OM SAKTHI

ADHIPARASAKTHI COLLEGE OF ENGINEERING
(NAAC ACCREDITED)

G. B. Nagar, Kalavai - 632 506, Ranipet District, Tamil Nadu.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CCS354-NETWORK SECURITY LAB MANUAL

Name :		Reg.No.:
Year/Semester :		

OM SAKTHI

ADHIPARASAKTHI COLLEGE OF ENGINEERING (NAAC ACCREDITED)

G. B. Nagar, Kalavai - 632 506, Ranipet District, Tamil Nadu.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Name	:		
Register Number	:		
Subject Code	: CCS354		
Subject Name	: NETWORK SECURITY		
Semester	: VI	Year:	III

CERTIFICATE

Certified that this is the bonafide record of work done by the above student in the ______________________________ laboratory during the year 2024-2025.

SIGNATURE OF FACULTY-IN-CHARGE **SIGNATURE OF HEAD OF THE DEPARTMENT**

Submitted for the Practical Examination held on ______________________________

Internal Examiner External Examiner

INDEX

EXP NO:1(A DATE:	IMPLEMENTATION OF THE SYMMETRIC KEY ALGORITHMS. DES

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:

DES.java

```
import  java.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: " + new
String(textDecrypted));
}catch(NoSuchAlgorithmException e){
e.printStackTrace();
}catch(NoSuchPaddingException e){
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 DES Algorithm has been implemented and the output verified successfully.

EX.NO: 1 B) DATE:	ADVANCED ENCRYPTION STANDARD (AES) ALGORITHM

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:

AES.java

```
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("
U TF-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 for URL Encryption and the output verified successfully.

EX. NO: 2 A) **DATE:**	**RSA ALGORITHM**

AIM:

To implement RSA (Rivest–Shamir–Adleman) algorithm by using HTML and Javascript.

ALGORITHM:

1. Choose two prime number p and q
2. Compute the value of n and **p**
3. Find the value of ***e*** (public key)
4. Compute the value of ***d*** (private key) using gcd() 5. Do the encryption and decryption

a. Encryption is given as, ***c = te mod n***
b. Decryption is given as, ***t = cd mod n***

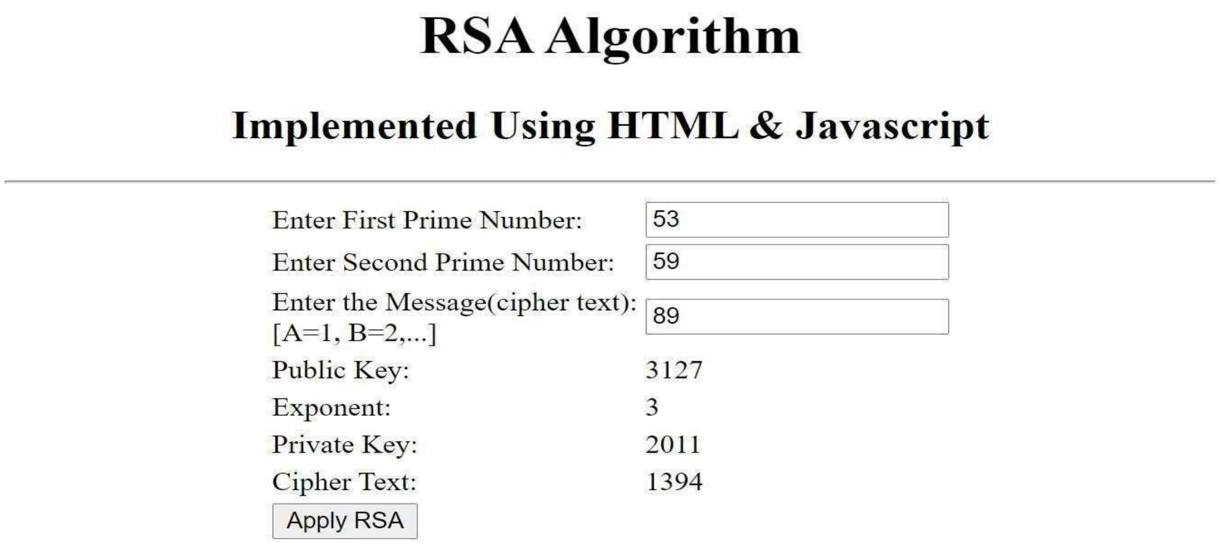
PROGRAM:

```
<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:



RSA Algorithm

Implemented Using HTML & Javascript

Enter First Prime Number:	53
Enter Second Prime Number:	59
Enter the Message(cipher text): [A=1, B=2,...]	89
Public Key:	3127
Exponent:	3
Private Key:	2011
Cipher Text:	1394

RESULT:

Thus the RSA algorithm has been implemented using HTML & CSS and the output has been verified successfully.

EX.NO: 2B) **DATE:**	**DIFFIE-HELLMAN KEY EXCHANGE ALGORITHM**

AIM:

To implement the Diffie-Hellman Key Exchange algorithm for a given problem.

ALGORITHM:

1. Alice and Bob publicly agree to use a modulus *p* = 23 and base *g* = 5 (which is a primitive root modulo 23).
2. Alice chooses a secret integer ***a*** = 4, then sends Bob
A = *g* ***a*** mod *P*, *A* = 5 **4** mod 23 = 4
3. Bob chooses a secret integer ***b*** = 3, then sends Alice
B = *g* ***b*** mod *p* ,*B* = 5 **3** mod 23 = 10
4. Alice computes ***s*** = *B* ***a*** mod *p* , ***s*** = 10 **4** mod 23 = 18
5. Bob computes ***s*** = *A* ***b*** mod *p* ,***s*** = 4 **3** mod 23 = 18

PROGRAM:

DiffieHellman.java

```
Class DiffieHellman {
public static void main(String args[]) {
int p = 23; /* publicly known (prime number) */
int g = 5; /* publicly known (primitive root) */
int x = 4; /* only Alice knows this secret */
int y = 3; /* only Bob knowsthis secret */
double aliceSends = (Math.pow(g, x)) % p;
double bobComputes = (Math.pow(aliceSends, y))
% p;
double bobSends = (Math.pow(g, y)) % p;
double aliceComputes = (Math.pow(bobSends, x)) % p;
double sharedSecret = (Math.pow(g, (x * y))) % p;
System.out.println("simulation of Diffie-Hellman key exchange algorithm\n-----
.............................................................");
System.out.println("Alice Sends : " + aliceSends);
System.out.println("Bob Computes : " + bobComputes);
System.out.println("Bob Sends : " + bobSends);
System.out.println("AliceComputes : " + aliceComputes);
System.out.println("Shared Secret : " + sharedSecret);
/* shared secrets should match and equality is transitive */
if ((aliceComputes == sharedSecret) && (aliceComputes ==
bobComputes))
System.out.println("Success: Shared Secrets Matches! " +
sharedSecret);
else
System.out.println("Error: Shared Secrets does not Match");
}
}
```

OUTPUT:

```
simulation of Diffie-Hellman key exchange algorithm
------------------------------------------------------------------------
AliceSends: 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 *Diffie-Hellman key exchange algorithm* has been implemented using JavaProgram and the output has been verified successfully.

EX. NO: 3 DATE:	DIGITAL SIGNATURE STANDARD

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.*;
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("Entersome 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(privKe
y); 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 Digital Signature Standard Signature Scheme has been implemented and the output has been verified successfully.

EX. NO: 4 **DATE:**	**INSTALLATION OF WIRE SHARK, TCP DUMP AND OBSERVE DATATRANSFERRED IN CLIENT-SERVER COMMUNICATION USING UDP/TCP AND IDENTIFY THE UDP/TCP.**

AIM:

To install the Wire shark,TCP dump and observe data transferred in client-server communication using UDP/TCP and identity the TCP/UDP datagram.

PROCEDURE:

UDP (User Datagram Protocol) is an alternative communications protocol to Transmission Control Proto- col (TCP) used primarily for establishing lowlatency
and loss tolerating connections between applica- tions on the Internet. Both UDP
and TCP run on top of the Internet Protocol (IP) and are sometimes re- ferred to as
UDP/IP or TCP/IP. Both protocols send short packets of data, called datagrams. To
look at the details of UDP (User Datagram Protocol). UDP is a transport protocol used throughout the Internet as an alternative to TCP when reliability is not required. UDP provides two services not provided by the IP layer. It provides port
numbers to help distinguish different user requests and, optionally, a checksum capability to verify that the data arrived intact. TCP has emerged as the dominant
protocol used for the bulk of Internet connectivity owing to services for breaking
large data sets into individual packets, check- ing for and resending lost packets and reassembling packets into the correct sequence. But these addi- tional services come at a cost in terms of additional data overhead, and delays called latency. In contrast, UDP just sends the packets, which means that it has much lower bandwidth overhead and latency. But packets can be lost or received out of order as a result, owing to the different paths individ- ual packets traverse between sender and receiver. UDP is an ideal protocol for network applications in which perceived latency is critical such as gaming, voice and

video communications, which can suffer some data loss without adversely affecting perceived quality. In
some cases, forward error correction techniques are used to improve audio and video quality in spite of some loss. UDP can also be used in applications that require lossless data transmission when the application is configured to manage the process of retransmitting lost packets and correctly arranging received packets.This approach can help to improve the data transfer rate of large files compared with TCP. We first examine UDP.

Step 1: Capture a UDP Trace

There are many ways to cause your computer to send and receive UDP messages since UDP is widelyused as a transport protocol. The easiest options are to:

- Do nothing but wait for a while. UDP is used for many "system protocols" that typically run in the background and produce small amounts of traffic, e.g., DHCP for IP address assignment andNTP for time synchronization.
- Use your browser to visit sites. UDP is used by DNS for resolving domain names to IP addresses,so visiting fresh sites will cause DNS traffic to be sent. Be careful not to visit unsafe sites; pick recommended sites or sites you know about but have not visited recently. Simply browsing the web is likely to cause a steady stream of DNS traffic.
- Start up a voice-over-IP call with your favorite client. UDP is used by RTP, which is the protocol commonly used to carry media samples in a voice or video call overthe Internet.

1. Launch Wireshark by entering Wireshark in the "ask my anything" search box in Windows.

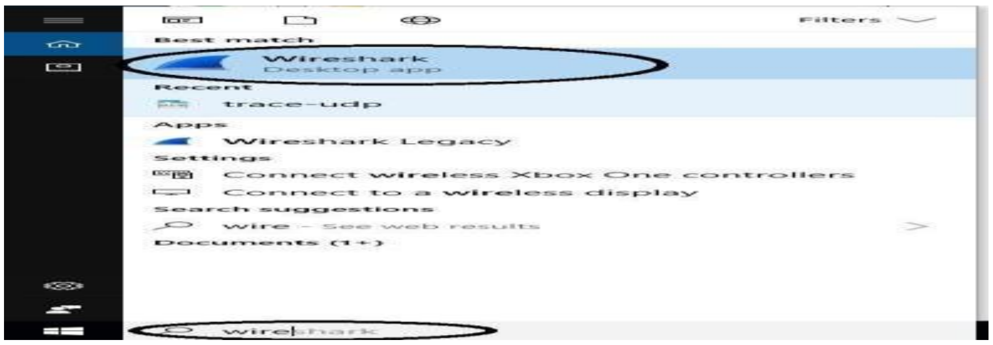


Figure 1: Starting Wireshark

2. Once Wireshark starts, select the Ethernet interface.

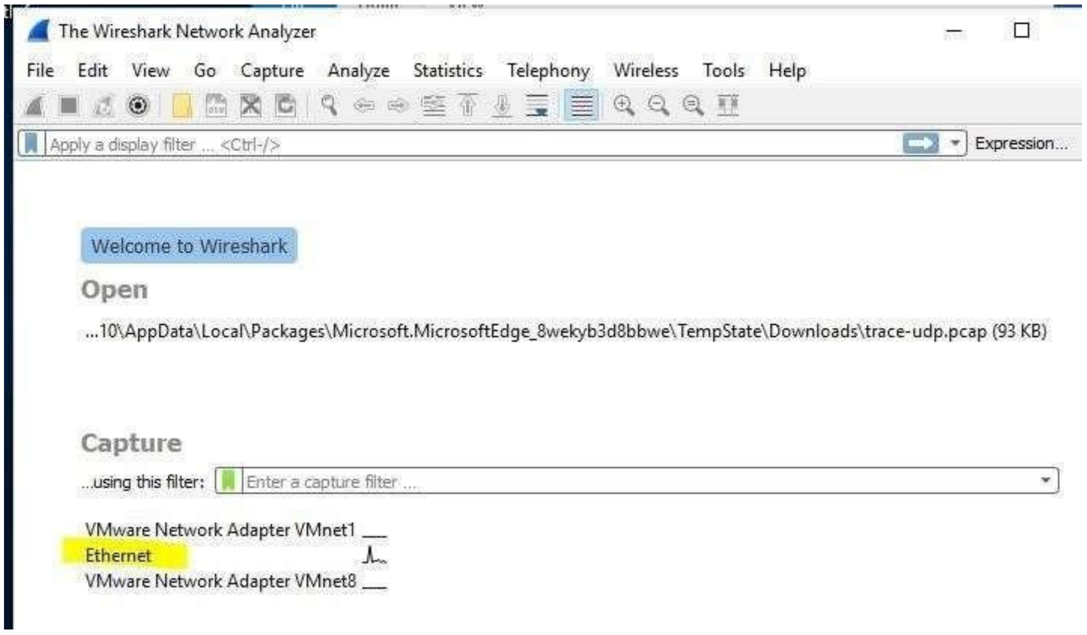
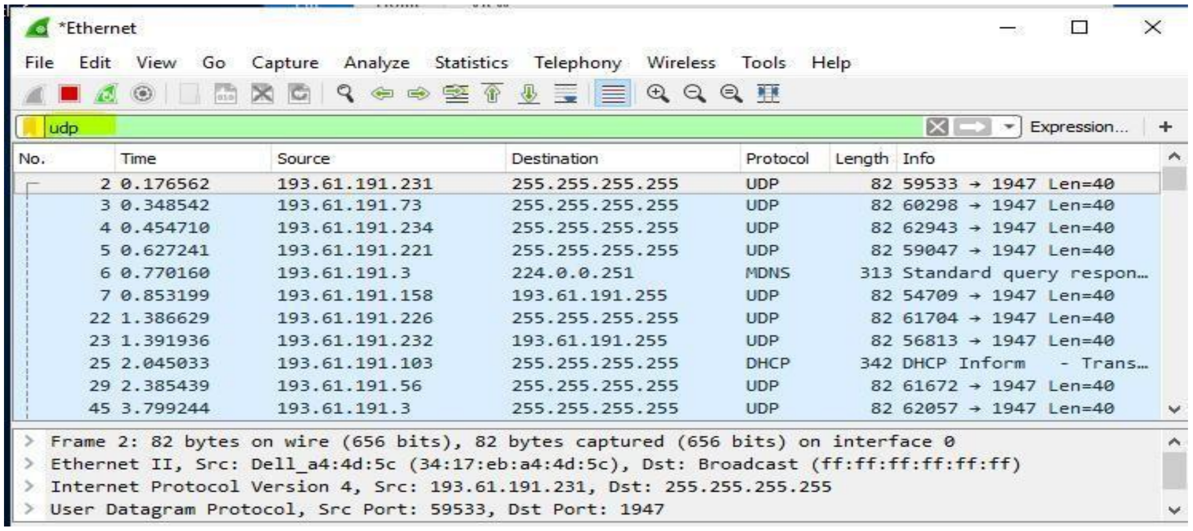

