HTTP In Detail

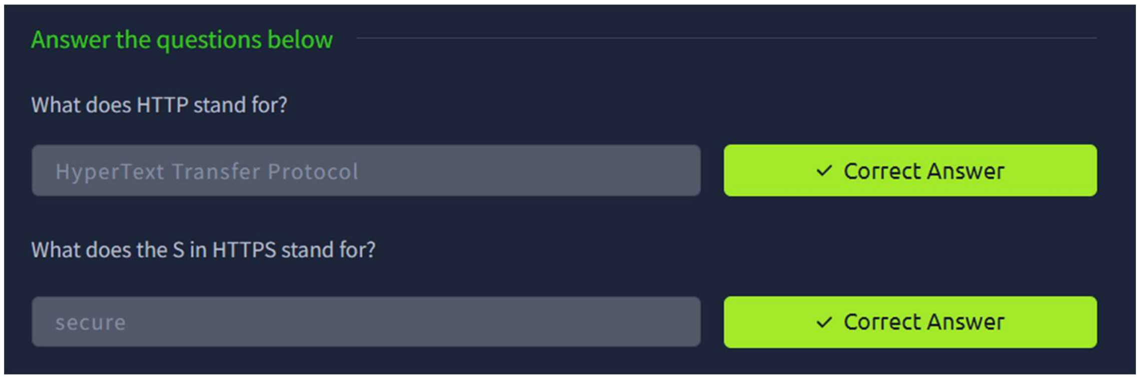
Task 1:- What is HTTP(S)

What is HTTP? (HyperText Transfer Protocol)

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.



Answer the questions below

What does HTTP stand for?

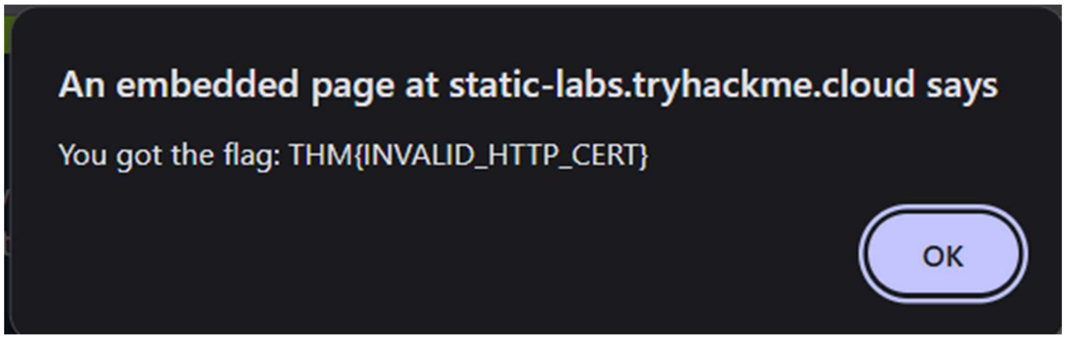
HyperText Transfer Protocol

✓ Correct Answer

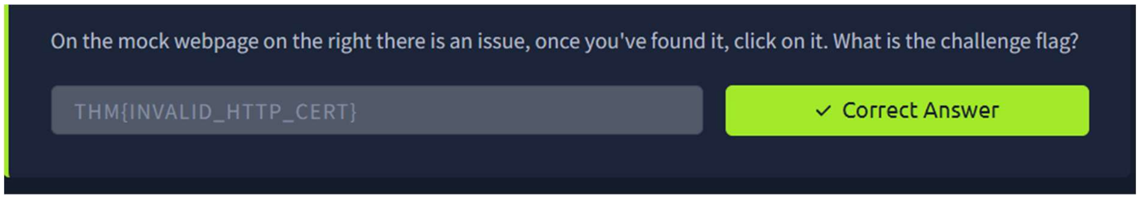
What does the S in HTTPS stand for?

secure

✓ Correct Answer


An embedded page at static-labs.tryhackme.cloud says

You got the flag: THM{INVALID_HTTP_CERT}

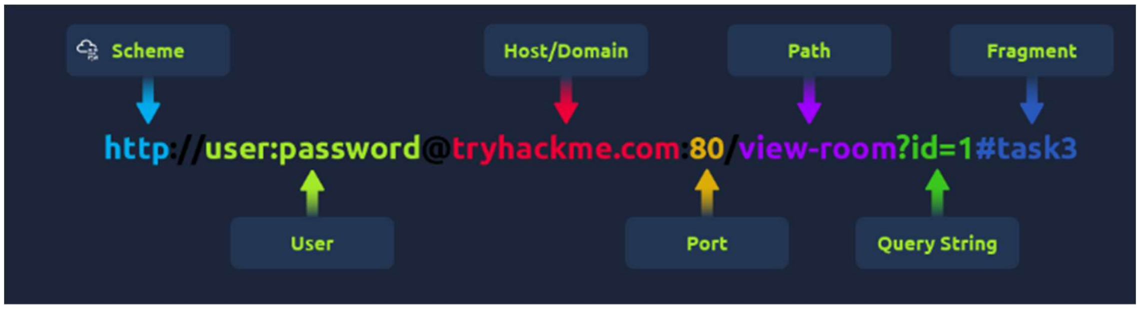

On the mock webpage on the right there is an issue, once you've found it, click on it. What is the challenge flag?

THM{INVALID_HTTP_CERT}

Task 2:- Request's and Respsonses

What is a URL? (Uniform Resource Locator)

If you've used the internet, you've used a URL before. 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:- It is used to define the protocol like http , https or ftp etc

User:- Some services require authentication to access.

Host/Domain:- It is the domain name or IP address in which website we want to access.

