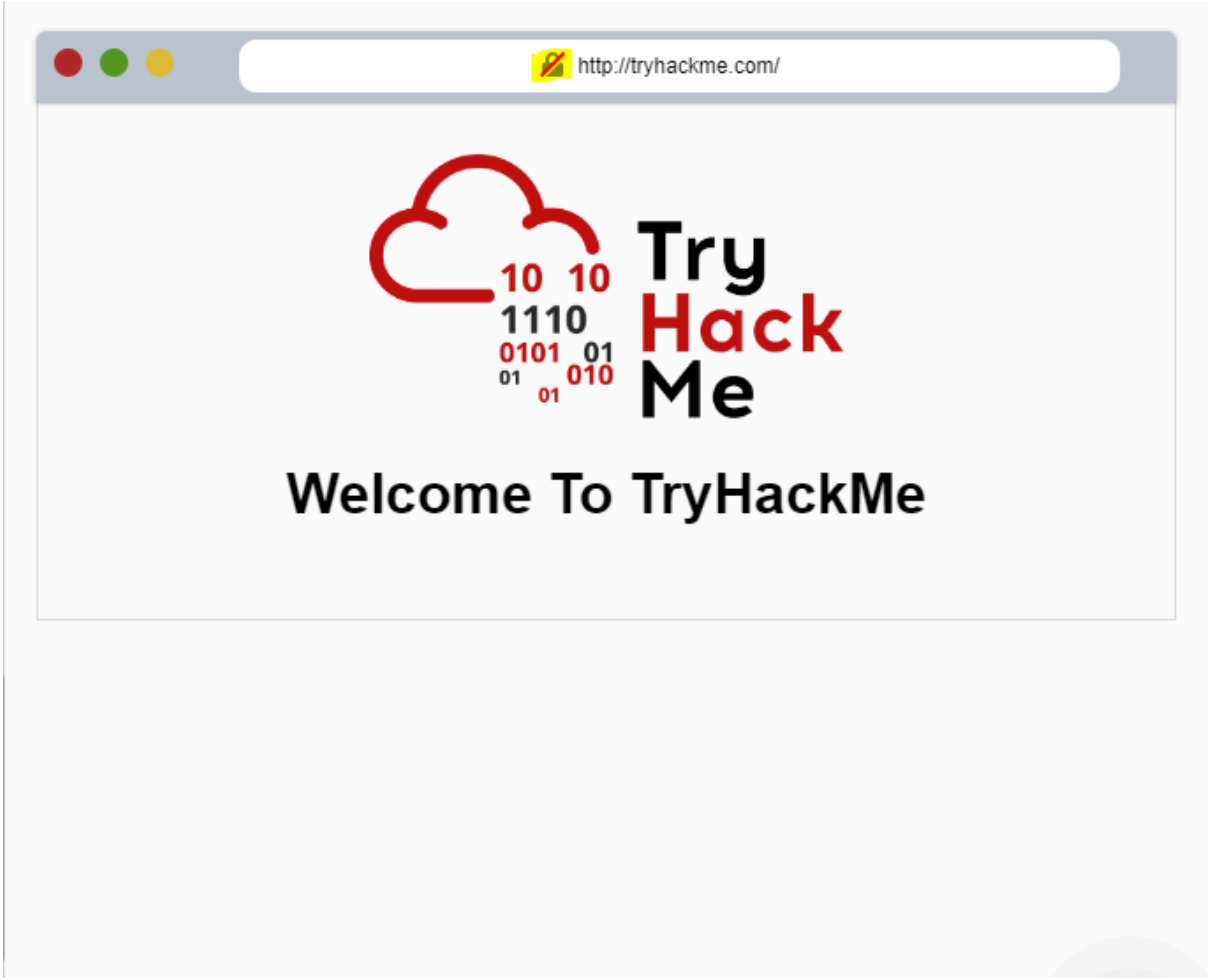
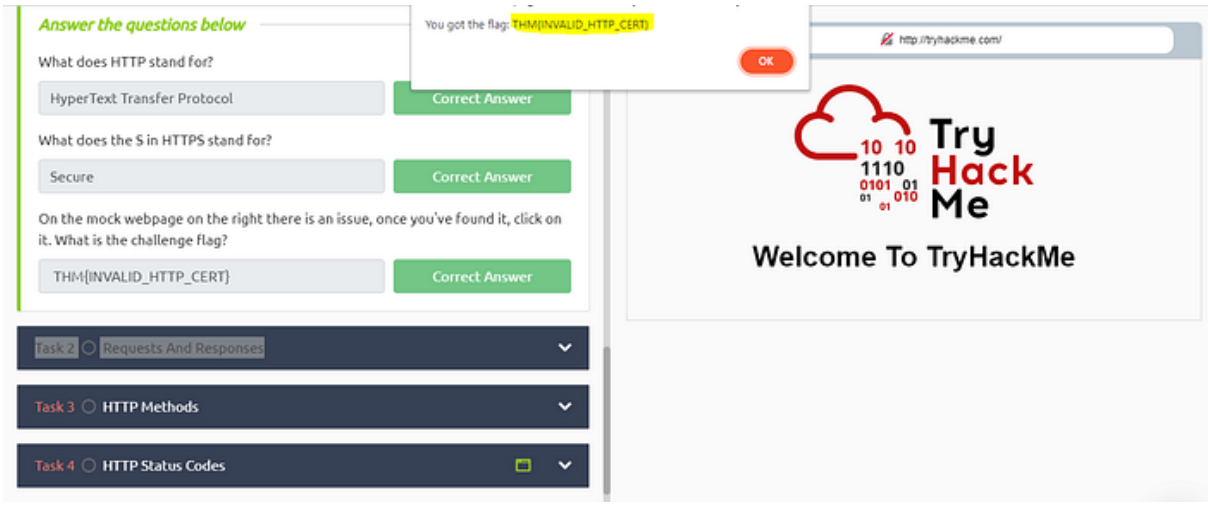
- **<u>Task 1</u> : What is HTTP(S)?:**
- **What does HTTP stand for?**
- **<u>Answer</u>:** HyperText Transfer Protocol
- **What does the S in HTTPS stand for?**
- **<u>Answer</u>:** Secure
- **On the mock webpage on the right, there is an issue, once you've found it, click on it. What is the challenge flag?**
- **<u>Answer</u>:** THM{INVALID_HTTP_CERT}
- **Note:**
- click on the lock of the URL.

