Lab 1: HTTP Requests and Responses

QUICK REVIEW

The HyperText Transfer Protocol (HTTP) had been in vogue since its first appearance. Ever since it was implemented for making it easier for scientists to share and access data, security was always an afterthought. As security breaches happened, new security patches were invented and bolted on. What is vulnerable, needs to be protected. There is a myriad of aspects to consider when looking to secure a site, and HTTP headers are a good place to start. Most of them are not all that complicated to implement. Keeping up with HTTP security headers best practices provides another security layer on top of your web assets.

When a user tries to access a page, his browser requests it from a web server. The server then responds with the content along with appropriate HTTP Response Headers which contain meta data, status error codes, cache rules and so on. A big subset of those headers are security headers which instruct your browser exactly how to behave when it handles your website's content and data.

HTTP security headers are a great way to tighten your website's security. There is actually no logic scenario when you shouldn't use them. By setting up your security headers correctly not only you help protect your site, but your users as well. This will also help you cut down on security flaws and working hours invested in tracking and fixing them. Setting security headers, the right way and keeping them up to date will greatly reduce the amount of risk mitigation actions needed in the future.

In this Lab, we will take a deeper look at the http requests and responses.

Important Notice:

Please carefully read the disclaimer declaration on the course webpage, before you start the lab practice, and make sure you fully understand all statements. The disclaimer is available on https://hogeschool.github.io/Software-Quality.

LAB PRACTICES

1.1. GET Request and Response using Postman

- **Choose your method**

  In this example, we are making a **GET** request to retrieve data from the server.

- **Enter a URL** (LIST USERS)

  Now let's send our first API request! Enter https://reqres.in/api/users?page=2 into the URL field.

- **Send a request**

  Click the "Send" button and inspect the returned response body.

Look at the response header and discuss it with other students.

Retry the request with different page numbers and find how many pages are available on the database.

1.2. GET Request and Response using Online Tool

Using API tester tool, you can make HTTP requests, extract values from the responses, assert the values are correct, reuse variables across steps, or inject custom logic using JavaScript. API tester Beta version is available on https://apitester.com.

Repeat Practice 1.1 using this online tool.

Compare the results with the results of Practice 1.1 and discuss it with other students.

1.3. More GET Requests and Responses

Repeat Practices 1.1 and 1.2 for the following URLs:

https://reqres.in/api/users/2 (SINGLE USER)
https://reqres.in/api/users/23 (SINGLE USER NOT FOUND)
https://reqres.in/api/unknown (LIST <RESOURCE>)
https://reqres.in/api/unknown/2 (SINGLE <RESOURCE>)
https://reqres.in/api/unknown/23 (SINGLE <RESOURCE> NOT FOUND)

1.4. POST Request and Response

Repeat Practices 1.1 and 1.2 with the **POST** request for the following URLs, using the given data below:

1.4.1. https://reqres.in/api/users (CREATE)

```
{
    "name": "morpheus",
    "job": "leader"
}
```

1.4.2. https://reqres.in/api/register (REGISTER - SUCCESSFUL)

```
{
    "email": "eve.holt@reqres.in",
    "password": "pistol"
}
```

1.4.3. https://reqres.in/api/register (REGISTER - UNSUCCESSFUL)

```
{
     "email": "sydney@fife"
}
```

1.4.4. https://reqres.in/api/login (LOGIN - SUCCESSFUL)

```
{
    "email": "eve.holt@reqres.in",
    "password": "cityslicka"
}
```

1.4.5. https://reqres.in/api/login (LOGIN - UNSUCCESSFUL)

```
{
    "email": "peter@klaven"
}
```