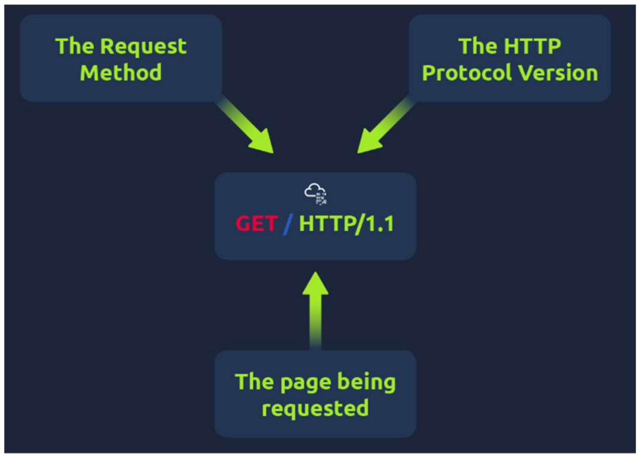
Port:-It tells which protocol we are using like 80 for http , 443 for https etc

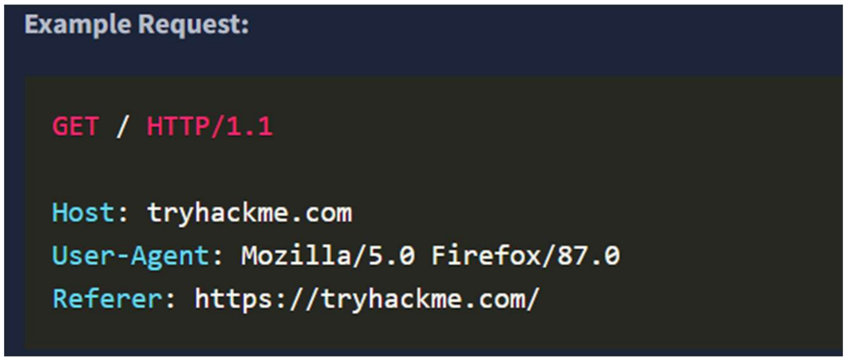
Path:-The file name or location of the resource you are trying to access.