The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
Welcome to Wireshark
Open
...10\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\trace-udp.pcap (93 KB)
Capture
...using this filter:
Enter a capture filter ...
VMware Network Adapter VMnet1
Ethernet
VMware Network Adapter VMnet8

Figure 2: Selecting the Ethernet Interface

3. Wireshark will automatically start capturing packets on the network. Now, enter a filter of udp. (This is shown below).


*Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
udp
Expression...

No.	Time	Source	Destination	Protocol	Length	Info
2	0.176562	193.61.191.231	255.255.255.255	UDP	82	59533 → 1947 Len=40
3	0.348542	193.61.191.73	255.255.255.255	UDP	82	60298 → 1947 Len=40
4	0.454710	193.61.191.234	255.255.255.255	UDP	82	62943 → 1947 Len=40
5	0.627241	193.61.191.221	255.255.255.255	UDP	82	59047 → 1947 Len=40
6	0.770160	193.61.191.3	224.0.0.251	MDNS	313	Standard query respon…
7	0.853199	193.61.191.158	193.61.191.255	UDP	82	54709 → 1947 Len=40
22	1.386629	193.61.191.226	255.255.255.255	UDP	82	61704 → 1947 Len=40
23	1.391936	193.61.191.232	193.61.191.255	UDP	82	56813 → 1947 Len=40
25	2.045033	193.61.191.103	255.255.255.255	DHCP	342	DHCP Inform - Trans…
29	2.385439	193.61.191.56	255.255.255.255	UDP	82	61672 → 1947 Len=40
45	3.799244	193.61.191.3	255.255.255.255	UDP	82	62057 → 1947 Len=40

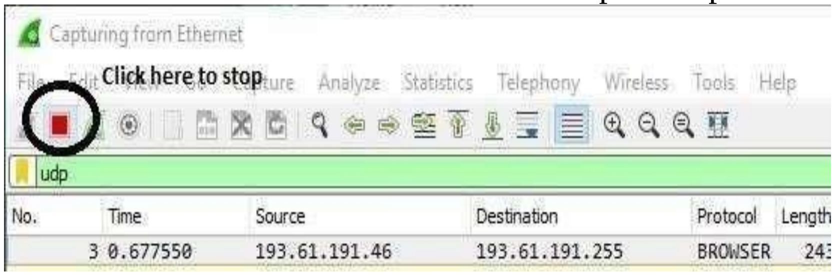
```
> Frame 2: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface 0
> Ethernet II, Src: Dell_a4:4d:5c (34:17:eb:a4:4d:5c), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 193.61.191.231, Dst: 255.255.255.255
> User Datagram Protocol, Src Port: 59533, Dst Port: 1947
```

Figure 3: Setting up the capture options.

4. When the capture is started, it will collect UDP traffic automatically.

5. Wait a little while (say 60 seconds) after you have stopped your activity to also observe any background UDP traffic. It is likely that you will observe a trickle of UDP traffic because system activity often uses UDP to communicate. We want to see some of this activity.

6. Use the Wireshark menus or buttons to stop the capture.


Capturing from Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Click here to stop
udp

No.	Time	Source	Destination	Protocol	Length
3	0.677550	193.61.191.46	193.61.191.255	BROWSER	243

7.You should now have a trace with many UDP packets.

Step 2: Inspect the Trace

Different computers are likely to capture different kinds of UDP traffic depending on the network setup and local activity. Observe that the protocol column is likely to show multiple protocols, none of which is UDP. This is because the listed protocol is an application protocol layered on top of UDP. Wireshark gives the name of the application protocol, not the (UDP) transport protocol unless Wireshark cannot determine the application protocol. However, even if the packets are listed as an application protocol, they will have a UDP protocol header for us to study, following the IP and lower-layer protocol headers.

Select different packets in the trace (in the top panel) and browse the expanded UDP header (in the mid- dle panel). You will see that it contains the following fields:

• Source Port, the port from which the UDP message is sent. It is given as a number and possibly a text name; names are given to port values that are registered for use with a specific application.

- Destination Port. This is the port number and possibly name to which the UDP message is des-tined. Ports are the only form of addressing in UDP. The computer is identified using the IP ad-dress in the lower IP layer.

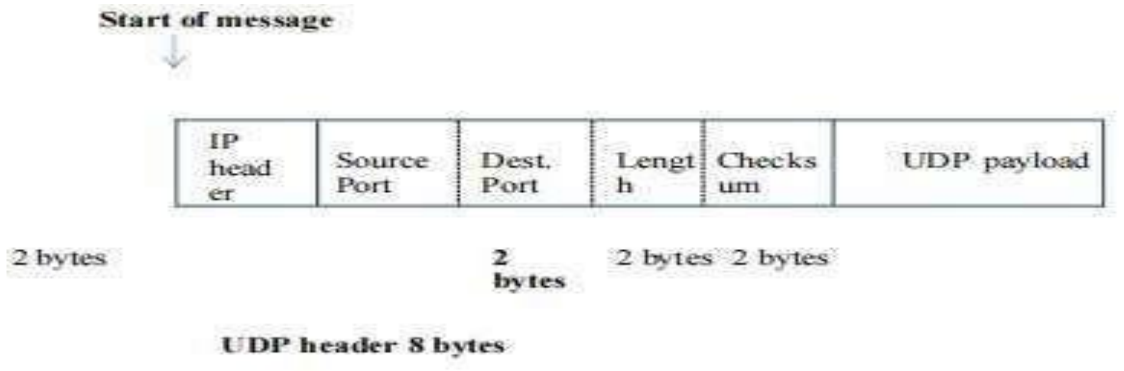
- Length. The length of the UDP message.

- Checksum. A checksum over the message that is used to validate its contents. Is your checksum carrying 0 and flagged as incorrect for UDP messages sent from your computer? On some com- puters, the operating system software leaves the checksum blank (zero) for the NIC to compute and fill in as the packet is sent. This is called protocol offloading. It happens after Wireshark seesthe packet, which causes Wireshark to believe that the checksum is wrong and flag it with a dif- ferent color to signal a problem. You can remove these false errors if they are occurring by tell- ing Wireshark not to validate the checksums. Select “Preferences” from the
Wireshark menus and expand the “Protocols” area. Look under the list until you come to UDP. Uncheck “Validate checksum if possible”.

The UDP header has different values for different messages, but as you can see, it is short and sweet. The remainder of the message is the UDP payload that is normally identified the higher-layer pro-tocol that it carries, e.g., DNS, or RTP.

Step 3: UDP Message Structure

The figure below shows the UDP message structure as you observed. It shows the position of the IP header, UDP header, and UDP payload. Within the UDP header, it shows the position and size of each UDP field. Note how the Length field gives the length of the UDP payload plus the UDP header. The checksum is 16 bits long and the UDP header is 8 bytes long.



Step 4: UDP Usage

The Protocol field in the IP header is how IP knows that the next higher protocollayer is UDP. The IP Pro-tocol field value of 17 indicates UDP.

You might be surprised to find UDP messages in your trace that neither come from your computer or aresent only to your computer. You can see this by sorting on the Source and Destination columns. The source and destinations will be domain names, if Network layer name resolution is turned on, and oth-erwise IP addresses. (You can toggle this setting using the View menu and selecting Name resolution.) You can find out the IP address of your computer using the “ipconfig” command(Windows).

The reason you may find UDP messages without your computer’s IP address as either the source or des-tination IP address is that UDP is widely used as part of system protocols. These protocols often send messages to all local computers who are interested in them using broadcast and multicast addresses. In our traces, we find DNS (the domain name system), MDNS (DNS traffic that uses IP multicast), NTP (for time synchronization), NBNS (NetBIOS traffic), DHCP (for IP addressassignment), SSDP (a service discov- ery protocol), STUN (a NAT traversal protocol), RTP (for carrying audio and video samples), and more.

A variety of broadcast and multicast addresses may be found. These include the Internet broadcast ad- dress of 255.255.255.255, subnet broadcast addresses such as 192.168.255.255 and multicast IP ad- dresses such as 224.0.xx.xx for multicast DNS.

Note also that UDP messages can be as large as roughly 64Kbytes but most often they are a few hun-dred bytes or less, typically around 100 bytes.

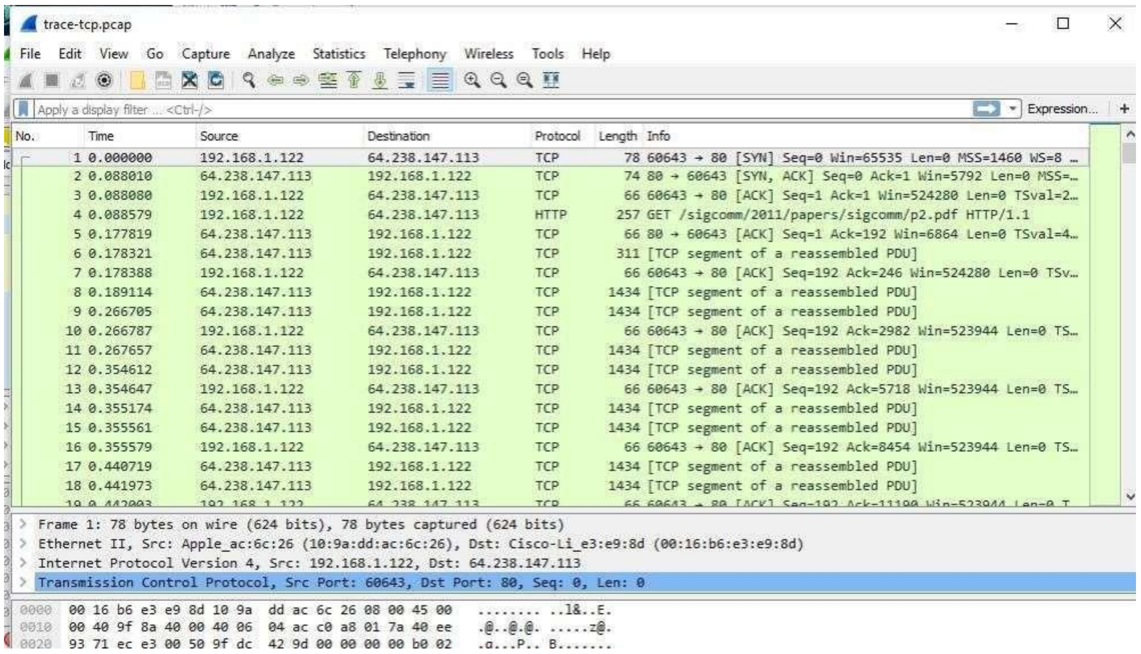
TCP

Objective

To see the details of TCP (Transmission Control Protocol). TCP is the main transport layer protocol usedin the Internet.

