Ex.No. : 1 (a) **DATA ENCRYPTION STANDARD (DES)**
Date :

AIM:

To apply 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 (/).
   - Algorithm name
   - Mode (optional)
   - 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.

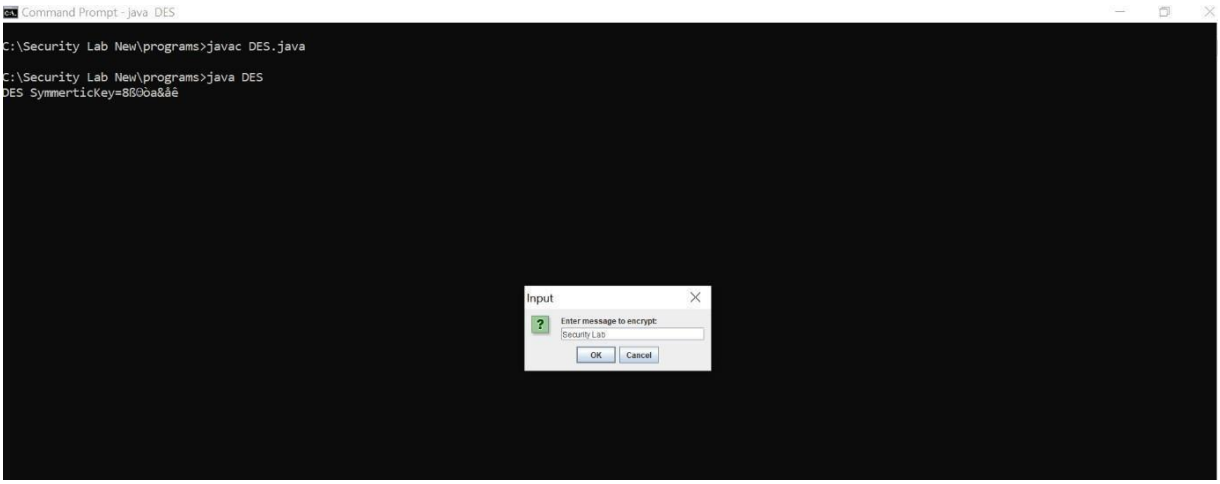
PROGRAM:

```
import javax.swing.*;
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Random;
class DES
{
        byte[] skey=new byte[1000];
        String skeystring;
        static byte[] raw;
        String inputmessage,encryptedata,decryptedmessage;
        public DES()
        {
                try
                {
                        generatesymmetrickey();
                        inputmessage=JOptionPane.showInputDialog(null,"Enter message to
                        encrypt:");
                        byte[] ibyte =inputmessage.getBytes();
                        byte[] ebyte=encrypt(raw, ibyte);
                        String encrypteddata=new String(ebyte);
                        System.out.println("Encrypted message:"+encrypteddata);
                        JOptionPane.showMessageDialog(null,"Encrypted
                        Data"+"\n"+encrypteddata);
                        byte[] dbyte=decrypt(raw,ebyte);
                        String decryptedmessage=new String(dbyte);
                        System.out.println("Decrypted message:"+decryptedmessage);
```

```
                    JOptionPane.showMessageDialog(null,"Decrypted Data
                    "+"\n"+decryptedmessage);
                }
                catch(Exception e)
                {
                    System.out.println(e);
                }
        }
    void generatesymmetrickey()
    {
         try
        {
                Random r = new Random();
                int num=r.nextInt(10000);
                String knum=String.valueOf(num);
                byte[] knumb=knum.getBytes();
                skey=getRawKey(knumb);
                skeystring=new String(skey);
                System.out.println("DES
                SymmerticKey="+skeystring);
        }

        catch(Exception e)
        {
                System.out.println(e);
        }
    }

    private static byte[] getRawKey(byte[] seed) throws Exception
    {
        KeyGenerator kgen=KeyGenerator.getInstance("DES ");
        SecureRandom sr =SecureRandom.getInstance("SHA1PRNG");
        sr.setSeed(seed);
        kgen.init(56,sr);
        SecretKey skey=kgen.generateKey();
        raw=skey.getEncoded();
        return raw;
    }

    private static byte[] encrypt(byte[] raw,byte[] clear) throws Exception
    {
        SecretKey seckey = new SecretKeySpec(raw, "DES");
        Cipher cipher = Cipher.getInstance("DES");
        cipher.init(Cipher.ENCRYPT_MODE,seckey);
        byte[] encrypted=cipher.doFinal(clear);
        return encrypted;
    }

    private static byte[] decrypt(byte[] raw,byte[] encrypted) throws Exception
    {
        SecretKey seckey = new SecretKeySpec(raw, "DES");
```

```
        Cipher cipher = Cipher.getInstance("DES");
        cipher.init(Cipher.DECRYPT_MODE,seckey);
        byte[] decrypted = cipher.doFinal(encrypted);
        return decrypted;
 }
 public static void main(String args[])
{
        DES des=new DES();
}

}
```

OUTPUT:



RESULT:

Thus the java program for applying Data Encryption Standard (DES) Algorithm for a practical application of User Message Encryption is written and executed successfully.

Ex.No. : 1(b) **AES ALGORITHM**
Date :

AIM:

To apply 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) {

        System.out.println("Enter the secret key: ");
        String secretKey = System.console().readLine();

        System.out.println("Enter the original URL: ");
        String originalString = System.console().readLine();

        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:

```
C:\Security Lab New\programs>java AES
Enter the secret key:
annaUniversity
Enter the original URL:
www.annauniv.edu
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 applying Advanced Encryption Standard (AES) Algorithm for a practical application of URL encryption is written and executed successfully.

Ex.No. : 2(a) **RSA ALGORITHM**
Date :

AIM:

To implement a RSA algorithm using HTML and Javascript.

ALGORITHM:

1. Choose two prime number p and q.
2. Compute the value of n and t.
3. Find the value of public key e.
4. Compute the value of private key d.
5. Do the encryption and decryption
   a. Encryption is given as,
      $c = t^e \bmod n$
   b. Decryption is given as,
      $t = c^d \bmod n$

PROGRAM:

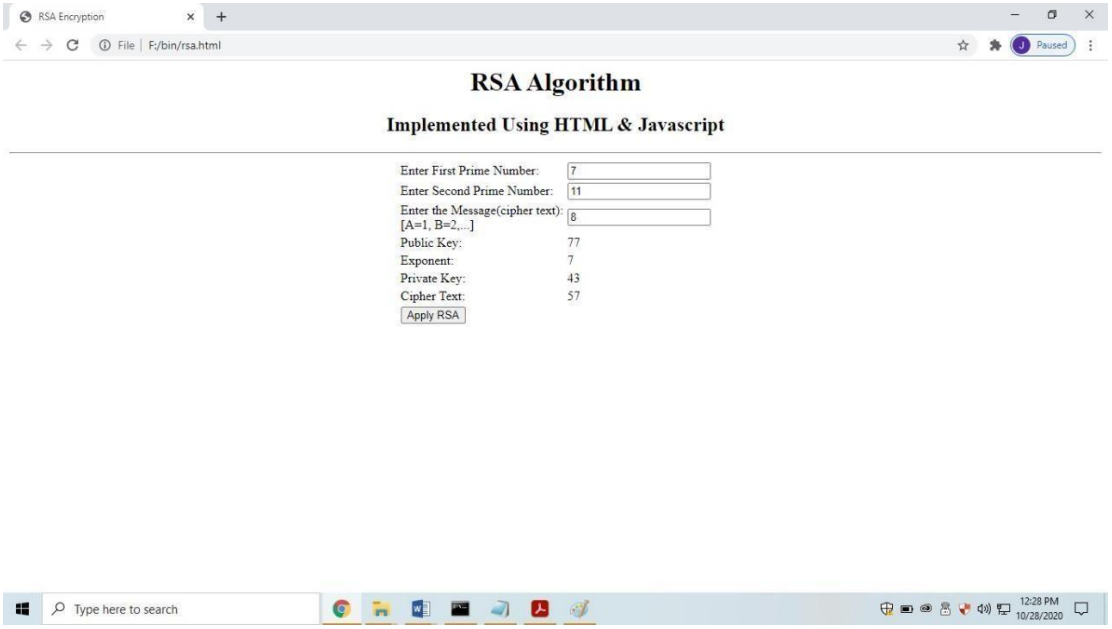
rsa.html

```
<html>
<head>
  <title>RSA Encryption</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
  <center>
    <h1>RSA Algorithm</h1>
    <h2>Implemented Using HTML & Javascript</h2>
    <hr>
    <table>
      <tr>
        <td>Enter First Prime Number:</td>
        <td><input type="number" value="53" id="p"></td>
      </tr>
      <tr>
        <td>Enter Second Prime Number:</td>
        <td><input type="number" value="59" id="q"></p> </td>
      </tr>
      <tr>
        <td>Enter the Message(cipher text):<br>[A=1, B=2,...]</td>
        <td><input type="number" value="89" id="msg"></p> </td>
      </tr>
      <tr>
        <td>Public Key:</td>
        <td><p id="publickey"></p> </td>
      </tr>
      <tr>
        <td>Exponent:</td>
        <td><p id="exponent"></p> </td>
      </tr>
```