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




Example Request:

```
GET / HTTP/1.1

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

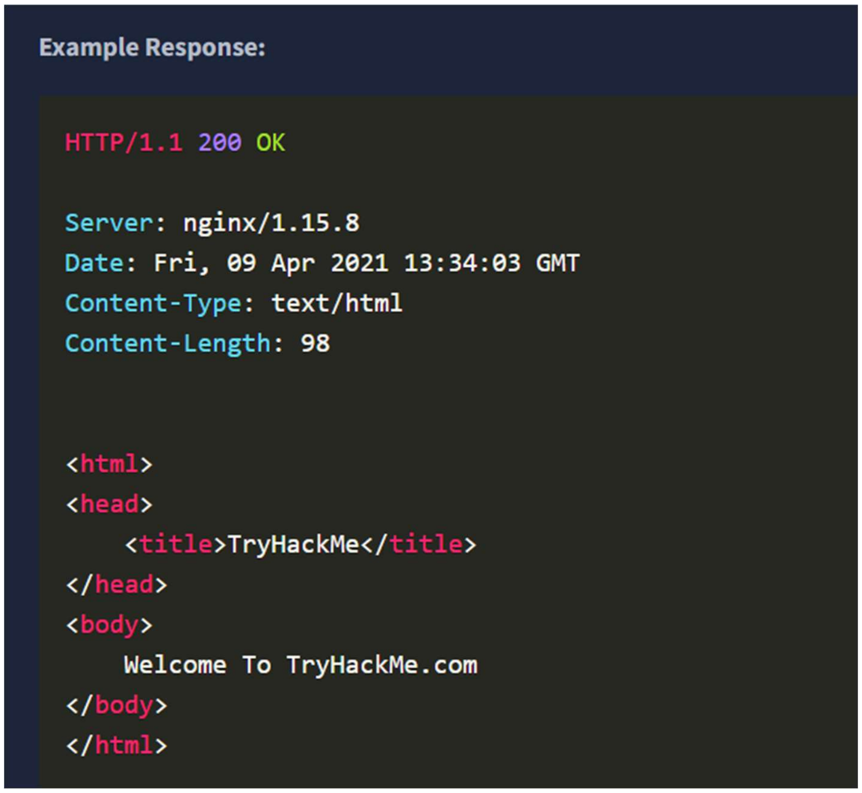
Line 1: This request is sending the GET method (more on this in the HTTP Methods task), request the home page with / and telling the web server we are using HTTP protocol version 1.1.

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.

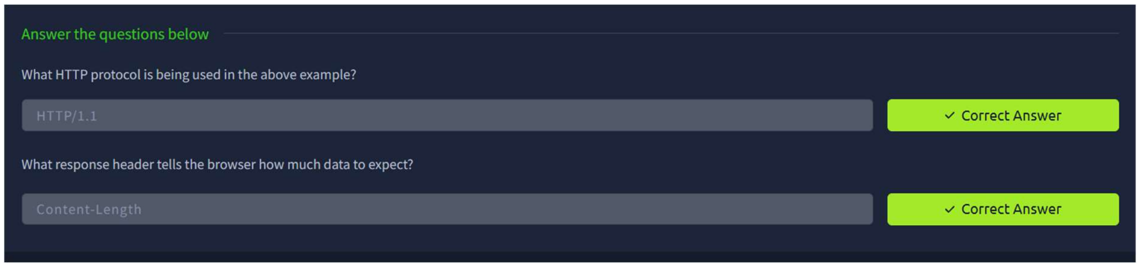
Line 3: The current date, time and timezone of the web server.

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.


Answer the questions below

What HTTP protocol is being used in the above example?

HTTP/1.1

What response header tells the browser how much data to expect?

Content-Length

Task 3:- HTTP Methods

HTTP methods are a way for the client to show their intended action when making an HTTP request.

Few methods are:-

GET Request

This is used for getting information from a web server.

POST Request

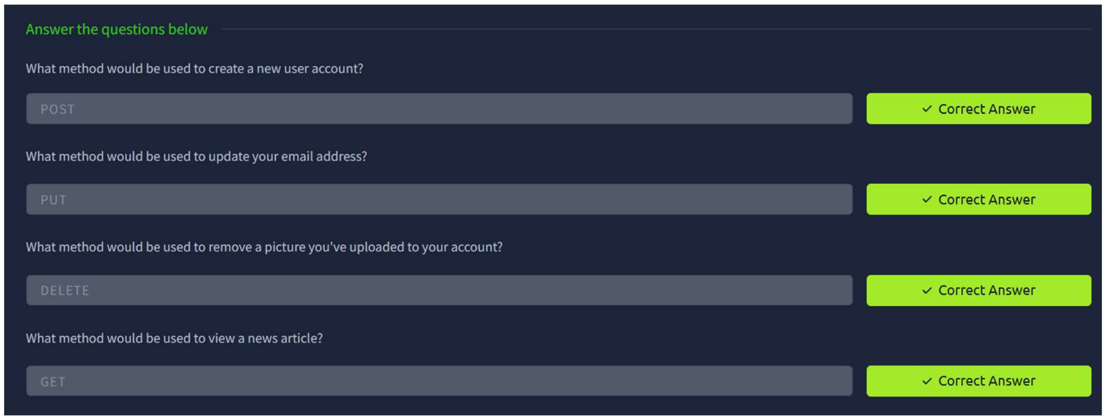
This is used for submitting data to the web server and potentially creating new records

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.



Answer the questions below

What method would be used to create a new user account?

POST ✓ Correct Answer

What method would be used to update your email address?

PUT ✓ Correct Answer

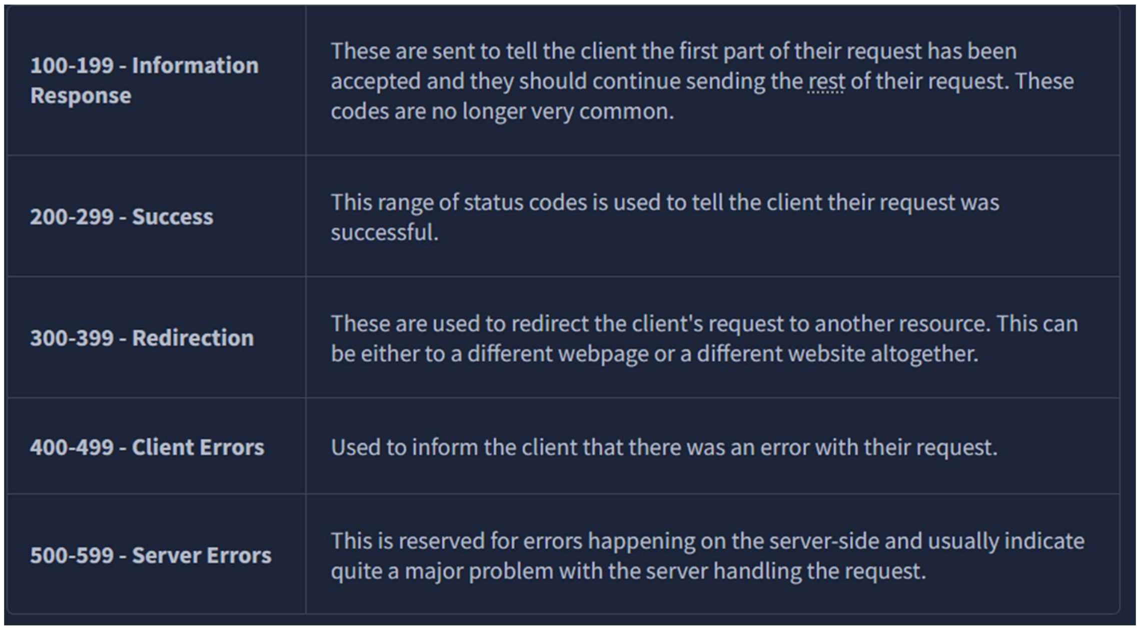
What method would be used to remove a picture you've uploaded to your account?

DELETE ✓ Correct Answer

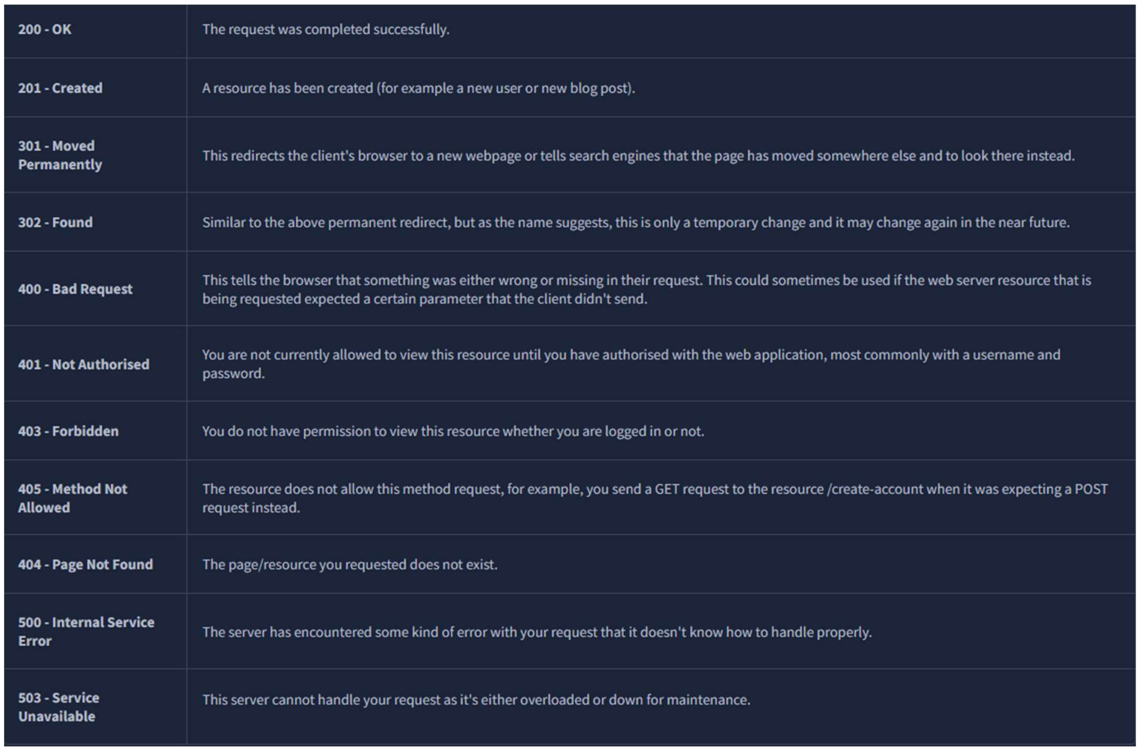
What method would be used to view a news article?

GET ✓ Correct Answer

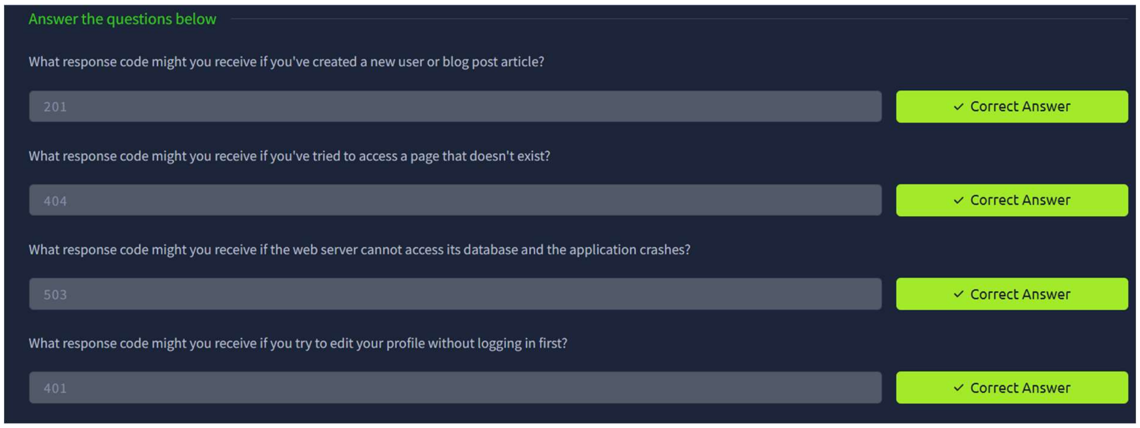
Task 4:- HTTP Status Codes



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.



200 - OK	The request was completed successfully.
201 - Created	A resource has been created (for example a new user or new blog post).
301 - Moved Permanently	This redirects the client's browser to a new webpage or tells search engines that the page has moved somewhere else and to look there instead.
302 - Found	Similar to the above permanent redirect, but as the name suggests, this is only a temporary change and it may change again in the near future.
