Zoltan Batoczki
Robert Florence

Wireshark Report (HTTP)

Objective

The purpose of the HTTP Wireshark lab is to familiarize ourselves with the different properties of the HTTP protocol including the GET interaction, HTTP message formats, handling large HTML files and files that contain embedded objects, and HTTP authentication and security.

LAB

Basic HTTP GET/response interaction:

We first begin by filtering the packet sniffer to only observe the HTTP protocols. We enter in a test URL to download a simple html page and the sniffer captured two HTTP packets: the GET message (from the local browser to the requested server) and the response message from the server to the browser.

```
4369 472.500209    10.31.21.119        128.119.245.12      HTTP      473 GET /wireshark-labs/HTTP-wireshar…
4371 472.587344    128.119.245.12      10.31.21.119        HTTP      540 HTTP/1.1 200 OK  (text/html)
```

Since we captured an HTTP request, the HTTP message is enclosed within a TCP segment which is within an IP datagram, which is enclosed within an Ethernet frame. With text results as follows:

```
▷ Frame 118: 540 bytes on wire (4320 bits), 540 bytes captured (4320 bits) on interface 0
▷ Ethernet II, Src: Alcatel-_f2:8e:01 (e8:e7:32:f2:8e:01), Dst: HonHaiPr_ce:a5:42 (c4:8e:8f:ce:a5:42)
▷ Internet Protocol Version 4, Src: 128.119.245.12, Dst: 10.31.21.116
▷ Transmission Control Protocol, Src Port: 80 (80), Dst Port: 51561 (51561), Seq: 1, Ack: 420, Len: 486
▷ Hypertext Transfer Protocol
◢ 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
```

Questions

1. Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?

a. Our browser is running on HTTP version 1.1

```
Hypertext Transfer Protocol
  GET /wireshark-labs/HTTP-wireshark-file1.html HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET /wireshark-labs/HTTP-wireshark-file1.html HTTP/1.1\r\n]
    Request Method: GET
    Request URI: /wireshark-labs/HTTP-wireshark-file1.html
    Request Version: HTTP/1.1
```

b. The server is also running HTTP version 1.1

```
HTTP/1.1 200 OK\r\n
  [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
  Request Version: HTTP/1.1
  Status Code: 200
  Response Phrase: OK
```

2. What languages (if any) does your browser indicate that it can accept to the server?
   a. It can accept PHP and Perl

```
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips PHP/5.4.16 mod_perl/2.0.10 Perl/v5.16.3\r\n
```

3. What is the IP address of your computer? Of the gaia.cs.umass.edu server?
   a. Our IP: 10.31.21.116. Server IP: 128.119.245.12

```
10.31.21.116        128.119.245.12
128.119.245.12      10.31.21.116
```

4. What is the status code returned from the server to your browser?
   a. The server returned a status code of 200:

```
HTTP/1.1 200 OK\r\n
  [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
  Request Version: HTTP/1.1
  Status Code: 200
  Response Phrase: OK
```

5. When was the HTML file that you are retrieving last modified at the server?
   a.

```
Last-Modified: Wed, 08 Mar 2017 06:59:01 GMT\r\n
```

6. How many bytes of content are being returned to your browser?
   a. The content length of the HTTP response is 128 bytes

```
Accept-Ranges: bytes\r\n
Content-Length: 128\r\n
```

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.
   a. Based on what we saw, there were no headers that were not displayed within any of the network layers.