- 

- The URL lock
- Press enter or click to view image in full size
- 

- the flag
- **<u>Task 2</u>: Requests And Responses:**
- **What HTTP protocol is being used in the above example?**

- **<u>Answer</u>:** HTTP/1.1

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>
```

-
- The protocol
- **What response header tells the browser how much data to expect?**

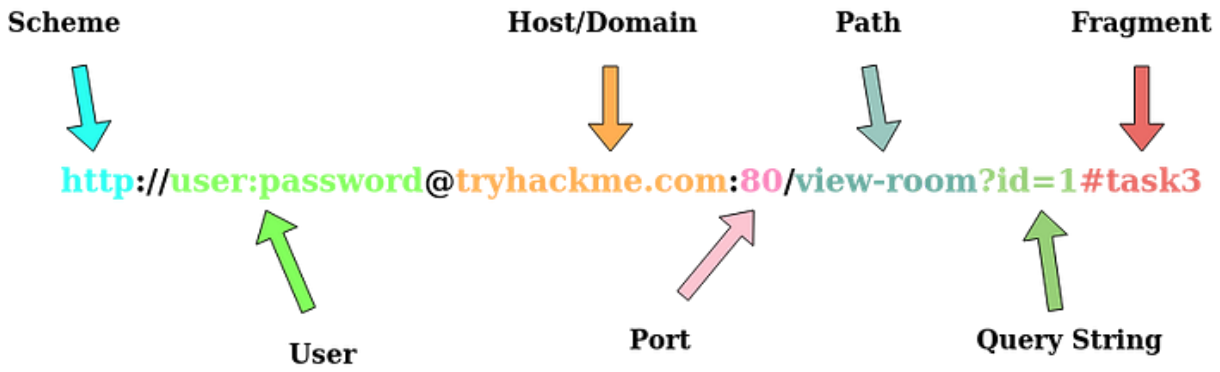
- **<u>Answer</u>:** Content-Length

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>
```

-
- The content-length
- **Note:**
- **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).
- Press enter or click to view image in full size



- **Scheme:** This instructs on what protocol to use for accessing the resource such as HTTP, HTTPS, or 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, is 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:** A fragment is an internal page reference, sometimes called a named anchor. It usually appears at the end of a URL and begins with a hash (#) character followed by an identifier. It refers to a section(any option)within a web page(the web page where the user is staying).
- **Task 3: HTTP Methods:**
- **What method would be used to create a new user account?**
- **Answer:** POST
- **What method would be used to update your email address?**
- **Answer:** PUT
- **What method would be used to remove a picture you've uploaded to your account?**
- **Answer:** DELETE
- **What method would be used to view a news article?**
- **Answer:** GET

- **Note:**
- 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 methods**.
- **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/uploading data/files to a web server from clients side to update information
- **DELETE Request**
- This is used for deleting information/records from a web server.
- **TRACE Request**

- The HTTP TRACE method **performs a message loop-back test along the path to the target resource**, providing a useful debugging mechanism
- **Task 4: HTTP Status Codes:**
- **What response code might you receive if you've created a new user or blog post article?**
- **Answer:** 201
- **What response code might you receive if you've tried to access a page that doesn't exist?**
- **Answer:** 404
- **What response code might you receive if the web server cannot access its database and the application crashes?**
- **Answer:** 503
- **What response code might you receive if you try to edit your profile without logging in first?**
- **Answer:** 401
- **Note:**
- **Different types of responses :**

- **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 Side Errors:** U**sed to inform the client** that there was **an error** with **their request**.
- **500–599 — Server Side Errors:** This is **reserved** for **errors happening on the server side and usually indicates quite a major problem with the server handling the request.**
- **Common HTTP Status Codes:**
- **200 — OK:** The request was **completed successfully.**
- **201 — Created:** A **resource** has been **created** (for example a **new user or new blog post)**.

- **301 — Permanent Redirect:** 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 — Temporary Redirect:** 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 authorized 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.**
- **<u>Task 5</u> :Headers:**
- **What header tells the web server what browser is being used?**
- **<u>Answer</u>:** User-Agent
- **What header tells the browser what type of data is being returned?**
- **<u>Answer</u>:** Content-Type

- **What header tells the web server which website is being requested?**
- **<u>Answer</u>:** Host
- **<u>Note:</u>**
- Headers are **additional bits** of data you can send to the **web server when making requests.**
- Although **no headers are strictly** required when making an HTTP request, you'll find it difficult to view a website properly.
- **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 to the server** 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**, Tell the web server your browser software helps it format the website properly for your browser and also some