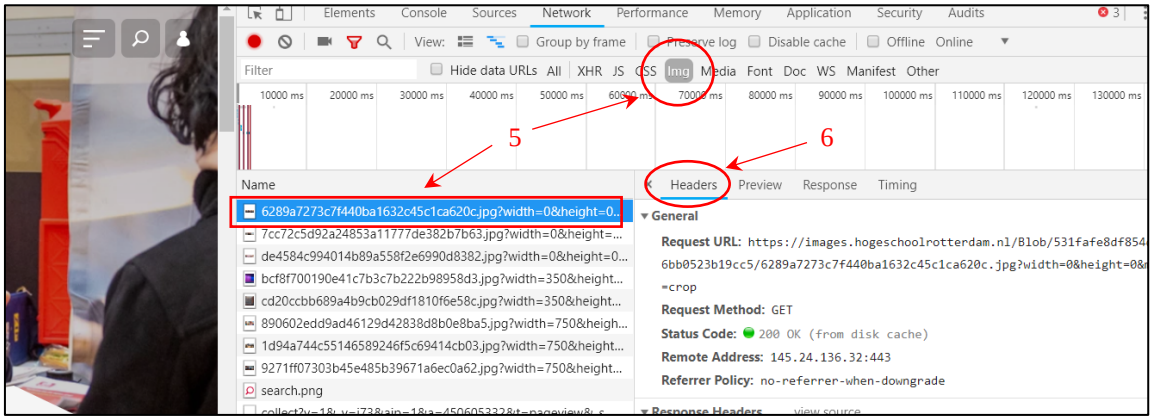
1.5. Referer Header

In this practice, we would like to look at Refer Header using Google Chrome.

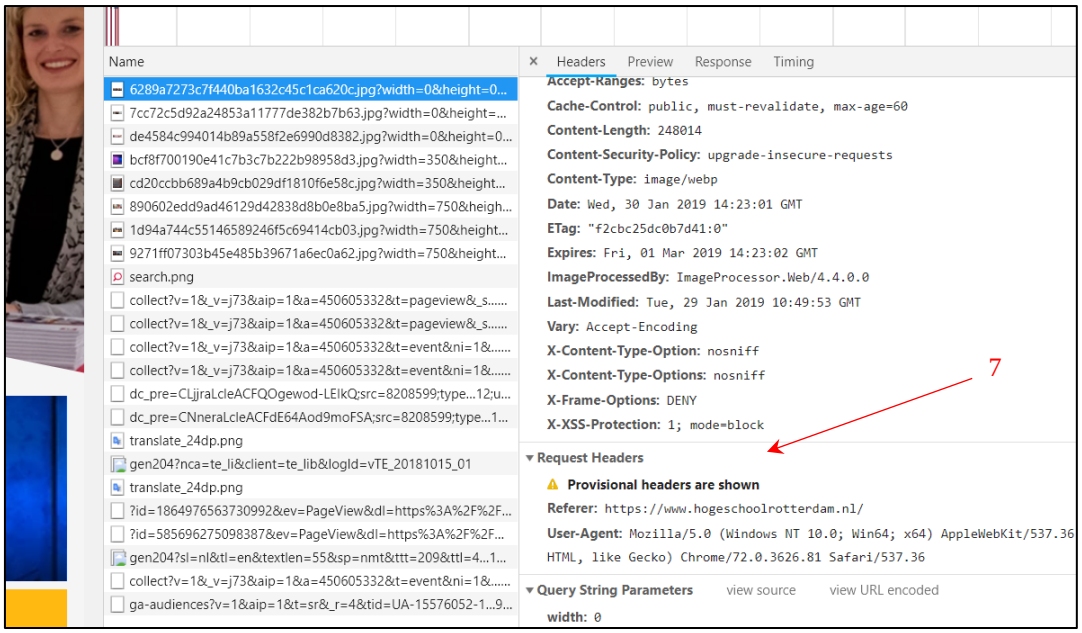
For this purpose, open your Google Chrome (if you do not have Google Chrome, please install it).
Follow the instructions below and note your observations:

1. Go to https://www.hogeschoolrotterdam.nl/

2. In the `Developer Tools` (you can find it in Chrome Menu), go to the `Network` view (If it was not open when you loaded the page, you will need to reload to get it populated).
3. Click on the main page request (It should be at the top of the list).
4. When you click, the right pane will probably show the HTML for it.
5. Click on `Img` tab, then Select an image in the list.

