COMP3331 LAB02

Exercise 3: Using Wireshark to understand basic HTTP request/response messages (marked, include in your report)

We will not be running Wireshark on a live network connection (You are strongly encouraged to try this on your own machine. Pointers provided at the end of this exercise). The CSE network administrators do not permit live traffic monitoring for security reasons. Instead, for all our lab exercises we will make use of several trace files, which were collected by running Wireshark by one of the textbook's authors. For this particular experiment download the following trace file: http-wireshark-trace-1

NOTE: IT IS NOT POSSIBLE TO RUN WIRESHARK VIA SSH. IT IS A RESOURCE INTENSIVE PROGRAM AND IT WOULD SLOW DOWN THE CSE LOGIN SERVERS. IF YOU WANT TO WORK REMOTELY, THEN YOU CAN DOWNLOAD AND INSTALL WIRESHARK ON YOUR PERSONAL MACHINE. WIRESHARK IS AVAILABLE ON ALL LAB MACHINES.

The following indicate the steps involved:

Step 1: Start Wireshark by typing *wireshark* at the command prompt.

Step 2: Load the trace file *http-wireshark-trace-1* by using the *File* pull down menu, choosing *Open* and selecting the appropriate trace file. This trace file captures a simple request/response interaction between a browser and a web server.

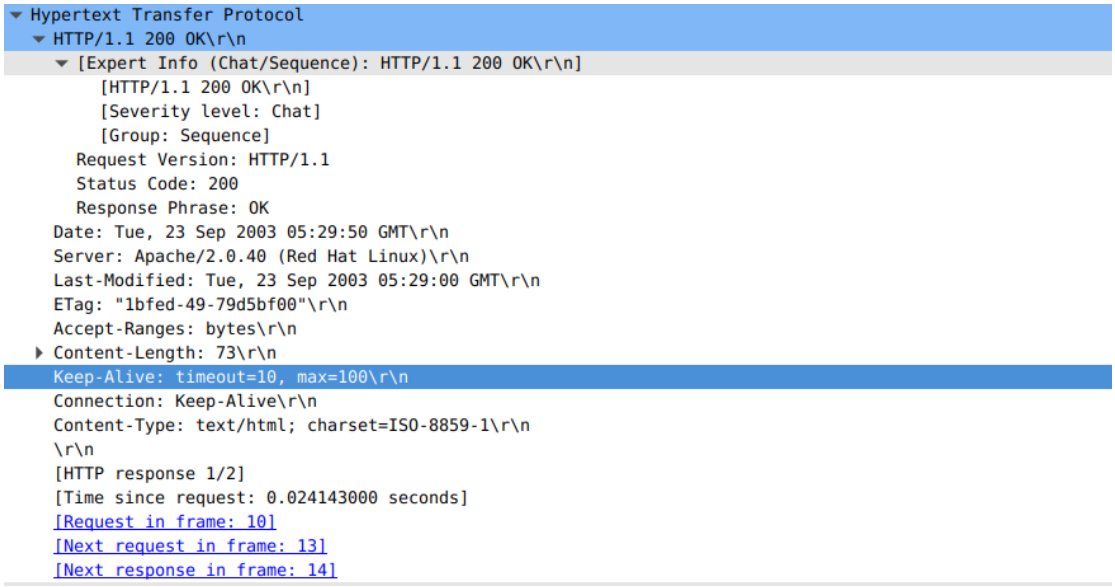
Step 3: You will see a large number of packets in the packet-listing window. Since we are currently only interested in HTTP we will filter out all the other packets by typing "http" in lower-case in the *Filter* field and press Enter. You should now see only HTTP packets in the packet-listing window.

Step 4: Select the first HTTP message in the packet-listing window and observe the data in the packet-header detail window. Recall that since each 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 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: Please neglect the HTTP GET and response for favicon.ico, (the third and fourth HTTP messages in the trace file. Most browsers automatically ask the server if the server has a small icon file that should be displayed next to the displayed URL in the browser. We will ignore references to this pesky file in this lab.)

By looking at the information in the HTTP GET and response messages (the first two messages), answer the following questions:

Question 1: What is the status code and phrase returned from the server to the client browser?



Answer:

status code:200

Response Phrase: OK

Question 2: When was the HTML file that the browser is retrieving last modified at the server? Does the response also contain a DATE header? How are these two fields different?

Answer:

Last modified: Tue, 23 Sep 2003 05:29:00 GMT

Yes, it has. DATE contains the data and time at which message was originated. Last- modified is the date that the file was last modified. It is used for determining if a resource received or stored is the same.