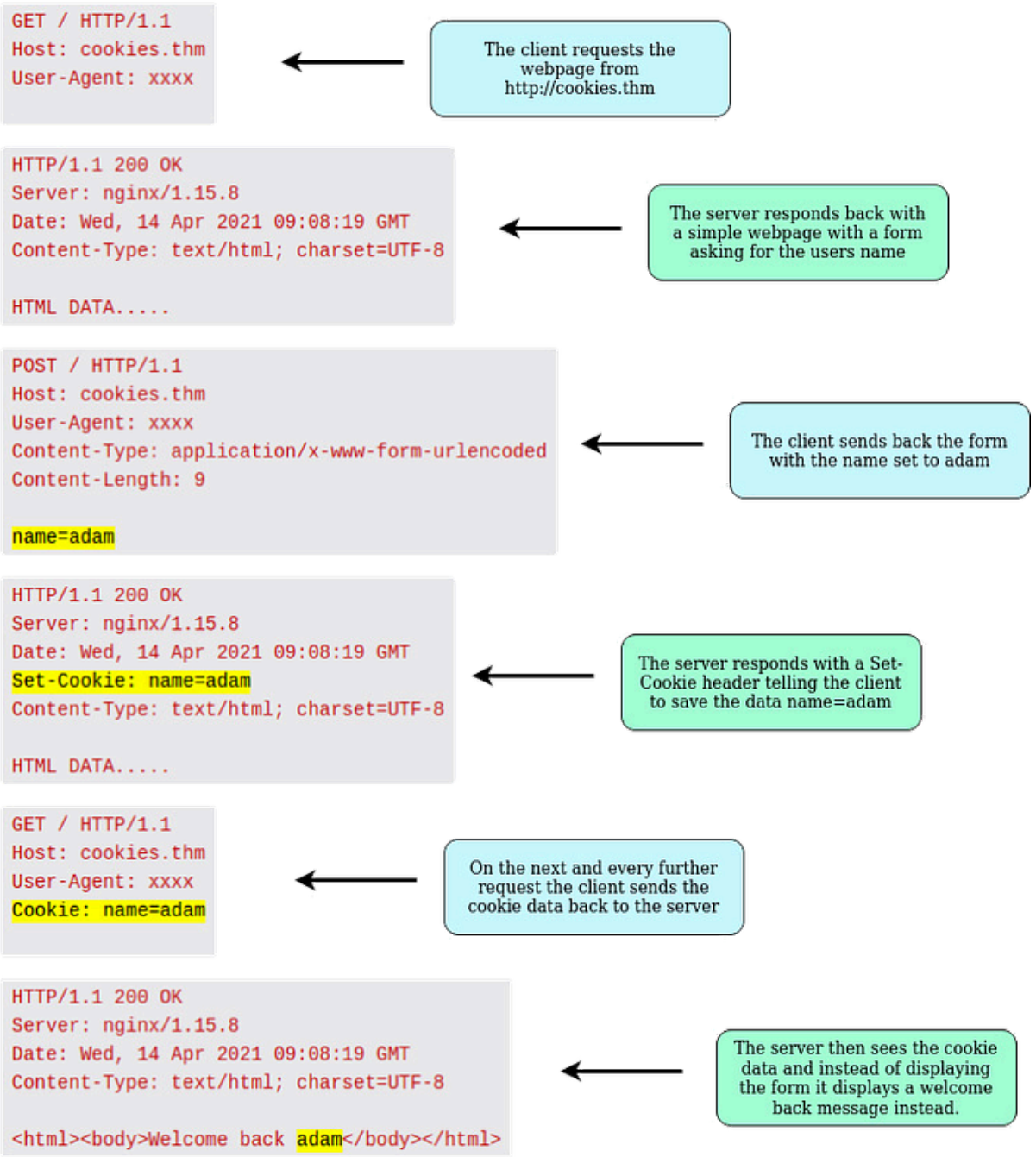
elements of **HTML, JavaScript, and CSS** are only available in certain browsers.

- **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 interne**t.
- **Task 6 : Cookies:**
- **Which header is used to save cookies to your computer?**