```
            <tr>
                <td>Private Key:</td>
                <td><p id="privatekey"></p></td>
            </tr>
            <tr>
                <td>Cipher Text:</td>
                <td><p id="ciphertext"></p> </td>
            </tr>
            <tr>
                <td><button onclick="RSA();">Apply RSA</button></td>
            </tr>
        </table> </center>
</body>
<script type="text/javascript">

function RSA()
{
        var gcd, p, q, no, n, t, e, i, x;
        gcd = function (a, b) { return (!b) ? a : gcd(b, a % b); };
        p = document.getElementById('p').value;
        q = document.getElementById('q').value;
        no = document.getElementById('msg').value;
        n = p * q;
        t = (p - 1) * (q - 1);
        for (e = 2; e < t; e++)
        {
            if (gcd(e, t) == 1)
            {
                break;
            }
        }
        for (i = 0; i < 10; i++)
        {
            x = 1 + i * t
            if (x % e == 0)
            {
                d = x / e;
                break;
            }
        }
        ctt = Math.pow(no, e).toFixed(0);
        ct = ctt % n;
        dtt = Math.pow(ct, d).toFixed(0);
        dt = dtt % n;
        document.getElementById('publickey').innerHTML = n;
        document.getElementById('exponent').innerHTML = e;
        document.getElementById('privatekey').innerHTML = d;
        document.getElementById('ciphertext').innerHTML = ct;
    }
</script>
```

</html>

OUTPUT:



RESULT:

Thus the RSA algorithm was implemented using HTML and Javascript and executed successfully.

Ex.No. : 2(b) DIFFIE-HELLMAN KEY EXCHANGE ALGORITHM
Date :

AIM:

To implement a Diffie-Hellman Key Exchange algorithm.

ALGORITHM:

1. Sender and receiver publicly agree to use a modulus p and base g which is a primitive root modulo p.
2. Sender chooses a secret integer x then sends Bob R1 = g^x mod p
3. Receiver chooses a secret integer y, then sends Alice R2 = g^y mod p
4. Sender computes k1 = B^x mod p
5. Receiver computes k2 = A^y mod p
6. Sender and Receiver now share a secret key.

PROGRAM:

```
import java.io.*;
import java.math.BigInteger;
class dh
{
public static void main(String[]args)throws IOException
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter prime number:");
BigInteger p=new BigInteger(br.readLine());

System.out.print("Enter primitive root of "+p+":");
BigInteger g=new BigInteger(br.readLine());

System.out.println("Enter value for x less than "+p+":");
BigInteger  x=new BigInteger(br.readLine());
BigInteger R1=g.modPow(x,p);
System.out.println("R1="+R1);

System.out.print("Enter value for y less than "+p+":");
BigInteger y=new BigInteger(br.readLine());
BigInteger R2=g.modPow(y,p);
System.out.println("R2="+R2);

BigInteger k1=R2.modPow(x,p);
System.out.println("Key calculated at Sender's side:"+k1);
BigInteger k2=R1.modPow(y,p);
System.out.println("Key calculated at Receiver's side:"+k2);
System.out.println("Diffie-Hellman secret key was calculated.");
}
}
```

OUTPUT

```
C:\Security Lab New\programs>javac dh.java

C:\Security Lab New\programs>java dh
Enter prime number:
11
Enter primitive root of 11:7
Enter value for x less than 11:
3
R1=2
Enter value for y less than 11:6
R2=4
Key calculated at Sender's side:9
Key calculated at Receiver's side:9
Diffie-Hellman secret key was calculated.
```

RESULT:

Thus the Diffie-Hellman key exchange algorithm was implemented and executed successfully.

Ex.No. : 3 **DIGITAL SIGNATURE SCHEME**
Date :

AIM:

To implement the signature scheme - Digital Signature Standard.

ALGORITHM:

1. Declare the class and required variables.
2. Create the object for the class in the main program.
3. Access the member functions using the objects.
4. Implement the SIGNATURE SCHEME - Digital Signature Standard.
5. It uses a hash function.
6. The hash code is provided as input to a signature function along with a random number K generated for the particular signature.
7. The signature function also depends on the sender,,s private key.
8. The signature consists of two components.
9. The hash code of the incoming message is generated.
10. The hash code and signature are given as input to a verification function.

PROGRAM:

```
import java.util.*;
import java.math.BigInteger;
class dsaAlg {
final static BigInteger one = new BigInteger("1");
final static BigInteger zero = new BigInteger("0");
public static BigInteger getNextPrime(String ans)
{
BigInteger test = new BigInteger(ans);
while (!test.isProbablePrime(99))
e:
{
test = test.add(one);
}
return test;
}
public static BigInteger findQ(BigInteger n)
{
BigInteger start = new BigInteger("2");
while (!n.isProbablePrime(99))
{
while (!((n.mod(start)).equals(zero)))
{
start = start.add(one);
}
n = n.divide(start);
}
return n;
}
```

```
public static BigInteger getGen(BigInteger p, BigInteger q,
Random r)
{
BigInteger h = new BigInteger(p.bitLength(), r);
h = h.mod(p);
return h.modPow((p.subtract(one)).divide(q), p);
}
public static void main (String[] args) throws
java.lang.Exception
{
Random randObj = new Random();
BigInteger p = getNextPrime("10600"); /* approximate
prime */
BigInteger q = findQ(p.subtract(one));
BigInteger g = getGen(p,q,randObj);
System.out.println(" \n simulation of Digital Signature Algorithm \n");
System.out.println(" \n global public key components are:\n");
System.out.println("\np is: " + p);
System.out.println("\nq is: " + q);
System.out.println("\ng is: " + g);
BigInteger x = new BigInteger(q.bitLength(), randObj);
x = x.mod(q);
BigInteger y = g.modPow(x,p);
BigInteger k = new BigInteger(q.bitLength(), randObj);
k = k.mod(q);
BigInteger r = (g.modPow(k,p)).mod(q);
BigInteger hashVal = new BigInteger(p.bitLength(),
randObj);
BigInteger kInv = k.modInverse(q);
BigInteger s = kInv.multiply(hashVal.add(x.multiply(r)));
s = s.mod(q);
System.out.println("\nsecret information are:\n");
System.out.println("x (private) is:" + x);
System.out.println("k (secret) is: " + k);
System.out.println("y (public) is: " + y);
System.out.println("h (rndhash) is: " + hashVal);
System.out.println("\n generating digital signature:\n");
System.out.println("r is : " + r);
System.out.println("s is : " + s);
BigInteger w = s.modInverse(q);
BigInteger u1 = (hashVal.multiply(w)).mod(q);
BigInteger u2 = (r.multiply(w)).mod(q);
BigInteger v = (g.modPow(u1,p)).multiply(y.modPow(u2,p));
v = (v.mod(p)).mod(q);
System.out.println("\nverifying digital signature (checkpoints)\n:");
System.out.println("w is : " + w);
System.out.println("u1 is : " + u1);
System.out.println("u2 is : " + u2);
System.out.println("v is : " + v);
if (v.equals(r))
```

```
{
System.out.println("\nsuccess: digital signature is verified!\n " + r);
}
else
{
System.out.println("\n error: incorrect digital signature\n ");
}
}
}
```

OUTPUT:

