REDES DE COMPUTADORES
2ND LAB – HTTP

BASED ON THE LABS FROM THE BOOK

Having gotten our feet wet with the Wireshark packet sniffer in the previous labs, 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. Before beginning these labs, you might want to review the textbook.

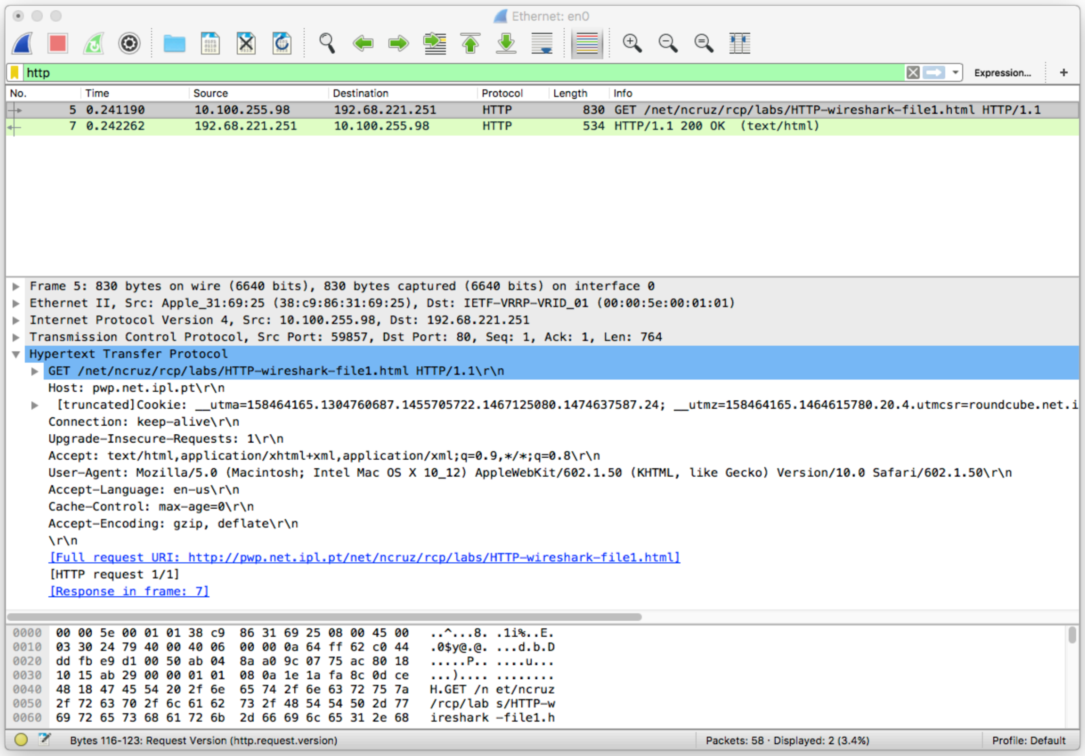
1. THE BASIC HTTP GET/RESPONSE INTERACTION

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. 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://supp.lrcd.local/rcp/labs/HTTP-wireshark-file1.html
5. Your browser should display the very simple, one-line HTML file.
6. Stop Wireshark packet capture.

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



```
No.     Time       Source            Destination       Protocol  Length  Info
      5 0.241190   10.100.255.98     192.68.221.251    HTTP        830  GET /net/ncruz/rcp/labs/HTTP-wireshark-file1.html HTTP/1.1
      7 0.242262   192.68.221.251    10.100.255.98     HTTP        534  HTTP/1.1 200 OK  (text/html)
```

```
▶ Frame 5: 830 bytes on wire (6640 bits), 830 bytes captured (6640 bits) on interface 0
▶ Ethernet II, Src: Apple_31:69:25 (38:c9:86:31:69:25), Dst: IETF-VRRP-VRID_01 (00:00:5e:00:01:01)
▶ Internet Protocol Version 4, Src: 10.100.255.98, Dst: 192.68.221.251
▶ Transmission Control Protocol, Src Port: 59857, Dst Port: 80, Seq: 1, Ack: 1, Len: 764
▼ Hypertext Transfer Protocol
  ▶ GET /net/ncruz/rcp/labs/HTTP-wireshark-file1.html HTTP/1.1\r\n
    Host: pwp.net.ipl.pt\r\n
  ▶  [truncated]Cookie: __utma=158464165.1304760687.1455705722.1467125080.1474637587.24; __utmz=158464165.1464615780.20.4.utmcsr=roundcube.net.i
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12) AppleWebKit/602.1.50 (KHTML, like Gecko) Version/10.0 Safari/602.1.50\r\n
    Accept-Language: en-us\r\n
    Cache-Control: max-age=0\r\n
    Accept-Encoding: gzip, deflate\r\n
    \r\n
    [Full request URI: http://pwp.net.ipl.pt/net/ncruz/rcp/labs/HTTP-wireshark-file1.html]
    [HTTP request 1/1]
    [Response in frame: 7]
```

```
0000  00 00 5e 00 01 01 38 c9  86 31 69 25 08 00 45 00   ..^...8. .1i%..E.
0010  03 30 24 79 40 00 40 06  00 00 0a 64 ff 62 c0 44   .0$y@.@. ...d.b.D
0020  dd fb e9 d1 00 50 ab 04  8a a0 9c 07 75 ac 80 18   .....P.. ....u...
0030  10 15 ab 29 00 00 01 01  08 0a 1e 1a fa 8c 0d ce   ...).... ........
0040  48 18 47 45 54 20 2f 6e  65 74 2f 6e 63 72 75 7a   H.GET /n et/ncruz
0050  2f 72 63 70 2f 6c 61 62  73 2f 48 54 54 50 2d 77   /rcp/lab s/HTTP-w
0060  69 72 65 73 68 61 72 6b  2d 66 69 6c 65 31 2e 68   ireshark -file1.h
```

```
Bytes 116-123: Request Version (http.request.version)        Packets: 58 · Displayed: 2 (3.4%)        Profile: Default
```

Figure 1: Wireshark HTTP example

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 supp.lrcd.local 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 is 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.

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 supp.lrcd.local server?
4. What is the status code returned from the server to your browser? If your status code is “304” you should restart the capture from the beginning and hit “control+f5” on your browser to force a refresh of the web page.
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. Hint: Look for compressed pages.

2. THE HTTP CONDITIONAL GET/RESPONSE INTERACTION

Recall from the text, that most web browsers perform object caching and thus perform a conditional GET when retrieving an HTTP object. 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://supp.lrcd.local/rcp/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 (do not use the refresh button on your browser). If you don’t see any new packets on Wireshark you need to hit the refresh button of your browser. This happens because different browsers have different behaviours.
- 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://supp.lrcd.local/rcp/labs/HTTP-wireshark-file3.html

- Your browser should display the rather lengthy text.

- 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 TCP response to your HTTP GET request. This multiple-packet response deserves a bit of explanation. Recall from 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. 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.

Answer the following questions:

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

13. Which packet 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?

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://supp.lrcd.local/rcp/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. Pearson logo is retrieved from the supp.lrcd.local web site. ISEL logo is stored at the biblio.isel.pt server.
- 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:

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.