KNOWLEDGE INSTITUTE OF TECHNOLOGY

(AUTONOMOUS)

(Affiliated to Anna University, Chennai)

KAKAPALAYAM (PO), SALEM – 637 504

RECORD NOTE BOOK

Register Number

Certified that this is the bonafide record of work done by Selvan/Selvi……of the…….. Semester……Branch during the year…..…….in the……...……Laboratory.

Staff In-charge **Head of the Department**

Submitted for the University Practical Examination on …..……

Internal Examiner **External Examiner**

INDEX

Ex. No.	Date	Name of the Experiment	Page No.	Marks Awarded	Faculty Signature

Vision:

To develop Computer Science engineers and skilled software professionals who can meet evolving industry demands and global challenges along with strong social values.

Mission:

- To provide need-based technical education through appropriate infrastructure, effective teaching and research.
- To meet industry requirements through collaborative projects on emerging technologies.
- To develop Computer Science and Engineering graduates with ethical values, social responsibility, entrepreneurship skills, leadership and teamwork.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1: Enable graduates to pursue higher education and research, or have a successful career in industries associated with Computer Science and Engineering or as entrepreneurs.

PEO2: Ensure that graduates will have the ability and attitude to adapt to emerging technological changes.

PEO3: Acquire leadership skills to perform professional activities with social consciousness.

PROGRAMME OUTCOMES (POs)

(POs)	Program Outcomes (POs)	Engineering Graduates will be able to
PO1	Engineering Knowledge	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	Problem Analysis	Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.
PO3	Design / Development of Solutions	Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal and environmental considerations.
PO4	Conduct Investigations of Complex Problems	Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of the information to provide valid conclusions.
PO5	Modern Tool Usage	Create, select and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	The Engineer and Society	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	Environment and Sustainability	Understand the impact of the professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of and need for sustainable development.

PO8	Ethics	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	Individual and Team Work	Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary settings.
PO10	Communication	Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.
PO11	Project Management and Finance	Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	Life-Long Learning	Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAMME SPECIFIC OUTCOMES (PSOs)

PSO1: Analyze large volume of data and make business decisions to improve efficiency with different algorithms and tools.

PSO2: Have the capacity to develop web and mobile applications for real time scenarios.

PSO3: Provide automation and smart solutions in various forms to the society with Internet of Things.

Ex. No. : 1 (A)	Implement Symmetric Key Algorithms - DES
Date:	

AIM:

To use Data Encryption Standard (DES) Algorithm for a practical application like User Message Encryption.

ALGORITHM:

1. Create a DES Key.
2. Create a Cipher instance from Cipher class, specify the following information and separated by a slash (/).
    a. Algorithm name
    b. Mode (optional)
    c. Padding scheme (optional)
3. Convert String into Byte[] array format.
4. Make Cipher in encrypt mode, and encrypt it with Cipher.doFinal() method.
5. Make Cipher in decrypt mode, and decrypt it with Cipher.doFinal() method.

PROGRAM:

```
importjava.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
 public class DES
{
public static void main(String[] argv)
{
```

```
try
 {
System.out.println("Message Encryption Using DES Algorithm\n          ");
KeyGenerator keygenerator = KeyGenerator.getInstance("DES");
SecretKey  myDesKey = keygenerator.generateKey();
Cipher desCipher;
desCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
desCipher.init(Cipher.ENCRYPT_MODE,  myDesKey);
byte[] text = "Secret Information ".getBytes();
System.out.println("Message [Byte Format] : " + text);
System.out.println("Message : " + new String(text));
byte[] textEncrypted = desCipher.doFinal(text);
System.out.println("Encrypted Message: " + textEncrypted);
desCipher.init(Cipher.DECRYPT_MODE, myDesKey);
byte[] textDecrypted = desCipher.doFinal(textEncrypted);
System.out.println("Decrypted Message: " + newString(textDecrypted));
}catch(NoSuchAlgorithmException e)
{
 e.printStackTrace();
}
catch(NoSuchPaddingExceptione)
{
e.printStackTrace();
}
catch(InvalidKeyException e)
{
e.printStackTrace();
}
catch(IllegalBlockSizeException  e)
{

e.printStackTrace();
}
```

```
catch(BadPaddingException e)
{
e.printStackTrace();
}
}
}
```

OUTPUT:

```
Message Encryption Using DES Algorithm
------------------------------------------

Message [Byte Format] : [B@4dcbadb4
Message : Secret Information Encrypted
Message: [B@504bae78 Decrypted
Message: Secret Information
```

RESULT:

Thus, the java program for the DES algorithm has been implemented and the output has been verified successfully.

Ex. No. : 1 (B)	Implement Symmetric Key Algorithms - AES
Date:	

AIM:

To use Advanced Encryption Standard (AES) Algorithm for a practical application like URL Encryption.

ALGORITHM:

1. AES is based on a design principle known as a substitution–permutation.
2. AES does not use a Feistel network like DES, it uses variant of Rijndael.
3. It has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits.
4. AES operates on a 4 × 4 column-major order array of bytes, termed the state

PROGRAM:

```
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
public class AES
{
private static SecretKeySpec secretKey;
 private static byte[] key;
public static void setKey(String myKey)
{
MessageDigest sha = null;
try
{
```

```
key = myKey.getBytes("UTF-8");
sha = MessageDigest.getInstance("SHA-1");
key = sha.digest(key);
key = Arrays.copyOf(key, 16);
secretKey = new SecretKeySpec(key, "AES");
}
catch (NoSuchAlgorithmException e)
{
e.printStackTrace();
}
catch (UnsupportedEncodingException e)
{
e.printStackTrace();
}
}
public static String encrypt(String strToEncrypt, String secret)
{
try
{
setKey(secret);
Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
cipher.init(Cipher.ENCRYPT_MODE, secretKey);
return
Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));
}
catch (Exception e)
{
System.out.println("Error while encrypting: " + e.toString());
}
return null;
```

```
}
public static String decrypt(String strToDecrypt, String secret)
{
try
{
setKey(secret);
Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
cipher.init(Cipher.DECRYPT_MODE, secretKey);
return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
}
catch (Exception e)
{
System.out.println("Error while decrypting: " + e.toString());
}
return null;
}
public static void main(String[] args)
{
final String secretKey = "annaUniversity";
String originalString = "www.annauniv.edu";
String encryptedString = AES.encrypt(originalString, secretKey);
 String decryptedString = AES.decrypt(encryptedString, secretKey);
System.out.println("URL Encryption Using AES Algorithm\n      ");
System.out.println("Original URL : " + originalString);
 System.out.println("Encrypted URL : " + encryptedString);
System.out.println("Decrypted URL : " + decryptedString);
}
}
```

OUTPUT:

URL Encryption Using AES Algorithm

Original URL : www.annauniv.edu

Encrypted URL : vibpFJW6Cvs5Y+L7t4N6YWWe07+JzS1d3CU2h3mEvEg=

Decrypted URL : www.annauniv.edu

RESULT:

Thus, the java program for AES algorithm has been implemented and the output has been verified successfully.

Ex. No. : 1 (C)	Implement Symmetric Key Algorithms - Blowfish
Date:	

AIM:

To implement the symmetric key algorithm using blowfish.

ALGORITHM:

1. Hard code a password and a key for demonstration purposes.
2. Print the original password.
3. Create an instance of Blowfish Demo.
4. Generate an encrypted text by calling the encrypt method with the password and key.
5. Print the encrypted text.
6. Decrypt the encrypted text using the decrypt method with the key.
7. Print the decrypted text.

PROGRAM :

```
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
importjava.nio.charset.StandardCharsets;
import java.util.Base64;
import java.util.Scanner;
publicclassBlowfishImplementationWithInput
{
publicstaticbyte[]encrypt(Stringkey, Stringplaintext) throwsException
{
SecretKeysecretKey   =newSecretKeySpec(key.getBytes(StandardCharsets.UTF_8),“Blowfish");
Cipher cipher = Cipher.getInstance("Blowfish");
cipher.init(Cipher.ENCRYPT_MODE,secretKey);
returncipher.doFinal(plaintext.getBytes(StandardCharsets.UTF_8));
}
publicstaticStringdecrypt(Stringkey,byte[] ciphertext)throwsException
{
SecretKeysecretKey=newSecretKeySpec(key.getBytes(StandardCharsets.UTF_8),"Blowfish");
```

```
Cipher cipher = Cipher.getInstance("Blowfish");
cipher.init(Cipher.DECRYPT_MODE, secretKey);
byte[]decryptedBytes=cipher.doFinal(ciphertext);
returnnewString(decryptedBytes,StandardCharsets.UTF_8);
}
publicstaticvoidmain(String[]args)
{ try {
Scannerscanner=newScanner(System.in);
/Input keyfrom user
System.out.print("Enterthekey(upto56bytes):");
String key = scanner.nextLine();
// Input plaintext from user System.out.print("Entertheplaintext:");
String plaintext = scanner.nextLine();
// Encryption
byte[] ciphertext = encrypt(key, plaintext); System.out.println("Ciphertext(Base64):"+
Base64.getEncoder().encodeToString(ciphertext));
// Decryption
String decryptedText = decrypt(key, ciphertext);
System.out.println("DecryptedText:"+decryptedText);
scanner.close();
}catch(Exceptione)
{
e.printStackTrace();
}}}
```

OUTPUT:

```
Message data = 12.000000
Encrypted data = 3.000000
Original Message Sent = 12.000000
```

RESULT:

Thus, the java program for symmetric key using blowfish algorithm has been implemented and the output has been verified successfully.

Ex. No. : 2 (A)	**Implement Asymmetric Key Algorithms - Diffie Hellman Key Exchange**
Date:	

AIM:

To implement the asymmetric key algorithm using diffie hellman key exchange algorithm.

ALGORITHM:

1. Choose two prime numbers p and q.
2. Calculate n =p * q, which is part of the public key.
3. Choose a public exponent e such that it is coprime with (p-1)* (q-1)(denoted as phi).
4. Find a private exponent d such that (d* e)%phi=1.
   n and e form the public key, while n and d form the private key.
5. Choose a message msg to be encrypted.
6. Encrypt the message using the public exponent e and modulus n.
7. Decrypt the cipher text c using the private exponent d and modulus n.
8. Print the original message, encrypted data, and the decrypted original message.
9. Function to calculate the greatest common divisor (GCD) of two numbers.

