WireShark Lab 1 – HTTP

Having gotten our feet wet with the WireShark packet sniffer in the introductory lab, we're now ready to use WireShark to investigate protocols in operation. In this lab, we'll explore several aspects of the HTTP protocol: the basic GET response interaction, HTTP message formats, retrieving large HTML files, retrieving HTML files with embedded objects, and HTTP authentication and security.

1. The Basic HTTP GET response interaction

Let's begin our exploration of HTTP by downloading a very simple HTML file - one that is very short, and contains no embedded objects. Do the following:
A. Start up your web browser.
B. Start up the WireShark packet sniffer, as described in the introductory lab (but don't start packet capture yet). Enter "http" (just the letters, not the quotation marks) in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window. (We're only interested in the HTTP protocol here, and don't want to see the clutter of all captured packets).
C. Enter the following to your browser: **http://www.rishiheerasing.net/wireshark/file1.html**
Press Enter. Your browser should display the very simple, one-line HTML page.
D. Stop WireShark packet capture.
Your WireShark window should look similar to the window shown in Fig. 1 below:

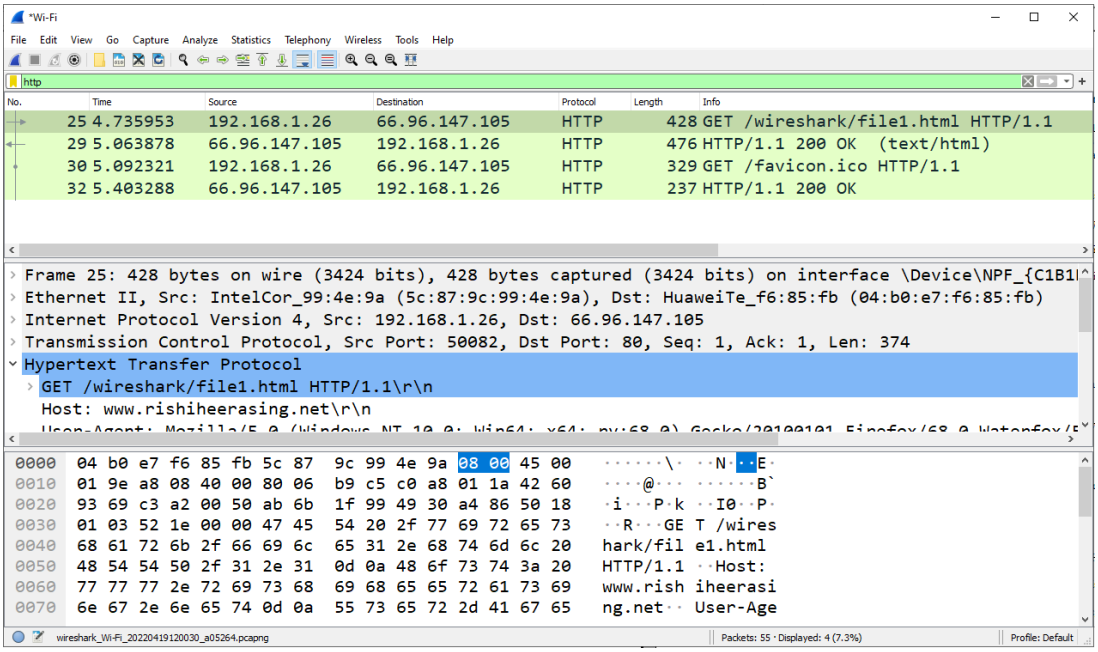

Fig. 1

The example in Fig. 1 shows in the packet-listing window that four HTTP messages were captured: the GET message (from your browser to the **www.rishiheerasing.net** web server) and the response message from the server to your browser. There are also two additional messages that are captured but this is dependent on your browser you used. The packet-contents window shows details of the selected message (in this case the HTTP GET message, which is highlighted in the packet-listing window). Recall that since the HTTP message was carried inside a TCP segment, which was carried inside an IP datagram, which was carried within an Ethernet frame, WireShark displays the Frame, Ethernet, IP, and TCP packet information as well. We want to minimize the amount of non-HTTP data displayed (we are interested in HTTP here, and will be investigating the other protocols is later labs), so make sure the boxes at the far left of the Frame, Ethernet, IP and TCP information have an greater than sign (>) (meaning there is hidden, not displayed information), and the HTTP line has a down arrow (v) (all information about the HTTP message is displayed).

