National University of Computer and Emerging Sciences

Laboratory Manuals

for

Computer Networks - Lab

(CL -3001)

Course Instructor	Sir Nauman Moazzam Hayat
Lab Instructor(s)	Mr. Usama Khan Mr. Muzammil Muneer
Section	BCS-6B
Semester	Spring 2023

Department of Computer Science
FAST-NU, Lahore, Pakistan

Lab Manual 06

Objective:

- Introduction to Wireshark
- Analyzing the HTTP packets using Wireshark

Introduction to Wireshark

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.

- **Running Wireshark**

When you run the Wireshark program and start capturing the internet traffic onto a specific interface (e.g., Ethernet, WiFi), the Wireshark graphical user interface will be displayed.

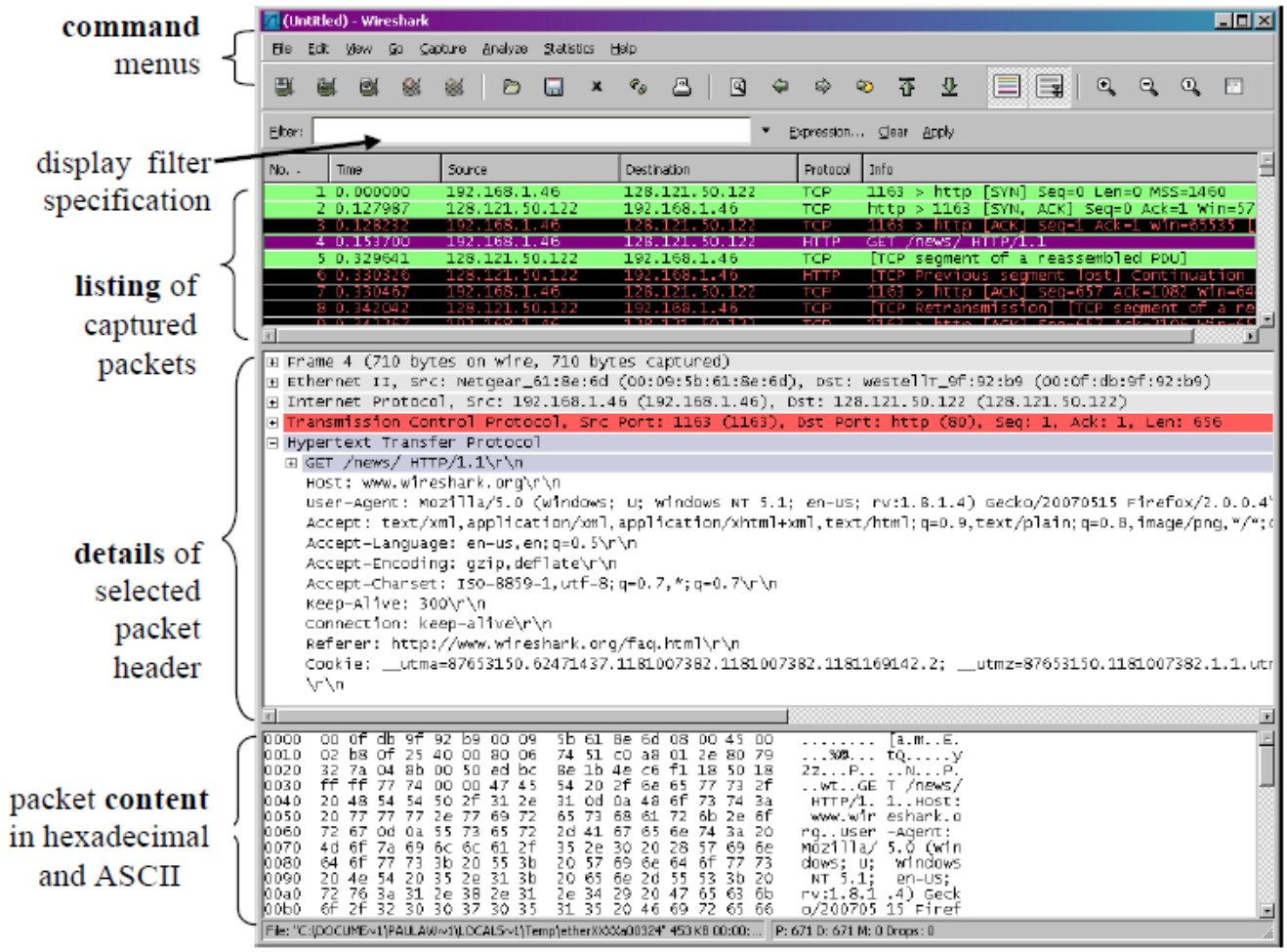

Figure: Wireshark Graphical User Interface

Taking Wireshark for a Test Run:

1. Open Wireshark, select interface from list of interfaces (Ethernet in your case). Then Press the capture Start button to start capturing the internet traffic packets at run time.
   If none of the interface is working you can use the Ethereal Packet Traces given in the Traces folder to answer the questions below (In this case you don't need to go through Steps 2 to 4 to capture the internet traffic as it is already done in the corresponding trace. To open the traffic in a trace go to File->Open and then select corresponding trace file)

2. While Wireshark is running, enter the URL: http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html and have that page displayed in your browser.

3. Now enter another URL http://gaia.cs.umass.edu/favicon.ico and you will see that this page is not found on the server.

4. In order to display both the pages, your browser will contact the HTTP server at gaia.cs.umass.edu and exchange HTTP messages with the server in order to download this page. The Ethernet frames containing these HTTP messages will be captured by Wireshark.