PROGRAM:

```
import java.io.*;
importjava.math.*;
import java.util.*;
public class GFG
{
publicstatic doublegcd(doublea, double h)
{
doubletemp; while (true)
{
temp = a % h; if (temp == 0) return h;
a=h; h =temp;
}}
publicstaticvoidmain(String[]args)
{
double p = 3;
```

```
double q = 7;
doublen=p*q;
double e = 2;
doublephi=(p-1)*(q-1);
while (e< phi)
{
if(gcd(e,phi)==1)
break;
else
e++;   }
int k =2;
doubled=(1+(k*phi))/e;
double msg = 12;
System.out.println("Messagedata="+msg);
doublec=Math.pow(msg,e);
c = c% n;
System.out.println("Encrypteddata="+c);
doublem=Math.pow(c,d); m = m% n;
System.out.println("OriginalMessageSent= "+ m);
}}
```

OUTPUT:

```
The value of P : 23
The value of G : 9
The private key a for Alice : 4
The private key b for Bob : 3
Secret key for the Alice is : 9
Secret Key for the Bob is : 9
```

RESULT:

Thus, the java program for diffie hellman key exchange algorithms has been implemented and the output has been verified successfully.

Ex. No. : 2 (B)	**Implement Asymmetric Key Algorithms - RSA Algorithm**
Date:	

AIM:

To implement the asymmetric key exchange algorithm using RSA algorithm.

ALGORITHM:

1. Initialize prime number P and primitive root G.
2. Choose private keys for Alice(a) and Bob(b).
3. Calculate public keys for Alice(x) and Bob(y) using the chosen primitive root and private keys.
4. Calculate the shared secrets for Alice(ka) and Bob (kb) using their private keys and the other party's public key.
5. Print the computed shared secrets for Alice and Bob.
6. Implement a power function o efficiently calculate modular exponentiation.

PROGRAM:

```
classGFG
{
privatestatic longpower(longa, longb, longp)
{
if(b==1) return a;
else
return(((long)Math.pow(a,b))%p);
}
publicstaticvoidmain(String[]args)
{
longP,G,x,a,y,b,ka,kb;
P = 23;
System.out.println("ThevalueofP:"+P);
G= 9;
```

```
System.out.println("ThevalueofG:"+G);
a= 4;
System.out.println("TheprivatekeyaforAlice:"+a);
x = power(G, a, P);
b =3;
System.out.println("TheprivatekeybforBob:"+b);
y = power(G, b, P);
ka=power(y,a,P);
//SecretkeyforAlice
kb =power(x, b, P);
// Secret key for Bob
System.out.println("SecretkeyfortheAliceis:"+ka);
System.out.println("Secret key for the Bob is:"+ kb);
}
}
```

OUTPUT:

Simulation of Diffie-Hellman key exchange algorithm

Alice Sends: 4.0

Bob Computes: 18.0

Bob Sends: 10.0

Alice Computes: 18.0

Shared Secret: 18.0

Success: Shared Secrets Matches! 18.0

RESULT:

Thus, the java program for the RSA algorithm has been implemented and the output has been verified successfully.

Ex. No. : 3	Implement Digital Signature Schemes
Date:	

AIM:

To implement the signature scheme - Digital Signature Standard.

ALGORITHM:

1. Create a KeyPairGenerator object.
2. Initialize the KeyPairGenerator object.
3. Generate the KeyPairGenerator.
4. Get the private key from the pair.
5. Create a signature object.
6. Initialize the Signature object.
7. Add data to the Signature object
8. Calculate the Signature

PROGRAM:

```
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.Signature;
import java.util.Scanner;
public class CreatingDigitalSignature
{
public static void main(String args[]) throws Exception
{
Scanner sc = new Scanner(System.in);
System.out.println("Enter some text");
String msg = sc.nextLine();
KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("DSA");
keyPairGen.initialize(2048);
KeyPair pair = keyPairGen.generateKeyPair();
```

```
PrivateKey privKey = pair.getPrivate();
Signature sign = Signature.getInstance("SHA256withDSA");
sign.initSign(privKey);
 byte[] bytes = "msg".getBytes();
sign.update(bytes);
byte[] signature = sign.sign();
System.out.println("Digital signature for given text: "+new String(signature, "UTF8"));
}
}
```

OUTPUT:

```
Enter some text:
Hi how are you
Digital signature for given text: 0=@gRD???-?.???? /yGL?i??a!?
```

RESULT:

Thus, the java program for the digital signature standard algorithm has been implemented and the output has been verified successfully.

Ex. No. : 4	**Installation of Wire shark, TCP dump and observe Data Transferred in Client-Server Communication using UDP/TCP and Identify the UDP/TCP Datagram.**
Date:	

AIM:

To install the wireshark, TCP dump and observe Data Transferred in Client-Server Communication using UDP/TCP and Identify the UDP/TCP Datagram.

PROCEDURE:

Step 1:

Open your web browser and go to the official page of wireshark.org. And downdload the appropriate version of wireshark according to your system.

Step 2:

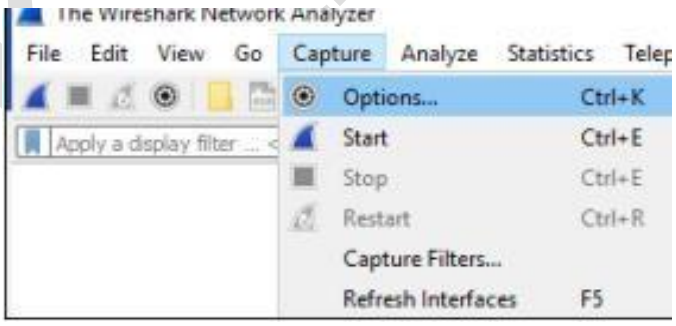
Once the download is completed, navigate to the location where the installer file was saved.

Step 3:

Double click on the installer file and install the wireshark by click on “next” step.

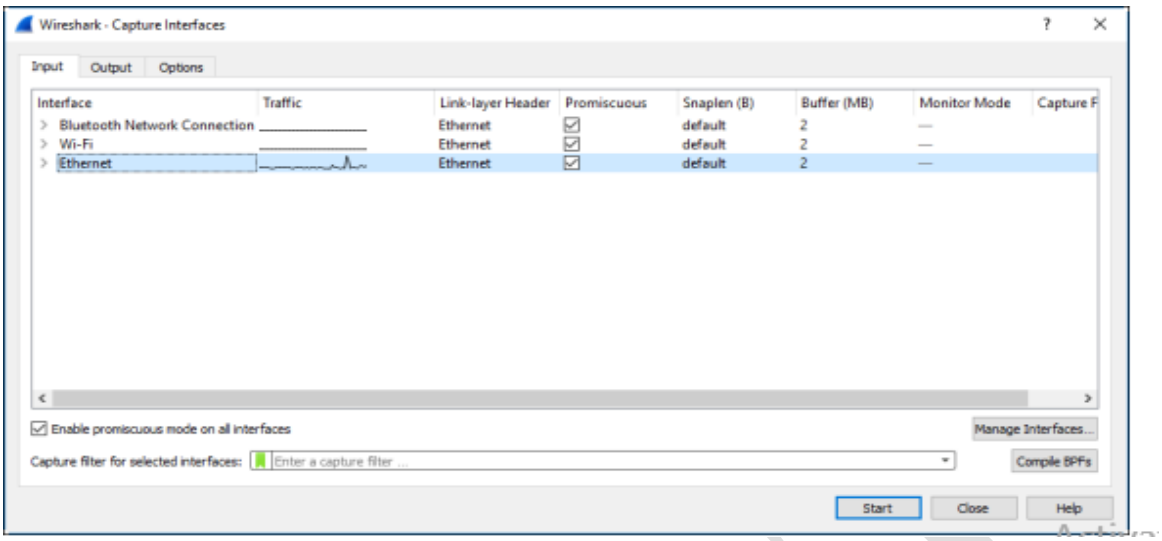
Step 4:

After the installation was successfully completed. Once the application launches, click on the Capture > Options menu



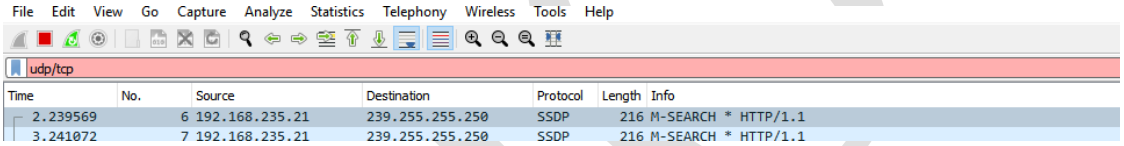
Step 5:

From the Capture Interfaces panel, select your network Interface on the Input tab, and then click Start.



