Lab1 TDDE35

Simon Jakobsson simja649
Erik Halvardsson eriha353

Note: We use the provided ethereal traces for all questions and tasks in this assignment unless otherwise specified.

1. The browser we're using is running HTTP/1.1
This can be seen first in the info section of the GET request packet capture listing, and also in the details section under Hypertext Transfer Protocol->GET->Response Version

2. Accept-Language: en-GB,en;q=0.9,sv-SE;q=0.8,sv;q=0.7,de-DE;q=0.6,de;q=0.5,en-US;q=0.4
(Corresponds to UK English, Swedish, Geman, and US English, respectively)
This can be found in the details section under Hypertext Transfer Protocol->Accepted-Language

3. Source IP is 192.168.1.150, which is local host, and destination IP is 128.119.245.12
This can be seen in the packet capture listing.

4. 200 OK
Can be seen in the info section of the response in the packet capture listing, or Hypertext Transfer Protocol->GET->Status Code in the details section

5. Last-Modified: Mon, 30 Mar 2020 05:59:03 GMT (received 31-03-2020 11:12)
Can be found in the details section of the server response, under Hypertext Transfer Protocol->Last-Modified

6. Content-Length: 81 bytes
Can be seen in the details section of the server response, under Hypertext Transfer Protocol->Content-Length

7. We did not find any http headers in the raw data that was displayed in the packet-listing window.

Task A:
From task A we can observe that all meta-data can be retrieved from the Hypertext transfer protocol section. From Wiresharks' interface we can also get the info of source and destination IP to know who's sending and whose receiving. We can also get the info of what is being sent and what is being retrieved (in the info bar)

8. Yes.
If-Modified-Since: Tue, 31 Mar 2020 05:59:02 GMT

9. Yes, the server did explicitly return the contents.
We can tell because it contains the Content-Type: text/html; charset=UTF-8\r\n and Content-Length headers

10. Yes, it contains the following: If-Modified-Since: Tue, 31 Mar 2020 05:59:02 GMT

11. The second response returns HTTP 304 Not Modified. The server did not explicitly return the contents of the file this time.

Task B:
Our observation of task B is that the first time we call a server it gives us data, when it has been modified, and what content it has. If we then at a later time try to access the same server and it has not been modified, we get a response that it has not been modified since the last time we tried to access the content, thus the cache will have the data we need and we do not need to get the information from the server. If we clear the cache we do not have the information from previous visits to the server, and we would have to get it back with a new GET request.

12. The browser sent 1 GET request to the server. Packet number 8.

13. Packet number 10, status code 200 OK.

14. Answer: 4 data-containing TCP segments. xxx
The first three segments can be seen in the list view in Wireshark as coming from address 128.119.245.12 using TCP with a non-zero length. These have numbers 10, 11, and 13 in the list. The last segment is contained in the HTTP/1.1 200 OK response, number 14 in the list. According to Wireshark the last segment has a length of 436 bytes, while the 3 first segments are 1460 bytes.

15. No. We cannot find anything in the HTTP section of either the GET request or the server response that relates to TCP segmentation.

Task C:
Our observation of task C is that we have multiple packets that are delivered to get all data to be transferred. We make a GET request at packet 8 after establishing a connection. Due to a big package we get 4 packets to download. 10,11,13, the last segment arriving in packet 14 along with 200 OK. Then packet 14, having received 200 OK confirms that we have gotten all the packages. Which we then reply to the server we have gotten.

16. 3 GET requests.
The first goes to: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html
The second goes to:

http://gaia.cs.umass.edu/pearson.png
And the third goes to:
http://gaia.cs.umass.edu/~kurose/cover_5th_ed.jpg

17. They are downloaded in parallel.

Task D:
We have 3 separate GET HTTP requests. The first GET request asks for the main website, which contains information telling the browser where to get the pictures from, and the other two GET requests are for the URI to download the pictures.
Additionally, the two pictures are downloaded in separate TCP streams, with separate TCP Source Ports.

20/Task E. According to RFC 2068 section 19.7.1, Connection: keep-alive implies that the connection will be kept alive until either the client terminates the connection, or the connection times out. This has the benefit that the connection can be persistent, so you don’t have to redo the handshake every time the client sends a request.
According to RFC 2616 section 14.10, Connection: close means that the connection will be closed immediately after the server sends its response. This method requires more overhead in each request, as the handshake needs to be performed for every request. Connection: close is more efficient in the situation where the client sends only a single HTTP request.