400 - Bad Request	This tells the browser that something was either wrong or missing in their request. This could sometimes be used if the web server resource that is being requested expected a certain parameter that the client didn't send.
401 - Not Authorised	You are not currently allowed to view this resource until you have authorised with the web application, most commonly with a username and password.
403 - Forbidden	You do not have permission to view this resource whether you are logged in or not.
405 - Method Not Allowed	The resource does not allow this method request, for example, you send a GET request to the resource /create-account when it was expecting a POST request instead.
404 - Page Not Found	The page/resource you requested does not exist.
500 - Internal Service Error	The server has encountered some kind of error with your request that it doesn't know how to handle properly.
503 - Service Unavailable	This server cannot handle your request as it's either overloaded or down for maintenance.



Answer the questions below

What response code might you receive if you've created a new user or blog post article?

201

What response code might you receive if you've tried to access a page that doesn't exist?

404

What response code might you receive if the web server cannot access its database and the application crashes?

503

What response code might you receive if you try to edit your profile without logging in first?

401

Task 5:- Headers

Headers are additional bits of data you can send to the web server when making requests.

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.

Common Response Headers

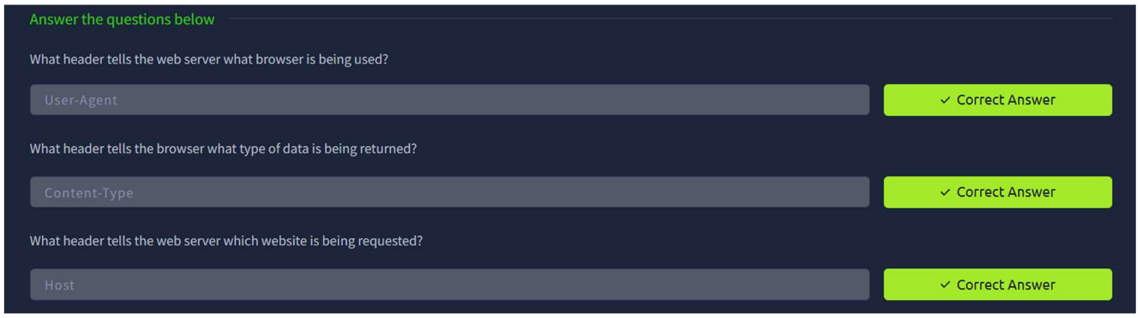
These are the headers that are returned to the client from the server after a request.

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.



Answer the questions below

What header tells the web server what browser is being used?

User-Agent

What header tells the browser what type of data is being returned?

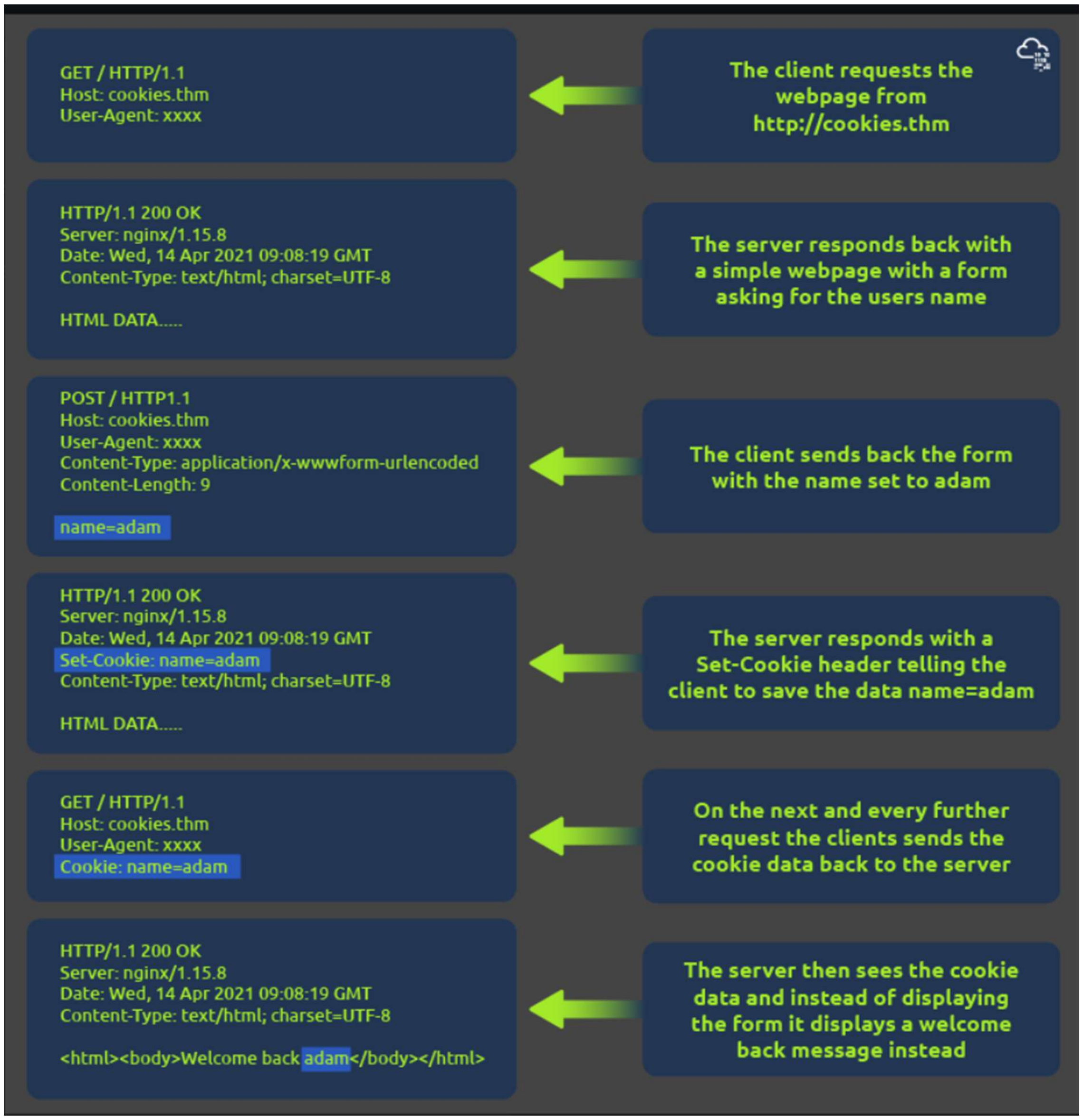
Content-Type

What header tells the web server which website is being requested?

Host

Task 6:-Cookies

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.