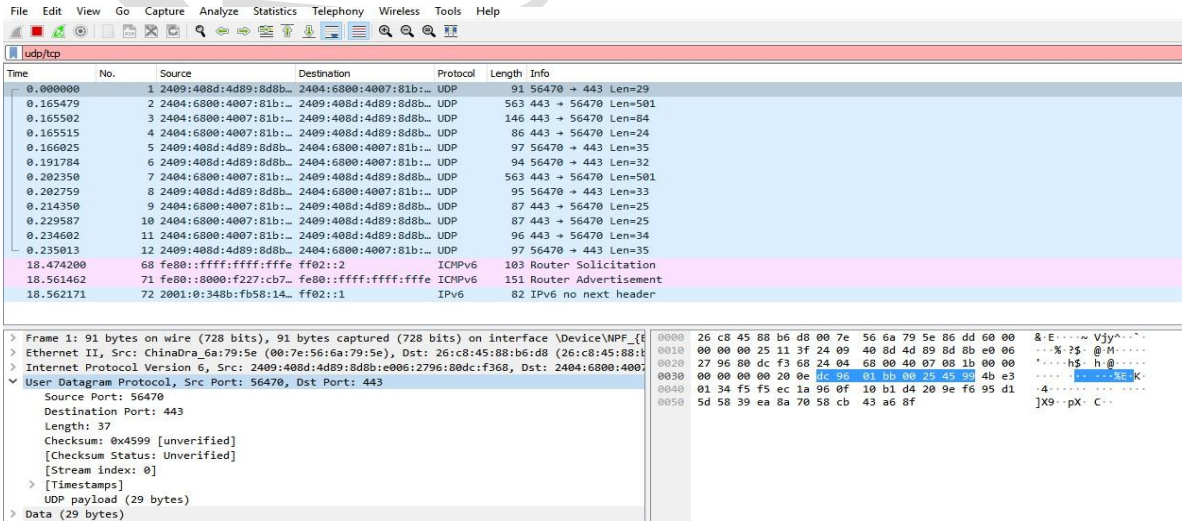
Step 6:

To communicate with UDP/TCP protocols click on the fliter tab and in that type UDP/TCP as shown

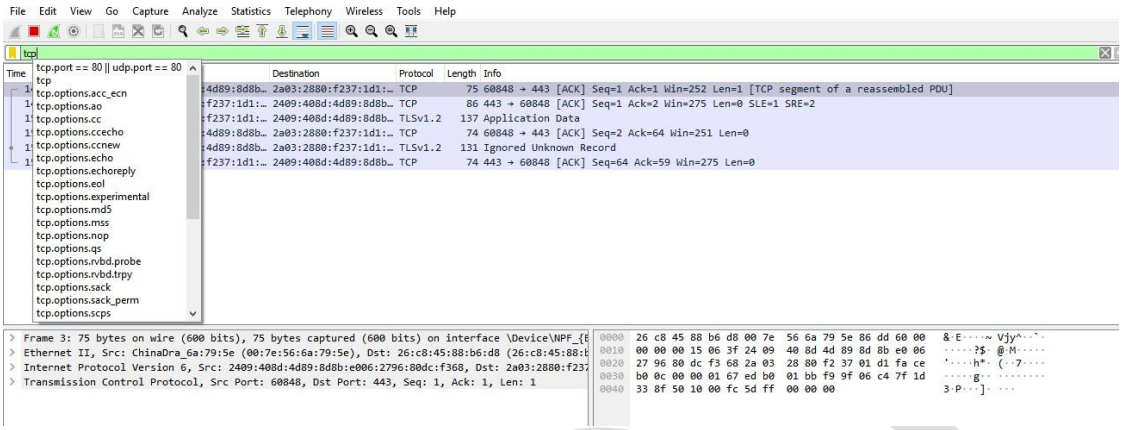


Step7:

Then click on “stop” button and click on start button to identify the udp/tcp protocol.
(I) The identification of UDP protocol as shown below:



(II) The identification of TCP protocol as shown:



RESULT:

Thus, the installation of wireshark, TCP dump and data transformation in client-server communication using UDP/TCP has been implemented and the output has been verified successfully.

Ex. No. : 5	Check Message Integrity and Confidentiality using SSL
Date:	

AIM:

To check message integrity and confidentiality using secure socket layer.

SERVER ALGORITHM:

1. Set the path to the server's key store and trust store files.
2. Create an SSL server socket using the default SSL server socket factory.
3. Wait for a client connection by accepting incoming connections on the SSL server socket.
4. Once a client connects, create an SSL socket for communication.
5. Setup input and output streams for reading from and writing to the SSL socket.
6. Read the incoming message from the client.
7. Process the received message(perform any necessary operations).
8. Prepare a response message.
9. Write the response message to the output stream to send it back to the client.
10. Close the input/output streams and the SSL socket.
11. Close the SSL server socket.

PROGRAM:

SERVER

```
importjava.io.BufferedReader;
importjava.io.InputStreamReader;
import java.io.OutputStream;
import javax.net.ssl.SSLServerSocket;
importjavax.net.ssl.SSLServerSocketFactory;
import javax.net.ssl.SSLSocket;
publicclassSecureServer
{
publicstaticvoidmain(String[]args)
{
try
{
```

System.setProperty("javax.net.ssl.keyStore", "serverkeystore.jks");
System.setProperty("javax.net.ssl.keyStorePassword","keystorepassword");

PROGRAM:

SERVER

```
import java.io.BufferedReader;
importjava.io.InputStreamReader;
import java.io.OutputStream;
import javax.net.ssl.SSLServerSocket;
importjavax.net.ssl.SSLServerSocketFactory;
import javax.net.ssl.SSLSocket;
publicclassSecureServer
{
publicstaticvoidmain(String[]args)
{
try
{
System.setProperty("javax.net.ssl.keyStore", "serverkeystore.jks");
System.setProperty("javax.net.ssl.keyStorePassword","keystorepassword");
System.setProperty("javax.net.ssl.trustStore", "servertruststore.jks");
System.setProperty("javax.net.ssl.trustStorePassword", "truststorepassword");
SSLServerSocketFactorysslServerSocketFactory=(SSLServerSocketFactory)
SSLServerSocketFactory.getDefault();
SSLServerSocketserverSocket = (SSLServerSocket)
sslServerSocketFactory.createServerSocket(9999);
System.out.println("Serverwaitingforclientconnection...");
SSLSocketsslSocket = (SSLSocket) serverSocket.accept();
System.out.println("Client connected.");
BufferedReader reader = new BufferedReader(new
InputStreamReader(sslSocket.getInputStream()));
OutputStreamoutputStream = sslSocket.getOutputStream();
String clientMessage = reader.readLine();
System.out.println("Receivedfromclient:"+clientMessage);
```

```
String processedMessage = "Processed: " + clientMessage;
outputStream.write(processedMessage.getBytes());
System.out.println("Sent to client: " + processedMessage);
reader.close();
outputStream.close();
sslSocket.close();
serverSocket.close();
}
catch(Exceptione)
{
e.printStackTrace();
}
}
}
```

CLIENT ALGORITHM:

1. Set the path to the client's key store and trust store files.
2. Create an SSL socket factory using the default SSL socket factory.
3. Create an SSL socket and connect to the server.
4. Set up input and output streams for reading from and writing to the SSL socket.
5. Prepare a message to send to the server.
6. Write the message to the output stream to end it to the server.
7. Read the response message from the server
8. Process the received response (perform any necessary operations).
9. Close the input/output streams and the SSL socket.

PROGRAM :

CLIENT

```
import java.io.BufferedReader;
importjava.io.InputStreamReader;
import java.io.OutputStream;
import javax.net.ssl.SSLSocket;
importjavax.net.ssl.SSLSocketFactory;
public class SecureClient
```

```
{
publicstaticvoidmain(String[]args)
{
try
{
System.setProperty("javax.net.ssl.keyStore", "clientkeystore.jks");
System.setProperty("javax.net.ssl.keyStorePassword", "keystorepassword");
System.setProperty("javax.net.ssl.trustStore", "clienttruststore.jks");
System.setProperty("javax.net.ssl.trustStorePassword", "truststorepassword");
SSLSocketFactorysslSocketFactory = (SSLSocketFactory) SSLSocketFactory.getDefault();
SSLSocketsslSocket = (SSLSocket) sslSocketFactory.createSocket("localhost", 9999);
System.out.println("Connected to server.");
BufferedReaderreader=newBufferedReader(newInputStreamReader(sslSocket.getInputStre
am()));
OutputStreamoutputStream = sslSocket.getOutputStream();
String message = "Hello from the client!";
System.out.println("Sendingmessagetoserver:"+message;
outputStream.write(message.getBytes());
StringserverResponse=reader.readLine();
System.out.println("Receivedresponsefromserver:"+serverResponse);
reader.close();
outputStream.close();
sslSocket.close();
}
catch(Exceptione)
{
e.printStackTrace();
}
}
}
```

SERVER OUTPUT:

Server waiting for client connection...
Client connected.
Received from client:
Hello from the client!
Sent to client: Processed:
Hello from the client!

CLIENT OUTPUT:

Connected to server.
Sending message to server:
Hello from the client!
Received response from server: Processed:
Hello from the client!

ATTACK PREVENTION

Strong WEP/WAP Encryption on Access Points.
Strong Router Login Credentials.
Virtual Private Network. Force HTTPS.
Public Key Pair Based Authentication.

RESULT:

Thus, the java program for the secure socket layer has been implemented and the output has been verified successfully.

Ex. No. : 6	Experiment Eavesdropping, Dictionary attacks, MITM Attacks
Date:	

AIM:

To study on Eavesdropping, Dictionary attacks and MITM attacks.

EAVESDROPPING:

Definition:

An Eavesdropping attack occurs when a hacker intercepts, deletes, or modifies data that is transmitted between two devices. Eavesdropping, also known as sniffing or snooping, relies on unsecured network communications to access data in transit between devices. The data is transmitted across an open network, which gives an attacker the opportunity to exploit vulnerability and intercept it via various methods. Eavesdropping attacks can often be difficult to spot.

Attacks:

Eavesdropping attacks are a big threat to the integrity and confidentiality of the data. It allows an attacker to gather sensitive information, such as login credentials, financial data, or personal conversations, without the victim's knowledge. Furthermore, attackers can use the extracted information for various malicious purposes, such as identity theft, extortion, or espionage.

Let's look at the general steps in order to launch an Eaves dropping attack:

The first step is identifying a target for the attack, such as a specific individual or organization. As soon as the attacker identifies the target, it starts gathering information about it. Some useful information the attacker wants to extract includes the communication systems and vulnerabilities that can be exploited.

The next step is to choose an appropriate method for the successful execution of the attack. There're several different methods that an attacker can use. Some examples are intercepting communication over unsecured networks, using malware to gain access to a device, or using hardware devices.

The next step is to execute the chosen method in the target system and intercept the target's communication. Finally, the attacker analyzes the intercepted communication and extracts valuable information.

Prevention Techniques:

We can use several techniques to prevent eavesdropping attacks. Some popular techniques include encryption, virtual private networks, secure communication protocols, firewalls, and network segmentation. Encrypting communication makes it difficult for attackers to intercept and

read messages. In order to encrypt communication, we can use different types of encryption algorithms, such as symmetric key algorithms and public key algorithms. Advanced Encryption Standard (AES) is an example of a symmetric key algorithm. Additionally, Rivest–Shamir–Adleman (RSA) is a widely used public key algorithm.