Step 1: Open the Trace

Open the trace file here: https://kevincurran.org/com320/labs/wireshark/trace tcp.pcap


trace-tcp.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.   Time       Source          Destination     Protocol Length Info
    1 0.000000   192.168.1.122   64.238.147.113  TCP        78 60643 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 …
    2 0.088010   64.238.147.113  192.168.1.122   TCP        74 80 → 60643 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=…
    3 0.088080   192.168.1.122   64.238.147.113  TCP        66 60643 → 80 [ACK] Seq=1 Ack=1 Win=524280 Len=0 TSval=2…
    4 0.088579   192.168.1.122   64.238.147.113  HTTP      257 GET /sigcomm/2011/papers/sigcomm/p2.pdf HTTP/1.1
    5 0.177819   64.238.147.113  192.168.1.122   TCP        66 80 → 60643 [ACK] Seq=1 Ack=192 Win=6864 Len=0 TSval=4…
    6 0.178321   64.238.147.113  192.168.1.122   TCP       311 [TCP segment of a reassembled PDU]
    7 0.178388   192.168.1.122   64.238.147.113  TCP        66 60643 → 80 [ACK] Seq=192 Ack=246 Win=524280 Len=0 TSv…
    8 0.189114   64.238.147.113  192.168.1.122   TCP      1434 [TCP segment of a reassembled PDU]
    9 0.266705   64.238.147.113  192.168.1.122   TCP      1434 [TCP segment of a reassembled PDU]
   10 0.266787   192.168.1.122   64.238.147.113  TCP        66 60643 → 80 [ACK] Seq=192 Ack=2982 Win=523944 Len=0 TS…
   11 0.267657   64.238.147.113  192.168.1.122   TCP      1434 [TCP segment of a reassembled PDU]
   12 0.354612   64.238.147.113  192.168.1.122   TCP      1434 [TCP segment of a reassembled PDU]
   13 0.354647   192.168.1.122   64.238.147.113  TCP        66 60643 → 80 [ACK] Seq=192 Ack=5718 Win=523944 Len=0 TS…
   14 0.355174   64.238.147.113  192.168.1.122   TCP      1434 [TCP segment of a reassembled PDU]
   15 0.355561   64.238.147.113  192.168.1.122   TCP      1434 [TCP segment of a reassembled PDU]
   16 0.355579   192.168.1.122   64.238.147.113  TCP        66 60643 → 80 [ACK] Seq=192 Ack=8454 Win=523944 Len=0 TS…
   17 0.440719   64.238.147.113  192.168.1.122   TCP      1434 [TCP segment of a reassembled PDU]
   18 0.441973   64.238.147.113  192.168.1.122   TCP      1434 [TCP segment of a reassembled PDU]
   19 0.442003   192.168.1.122   64.238.147.113  TCP        66 60643 → 80 [ACK] Seq=192 Ack=11190 Win=523944 Len=0 T

> Frame 1: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
> Ethernet II, Src: Apple_ac:6c:26 (10:9a:dd:ac:6c:26), Dst: Cisco-Li_e3:e9:8d (00:16:b6:e3:e9:8d)
> Internet Protocol Version 4, Src: 192.168.1.122, Dst: 64.238.147.113
> Transmission Control Protocol, Src Port: 60643, Dst Port: 80, Seq: 0, Len: 0

0000  00 16 b6 e3 e9 8d 10 9a  dd ac 6c 26 08 00 45 00   ........ ..l&..E.
0010  00 40 9f 8a 40 00 40 06  04 ac c0 a8 01 7a 40 ee   .@..@.@. .....z@.
0020  93 71 ec e3 00 50 9f dc  42 9d 00 00 00 00 b0 02   .q...P.. B.......
```

Step 2: Inspect the Trace

Select a long packet anywhere in the middle of your trace whose protocol is listed as TCP. Expand the TCP protocol section in the middle panel (by using the “+”expander or icon). All packets except the initial HTTP GET and last packet of the HTTP response should be listed as TCP. Picking a long packet ensures that we are looking at a download packet from the server to your computer. Looking at the protocol lay- ers, you should see an IP block before the TCP block. This is because the TCP segment is carried in an IP. We have shown the TCP block expanded in our figure.

You will see roughly the following fields

First comes the source port, then the destination port. This is the addressing that TCP adds be- yond the IP address. The source port is likely to be 80 since the packet was sent by a web server and the standard web server port is 80.

Then there is the sequence number field. It gives the position in the byte stream of the first pay-load byte.

Next is the acknowledgement field. It tells the last received position in the reverse byte stream.

The header length giving the length of the TCP header.

The flags field has multiple flag bits to indicate the type of TCP segment. You can expand it andlook at the possible flags.

Next is a checksum, to detect transmission errors.

There may be an Options field with various options. You can expand this field and explore if youwould like, but we will look at the options in more detail later.

Finally, there may be a TCP payload, carrying the bytes that are being transported.

As well as the above fields, there may be other informational lines that Wireshark provides to help youinterpret the packet. We have covered only the fields that are carried across the network.

Step 3: TCP Segment Structure

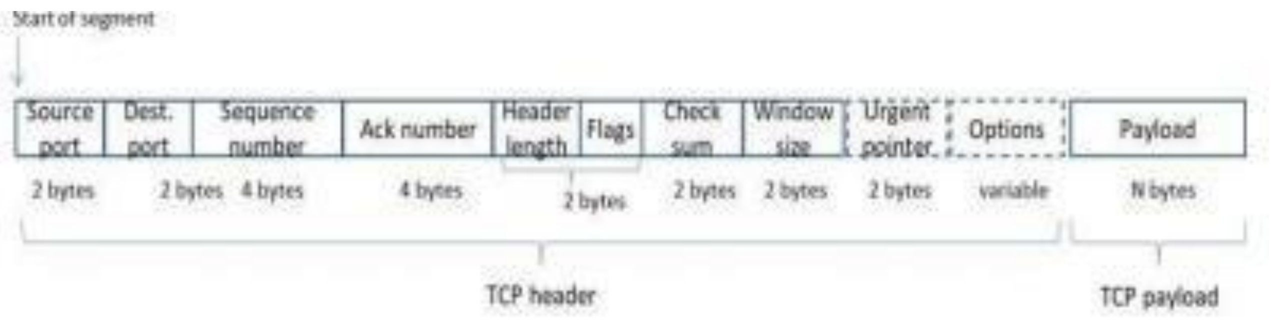


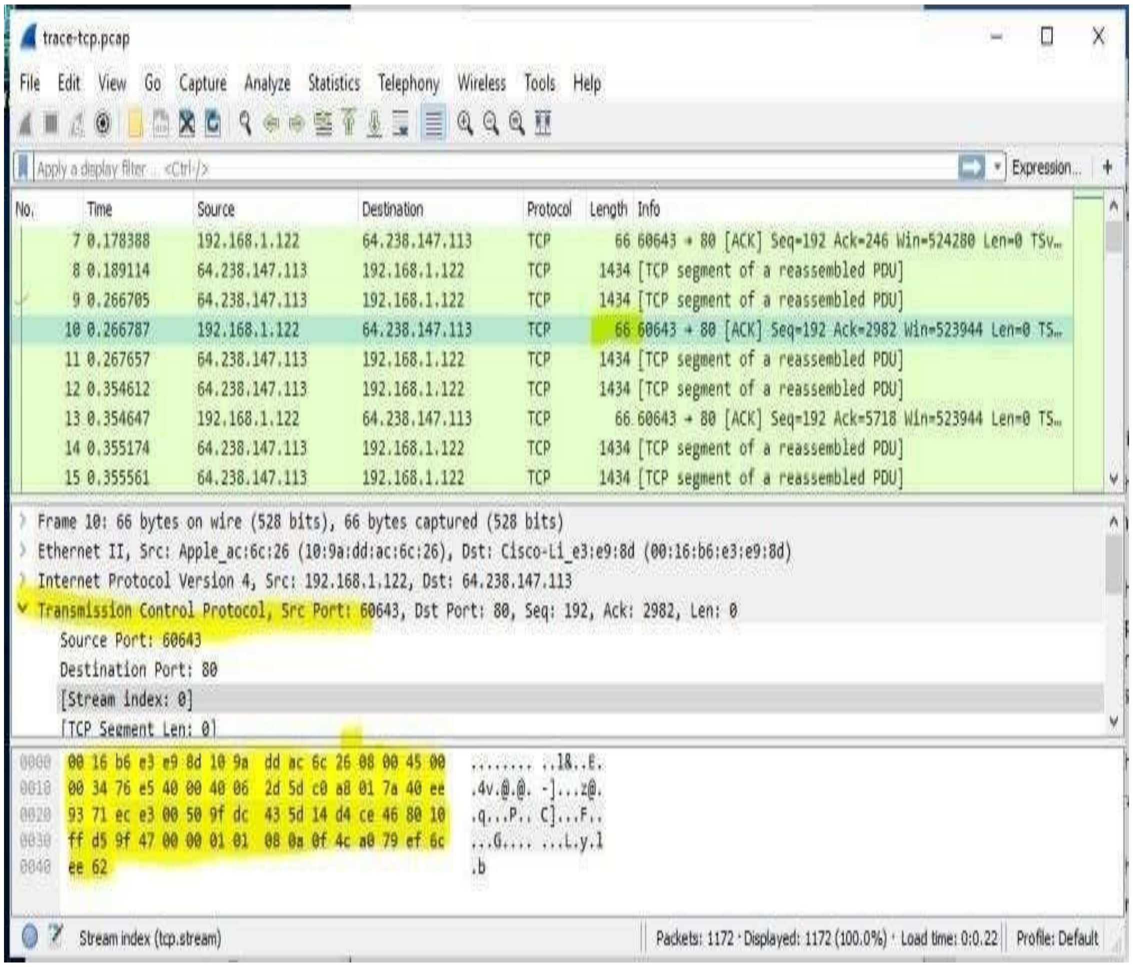
Figure 7: Structure of a TCP segment

This drawing differs from the text drawing in the book in only minor respects:

The Header length and Flags fields are combined into a 2-byte quantity. It is not easy to deter-mine their bit lengths with Wireshark.

The Urgent Pointer field is shown as dotted. This field is typically not used, and so does not showup in Wireshark and we do not expect you to have it in your drawing. You can notice its exist- ence in Wireshark, however, by observing the zero bytes in the segment that are skipped over as you select the different fields.

The Options field is shown dotted, as it may or may not be present for the segments in your trace. Most often it will be present, and when it is then its length will be a multiple of four bytes.
Figure 8: Examining the size of segment


trace-tcp.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

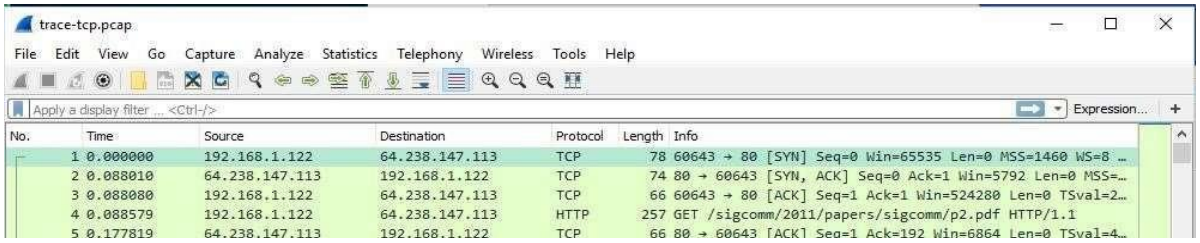
```
No.  Time      Source          Destination     Protocol Length Info
   7 0.178388  192.168.1.122   64.238.147.113  TCP        66 60643 → 80 [ACK] Seq=192 Ack=246 Win=524280 Len=0 TSv…
   8 0.189114  64.238.147.113  192.168.1.122   TCP      1434 [TCP segment of a reassembled PDU]
   9 0.266705  64.238.147.113  192.168.1.122   TCP      1434 [TCP segment of a reassembled PDU]
  10 0.266787  192.168.1.122   64.238.147.113  TCP        66 60643 → 80 [ACK] Seq=192 Ack=2982 Win=523944 Len=0 TS…
  11 0.267657  64.238.147.113  192.168.1.122   TCP      1434 [TCP segment of a reassembled PDU]
  12 0.354612  64.238.147.113  192.168.1.122   TCP      1434 [TCP segment of a reassembled PDU]
  13 0.354647  192.168.1.122   64.238.147.113  TCP        66 60643 → 80 [ACK] Seq=192 Ack=5718 Win=523944 Len=0 TS…
  14 0.355174  64.238.147.113  192.168.1.122   TCP      1434 [TCP segment of a reassembled PDU]
  15 0.355561  64.238.147.113  192.168.1.122   TCP      1434 [TCP segment of a reassembled PDU]
```

```
> Frame 10: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
> Ethernet II, Src: Apple_ac:6c:26 (10:9a:dd:ac:6c:26), Dst: Cisco-Li_e3:e9:8d (00:16:b6:e3:e9:8d)
> Internet Protocol Version 4, Src: 192.168.1.122, Dst: 64.238.147.113
v Transmission Control Protocol, Src Port: 60643, Dst Port: 80, Seq: 192, Ack: 2982, Len: 0
    Source Port: 60643
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 0]
```

```
0000  00 16 b6 e3 e9 8d 10 9a  dd ac 6c 26 08 00 45 00   ........ ..l&..E.
0010  00 34 76 e5 40 00 40 06  2d 5d c0 a8 01 7a 40 ee   .4v.@.@. -]...z@.
0020  93 71 ec e3 00 50 9f dc  43 5d 14 d4 ce 46 80 10   .q...P.. C]...F..
0030  ff d5 9f 47 00 00 01 01  08 0a 0f 4c a0 79 ef 6c   ...G.... ...L.y.l
0040  ee 62                                              .b
```

Stream index (tcp.stream)   Packets: 1172 · Displayed: 1172 (100.0%) · Load time: 0:0.22   Profile: Default

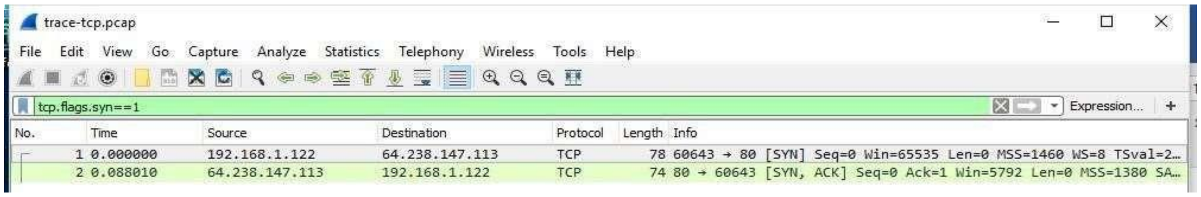
Step 4: TCP Connection Setup/Teardown
Three-Way Handshake To see the "three-way handshake" in action, look for


trace-tcp.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.  Time      Source          Destination     Protocol Length Info
  1 0.000000   192.168.1.122   64.238.147.113  TCP       78 60643 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 …
  2 0.088010   64.238.147.113  192.168.1.122   TCP       74 80 → 60643 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=…
  3 0.088080   192.168.1.122   64.238.147.113  TCP       66 60643 → 80 [ACK] Seq=1 Ack=1 Win=524280 Len=0 TSval=2…
  4 0.088579   192.168.1.122   64.238.147.113  HTTP     257 GET /sigcomm/2011/papers/sigcomm/p2.pdf HTTP/1.1
  5 0.177819   64.238.147.113  192.168.1.122   TCP       66 80 → 60643 [ACK] Seq=1 Ack=192 Win=6864 Len=0 TSval=4…
```

Figure 9: Selecting a TCP segment with SYN flag.

