The example database has a table named users with the following columns: username and password.

```
# Define POST variables uname = request.POST['username'] passwd = request.POST['password']


# SQL query vulnerable to SQLi

sql = “SELECT id FROM users WHERE username=’” + uname + “’ AND password=’” + passwd + “’”


# Execute the SQL statement database.execute(sql)
```

These input fields are vulnerable to SQL Injection. An attacker could use SQL commands in the input in a way that would alter the SQL statement executed by the database server. For example, they could use a trick involving a single quote and set the passwd field to:

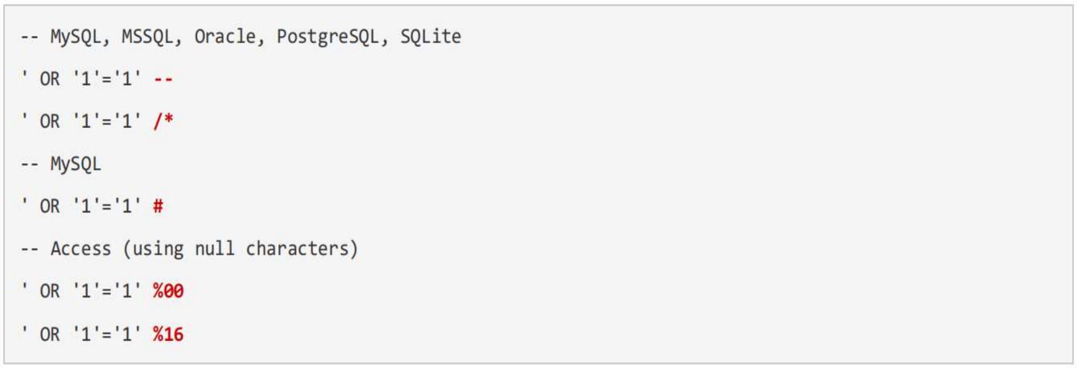
```
password' OR 1=1
```

As a result, the database server runs the following SQL query:



```
SELECT id FROM users WHERE username='username' AND password='password' OR 1=1'
```

Because of the OR 1=1 statement, the WHERE clause returns the first id from the users table no matter what the username and password are. The first user id in a database is very often the administrator. In this way, the attacker not only bypasses authentication but also gains administrator privileges. They can also comment out the rest of the SQL statement to control the execution of the SQL query further:



```
-- MySQL, MSSQL, Oracle, PostgreSQL, SQLite
' OR '1'='1' --
' OR '1'='1' /*
-- MySQL
' OR '1'='1' #
-- Access (using null characters)
' OR '1'='1' %00
' OR '1'='1' %16
```

<u>CONCLUSION:</u>

Thus, we have studied SQL Injection successfully.

x-x-x

Experiment 06 Over

((((( ))))))

(You have reached the end of the CSS Lab Manual)