Lab Assignment 2   ECE325 2020 Spring

Note:

(1) Please show as much of your work as you can. Even if you get a wrong answer, you can get partial credit if you show your work.

(2) Please submit your assignment as a PDF file.

(3) When you hand in your assignment, please attach your packet printout or screenshot and annotate the output so that it's clear where in the output you're getting the information for your answer.

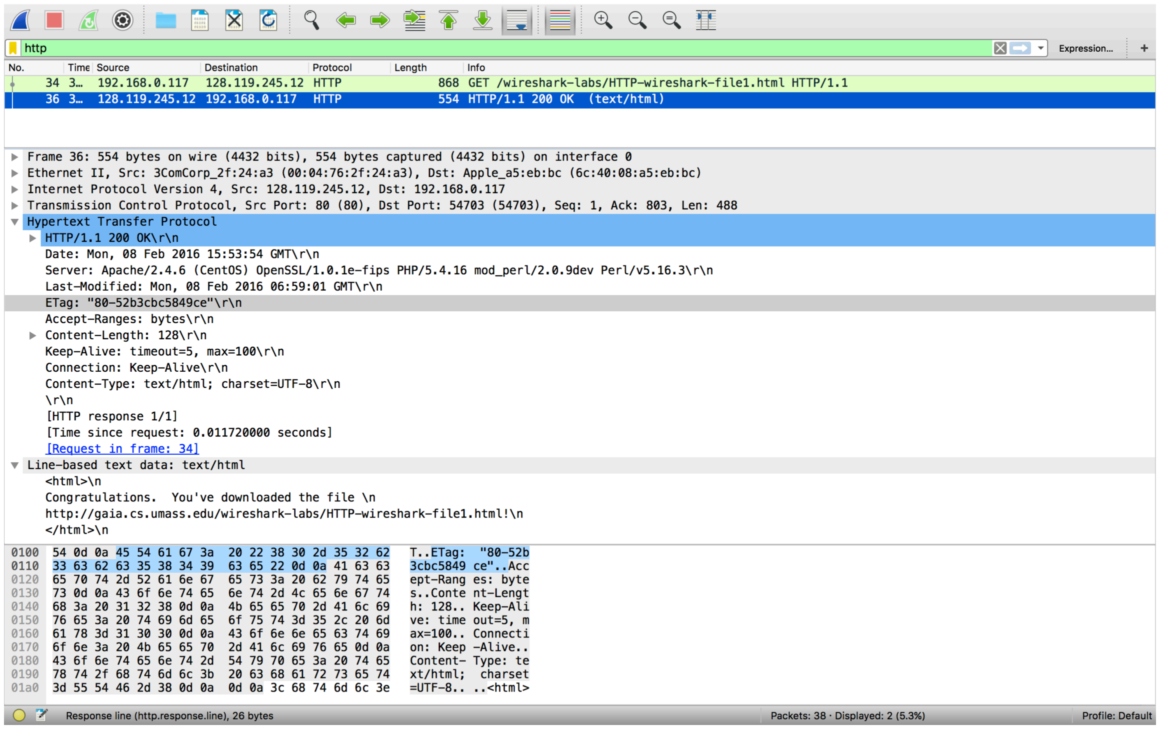
Having gotten our feet wet with the Wireshark packet sniffer in lab 1, we're now ready to use Wireshark to investigate protocols in operation. In this lab, we'll explore several aspects of the HTTP protocol: the basic GET/response interaction, HTTP message formats, retrieving large HTML files, and retrieving HTML files with embedded objects. Before beginning these labs, you might want to review the slides of Chapter 2 on HTML or Section 2.2 of the book.

