Lab 1: Wireshark Lab
Deadline: Friday 20/9 17:00

In this lab, you will 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.

- The lab can be done by at most two students.
- Email your solution to: elmira.zohrevandi@liu.se

0. Preparation for the Lab

- Read the article *Packet Capture & Traffic Analysis with Wireshark* available at http://dellacqua.se/education/courses/tnm031/labs_material/intro%20wireshark.pdf
- For this lab you need to download Wireshark sniffer at www.wireshark.org

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:

- Start up your web browser.
- Start up the Wireshark sniffer.
- Enter http 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).
- Start Wireshark packet capture.
- Enter the following to your browser
  http://gaia.cs.umass.edu/ethereal-labs/HTTP-ethereal-file1.html
  Your browser should display a very simple, one-line HTML file.
- Stop Wireshark packet capture.

Your Wirewhark window should look similar to the window shown in Figure 1.

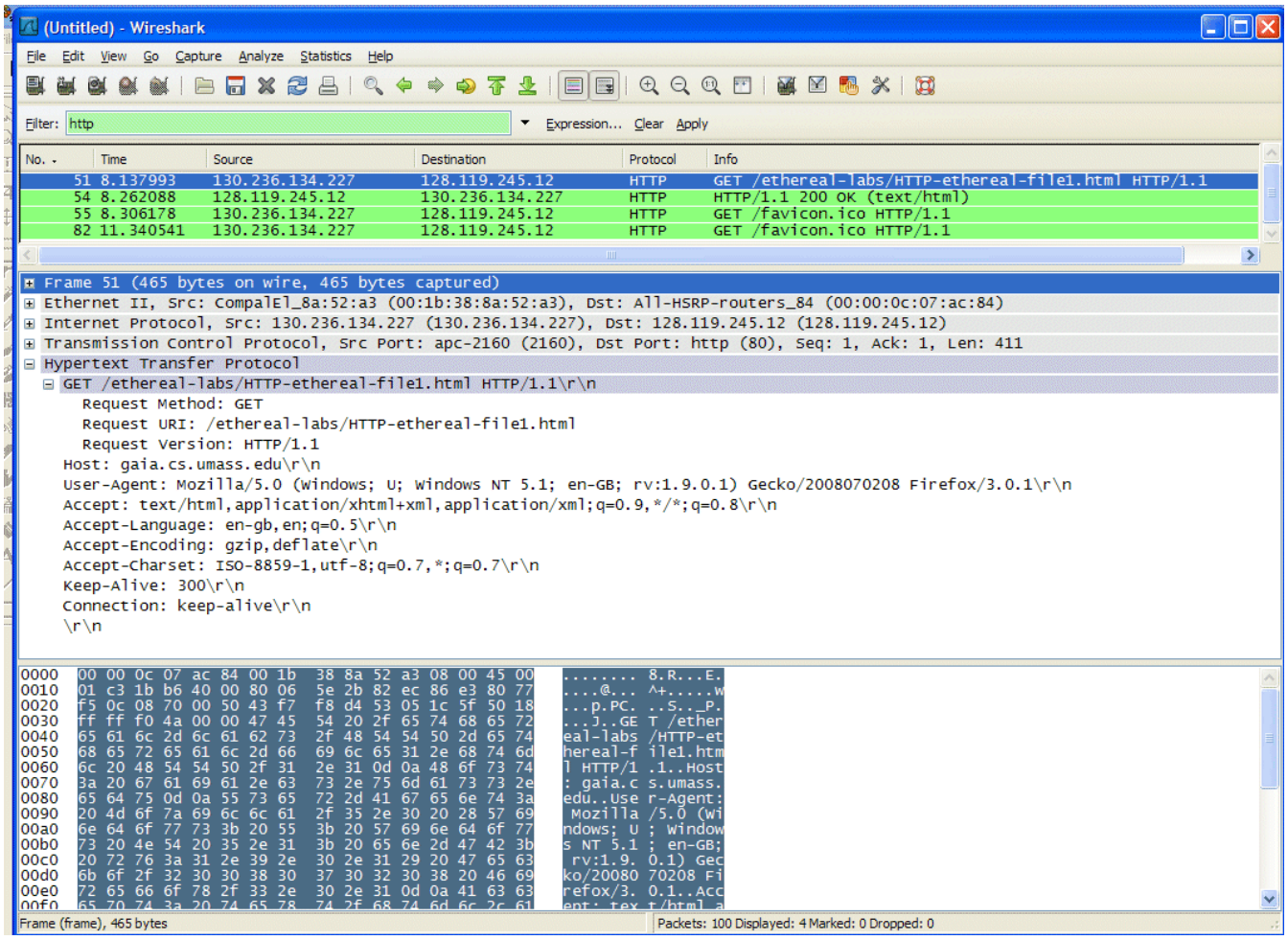

Figure 1: Wireshark display after the file HTTP-ethereal-file1.html has been retrieved by your browser

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 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 packet, 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), so make sure the boxes at the far left of the Frame, Ethernet, IP and TCP information have a right-pointing arrowhead (which means there is hidden, undisplayed information), and the HTTP line has a down-pointing arrowhead (which means that all information about the HTTP message is displayed).

Note: Ignore any HTTP GET and response for favicon.ico.[1]

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 (see the Wireshark user guide for an explanation of how to do this) and indicate where in the message you've 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 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 recently modified 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.

2. 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. (To do this under Chrome, from the Menu button in the upper-right corner of the Chrome window, choose More Tools > Clear browsing data, and clear the memory and disk cache. This action 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://gaia.cs.umass.edu/ethereal-labs/HTTP-ethereal-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.

[1] These are generated automatically by your browser when asks the server if it has an icon file to be displayed next to the displayed URL in your browser.

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 previous examples, 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://gaia.cs.umass.edu/ethereal-labs/HTTP-ethereal-file3.html