By looking at the information in the HTTP request/response messages, **answer the following questions**. When answering the following questions, you should print out the request/response messages (see the WireShark Lab 0 for an explanation of how to do this) and indicate where in the message you've found the information. **(The questions below pertains to the first two messages captured only)**

1. Is your browser running HTTP version 1.0 or 1.1?
2. Which version of HTTP is the server running?
3. What languages (if any) does your browser indicate that it can accept to the server?
4. What is the IP address of your computer? Of the www.rishiheerasing.net server?
5. What is the status code returned from the server to your browser?
6. When was the HTML file you just retrieved last modified on the server?
7. How many bytes of content are being returned to your browser?

2. The HTTP CONDITIONAL GET/response interaction

Recall that most web browsers perform object caching and thus perform a conditional GET when retrieving an HTTP object. Before performing the steps below, make sure your browser's cache is empty. (For Google Chrome, enter the following in the address bar and press Enter: chrome://settings/privacy; then click Clear Browsing Data and ensure Cached Images and Files is ticked and click Clear Data. Now do the following:

A. Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
B. Start up the WireShark packet sniffer like before.
C. Enter the following URL into your browser: http://www.rishiheerasing.net/wireshark/file2.html
Press Enter. Your browser should display a very simple one line HTML page.
D. Once the page has loaded, reload the same page by clicking the Refresh icon on your browser or press F5.
E. Stop WireShark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.

Answer the following questions:

8. Inspect the **first** HTTP GET request contents from your browser to the server. Do you see an "IF-MODIFIED-SINCE" line in the HTTP GET message?
9. Inspect the **first** server response. Did the server explicitly return the contents of the file? How can you tell?
10. Now inspect the **second** HTTP GET request contents. Do you see an "IF-MODIFIED-SINCE:" line in the HTTP GET message? If so, what information follows the "IF-MODIFIED-SINCE:" header?
11. 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.

3. 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:
A. Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
B. Start up the WireShark packet sniffer
C. Enter the following URL into your browser: http://www.rishiheerasing.net/wireshark/file3.html
Press Enter. Your browser should display the rather lengthy UTM Act 2002.
D. Stop WireShark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed.
In the packet-listing window, you should see your HTTP GET message, followed by a multiple-packet response to your HTTP GET request. This multiple-packet response deserves a bit of explanation. Recall that the HTTP response

message consists of a status line, followed by header lines, followed by a blank line, followed by the payload. In the case of our HTTP response, the payload is the *entire* requested HTML file. The HTML content is rather large at 45,652 bytes. This is too large to fit in one single 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. Each TCP segment is recorded as a separate packet by WireShark, and the fact that the single HTTP response was fragmented across multiple TCP packets is indicated by the "Continuation" phrase displayed by WireShark.

Answer the following questions:

12. How many HTTP GET request messages were sent by your browser?
13. How many data-containing TCP segments were needed to carry the single HTTP response?
14. What is the status code and phrase associated with the response to the HTTP GET request?
15. Are there any HTTP status lines in the data associated with a TCP induced "Continuation"?

4 HTTP Authentication

Finally, let's try visiting a web page that is password-protected and examine the sequence of HTTP message exchanged for such a site. The password protected page is actually the Post-Semester Site on my webpage.
The password is "**cndm**" (without the quotes). So let us access this "secure" password-protected page.

Do the following:
A. Start up your browser
B. Go to the following page in your browser: **http://www.rishiheerasing.net/protectedpost/login.php**
C. Once the page has fully loaded, start WireShark packet capture
D. Now enter the password given above in the text box and click on **Validate**. button (Note: Don't press Enter).
You will be then be brought to the CNDM Post-Semester page if you typed the password correctly.
E. Stop WireShark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.

Now let's examine the WireShark output.

Answer the following questions:

16. Inspect the POST message contents, especially HTML Form URL encoded part. Do you see the password submitted in clear in the capture?
17. What should be done to ensure that the password and any other form data is encrypted?

The password 'cndm' that you entered can be found in the HTTP POST message sent to the server. If you scroll down in the contents window at the bottom, you should find the password in CLEAR !!!