```
C:\Security Lab New\programs>javac dsaAlg.java
C:\Security Lab New\programs>java dsaAlg
simulation of Digital Signature Algorithm
global public key components are:
p is: 10601
q is: 53
g is: 6089
secret information are:
x (private) is:6k
(secret) is: 3
y (public) is: 1356
h (rndhash) is: 12619 generating
digital signature:
r is : 2s is :
41
verifying digital signature (checkpoints):
w is : 22 u1 is
: 4 u2 is : 44v
is : 2
success: digital signature is verified!2
```

RESULT:

Thus the Digital Signature Standard Signature Scheme has been implemented and executed successfully.

Ex.No. 4:Installation of Wireshark,tcpdump,etc,observe the data transferin client server communication using TCP/UDP and identify the TCP/UDP datagram.

The purpose of this document is to introduce the packet sniffer Wireshark. Wireshark would be used for the lab experiments. This document introduces the basic operation of a packet sniffer, installation, and a test run of Wireshark.

What is Wireshark*?*

Wireshark 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 examiningwhat's happening inside an electric cable (but at a higher level, of course).

Wireshark, a network analysis tool formerly known as Ethereal, captures packets in real time and display them in human-readable format. Wireshark includes filters, color-coding, and other features that let you dig deep into network traffic and inspect individual packets.

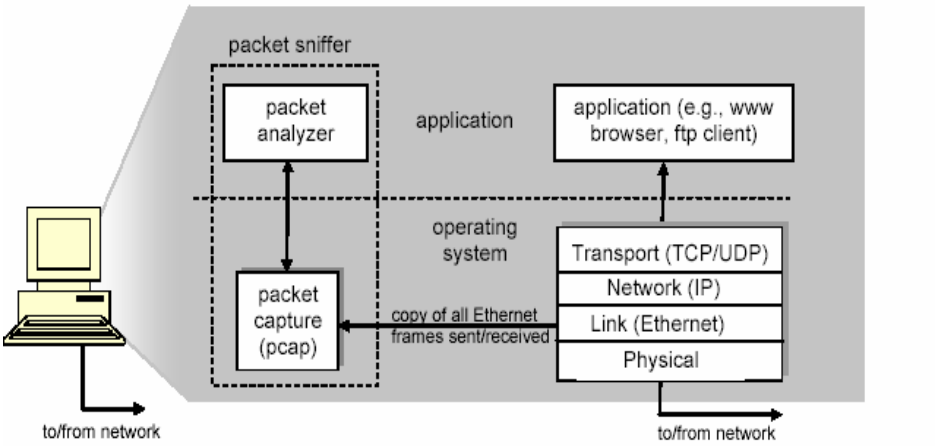


Figure 1: Packet sniffer structure

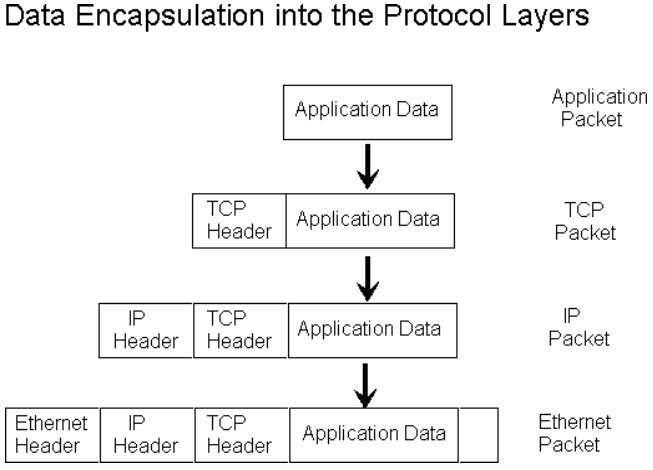


Figure 1b: Protocol Data Encapsulation

PACKET SNIFFER

The basic tool for observing the messages exchanged as packets on the network using a variety of protocols is called a **packet sniffer**. As the name suggests, a packet sniffer captures ("sniffs") messages being sent/received from/by your computer; it will also typically store and/or display the contents of the various protocol fields in these captured messages. A packet sniffer itself is passive. It observes messages being sent and received by applications and protocols running on your computer, but never sends packets itself. Similarly, received packets are never explicitly addressed to the packet sniffer. Instead, a packet sniffer receivesa *copy* of packets that are sent / received from/by application and protocols executing on your machine.

Figure 1 shows the structure of a packet sniffer. At the right of Figure 1 are the protocols (in this case, Internet protocols) and applications (such as a web browser or ftp client) that normally run on your computer. The overall packet structure. The packet sniffer, shown within the dashed rectangle in Figure 1 is an addition totheusual software in your computer and consists of two parts. The **packet capture library** receives a copy of every link-layer frame that is sent from or received by your computer. Messages exchanged by higher layer protocols such as HTTP, FTP, TCP, UDP, DNS, or IP all are eventually encapsulated in link-layer frames that are transmitted over physical media such as an Ethernet cable. **In Figure 1,** the assumed physical media is an Ethernet, and so all upper-layer protocols are eventually encapsulated within an Ethernet frame. Capturing all link-layer frames thus gives you access to all messages sent/received from/by all protocols andapplications executing inyour computer.The secondcomponent of a packet sniffer is the **packet analyzer**, which displays the contents of all fields within a protocol message. In order to do so, the packet analyzer must "understand" the structure of all messages exchanged by protocols. For example, suppose we are interested in displaying the various fields in messages exchanged by the HTTP protocol in Figure 1.