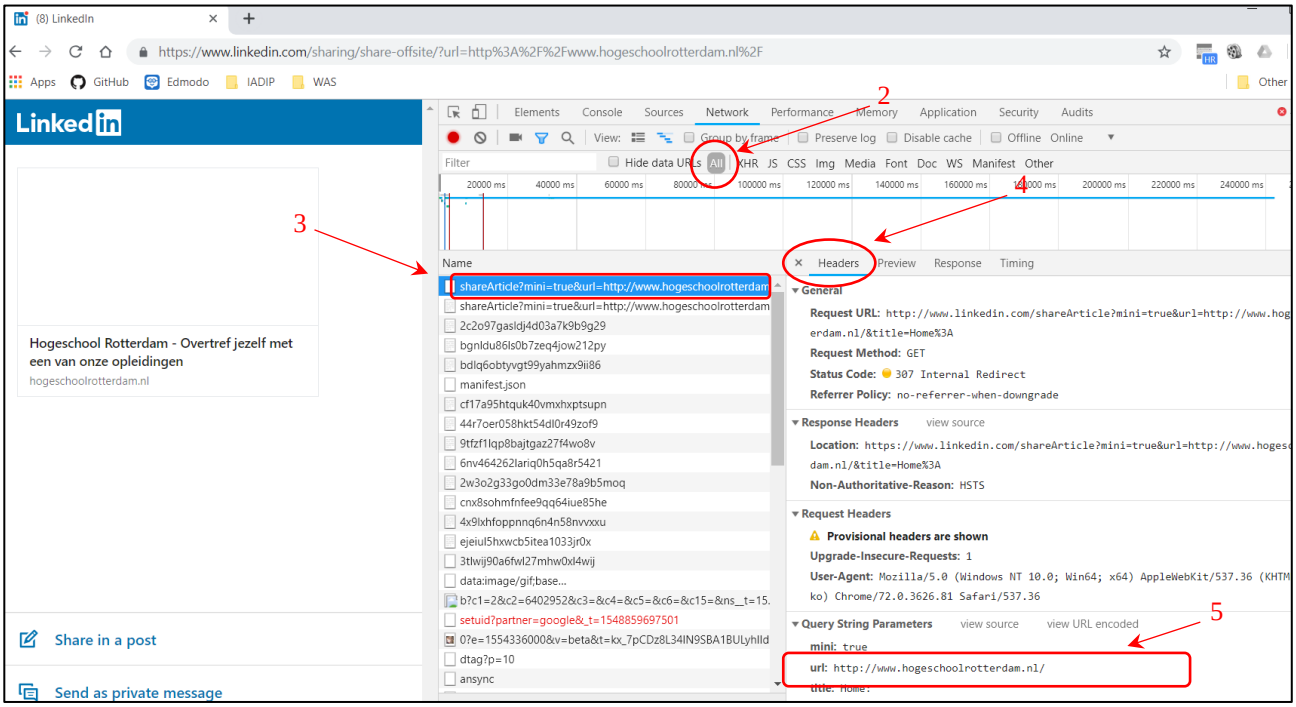


6. Click on the `Headers` tab on the right side.
7. The Referer is shown in the list of `Request Headers` on the right panel (Scroll down to find it).



1.5.1. LinkedIn

1. On the page of HR, click on the LinkedIn icon, at the bottom of page (The icon with share this page on LinkedIn).
2. Click on `all` items tab.
3. On the `Name` column, click on the first item (you may try other objects too).
4. On the right column, click on the `Headers` tab.
5. At the bottom, in `Query String Parameters` section or other sections, try to find the refere URL: `https://www.hogeschoolrotterdam.nl/`



1.5.2. Instagram

Return to HR main page and Repeat the practice 1.5.1 for Instagram link.

FURTHER EXERCISES (HOMEWORK)

1. If you could not complete all the Lab practices in the class, please complete them at home and note your observations.