- **Answer:** Set-Cookie
- **Note:**
- 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. Let's take a look at this as an example HTTP request:

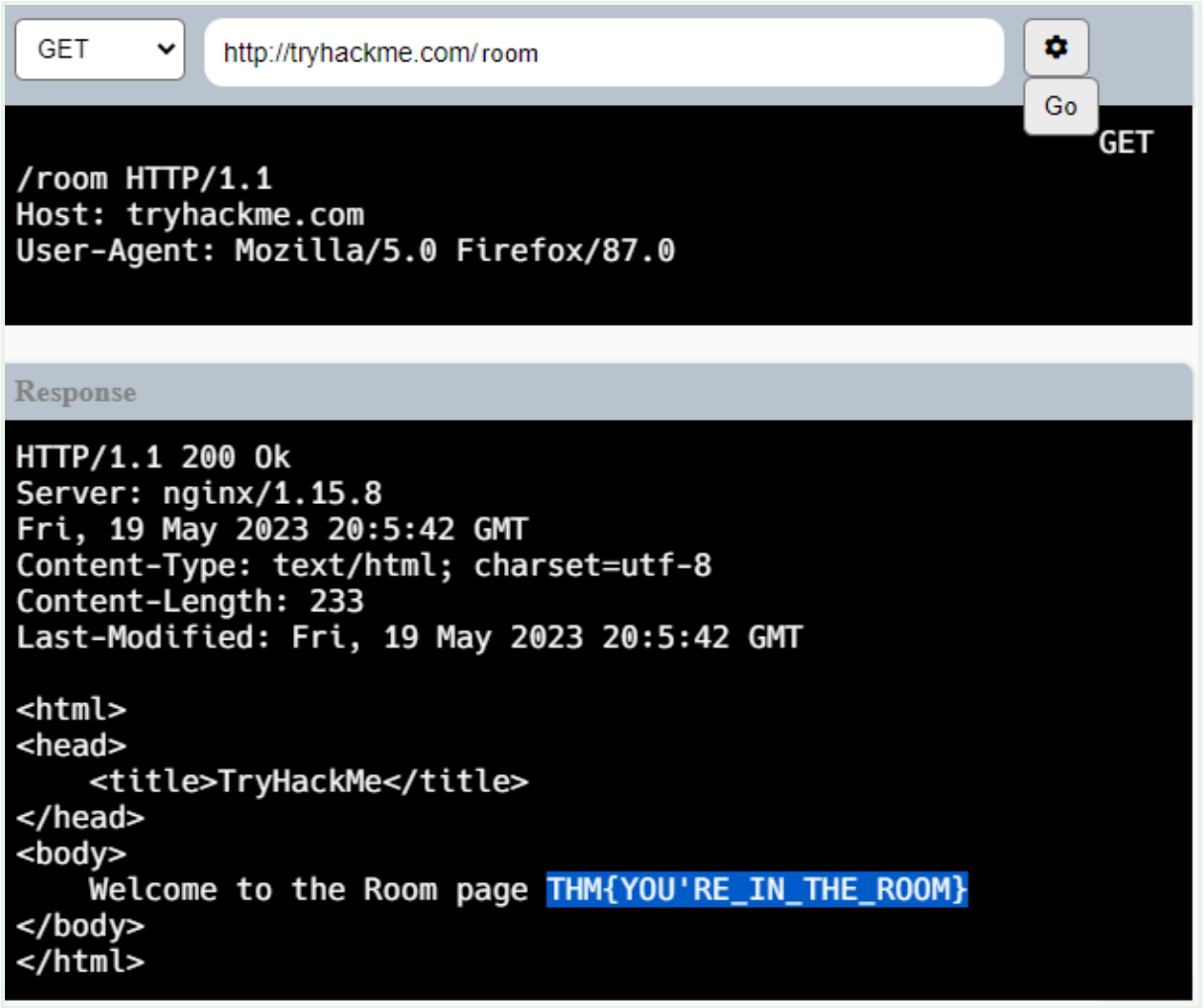
- Press enter or click to view image in full size