The packet analyzer understands the format of Ethernet frames, and so can identify the IP datagram within an Ethernet fram. It also understands the IP datagram format, so that it can extract the TCP segmentwithin the IP datagram. Finally, it understands the TCP segment structure, so it can extract the HTTP message contained in the TCP segment. Finally, it understands the HTTP protocol and so, for example, knows that the first bytes of an HTTP message will contain the string "GET," "POST," or"HEAD".

Getting Wireshark

Wireshark can be installed in any operating system, just go https://www.wireshark.org/download.html

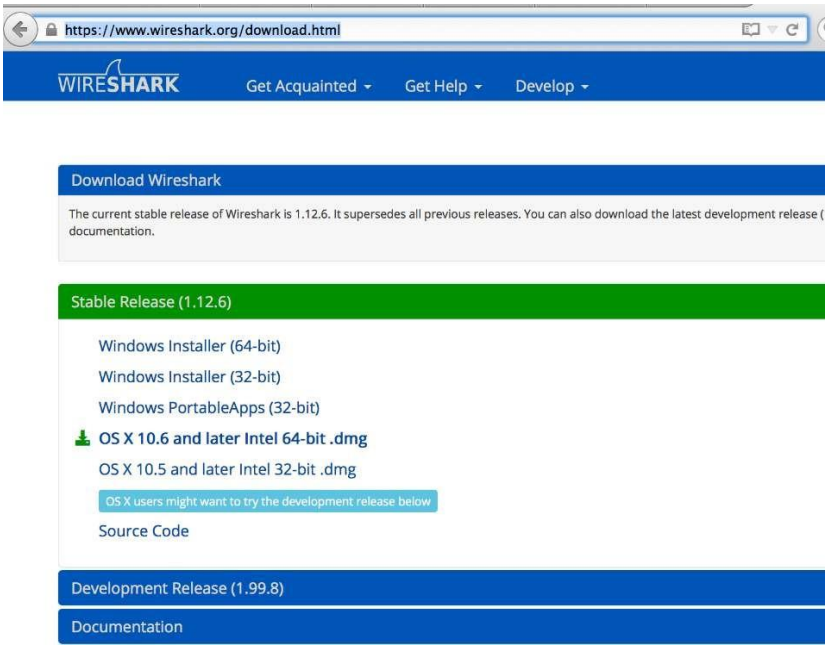


Figure 2: Wireshark Download Page.

You can download the version of wireshark that matches your operating system.

Running Wireshark

When you run the Wireshark program, the Wireshark graphical user interface shown in Figure 2 will be displayed. Initially, no data will be displayed in the various windows.

WIRESHARK USER INTERFACE

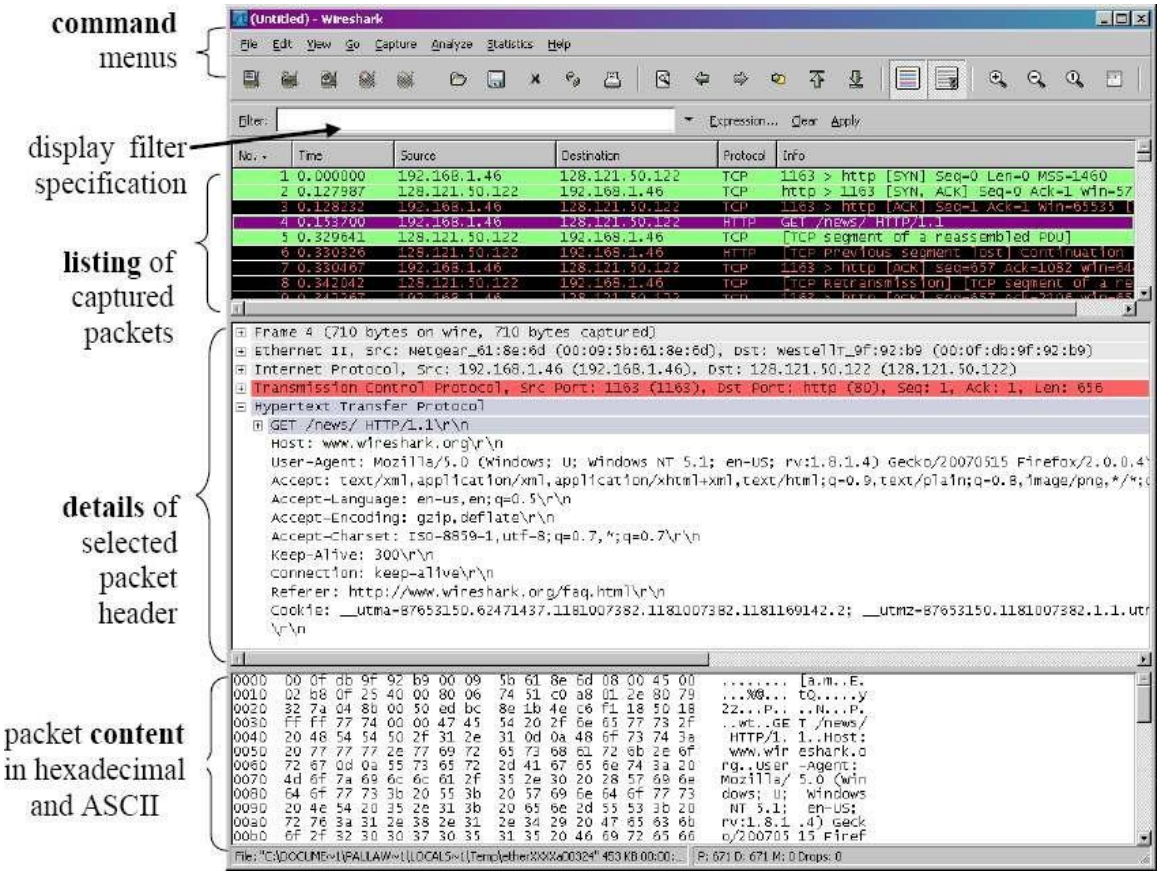


Figure 4a: Wireshark Graphical User Interface for Windows OS.

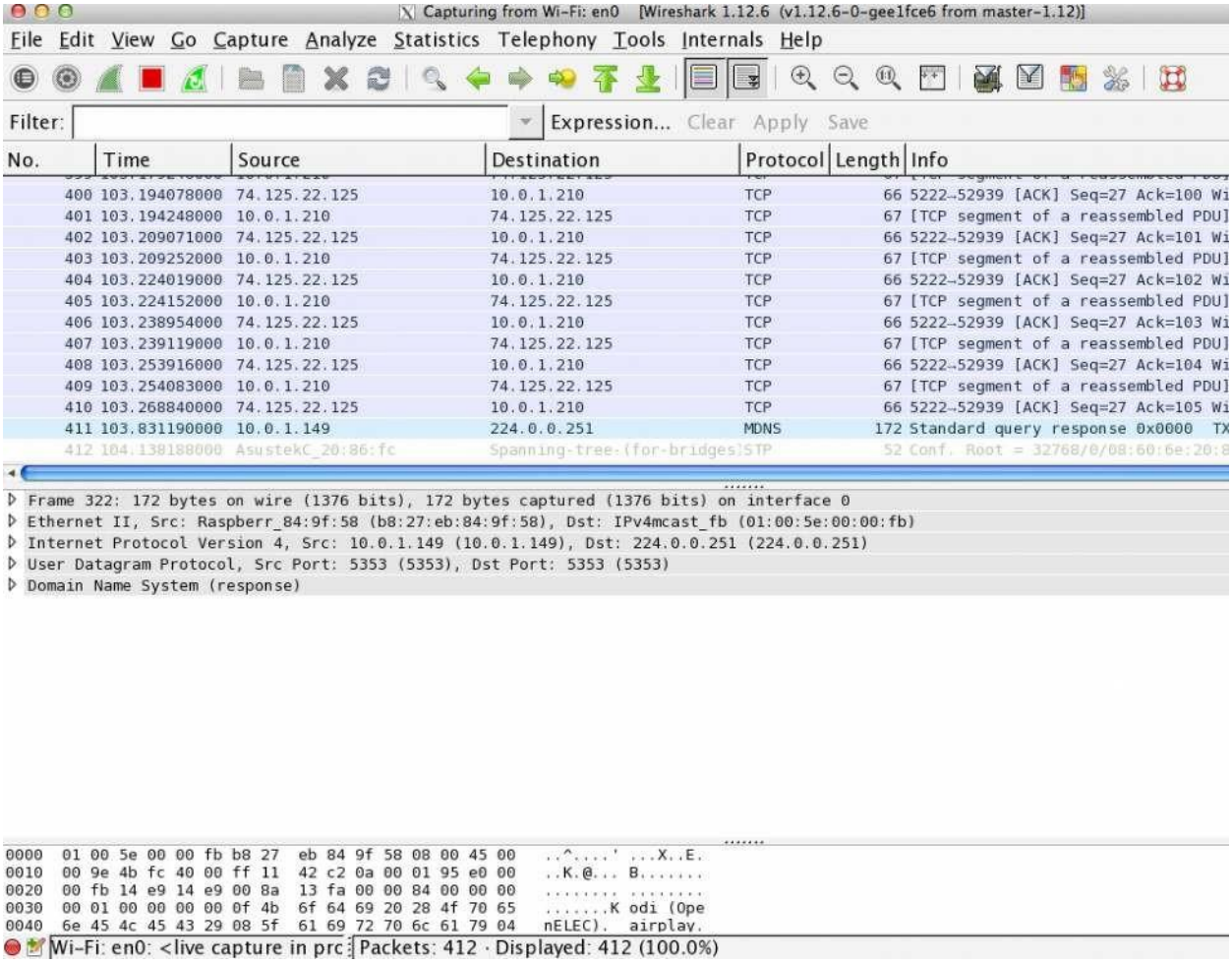


Figure 4b: Wireshark Graphical User Interface for Mac OSX.

The Wireshark interface has five major components:

- The command **menus are** standard pulldown menus located at the top window. Of interest to us now are the File and Capture menus. The File menu allows you to save captured packet data or open a file containing previously captured packet data and exit the Wireshark application. The Capture menu allows you to begin packetcapture.

- The **packet-listing window** displays a one-line summary for each packetcaptured, including the packet number (assigned by Wireshark; this is *not* a packet number contained in any protocol's header), the time at which the packet was captured, the packet's source and destination addresses, the protocol type, and protocol-specific information contained in the packet. The packet listing can be sorted according to any of these categories by clicking on acolumnname. The protocol type field lists the highest-level protocol that sent or received this packet, i.e., the protocol that is the source or ultimate sink for this packet.

- The packet-**header details window**provides details about the packet selected (highlighted) in the packet-listing window. (To select a packet in the packet-listing window, place the cursor over the packet's one-line summary in the packet-listing window and click with the left mouse button.). These details include information about the Ethernet frame and IP datagram that contains this packet. The amount of Ethernet and IP-layer detail displayed can be expanded or minimized by clicking on the right-pointing or down-pointing arrowhead to the left of the Ethernet frame or IP datagram line in the packet details window. If the packet has been carried over TCP or UDP, TCP or UDP details will also

be displayed, which can similarly be expanded or minimized. Finally, details about the highest-level protocol that sent or received this packet are alsoprovided.