Virtual private networks (VPNs) create a secure, encrypted connection between a device and a remote server. They can help to prevent eaves dropping attacks by encrypting communication and making it difficult for attackers to intercept.

DICTIONARYATTACKS:

Definition:

A dictionary attack is a method of breaking into a password-protected computer, network or other IT resource by systematically entering every word in a dictionary as a password. A dictionary attack can also be used in an attempt to find the key necessary to decrypt an encrypted message or document.

Dictionary attacks work because many computer users and businesses insist on using ordinary words as passwords. These attacks are usually unsuccessful against systems using multiple-word passwords and are also often unsuccessful against passwords made up of uppercase and lowercase letters and numbers in random combinations.

How do dictionary attacks work?

A dictionary attack uses a preselected library of words and phrases to guess possible passwords. It operates under the assumption that users tend to pull from a basic list of passwords, such as "password," "123abc" and "123456."

These lists include predictable patterns that can vary by region. For example, hackers looking to launch a dictionary attack on a New York-based group of targets might look to test phrases like "knicksfan2020" or "newyorkknicks1234." Attackers incorporate words related to sports teams, monuments, cities, addresses and other regionally specific items when building their attack library dictionaries.

How effective is a dictionary attack?

How successful a dictionary attack is depending on how strong the passwords are for the individuals a hacker is targeting? Because weak passwords are still common, attackers continue to have success with these attacks. Individual users, however, aren't the only ones who are subject to weak password security.

Take steps to prevent a dictionary attack

Dictionary hacking is a very common type of cybercrime that hackers use to gain access to an individual’s personal accounts, including bank accounts, social media profiles, and emails. With this access, hackers can perpetrate all sorts of actions, from financial fraud and malicious social media posts to further cybercrimes like phishing. However, dictionary attack prevention can be as simple as implementing certain safeguards to minimize the risk of falling victim to these attacks.

Using smart password management habits, employing different types of authentications, and using readily available password managers, for example, can all help keep passwords and accounts secure.

MITM ATTACKS:

What is a Man-in-the-Middle (MITM) Attack?

Man-in-the-middle attacks (MITM) are a common type of cyber security attack that allows attackers to eavesdrop on the communication between two targets. The attack takes place in between two legitimately communicating hosts, allowing the attacker to "listen" to a conversation they should normally not be able to listen to, hence the name "man-in-the-middle.

Types of Man-in-the- Middle Attacks Rogue Access Point

Devices equipped with wireless cards will often try to auto-connect to the access point that is emitting the strongest signal. Attackers can set up their own wireless access point and trick nearby devices to join its domain.

ARP Spoofing

ARP is the Address Resolution Protocol. It is used to resolve IP addresses to physical MAC (media access control) addresses in a local area network. When a host needs to talk to a host with a given IP address, it references the ARP cache to resolve the IP address to a MAC address.

MDNS Spoofing

Multicast DNS is similar to DNS, but it's done on a local area network (LAN) using broadcast like ARP. This makes it a perfect target for spoofing attacks. The local name resolution system is supposed to make the configuration of network devices extremely simple.

DNS Spoofing

Similar to the way ARP resolves IP addresses to MAC addresses on a LAN, DNS resolves domain names to IP addresses. When using a DNS spoofing attack, the attacker attempts to introduce corrupt DNS cache information to a host in an attempt to access another host using their domain name, such as www.onlinebanking.com.

Man-in-the- Middle Attack Techniques Sniffing

Attackersusepacketcapturetoolstoinspectpacketsatalowlevel.Usingspecificwireless devices that are allowed to be put into monitoring or promiscuous mode can allow an attacker to see packets. That is not intended for it to see, such as packets addressed to other hosts.

Packet Injection

An attacker can also leverage their device's monitoring mode to inject malicious packets into data communication streams. The packets can blend in with valid data communication streams, appearing to be part of the communication, but malicious in nature. Packet injection usually involves first sniffing to determine how and when to craft and send packets.

Session Hijacking

Most web applications use a login mechanism that generates a temporary session token to use for future requests to avoid requiring the user to type a password at every page. An attacker can sniff sensitive traffic to identify the session token for a user and use it to make requests as the user.

The attacker does not need to spoof once he has a session token

SSL Stripping

Since using HTTPS is a common safeguard against ARP or DNS spoofing, attackers use SSL stripping to intercept packets and alter their HTTPS-based address requests to go to their HTTP equivalent endpoint, forcing the host to make requests to the server unencrypted. Sensitive information can be leaked in plain text.

How to Detect a Man-in-the-Middle Attack

Detecting a Man-in-the-middle attack can be difficult without taking the proper steps. If you aren't actively searching to determine if your communications have been intercepted, a Man-in-the-middle attack can potentially go unnoticed until it's too late. Checking for proper page authentication and implementing some sort of tamper detection are typically the key methods to detect a possible attack, but these procedures might require extra forensic analysis after- the-fact.

RESULT:

Thus, the Eavesdropping, dictionary attacks and MITM attacks are observed.

Ex. No. : 7	Experiment with Sniff Traffic using ARP Poisoning
Date:	

AIM:

To perform the experiment with sniff traffic using ARP poisoning.

PROCEDURE:

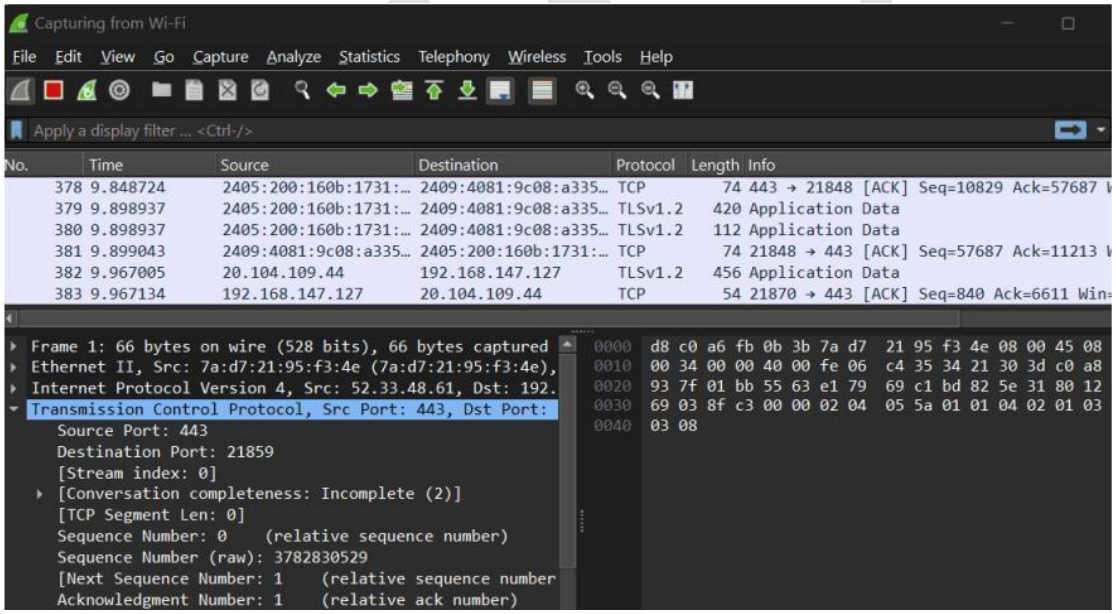
Packet Sniffing in Wireshark Step 1: Open Wireshark Window

Open your Wireshark tool in a new window or a Linux virtual machine and begin network capture. Assume we are capturing wireless fidelity (Wi-Fi Traffic).

SNIFF TRAFFIC USING ARP POISONING

Step 2: Start the Wireless Fidelity

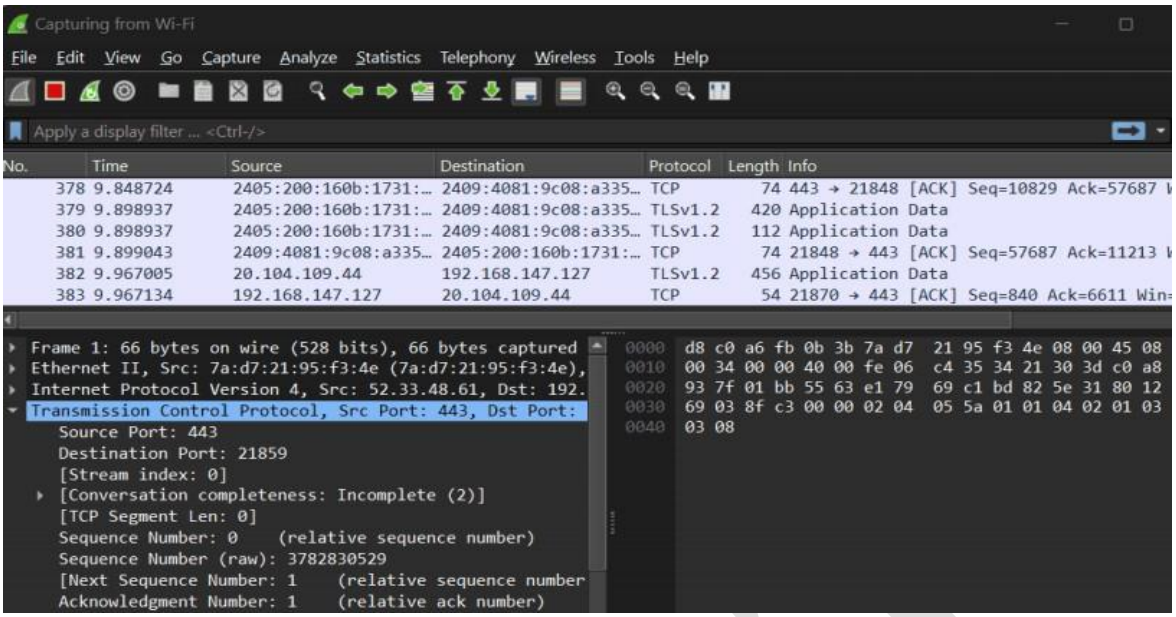
Open Wireshark, select your Wi-Fi interface and start capturing packets. Engage in network activities, stop the capture, and analyze the captured packets for network insights.



