Hypertext Transfer Protocol

In this exercise, you will be inspecting HTTP packets to explore their content and relate these to the protocol specifications defined in the HTTP RFC.

Part 1: Experience an HTTP Session (Optional but recommended)

1. Run Wireshark and set to capture from your network connection
2. Using any browser, visit http://www.ucla.edu/.
3. Stop the Wireshark transfer and filter the packets such that only HTTP traffic involving the UCLA server is shown (ip.addr == 164.67.228.152 && tcp.port == 80)

Part 2: Examine HTTP Packets

Note: If you did not perform Part 1, download the wireshark capture file HTTP.pcapng from the assignment details. This is a capture file of an HTTP session between a client (192.168.1.60) and the UCLA web server

4. The filtered packets will start with the series of 3-way handshakes being performed at the beginning of the HTTP session with the server. It is highly likely that more than one handshake was initiated by the client to the server.

   In your capture file, how many simultaneous TCP connections were initiated by the client at the onset of the HTTP session?

   6

5. Examine the first SYN packet and examine its TCP properties.

   Which is the source TCP port of the packet?

   23802

6. Right click on the first SYN packet then click Follow TCP stream. This should open a separate window that allows you to see all data transmitted between client and server using the same TCP connection. Text in red represents data sent from client to server, while those in blue are from server to client

7. Examine the first HTTP message. This should be the request message to retrieve the home page of the UCLA website.

   What HTTP operation / method is requested in the message and what is the purpose of this method?

   GET

   What resource was requested by the client?

   http://www.ucla.edu/

   Based on the request what browser and language is used by the client?

   The browser and language used by the client is Firefox and US English.

   What type and encoding can be accepted by the client for the expected content of the reply?

   Accept-Encoding: gzip, deflate\r\n

8. Immediately after the request, the corresponding response will be shown in blue. Examine the contents of the response.

   What is the response code and what does it signify?

   The response code is 200 and it signifies that the request was successful, and the server is returning the requested resource.

   What is the type and encoding of the content sent back by the server? Do these match those expected by the client based on the earlier request?

   Accept-Encoding: gzip, deflate\r\n Yes they do match those expected by the client based on the earlier request.

   What do you think is the content returned by the server in its response?

   I think the content returned by the server in its response is the

9. Scroll down further and examine the rest of the client and server exchange in the stream.

   How many request and response pairs did you find?

   30 request and response pairs

   Why did multiple HTTP requests use the same TCP connection? Hint: Check the Connection type used in the message headers (2pts).

   They use the same TCP connection to

10. Go back to viewing all TCP packets between client and server by reapplying the filter for web traffic (ip.addr == 164.67.228.152 && tcp.port == 80)

11. Click on the 2nd SYN packet.

    What is the source port used for connection establishment this time?

    23803

12. View the TCP stream for the 2nd connection this time and examine the data exchanged.

    Briefly describe how the contents are different from those in the data exchanged in the 1st connection.

    The contents are different from those in the data exchanged in the 1st connection is with the checksum and the stream index number.

13. Do the same for the 3rd connection if your packet capture contains it.

    What do you think is the likely reason for the browser using multiple TCP connections simultaneously when loading webpages?

    The likely reason for using multiple TCP connections is to improve webpage load performance by enabling parallel downloads, overcoming protocol limitations, and reducing the impact of latency.