The SYN flag is noted in the Info column. You can also search for packets with the SYN flag on using the filter expression "tcp.flags.syn==1". (Se below)


trace-tcp.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.flags.syn==1
Expression...

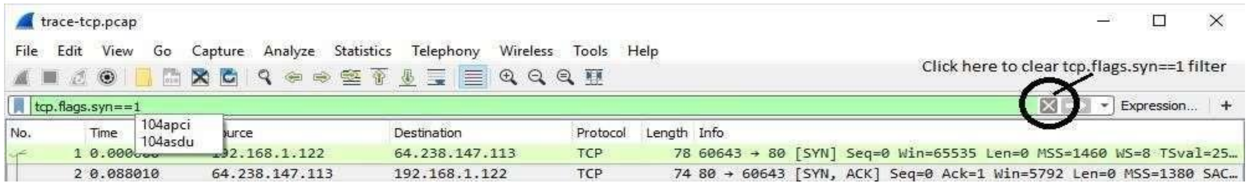
```
No.  Time      Source          Destination     Protocol Length Info
  1 0.000000   192.168.1.122   64.238.147.113  TCP       78 60643 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 TSval=2…
  2 0.088010   64.238.147.113  192.168.1.122   TCP       74 80 → 60643 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1380 SA…
```

Figure 10: Selecting a TCP segment with SYN flag on.

a TCP segment with the SYN flag on. These are up at the beginning of your trace, and the packets that follow it (see below).
A "SYN packet" is the start of the three-way handshake. In this case it will be sent from your computer to the remote server. The remote server should reply with a TCP segment with the SYN and ACK flags set, or a "SYN ACK packet". On receiving this segment, your computer will ACK it, consider the connec-tion set up, and begin sending data, which in this case will be the HTTP request.

Step 5: TCP Connection Setup/Teardown

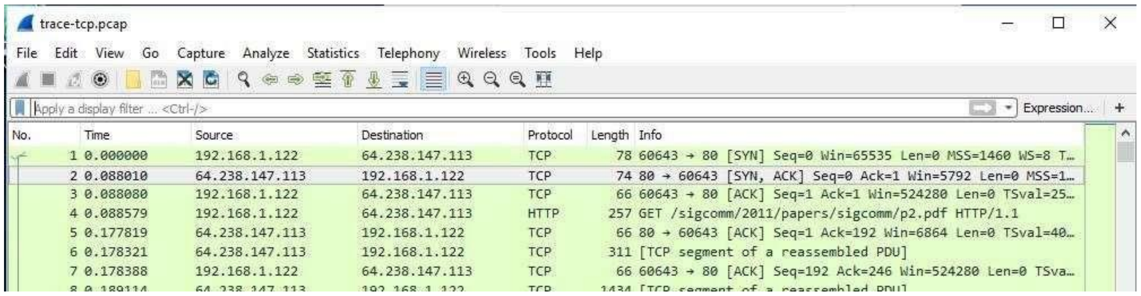
Next, we wish to clear the display filter tcp.flags.syn==1 so that we can once again see all the packets inour original trace. Do this by clearing the display filter as shown below.


trace-tcp.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Click here to clear tcp.flags.syn==1 filter
tcp.flags.syn==1
104apci
104asdu
Expression...

```
No.  Time      Source          Destination     Protocol Length Info
  1 0.000000   192.168.1.122   64.238.147.113  TCP       78 60643 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 TSval=25…
  2 0.088010   64.238.147.113  192.168.1.122   TCP       74 80 → 60643 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1380 SAC…
```

Figure 11: Clearing the display filter TCP segment with SYN flag on

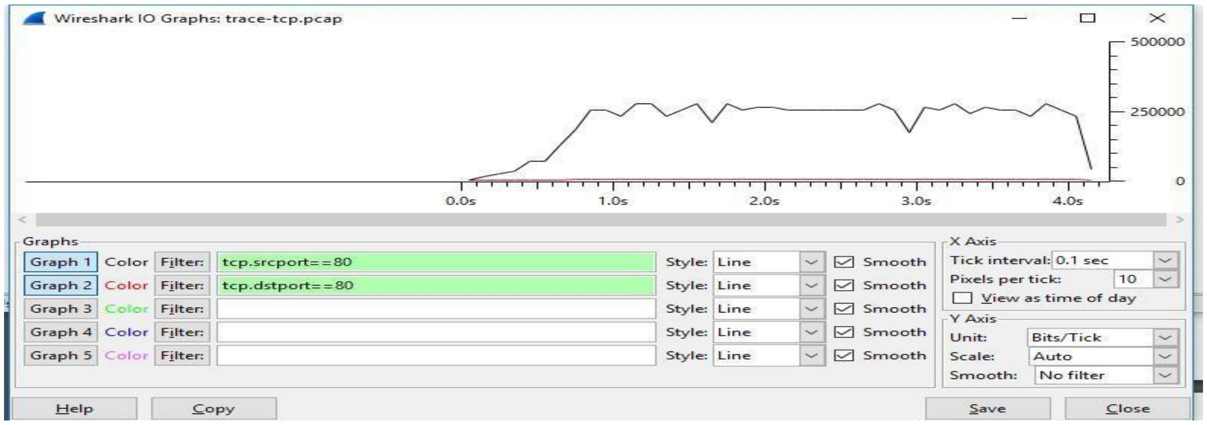
If you do this correctly, you should see the full trace. We are most interested in the first three packets.


trace-tcp.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.  Time      Source          Destination     Protocol Length Info
   1 0.000000  192.168.1.122   64.238.147.113  TCP          78 60643 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 T…
   2 0.088010  64.238.147.113  192.168.1.122   TCP          74 80 → 60643 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1…
   3 0.088080  192.168.1.122   64.238.147.113  TCP          66 60643 → 80 [ACK] Seq=1 Ack=1 Win=524280 Len=0 TSval=25…
   4 0.088579  192.168.1.122   64.238.147.113  HTTP        257 GET /sigcomm/2011/papers/sigcomm/p2.pdf HTTP/1.1
   5 0.177819  64.238.147.113  192.168.1.122   TCP          66 80 → 60643 [ACK] Seq=1 Ack=192 Win=6864 Len=0 TSval=40…
   6 0.178321  64.238.147.113  192.168.1.122   TCP         311 [TCP segment of a reassembled PDU]
   7 0.178388  192.168.1.122   64.238.147.113  TCP          66 60643 → 80 [ACK] Seq=192 Ack=246 Win=524280 Len=0 TSva…
   8 0.189114  64.238.147.113  192.168.1.122   TCP        1434 [TCP segment of a reassembled PDU]
```

Figure 12: Viewing the complete trace

Below is a time sequence diagram of the three-way handshake in your trace, up to and including the first data packet (the HTTP GET request) sent by 'your computer' when the connection is established. As usual, time runs down the page, and lines across the page indicate segments.



Graph 1	Color	Filter:	tcp.srcport==80	Style: Line	☑ Smooth
Graph 2	Color	Filter:	tcp.dstport==80	Style: Line	☑ Smooth
Graph 3	Color	Filter:		Style: Line	☑ Smooth
Graph 4	Color	Filter:		Style: Line	☑ Smooth
Graph 5	Color	Filter:		Style: Line	☑ Smooth

X Axis: Tick interval: 0.1 sec; Pixels per tick: 10; ☐ View as time of day

Y Axis: Unit: Bits/Tick; Scale: Auto; Smooth: No filter

The initial SYN has no ACK number, only a sequence number. All subsequent packets have ACKnumbers.

The initial sequence numbers are shown as zero in each direction. This is because our Wireshark is configured to show relative sequence numbers. The actual sequence number is some large 32-bit number, and it is different for each end.

The ACK number is the corresponding sequence number plus 1. Our computer sends the third part of the handshake (the ACK) and then sends data right away in a different packet. It would be possible to combine these packets, but they are typically sepa- rate (because one is triggered by the OS and one by the application).

For the Data segment, the sequence number and ACK stay with the previous values. The sequence number will advance as the sender sends more data. The ACK number will advance as the sender receives more data from the remote server.

The three packets received and sent around 88ms happen very close together compared to the gap between the first and second packet. This is because they are local operations; there is no network delay involved.

Step 5: Connection Options

As well as setting up a connection, the TCP SYN packets negotiate parameters between the two ends using Options. Each end describes its capabilities, if any, to the other end by including the appropriate Options on its SYN. Often both ends must support the behavior for it to be used during data transfer.

Common Options include Maximum Segment Size (MSS) to tell the other side the largest segment that can be received, and Timestamps to include information on segments for estimating the round trip time. There are also Options such as NOP (No-operation) and End of Option list that serve to format the Op- tions but do not advertise capabilities. You do not need to include these formatting options in your an- swer above. Options can also be carried on regular segments after the connection is set up when they play a role in data transfer. This depends on the Option. For example: the MSS option is not carried on each packet because it does not convey new information; timestamps may be included on each packet to keep a fresh estimate of the RTT; and options such as SACK (Selective Acknowledgments) are used only when data is received out of order.

Our TCP Options are Maximum Segment Size, Window Scale, SACK permitted, and Timestamps. Each of these Options is used in both directions. There are also the NOP & End of Option List formatting options.

Here is an example of a FIN teardown:

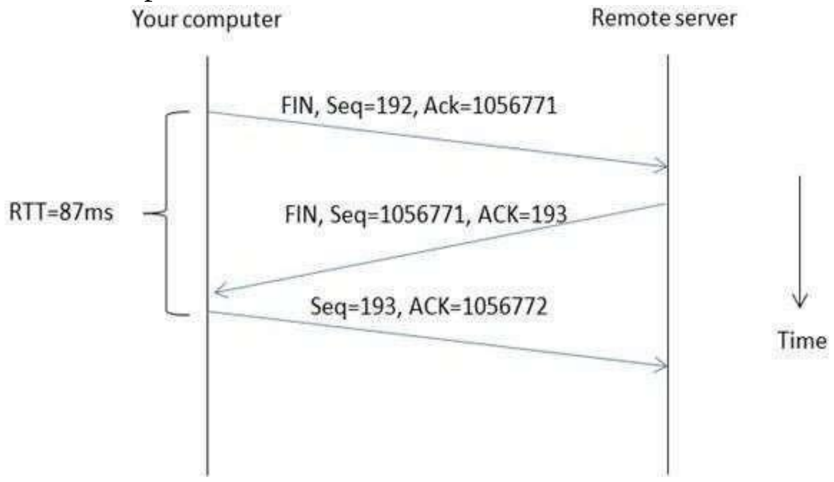


Figure 14: Time sequence diagram for FIN teardown

The teardown is initiated by the computer; it might also be initiated by the

Like the SYN, the FIN flag occupies one sequence number. Thus, when the sequence number ofthe FIN is 192, the corresponding Ack number is 193.

(

Your sequence numbers will vary. Our numbers are relative as computed by Wireshark) but clearly depend on the resource that is fetched. You can tell that it is around 1 MB long.

Points to note:
The RTT in the FIN exchange is like that in the SYN exchange, as it should be. Your RTT will vary depending on the distance between the computer and server as before.

Step 6: FIN/RST Teardown

Finally, the TCP connection is taken down after the download is complete. This is typically done with FIN (Finalize) segments. Each side sends a FIN to the other and acknowledges the FIN they receive; it is simi- lar to the three-way handshake. Alternatively, the connection may be torn down abruptly when one end sends a RST (Reset). This packet does not need to be acknowledged by the other side.

Below is a picture of the teardown in your trace, starting from when the first FIN or RST is issued untilthe connection is complete. It shows the sequence and ACK numbers on each segment.

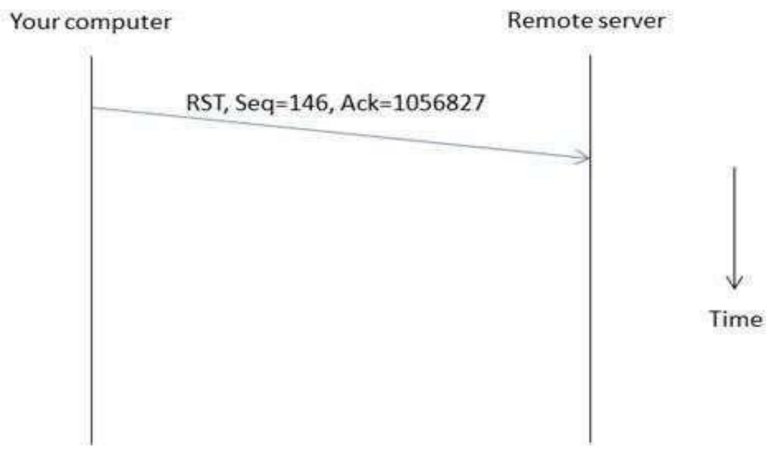


Figure 15: Time sequence diagram for RST teardown

The teardown is initiated by the computer; it might also be initiated by the

Points to note:

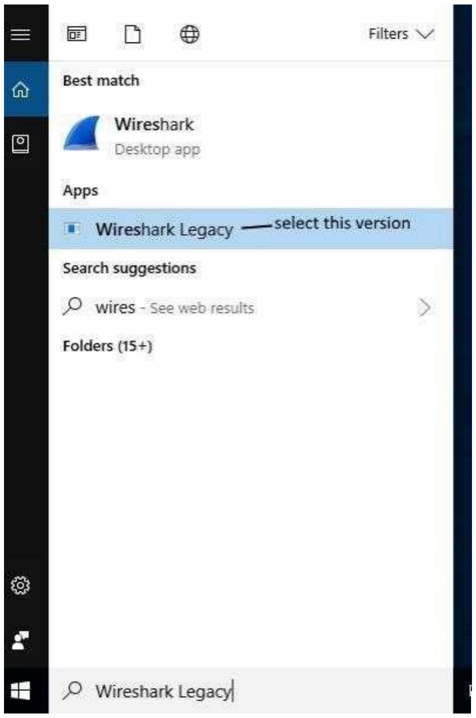
The teardown is abrupt – a single RST in this case, and then it is closed, which the other end must accommodate.

The sequence and Ack numbers do not really matter here. They are simply the (relativeWireshark) values at the end of the connection.

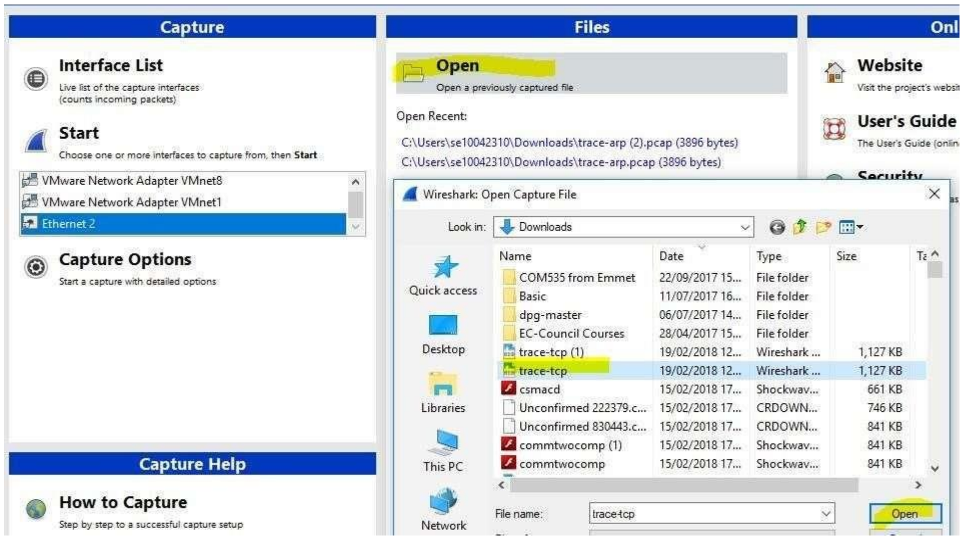
Since there is no round trip exchange, no RTT can be estimated.

Step 7: TCP Data Transfer

For this part, we are going to launch an older version of Wireshark called Wireshark legacy. You do thisby selecting the Wireshark legacy application as follows.



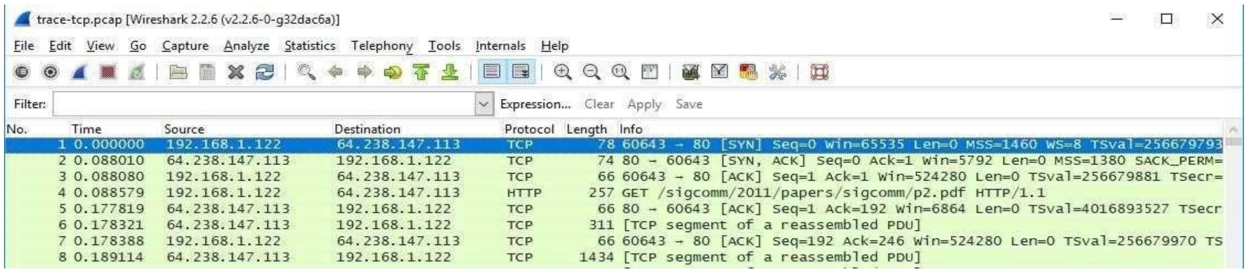
When it launches, you should open the trace-tcp file which is in your downloads folder from earlier.


Capture
Interface List
Live list of the capture interfaces (counts incoming packets)
Start
Choose one or more interfaces to capture from, then Start
VMware Network Adapter VMnet8
VMware Network Adapter VMnet1
Ethernet 2
Capture Options
Start a capture with detailed options
Capture Help
How to Capture
Step by step to a successful capture setup
Files
Open
Open a previously captured file
Open Recent:
Website
User's Guide
Security
Wireshark: Open Capture File
Look in: Downloads
Quick access
Desktop
Libraries
This PC
Network
File name: trace-tcp
Open

C:\Users\se10042310\Downloads\trace-arp (2).pcap (3896 bytes)
C:\Users\se10042310\Downloads\trace-arp.pcap (3896 bytes)

Name	Date	Type	Size
COM535 from Emmet	22/09/2017 15...	File folder	
Basic	11/07/2017 16...	File folder	
dpg-master	06/07/2017 14...	File folder	
EC-Council Courses	28/04/2017 15...	File folder	
trace-tcp (1)	19/02/2018 12...	Wireshark ...	1,127 KB
trace-tcp	19/02/2018 12...	Wireshark ...	1,127 KB
csmacd	15/02/2018 17...	Shockwav...	661 KB
Unconfirmed 223379.c...	15/02/2018 17...	CRDOWN...	746 KB
Unconfirmed 830443.c...	15/02/2018 17...	CRDOWN...	841 KB
commtwocomp (1)	15/02/2018 17...	Shockwav...	841 KB
commtwocomp	15/02/2018 17...	Shockwav...	841 KB

You should then be presented with the same trace-tcp as used previously in this exercise.