- The **packet-contents window** displays the entire contents of thecaptured frame, in both ASCII and hexadecimalformat.

- Towards the top of the Wireshark graphical user interface, is the **packet display filter field,** into which a protocol name or other information can be entered in order to filter the information displayedin the packet-listing window (and hence the packet-header and packet-contents windows). In the example below, we'll use the packet-display filter field to have Wireshark hide (not display) packets except those that correspond to HTTPmessages.

Capturing Packets

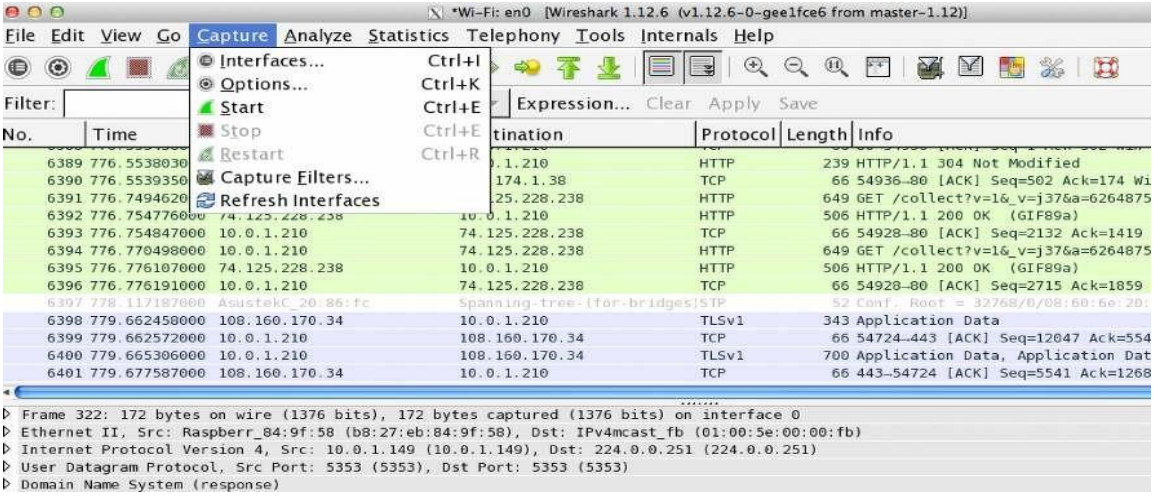
After downloading and installing Wireshark, you can launch it and click the name of an interface under Interface List to start capturing packets on that interface. For example, if you want to capture traffic on the wireless network, click your wireless interface. You can configure advanced features by clicking Capture Options, but this isn't necessary fornow.

Test Run

1. Start up your favorite webbrowser.

2. Start up the Wireshark software. You will initially see a window similar tothat shown in **Figure 3. You need to select an interface and pressStart.**

2. After your browser has displayed the http://www.gmu.edu page, stop Wireshark packet capture by selecting stop in the Wireshark capture window. This will cause the Wireshark capture window to disappear and the main Wireshark window to display all packets captured since you began packet capture see image below:

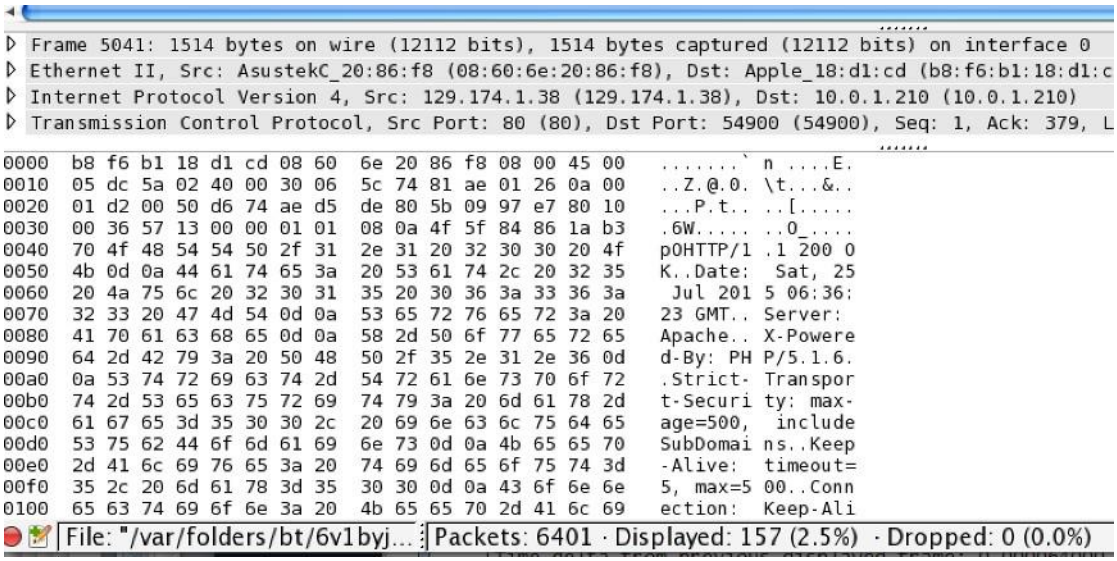


4. Color Coding: You'll probably see packets highlighted in green, blue, and black. Wireshark uses

colors to help you identify the types of traffic at a glance. By default, green is TCP traffic, dark blue is DNS traffic, light blue is UDP traffic, and black identifies TCP packets with problems — for example, they could have been deliveredout-of-order.

Inspecting Packets

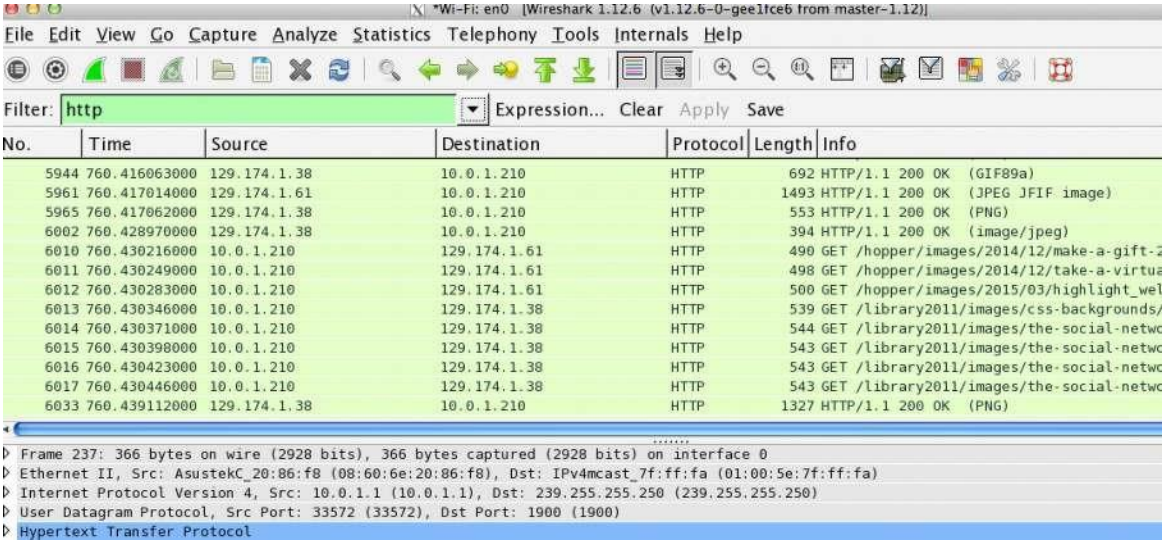
To inspect packets, click on one of the packets and go to the bottom pane:



Notice that although there is a lot of interesting information, the view of the packet is not very easy to read.

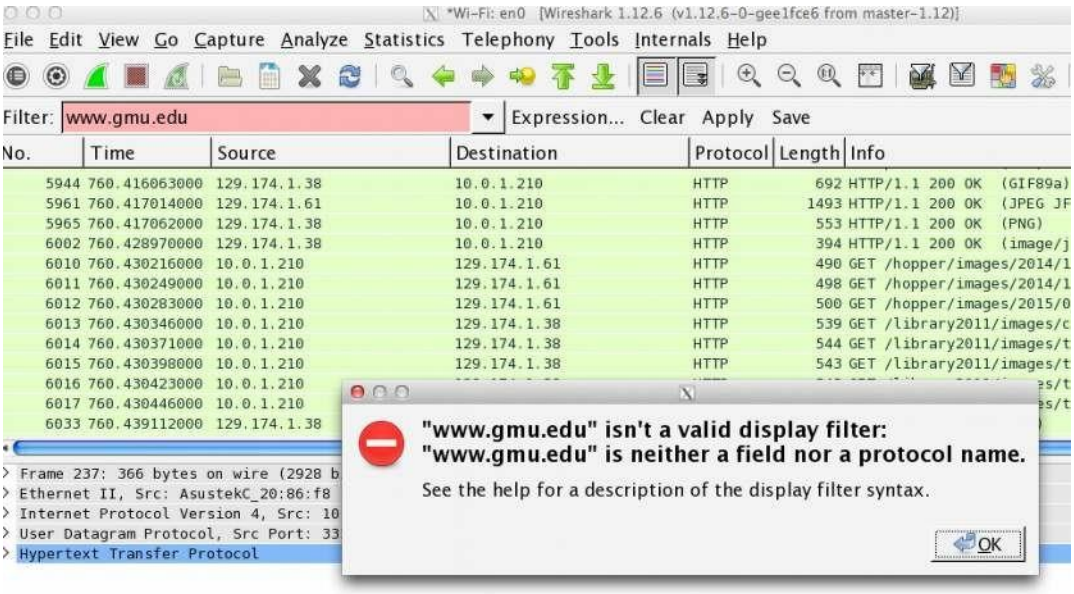
Inspecting Packet Flows (Network Connections)

5. You now have live packet data that contains all protocol messages exchanged between your computer and other network entities! However, as you will notice the HTTP messages are not clearly shown because there are many other packets included in the packet capture. Even though the only action you took was to open your browser, there are many other programs in your computer that communicate via the network in the background. To filter the connections to the ones we want to focus on, we have to use the filtering functionality of Wireshark by typing "http" in the filtering field as shown below:

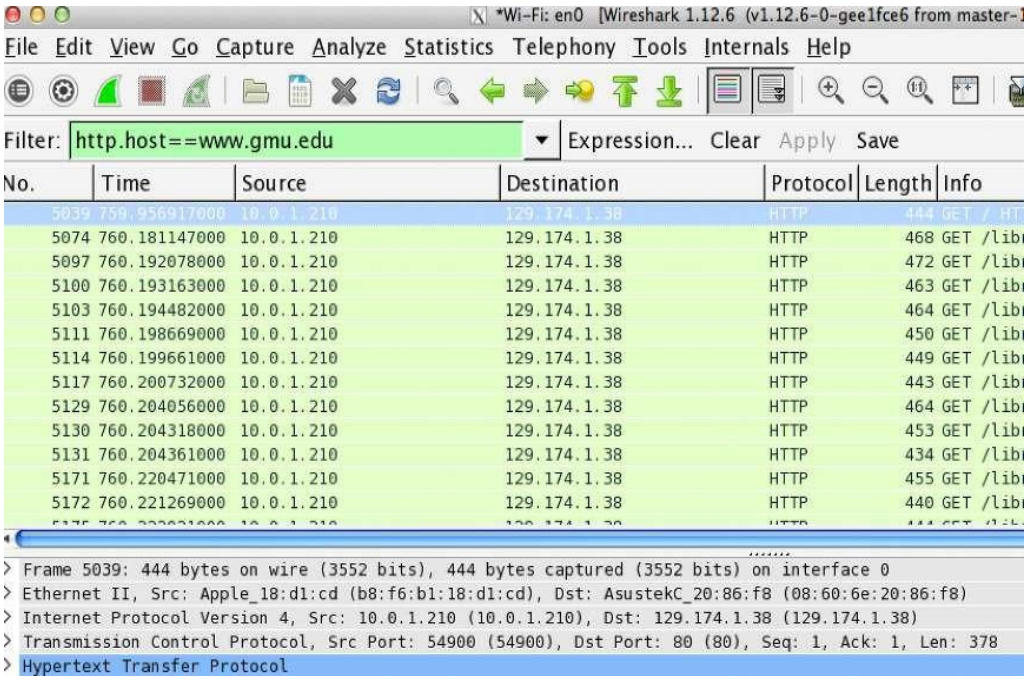


Notice that we now view only the packets that are of protocol HTTP. However, we also still do not have the exact communication we want to focus on because using HTTP as a filter is not descriptive enough to allow us to find our connection to http://www.gmu.edu. We need to be more precise if we want to capture the correct set of packets.

6. One potential filter that comes to mind is to type the destination host (www.gmu.edu) directly in the filter area. Unfortunately, this will end up bringing up an error (see screenshot below) and will not work because Wireshark does not have the ability to discern which protocol fields you need tomatch.