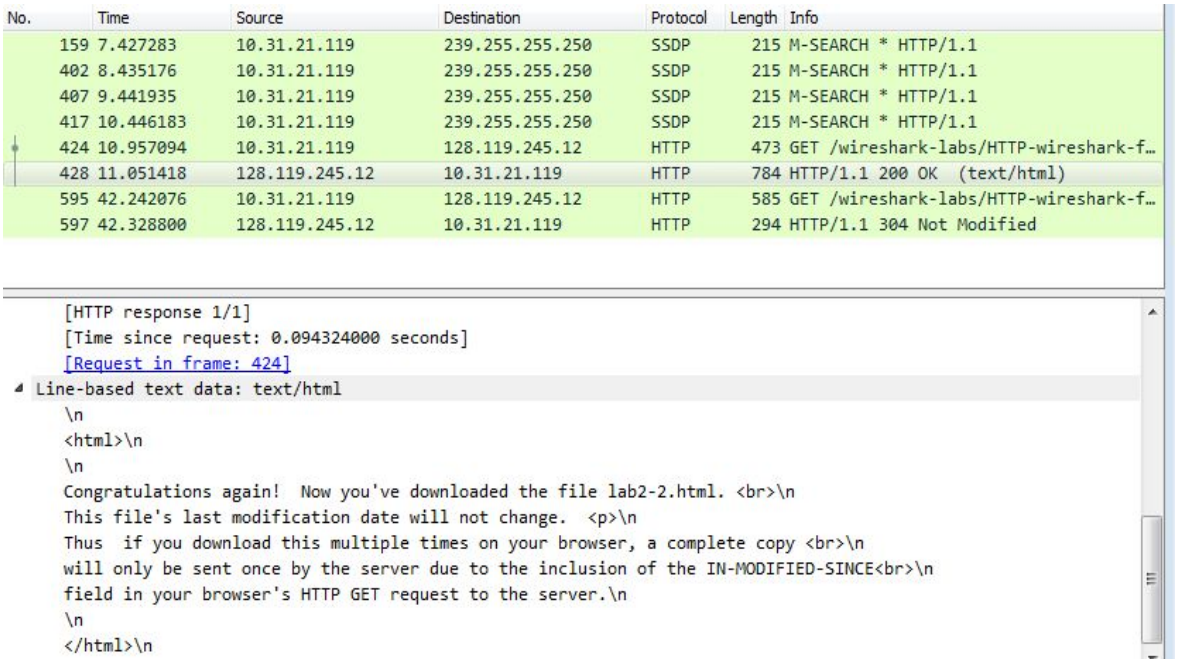
The HTTP CONDITIONAL Get/Response Interaction

Prior to continuing, we were asked to clear the browser's cache. Followed by accessing the webpage once. After, we were requested to refresh the webpage. While running the packet

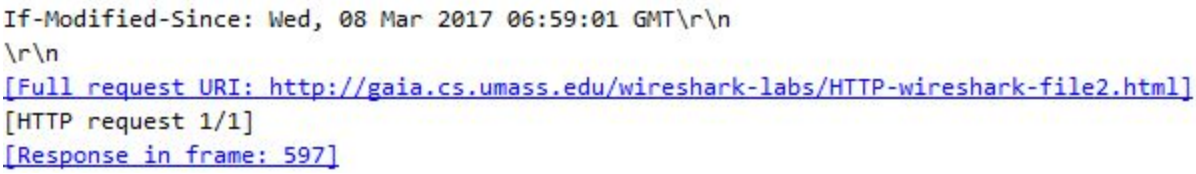
sniffer we saw that, our browser downloaded the webpage the first time, and cached it for future use. The next time it was to be downloaded from the server, it checks to see if it had been modified since last retrieval and just sends a status message that it was unchanged. This concept of only downloading pages if it has been modified, is critical to the underlying structure of the internet and how its higher speeds can be maintained.

Questions

1. 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?
    a. There is no “IF-MODIFIED-SINCE” line in the HTTP GET packet
2. Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?
    a. Yes, the server did return content. Wireshark captured an OK HTTP packet which contained the content of the html page

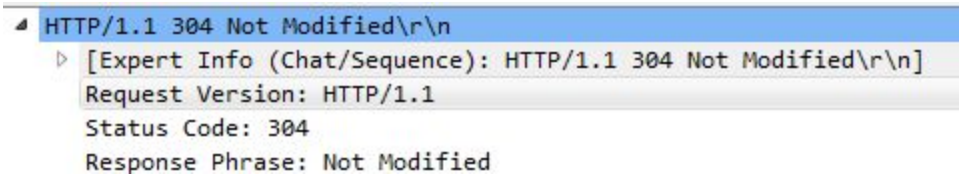


3. Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE:” line in the HTTP GET? If so, what information follows the “IF-MODIFIED-SINCE:” header?
    a. Yes, the second GET does contain a “IF-MODIFIED-SINCE:” header.