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 first assignment (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. Wait a bit more than one minute (we'll see why shortly), and then begin Wireshark packet capture.
4. Enter the following to your browser http://gaia.cs.umass.edu/wireshark-labs/HTTP-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 Figure 1. If you capture more than two packets, please don't worry about that. If the status code returned from the server to your browser is "304" which means "Not Modified", please clear the cached files in your browser. To do this under Firefox, select Tools->Clear Recent History and check the Cache box, or for Internet Explorer, select Tools->Internet Options->Delete File; these actions will remove cached files from your browser's cache.



```
No.  Time Source         Destination     Protocol  Length  Info
 34  3…   192.168.0.117  128.119.245.12  HTTP         868  GET /wireshark-labs/HTTP-wireshark-file1.html HTTP/1.1
 36  3…   128.119.245.12 192.168.0.117   HTTP         554  HTTP/1.1 200 OK  (text/html)

▶ Frame 36: 554 bytes on wire (4432 bits), 554 bytes captured (4432 bits) on interface 0
▶ Ethernet II, Src: 3ComCorp_2f:24:a3 (00:04:76:2f:24:a3), Dst: Apple_a5:eb:bc (6c:40:08:a5:eb:bc)
▶ Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.0.117
▶ Transmission Control Protocol, Src Port: 80 (80), Dst Port: 54703 (54703), Seq: 1, Ack: 803, Len: 488
▼ Hypertext Transfer Protocol
  ▶ HTTP/1.1 200 OK\r\n
    Date: Mon, 08 Feb 2016 15:53:54 GMT\r\n
    Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips PHP/5.4.16 mod_perl/2.0.9dev Perl/v5.16.3\r\n
    Last-Modified: Mon, 08 Feb 2016 06:59:01 GMT\r\n
    ETag: "80-52b3cbc5849ce"\r\n
    Accept-Ranges: bytes\r\n
  ▶ Content-Length: 128\r\n
    Keep-Alive: timeout=5, max=100\r\n
    Connection: Keep-Alive\r\n
    Content-Type: text/html; charset=UTF-8\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.011720000 seconds]
    [Request in frame: 34]
▼ Line-based text data: text/html
    <html>\n
    Congratulations.  You've downloaded the file \n
    http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html!\n
    </html>\n

0100  54 0d 0a 45 54 61 67 3a  20 22 38 30 2d 35 32 62   T..ETag:  "80-52b
0110  33 63 62 63 35 38 34 39  63 65 22 0d 0a 41 63 63   3cbc5849 ce"..Acc
0120  65 70 74 2d 52 61 6e 67  65 73 3a 20 62 79 74 65   ept-Rang es: byte
0130  73 0d 0a 43 6f 6e 74 65  6e 74 2d 4c 65 6e 67 74   s..Conte nt-Lengt
0140  68 3a 20 31 32 38 0d 0a  4b 65 65 70 2d 41 6c 69   h: 128.. Keep-Ali
0150  76 65 3a 20 74 69 6d 65  6f 75 74 3d 35 2c 20 6d   ve: time out=5, m
0160  61 78 3d 31 30 30 0d 0a  43 6f 6e 6e 65 63 74 69   ax=100.. Connecti
0170  6f 6e 3a 20 4b 65 65 70  2d 41 6c 69 76 65 0d 0a   on: Keep -Alive..
0180  43 6f 6e 74 65 6e 74 2d  54 79 70 65 3a 20 74 65   Content- Type: te
0190  78 74 2f 68 74 6d 6c 3b  20 63 68 61 72 73 65 74   xt/html;  charset
01a0  3d 55 54 46 2d 38 0d 0a  0d 0a 3c 68 74 6d 6c 3e   =UTF-8.. ..<html>

Response line (http.response.line), 26 bytes          Packets: 38 · Displayed: 2 (5.3%)          Profile: Default
```

Figure 1 Wireshark Display after http://gaia.cs.umass.edu/wireshark-labs/ HTTPwireshark-file1.html has been retrieved by your browser

The example in Figure 1 shows in the packet-listing window that two HTTP messages were captured: the GET message (from your browser to the gaia.cs.umass.edu 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 OK 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're interested in HTTP here, and will be investigating these other protocols in later labs), so make sure the boxes at the far left of the Frame, Ethernet, IP and TCP information have a plus sign or a right-pointing triangle (which means there is hidden, undisplayed information), and the HTTP line has a minus sign or a down-pointing triangle (which means that all information about the HTTP message is displayed).