7. To further filter packets in Wireshark, we need to use a more precise filter. By setting the http.host==www.gmu.edu, we are restricting the view to packets that have as an http host the www.gmu.eduwebsite. Notice that we need two equal signs to perform the match "==" not justone!



Ex.No. 5 Perform an experiment to sniff traffic using ARPPoisoning

Address Resolution Protocol (ARP) is used to convert IP address to physical address.The host sends an ARP broadcast on the network, and the recipient computer responds with its MAC address.The resolved IP/MAC address is then used to communicate. ARP poisoning is sending fake MAC addresses to the switch so that it can associate the fake MAC addresses with the IP address of a computer on a network and hijack the traffic.

ARP Poisoning Countermeasures

Static ARP entries: these can be defined in the local ARP cache and the switch configured to ignore all auto ARP reply packets. The disadvantage of this method is, it's difficult to maintain on largenetworks. IP/MAC address mapping has to be distributed to all the computers on the network.

ARP poisoning detection software: these systems can be used to cross check the IP/MAC address resolution and certify them if they are authenticated. Uncertified IP/MAC address resolutions can then be blocked.

Computers communicate using networks. These networks could be on a local area network LAN or exposed to the internet. Network Sniffers are programs that capture low-level package data that is transmitted over a network. An attacker can analyze this information to discover valuable information such as user ids and passwords.

In this article, we will introduce you to common network sniffing techniques and tools used to sniff networks.

What is network sniffing?

Computers communicate by broadcasting messages on a network using IP addresses. Once a message has been sent on a network, the recipient computer with the matching IP address responds with its MAC address.

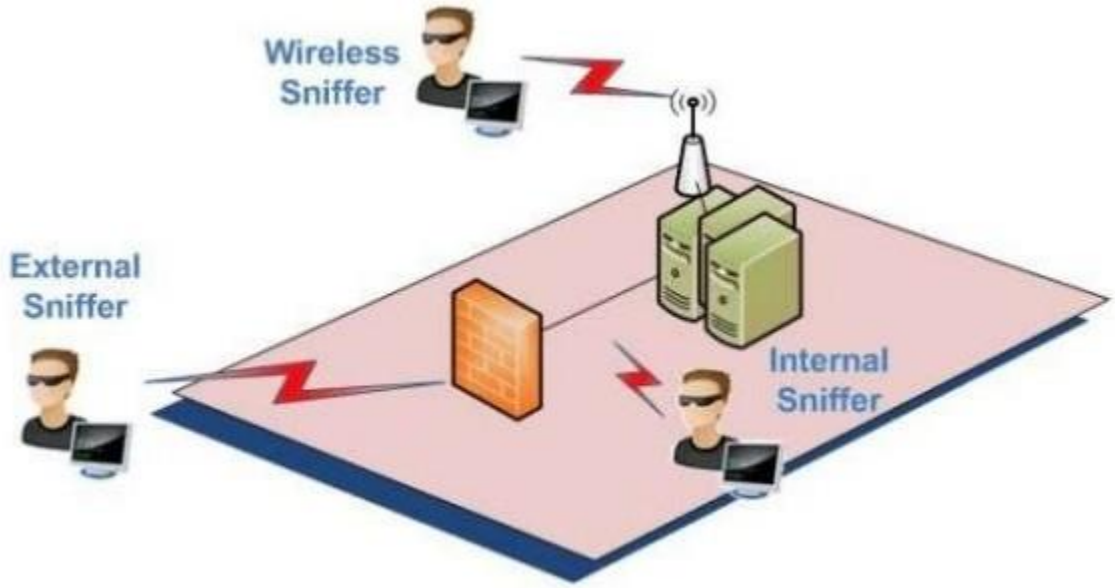
Network sniffing is the process of intercepting data packets sent over a network. This can be done by the specialized software program or hardware equipment. Sniffing can be used to;

- Capture sensitive data such as login credentials
- Eavesdrop on chat messages
- Capture files have been transmitted over a network

The following are protocols that are vulnerable to sniffing

- Telnet
- Rlogin
- HTTP
- SMTP
- NNTP
- POP
- FTP
- IMAP

The above protocols are vulnerable if login details are sent in plain text



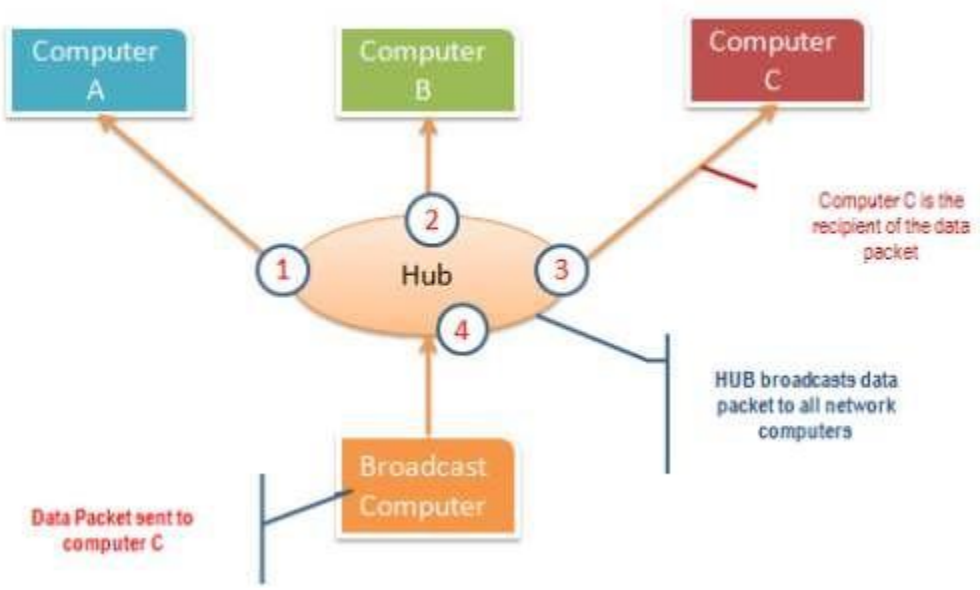
.

Passive and Active Sniffing

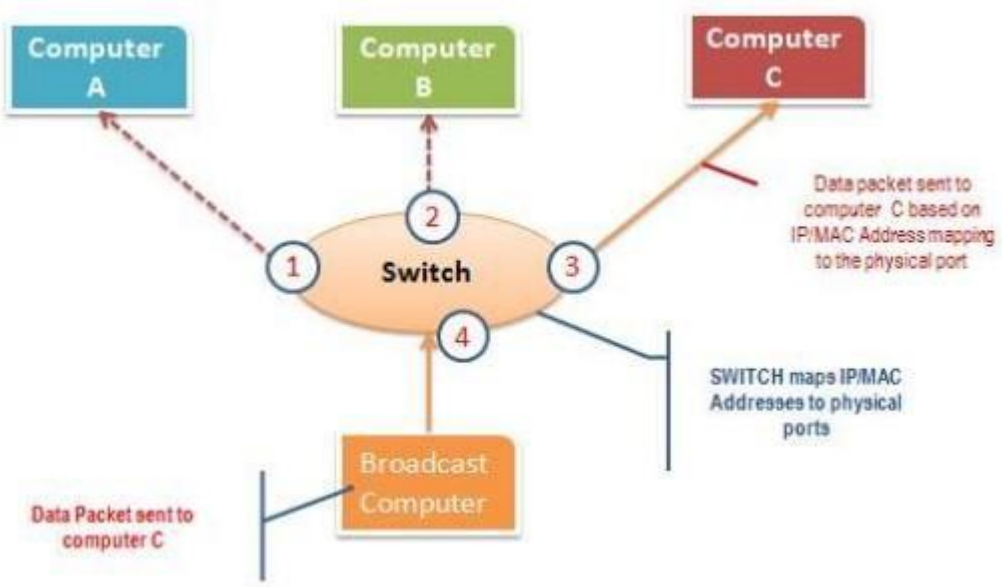
Before we look at passive and active sniffing, let's look at two major devices used to network computers; hubs and switches.

A hub works by sending broadcast messages to all output ports on it except the one that has

sent the broadcast. The recipient computer responds to the broadcast message if the IP address matches. This means when using a hub, all the computers on a network can see the broadcast message. It operates at the physical layer (layer 1) of the OSI Model. Passive and Active Sniffing Before we look at passive and active sniffing, let's look at two major devices used to network computers; hubs and switches.A hub works by sending broadcast messages to all output ports on it except the one that has sent the broadcast. The recipient computer responds to the broadcast message if the IP address matches. This means when using a hub, all the computers on a network can see the broadcast message. It operates at the physical layer (layer 1) of the OSI Model. The diagram below illustrates how the hub works.

