

Lab 1: HTTP Requests and Responses

QUICK REVIEW

The HyperText Transfer Protocol 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.

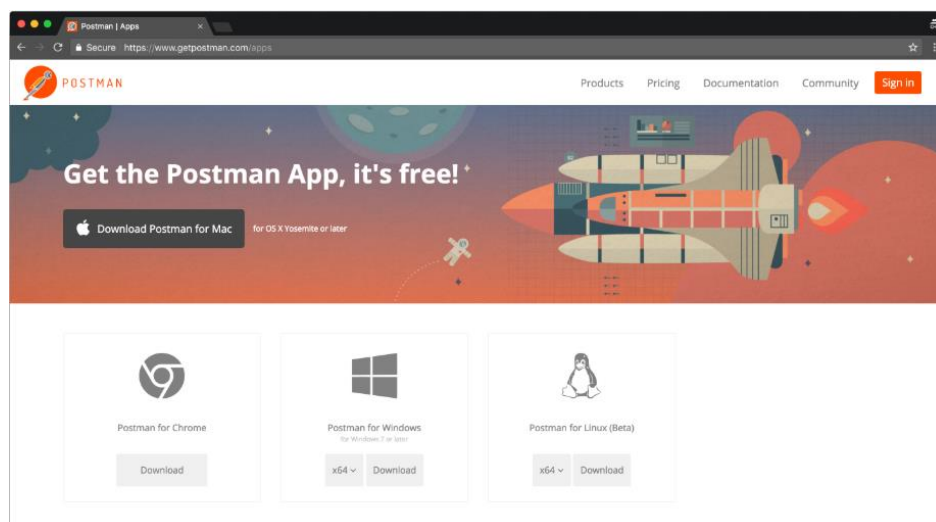
HTTP security headers are a great way to tighten your website's security. There is actually no logic scenario when you shouldn't use them. By setting up your security headers correctly not only you help protect your site, but your users as well. This will also help you cut down on security flaws and 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.

LAB PRACTICES

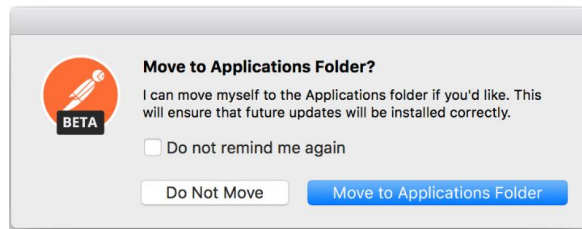
1.1. Installation of Postman

Postman is a powerful tool for performing integration testing with your API. It allows for repeatable, reliable tests that can be automated and used in a variety of environments and includes useful tools for persisting data and simulating how a user might be interacting with the system. Postman is available as a native app for macOS, Windows, and Linux operating systems. To install Postman, go to the apps page available on <https://www.getpostman.com> and click Download for macOS / Windows / Linux depending on your platform.



- **macOS installation**

Once you've downloaded and unzipped the app, double click on Postman. You will be prompted to move file into the "Applications" folder. Click "Move to Applications Folder" to ensure future updates can be installed correctly. The app will open after the prompt.



- **Windows installation**

- Download the setup file
- Run the installer

- **Linux installation**

For installation on Linux, perform the following steps:

- First download and unzip the file
- And then create a desktop file by name `Postman.desktop`. Create the `Postman.desktop` file in the following location:

```
~/.local/share/applications/Postman.desktop
```

Use the content below in the above file:

```
[Desktop Entry]
Encoding=UTF-8
Name=Postman
Exec=YOUR_INSTALL_DIR/Postman/app/Postman %U
Icon=YOUR_INSTALL_DIR/Postman/app/resources/app/assets/icon.png
Terminal=false
Type=Application
Categories=Development;
```

Once the `Postman.desktop` file is created, the Postman app can be opened using application launchers. You can check your desktop and double-click the Postman icon.

Note:

1. Avoid starting postman using `sudo` command, this will create permission issues on the files created by postman.
2. Make sure you have read/write permission for `~/.config` folder where Postman stores the information
3. If you are an Ubuntu 18 user, you must additionally install `libgconf-2-4` package to ensure a smooth Postman run. This package, shipped by default until Ubuntu version 18, has been dropped, and is mandatory for Postman to run. Use the following command to install `libgconf-2-4`:

```
apt-get install libgconf-2-4
```

1.2. 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 in your group.

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

1.3. 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 1.2 using this online tool.

Compare the results with the results of Practice 1.2 and discuss it in your group.

1.4. More GET Requests and Responses

Repeat Practices 1.2 and 1.3 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)

1.5. POST Request and Response

Repeat Practices 1.2 and 1.3 with the **POST** request for the following URLs, using the given data below:

- <https://reqres.in/api/users> (CREATE)

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

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

```
{
  "email": "sydney@fife",
  "password": "pistol"
}
```

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

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

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

```
{
  "email": "peter@klaven",
  "password": "cityslicka"
}
```

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

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

EXERCISES

1. To be added.

2. To be added.