-
- 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 s**tring where you can see the password, but a token **(unique secret code that isn't easily humanly guessable).**

- **Task 7: Making Requests:**
- **Make a GET request to /room**

- **Answer:** THM{YOU’RE_IN_THE_ROOM}

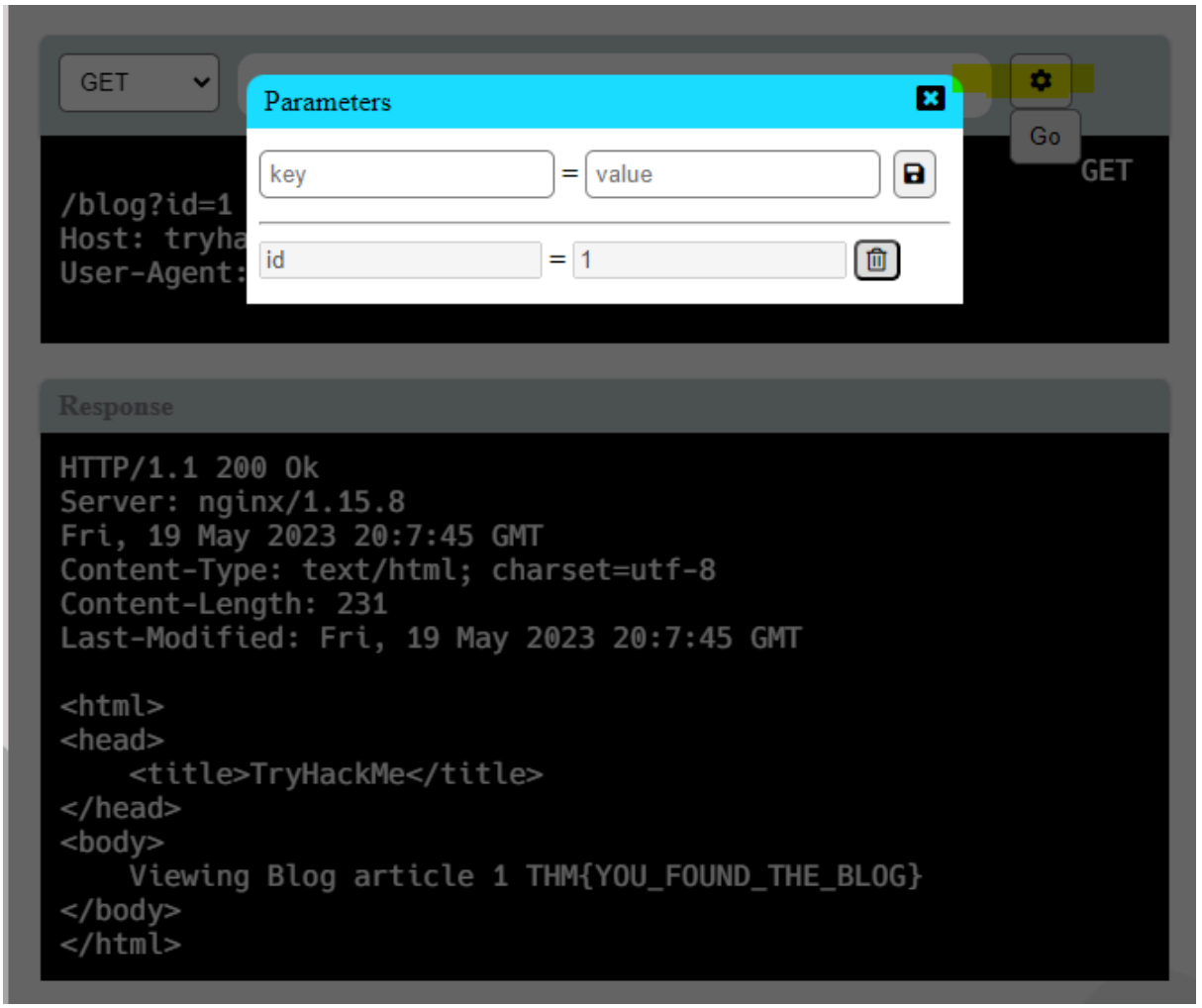


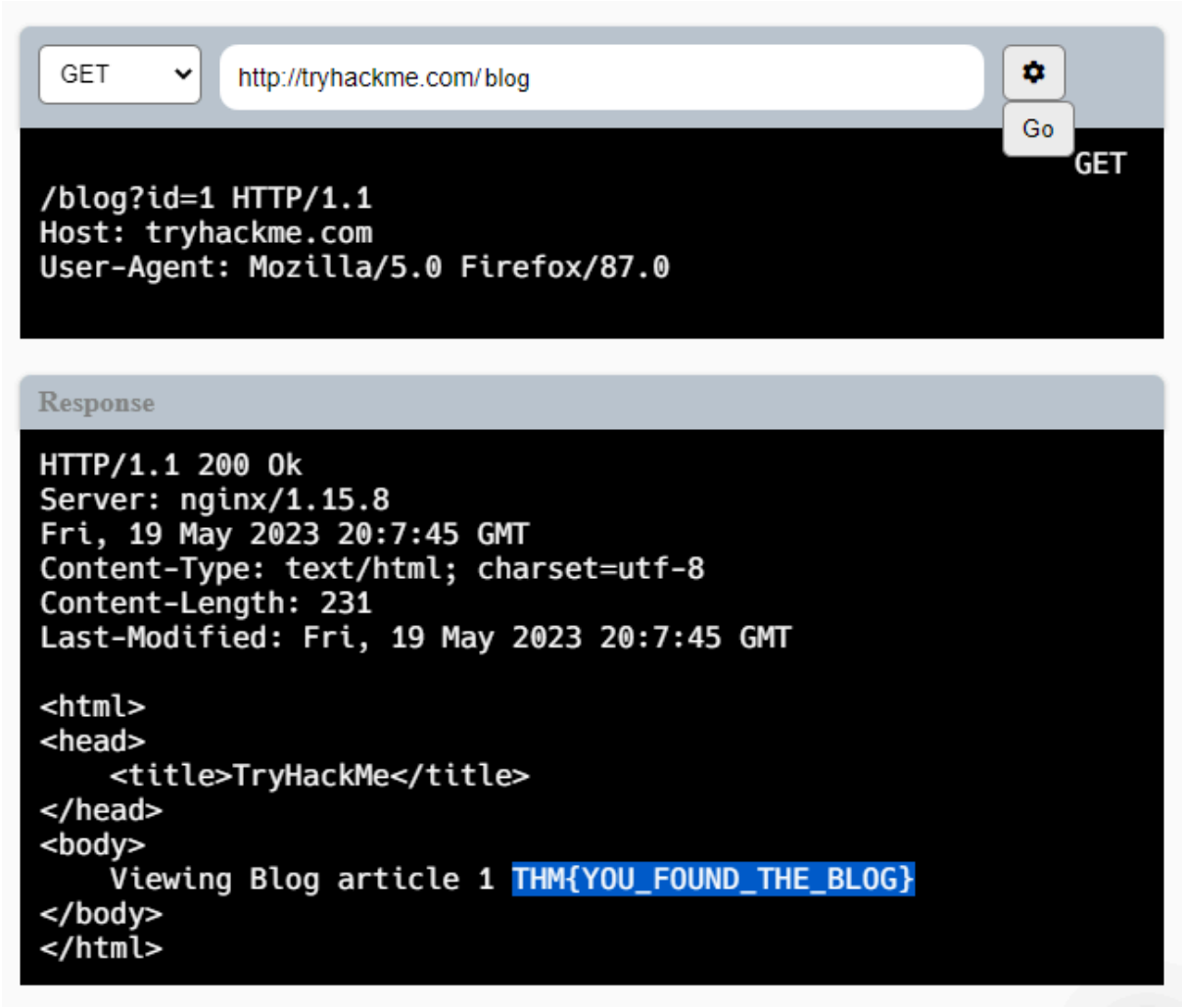
- The flag
- **Make a GET request to /blog and using the gear icon**

