Wireshark Lab: HTTP

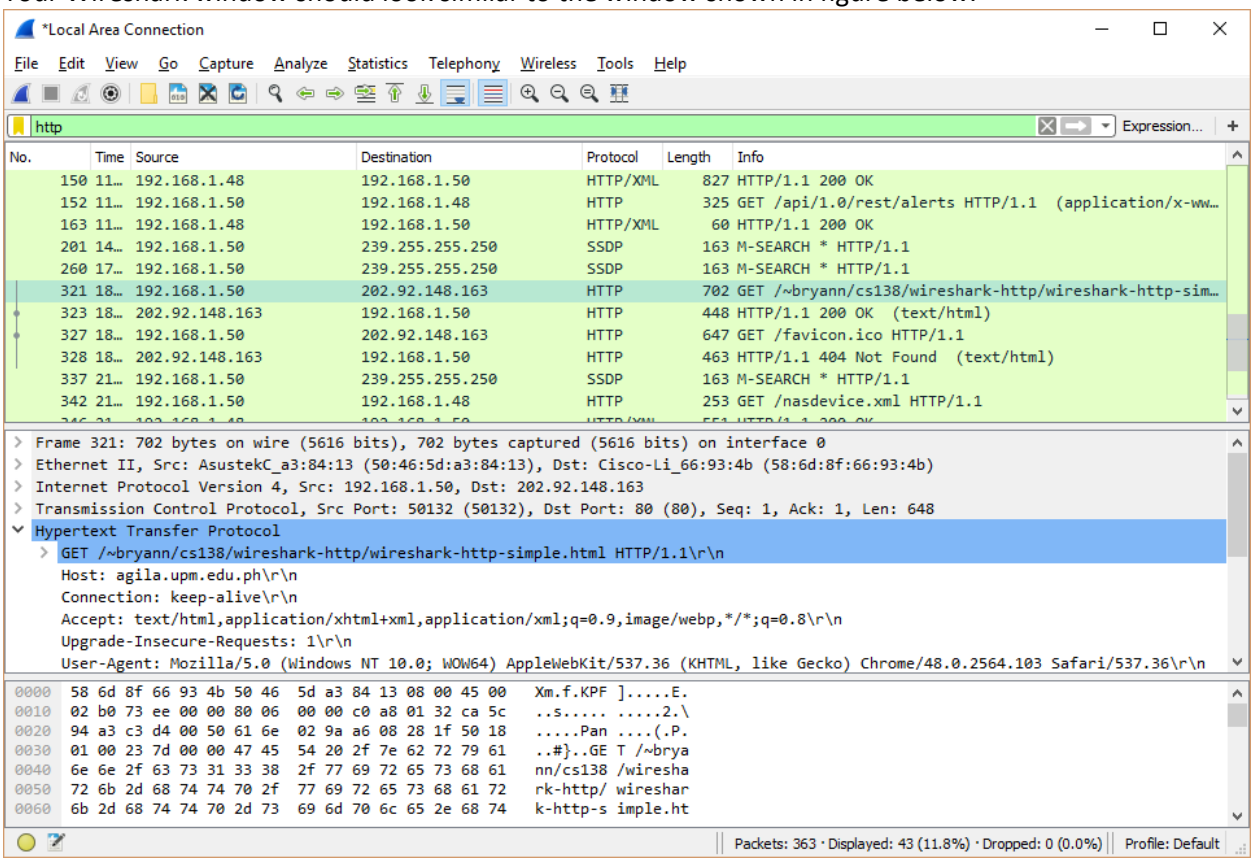
We are going to use Wireshark, a packet capturing and inspection software, to look at the HTTP protocol in this lab activity.

Basic HTTP GET/response Interaction

Let us see how HTTP works by downloading a very simple HTML file (i.e. one that is very short, and contain no embedded objects. Do the following:

1. Start Wireshark
2. Start up your web browser and clear its cache.
3. Start the packet capture in Wireshark.
4. Enter http://agila.upm.edu.ph/~bryann/cs138/wireshark-http/wireshark-http-simple.html. Your browser should display a simple HTML file.
5. After waiting for a short time, stop Wireshark packet capture.

Your Wireshark window should look similar to the window shown in figure below.



The example in Figure 1 shows in the packet-listing window the two HTTP messages (nos. 321 and 323) that are related to getting the HTML file from http://agila.upm.edu.ph/~bryann/cs138/wireshark-http/wireshark-http-simple.html: the GET message from your browser to agila.upm.edu.ph server and the response message from the server to your browser. The packet contents window shows the details of the selected message (in this case the HTTP GET message which is highlighted in the packet-listing window).

By looking at the information in the HTTP GET and response messages, answer the following questions. **When answering the following questions, you should show the GET and response messages (via screenshots) and indicate where in the message you have found the information that answers the following questions.**

1. Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?
2. What is the IP address of your computer? Of the agila.upm.edu.ph?
3. What is the status code returned from the server to your browser?
4. When was the HTML file that you are retrieving (in MNL time) last modified at the server?
5. How many bytes of content are being returned to your browser?

The HTTP conditional GET/response interaction

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. Now do the following:

- Start up your web browser, and make sure your browser's cache is cleared.
- Start up Wireshark packet sniffer.
- Enter the following URL into your browser http://agila.upm.edu.ph/~bryann/cs138/wireshark-http/wireshark-http-simple.html (this is the same as the previous URL).
- Quickly enter the same URL into your browser again (or simply select the refresh button on your browser).
- 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:

6. 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?
7. Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?
8. 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?
9. 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.

Retrieving Long Documents

Up to this point, all the documents we have retrieved are simple and short. Let's 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.
- Start up the Wireshark packet sniffer.
- Enter the following URL into your browser http://agila.upm.edu.ph/~bryann/cs138/wireshark-http/aup.html. Your browser should display the rather lengthy AUP policy of UP.
- Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed.

The HTML file that is returned is too large to fit into one packet, so this is normally fragmented by the protocol layers below the application layer, but Wireshark automatically assembles the fragmented packets and still displays them as one. In spite of this feature, it is still possible that a particular object was fragmented into multiple parts as it was transmitted over the network.

Answer the following questions:

10. How many HTTP GET request messages were sent by your browser?
11. How many data-containing TCP segments were needed to carry the single HTTP response?
12. How did you know that a particular response was fragmented into multiple parts?
13. What is the status code and phrase associated with the response to the HTTP GET request?

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 like image files)

Do the following:

- Start up your web browser, and make sure your browser's cache is cleared.
- Start up the Wireshark packet sniffer.
- Enter the following URL into your browser http://agila.upm.edu.ph/~bryann/cs138/wireshark-http/forbidden/. Your browser should display a page that contains images. The images are referenced in the base HTML file. That is, the images themselves are not contained in the HTML file; instead the URLs of the images are contained in the downloaded HTML file.
- Stop the Wireshark packet capture, and enter "http" in the display-filter-specification window.

Answer the following questions:

14. How many HTTP GET request messages were sent by your browser to retrieve all the objects that were displayed by the browser?
15. Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two web sites in parallel? Explain.

HTTP Security Problem

Let us look at the security problem of HTTP by seeing the danger of transmitting confidential information (e.g. username and password) via HTTP instead of via HTTPS.

- Go to http://agila.upm.edu.ph/~bryann/cs138/login/
- Register for an account.
- Start your Wireshark.
- Go to http://agila.upm.edu.ph/~bryann/cs138/login/ again and use the account that you have used in registration to login.
- Stop Wireshark.

Answer the following questions:

16. You should be able to capture the packet that contain the username and password that you have just entered in the login. Show the packet that contains the username and password.