```
GET / HTTP/1.1
Host: cookies.thm
User-Agent: xxxx
```

The client requests the webpage from http://cookies.thm

```
HTTP/1.1 200 OK
Server: nginx/1.15.8
Date: Wed, 14 Apr 2021 09:08:19 GMT
Content-Type: text/html; charset=UTF-8

HTML DATA.....
```

The server responds back with a simple webpage with a form asking for the users name

```
POST / HTTP1.1
Host: cookies.thm
User-Agent: xxxx
Content-Type: application/x-wwwform-urlencoded
Content-Length: 9

name=adam
```

The client sends back the form with the name set to adam

```
HTTP/1.1 200 OK
Server: nginx/1.15.8
Date: Wed, 14 Apr 2021 09:08:19 GMT
Set-Cookie: name=adam
Content-Type: text/html; charset=UTF-8

HTML DATA.....
```

The server responds with a Set-Cookie header telling the client to save the data name=adam

```
GET / HTTP/1.1
Host: cookies.thm
User-Agent: xxxx
Cookie: name=adam
```

On the next and every further request the clients sends the cookie data back to the server

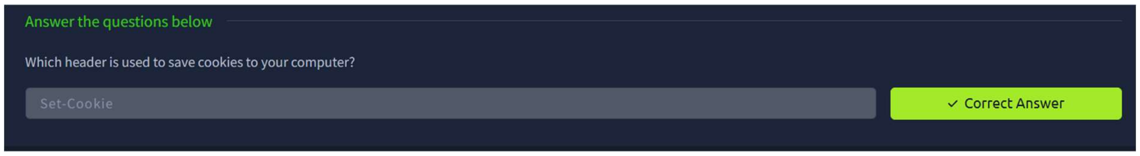
```
HTTP/1.1 200 OK
Server: nginx/1.15.8
Date: Wed, 14 Apr 2021 09:08:19 GMT
Content-Type: text/html; charset=UTF-8

