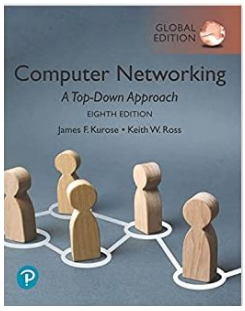
Lab1: Wireshark (HTTP & DNS)

Computer Communication

Romaric Duvignau

January 26, 2023

Based on and modified from **Wireshark Lab from J.F. Kurose and K.W. Ross** with contributions and support from Hans-Martin Heyn, Roman Melnik, Ali Salehson and Marina Papatriantafilou.



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

Lab1, 7 tasks, 24 questions.

Packet and traffic analysis using network sniffing software This lab is adapted from selected parts from the Wireshark Labs which can be downloaded from the home page of the course book ("Student resources" / "Wireshark labs"). **Wireshark**[1] is a popular network protocol analyzer under the *GNU General Public License*.

Labs' format: Read instructions and practical information available in Canvas about (i) forming student groups, (ii) attending lab sessions, (iii) some time estimation for completing each part and (iv) how the submission process works.

Purpose: To learn how to listen to local network traffic and analyze different protocols as well as learning to use some useful network utilities.

Background/Preparation: Having read relevant sections in **Chapter 2** (especially **§2.2 "The Web and HTTP" and 2.4 "DNS–The Internet's Directory Service"**) of the course book and having completed "Preparation to Lab1" available in Canvas.

Contents

[1] available to download using the URL `http://www.wireshark.org/download.html`.

1 HTTP

Main Task: Use Wireshark to investigate the HTTP protocol in operation. You will explore several aspects of the HTTP protocol: the basic GET/response interaction, HTTP message format and retrieving HTML file with embedded objects.

1.1 Basic HTTP GET/Response interaction

Let's begin the exploration of HTTP by downloading a very simple HTML file, one that is very short, and contains no embedded objects. Do the following **in the same order**:

1. Start up your web browser.
2. Start up the Wireshark packet sniffer, as described in the **Introductory lab**. Enter the letters **"http"** in the field of the display filter. Click the button **Apply** (or press **Enter**) so that only captured HTTP messages will be displayed in the packet list.
3. Begin Wireshark packet capture.
4. Open in your browser[2]:
   `http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html`
5. Your browser should display the very simple, one-line HTML file.
6. Stop Wireshark packet capture.

The window-pane of the packet list will show that two HTTP messages were captured: the GET message (from your browser to the web server) and the response message from the server to your browser.

Tips

You should ignore any HTTP GET request and response for `favicon.ico`. If you see a request to this file, it is because your browser is automatically asking the server if it has a small image file to be displayed next to the URL in the address field.

The window-pane of the packet details shows details of the selected message. case the HTTP GET message, which is highlighted in the packet list). Recall that since the HTTP message is carried inside a TCP segment, which is carried inside an IP datagram, which is carried within an Ethernet frame, Wireshark displays:

1. The link-layer Frame (called packet) with MAC address;
2. IP-header with information such as source and destination IP addresses;
3. TCP information: source/destination port numbers, etc;
4. Application-layer message: HTTP header and HTTP data.

By inspecting and looking at the information in the HTTP GET and response messages, answer the following questions.

Task 1. Answer the following questions:

(a) What is the IP address of your computer and of the **gaia.cs.umass.edu** web server?

(b) What is the HTTP version, status code and phrase returned from the server to your browser?

[2]If you opened the link before starting the packet capture, simply refresh the webpage while bypassing your browser's cache (by pressing Ctrl+F5 in your browser while the page is open); alternatively you can clear also your browser's cache (in Chromium, enter `chrome://settings/clearBrowserData` and click "Clear Data"), then restart the packet capture, and finally re-open the link.

(c) Explore the HTTP content and find the HTTP header lines that provide the following information:

(1) The languages accepted by your browser as indicated to the server.

(2) The last modification time of the HTML-file, provided by the server.

(3) The size of the file (bytes of HTTP content) returned to your browser.

Hint

It is highly desired to minimize the amount of non-HTTP data displayed so make sure that the boxes at the far left of the Frame, Ethernet, IP and TCP information have a plus sign or a right-pointing triangle, and the HTTP line has a minus sign or a down-pointing triangle (which means that all information about the HTTP message is displayed).