```
trace-tcp.pcap [Wireshark 2.2.6 (v2.2.6-0-g32dac6a)]
File  Edit  View  Go  Capture  Analyze  Statistics  Telephony  Tools  Internals  Help
Filter:                                   Expression...  Clear  Apply  Save
No.  Time      Source          Destination     Protocol Length Info
   1 0.000000  192.168.1.122   64.238.147.113  TCP        78 60643 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 TSval=256679793
   2 0.088010  64.238.147.113  192.168.1.122   TCP        74 80 → 60643 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1380 SACK_PERM=
   3 0.088080  192.168.1.122   64.238.147.113  TCP        66 60643 → 80 [ACK] Seq=1 Ack=1 Win=524280 Len=0 TSval=256679881 TSecr=
   4 0.088579  192.168.1.122   64.238.147.113  HTTP      257 GET /sigcomm/2011/papers/sigcomm/p2.pdf HTTP/1.1
   5 0.177819  64.238.147.113  192.168.1.122   TCP        66 80 → 60643 [ACK] Seq=1 Ack=192 Win=6864 Len=0 TSval=4016893527 TSecr
   6 0.178321  64.238.147.113  192.168.1.122   TCP       311 [TCP segment of a reassembled PDU]
   7 0.178388  192.168.1.122   64.238.147.113  TCP        66 60643 → 80 [ACK] Seq=192 Ack=246 Win=524280 Len=0 TSval=256679970 TS
   8 0.189114  64.238.147.113  192.168.1.122   TCP      1434 [TCP segment of a reassembled PDU]
```

The middle portion of the TCP connection is the data transfer, or download, in our trace. This is the mainevent. To get an overall sense of it, we will first look at the download rate over time.

Under the *Statistics menu select an "IO Graph"* (as shown below).

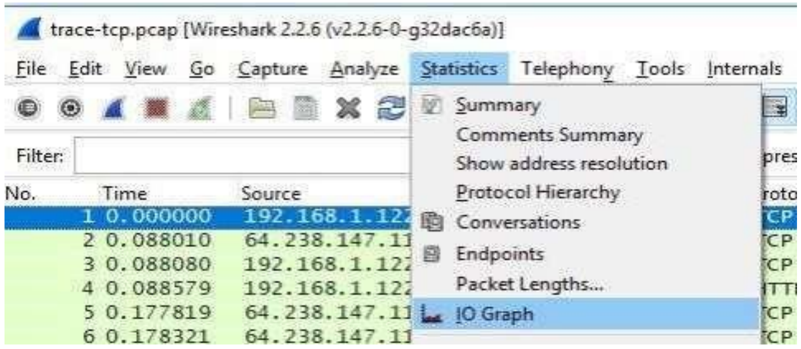

trace-tcp.pcap [Wireshark 2.2.6 (v2.2.6-0-g32dac6a)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals
Summary
Comments Summary
Show address resolution
Protocol Hierarchy
Conversations
Endpoints
Packet Lengths...
IO Graph

Figure 16: Opening an IO graph

Under the Statistics menu select an "IO Graph" (as shown below).

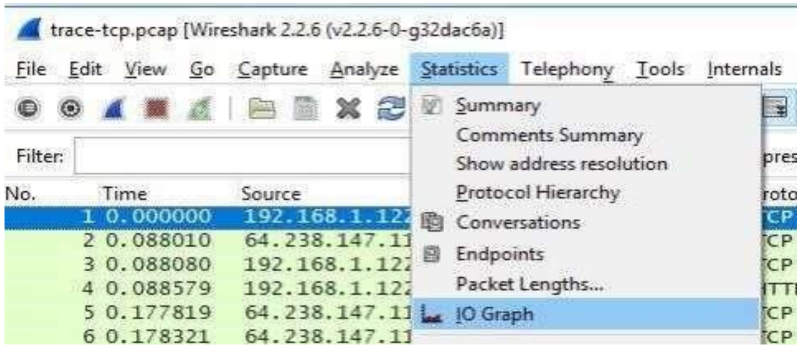
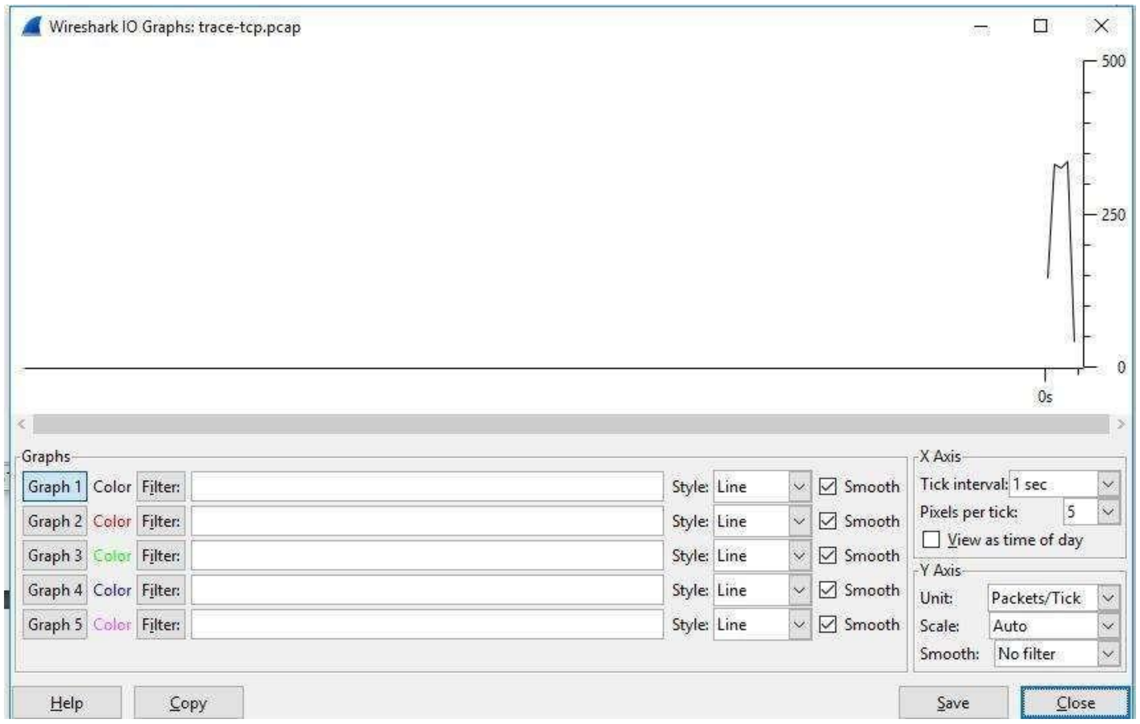

trace-tcp.pcap [Wireshark 2.2.6 (v2.2.6-0-g32dac6a)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals
Summary
Comments Summary
Show address resolution
Protocol Hierarchy
Conversations
Endpoints
Packet Lengths...
IO Graph

Figure 16: Opening an IO graph

You should end up with a graph like below. By default, this graph shows the rate of packets over time. You might be tempted to use the “TCP Stream Graph” tools under the Statistics menu instead. However,these tools are not useful for our case because they assume the trace is taken near the computer send- ing the data; our trace is taken near the computer receiving the data.



Style: Line ☑ Smooth

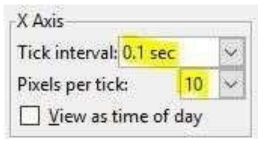
X Axis
Tick interval: 1 sec
Pixels per tick: 5
☐ View as time of day

Y Axis
Unit: Packets/Tick
Scale: Auto
Smooth: No filter

Figure 17: The IO graph

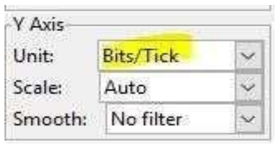
Now we will tweak it to show the download rate with the changes given below

On the x-axis, adjust the tick interval and pixels per tick. The tick interval should be small enoughto see into the behavior over the trace, and not so small that there is no averaging. 0.1 seconds is a good choice for a several second trace. The pixels per tick can be adjusted to make the graph wider or narrower to fill the window. Make this 10. See below



X Axis
Tick interval: 0.1 sec
Pixels per tick: 10
☐ View as time of day

On the y-axis, change the unit to be Bits/Tick. The default is Packet/Tick. By changing it, we caneasily work out the bits/sec throughput by taking the y-axis value and scaling as appropriate, e.g., 10X for ticks of 0.1 seconds.



Y Axis
Unit: Bits/Tick
Scale: Auto
Smooth: No filter

Add a filter expression to see only the download packets. So far we are looking at all of the pack-ets. Assuming the download is from the usual web server port of 80, you can filter for it with a filter of “tcp.srcport==80”. Don’t forget to press Enter, and you may need to click the “Graph” button to cause it to redisplay.

To see the corresponding graph for the upload traffic, enter a second filter in the next box. Againassuming the usual web server port, the filter is “tcp.dstport==80”. After you press Enter and click the Graph button, you should have two lines on the graph.

Our graph for this procedure is shown in the figure on next page. From it we can see the sample down- load rate quickly increase from zero to a steady rate, with a bit of an exponential curve. This is slow- start. The download rate when the connection is running is approximately 2.5 Mbps. The upload rate is asteady, small trickle of ACK traffic. Our download also proceeds fairly steadily until it is done. This is the ideal, but many downloads may display more variable behavior if, for example, the available bandwidth varies due to competing downloads, the download rate is set by the server rather than the network, or enough packets are lost to disrupt the transfer.

Note, you can click on the graph to be taken to the nearest point in the trace if there is a feature youwould like to investigate.

Try clicking on parts of the graph and watch where you are taken in the Wireshark trace window.

Since this is a download, the sequence number of transmitted segments will not increase (afterthe initial get). Thus the ACK number on received segments will not increase either.

Each segment carries Window information to tell the other end how much space remains in the buffer. The Window must be greater than zero, or the connection will be stalled by flow control.

Note the data rate in the download direction in packets/second and bits/second once the TCP connec-tion is running well is 250 packet/sec and 2.5 Mbps.

Our download packets are 1434 bytes long, of which 1368 bytes are the TCP payload carrying contents.Thus 95% of the download is content.

The data rate in the upload direction in packets/second and bits/second due to the ACK packets is 120 packets/sec and 60,000 bits/sec. We expect the ACK packet rate to be around half of the data packet rate for the typical pattern of one delayed ACK per two data packets received. The ACK bit rate will be at least an order of magnitude below the data bit rate as the packets are much smaller, around 60 bytes.

The Ack number tells the next expected sequence number therefore it will be X plus the number of TCPpayload bytes in the data segment.

RESULT:

Thus the above Installation of Wire shark, tcp dump and observe data transferred in client-server communication using UDP/TCP and identify the UDP/TCP datagram is installed and verified Successfully.

EX. NO: 5 DATE:	CHECK MESSAGE INTEGRITY AND CONFIDENTIALITY USING SSL.

AIM:

To implement the Check message integrity and confidentiality using SSL.

ALGORITHM:

1. **Setting up SSL/TLS:**
Use Java's `SSLSocket` and `SSLServerSocket` classes to establish a secureconnection between client and server.
2. **Ensuring Confidentiality:**
- Use SSL/TLS to encrypt the communication between client and server. Thisencryption ensures that the message content is secure from eavesdropping.
3. **Ensuring Integrity:**
- SSL/TLS provides integrity by using cryptographic hashing algorithms (likeHMAC) to verify that the transmitted data has not been altered during transmission.

PROGRAM:

Server