Question 3: Is the connection established between the browser and the server persistent or non-persistent? How can you infer this?

Answer:

Persistent, request and response connection header are **Keep-Alive**, which means use single TCP connections to send and receive multiple HTTP request and responses rather than open a new connection for every single request and response pair.

Question 4: How many bytes of content are being returned to the browser?

Answer:

73 bytes.

Question 5: What is the data contained inside the HTTP response packet?

Answer:

```
<html>\n
Congratulations.  You've downloaded the file lab2-1.html!\n
</html>\n
```

Exercise 4: Using Wireshark to understand the HTTP CONDITIONAL GET/response interaction (marked, include in your report)

For this particular experiment download the second trace file: http-wireshark-trace-2

The following indicate the steps for this experiment:

Step 1: Start Wireshark by typing *wireshark* at the command prompt.

Step 2: Load the trace file *http-wireshark-trace-2* by using the *File* pull down menu, choosing *Open* and selecting the appropriate trace file. This trace file captures a request response between a client browser and web server where the client requests the same file from the server within a span of a few seconds.

Step 3: Filter out all the non-HTTP packets and focus on the HTTP header information in the packet-header detail window.

By looking at the information in the HTTP GET and response messages (the first two messages), answer the following questions:

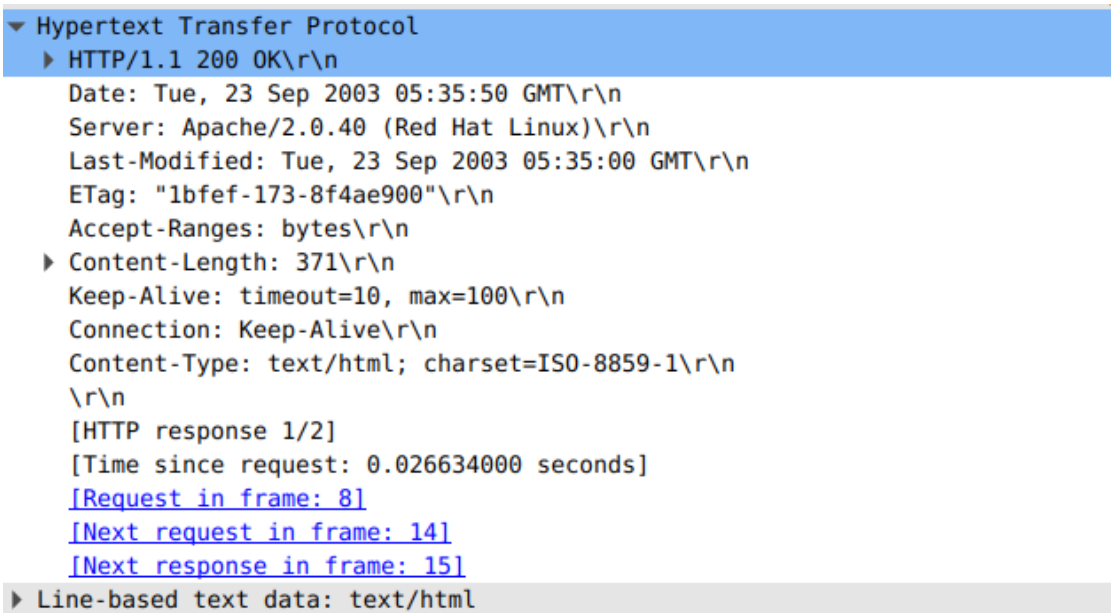
Question 1: Inspect the contents of the first HTTP GET request from the browser to the server. Do you see an “IF-MODIFIED-SINCE” line in the HTTP GET?

Answer:

No.

Question 2: Does the response indicate the last time that the requested file was modified?

Answer:



Yes, it is. That last-modified is Tue, 23 Sep 2003 05:35:00 GMT

Question 3: Now inspect the contents of the second HTTP GET request from the browser to the server. Do you see an “IF-MODIFIED-SINCE:” and “IF-NONE-MATCH” lines in the HTTP GET? If so, what information is contained in these header lines?

Answer:

Yes it have IF-MODIFIED-SINCE and IF-NONE-MATCH line.

