

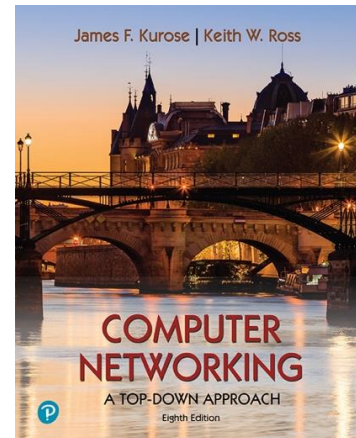
Wireshark Lab:

HTTP v8.1

Supplement to *Computer Networking: A Top-Down Approach*, 8th ed., J.F. Kurose and K.W. Ross

"Tell me and I forget. Show me and I remember. Involve me and I understand." Chinese proverb

© 2005-2021, J.F Kurose and K.W. Ross, All Rights Reserved



Name: Urooba Gohar
Roll No: 22P-9216
Section: BSCS-5A

Having gotten our feet wet with the Wireshark packet sniffer in the introductory lab, 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 Section 2.2 of the text.¹

1. The HTTP CONDITIONAL GET/response interaction

Recall from Section 2.2.5 of the text, that most web browsers perform object caching and thus often perform a conditional GET when retrieving an HTTP object. Before performing the steps below, make sure your browser's cache is empty². 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/wireshark-labs/HTTP-wireshark-file2.html>

Your browser should display a very simple five-line HTML file.

¹ References to figures and sections are for the 8th edition of our text, *Computer Networks, A Top-down Approach*, 8th ed., J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2020. Our authors' website for this book is http://gaia.cs.umass.edu/kurose_ross You'll find lots of interesting open material there.

² See <https://www.howtogeek.com/304218/how-to-clear-your-history-in-any-browser/> for instructions on clearing your browser cache.

- 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” (again, in lower case without the quotation marks) in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.

If you’re unable to run Wireshark on a live network connection (or unable to get your browser to issue an If-Modified-Since field on the second HTTP GET request), you can download a packet trace that was created when the steps above were followed.³ Answer the following 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? (0.3)

Ans: No, we cannot see an “IF-MODIFIED-SINCE” line in the HTTP GET because it can be seen only when the requested resource is already cached on the website, but here since we have deleted the browser history therefore it cannot be seen.

No.	Time	Source	Destination	Protocol	Length	Info
141	2.493407	192.168.62.119	128.119.245.12	HTTP	526	GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
155	2.833448	128.119.245.12	192.168.62.119	HTTP	784	HTTP/1.1 200 OK (text/html)
159	2.864041	192.168.62.119	128.119.245.12	HTTP	472	GET /favicon.ico HTTP/1.1
183	3.119753	128.119.245.12	192.168.62.119	HTTP	538	HTTP/1.1 404 Not Found (text/html)
241	6.743456	192.168.62.119	128.119.245.12	HTTP	638	GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
251	7.036756	128.119.245.12	192.168.62.119	HTTP	293	HTTP/1.1 304 Not Modified
286	8.838645	192.168.62.119	128.119.245.12	HTTP	638	GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
288	9.108686	128.119.245.12	192.168.62.119	HTTP	293	HTTP/1.1 304 Not Modified

2. Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell? (0.3)

Ans: Yes, the server has explicitly returned the contents of the file. We can tell it by the “HTTP 200 OK” message which has returned text/html.

141	2.493407	192.168.62.119	128.119.245.12	HTTP	526	GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
155	2.833448	128.119.245.12	192.168.62.119	HTTP	784	HTTP/1.1 200 OK (text/html)
159	2.864041	192.168.62.119	128.119.245.12	HTTP	472	GET /favicon.ico HTTP/1.1
183	3.119753	128.119.245.12	192.168.62.119	HTTP	538	HTTP/1.1 404 Not Found (text/html)
241	6.743456	192.168.62.119	128.119.245.12	HTTP	638	GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
251	7.036756	128.119.245.12	192.168.62.119	HTTP	293	HTTP/1.1 304 Not Modified
286	8.838645	192.168.62.119	128.119.245.12	HTTP	638	GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
288	9.108686	128.119.245.12	192.168.62.119	HTTP	293	HTTP/1.1 304 Not Modified

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 GET⁴? If so, what information follows the “IF-MODIFIED-SINCE:” header? (0.4)

Ans: Yes, an “IF-MODIFIED-SINCE” line can be seen in the second HTTP GET. The information shows the date and time about when the file was last modified.

³ If you’re unable to run Wireshark on a live network connection, you can download the zip file <http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip> and extract the trace file http-wireshark-trace2-1.

⁴ *Hint:* ideally, you should see an If-Modified-Since header since you’ve just downloaded this page a few seconds ago. However, depending on the browser you’re using, and the format of the server’s earlier response to your initial GET, your browser may not include an If-Modified-Since even if the document has been downloaded and caches. The Chrome browser is pretty good at regularly using If-Modified-Since. But Safari and Firefox are much more finicky about when to use If-Modified-Since. Life isn’t always as easy in practice as it is in theory!

```

Hypertext Transfer Protocol
  GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1\r\n
Host: gaia.cs.umass.edu\r\n
Connection: keep-alive\r\n
Cache-Control: max-age=0\r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;
Accept-Encoding: gzip, deflate\r\n
Accept-Language: en-US,en;q=0.9\r\n
If-None-Match: "173-6234fe6d70020"\r\n
If-Modified-Since: Mon, 30 Sep 2024 05:59:02 GMT\r\n
\r\n
[Full request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html]
[HTTP request 3/4]
[Prev request in frame: 159]

```

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. (1)

Ans: The HTTP status code returned from the server in response to the second HTTP GET is “304 Not Modified” which means that the server has not explicitly returned the contents of the file and the file has not changed since the last request.

