What is HTTP? (HyperText Transfer Protocol)

HTTP is what's used whenever you view a website, developed by Tim Berners-Lee and his team between 1989-1991. HTTP is the set of rules used for communicating with web servers for the transmitting of webpage data, whether that is HTML, Images, Videos, etc.

What is HTTPS? (HyperText Transfer Protocol Secure)

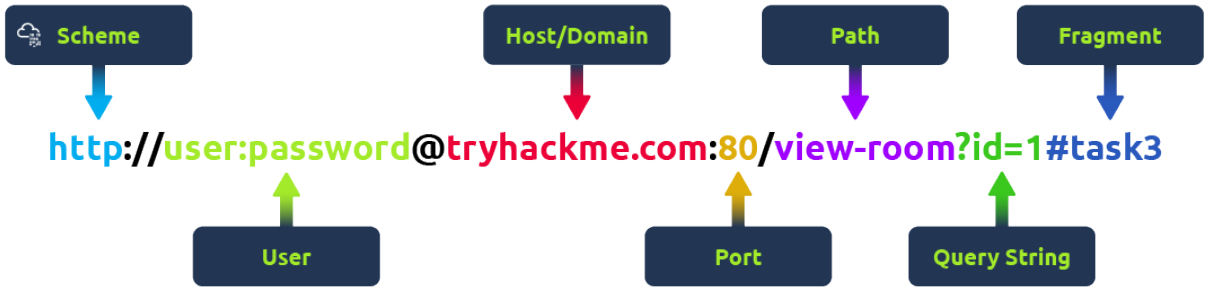
HTTPS is the secure version of HTTP. HTTPS data is encrypted so it not only stops people from seeing the data you are receiving and sending, but it also gives you assurances that you're talking to the correct web server and not something impersonating it.

Requests & Responses

When we access a website, your browser will need to make requests to a web server for assets such as HTML, Images, and download the responses. Before that, you need to tell the browser specifically how and where to access these resources, this is where URLs will help.

What is a URL? (Uniform Resource Locator)

A URL is predominantly an instruction on how to access a resource on the internet. The below image shows what a URL looks like with all of its features (it does not use all features in every request).



Scheme: This instructs on what protocol to use for accessing the resource such as HTTP, HTTPS, FTP (File Transfer Protocol).

User: Some services require authentication to log in, you can put a username and password into the URL to log in.

Host: The domain name or IP address of the server you wish to access.

Port: The Port that you are going to connect to, usually 80 for HTTP and 443 for HTTPS, but this can be hosted on any port between 1 - 65535.

Path: The file name or location of the resource you are trying to access.

Query String: Extra bits of information that can be sent to the requested path. For example, /blog?**id=1** would tell the blog path that you wish to receive the blog article with the id of 1.

Fragment: This is a reference to a location on the actual page requested. This is commonly used for pages with long content and can have a certain part of the page directly linked to it, so it is viewable to the user as soon as they access the page.

Making a Request

It's possible to make a request to a web server with just one line **GET / HTTP/1.1**

But for a much richer web experience, you'll need to send other data as well. This other data is sent in what is called headers, where headers contain extra information to give to the web server you're communicating with, but we'll go more into this in the Header task.

Example Request:

```
GET / HTTP/1.1

Host: tryhackme.com
User-Agent: Mozilla/5.0 Firefox/87.0
Referer: https://tryhackme.com/
```

Line 1: This request is sending the GET method, request the home page with / and telling the web server we are using HTTP protocol version 1.1.

Line 2: We tell the web server we want the website tryhackme.com

Line 3: We tell the web server we are using the Firefox version 87 Browser

Line 4: We are telling the web server that the web page that referred us to this one is https://tryhackme.com

Line 5: HTTP requests always end with a blank line to inform the web server that the request has finished.

