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:
1. Start up your web browser.
2. Start up the WireShark packet sniffer, as described in the introductory lab (but don't yet begin packet capture). 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).
3. Enter the following to your browser *http://pages.intnet.mu/rhh/wireshark/file1.html*
Your browser should display the very simple, one-line HTML file.
5. Stop WireShark packet capture.
Your WireShark window should look similar to the window shown in Fig. 1.

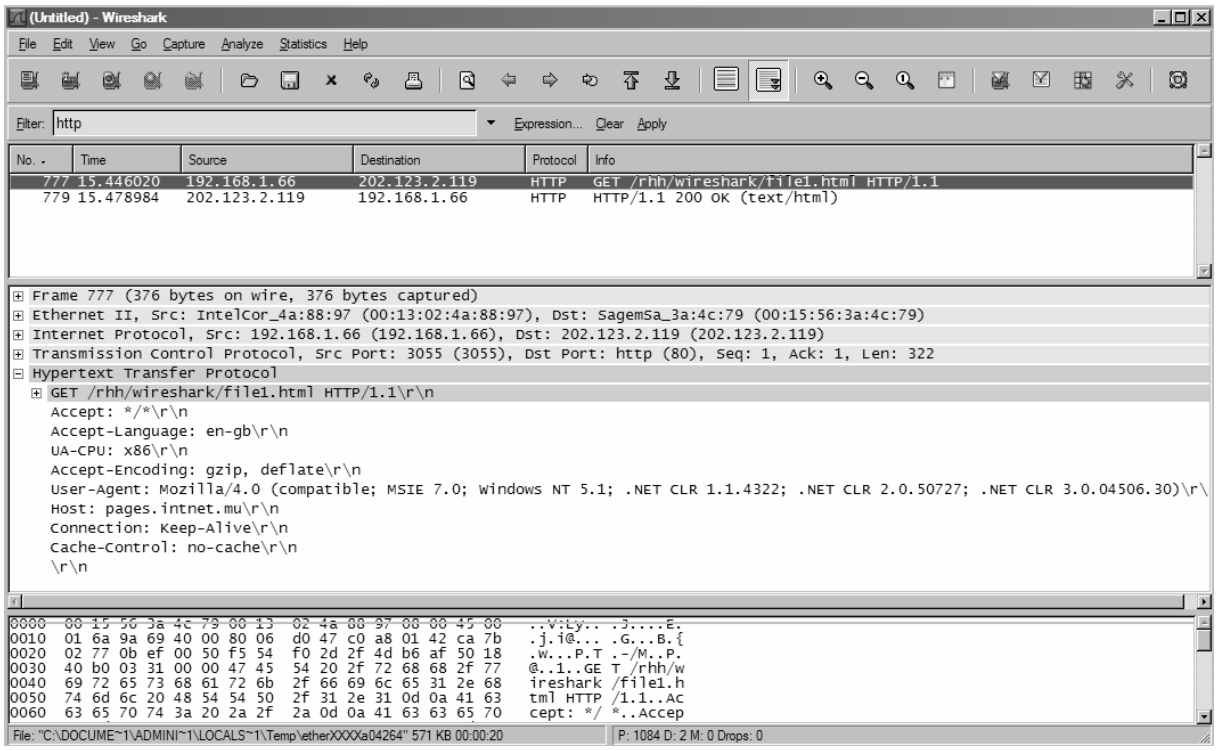

Fig. 1

The example in Fig. 1 shows in the packet-listing window that two HTTP messages were captured: the GET message (from your browser to the *pages.intnet.mu* web server) and the response message from the server to your browser. 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 crosshair (meaning there is hidden, not displayed information), and the HTTP line has a minus (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.

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 *pages.intnet.mu* 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 Internet Explorer, select *Tools->Internet Options->Delete File;* these actions will remove cached files from your browser's cache.) Now do the following:

• Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
• Start up the WireShark packet sniffer
• Enter the following URL into your browser *http://pages.intnet.mu/rhh/wireshark/file2.html*
Your browser should display a very simple one line HTML file.
• Quickly enter the same URL into your browser again (or simply select the Refresh button on your browser or press F5).
• 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 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?

9. Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?

10. 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?

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:
• Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
• Start up the WireShark packet sniffer
• Enter the following URL into your browser *http://pages.intnet.mu/rhh/WireShark/file3.html*
Your browser should display the rather lengthy UTM Act 2002.
• 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 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 45,652 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. 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. HTML Documents with Embedded Objects

Now that we've seen how WireShark displays the captured packet traffic for large HTML files, we can look at what happens when your browser downloads a file with embedded objects, i.e., a file that includes other objects (e.g. image files) that are stored on another server(s).
Do the following:
• Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
• Start up the WireShark packet sniffer.
• Enter the following URL into your browser *http://pages.intnet.mu/rhh/index.htm*
Your browser should display my website with an external link to *www.mauritiustopsites.com*.
The two images themselves are not contained in the HTML; instead the URLs for the images are contained in the *style.css* file. Your browser will get these two gifs from the indicated web sites.
• Stop WireShark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed.
(*Note:* You should ignore any HTTP GET and response for myicon.ico. If you see a reference to this file, it is your browser automatically asking the server if it has a small icon file (Nefertum) that is displayed next to the URL in the address bar).

Answer the following question:

16. How many HTTP GET request messages were sent by your browser? To which Internet addresses were these GET requests sent?

5 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 URL *http://172.16.16.66/wireshark/file4.aspx* is password protected. The username is "CNDM" (without the quotes), and the password is "wiresh@rk" (again, without the quotes). So let's access this "secure" password-protected page.

Do the following:
• Make sure your browser's cache is cleared, as discussed above, and close down your browser. Then, start up your browser
• Start up the WireShark packet sniffer
• Enter the following URL into your browser *http://172.16.16.66/WireShark/file4.aspx*
Type the requested user name and password into the form. You will be brought to the *file4.aspx* page otherwise an error message will appear if you type the username/password incorrectly.
• 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:

18. What is the server's response (status code and phrase) in response to the initial HTTP GET message from your browser?

19. Why and where does your browser send the second HTTP GET message?

20. Why are 3 HTTP GET messages needed while requesting only a single page with no external referencing objects?

The username (CNDM) and password (wiresh@rk) that you entered can be found in the HTTP POST message sent to my server. If you scroll down in the contents window at the bottom, you will find the username and password in CLEAR !!!