TÉCNICO LISBOA

Computer Networks — 2021/22		Assignment:	Lab 3
Understanding the Web		Issued:	2021-12-06
The Web & The HTTP Protocol		Submission Due:	2021-12-09
Authors:	Francisco Chamiça Pereira Prof. Luis D. Pedrosa	Comments Due:	2021-12-12
		Version:	1.0

Submission note. You will need to later submit all the answers to this lab's questions in Moodle. As such, we recommend that you take note of each answer during the lab so that later you can just copy them into Moodle. Be sure to also save any screenshots you take, as you can later upload them into your Moodle submission.

1 Web/HTTP

The goals of this lab are:

1. Analyze waterfall charts and understand resource dependencies in a webpage.
2. Understand the difference between a *persistent* and a *non-persistent* connection.
3. Understand the concept of virtual hosting.
4. Show why unencrypted authentication requests passing through a network are unsecure.

Important. Using the provided virtual machine execute the command `git pull origin master` in `/home/rc/lab-files`. Next, execute `/home/rc/lab-files/lab-http/setup.sh <student #>`, replacing `<student #>` with your own IST student ID number (it should be a 5-6 digit number). Leave the command running until you have completed the lab, at which point you can stop it with `Ctrl-C`.

Inside the VM, open the browser (the VM comes with `Firefox`) and visit `cats.rc.com`. Take a look at its HTML (if you are using `Firefox` you can right-click the page and select *View Page Source*, or press `Ctrl-U`).

Q1 Considering the HTML you just analyzed, how many HTTP requests does the browser need to make in order to fully render the page?

Still in `cats.rc.com`, now open the developer tools of your browser and access the *Network* tab (you can access the developer tools by pressing `Ctrl+Shift+C`). Now refresh the page. You can now see every HTTP request made by the browser regarding `cats.rc.com`. Take special attention to the waterfall chart drawn on the right side.

Q2 Submit a screenshot of the waterfall chart. Are all requests made at the same time, or is a specific request made first? Why does this happen?

Open `wireshark` and watch interface *Loopback*. Add the filter `http.host == "cats.rc.com"`, go back to the browser and refresh the page. Now come back again to `wireshark`. You should see the same requests that were presented in the waterfall chart. Right-click on the *Get /* request, and select *Conversation filter* > *TCP*. This gives you all the TCP and HTTP packets associated with that specific request.

Q3 Is the connection *persistent* or *non-persistent*? Include a screenshot from `wireshark` to justify your answer.

Now instead of going to `cats.rc.com`, visit `dogs.rc.com`. The source page is very similar to the one retrieved by visiting `cats.rc.com`: it should request the same number of resources (but now instead of images of cats, we receive images of dogs). Open `wireshark`, apply the filter `http.host == "dogs.rc.com"`, and refresh the page. Again, select the packet corresponding to the HTTP request *Get /*, and apply the TCP conversion filter.

Q4 This time, is the connection *persistent* or *non-persistent*? Again, include a screenshot from `wireshark` to justify your answer.

Q5 Compare the destination IP of the requests *Get /* to `dogs.rc.com` and `cats.rc.com` (you can retrieve this information from `wireshark`). How does the server know which content it should give (i.e., the webpage related to dogs, or the one related to cats)? *Hint: look at the HTTP request headers. This technique for disambiguating hosts is called virtual hosting.*

Using `wireshark`, apply the filter `http.host == "dogs.rc.com"`. Even without accessing the webpage, you should see requests coming in. It seems like someone else is accessing `dogs.rc.com`...

Q6 Take a screenshot of this misterious request. What is its source IP? What HTTP request type is it? Which URL is being accessed?

Go visit this URL. It seems that you need to authenticate yourself to visit this webpage.

Q7 Look at the misterious request and analyze its HTTP headers. Retrieve the authentication information from that request, and use it to access the protected webpage. Explain which header you used, and provide the authentication information retrieved.

The relevant authentication header uses base64 to encode its information. **Base64 is not encryption**, and this is why you can steal this information and authenticate as another user.