```
import javax.net.ssl.*;
import java.io.*;
import java.security.*;
public class Server {
public static void main(String[] args) {
 try
{
SSLServerSocketFactory serverSocketFactory = (SSLServerSocketFactory)
SSLServerSocketFactory.getDefault();
SSLServerSocket serverSocket = (SSLServerSocket)
serverSocketFactory.createServerSocket(9999);
SSLSocket sslSocket = (SSLSocket) serverSocket.accept();
// Read data from client
BufferedReader input = new BufferedReader(new
InputStreamReader(sslSocket.getInputStream()));
String clientMessage = input.readLine();
System.out.println("Received from client: " + clientMessage);
 // Close streams and socket
input.close();
sslSocket.close();
serverSocket.close();
}
 catch (IOException e)
{  e.printStackTrace();
}
}
}
```

Client

```
import javax.net.ssl.*;
import java.io.*;
import java.security.*;
public class Client {
public static void main(String[] args) {
try {
SSLSocketFactory sslSocketFactory = (SSLSocketFactory)
SSLSocketFactory.getDefault();
SSLSocket sslSocket = (SSLSocket) sslSocketFactory.createSocket("localhost",
9999);  // Send data to server
PrintWriter output = new PrintWriter(sslSocket.getOutputStream(), true);
output.println("Hello, server!");
output.close();
sslSocket.close();  }
catch (IOException e) {  e.printStackTrace();
}
}
}
```

OUTPUT:

Hello,server!

RESULT:

Thus the message integrity and confidentiality using SSL is executed Successfully

EX. NO: 6 **DATE:**	**EXPERIMENT EAVESDROPPING, DICTIONARY ATTACKS, MITMATTACKS**

AIM:

To study the Eavesdropping, Dictionary attacks, MITM attacks.

PROCEDURE:

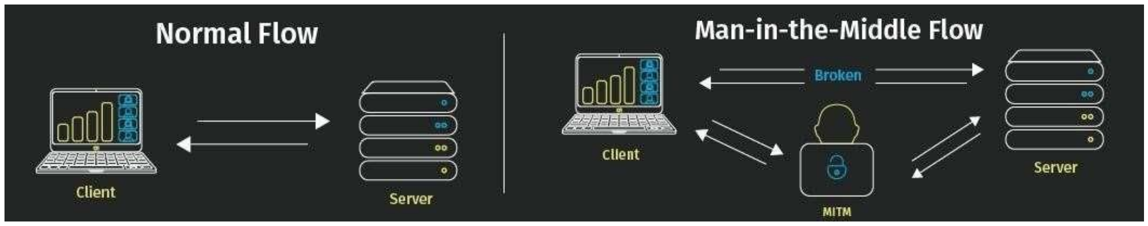
A man-in-the-middle attack is a type of eavesdropping attack, where attackers interrupt an existing conversation or data transfer. After inserting themselves in the "middle" of the transfer, the attackers pretend to be both legitimate participants. This enables an attacker to intercept information and data from either party while also sending malicious links or other information to both legitimate participants in a way that might not be detected until it is too late. You can think of this type of attack as similar to the game of telephone where one person's words are carried along from participant to participant until it has changed by the time it reaches the final person. In a man-in-the-middle attack, the middle participant manipulates the conversation unknown to either of the two legitimate participants, acting to retrieve confidential information and otherwise cause damage.

Man-in-the-middle attacks:

- Are a type of session hijacking
- Involve attackers inserting themselves as relays or proxies in an ongoing, legitimate conversation or data transfer
- Exploit the real-time nature of conversations and data transfers to go undetected
- Allow attackers to intercept confidential data
- Allow attackers to insert malicious data and links in a way indistinguishable from legitimate data

Examples of MITM Attacks

Although the central concept of intercepting an ongoing transfer remains the same, there are several different ways attackers can implement a man-in-the-middle attack.



Scenario 1: Intercepting Data

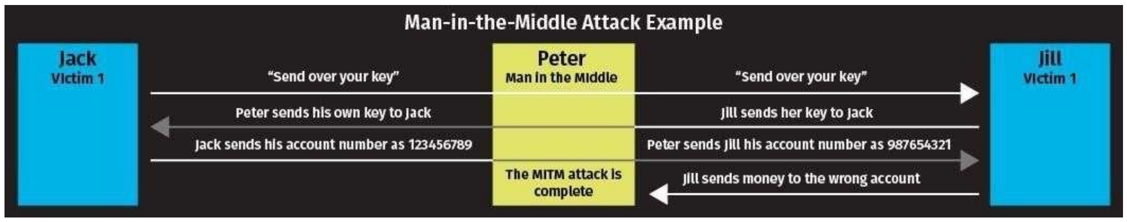
1. The attacker installs a packet sniffer to analyze network traffic for insecure communications.
2. When a user logs in to a site, the attacker retrieves their user information and redirects them to a fake site that mimics the real one.
3. The attacker's fake site gathers data from the user, which the attacker can then use on the real site to access the target's information.

In this scenario, an attacker intercepts a data transfer between a client and server. By tricking the client into believing it is still communicating with the server and the server into believing it is still receiving information from the client, the attacker is able to intercept data from both as well as inject their own false information into any future transfers.

Scenario 2: Gaining Access to Funds

1. The attacker sets up a fake chat service that mimics that of a well-known bank.
2. Using knowledge gained from the data intercepted in the first scenario, the attacker pretends to be the bank and starts a chat with the target.
3. The attacker then starts a chat on the real bank site, pretending to be the target and passing along the needed information to gain access to the target's account.

In this scenario, the attacker intercepts a conversation, passing along parts of the discussion to both legitimate participants.


Man-in-the-Middle Attack Example
Jack
Victim 1
Peter
Man in the Middle
Jill
Victim 1
"Send over your key"
"Send over your key"
Peter sends his own key to Jack
Jill sends her key to Jack
Jack sends his account number as 123456789
Peter sends Jill his account number as 987654321
The MITM attack is complete
Jill sends money to the wrong account

Real-World MITM Attacks

In 2011, Dutch registrar site DigiNotar was breached, which enabled a threat actor to gain access to 500 certificates for websites like Google, Skype, and others. Access to these certificates allowed the attacker to pose as legitimate websites in a MITM attack, stealing users' data after tricking them into entering passwords on malicious mirror sites. DigiNotar ultimately filed for bankruptcy as a result of the breach.

In 2017, credit score company Equifax removed its apps from Google and Apple after a breach resulted in the leak of personal data. A researcher found that the app did not consistently use HTTPS, allowing attackers to intercept data as users accessed their accounts.

Interactions Susceptible to MITM Attacks

Any improperly secured interaction between two parties, whether it's a data transfer between a client and server or a communication between two individuals over an internet messaging system, can be targeted by man-in-themiddle attacks. Logins and authentication at financial sites, connections that should be secured by public or private keys, and any other situation where an ongoing transaction could grant an attacker access to confidential information are all susceptible.

Other Forms of Session Hijacking

Man-in-the-middle attacks are only one form of session hijacking. Others include:

- Sniffing - An attacker uses software to intercept (or "sniff") data being sent to or from your device.

- Sidejacking - An attacker sniffs data packets to steal session cookies from your device, allowing them to hijack a user session if they find unencrypted login information.

- Evil Twin - An attacker duplicates a legitimate Wi-Fi network, enabling them to intercept data from users who believe they are signing on to the real network.

RESULT:

Thus the study of Eavesdropping, Dictionary attacks, MITM attacks is successfully completed.

EX. NO: 7 DATE:	EXPERIMENT WITH SNIFF TRAFFIC USING ARP POISONING

AIM:

To demonstrate Intrusion Sniff Traffic using ARP Poisoning.

Steps for Configuring ARP:

ARP is the acronym for Address Resolution Protocol. It is used to convert IP address to physical addresses [MAC address] on a switch. The host sends an ARP broadcast on the network, and the recipient computer responds with its physical address [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 genuine 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 large networks. 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.

Operating System Security: this measure is dependent on the operating system been used. The following are the basic techniques used by various operating systems.

- **Linux based**: these work by ignoring unsolicited ARP reply packets.

- **Microsoft Windows**: the ARP cache behavior can be configured via the registry. The following list includes some of the software that can be used to protect networks against sniffing;

- **AntiARP**– provides protection against both passive and active sniffing

• **Agnitum Outpost Firewall**–provides protection against passive Sniffing.
• **XArp**– provides protection against both passive and active sniffing

• **Mac OS**: ArpGuard can be used to provide protection. It protects against both active and passive sniffing.

Hacking Activity: Configure ARP entries in Windows

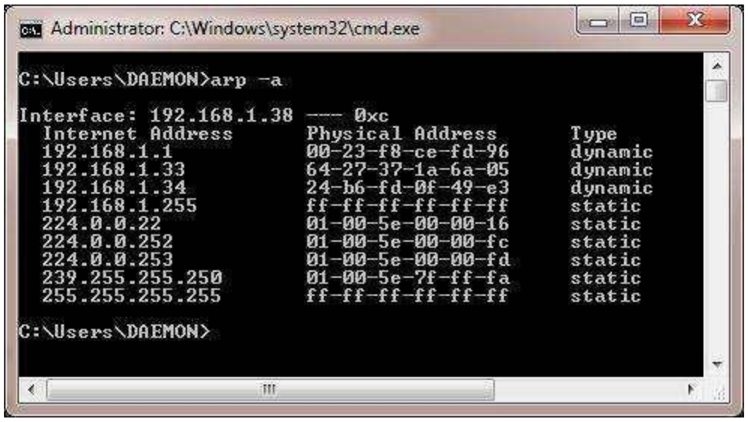
We are using Windows 7 for this exercise, but the commands should be able to work on other versions of windows as well.

Open the command prompt and enter the following command
arp –a

• **apr**calls the ARP configure program located in Windows/System32

• **-a** is the parameter to display to contents of the ARP cache

You will get results similar to the following.


Administrator: C:\Windows\system32\cmd.exe