If-Modified-Since: Tue, 23 Sep 2003 05:35:00 GMT

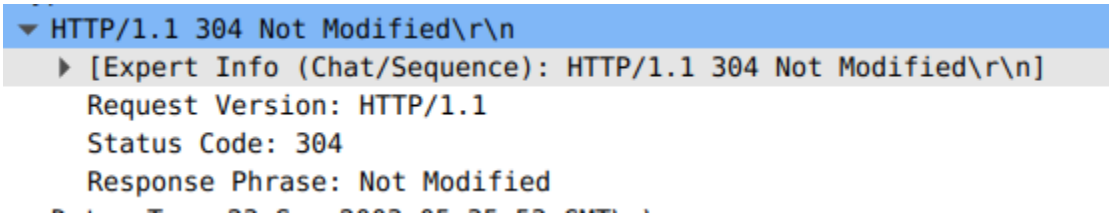
IF-MODIFIED-SINCE contains day, time, date. Similar like DATE and LAST-MODIFIED header from previous response message.

If-None-Match: "1bfef-173-8f4ae900"

IF-NONE-MATCH contains Etag value.

Question 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.

Answer:



Status Code:304, Phrase: Not Modified

Not return the contents of the file. there is no need to retransmit the requested resources because the resource has not been modified since the version specified by the request headers. Client still has a previously-downloaded (local) copy.

Question 5: What is the value of the Etag field in the 2nd response message and how it is used? Has this value changed since the 1 st response message was received?

Answer:

```
Date: Tue, 23 Sep 2003 05:35:53 GMT\r\n
Server: Apache/2.0.40 (Red Hat Linux)\r\n
Connection: Keep-Alive\r\n
Keep-Alive: timeout=10, max=99\r\n
ETag: "1bfef-173-8f4ae900"\r\n
```

ETag: "1bfef-173-8f4ae900".

Not changed. The ETag use to compare the server's ETag value and check whether any different between the cache and server resources. To determine whether need to retransmit the requested resources or just use local cached resource.

Exercise 5 Example (python3 .7.3 CSE machine):

```
z5223796@vx2:/tmp_amd/kamen/export/kamen/4/z5223796/Desktop/comp3331/lab02$ pyth
on3 PingClient.py 127.0.0.1 1400
Ping to 127.0.0.1, seq = 3331, rtt = 251 ms
Ping to 127.0.0.1, seq = 3332, rtt = 68 ms
Ping to 127.0.0.1, seq = 3333, rtt = 175 ms
Ping to 127.0.0.1, seq = 3334, rtt = 62 ms
Ping to 127.0.0.1, seq = 3335, rtt = 157 ms
Ping to 127.0.0.1, seq = 3336, rtt = 143 ms
Ping to 127.0.0.1, seq = 3337, rtt = 80 ms
Ping to 127.0.0.1, seq = 3338, rtt = 48 ms
Ping to 127.0.0.1, seq = 3339, rtt = 95 ms
Ping to 127.0.0.1, seq = 3340, rtt = 74 ms
Ping to 127.0.0.1, seq = 3341, rtt = 175 ms
Ping to 127.0.0.1, seq = 3342, timeout
Ping to 127.0.0.1, seq = 3343, rtt = 157 ms
Ping to 127.0.0.1, seq = 3344, timeout
Ping to 127.0.0.1, seq = 3345, rtt = 197 ms
transmission finished
MINIMUM RTT is 48 ms, MAXIMUM RTT is 251 ms, AVERAGE RTT is 130 ms
```

```
z5223796@vx2:/tmp_amd/kamen/export/kamen/4/z5223796/Desktop/comp3331/lab02$ java
 PingServer 1400
Received from 127.0.0.1: PING 3331 1592821003986.8801
   Reply sent.
Received from 127.0.0.1: PING 3332 1592821004238.066
   Reply sent.
Received from 127.0.0.1: PING 3333 1592821004307.083
   Reply sent.
Received from 127.0.0.1: PING 3334 1592821004482.9004
   Reply sent.
Received from 127.0.0.1: PING 3335 1592821004545.766
   Reply sent.
Received from 127.0.0.1: PING 3336 1592821004703.7446
   Reply sent.
Received from 127.0.0.1: PING 3337 1592821004847.193
   Reply sent.
Received from 127.0.0.1: PING 3338 1592821004928.1875
   Reply sent.
Received from 127.0.0.1: PING 3339 1592821004977.228
   Reply sent.
Received from 127.0.0.1: PING 3340 1592821005073.005
   Reply sent.
Received from 127.0.0.1: PING 3341 1592821005147.7383
   Reply sent.
Received from 127.0.0.1: PING 3342 1592821005323.6855
   Reply not sent.
Received from 127.0.0.1: PING 3343 1592821005924.482
   Reply sent.
Received from 127.0.0.1: PING 3344 1592821006082.0796
   Reply not sent.
Received from 127.0.0.1: PING 3345 1592821006682.8582
   Reply sent.
```