  set the id parameter to 1 in the URL field

- **<u>Answer</u>:** THM{YOU_FOUND_THE_BLOG}

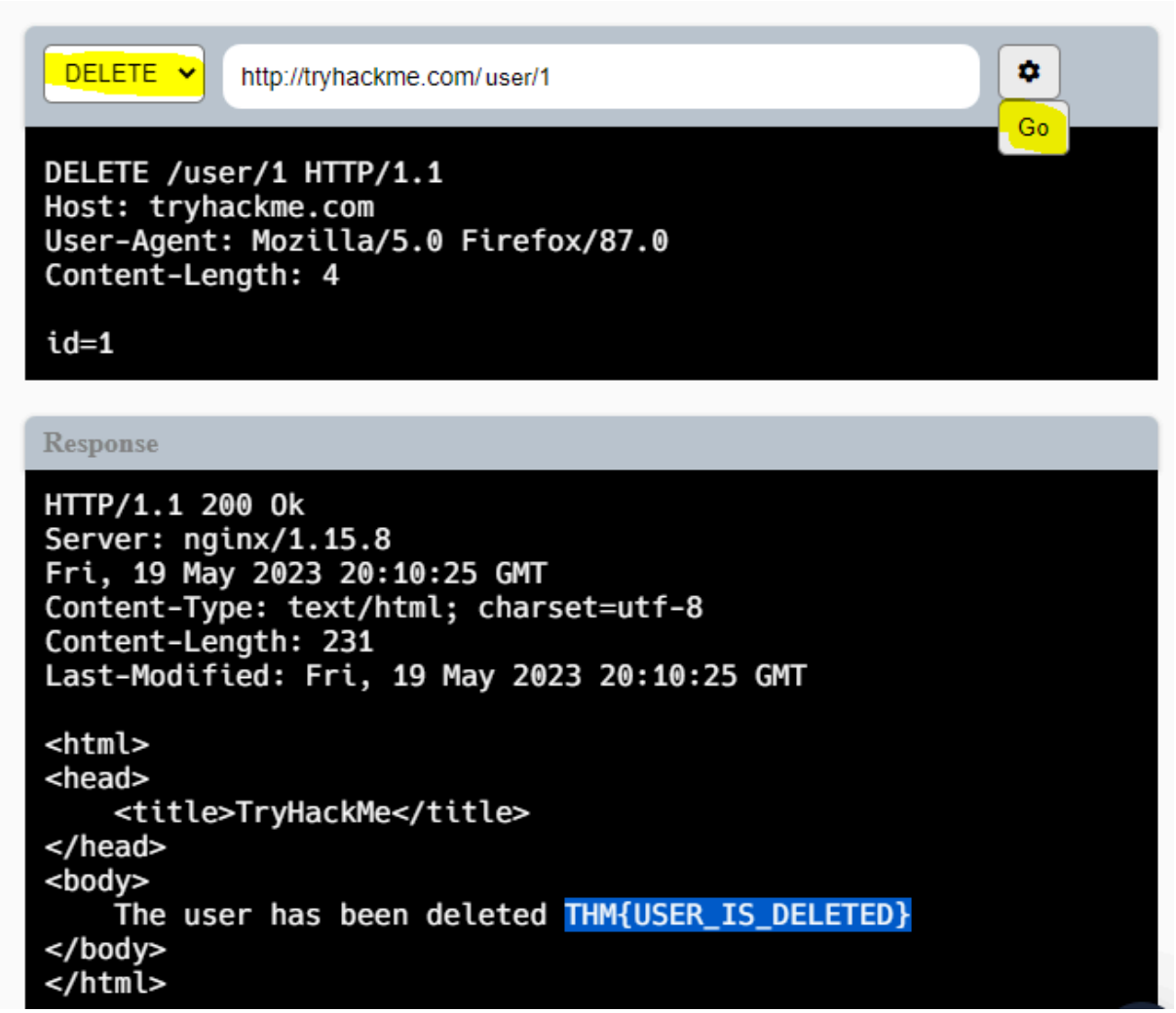
- **Press the gear icon**

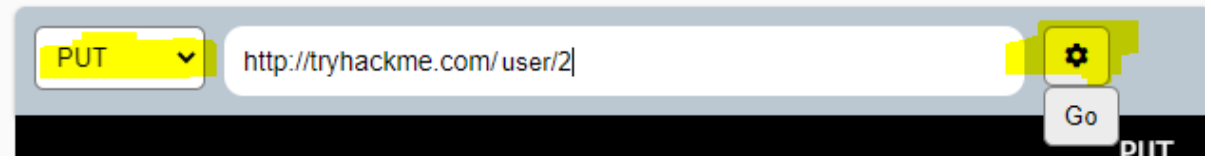
- 
GET
http://tryhackme.com/blog
Go
- 
GET
Parameters
key
value
id
1
Go
GET
/blog?id=1
Host: tryha
User-Agent:
Response
HTTP/1.1 200 Ok
Server: nginx/1.15.8
Fri, 19 May 2023 20:7:45 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 231
Last-Modified: Fri, 19 May 2023 20:7:45 GMT
<html>
<head>
<title>TryHackMe</title>
</head>
<body>
Viewing Blog article 1 THM{YOU_FOUND_THE_BLOG}
</body>
</html>
- The id