<html><body>Welcome back adam</body></html>
```

The server then sees the cookie data and instead of displaying the form it displays a welcome back message instead

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.



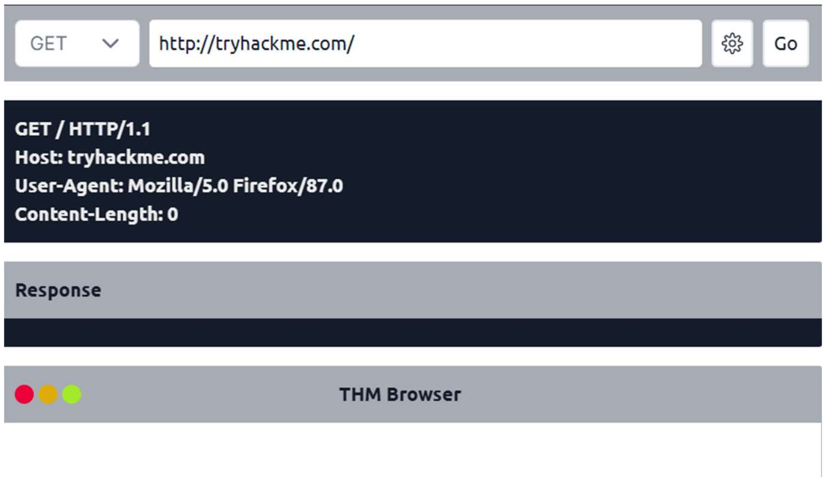
Answer the questions below

Which header is used to save cookies to your computer?

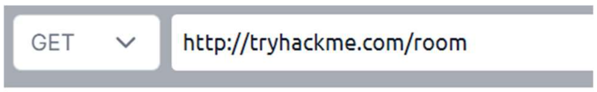
Set-Cookie

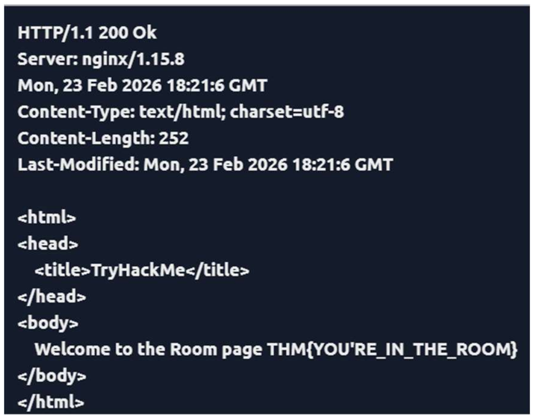
Task 7:- Making Requests

1)Make a GET request to /room page



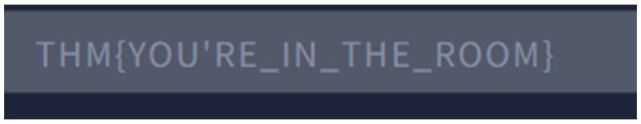
```
GET / HTTP/1.1
Host: tryhackme.com
User-Agent: Mozilla/5.0 Firefox/87.0
Content-Length: 0
```





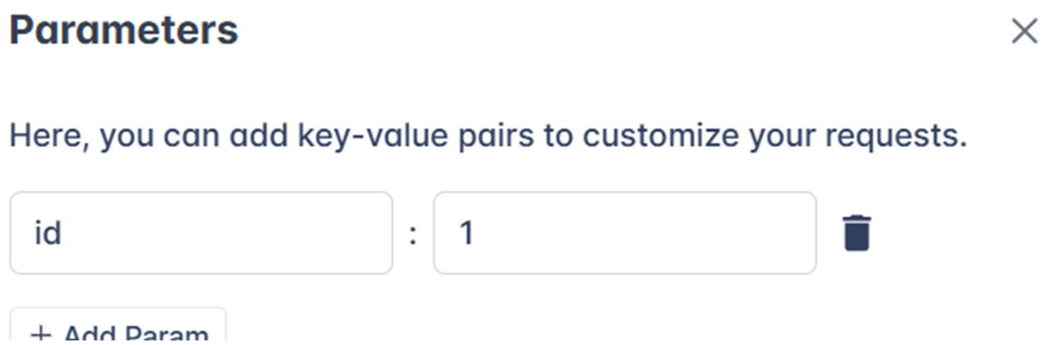
```
HTTP/1.1 200 Ok
Server: nginx/1.15.8
Mon, 23 Feb 2026 18:21:6 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 252
Last-Modified: Mon, 23 Feb 2026 18:21:6 GMT

<html>
<head>
  <title>TryHackMe</title>
</head>
<body>
  Welcome to the Room page THM{YOU'RE_IN_THE_ROOM}
</body>
</html>
```



THM{YOU'RE_IN_THE_ROOM}

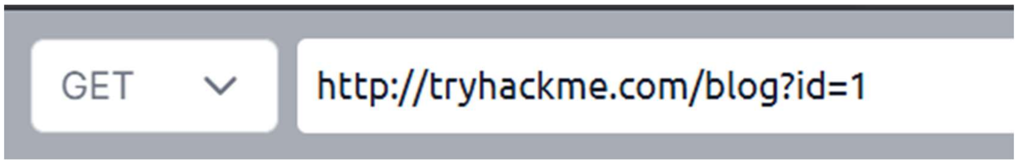
2)Make a GET request to /blog page and set the id parameter to 1
Note: Use the gear button on the right to manage URI parameters

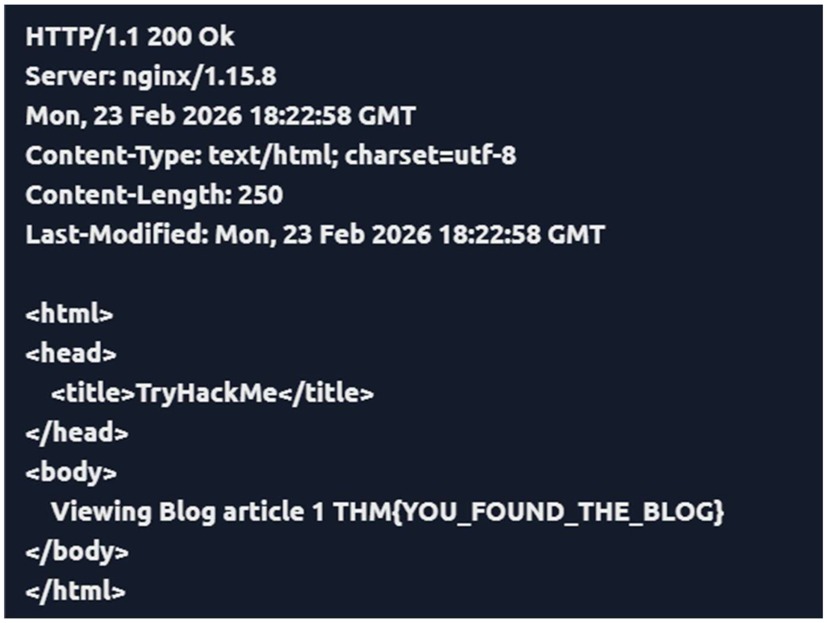


Parameters

Here, you can add key-value pairs to customize your requests.

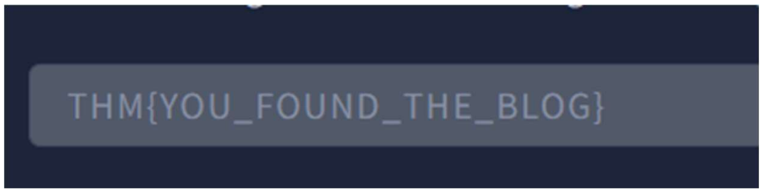
id : 1





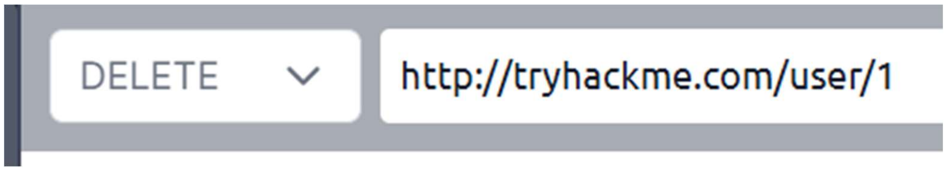
```
HTTP/1.1 200 Ok
Server: nginx/1.15.8
Mon, 23 Feb 2026 18:22:58 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 250
Last-Modified: Mon, 23 Feb 2026 18:22:58 GMT