Example Response:

```
HTTP/1.1 200 OK

Server: nginx/1.15.8
Date: Fri, 09 Apr 2021 13:34:03 GMT
Content-Type: text/html
Content-Length: 98


<html>
<head>
   <title>TryHackMe</title>
</head>
<body>
   Welcome To TryHackMe.com
</body>
</html>
```

Line 1: HTTP 1.1 is the version of the HTTP protocol the server is using and then followed by the HTTP Status Code in this case "200 OK" which tells us the request has completed successfully.

Line 2: This tells us the web server software and version number.

Line 3: The current date, time and timezone of the web server.

Line 4: The Content-Type header tells the client what sort of information is going to be sent, such as HTML, images, videos, pdf, XML.

Line 5: Content-Length tells the client how long the response is, this way we can confirm no data is missing.

Line 6: HTTP response contains a blank line to confirm the end of the HTTP response.

Lines 7-14: The information that has been requested, in this instance the homepage.

HTTP Methodes

HTTP methods are a way for the client to show their intended action when making an HTTP request. There are a lot of HTTP methods but we'll cover the most common ones, although mostly you'll deal with the GET and POST method.

GET Request

This is used for getting information from a web server.

POST Request

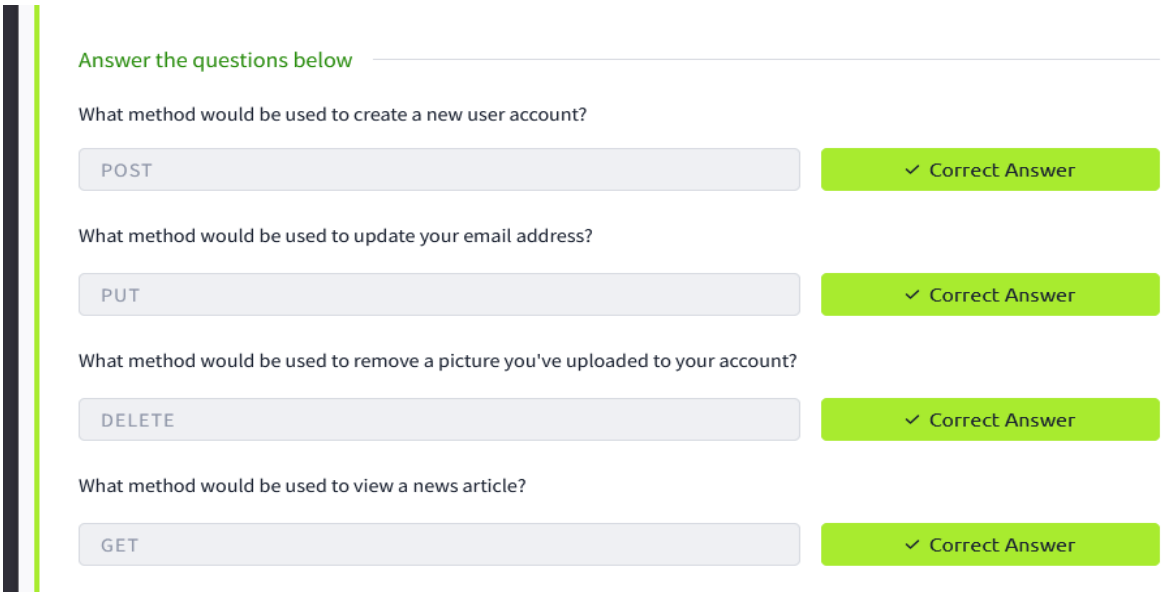
This is used for submitting data to the web server and potentially creating new records

PUT Request

This is used for submitting data to a web server to update information

DELETE Request

This is used for deleting information/records from a web server.



HTTP Status Codes

In the previous task, you learnt that when a HTTP server responds, the first line always contains a status code informing the client of the outcome of their request and also potentially how to handle it. These status codes can be broken down into 5 different ranges:

100-199 - Information Response	These are sent to tell the client the first part of their request has been accepted and they should continue sending the rest of their request. These codes are no longer very common.
200-299 - Success	This range of status codes is used to tell the client their request was successful.
300-399 - Redirection	These are used to redirect the client's request to another resource. This can be either to a different webpage or a different website altogether.