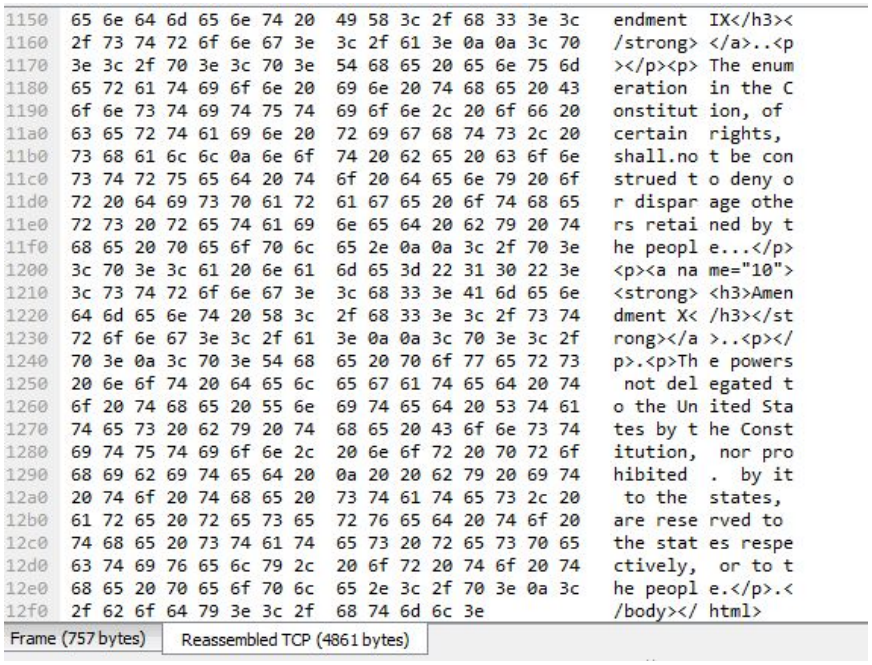
4. What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain.

a. The server sent back to the browser a status code of 304 with a response phrase of “Not Modified”. Hence, the server did not return the content of the page since it was already downloaded from the first GET request.



Retrieving Long Documents

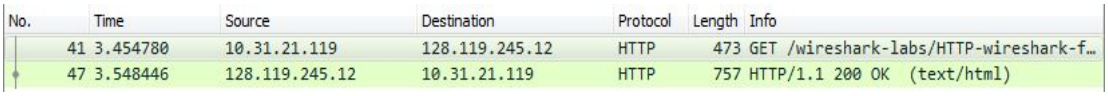
The purpose of this section will allow us to see how requests for longer HTML pages work. We have cleared the browser’s cache to avoid any missed requests. We request a web page that contains text from the U.S. BIll of Rights. One important notice is that the requested HTML file was too large to store in one TCP packet. Because of that, the HTTP response was broken down into several pieces by TCP, with each piece being contained in a TCP segment, represented by the “Reassembled TCP” tab.



Questions

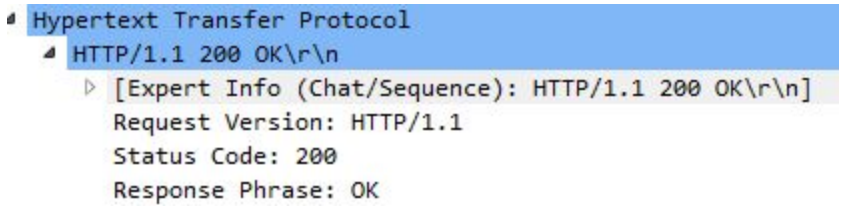
1. How many HTTP GET request messages did your browser send? Which packet number in the trace contains the GET message for the Bill or Rights?

a. Our browser sent one GET request for the web page. The GET message was contained in packet 41 in the trace.



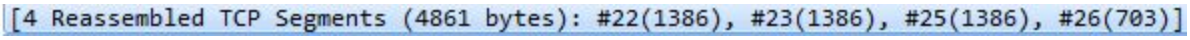
No.	Time	Source	Destination	Protocol	Length	Info
41	3.454780	10.31.21.119	128.119.245.12	HTTP	473	GET /wireshark-labs/HTTP-wireshark-f…
47	3.548446	128.119.245.12	10.31.21.119	HTTP	757	HTTP/1.1 200 OK (text/html)

2. Which packet number in the trace contains the status code and phrase associated with the response to the HTTP GET request?
    a. The Response with packet number (47) contained the status code: 200
3. What is the status code and phrase in the response?
    a. The response contains the status code of 200 with the response phrase “OK”



```
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
    Request Version: HTTP/1.1
    Status Code: 200
    Response Phrase: OK
```

4. How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights?
    a. A total of four extra TCP segments were needed to carry the HTTP request



```
[4 Reassembled TCP Segments (4861 bytes): #22(1386), #23(1386), #25(1386), #26(703)]
```

    b. In wireshark, the 4 TCP segments that were broken down were reassembled under one single response packet.