<html>
<head>
  <title>TryHackMe</title>
</head>
<body>
  Viewing Blog article 1 THM{YOU_FOUND_THE_BLOG}
</body>
</html>
```

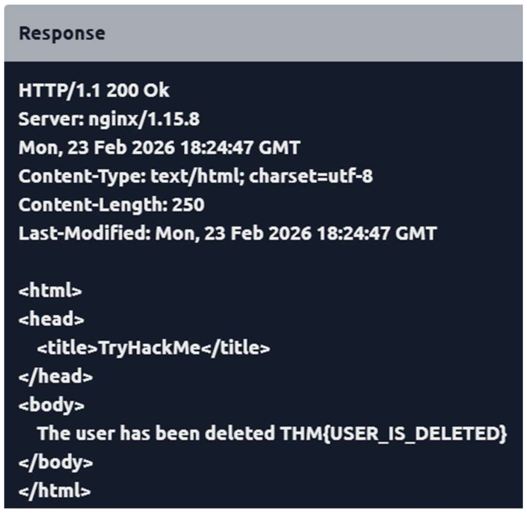


THM{YOU_FOUND_THE_BLOG}

3)Make a DELETE request to /user/1 page



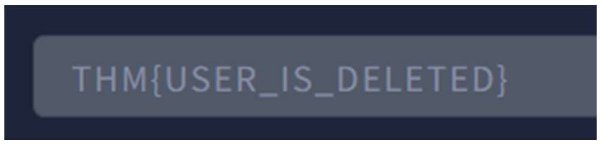
DELETE http://tryhackme.com/user/1



Response

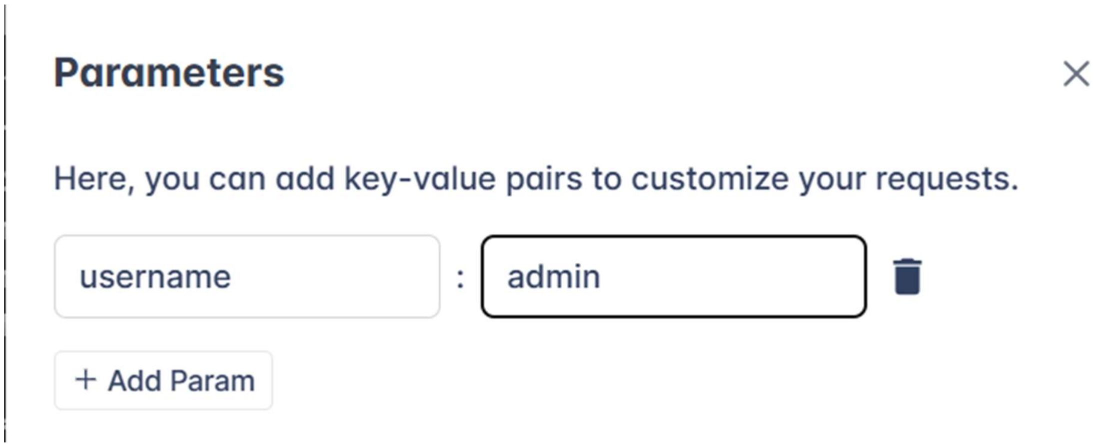
```
HTTP/1.1 200 Ok
Server: nginx/1.15.8
Mon, 23 Feb 2026 18:24:47 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 250
Last-Modified: Mon, 23 Feb 2026 18:24:47 GMT