241	6.743456	192.168.62.119	128.119.245.12	HTTP	638	GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
251	7.036756	128.119.245.12	192.168.62.119	HTTP	293	HTTP/1.1 304 Not Modified

2. 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://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.

In the packet-listing window, you should see your HTTP GET message, followed by a multiple-packet TCP response to your HTTP GET request. Make sure you your Wireshark display filter is cleared so that the multi-packet TCP response will be displayed in the packet listing.

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.

Answer the following questions⁵:

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

Ans: Only one HTTP GET request was sent, i.e: packet number 38. It also contains the GET message for the bill of rights.

Time	Source	Destination	Protocol	Length	Info
38 0.923186	192.168.62.119	128.119.245.12	HTTP	526	GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1
47 1.180135	128.119.245.12	192.168.62.119	HTTP	535	HTTP/1.1 200 OK (text/html)

6. Which packet number in the trace contains the status code and phrase associated with the response to the HTTP GET request? (1)

Ans: Packet number 47 contains the status code and phrase as a response.

Time	Source	Destination	Protocol	Length	Info
38 0.923186	192.168.62.119	128.119.245.12	HTTP	526	GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1
47 1.180135	128.119.245.12	192.168.62.119	HTTP	535	HTTP/1.1 200 OK (text/html)

7. What is the status code and phrase in the response? (1)

Ans: The status code is 200 and the phrase is OK.

8. How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights? (1)

Ans: 4 TCP segments were needed to carry.

3. 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.

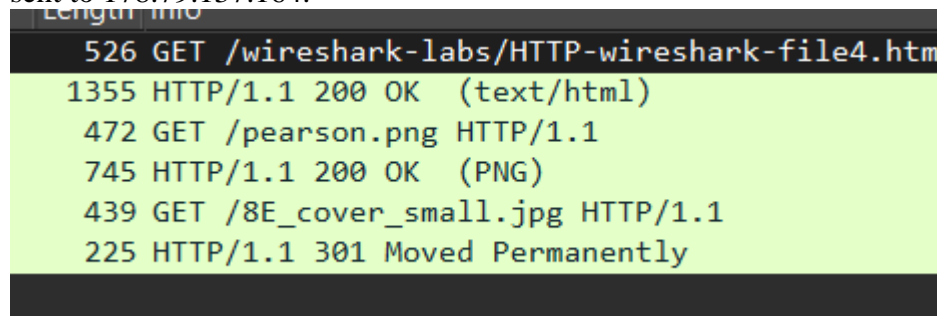
⁵ If you’re unable to run Wireshark on a live network connection, you can download the zip file <http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip> and extract the trace file http-wireshark-trace3-1.

- 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 gaia.cs.umass.edu web site. The image of our 8th edition cover (one of our favorite covers) is stored at a server in France.
- 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 questions⁶:

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

Ans: Three HTTP GET requests were sent by the browser. The first GET was sent to 128.119.245.12. The second GET was sent to 128.119.245.12. The third GET was sent to 178.79.137.164.



```

526 GET /wireshark-labs/HTTP-wireshark-file4.htm
1355 HTTP/1.1 200 OK (text/html)
472 GET /pearson.png HTTP/1.1
745 HTTP/1.1 200 OK (PNG)
439 GET /8E_cover_small.jpg HTTP/1.1
225 HTTP/1.1 301 Moved Permanently
  
```

10. Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two web sites in parallel? Explain. (1)

Ans: The first image, i-e: pearson.png was downloaded at 2.336047 seconds and the second image, i-e: 8E_cover_small.jpg was downloaded at 2.894589 seconds. As the request for the second image was made shortly after the first image, therefore it is in parallel.

4 HTTP Authentication

⁶ If you're unable to run Wireshark on a live network connection, you can download the zip file <http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip> and extract the trace file http-wireshark-trace4-1.

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 79.

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:

- 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-wireshark-file5.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.
- *Note:* If you are unable to run Wireshark on a live network connection, you can use the "classic" http-ethereal-trace-5 packet trace, or other additional traces, as notes in footnote 2, to answer the questions below.

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](http://frontier.userland.com/stories/storyReader$2159)

Answer the following questions⁷:

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

Ans: The server's response to the initial HTTP GET message is: 401 unauthorized which means that an attempt was made to access the resource that requires authentication which was incorrect or missing.

ne	Info
	GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1
	HTTP/1.1 401 Unauthorized (text/html)
	GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1
	HTTP/1.1 200 OK (text/html)

12. When your browser's sends the HTTP GET message for the second time, what new field is included in the HTTP GET message? (1)

⁷ If you're unable to run Wireshark on a live network connection, you can download the zip file <http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip> and extract the trace file http-wireshark-trace5-1.

Ans: The second HTTP GET message contains the credentials, i-e: username and password.

```
▼ Authorization: Basic d2lyZXNoYXJrLXN0dWRlbnRzOm5ldHdvcm5=\r\n
  Credentials: wireshark-students:network
```

The username (wireshark-students) and password (network) that you entered are encoded in the string of characters (d2lyZXNoYXJrLXN0dWRlbnRzOm5ldHdvcm5=) 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 Om5ldHdvcm5= 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.

Fear not! As we will see in Chapter 8, there are ways to make WWW access more secure. However, we’ll clearly need something that goes beyond the basic HTTP authentication framework!

NOTE: Please give valid reasons and a clear explanation in a few lines, for every question you answer!