```
C:\Users\DAEMON>arp -a

Interface: 192.168.1.38 --- 0xc
  Internet Address      Physical Address      Type
  192.168.1.1           00-23-f8-ce-fd-96     dynamic
  192.168.1.33          64-27-37-1a-6a-05     dynamic
  192.168.1.34          24-b6-fd-0f-49-e3     dynamic
  192.168.1.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  224.0.0.253           01-00-5e-00-00-fd     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

C:\Users\DAEMON>
```

directory.

Note: dynamic entries are added and deleted automatically when using TCP/IP sessions with remote computers.

Static entries are added manually and are deleted when the computer is restarted, and the network interface card restarted or other activities that affect it.

Adding static entries

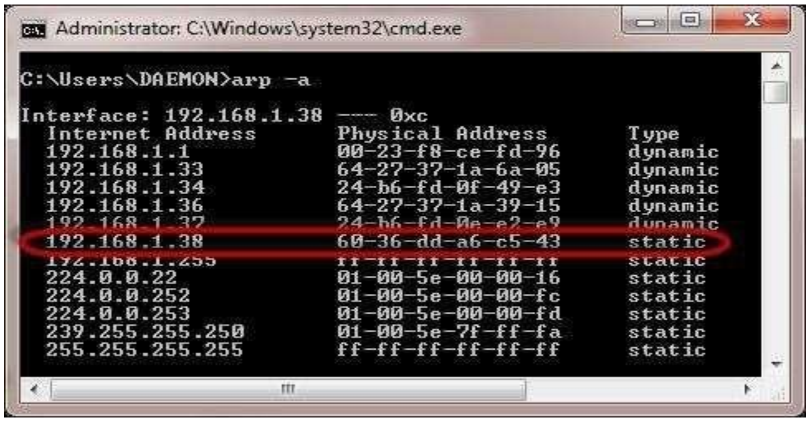
Open the command prompt then use the ipconfig /all command to get the IP and MAC address.

Note: The IP and MAC address will be different from the ones used here. This is because they are unique.

Use the following command to view the ARP cache

arp –a

You will get the following results



```
C:\Users\DAEMON>arp -a

Interface: 192.168.1.38 --- 0xc
  Internet Address      Physical Address      Type
  192.168.1.1           00-23-f8-ce-fd-96     dynamic
  192.168.1.33          64-27-37-1a-6a-05     dynamic
  192.168.1.34          24-b6-fd-0f-49-e3     dynamic
  192.168.1.36          64-27-37-1a-39-15     dynamic
  192.168.1.37          24-b6-fd-0e-e2-e9     dynamic
  192.168.1.38          60-36-dd-a6-c5-43     static
  192.168.1.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  224.0.0.253           01-00-5e-00-00-fd     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

Note the IP address has been resolved to the MAC address we provided and it is

RESULT:

Thus the Sniff Traffic using ARP Poisoning is demonstrated successfully.

EX. NO: 8 DATE:	DEMONSTRATION OF INTRUSION DETECTION SYSTEM (IDS).

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

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 of the "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: snortalerts.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)

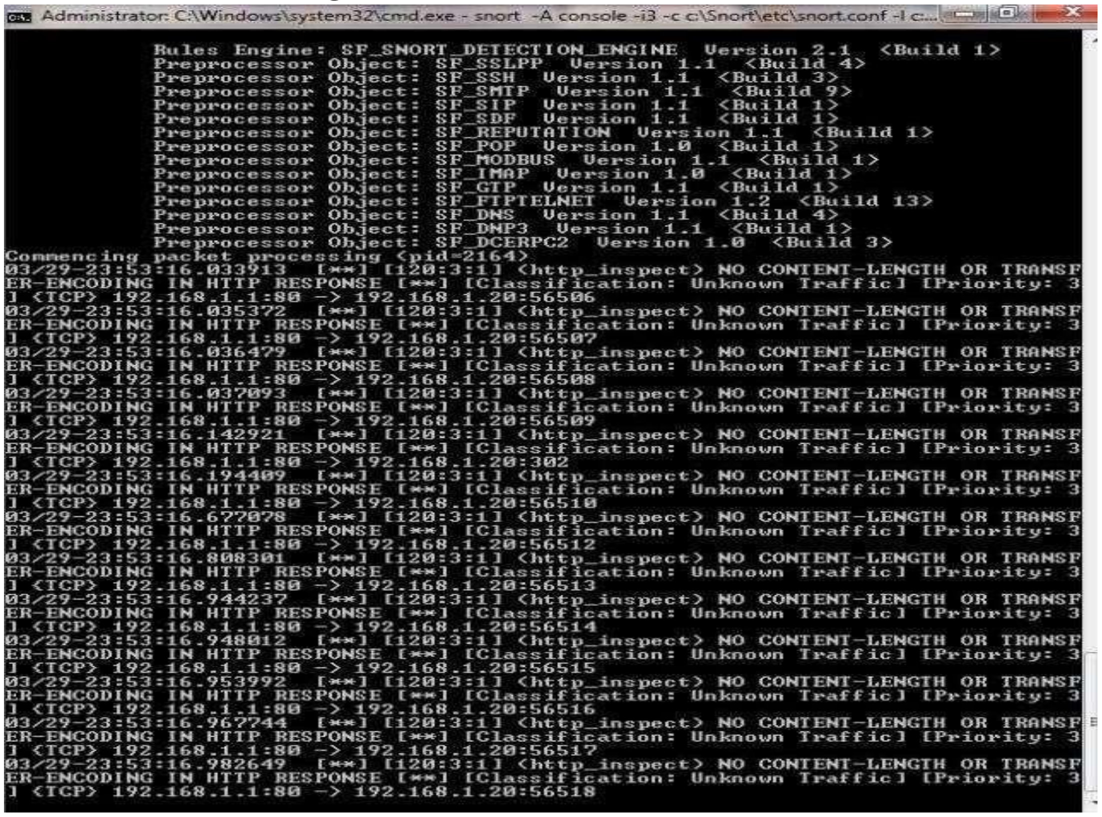
If a log is created, select the appropriate program to open it. You can use WordPard or NotePad++ to read the file.

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 –



```
Administrator: C:\Windows\system32\cmd.exe - snort  -A console -i3 -c c:\Snort\etc\snort.conf -l c...
           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 2.1  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
Commencing packet processing (pid=2164)
03/29-23:53:16.033913  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56506
03/29-23:53:16.035372  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56507
03/29-23:53:16.036479  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56508
03/29-23:53:16.037093  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56509
03/29-23:53:16.142921  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:302
03/29-23:53:16.194409  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56510
03/29-23:53:16.677078  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56512
03/29-23:53:16.808301  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56513
03/29-23:53:16.944237  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56514
03/29-23:53:16.948012  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56515
03/29-23:53:16.953992  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56516
03/29-23:53:16.967744  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56517
03/29-23:53:16.982649  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56518
```

RESULT

Thus the Intrusion Detection System (IDS) using Snort software tool was demonstrated sucessfully.

Ex. No: 9 **Date:**	**EXPLORE NETWORK MONITORING TOOLS**

AIM:

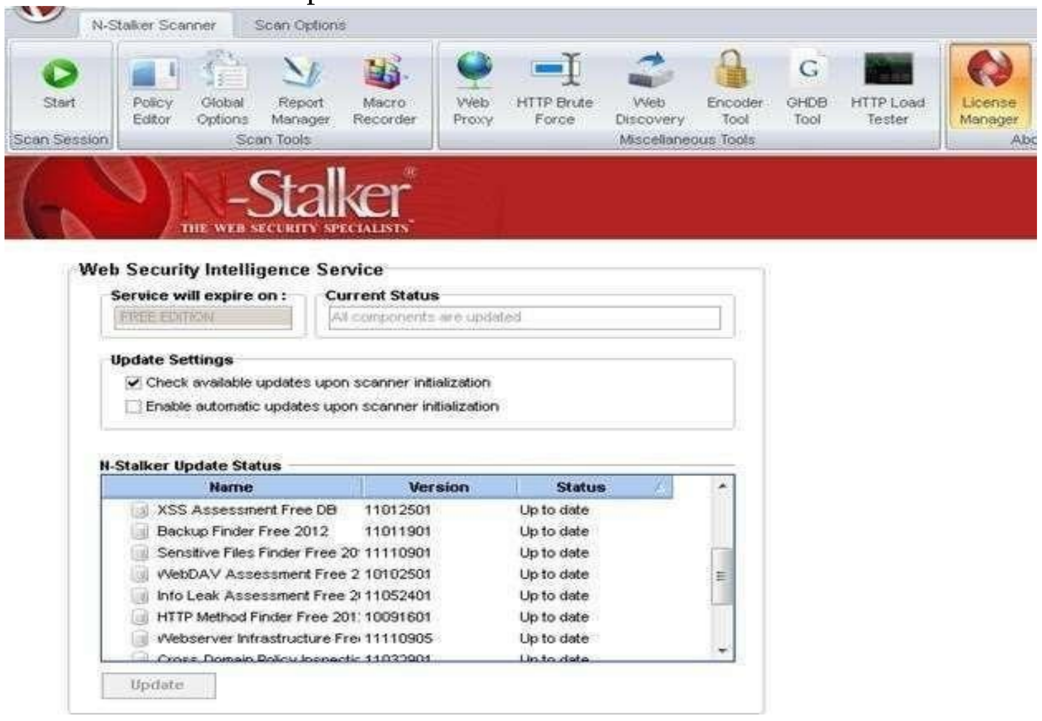
To download the N-Stalker Vulnerability Assessment Tool and exploring the features.

EXPLORING N-STALKER:

- N-Stalker Web Application Security Scanner is a Web security assessment tool.
- It incorporates with a well-known N-Stealth HTTP Security Scanner and 35,000 Web attack signature database.
- This tool also comes in both free and paid version.
- Before scanning the target, go to “License Manager” tab, perform the update.
- Once update, you will note the status as up to date.
- You need to download and install N-Stalker from www.nstalker.com.

1. Start N-Stalker from a Windows computer. The program is installedunder Start ➪ Programs ➪ N-Stalker ➪ N-Stalker Free Edition.
2. Enter a host address or a range of addresses to scan.
3. Click Start Scan.
4. After the scan completes, the N-Stalker Report Manager will prompt
5. you to select a format for the resulting report as choose Generate HTML.

6. Review the HTML report for vulnerabilities



Web Security Intelligence Service

Service will expire on : FREE EDITION

Current Status: All components are updated

Update Settings

[✓] Check available updates upon scanner initialization

[ ] Enable automatic updates upon scanner initialization

N-Stalker Update Status

Name	Version	Status
XSS Assessment Free DB	11012501	Up to date
Backup Finder Free 2012	11011901	Up to date
Sensitive Files Finder Free 20	11110901	Up to date
WebDAV Assessment Free 2	10102501	Up to date
Info Leak Assessment Free 2	11052401	Up to date
HTTP Method Finder Free 201	10091601	Up to date
Webserver Infrastructure Fre	11110905	Up to date
Cross Domain Policy Inspectic	11032901	Up to date

Now goto “Scan Session”, enter the target URL.

In scan policy, you can select from the four options,

• Manual test which will crawl the website and will be waiting for manual attacks.

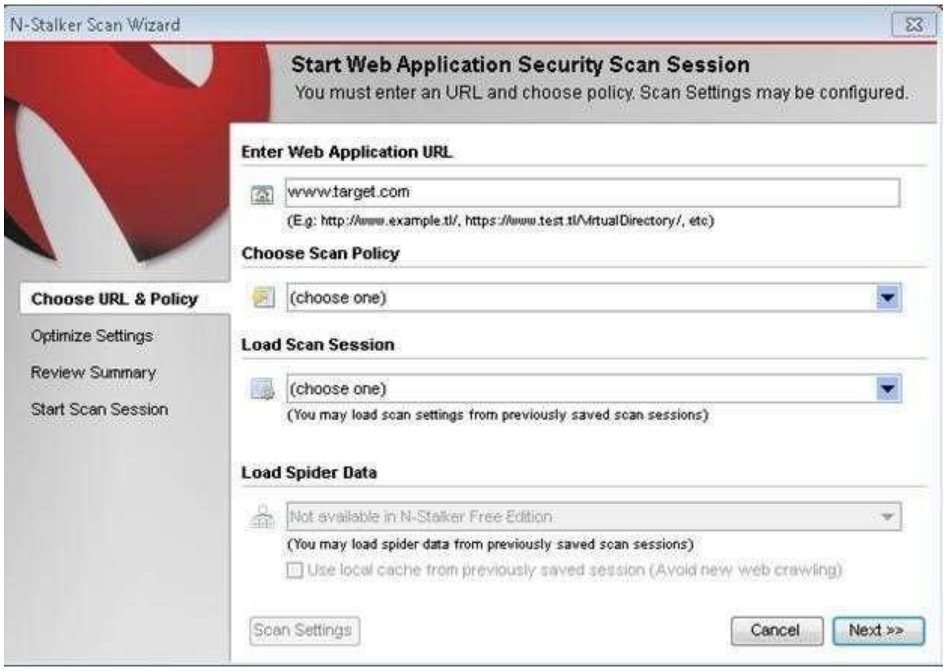
• full xss assessment

• owasp policy

• Web server infrastructure analysis.