Starting Wireless Fidelity

ARP Spoofing in WireShark Step 1: Capture ARP Traffic

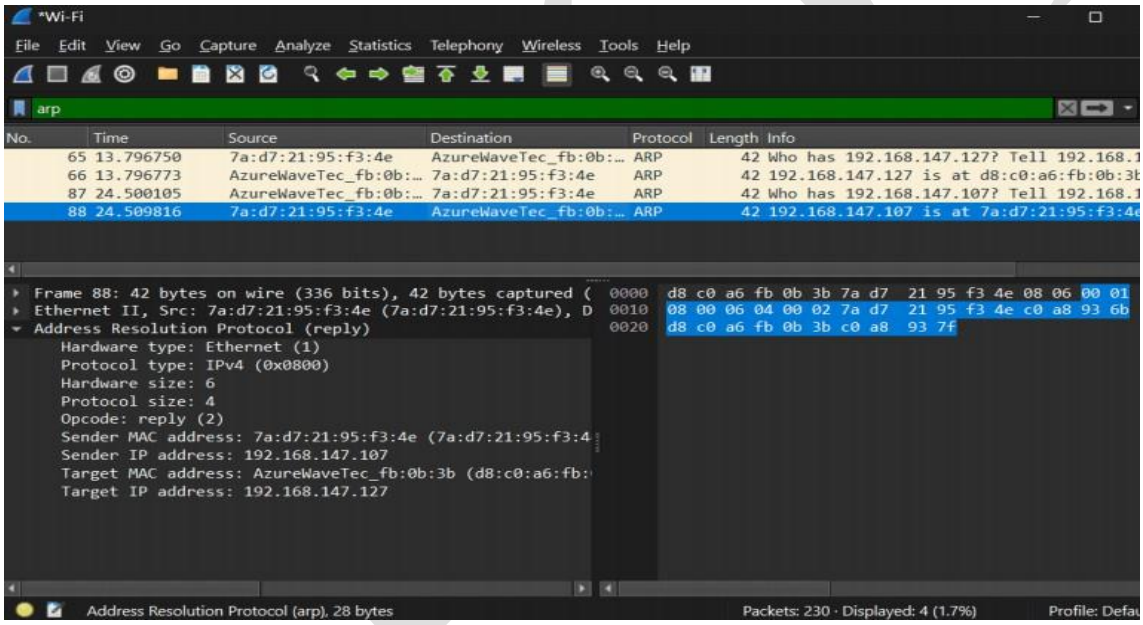
Open Wireshark, and choose the network interface. Start capturing by clicking the interface and selecting "**Start**."



Capturing Wireless Fidelity

Step 2: Filter for ARP Traffic

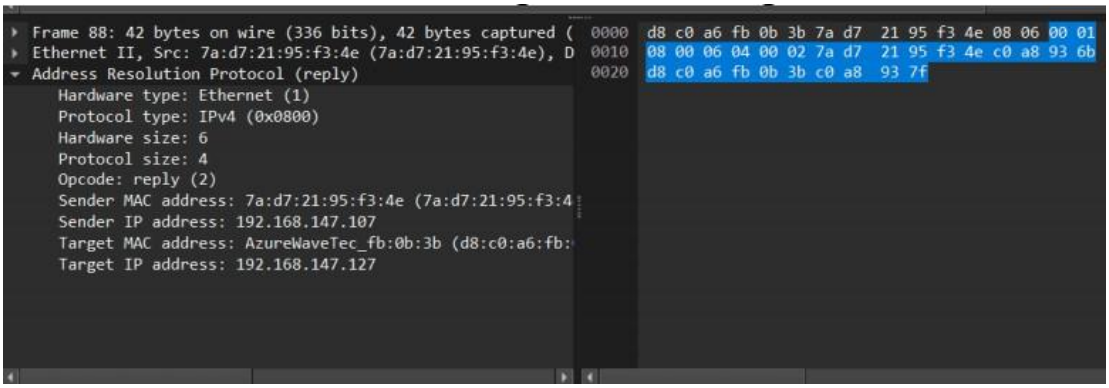
Apply the **ARP** filter in the display filter bar to focus on ARP packets.



Filtering ARP Packets

Step 3: Analyze ARP Requests and Replies:

Observe normal ARP traffic—devices asking for and receiving MAC addresses.



Analyze ARP Packet requests

1. **Identify Inconsistencies:** Look for anomalies like multiple devices responding to one ARP request or multiple MACs for one IP.
2. **Check Gratuitous ARP:** Detect gratuitous ARP packets—devices announcing MAC for an IP without prompting.
3. **Compare with Network Topology:** Understand legitimate devices and MAC addresses; compare with observed ARP responses.

RESULT:

Thus, the experiment with sniff traffic using ARP poisoning has been executed successfully.

Ex. No. : 8	Demonstrate Intrusion Detection System using any tool
Date:	

AIM:

To demonstrate Intrusion Detection System (IDS) using Snort software tool.

PROCEDURE:

STEPS ON CONFIGURING AND INTRUSION DETECTION:

1. For Windows 10 64 bit supported SNORT's executable file can be downloaded from **here.**
2. Open the downloaded snort executable file.
3. Click On 'I Agree' on the license agreement.