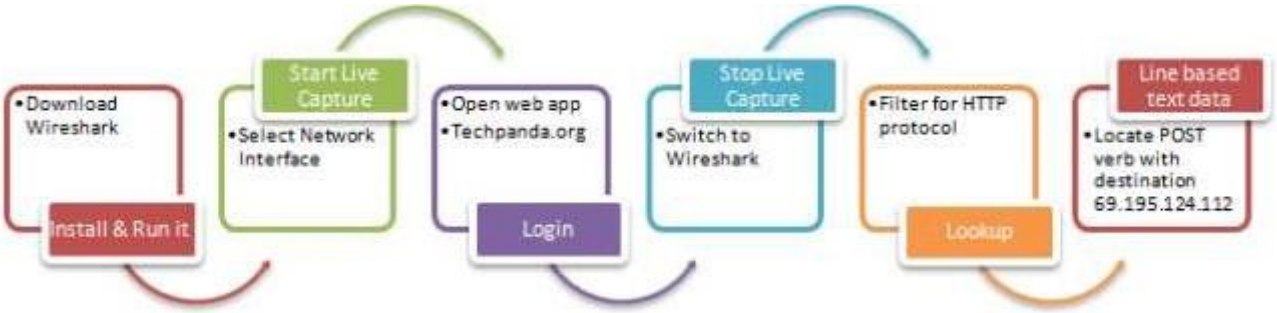


A switch works differently; it maps IP/MAC addresses to physical ports on it. Broadcast messages are sent to the physical ports that match the IP/MAC address configurations for the recipient computer. This means broadcast messages are only seen by the recipient computer. Switches operate at the data link layer (layer 2) and network layer (layer 3). The diagram below illustrates how the switch works.



Sniffing the network using Wireshark

The illustration below shows you the steps that you will carry out to complete this exercise without confusion

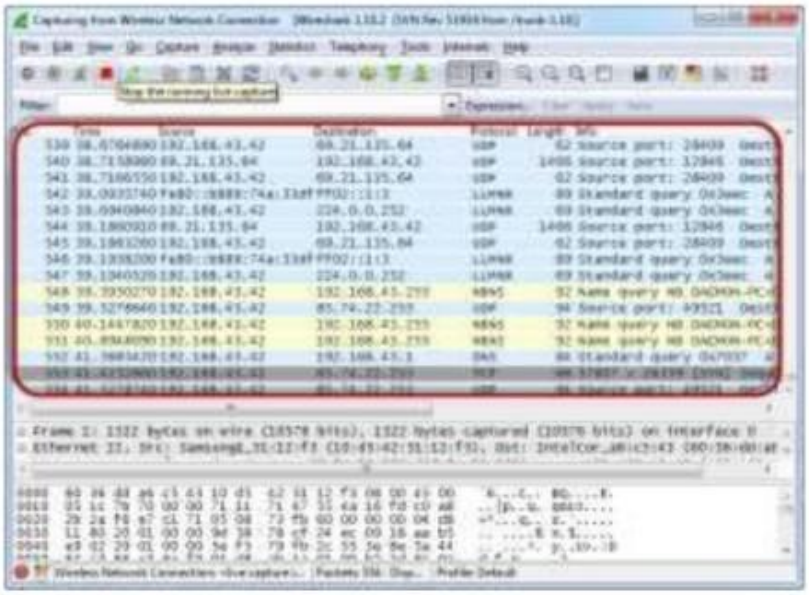


Download Wireshark from this link http://www.wireshark.org/download.html

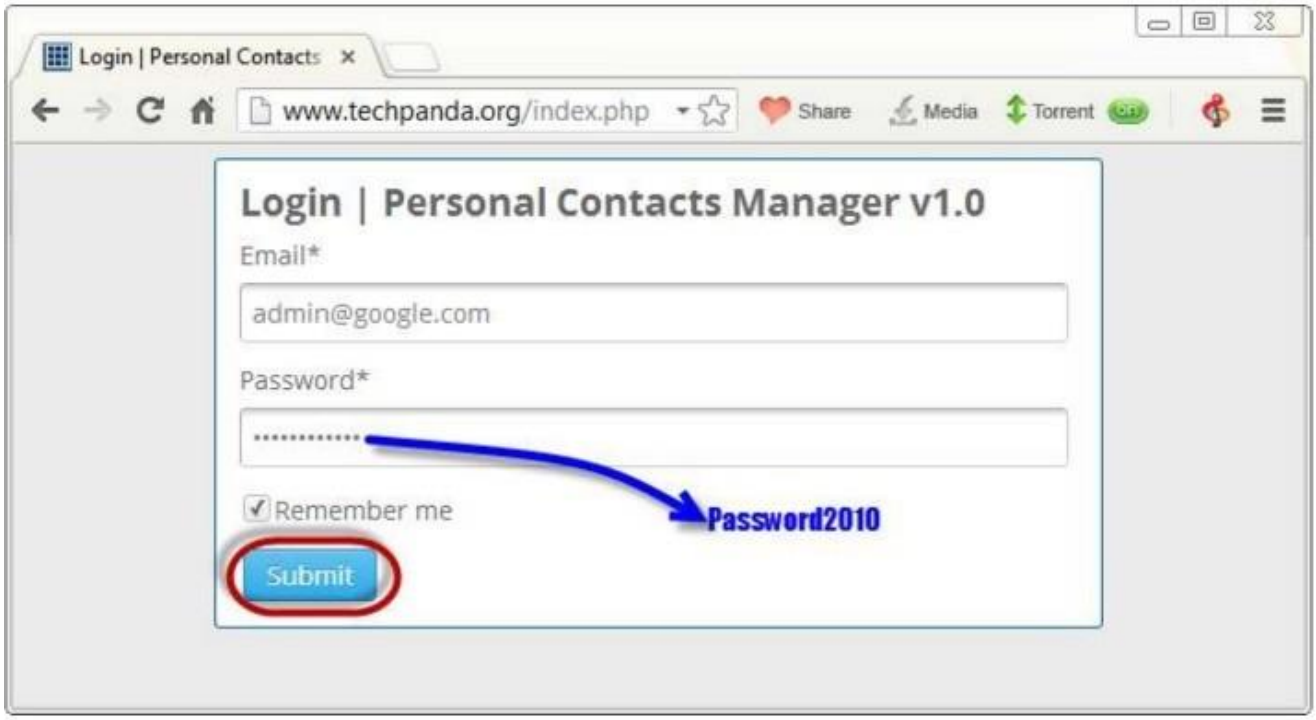
- Open Wireshark
- You will get the following screen

🎬 Select the network interface you want to sniff. Note for this demonstration, we are using a wireless network connection. If you are on a local area network, then you should select the local area network interface.

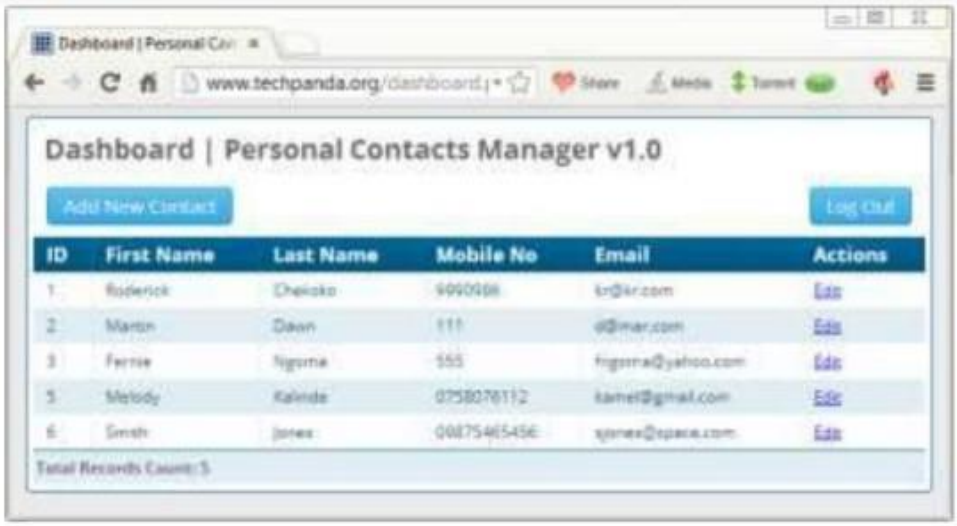
🎬 Click on start button as shown above



🎬 Open your web browser and type in http://www.techpanda.org/

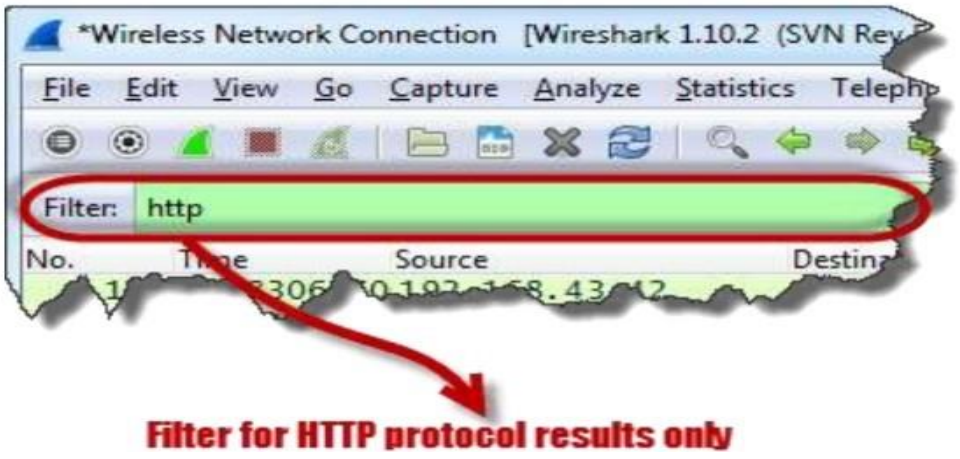


- The login email is admin@google.com and the password is Password2010
- Click on submit button
- A successful logon should give you the following dashboard

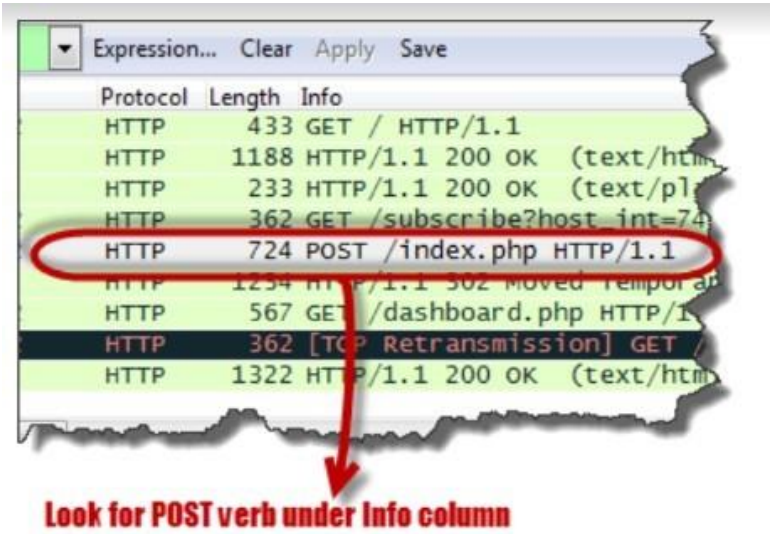


- Go back to Wireshark and stop the live capture

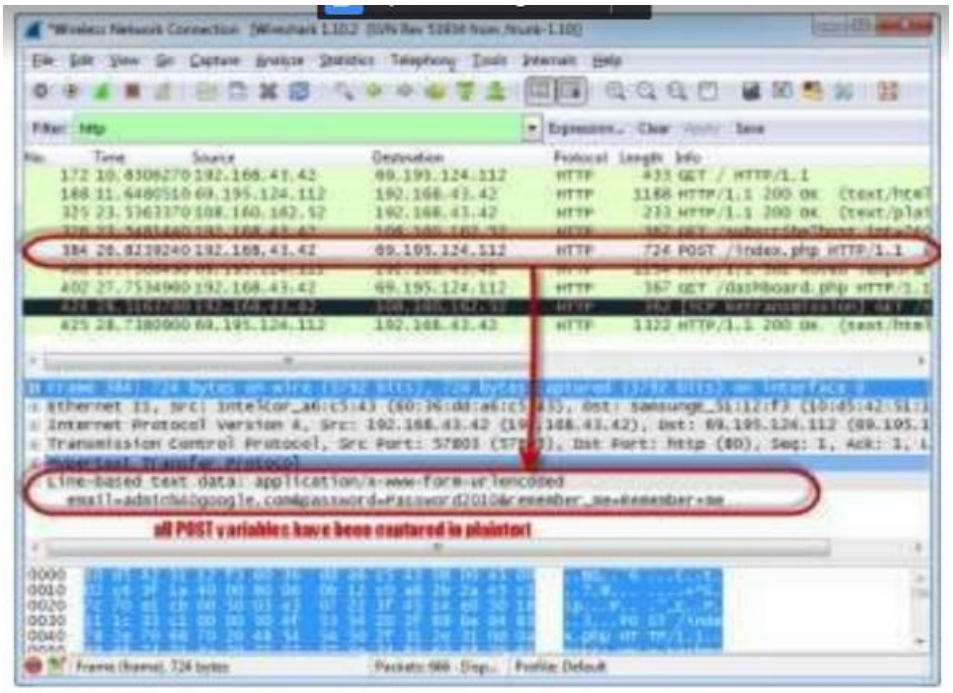
- Filter for HTTP protocol results only using the filter textbox



- Locate the Info column and look for entries with the HTTP verb POST and click on it



- Just below the log entries, there is a panel with a summary of captured data. Look for the summary that says Line-based text data: application/x-www-form-urlencoded



🗔 We should be able to view the plaintext values of all the POST variables submitted to the server via HTTPprotocol.

Ex. No. : 7 **INTRUSION DETECTION SYSTEM (IDS)**
Date:

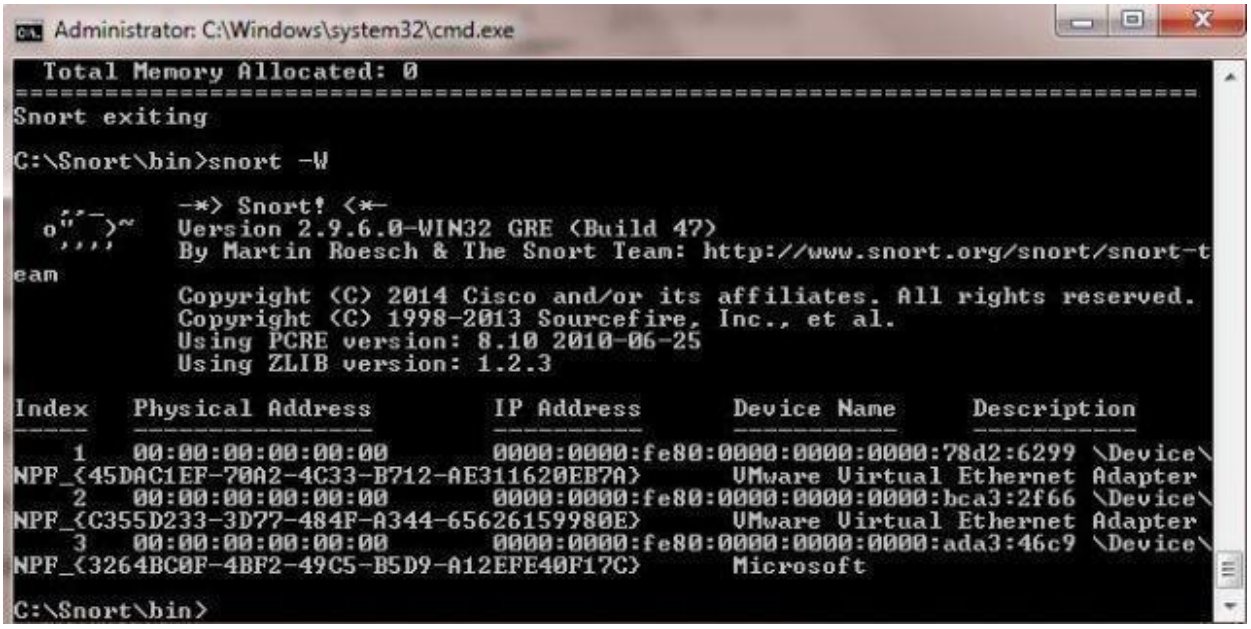
AIM:

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

STEPS ON CONFIGURING AND INTRUSION DETECTION:

1. Download Snort from the Snort.org website. (http://www.snort.org/snort-downloads)
2. Download Rules(https://www.snort.org/snort-rules). You must register to get the rules. (You should download these often)
3. Double click on the .exe to install snort. This will install snort in the "C:\Snort" folder.It is important to have WinPcap (https://www.winpcap.org/install/) installed
4. Extract the Rules file. You will need WinRAR for the .gz file.
5. Copy all files from the "rules" folder of the extracted folder. Now paste the rules into *"C:\Snort\rules"* folder.
6. Copy "snort.conf" file from the "etc" folder of the extracted folder. You must paste it into "C:\Snort\etc" folder. Overwrite any existing file. Remember if you modify your snort.conf file and download a new file, you must modify it for Snort to work.
7. Open a command prompt (cmd.exe) and navigate to folder "C:\Snort\bin" folder. (at the Prompt, type cd\snort\bin)