(*Note:* You should ignore any HTTP GET and response for favicon.ico. If you see a reference to this file, it is your browser automatically asking the server if it (the server) has a small icon file that should be displayed next to the displayed URL in your browser. We'll ignore references to this pesky file in this lab.)

By looking at the information in the HTTP GET and response messages, answer the following questions. When answering the following questions, you should print out the GET and response messages (to do so, select Print from the Wireshark File command menu, and select the "Selected Packet Only" and "Print as displayed" radial buttons, and then click OK) and indicate where in the message you've found the information that answers the following questions.

Answer 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 languages (if any) does your browser indicate that it can accept to the server?
3. What is the IP address of your computer? Of the gaia.cs.umass.edu server?
4. What is the status code returned from the server to your browser?
5. When was the HTML file that you are retrieving last modified at the server?
6. How many bytes of content are being returned to your browser?

In your answer to question 5 above, you might have been surprised to find that the document you just retrieved was last modified within a minute before you downloaded the document. That's because (for this particular file), the gaia.cs.umass.edu server is setting the file's last-modified time to be the current time, and is doing so once per minute. Thus, if you wait a minute between accesses, the file will appear to have been recently modified, and hence your browser will download a "new" copy of the document.

2. The HTTP CONDITIONAL GET/response interaction

Recall from Section 2.2.6 of the text, that most web browsers perform object caching and thus perform a conditional GET when retrieving an HTTP object. Before performing the steps. Now do the following:

1. Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
2. Start up the Wireshark packet sniffer.

3. Enter the following URL into your browser

   http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html

   Your browser should display a very simple five-line HTML file.
4. Quickly enter the same URL into your browser again (or simply select the refresh button on your browser)
5. 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:

7. 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?
8. Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?
9. 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?
10. 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:

1. Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
2. Start up the Wireshark packet sniffer.
3. Enter the following URL into your browser

   http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html . Your browser should display the rather lengthy US Bill of Rights.
4. 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 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. 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 "[X Reassembled TCP Segments ...]" header displayed for the HTTP response message in Wireshark. We stress here that that Wireshark shows the reassemble HTTP response which in reality consists of more than one TCP segment.

Answer the following questions:

11. How many HTTP GET request messages were sent by your browser?
12. How many data-containing TCP segments were needed to carry the single HTTP response?
13. What is the status code and phrase associated with the response to the HTTP GET request?

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 (in the example below, image files) that are stored on another server(s).

Do the following:

1. Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
2. Start up the Wireshark packet sniffer.
3. Enter the following URL into your browser http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html . Your browser should display a short HTML file with two images. These two images are referenced in the base HTML file. That is, the images themselves are not contained in the HTML; instead the URLs for the images are contained in the downloaded HTML file. As discussed in the textbook, your browser will have to retrieve these logos from the indicated web sites. Our publisher's logo is retrieved from the www.aw-bc.com web site. The image of the cover for our 5th edition textbook is stored at the manic.cs.umass.edu server.
4. Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed.

Answer the following questions:

14. How many HTTP GET request messages did your browser send? To which Internet addresses were these GET requests sent?
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.

5. HTTP Authentication

Finally, let's try visiting a web site that is password-protected and examine the sequence of HTTP message exchanged for such a site. The URL http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html is password protected. The username is "wireshark-students" (without the quotes), and the password is "network" (again, without the quotes). So let's access this "secure" password-protected site.

Do the following:

1. Make sure your browser's cache is cleared, as discussed above, and close down your browser. Then, start up your browser.
2. Start up the Wireshark packet sniffer.
3. Enter the following URL into your browser.
   http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html
4. Type the requested user name and password into the pop up box.
5. 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. What is the server's response (status code and phrase) in response to the initial HTTP GET message from your browser?
17. When your browser sends the HTTP GET message for the second time, what new field is included in the HTTP GET message?