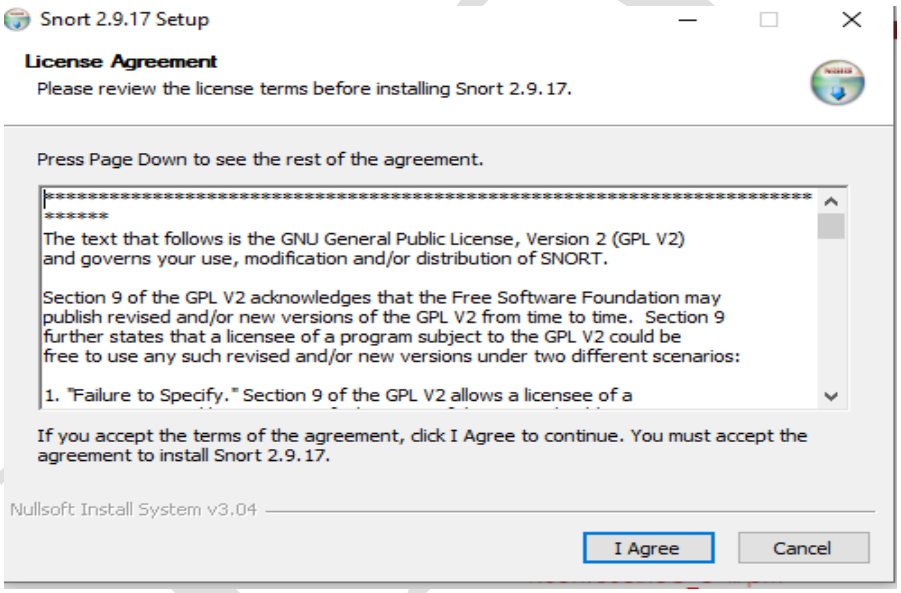


Figure 1: License agreement for Snort 2.9.17

4. Choose components of Snort to be installed.

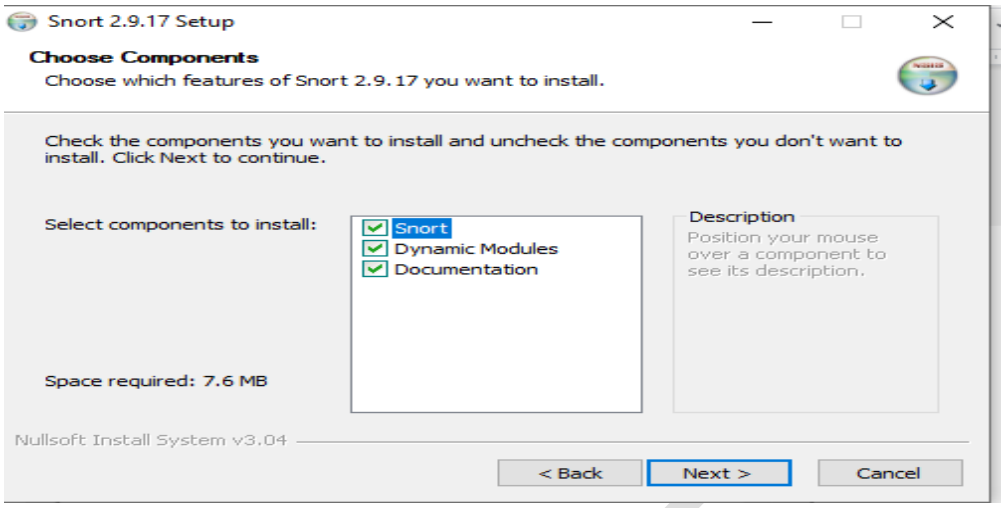


Figure 2: Choosing Components for Snort 2.9.17

5. Click "Next" and then choose install location for snort preferably a separate folder in Windows C Drive.

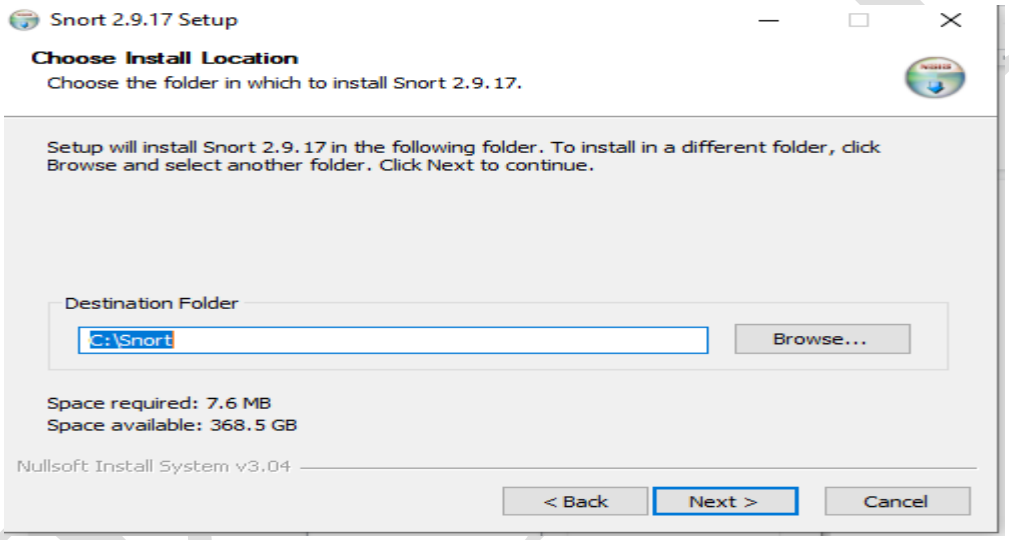


Figure 3: Choose Install location for Snort 2.9.17

6. Click "Next" Installation process starts and then it completes as shown in figure 04:

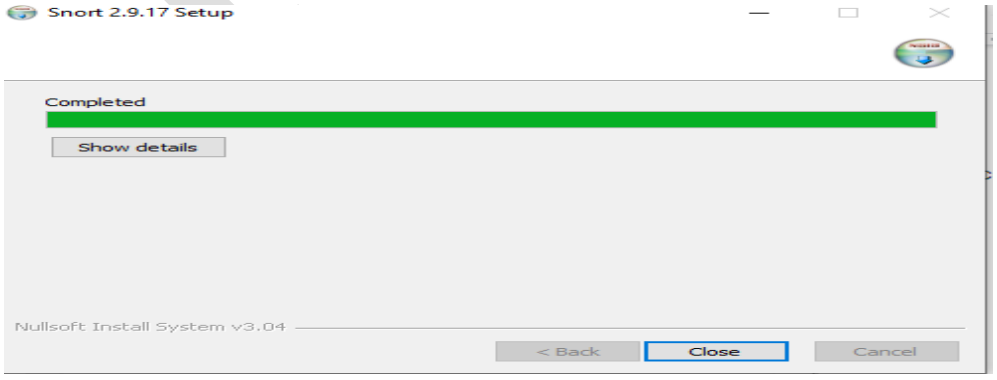


Figure 4: Setup Complete for Snort 2.9.17

7. When you click " Close" you are prompted with this dialogue box:

Figure 5: Window showing details of software needed to run Snort successfully

8. Installing Npcap is required by snort for proper functioning.
9. Npcap for Windows 10 can be downloaded from **here.**
10. Opening Npcap setup file, Click on 'I Agree' To license agreement.

Figure 6: License agreement for Npcap 1.10

11. Now we proceed to choose which components of Npcap are to be installed and then clicking on "Install".

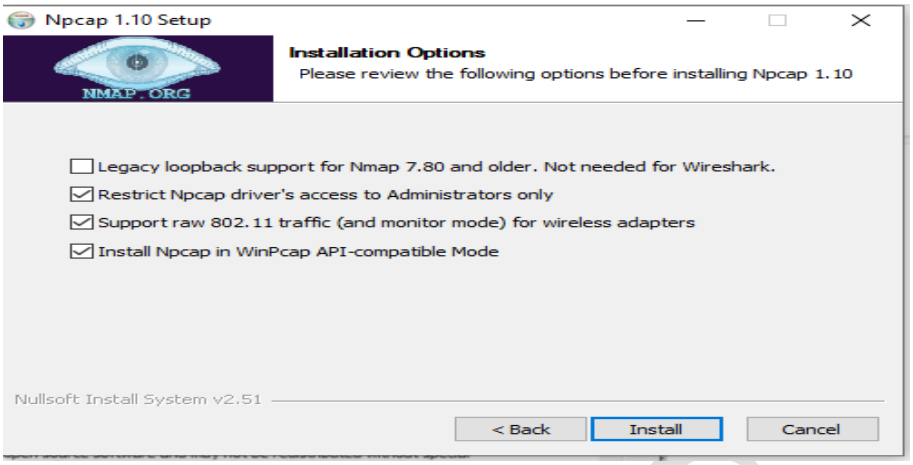


Figure 7: Choose Components to install for Npcap 1.10

12.Installation process starts and completes. Clicking on "Next" we have:

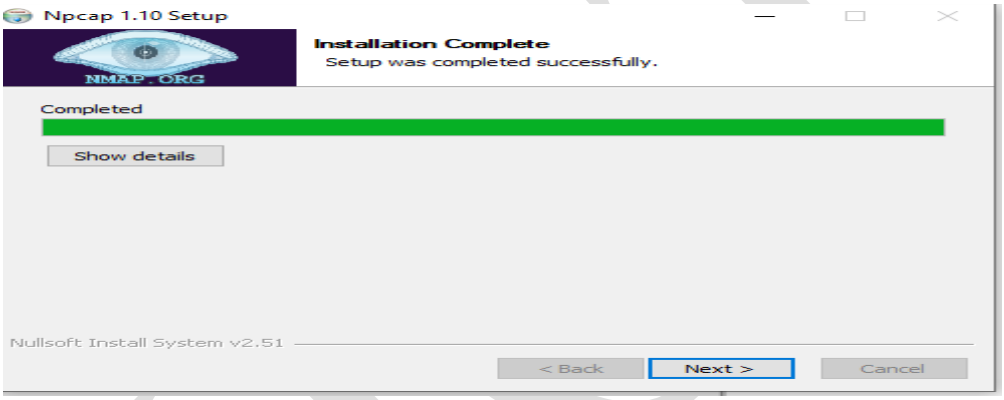


Figure 8: Setup completed for Npcap 1.10

13.Now the window for installation of Npcap shows it has been installed. Clicking "Finish".

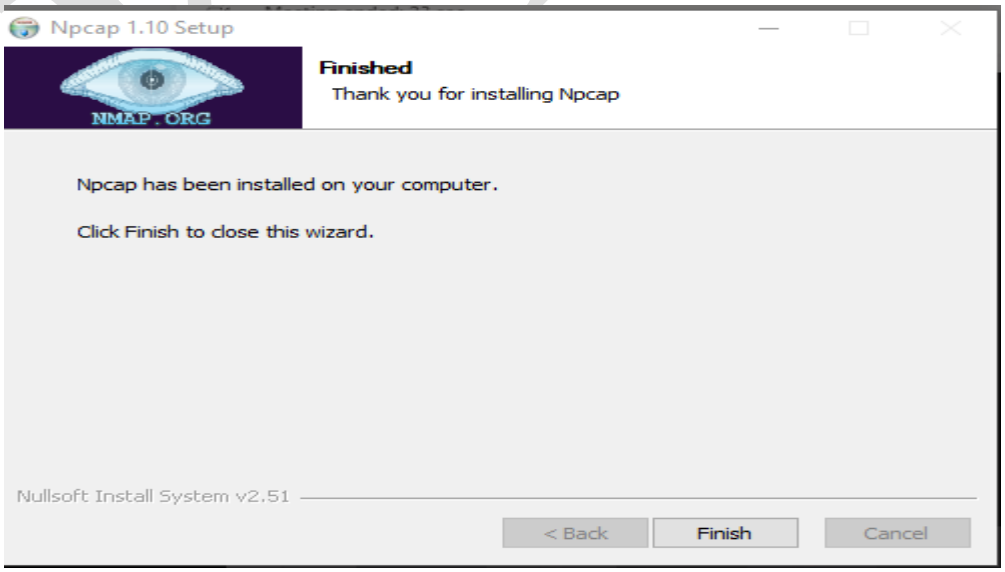


Figure 9: Successful installation for Npcap 1.10 completed

14. After installing Snort and Npcap enter these commands in windows 10 Command prompt to check snorts working

```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Snort\bin

C:\Snort\bin>snort -V

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.20-WIN64 GRE (Build 82)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.11
```

Figure 10: Successfully running Snort on Windows 10 through command prompt

15. As you can see in the above figure that snort runs successfully.

This is how you can download and install Snort along with its dependency i.e. Npcap.

Configuring Snort 2.9.17 on Windows 10:

After installing Snort on Windows 10, Another important step to get started with Snort is configuring it on Windows 10.

1. Go to this **link** and download latest snort rule file.
2. Extract 3 folders from the downloaded snortrules-snapshot-29170.tar folder into the Snorts corresponding folders in C drive.

Folders to be extracted are: **rules , preproc_rules , etc**

- **rules folder** contains the rules files and the most important **local.rules** file. Which we will use to enter all our rules.
- **etc folder** contains all configuration files and the most important file is **snort.conf** file which we will use for configuration

3. Now open the **snort.conf** file through the notepad++ editor or any other text editor to edit configurations of snort to make it work like we want it to.

4. Setup the network addresses you are protecting

ipvar HOME_NET any

Note: Mention your own host IP addresses that you want to protect.

```
44  # Setup the network addresses you are protecting
45  ipvar HOME_NET 192.168.100.27/24
46
```

Figure 11: Setting up the Home Network Address in Snort

5. Setup the external network into anything that is not the home network. That is why ! is used in the command it denotes ‘not’.

 # Set up the external network addresses. Leave as “any” in most situationsipvar EXTERNAL_NET any

```
47  # Set up the external network addresses. Leave as "any" in most situations
48  ipvar EXTERNAL_NET !$HOME_NET
49
```

Figure 12: Setting up the external Network Addresses in Snort

6.Now we have to define the directory for our rules and preproc rules folder

Path to your rules files (this can be a relative path)# Note for Windows users: You are advised to make this an absolute path,# such as: c:\snort\rulesvar RULE_PATH ../rulesvar SO_RULE_PATH ../so_rulesvar PREPROC_RULE_PATH ../preproc_rules

```
101 # Path to your rules files (this can be a relative path)
102 # Note for Windows users:  You are advised to make this an absolute path,
103 # such as:  c:\Snort\rules
104 var RULE_PATH c:\Snort\rules
105 # var SO_RULE_PATH ../so_rules
106 var PREPROC_RULE_PATH c:\Snort\preproc_rules
```

Figure 13: Setting up path to our rules files and preproc rules folder in Snort

7.Now we have to setup our white list and black list path it will be in our snorts’ rule folder

If you are using reputation preprocessor set thesevar WHITE_LIST_PATH ../rulesvar BLACK_LIST_PATH ../rules

```
113 var WHITE_LIST_PATH c:\Snort\rules
114 var BLACK_LIST_PATH c:\Snort\rules
```

Figure 14: Setting up our White List and Black List files paths in Snort

8.Next we have to enable to log directory, so that we store logs in our log folder. Uncomment this line and set absolute path to log directory