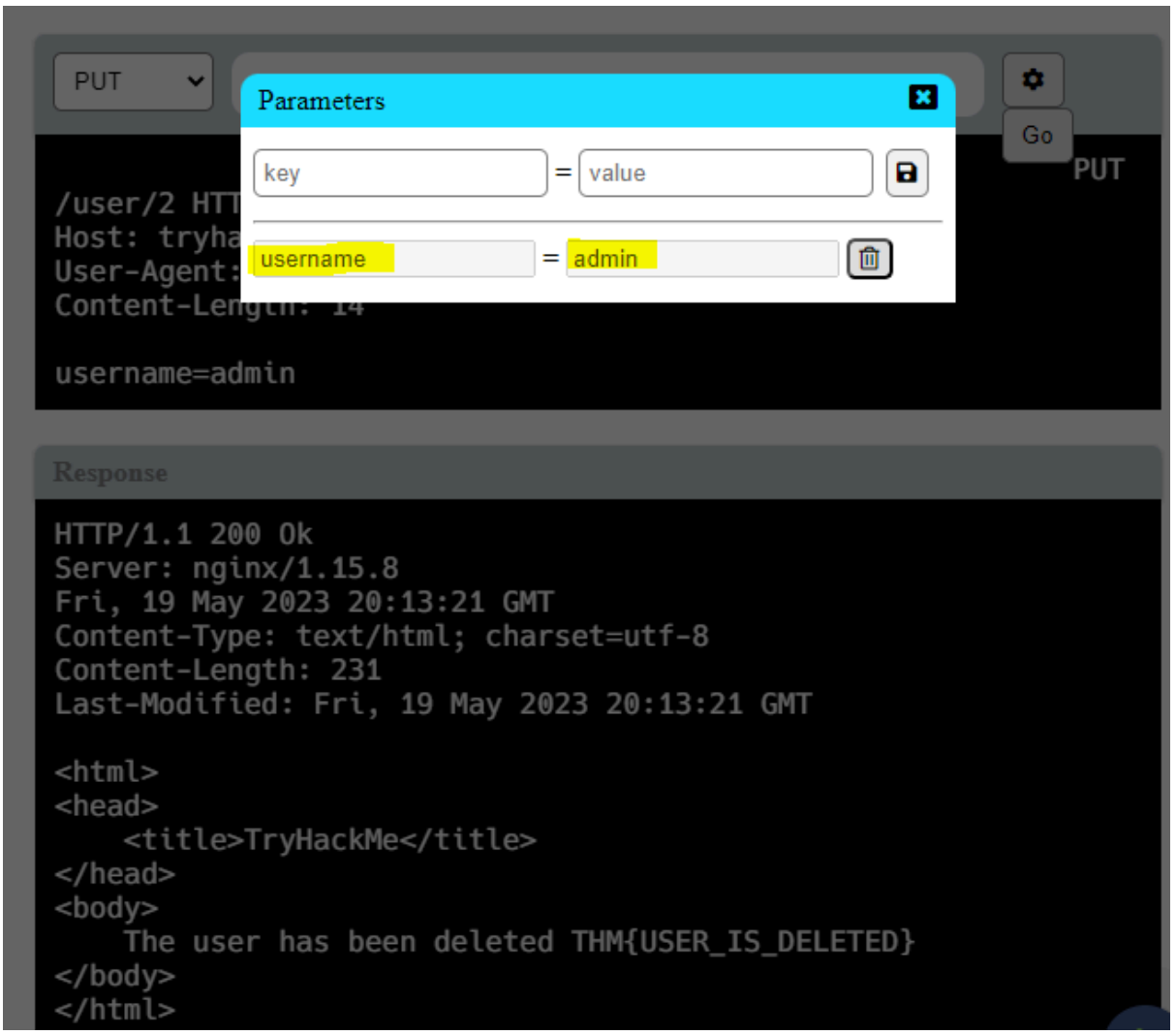


- The flag 2
- **Make a DELETE request to /user/1**

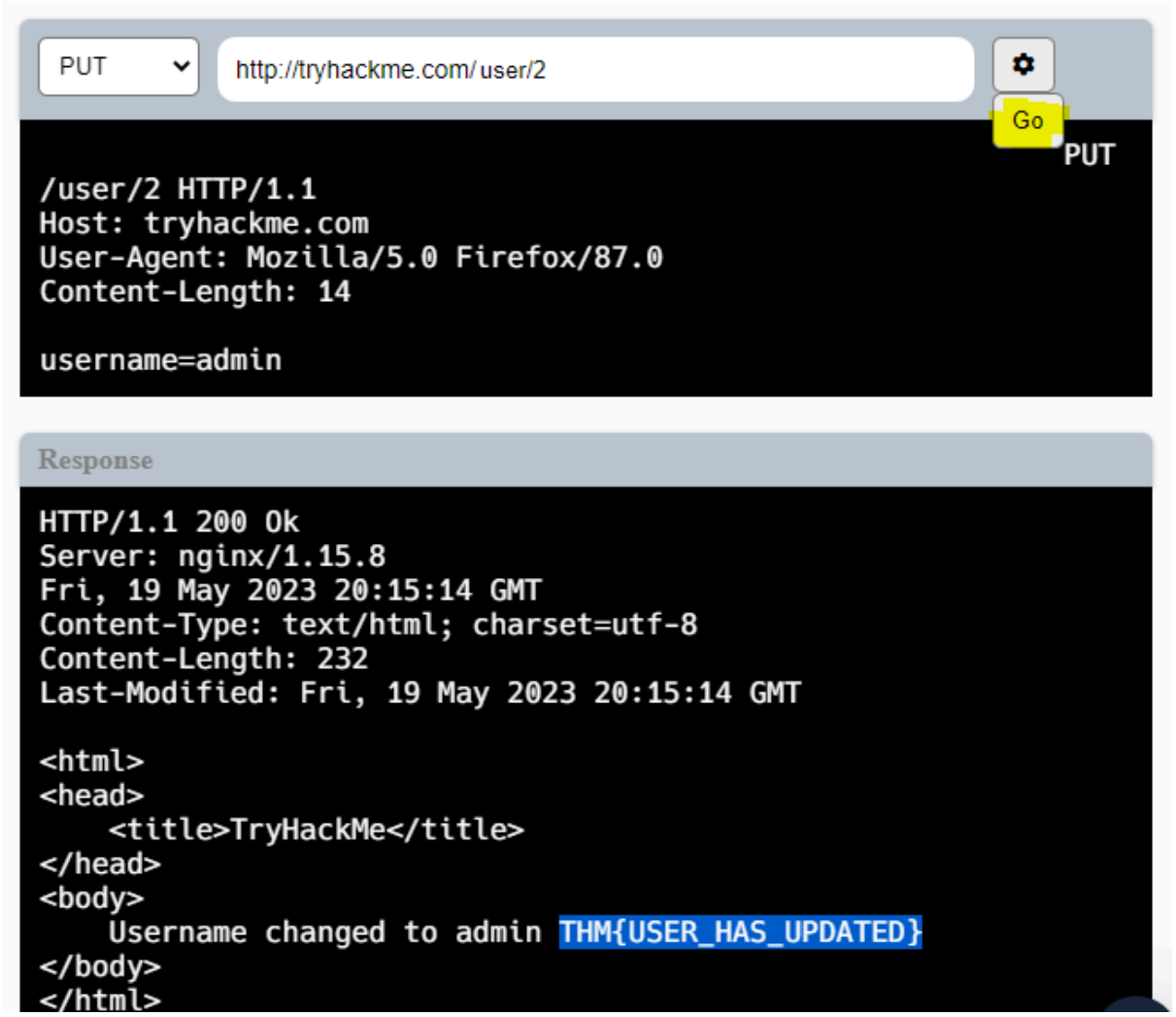
- **<u>Answer</u>:** THM{USER_IS_DELETED}