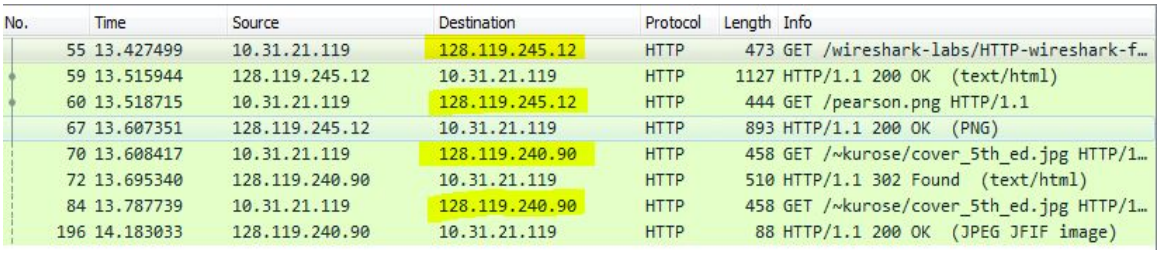
HTML Documents with Embedded Objects

Now that we know how the HTTP protocol handles larger HTML files, we will see how it interacts with embedded objects such as images that are from a different server. We go to http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html which contains two images that are *not* within the HTML but are downloaded with the file from separate servers. This implementation shows the layering capability of TCP requests, and how responses can lead to GET requests to other servers.

Questions

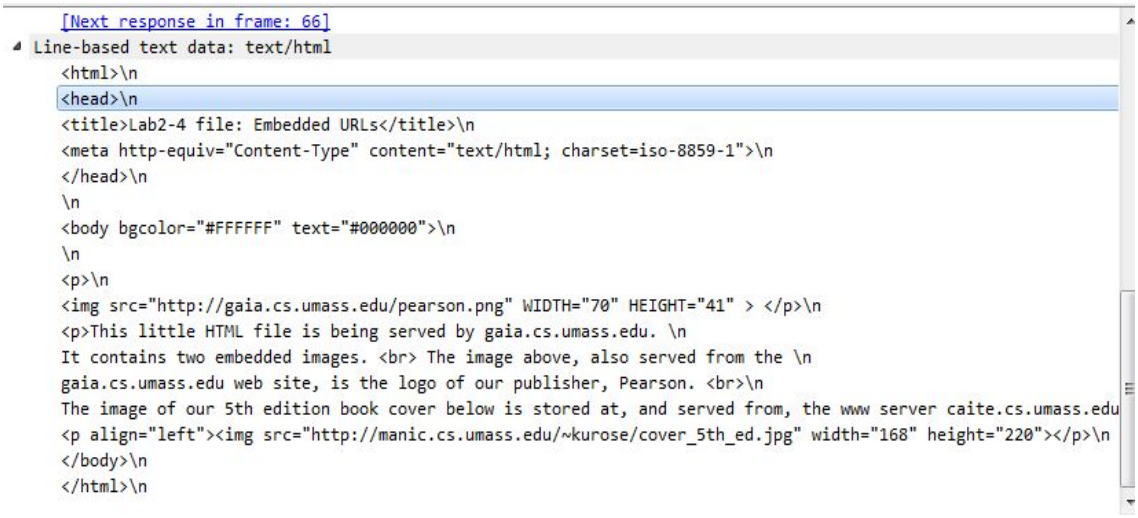
1. How many HTTP GET request messages did your browser send? To which Internet addresses were these GET requests sent?

a. The browser sent four GET requests; The first two went to address 128.119.245.12. The other two requests went to address 128.119.240.90



No.	Time	Source	Destination	Protocol	Length	Info
55	13.427499	10.31.21.119	128.119.245.12	HTTP	473	GET /wireshark-labs/HTTP-wireshark-f…
59	13.515944	128.119.245.12	10.31.21.119	HTTP	1127	HTTP/1.1 200 OK (text/html)
60	13.518715	10.31.21.119	128.119.245.12	HTTP	444	GET /pearson.png HTTP/1.1
67	13.607351	128.119.245.12	10.31.21.119	HTTP	893	HTTP/1.1 200 OK (PNG)
70	13.608417	10.31.21.119	128.119.240.90	HTTP	458	GET /~kurose/cover_5th_ed.jpg HTTP/1…
72	13.695340	128.119.240.90	10.31.21.119	HTTP	510	HTTP/1.1 302 Found (text/html)
84	13.787739	10.31.21.119	128.119.240.90	HTTP	458	GET /~kurose/cover_5th_ed.jpg HTTP/1…
196	14.183033	128.119.240.90	10.31.21.119	HTTP	88	HTTP/1.1 200 OK (JPEG JFIF image)