8. To start (execute) snort in sniffer mode use following command:
snort -dev -i 3
-i indicates the interface number. You must pick the correct interface number. In my case, it is 3.
-dev is used to run snort to capture packets on your network.

To check the interface list, use following command:
snort -W



Finding an interface

You can tell which interface to use by looking at the Index number and finding Microsoft. As you can see in the above example, the other interfaces are for VMWare. My interface is 3.

9. To run snort in IDS mode, you will need to configure the file "snort.conf" according to your network environment.
10. To specify the network address that you want to protect in snort.conf file, look for the following line.
var HOME_NET 192.168.1.0/24 (You will normally see any here)
11. You may also want to set the addresses of DNS_SERVERS, if you have some on your network.

Example:

example snort
12. Change the RULE_PATH variable to the path of rules folder.
var RULE_PATH c:\snort\rules

path to rules
13. Change the path of all library files with the name and path on your system. and you must change the path of snort_dynamicpreprocessorvariable.
C:\Snort\lib\snort_dynamiccpreprocessor
You need to do this to all library files in the "C:\Snort\lib" folder. The old path might be: "/usr/local/lib/…". you will need to replace that path with your system path. Using C:\Snort\lib
14. Change the path ofthe "dynamicengine" variable value in the "snort.conf" file..

Example:
dynamicengine C:\Snort\lib\snort_dynamicengine\sf_engine.dll

15 Add the paths for "include classification.config" and "include reference.config" files.
include c:\snort\etc\classification.config
include c:\snort\etc\reference.config
16. Remove the comment (#) on the line to allow ICMP rules, if it is commented with a #.
include $RULE_PATH/icmp.rules
17. You can also remove the comment of ICMP-info rules comment, if it is commented.
include $RULE_PATH/icmp-info.rules
18. To add log files to store alerts generated by snort, search for the "output log" test in snort.conf and add the following line:
output alert_fast: snort-alerts.ids
19. Comment (add a #) the whitelist $WHITE_LIST_PATH/white_list.rules and the blacklist

Change the nested_ip inner , \ to nested_ip inner #, \
20. Comment out (#) following lines:
#preprocessor normalize_ip4
#preprocessor normalize_tcp: ips ecn stream
#preprocessor normalize_icmp4
#preprocessor normalize_ip6
#preprocessor normalize_icmp6

21. Save the "snort.conf" file.
22. To start snort in IDS mode, run the following command:

snort -c c:\snort\etc\snort.conf -l c:\snort\log -i 3
(Note: 3 is used for my interface card)

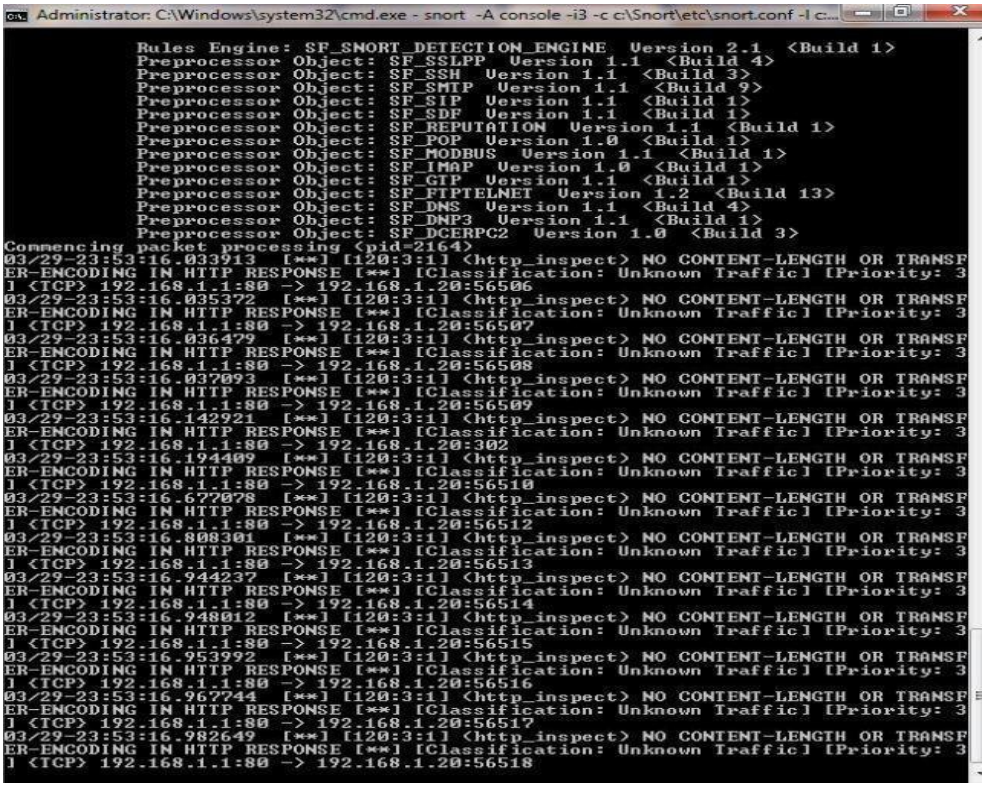
Ifa log is created, select the appropriate program to open it. You can use WordPard or NotePad++ to read the file.

To generate Log files in ASCII mode, you can use following command while running snort in IDS mode:
snort -A console -i3 -c c:\Snort\etc\snort.conf -l c:\Snort\log -K ascii

23. Scan the computer that is running snort from another computer by using PING or NMap (ZenMap).

After scanning or during the scan you can check the snort-alerts.ids file in the log folder to insure it is logging properly. You will see IP address folders appear.

Snort monitoring traffic –



RESULT:

Thus the Intrusion Detection System(IDS) has been demonstrated using the Open Source Intrusion Detection Tool Snort.