Concordia University
Engineering and Computer Science

COEN 445
Communication Networks and Protocols

Lab 2

Wireshark Lab: http

Claude Fachkha

Introduction

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. Before beginning these labs, you might want to review Section 2.2 of the textbook.

1.The Basic HTTP GET/response interaction

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.

3. Wait a bit more than one minute (we'll see why shortly), and then begin Wireshark packet capture.

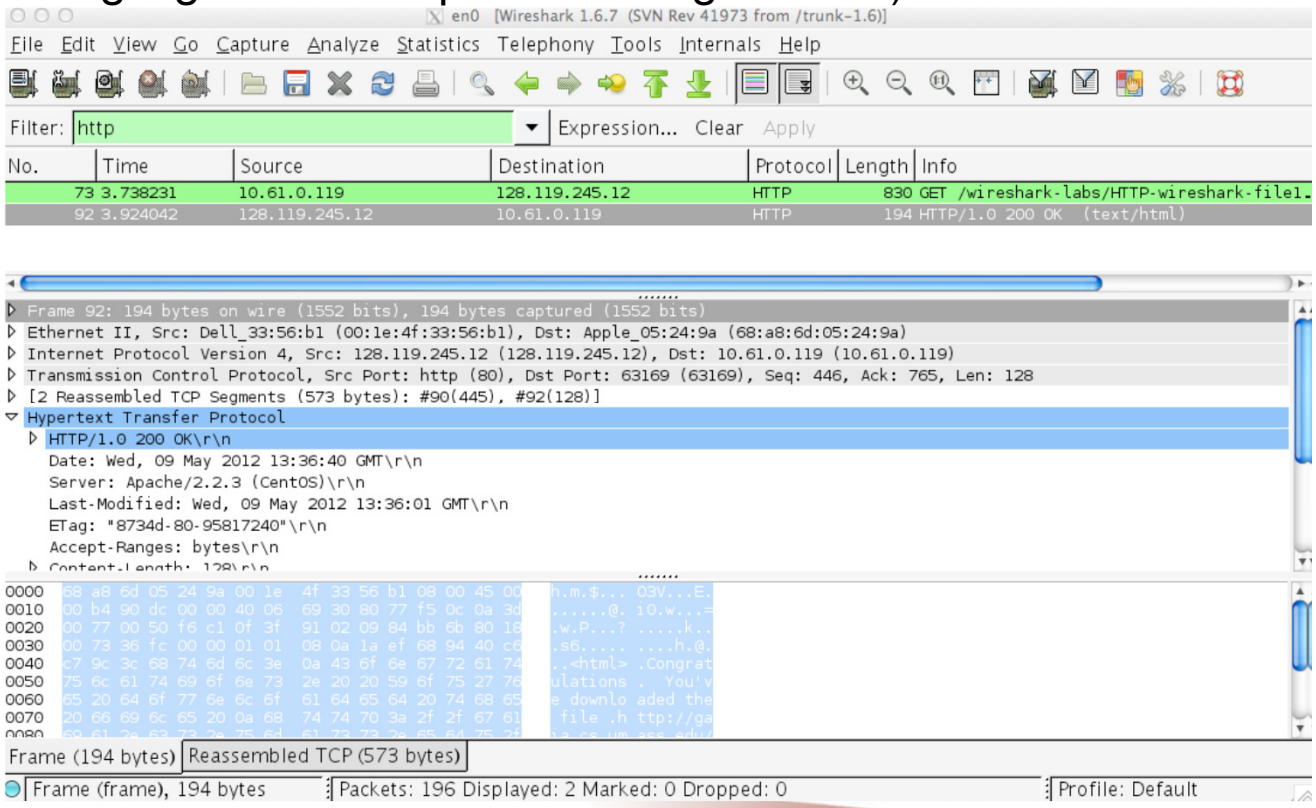
4. Enter the following to your browser
http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html

Your browser should display the very simple, one-line HTML file.

5. Stop Wireshark packet capture.

1.The Basic HTTP GET/response interaction (Cont.)

Figure 1 shows in the packet-listing window that two HTTP messages were captured: the GET message (from your browser to the chosen 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).



Quiz
(Based on the 1st experiment)

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?

7. By inspecting the raw data in the packet content window, do you see any headers within the data that are not displayed in the packet-listing window? If so, name one.

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 below, make sure your browser's cache is empty.

- Start up the Wireshark packet sniffer
- 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.

- 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.

Quiz
(Based on the 2nd experiment)

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 before.
- Start up the Wireshark packet sniffer
- 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.

- Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed.

3. Retrieving Long Documents (Cont.)

In the packet-listing window, you should see your HTTP GET message, followed by a multiple-packet TCP response to your HTTP GET request. This multiple-packet response deserves a bit of explanation. Recall from Section 2.2 (see Figure 2.9 in the text) 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 (see Figure 1.24 in the text).

In recent versions of Wireshark, Wireshark indicates each TCP segment as a separate packet, and the fact that the single HTTP response was fragmented across multiple TCP packets is indicated by the "TCP segment of a reassembled PDU" in the Info column of the Wireshark display.

Quiz
(Based on the 3rd Experiment)

12. How many HTTP GET request messages did your browser send? Which packet number in the trace contains the GET message for the Bill or Rights?

13. Which packet number in the trace contains the status code and phrase associated with the response to the HTTP GET request?

14. What is the status code and phrase in the response?

15. How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights?

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).

4. HTML Documents with Embedded Objects (Cont.)

• 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://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 (one of our favorite covers) is stored at the manic.cs.umass.edu server.

• Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed.

Quiz
(Based on the 4th experiment)

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

17. 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.

5. HTTP Authentication (Cont.)

• 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://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wiresharkfile5.html
Type the requested user name and password into the pop up box.

• 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.

Quiz
(Based on the 5th experiment)

Now let's examine the Wireshark output. You might want to first read up on HTTP authentication by reviewing the easy-to-read material on "HTTP Access Authentication Framework" at http://frontier.userland.com/stories/storyReader$2159

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

19. When your browser's sends the HTTP GET message for the second time, what new field is included in the HTTP GET message?

5. HTTP Authentication (Cont.)

The username (wireshark-students) and password (network) that you entered are encoded in the string of characters (d2lyZXNoYXJrLXN0dWRlbnRzOm5ldHdvcms=) following the "Authorization: Basic" header in the client's HTTP GET message. While it may appear that your username and password are encrypted, they are simply encoded in a format known as Base64 format. The username and password are not encrypted!

To see this, go to http://www.motobit.com/util/base64-decoder-encoder.asp and enter the base64-encoded string d2lyZXNoYXJrLXN0dWRlbnRz and decode. Voila! You have translated from Base64 encoding to ASCII encoding, and thus should see your username! To view the password, enter the remainder of the string Om5ldHdvcms= and press decode.

Since anyone can download a tool like Wireshark and sniff packets (not just their own) passing by their network adaptor, and anyone can translate from Base64 to ASCII (you just did it!), it should be clear to you that simple passwords on WWW sites are not secure unless additional measures are taken.

References

Online services - Computer Networking: A Top-Down Approach, 6/E
James F. Kurose, *University of Massachusetts, Amherst* - See more at: Keith W. Ross, *Polytechnic University, Brooklyn*

Claude Fachkha

c_fachkh@encs.concordia.ca