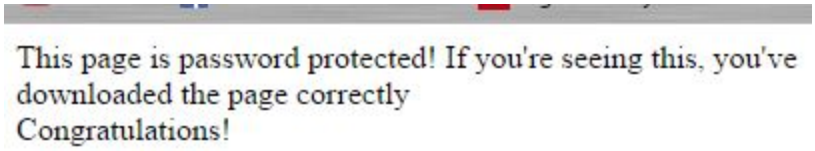
2. Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two web sites in parallel? Explain.
    a. From inspection it seems as if the first GET request returns the HTML for the page and GET requests to two separate images. The first image (pearson logo) is on the same server, but since it is not actually embedded on the html page, a second request had to be made to serially retrieve the image. The second image is on a separate server in which it makes a parallel request (two different ports on same server) to retrieve the book cover image. Within the first response from the server it includes <img src> tags in which the browser has to create other GET requests to those servers. Those GET requests create a response from their respective image data to plug into the origin GET/response with the HTML to create webpage. An example shown below:



```
[Next response in frame: 66]
Line-based text data: text/html
<html>\n
<head>\n
<title>Lab2-4 file: Embedded URLs</title>\n
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">\n
</head>\n
\n
<body bgcolor="#FFFFFF" text="#000000">\n
\n
<p>\n
<img src="http://gaia.cs.umass.edu/pearson.png" WIDTH="70" HEIGHT="41" > </p>\n
<p>This little HTML file is being served by gaia.cs.umass.edu. \n
It contains two embedded images. <br> The image above, also served from the \n
gaia.cs.umass.edu web site, is the logo of our publisher, Pearson. <br>\n
The image of our 5th edition book cover below is stored at, and served from, the www server caite.cs.umass.edu
<p align="left"><img src="http://manic.cs.umass.edu/~kurose/cover_5th_ed.jpg" width="168" height="220"></p>\n
</body>\n
</html>\n
```

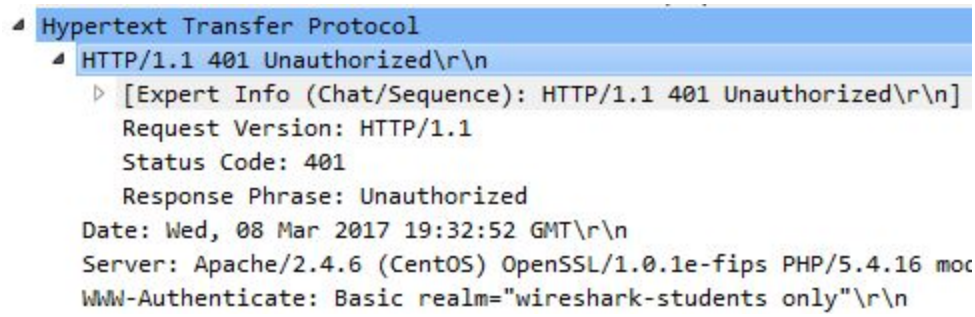
HTTP Authentication

In this section, we observe how a password protected page is sent to a client. We are provided a username and password in order to login.
The first GET request is denied because login credentials are required. Once the credentials are entered, a second GET is sent, and a success page is loaded.

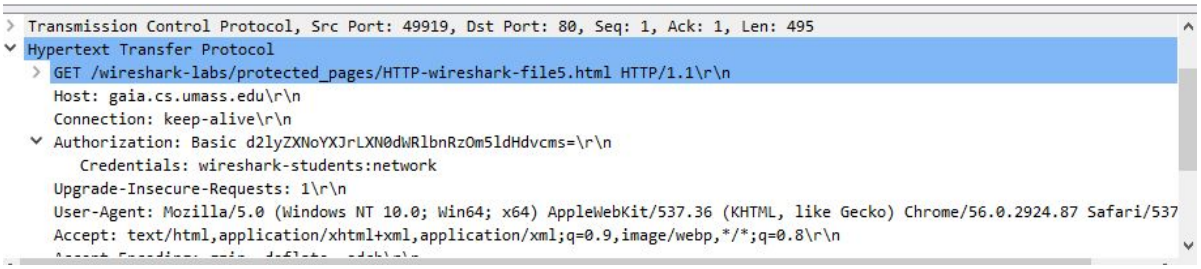


Questions

1. What is the server's response (status code and phrase) in response to the initial HTTP GET message from your browser?
    a. The server's initial response to the GET is status code:401 Unauthorized.



2. When your browser sends the HTTP GET message for the second time, what new field is included in the HTTP GET message?
    a. When the second GET is sent, an "Authorization" field is sent. It is assumed that field is used to authenticate with the server in order to log in.



NOTE: The new 'Authorization' field are the encoded login credentials. The credentials are *not* encrypted, only encoded in Base64 format. Extra security measures must then be made.