<html>
<head>
  <title>TryHackMe</title>
</head>
<body>
  The user has been deleted THM{USER_IS_DELETED}
</body>
</html>
```



THM{USER_IS_DELETED}

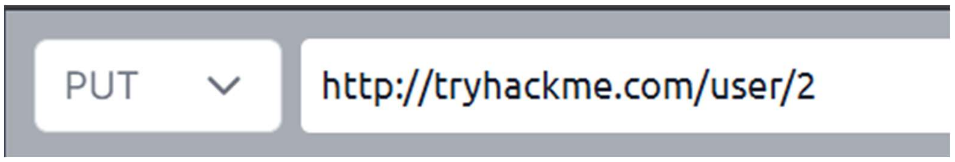
4)Make a PUT request to /user/2 page with the username parameter set to admin
Note: Use the gear button on the right to manage body parameters



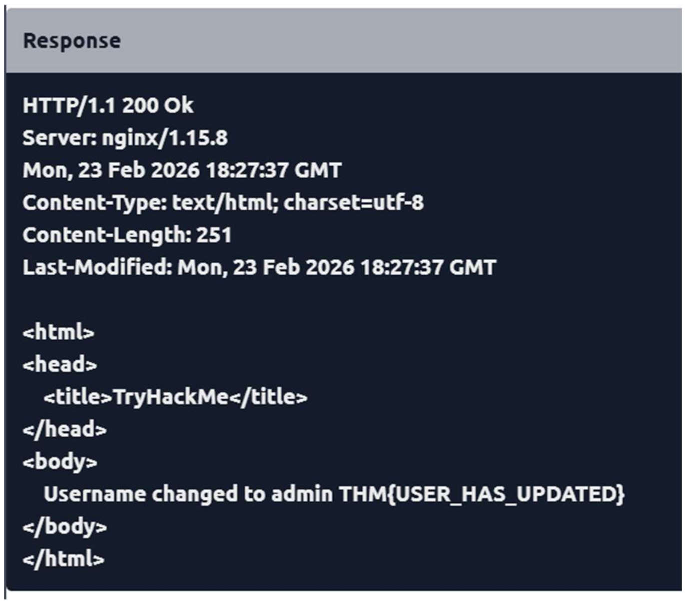
Parameters

Here, you can add key-value pairs to customize your requests.

username : admin



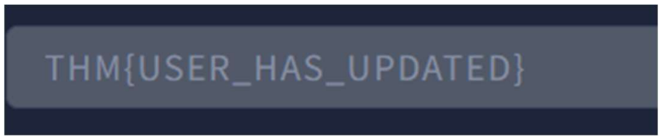
PUT http://tryhackme.com/user/2



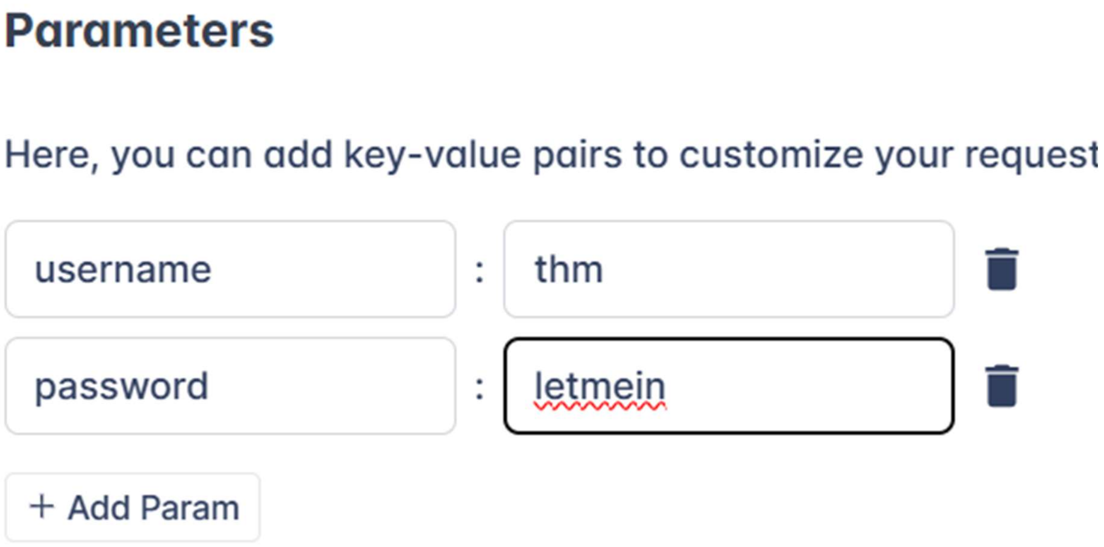
Response

```
HTTP/1.1 200 Ok
Server: nginx/1.15.8
Mon, 23 Feb 2026 18:27:37 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 251
Last-Modified: Mon, 23 Feb 2026 18:27:37 GMT

<html>
<head>
   <title>TryHackMe</title>
</head>
<body>
   Username changed to admin THM{USER_HAS_UPDATED}
</body>
</html>
```

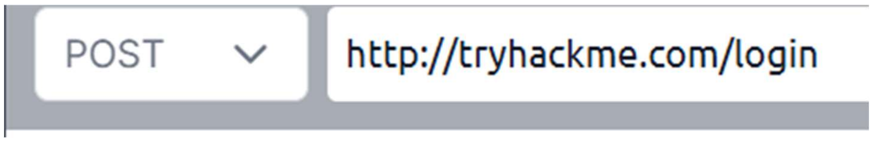

THM{USER_HAS_UPDATED}

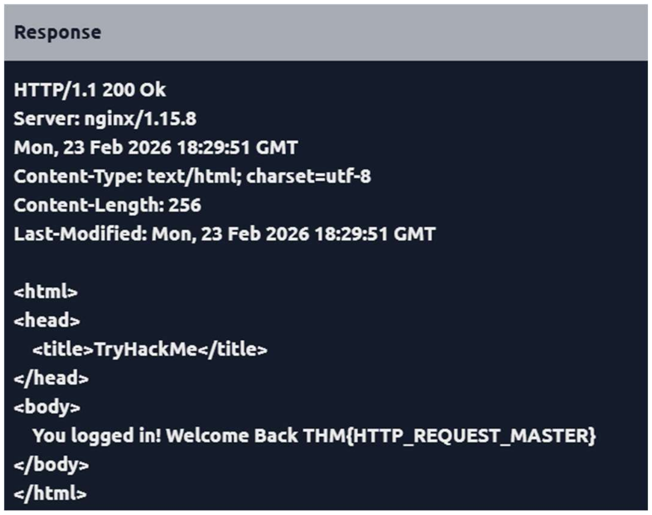
5)Make a POST request to /login page with the username of thm and a password of letmein
Note: Use the gear button on the right to manage body parameters


Parameters

Here, you can add key-value pairs to customize your request

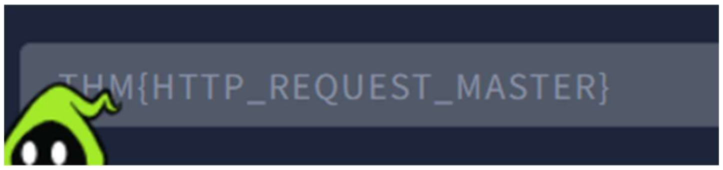
username : thm
password : letmein


POST http://tryhackme.com/login


Response

```
HTTP/1.1 200 Ok
Server: nginx/1.15.8
Mon, 23 Feb 2026 18:29:51 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 256
Last-Modified: Mon, 23 Feb 2026 18:29:51 GMT

<html>
<head>
  <title>TryHackMe</title>
</head>
<body>
  You logged in! Welcome Back THM{HTTP_REQUEST_MASTER}
</body>
</html>
```


THM{HTTP_REQUEST_MASTER}