Once, the option has been selected, next step is “Optimize settings” which will crawl the whole website for further analysis.

In review option, you can get all the information like host information, technologies used, policy name, etc.



N-Stalker Scan Wizard

Start Web Application Security Scan Session
You must enter an URL and choose policy. Scan Settings may be configured.

Enter Web Application URL

www.target.com

(E.g: http://www.example.tl/, https://www.test.tl/VirtualDirectory/, etc)

Choose Scan Policy

(choose one)

Load Scan Session

(choose one)

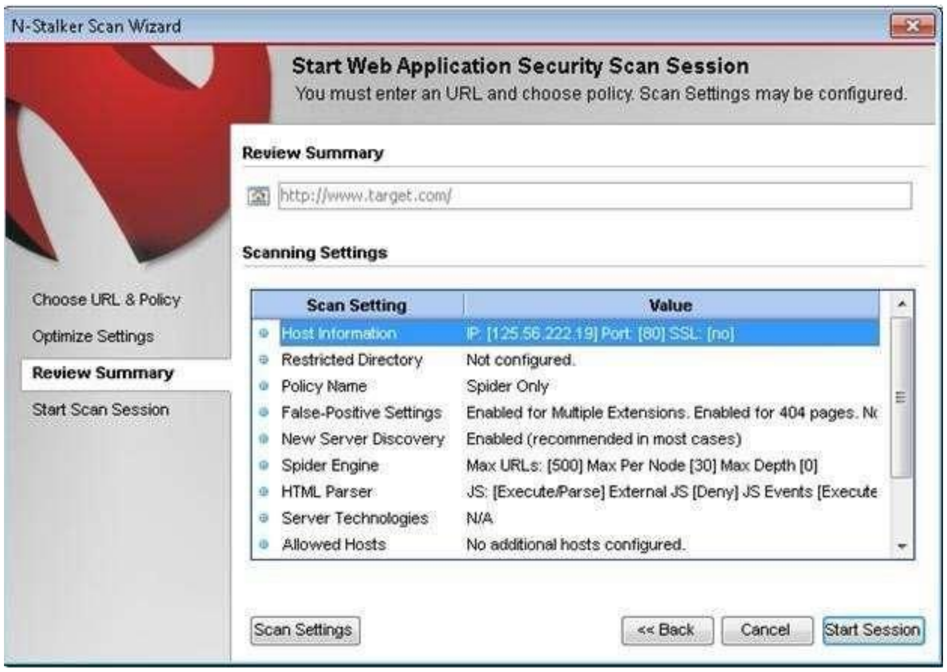
(You may load scan settings from previously saved scan sessions)

Load Spider Data

Not available in N-Stalker Free Edition

(You may load spider data from previously saved scan sessions)

☐ Use local cache from previously saved session (Avoid new web crawling)



N-Stalker Scan Wizard

Start Web Application Security Scan Session
You must enter an URL and choose policy. Scan Settings may be configured.

Review Summary

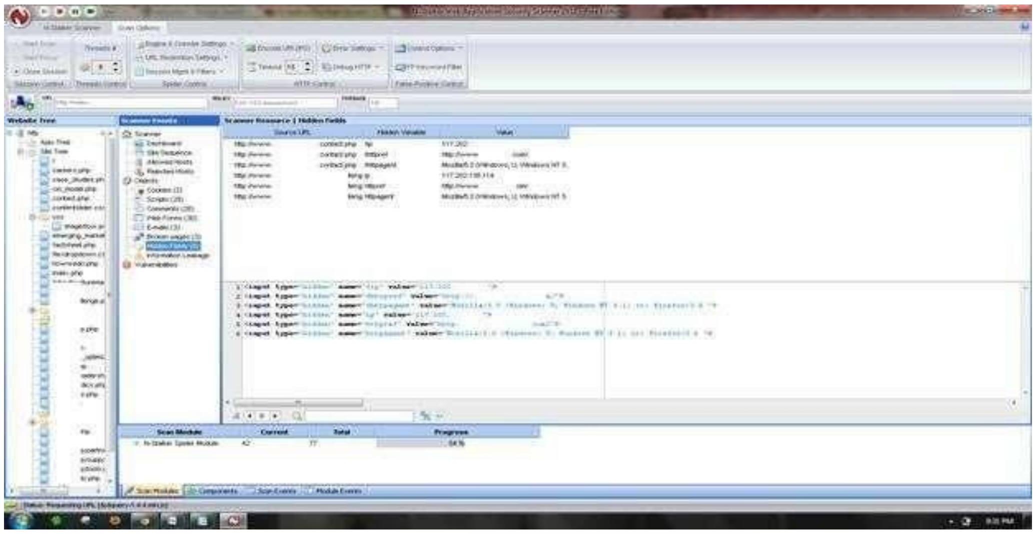
http://www.target.com/

Scanning Settings

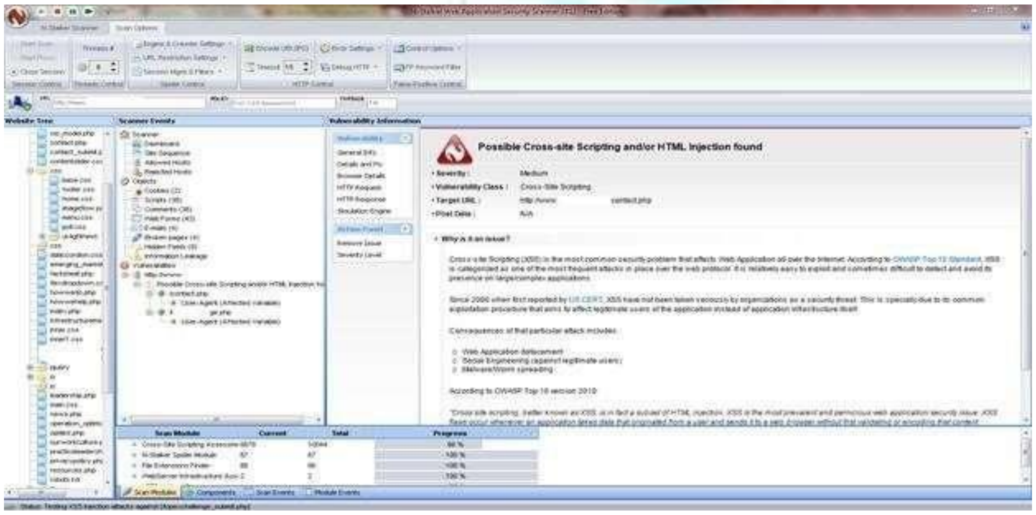
Scan Setting	Value
Host Information	IP: [125.56.222.19] Port: [80] SSL: [no]
Restricted Directory	Not configured.
Policy Name	Spider Only
False-Positive Settings	Enabled for Multiple Extensions. Enabled for 404 pages. N
New Server Discovery	Enabled (recommended in most cases)
Spider Engine	Max URLs: [500] Max Per Node [30] Max Depth [0]
HTML Parser	JS: [Execute/Parse] External JS [Deny] JS Events [Execute
Server Technologies	N/A
Allowed Hosts	No additional hosts configured.

Once done, start the session and start the scan.

The scanner will crawl the whole website and will show the scripts, broken pages, hidden fields, information leakage, web forms related information which helps to analyze further.



Once the scan is completed, the NStalker scanner will show details like severity level, vulnerability class, why is it an issue, the fix for the issue and the URL which is vulnerable to the particular vulnerability?


Possible Cross-site Scripting and/or HTML Injection found

• Severity : Medium
• Vulnerability Class : Cross-Site Scripting
• Target URL : http://www. contact.php
• Post Data : N/A

Why is it an issue?

[illegible]

Consequences of that particular attack includes:

- Web Application defacement
- Social Engineering (against legitimate users)
- Malware/Worm spreading

According to OWASP Top 10 version 2010

[illegible]

RESULT:

Thus the N-Stalker Vulnerability Assessment tool has been downloaded, installed and the features has been explored by using a vulnerable website.

EX. NO: 10 **DATE:**	**STUDY TO CONFIGURE FIREWALL, VPN**

Aim:

To configure the Firewall and VPN networks.

Configure firewall rules

When you configure Cloud VPN tunnels to connect to your peer network, review and modify firewall rules in your Google Cloud and peer networks to make sure that they meet your needs. If your peer network is another Virtual Private Cloud (VPC) network, then configure Google Cloud firewall rules for both sides of the network connection.

Google Cloud firewall rules

Google Cloud firewall rules apply to packets sent to and from virtual machine (VM) instances within your VPC network and through Cloud VPN tunnels.

The implied allow egress rules allow VM instances and other resources in your Google Cloud network to make outgoing requests and receive established responses. However, the implied deny ingress rule blocks all incoming traffic to your Google Cloud resources.

At a minimum, create firewall rules to allow ingress traffic from your peer network to Google Cloud. If you created egress rules to *deny* certain types of traffic, you might also need to create other egress rules.

Traffic containing the protocols UDP 500, UDP 4500, and ESP (IPsec, IP protocol 50) is always allowed to and from one or more external IP addresses on a Cloud VPN gateway. However, Google Cloud firewall rules do *not* apply to the post- encapsulated IPsec packets that are sent from a Cloud VPN gateway to a peer VPN gateway.

Example configurations

For multiple examples of restricting ingress or egress traffic, see the configuration examples in the VPC documentation.
The following example creates an *ingress allow* firewall rule. This rule permits all TCP, UDP, and ICMP traffic from your peer network's CIDR to your VMs in your VPC network.

Permissions required for this task

1. In the Google Cloud console, go to the **VPN tunnels** page.
Go to VPN tunnels
2. Click the VPN tunnel that you want to use.
3. In the **VPN gateway** section, click the name of the VPC network. This action directs you to the **VPC network details** page that contains the tunnel.
4. Click the **Firewall rules** tab.
5. Click **Add firewall rule**. Add a rule for TCP, UDP, and ICMP:
- **Name:** Enter allow-tcp-udp-icmp.
- **Source filter:** Select **IPv4 ranges**.
- **Source IP ranges:** Enter a **Remote network IP range** value from when you created the tunnel. If you have more than one peer network range, enter each one. Press the **Tab** key between entries. To allow traffic from all source IPv4 addresses in your peer network, specify 0.0.0.0/0.
- **Specified protocols or ports:** Select tcp and udp.
- **Other protocols:** Enter icmp.
- **Target tags:** Add any valid tag or tags.

6. Click **Create**.
If you need to allow access to IPv6 addresses on your VPC network from your peer network, add an allow-ipv6-tcp-udp-icmpv6 firewall rule.

1. Click **Add firewall rule**. Add a rule for TCP, UDP, and ICMPv6:
- **Name:** Enter allow-ipv6-tcp-udp-icmpv6.
- **Source filter:** Select **IPv6 ranges**.
- **Source IP ranges:** Enter a **Remote network IP range** value from when you created the tunnel. If you have more than one peer network range, enter each one.

Press the **Tab** key between entries. To allow traffic from all source IPv6 addresses in your peer network, specify::/0.
- **Specified protocols or ports:** Select tcp and udp.

- **Other protocols**: Enter 58. 58 is the protocol number for ICMPv6.
- **Target tags:** Add any valid tag or tags.

2. Click **Create**.

Create other firewall rules if necessary.

Alternatively, you can create rules from the Google Cloud console **Firewall** page.

Peer firewall rules

When configuring your peer firewall rules, consider the following:

- Configure rules to allow egress and ingress traffic to and from the IP ranges used by the subnets in your VPC network.

- You can choose to permit all protocols and ports, or you can restrict traffic to only the necessary set of protocols and ports to meet your needs.

- Allow ICMP traffic if you need to use ping to be able to communicate among peer systems and instances or resources in Google Cloud.

- If you need to access IPv6 addresses on your peer network with ping, allow ICMPv6 (IP protocol 58) in your peer firewall.

- Both your network devices (security appliances, firewall devices, switches, routers, and gateways) and software running on your systems (such as firewall software included with an operating system) can implement on-premises firewall rules. To allow traffic, configure all firewall rules in the path to your VPC network appropriately.
- If your VPN tunnel uses dynamic (BGP) routing, make sure that you allow BGP traffic for the link-local IP addresses. For more details, see the next section.

BGP considerations for peer gateways

Dynamic (BGP) routing exchanges route information by using TCP port 179. Some VPN gateways, including Cloud VPN gateways, allow this traffic automatically when you choose dynamic routing. If your peer VPN gateway does not, configure it to allow incoming and outgoing traffic on TCP port 179.

All BGP IP addresses use the link-local 169.254.0.0/16 CIDR block. If your peer VPN gateway is not directly connected to the internet, make sure that it and peer routers, firewall rules, and security appliances are configured to at least pass BGP traffic (TCP port 179) and ICMP traffic to your VPN gateway. ICMP is not required, but it is useful to test connectivity between a Cloud Router and your VPN gateway. The range of IP addresses to which your peer firewall rule should apply must include the BGP IP addresses of the Cloud Router and your gateway.

RESULT:

Thus the study of Firewall and VPN is demonstrated successfully.