400-499 - Client Errors	Used to inform the client that there was an error with their request.
500-599 - Server Errors	This is reserved for errors happening on the server-side and usually indicate quite a major problem with the server handling the request.

Headers

Headers are additional bits of data you can send to the web server when making requests. Although no headers are strictly required when making a HTTP request, you'll find it difficult to view a website properly.

→Header sagen dem Server, wer ich bin, was ich will und wie er antworten soll. (und andersrum)

Common Request Headers

These are headers that are sent from the client (usually your browser) to the server.

Host: Some web servers host multiple websites so by providing the host headers you can tell it which one you require, otherwise you'll just receive the default website for the server.

User-Agent: This is your browser software and version number, telling the web server your browser software helps it format the website properly for your browser and also some elements of HTML, JavaScript and CSS are only available in certain browsers.

Content-Length: When sending data to a web server such as in a form, the content length tells the web server how much data to expect in the web request. This way the server can ensure it isn't missing any data.

Accept-Encoding: Tells the web server what types of compression methods the browser supports so the data can be made smaller for transmitting over the internet.

Cookie: Data sent to the server to help remember your information (see cookies task for more information).

Common Response Headers

These are the headers that are returned to the client from the server after a request.

Set-Cookie: Information to store which gets sent back to the web server on each request (see cookies task for more information).

Cache-Control: How long to store the content of the response in the browser's cache before it requests it again.

Content-Type: This tells the client what type of data is being returned, i.e., HTML, CSS, JavaScript, Images, PDF, Video, etc. Using the content-type header the browser then knows how to process the data.

Content-Encoding: What method has been used to compress the data to make it smaller when sending it over the internet.

Cookies

You've probably heard of cookies before, they're just a small piece of data that is stored on your computer. Cookies are saved when you receive a "Set-Cookie" header from a web server. Then every further request you make, you'll send the cookie data back to the web server. Because HTTP is stateless (doesn't keep track of your previous requests), cookies

can be used to remind the web server who you are, some personal settings for the website or whether you've been to the website before.

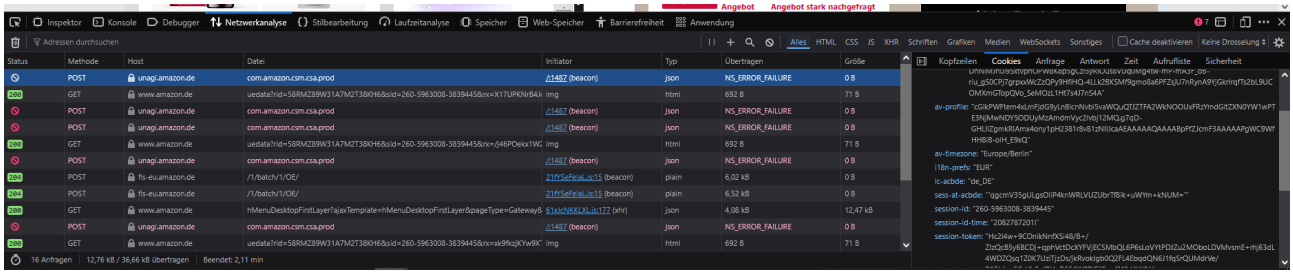
Cookies can be used for many purposes but are most commonly used for website authentication. The cookie value won't usually be a clear-text string where you can see the password, but a token (unique secret code that isn't easily humanly guessable).

Viewing Your Cookies

You can easily view what cookies your browser is sending to a website by using the developer tools, in your browser. If you're not sure how to get to the developer tools in your browser, click on the "View Site" button at the top of this task for a how-to guide.

Once you have developer tools open, click on the "Network" tab. This tab will show you a list of all the resources your browser has requested. You can click on each one to receive a detailed breakdown of the request and response. If your browser sent a cookie, you will see these on the "Cookies" tab of the request.

Meine Cookies auf Amazon:



Praxis: