

Laboratory: **under-construction**

Description:

Found this web application that is still under construction..
I'm sure it's vulnerable because of that. Can you find it?

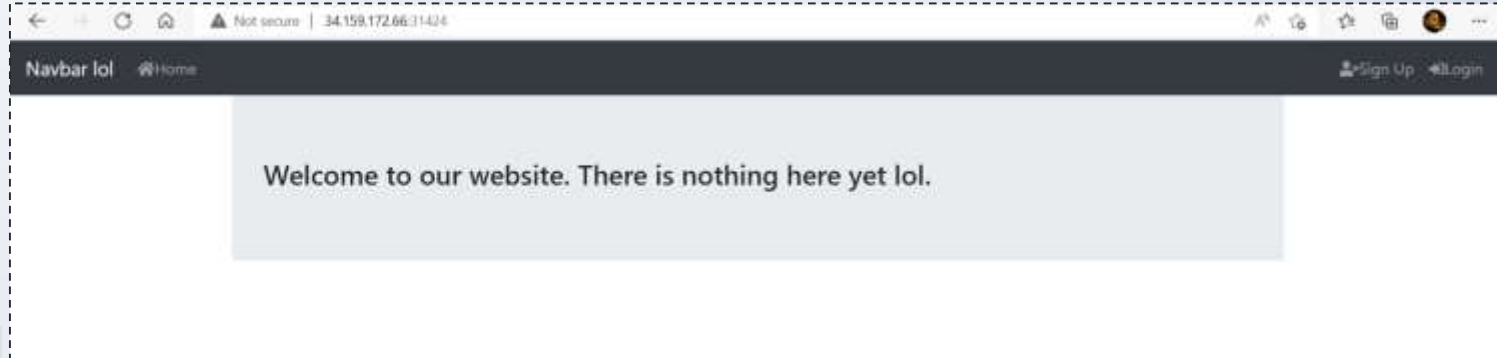
Flag format: CTF{sha256}

Level: Medium

Server: 34.159.172.66:31424

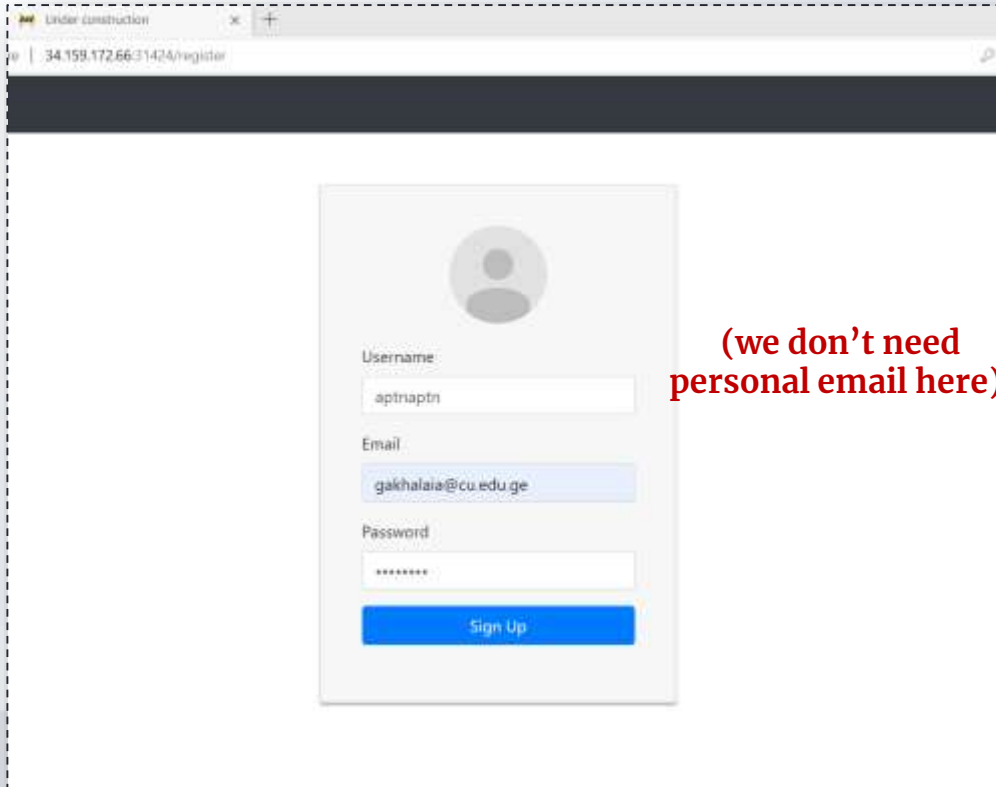
Hints:

- Hint 1: JWT Token with weak secret



Laboratory: **under-construction**

First, we need to **create an account** on the register page



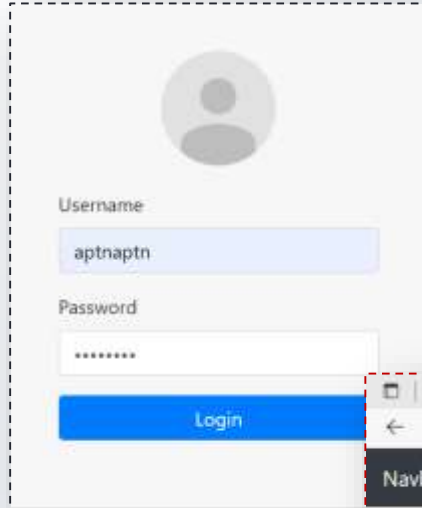
The screenshot shows a web browser window with the address bar displaying ".34.158.172.66:31424/register". The page has a dark header bar. Below it, a registration form is centered. The form includes a placeholder for a profile picture, followed by input fields for "Username" (containing "aptnaptn"), "Email" (containing "gakhalala@cu.edu.ge"), and "Password" (containing "*****"). A blue "Sign Up" button is at the bottom of the form.

(we don't need personal email here).

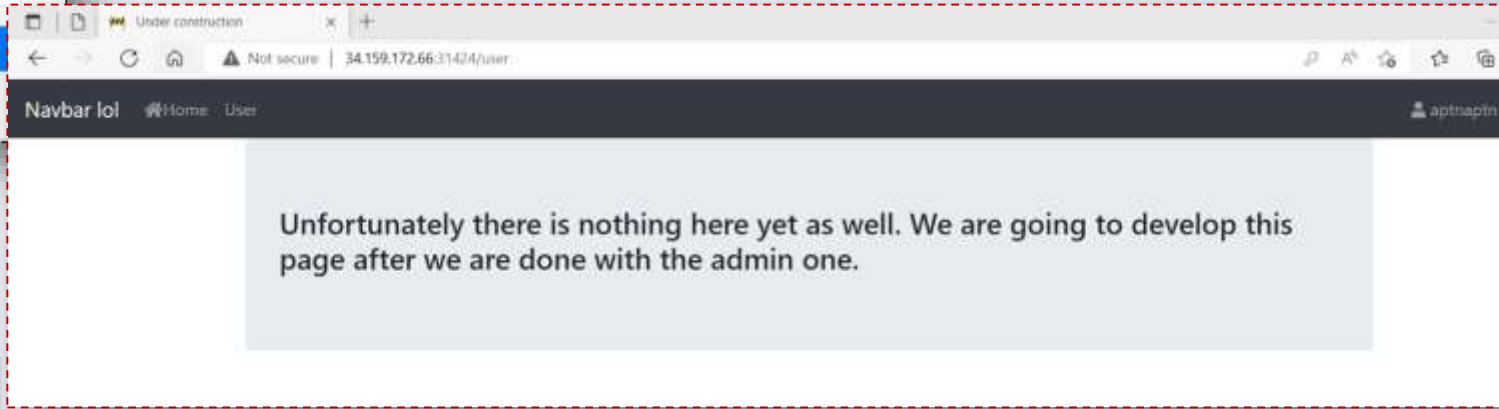


Laboratory: **under-construction**

After authentication process we can go to **user** area, but will get an error message.

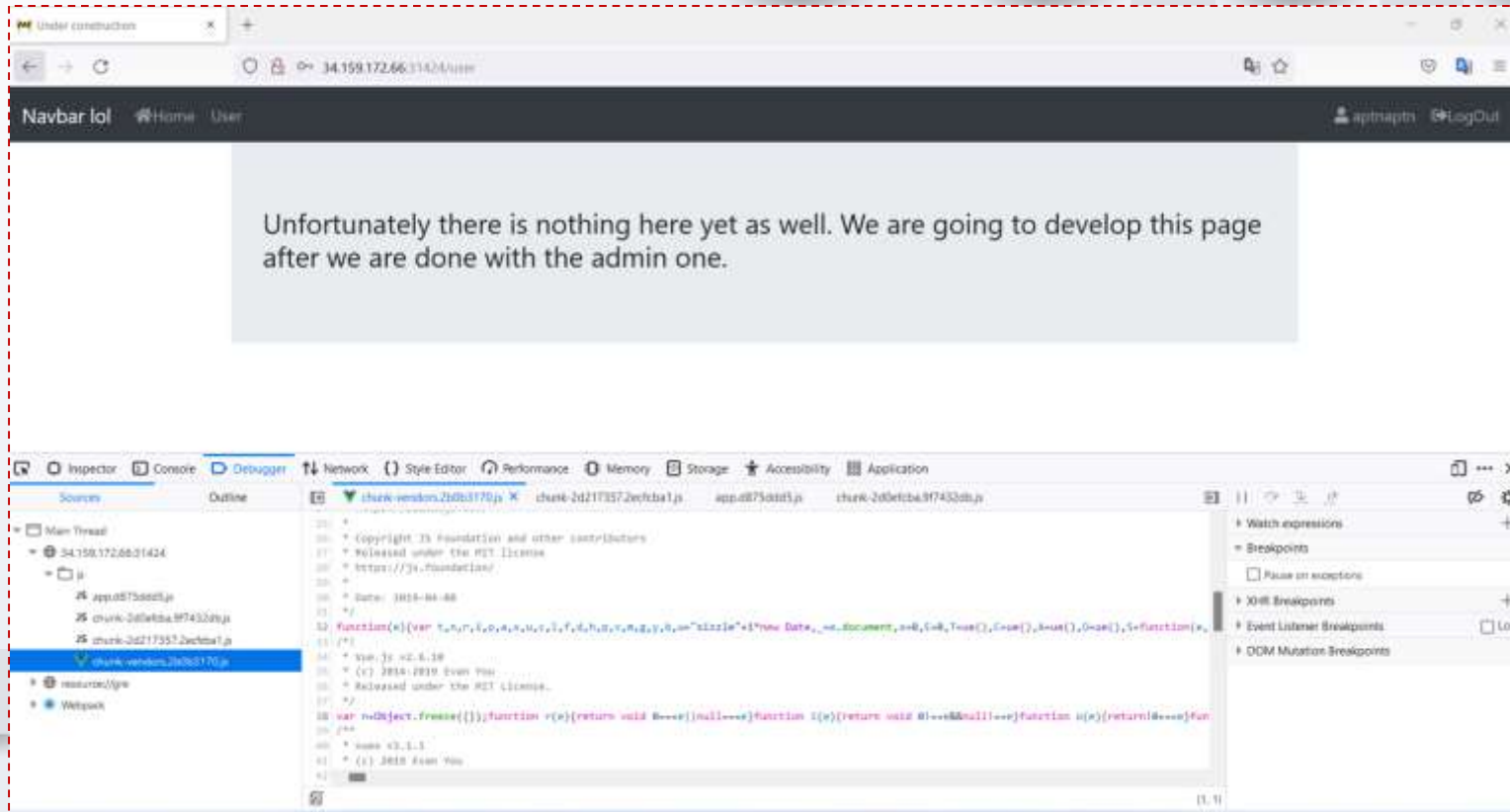


A login form with a dashed border. It features a circular profile icon placeholder at the top. Below it are two input fields: 'Username' containing 'aptnaptn' and 'Password' containing a masked password '*****'. A blue 'Login' button is positioned at the bottom of the form.



Laboratory: under-construction

From inspect/developer tools we can obtain, that the technology used is **Vue.js**



Laboratory: **under-construction**

Which means that:

- All javascript scripts included in the page are not readable, but if we add '**.map**' to the end of their URL we can see the original source.
- Application data is most likely stored in **localStorage** instead of cookies.
- If we look in **/js/app.d875ddd5.js.map**, we can see that there is a role called **ROLE_ADMIN**, which is most likely only given to admins. We can use the page debugging console to find an object in localStorage called user:

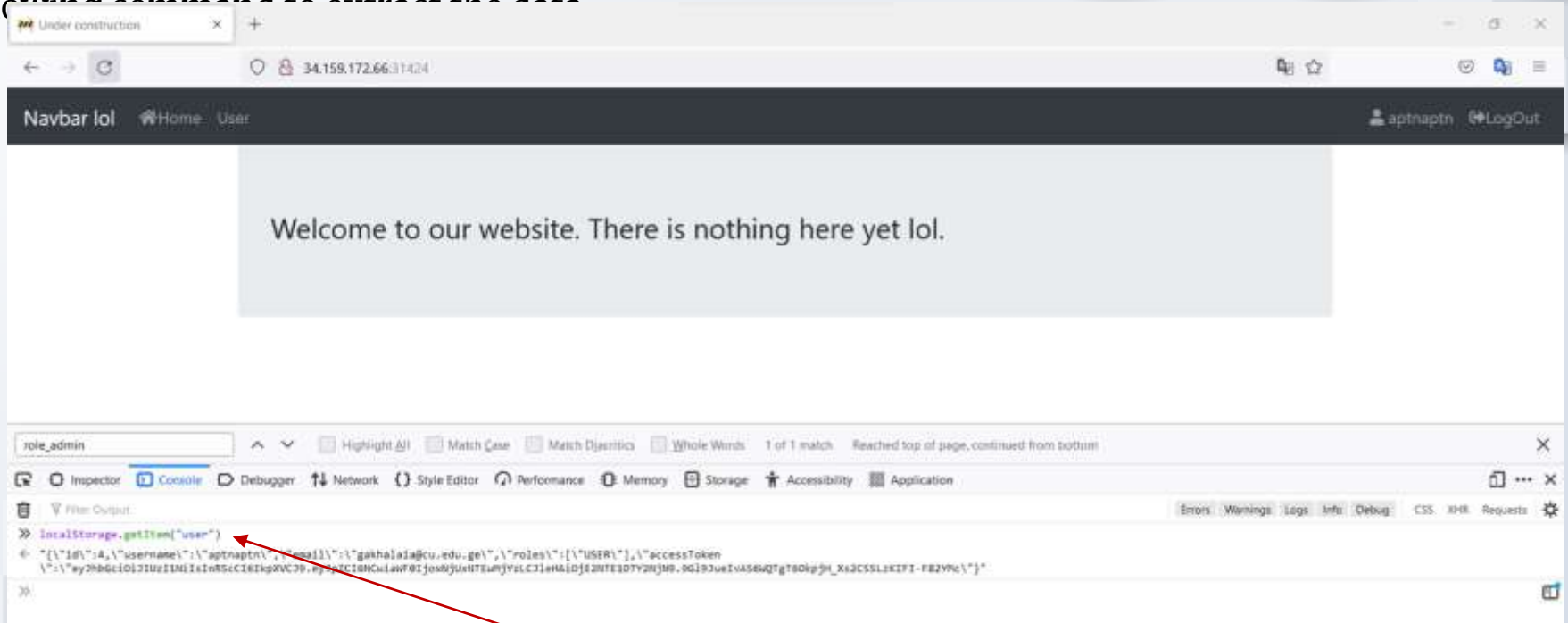
Laboratory: under-construction

If we add **.map** to one of the scripts, we can get the information, including **ROLE_ADMIN**. Now we know the exact name for the role of the administrator of the sy

The screenshot shows a web browser displaying a Vue.js application. The address bar shows the URL `34.159.172.66:31424/ju/app.d075ddc1.js.map`. The page content includes a search bar with the text `role_admin` and a dropdown menu. The developer console is open, showing the `Sources` tab with the file `resource://devtools-client-javascript`. The console also displays the `Ctrl+P` and `Ctrl+Shift+F` shortcuts. The application is running on a server with the IP address `34.159.172.66` and port `31424`.

Laboratory: **under-construction**

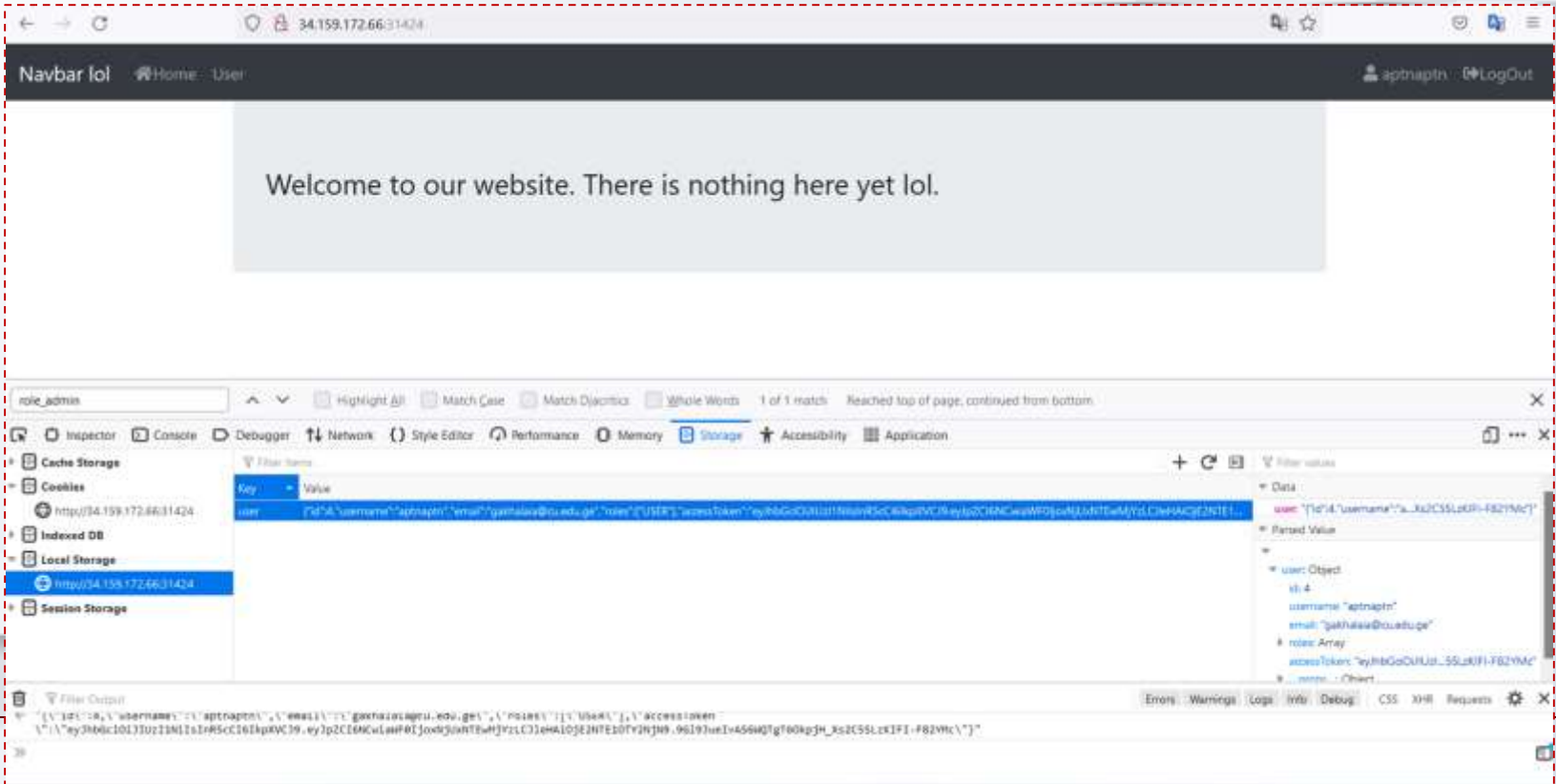
Now we need to modify the existing information using console of the browser. As we assumed, that information is saved in local storage, let us use the following command to extract the data



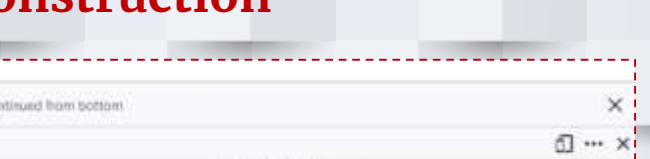
localStorage.getItem("user")

Laboratory: **under-construction**

Once we open the tab “Storage” the full value of it is shown. From here we can try to modify the existing information. We can see, that **accessToken** is also in value:



r-construction



Top of page, continued from bottom

Application

Filter values

Data

URI: "/id/4/username/"a_Xa2C55LzKFI-F82YMc"

Parsed Value

user Object

id: 4

username: "aptnagtri"

email: "gqkhtalala@ou.edu.ge"

roles Array

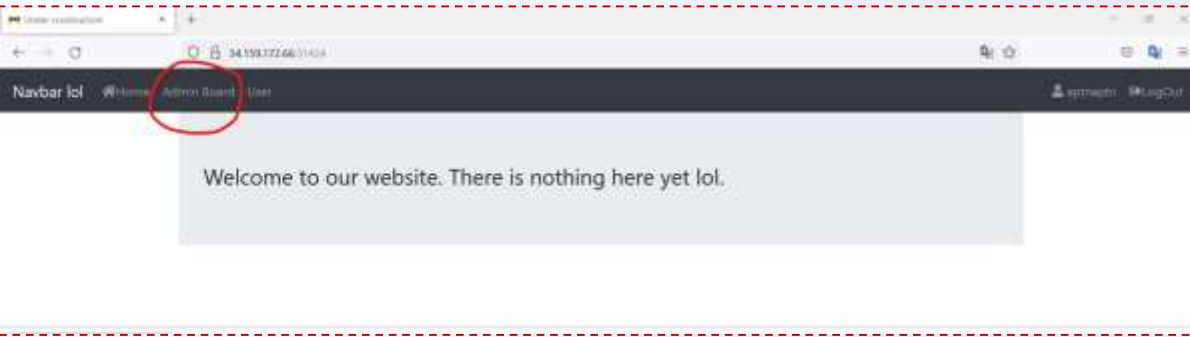
access_token: "eyJhbGciOiJIUzIuLCJ3IjoiOiJFb2YMc"

with "ROLE_ADMIN"

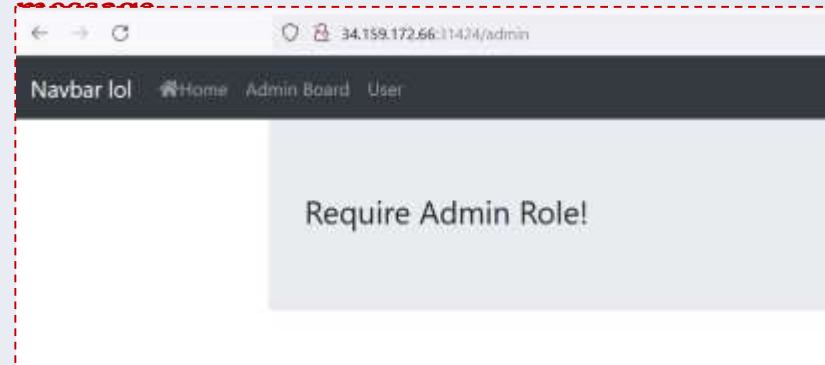


Laboratory: **under-construction**

Once we refresh the page, we will see new menu item “Admin board”

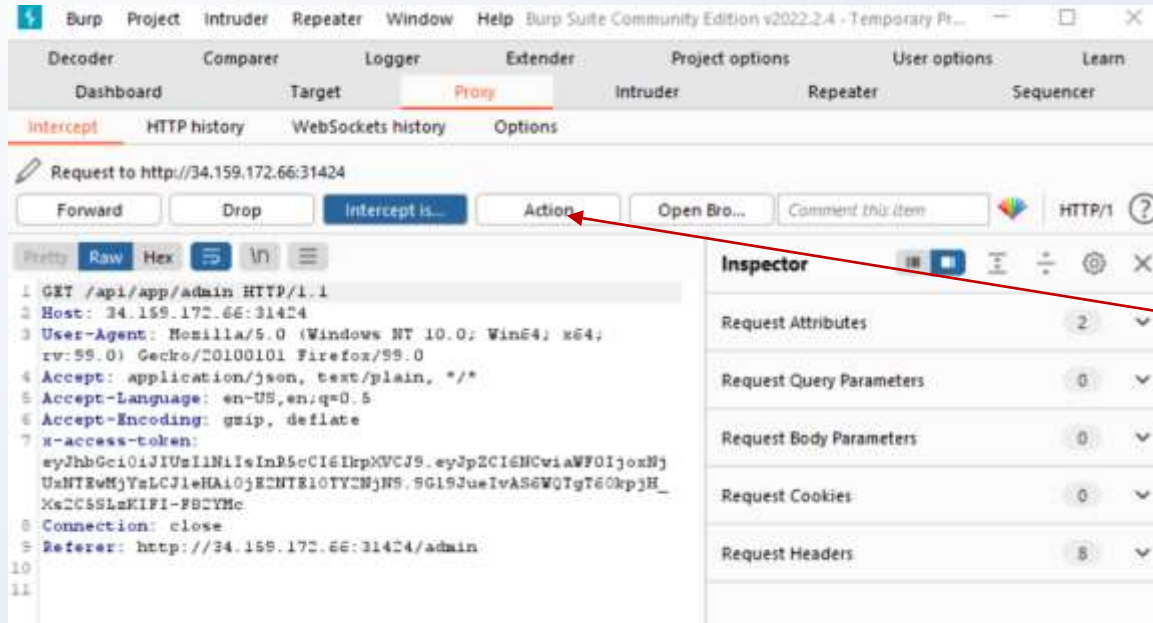


But when we click it, we are getting the **error**



Laboratory: **under-construction**

To modify the information “**on fly**” we can catch it using Burp suite tool. From the request we can see the pass of the application `/api/app/admin`



click **Action** and send it to the **Repeater**

Laboratory: **under-construction**

In this case, we need to **break the token** and **get the secret key**. We will use **JWT Tool** for these purposes.

The JSON Web Token Toolkit v2

`jwt_tool.py` is a toolkit for validating, forging, scanning and tampering JWTs (JSON Web Tokens).

version **v2.2.4** python **v3.6+**

The logo for JWT_Tool is displayed in a large, blue, pixelated font. The text 'JWT_Tool' is the main focus, with '@ticarpi' written in a smaller, blue, pixelated font directly below it. The entire logo is set against a dark background within a screenshot of a terminal or code editor.

Laboratory: **under-construction**

In `jwt_tool.py` we need to define the target [URL with the obtained path] and the **token** we want to be broken. We use brute force method to do it. Different wordlists can be used, including **rockyou.txt**, placed right in the directory with

```
kali@kali: ~/jwt_tool
File Actions Edit View Help
(kali@kali)-[~/jwt_tool]
$ python3 jwt_tool.py -t http://34.159.172.66:31424/api/app/admin -rc "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Njw0IjoxNjUxNTEwMjYzLCJleHAiOjE2NTE1OTY2NjN9.9Gl9JueIvAS6WQTgT60kpjH_Xs2C5SLzKIFI-F82YMc" -C -d rockyou.txt
```

Result

```
kali@kali: ~/jwt_tool
File Actions Edit View Help

  JWT_Tool
  Version 2.2.5 @t1carpi

Original JWT: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Njw0IjoxNjUxNTEwMjYzLCJleHAiOjE2NTE1OTY2NjN9.9Gl9JueIvAS6WQTgT60kpjH_Xs2C5SLzKIFI-F82YMc

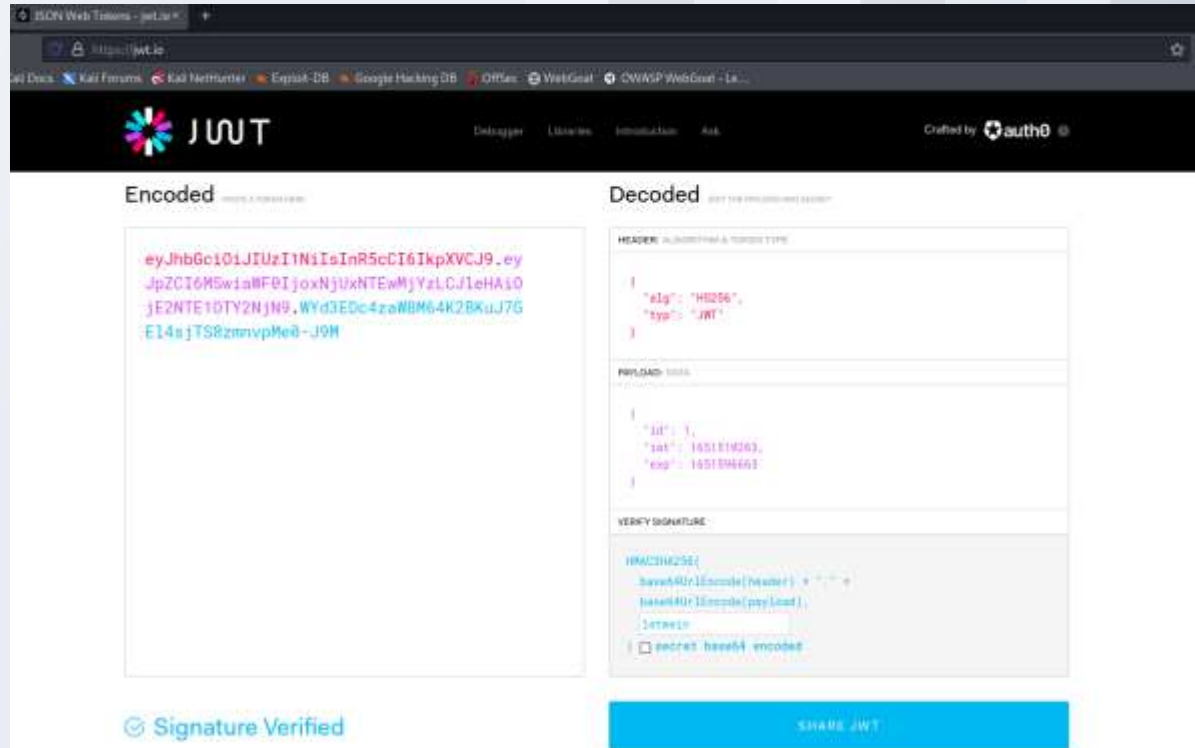
[+] letmein is the CORRECT key!
You can tamper/fuzz the token contents (-T/-I) and sign it using:
python3 jwt_tool.py [options here] -S H5256 -p "letmein"

(kali@kali)-[~/jwt_tool]
$
```

Laboratory: **under-construction**

Now, using jwt.io service we can decode the token from our request by adding the secret key.

Don't forget to change the ID on 1.