  Your browser should display the rather lengthy US Bill of Rights.
- Stop Wireshark packet capture, and enter "tcp" 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. 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.

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:

- 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://dellacqua.se/files/HTTP-ethereal-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. Your browser will have to retrieve these images 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.

Answer the following questions:

14. How many HTTP GET request messages were sent by your browser? 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
www.dellacqua.se/education/courses/tnm031
is password protected. The username is **pizza**, and the password is **margherita**.
To access this secure password-protected site, 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://dellacqua.se/education/courses/tnm031
  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.

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's sends the HTTP GET message for the second time, what new field is included in the HTTP GET message?

The username and password that you entered are encoded in the string of characters (cGl6emE6bWFyZ2hlcml0YQ==) 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 Base64 format. The username and password are *not* encrypted! To see this, open up Authorization: Basic and look at the Credentials.

You have translated from Base64 encoding to ASCII encoding, and thus should see both your username and password in clear text. 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[2], it should be clear to you that simple passwords on websites are not secure unless additional measures are taken.

Later in the course we will see how to make the website access more secure. However, we'll clearly need something that goes beyond the basic HTTP authentication framework!

6. Capturing VoIP (Optional)

Finally, we will see how one can capture media streaming with Wireshark. This part is optional; but doing it will take you only few minutes.

Wireshark can capture the entire inbound and outbound traffic of your machine including audio and video streaming. We will see here how to capture audio streaming with two popular protocols; SIP and RTP.

Look first at the two short videos:

- SIP vs. RTP [www.youtube.com/watch?v=7-C8oPTMQSQ]
- VOIP Wireshark Capture [www.youtube.com/watch?v=iFQKB0MQ-dY]

Now let's examine a previously captured pcap file. Download and open in Wireshark: http://dellacqua.se/education/courses/tnm031/labs_material/audio%20capture.pcap

18. Can you discover what the captured audio says?

[2] An online Base64 converter is available at www.branah.com/ascii-converter