1.2 HTTP conditional GET/Response interaction

Recall from **Sections 2.2.5** of the course book that most web browsers perform object caching and thus perform the conditional GET when retrieving HTTP objects.

Now do the following:

- Start up your web browser.
- Start up Wireshark, and apply the filter "`http && ip.addr==128.119.245.12` ", so that only captured HTTP messages exchanged with umass web server will be displayed in the packet-list panel.
- 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.
- Enter the same URL into your browser again (or simply refresh the page on your browser by pressing F5).
- Stop Wireshark packet capture.

Task 2. Answer the following questions:

(a) Inspect the content of the first HTTP GET *request* from your browser to the server. Is there an `If-Modified-Since` header line in the HTTP GET message? Why not?

(b) Inspect the content of the server *response*. Has the server explicitly returned the content of the file? How can you tell?

(c) Now inspect the content of the second HTTP GET *request* from your browser to the server. Is there an `If-Modified-Since` header line in the HTTP GET message? If so, what information follows the `If-Modified-Since` header line?

(d) What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Has the server explicitly returned the content of the file? Explain.

1.3 HTML documents with embedded objects

Now you can look at what happens when your browser downloads a file with embedded objects, i.e. a file with references to other objects (in the example below, image files) that are stored on another server(s). Do the following and answer the questions given below:

- Start up your web browser.
- Start Wireshark capture with the following filter: "`http && (ip.addr == 128.119.245.12 || ip.addr == 178.79.137.164)`"
- 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
- Once the page is fully loaded, stop Wireshark packet capture.

Tips

You can combine filters with `&&` (and) e.g. "`ip.addr == 128.119.245.12 && http`" will only display http packets where the address `128.119.245.12` appears (either as source or destination). Use instead `||` to perform a "logical or" such that "`udp.port == 53 || tcp.port == 53`".

Task 3. Answer the following questions:

(a) How many HTTP GET request messages has your browser sent? To which IP-address is each of these GET requests sent?

(b) How can you tell whether your browser has downloaded the two images serially, or whether they have been downloaded from the two web servers in parallel? Explain.

2 DNS

Main Task: Make extensive use of the nslookup tool, available in most Linux/Unix and Microsoft platforms. Also use Wireshark to investigate the operation of DNS at the client side.

Domain Name System (DNS) mainly translates hostnames to IP addresses, fulfilling a critical role in the Internet infrastructure. In this lab, you'll take a closer look at the client side of DNS. Recall that the client's role in the DNS is relatively simple: the client sends a *query* to its local DNS server, and receives a *response* back. As shown in Figures **2.21** and **2.22** in the course book, much can go on "under the covers," invisible to the DNS clients, as the hierarchical DNS servers communicate with each other to either recursively or iteratively resolve the client's query.

2.1 Using nslookup

Task 4. Provide the full line of command (using *nslookup*), its result, and the type of DNS RR records asked when retrieving the following:

(a) The IP address of a Web server in Africa *that you pick by yourself*[a].

(b) The mail servers for a university in South America *that you pick by yourself.*

(c) The address of a name server in charge of `svt.se`.

(d) Query the name server found in the previous question about the IP address of

www.amazon.com. Do you get an answer and why?

[a] "Governments' are easy picks, e.g. type "government + country" in google.

The next task asks you to perform an **iterative DNS lookup** in the same fashion as it is done at your local DNS name server (as illustrated by **Figure 2.19** in your text book). The goal of this task is to retrieve the answers for question (a) of the previous task but **without** asking your local DNS server along the process. Also when asking a name server, you are only **allowed to use its IP address**. First, visit the website https://root-servers.org and find the IP address of one of the root servers. Here is an example (some response lines have been omitted for clarity):

```
duvignau:~$ nslookup -type=ns ma 192.5.5.241
a.tld.ma        internet address = 81.192.171.83

duvignau:~$ nslookup -type=ns maroc.ma 81.192.171.83
dns3.menara.ma  internet address = 81.192.21.73

duvignau:~$ nslookup www.maroc.ma 81.192.21.73
Name:   www.maroc.ma
Address: 81.192.44.195
```

Task 5. (a) Write a list of command-lines[a] to retrieve the answer to Task 4 (a) such that each command always asks directly a name server using its IP address.

[a] A *command-line* refers to a full line of command including the command itself that is run, its option(s) if any and its argument(s).