The screenshot shows the JWT.io website interface. The 'Encoded' section on the left contains the following JWT token:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6M5wiaWF0IjoxNjUxNTEwMjYzLjE4HAiOiJlZ2NTE1OTY2NjN9.WYd3EDc4zaWBM64K2BKuJ7GE14sjsTS8zmvpmE0-J9M
```

The 'Decoded' section on the right shows the token's structure:

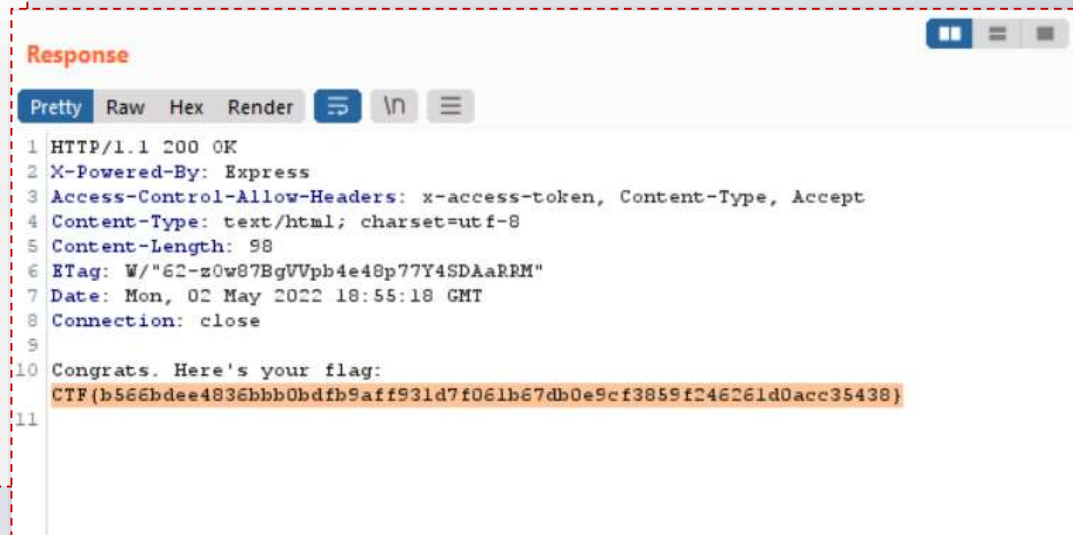
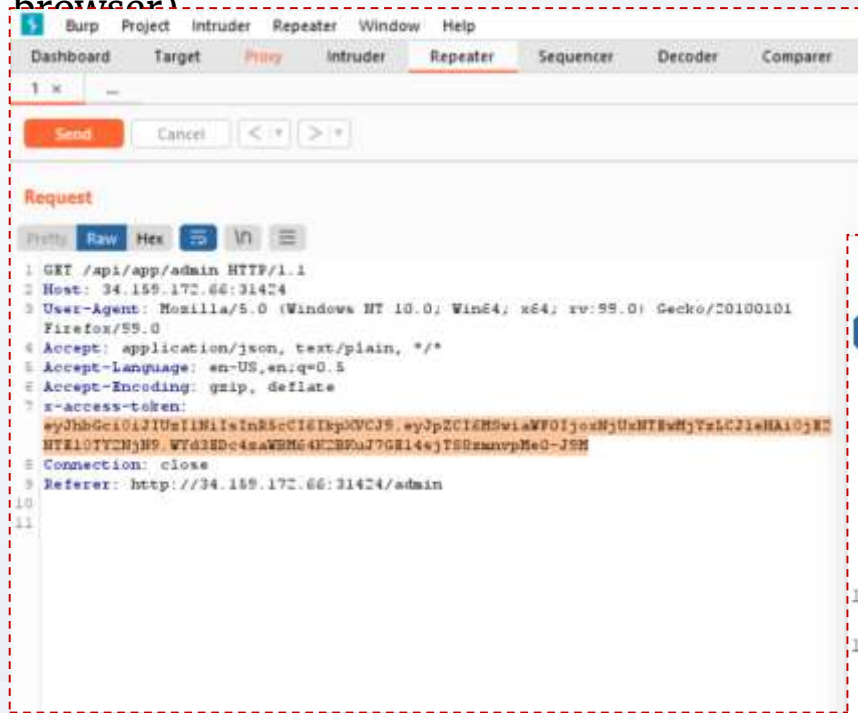
- HEADER:** {"alg": "HS256", "typ": "JWT"}
- PAYLOAD:** {"id": 1, "iat": 1451819263, "exp": 145184663}
- VERIFY SIGNATURE:**

```
function verify(jwt, secret) {
  const header = jwt.split('.')[0];
  const payload = jwt.split('.')[1];
  const token = header + '.' + payload;
  return crypto.timingSafeEqual(
    Buffer.from(token),
    Buffer.from(secret)
  );
}
```

At the bottom left, a green checkmark and the text 'Signature Verified' are displayed. At the bottom right, a blue button labeled 'SHARE JWT' is visible.

Laboratory: **under-construction**

Now we need to put the modified token into the request in Burp Suite (use manual proxy in the browser)



CTF{b566bdee4836bbb0bdfb9aff931d7f061b67db0e9cf3859f246261d0acc35438}