Configure default log directory for snort to log to. For more information see snort -h command line options (-l)## config logdir:

```
186 config logdir: c:\Snort\log
```

Figure 15: Setting up Log Directory Path in Snort

9.Now we will set the path to dynamic preprocessors and dynamic engine

path to dynamic preprocessor libraries
dynamic preprocessor directory/usr/local/lib/snort_dynamicpreprocessor/

```
246 # path to dynamic preprocessor libraries
247 dynamicpreprocessor directory c:\Snort\lib\snort_dynamicpreprocessor
```

Figure 16: Setting up path to dynamic preprocessors and dynamic engine in Snort

10. We will do same thing for dynamic preprocessor engine

```
# path to base preprocessor enginedynamicengine
/usr/local/lib/snort_dynamicengine/libsf_engine.so
```

```
249  # path to base preprocessor engine
250  dynamicengine c:\Snort\lib\snort_dynamicengine\sf_engine.dll
```

Figure 17: Setting up the path to dynamic preprocessor engine in Snort

11. Now lets set our reputation preprocessors:

```
# path to dynamic rules libraries# dynamicdetection directory
/usr/local/lib/snort_dynamicrules
```

```
252  # path to dynamic rules libraries
253  # dynamicdetection directory /usr/local/lib/snort_dynamicrules
```

Figure 18: Path to dynamic rules libraries in Snort

12. Just comment out these lines as shown in figure 19 in doing so we are excluding packet normalization of different packets.

```
263  # Inline packet normalization. For more information, see README.normalize
264  # Does nothing in IDS mode
265  # preprocessor normalize_ip4
266  # preprocessor normalize_tcp: ips ecn stream
267  # preprocessor normalize_icmp4
268  # preprocessor normalize_ip6
269  # preprocessor normalize_icmp6
```

Figure 19: Commenting out packet normalization commands in Snort

13. Scroll down to the reputation preprocessors. We will just change the name of the files since white list , black list are not rules they are just the list of IP addresses labelled as black or white

```
# Reputation preprocessor. For more information see README.reputationpreprocessor
reputation: \memcap 500, \priority whitelist, \nested_ip inner, \whitelist
$WHITE_LIST_PATH/whitelist, \blacklist $BLACK_LIST_PATH\black.list
```

```
511      whitelist $WHITE_LIST_PATH/white.list, \
512      blacklist $BLACK_LIST_PATH\black.list
```

Figure 20: Whitelisting and Blacklisting IPs through the command as shown in figure

14. Converted back slashes to forward slashes in lines 546–651.

```
# site specific rules
include $RULE_PATH\local.rules

include $RULE_PATH\app-detect.rules
include $RULE_PATH\attack-responses.rules
include $RULE_PATH\backdoor.rules
include $RULE_PATH\bad-traffic.rules
include $RULE_PATH\blacklist.rules
include $RULE_PATH\botnet-cnc.rules
include $RULE_PATH\browser-chrome.rules
include $RULE_PATH\browser-firefox.rules
include $RULE_PATH\browser-ie.rules
include $RULE_PATH\browser-other.rules
include $RULE_PATH\browser-plugins.rules
include $RULE_PATH\browser-webkit.rules
include $RULE_PATH\chat.rules
include $RULE_PATH\content-replace.rules
include $RULE_PATH\ddos.rules
include $RULE_PATH\dns.rules
include $RULE_PATH\dos.rules
include $RULE_PATH\experimental.rules
include $RULE_PATH\exploit-kit.rules
include $RULE_PATH\exploit.rules
include $RULE_PATH\file-executable.rules
include $RULE_PATH\file-flash.rules
include $RULE_PATH\file-identify.rules
include $RULE_PATH\file-image.rules
include $RULE_PATH\file-multimedia.rules
include $RULE_PATH\file-office.rules
include $RULE_PATH\file-other.rules
include $RULE_PATH\file-pdf.rules
include $RULE_PATH\finger.rules
include $RULE_PATH\ftp.rules
include $RULE_PATH\icmp-info.rules
include $RULE_PATH\icmp.rules
```

Figure 21: Converted back slashes to forward slashes in specific lines in snort.conf file

```
include $RULE_PATH\rservices.rules
include $RULE_PATH\scada.rules
include $RULE_PATH\scan.rules
include $RULE_PATH\server-apache.rules
include $RULE_PATH\server-iis.rules
include $RULE_PATH\server-mail.rules
include $RULE_PATH\server-mssql.rules
include $RULE_PATH\server-mysql.rules
include $RULE_PATH\server-oracle.rules
include $RULE_PATH\server-other.rules
include $RULE_PATH\server-webapp.rules
include $RULE_PATH\shellcode.rules
include $RULE_PATH\smtp.rules
include $RULE_PATH\snmp.rules
include $RULE_PATH\specific-threats.rules
include $RULE_PATH\spyware-put.rules
include $RULE_PATH\sql.rules
include $RULE_PATH\telnet.rules
include $RULE_PATH\tftp.rules
include $RULE_PATH\virus.rules
include $RULE_PATH\voip.rules
include $RULE_PATH\web-activex.rules
include $RULE_PATH\web-attacks.rules
include $RULE_PATH\web-cgi.rules
include $RULE_PATH\web-client.rules
include $RULE_PATH\web-coldfusion.rules
include $RULE_PATH\web-frontpage.rules
include $RULE_PATH\web-iis.rules
include $RULE_PATH\web-misc.rules
include $RULE_PATH\web-php.rules
include $RULE_PATH\x11.rules
```

Figure 22: Converted back slashes to forward slashes in specific lines in snort.conf file

15. Again just convert forward slashes to backslashes and uncomment the lines below:

```
# decoder and preprocessor event rules# include
$PREPROC_RULE_PATH/preprocessor.rules#  include
$PREPROC_RULE_PATH/decoder.rules# include $PREPROC_RULE_PATH/sensitive-
data.rules
```

```

# decoder and preprocessor event rules
  include $PREPROC_RULE_PATH\preprocessor.rules
  include $PREPROC_RULE_PATH\decoder.rules
  include $PREPROC_RULE_PATH\sensitive-data.rules
```

Figure 23: Converted back slashes to forward slashes in specific lines and uncommenting specific lines in snort.conf file

16. Now we just need to verify the presence of this command at the bottom of **snort.conf** file.

```
688  # Event thresholding or suppression commands. See threshold.conf
689  include threshold.conf
```

Figure 24: verifying presence of "include threshold.conf" command in snort.conf file

17. Click on Save file and save all changes to save the configuration file (**snort.conf**).

18. Now recalling the **Step 13** white list, black list are not rules they are just the list of IP addresses labelled as black or white right now these files don't exist in our rule path which is why we have to create them manually, save them in this folder **C:\Snort\rules.**

- Go to Notepad++ and create new file.
- Comment it #White-listed IPs.
- Name the file white.list and save the file.

Figure 25: Creating White List IPs file

- Create another new file.
- Comment it #Black-listed IPs.
- Name the file black.list and save the file.

Figure 26: Creating Black List IPs file in Snort

19. Now we test snort again by running Command prompt as admin. To check if it's running fine after all the configurations.

20. We can also the check the wireless interface cards from which we will be using snort by using the command below we can see the list of our wireless interface cards through entering this command in command prompt.

Snort — W

```
C:\Snort\bin>Snort -W

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.20-WIN64 GRE (Build 82)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.11

Index   Physical Address        IP Address      Device Name     Description
-----   ----------------        ----------      -----------     -----------
    1   00:7E:56:6A:79:5E       169.254.101.80  \Device\NPF_{33B66889-AD6E-4531-B49F-3A2521078383}      Microsoft Wi-Fi Direct Virtual Adapter
    2   00:7E:56:6A:79:5E       169.254.134.247 \Device\NPF_{8F40D862-BD56-41A0-89E1-DC2E7C94EDFE}      Microsoft Hosted Network Virtual Adapter
    3   00:7E:56:6A:79:5E       169.254.226.223 \Device\NPF_{B8F6F3E9-3D8C-4E96-A80D-01ED7473E45E}      Realtek RTL8188ETV Wireless LAN 802.11n USB 2.0 Network Adapter
    4   00:00:00:00:00:00       0000:0000:0000:0000:0000:0000:0000:0000 \Device\NPF_Loopback    Adapter for loopback traffic capture
    5   00:E0:4C:78:A5:A5       169.254.119.36  \Device\NPF_{6E82CA38-E953-4A1F-82CF-246A5EB97EEF}      Realtek PCIe FE Family Controller
```

Figure 27: Test Running of Snort in Windows 10 after Configuration

RESULT:

Thus, the Intrusion Detection System (IDS) has been demonstrated by using the Snort intrusion detection tool successfully.

Ex. No. : 9	Explore Network Monitoring Tools
Date:	

AIM

To explore network monitoring tool.

ALGORITHM:

1. Start the program
2. Get Network Devices
3. Select a Network Interface
4. Open the Selected Interface
5. Start Packet Capture
6. Stop the program

PROGRAM:

```
import jpcap.*;
importjpcap.p acket.*;
publicclassNetworkMonitor
{
publicstaticvoidmain(String[]args)throwsException
{
NetworkInterface[]devices= JpcapCaptor.getDeviceList();
if (devices.length == 0)
{
System.out.println("Nonetworkinterfacefound.Makesureyouhavetherequiredpermissions.");
return;
}
intselectedDeviceIndex=0;
NetworkInterfaceselectedDevice = devices[selectedDeviceIndex];
JpcapCaptorcaptor =JpcapCaptor.openDevice(selectedDevice,2000,true,20);
System.out.println("Monitoring " + selectedDevice.name + "...");
while(true)
{
Packetpacket=captor.getPacket();if (packet!=null)
```

```
{
System.out.println(packet);} }}}
```

OUTPUT:

```
Ethernet packet (source MAC: 00:11:22:33:44:55, destination MAC: AA:BB:CC:DD:EE:FF)
IPv4 packet (source IP: 192.168.1.2, destination IP: 8.8.8.8, protocol: TCP)
TCP packet (source port: 12345, destination port: 80, flags: SYN)
Payload data: Hello, this is a sample packet!
```


RESULT:

Thus, the program for network monitoring tools has been executed and verified successfully.