Tips

Note, there are only **13 root IP addresses** and those are normally cached in any DNS server (in particular your local DNS!). To find the address of one of the root servers, you can also try using the command "`nslookup -type=ns .`", however, you will likely find one of their canonical names. You can ask your local DNS server to solve one of these names to find the IP of a root server; here you will not be able to do it starting by the tld etc as first you need to know a root server's address to do so!

Hint

In the task, you need to ask "iteratively" each traversed domain in the web server's name you picked starting from the root. In rare cases, you may need to ask directly a name sever for several parts of the domain name that you picked (try this only if first asking the last domain part has failed, e.g. as in "co.uk" for British TLD). **Do not hesitate to change the domain you had initially picked!**

2.2 Tracing DNS with Wireshark

Now that you are familiar with *nslookup*, you'll investigate the DNS messages by capturing the DNS packets that are generated by ordinary Web-surfing activity.

- Open your browser and empty the DNS cache: enter in the address bar chrome://net-internals/#dns and click on "Clear host cache".
- Open Wireshark and apply a filter so that only DNS messages containing your host IP address should be displayed. This filter should remove all packets that neither originate nor are destined to your host **and** only DNS packets should be displayed.

- Start packet capture in Wireshark.
- Visit the Web page: `http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html`
- Stop packet capture.

Task 6. Answer the following questions:

(a) What was the filter you've applied?

(b) Locate the DNS *query* and *response* messages resolving `www.cse.chalmers.se`.

(1) Are they transported using UDP or TCP? Explain briefly why.

(2) What is the destination port for the DNS *query* message? And what is the source port of DNS *response* message? Are they the same?

(3) To what IP address is the DNS *query* message sent? Which network device is this IP address associated with?

(c) Compare the DNS *query* and *response* messages, and give the "sections" they each contain (see below hint). What was the "type" of query used here?

(d) This web page contains an image. Before retrieving the image, does the host's DNS client issue new DNS queries? Why or why not? (hint: check the image's domain).

(e) Try reloading the page. Did you capture any DNS messages, and why so?

Hint

Each DNS message consists of a fixed-length *header* and up to 4 *sections*: **Queries**, **Answers**, **Authoritative nameservers**, **Additional records**; some sections may be empty.

Tips

Wireshark indicates how many records are contained in each section.

2.3 Packet capture and nslookup

The goal of this final task is to understand the usage of canonical names by the DNS protocol. We'll capture DNS packets when executing command-lines using *nslookup*. At the end, we will find out the IP address of `www.tue.nl` (i.e. the hostname of the web server of the **Eindhoven University of Technology**) by asking directly the competent name servers.

For the next task, you will execute 4 different *nslookup* command-lines (i), (ii), (iii) and (iv), in bold in the task. For (i) to (iii), use the following procedure:

- Start a Wireshark packet capture using the same filter that you've used for Task 6.
- Execute the appropriate nslookup command-line in your terminal.
- Stop packet capture.

Task 7. (i) Using *nslookup*, ask your local DNS for the IP address of `www.tue.nl` and observe the captured packets in wireshark.

(a) What is the canonical name for `www.tue.nl`? What are the IP addresses returned

by nslookup? Examine the DNS *response* message. Describe the content of the captured DNS message and its different (non-empty) sections.

(ii) Using *nslookup*, ask your local DNS for the address of a name server on `tue.nl` domain and observe the captured packets in wireshark.

(b) How many name servers did you find? Does the DNS response contain their IPs?

(iii) Using *nslookup*, ask a name server on `tue.nl`'s domain to provide you with the IP address of `www.tue.nl` and observe the captured packets in wireshark.

(c) Which command-line did you use for that purpose? What is the answer you've got?

(d) How many name servers are actually responsible for `web.w3.tue.nl` and from how many Top-Level Domains (TLD) do they belong to? (Hint: check the "authoritative nameservers" section in wireshark!)

(iv) Using *nslookup*, query a name server discovered in the previous question to provide you with the IP address associated with `web.w3.tue.nl`.

(e) What is the command-line you've used? Does the received answer match what you got in (a)?

Tips

Remember that AAAA records are similar to A records but ask for IPv6 addresses.

Observe that when executing sub-task (ii) and in general when looking for name servers, the TLD name server is the one providing you with the IP address you are looking for. There is indeed a circular problem here, as it is not possible for the name server of the domain to provide you with its own IP address! The information is actually contained in a so-called glue record, that is a combination of a name server and its IP address, and are provided by top-domains when looking for sub-domains that are under their authority.