Note:

Make a Word file and post the screen shots of all the answers in it. Apart from the answers explore different settings of Wireshark and analyze all the layers of the Packets. Implement different filters in your data to view different grouping of packets. Make yourself familiar with the use of Wireshark.

Wireshark Filters:

- **Application Layer**: http, https
- **Transport Layer:** udp, udp.port==80, udp.srcport==443, udp.dstport==53
  tcp, udp.port==80, tcp.srcport==443, tcp.dstport==53
- **Network Layer:** ip.addr==192.168.1.2, ip.src==127.1.1.1, ip.dst==129.1.1.1
- **Data-Link Layer:** eth.addr == 00:00:5e:00:53:00, eth.src == 00:00:5e:00:53:00, eth.dst == 00:00:5e:00:53:00

To make combinations of different filters, you can use (&&, ||, !) signs as used in C++ e.g.,

a. http && tcp.port==80
b. tcp || ip.addr=192.168.1.2
c. !(ip.src==192.168.1.2)

In-Lab Statement 1: *Analyzing HTTP Protocol*

(10)

The HTTP GET/Response Interaction

Use the http-ethereal-trace-1 packet trace to answer the questions below apply the "http" filter

a. You have packet data that contains all protocol messages exchanged between your computer and other network entities! The HTTP message exchanges with the gaia.cs.umass.edu web server should appear somewhere in the listing of packets captured. But there will be many other types of packets displayed as well.

b. Type in **"http"** (without the quotes, and in lower case – all protocol names are in lower case in Wireshark) into the display filter specification window at the top of the main Wireshark window. Then select Apply (to the right of where you entered "http"). **This will cause only HTTP message to be displayed in the packet-listing window.**

c. Select the first http message shown in the packet-listing window. This should be the **HTTP GET message that was sent from your computer to the gaia.cs.umass.edu HTTP server**. When you select the HTTP GET message, the Ethernet frame, IP datagram, TCP segment, and HTTP message header information will be displayed in the packet-header window3. By clicking plus and- minus boxes to the left side of the packet details window, minimize the amount of Frame, Ethernet, Internet Protocol, and Transmission Control Protocol information displayed. Maximize the amount information displayed about the HTTP protocol. Your Wireshark display should now look roughly as shown in the figure above.

d. Now try to find out the packet which contains the second request you sent to the browser and also analyze the packet which your browser received as a result of second GET Request and answer the following questions:

1- List up to **4 different protocols** that appear in the protocol column in the unfiltered packet-listing window.
2- How long did it take from when the HTTP GET message was sent until the HTTP OK reply was received?
3- Was the second Get Request successful? How can you tell it from the corresponding response packet?

By looking at the information in the HTTP GET and Response Messages for **BOTH the HTTP Requests**, answer the following questions

4. Is your **browser** running HTTP version 1.0 or 1.1? What **version** of HTTP is the server running?
5. What **languages** (if any) does your **browser** indicate that it can accept to the server?
6. What is the **IP address** of the gaia.cs.umass.edu server and your computer?
7. What is the **MAC address** of the server and your computer?
8. What is sending and receiving **Port Number**? What does Port No. 80 represents?
9. What is the **status code** returned from the server to your browser?
10. When was the HTML file, that you are retrieving, **last modified** at the **server?**
11. How many bytes of total **packet content** are being returned to your browser?

The HTTP CONDITIONAL GET/response interaction

Use the http-ethereal-trace-2 packet trace to answer the questions below and apply the "http" filter

Answer the following questions:

1. Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE" line in the HTTP GET?

2. Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell from the Packet Bytes Window?

3. Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE:" line in the HTTP GET? If so, what information follows the "IF-MODIFIED-SINCE:" header? **What is meant by this information?**

4. What is the **HTTP status code** and phrase returned from the server in response to this **second HTTP GET**? Did the server explicitly return the contents of the file? Explain your answer

In-Lab Statement 2 : *Analyzing HTTP Protocol*

(10)

Retrieving Long Documents

In our examples thus far, the documents retrieved have been simple and short HTML files. Let's next see what happens when we **download a long HTML file**. Do the following:

In the packet-listing window, you should see your HTTP GET message, followed by a **multiple-packet TCP response** to your HTTP GET request. This multiple-packet response deserves a bit of explanation. The **HTTP RESPONSE MESSAGE** consists of a status line, followed by header lines, followed by a blank line, followed by the entity body. In the case of our HTTP GET, the entity body in the response is the *entire* requested HTML file. In our case here, the HTML file is rather long, and at **4500 bytes is too large to fit in one TCP packet.** The single HTTP response message is thus broken into several pieces by TCP, with each piece being contained within a separate TCP segment. In recent versions of Wireshark, Wireshark indicates each **TCP segment as a separate packet**, and the fact that the single HTTP response was fragmented across multiple TCP packets is indicated by the **"TCP segment of a reassembled PDU"** in the Info column of the Wireshark display.

- **Use the http-ethereal-trace-3 packet trace to answer the questions below and apply the "http" filter**

Answer the following questions:

5. How many HTTP GET request messages did your browser send?
6. Which **packet number** in the trace contains the GET message for **The Bill of Rights**?
7. Which **packet number** in the trace contains the status code and phrase associated with the response to the HTTP GET request?
8. What is the status code and phrase in the response?

9. How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights? What are the numbers of those packets?

In-Lab Statement 3: Trick Question (10)

What is the length of the text for The Bill of Rights in bytes? How do you justify this length of text when your Response Packet Size is only 490 bytes? Give complete explanation how the length of text in various packets add up to a total of 4500 Bytes.