Ex. No. : 10	Study to Configure Firewall and VPN
Date:	

AIM:

To study configure firewall, VPN and Create and Configure the Network.

PROCEDURE:

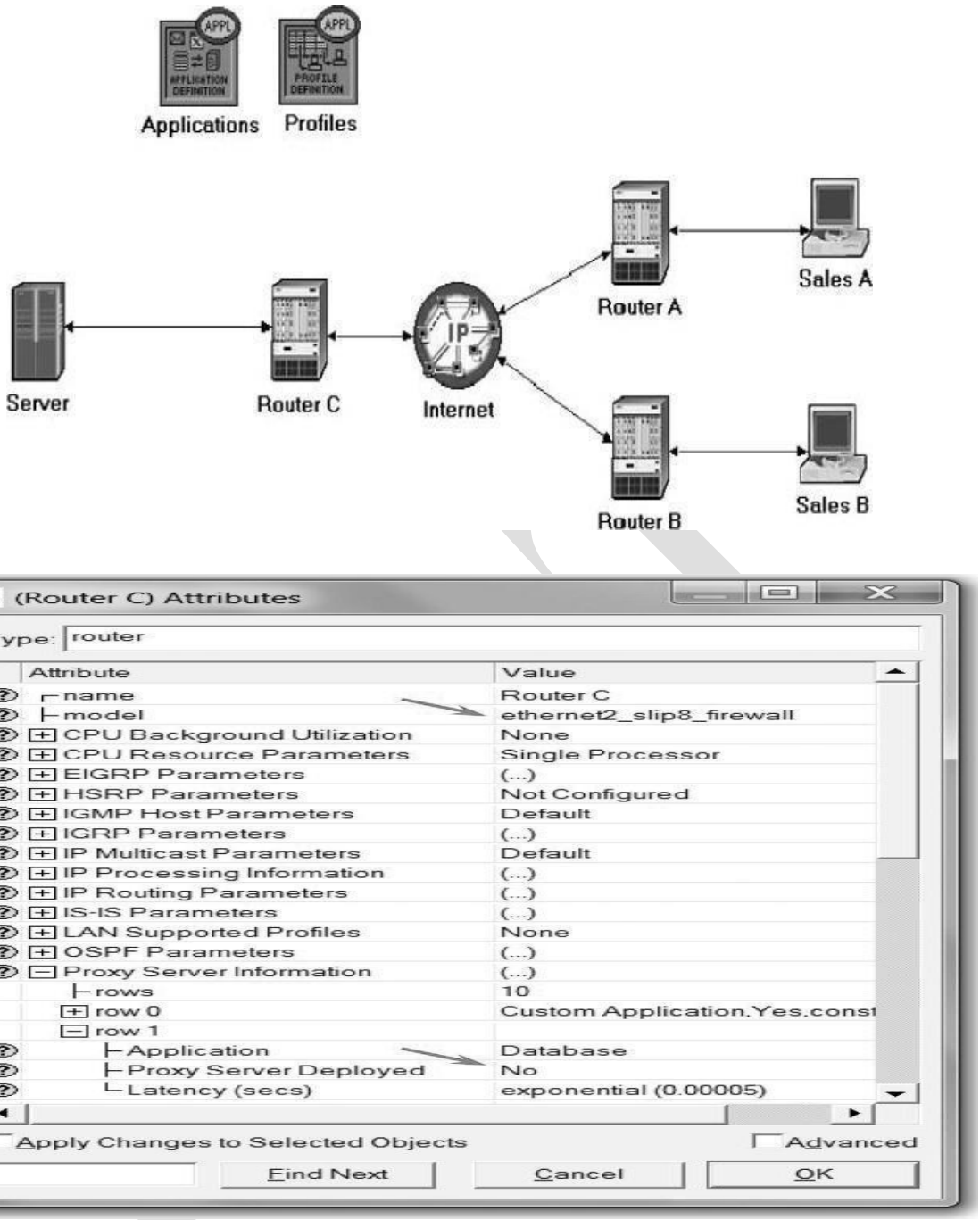
Initialize the Network:

1. Open the Object Palette dialog box by clicking. Make sure that the internet tool box item is selected from the pull-down menu on the object palette.
2. Add the following objects from the palette to the project workspace (see the following figure for placement): Application Config, Profile Config, an ip32_cloud, one ppp server, three ethernet4_slip8_gtwy routers, and two ppp wkstn hosts.
3. Rename the objects you added and connect the musing PPP_DS1 links, as shown.

The Firewall Scenario

In the network we just created, the Sales Person profile allows both sales sites to access applications such as database access, email, and Web browsing from the server (check the Profile Configuration of the Profiles node). Assume that we need to protect the database in the server from external access, including the salespeople. One way to do that is to replace **RouterC with a firewall as follows:**

1. Select Duplicate Scenario from the Scenarios menu and name its Firewall. Click OK.
2. In the new scenario, right-click on RouterC· Edit Attributes.
3. Assign ethernet2 _slip 8_firewall to the model attribute.
4. Expand the hierarchy of the Proxy Server Information attribute· Expand the row1, which is for the database application hierarchy· Assign Not the Proxy Server Deployed attribute as shown. Click OK, and Save your project.

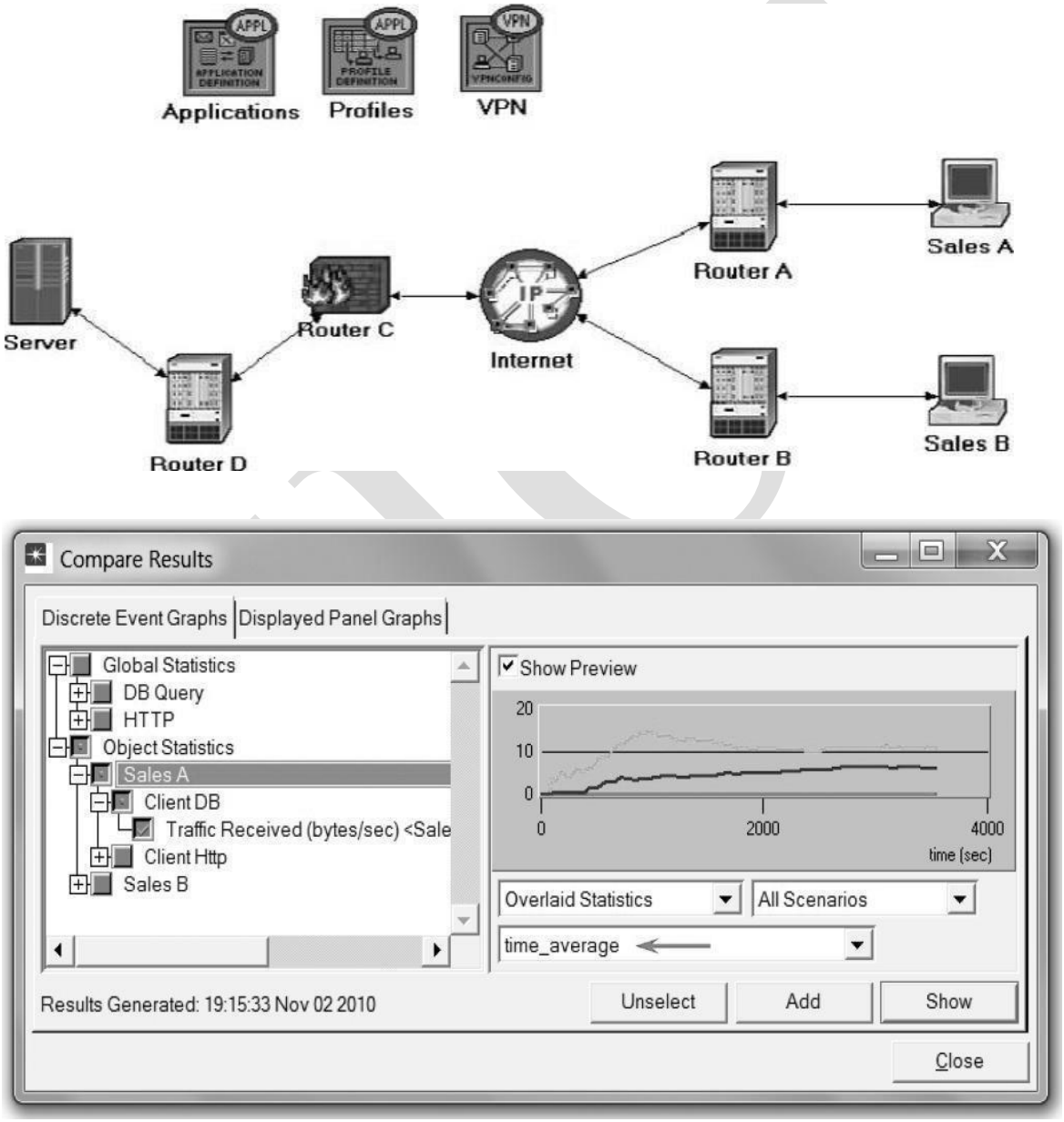


The Firewall VPN Scenario

In the Firewall scenario, we protected the databases in the server from "any" external access using a firewall router. Assume that we want to allow the people in the Sales A site to have access to the databases in the server. Because the firewall filters all database-related traffic regard less of the source of the traffic, we need to consider the VPN solution. A virtual tunnel can be used by Sales A to send database requests to the server. The firewall will not filter the traffic created by Sales A because the IP packets in the tunnel will be encapsulated inside an IP datagram.

1. While you are in the Firewall scenario, select Duplicate Scenario from the Scenarios menu and give it the name Firewall_VPN· Click OK.

2. Remove the link between RouterC and the Server.
3. Open the Object Palette dialog box by clicking. Make sure that the internet tool box is selected from the pull- down menu on the object palette.
   a. Add to the project work space one ethernet4_ slip8_ gtwyandone IPVPN Config(seethe following figure for placement).
   b. From the Object palette, use two PPP_DS1 links to connect the new router to the RouterC (the firewall) and to the Server, as shown in the following figure.
   c. Close the Object Palette dialog box
4. Rename the IP VPN Config object to VPN.
5. Rename the new router to Router Das shown in the following fig.



RESULT:

Thus, the Firewall and VPN configuration is completed successfully.