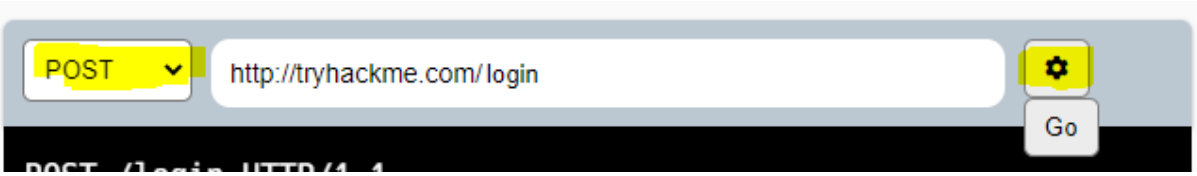
- 

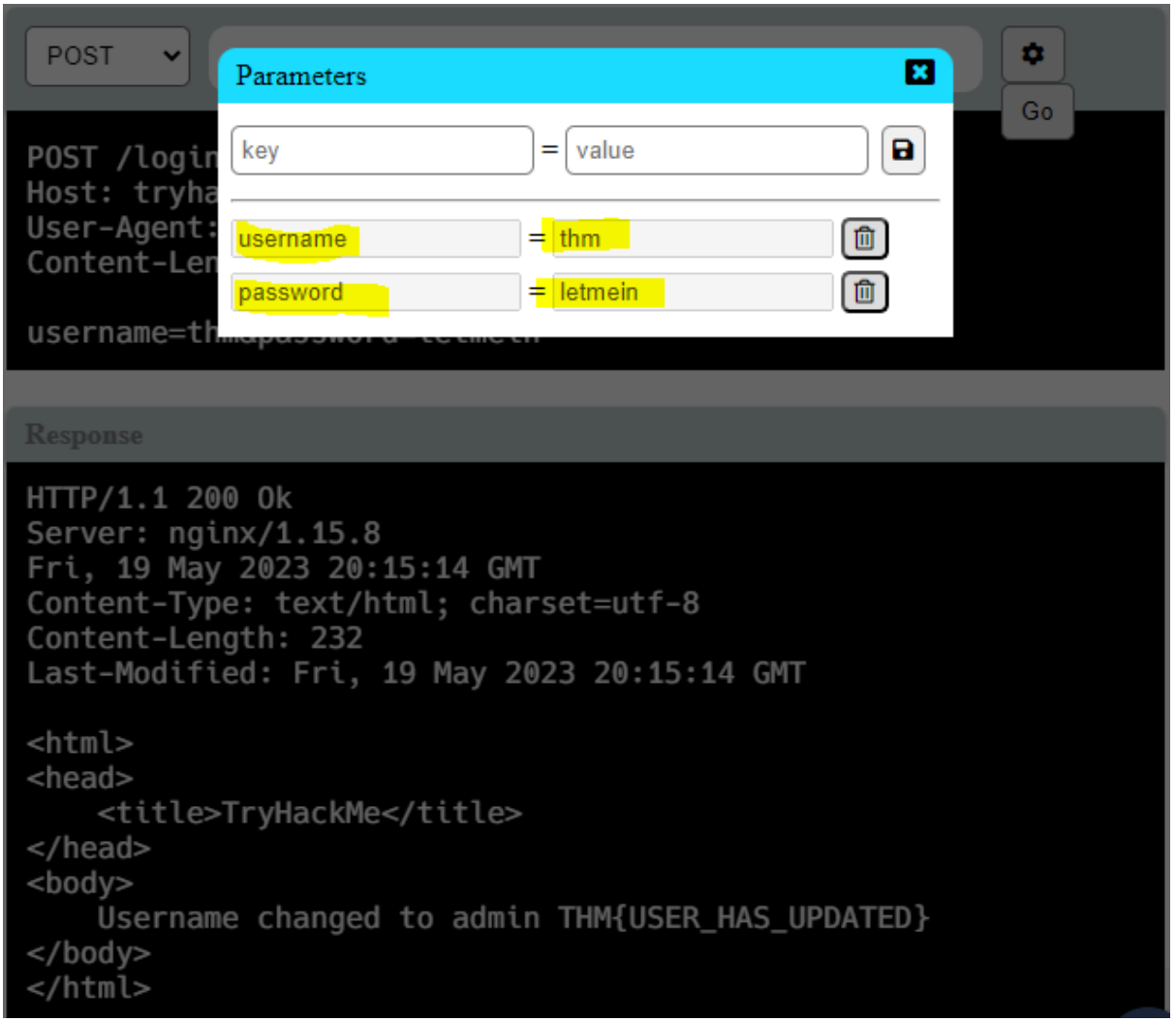
- The flag 3
- **Make a PUT request to /user/2 with the username parameter set to admin**
- **<u>Answer</u>:** THM{USER_HAS_UPDATED}
- **Press the gear icon**
- 

- 

- The username

- 

- The flag 4
- **POST the username of thm and a password of letmein to /login**
- **<u>Answer</u>:** THM{HTTP_REQUEST_MASTER}
- **Press the gear icon.**
- 

- 

- The username and password

- POST http://tryhackme.com/login Go

```
POST /login HTTP/1.1
Host: tryhackme.com
User-Agent: Mozilla/5.0 Firefox/87.0
Content-Length: 33

username=thm&password=letmein
```

Response

```
HTTP/1.1 200 Ok
Server: nginx/1.15.8
Fri, 19 May 2023 20:20:33 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 237
Last-Modified: Fri, 19 May 2023 20:20:33 GMT

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

- The flag 5
- **So, Happy learning happy journey.**
- **To get more interesting and detailed articles [follow my blog]**