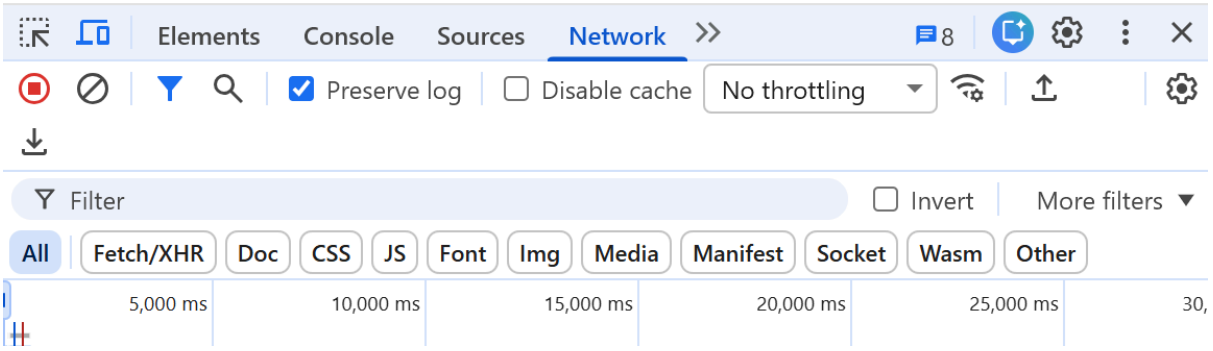
Fixed Sections from OWASP Top 10:

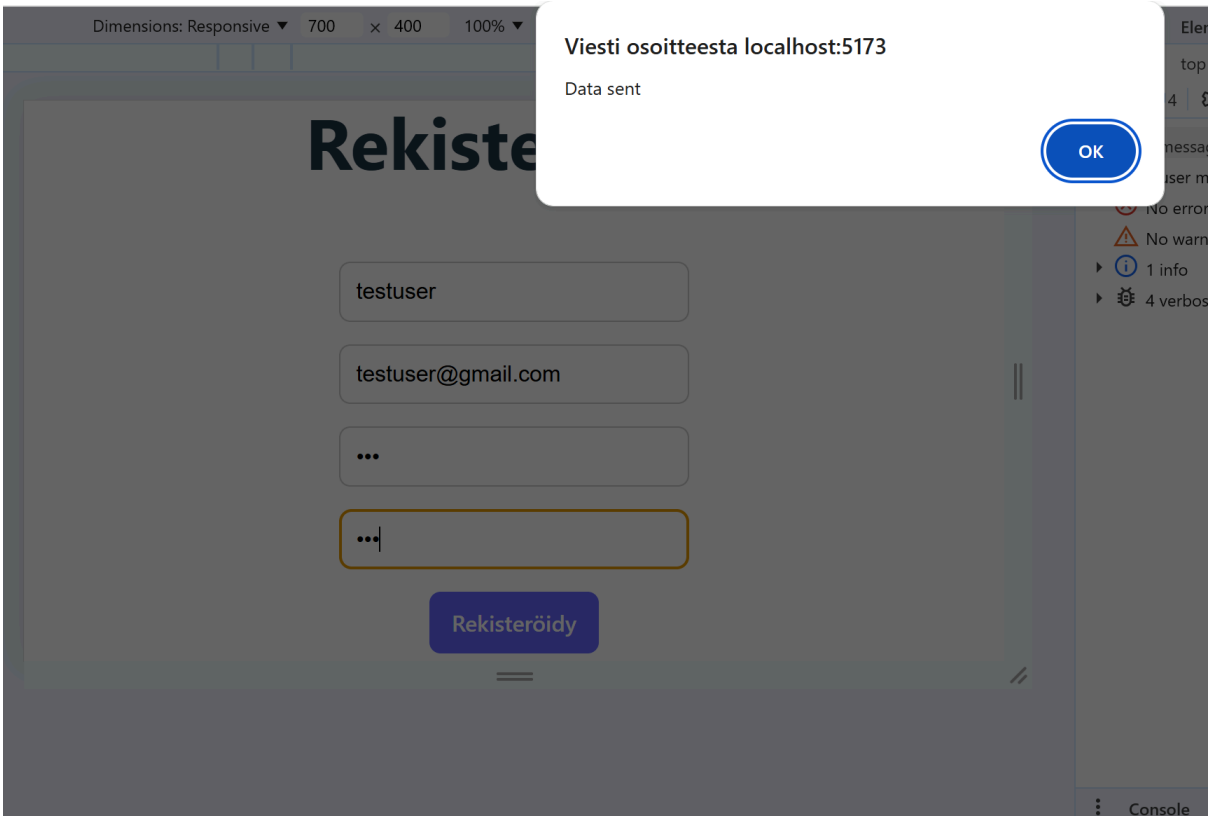
1. A02:2021 - Cryptographic Failures

Passwords being stored as plain text in storages

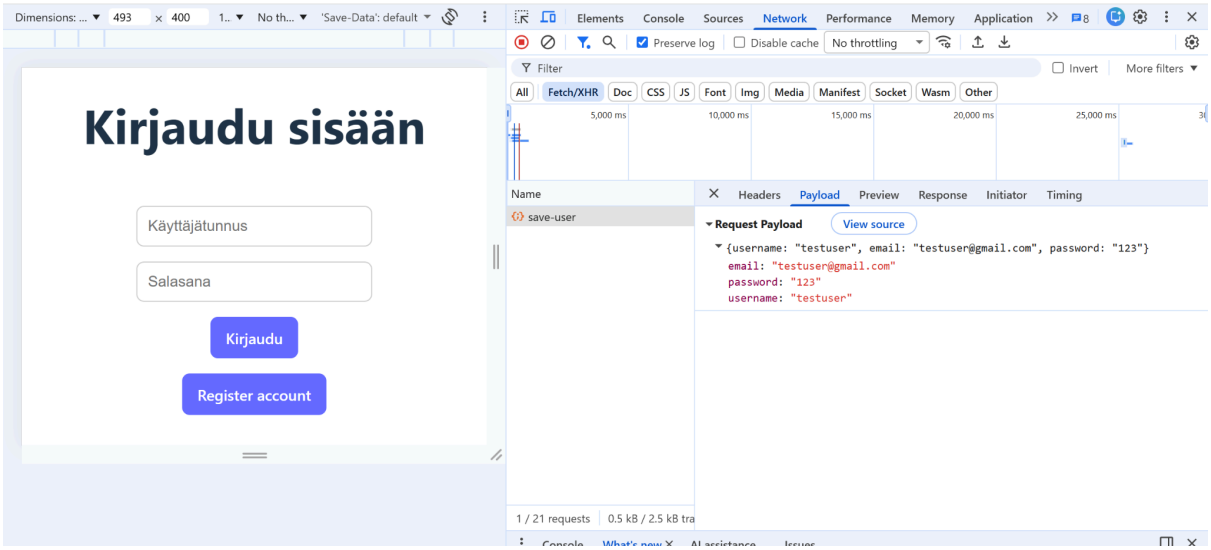
Assume we log in a test user:
Before sending the test user info, we go to developer tools.
Go to the network tab and check the “Preserve log” check box.



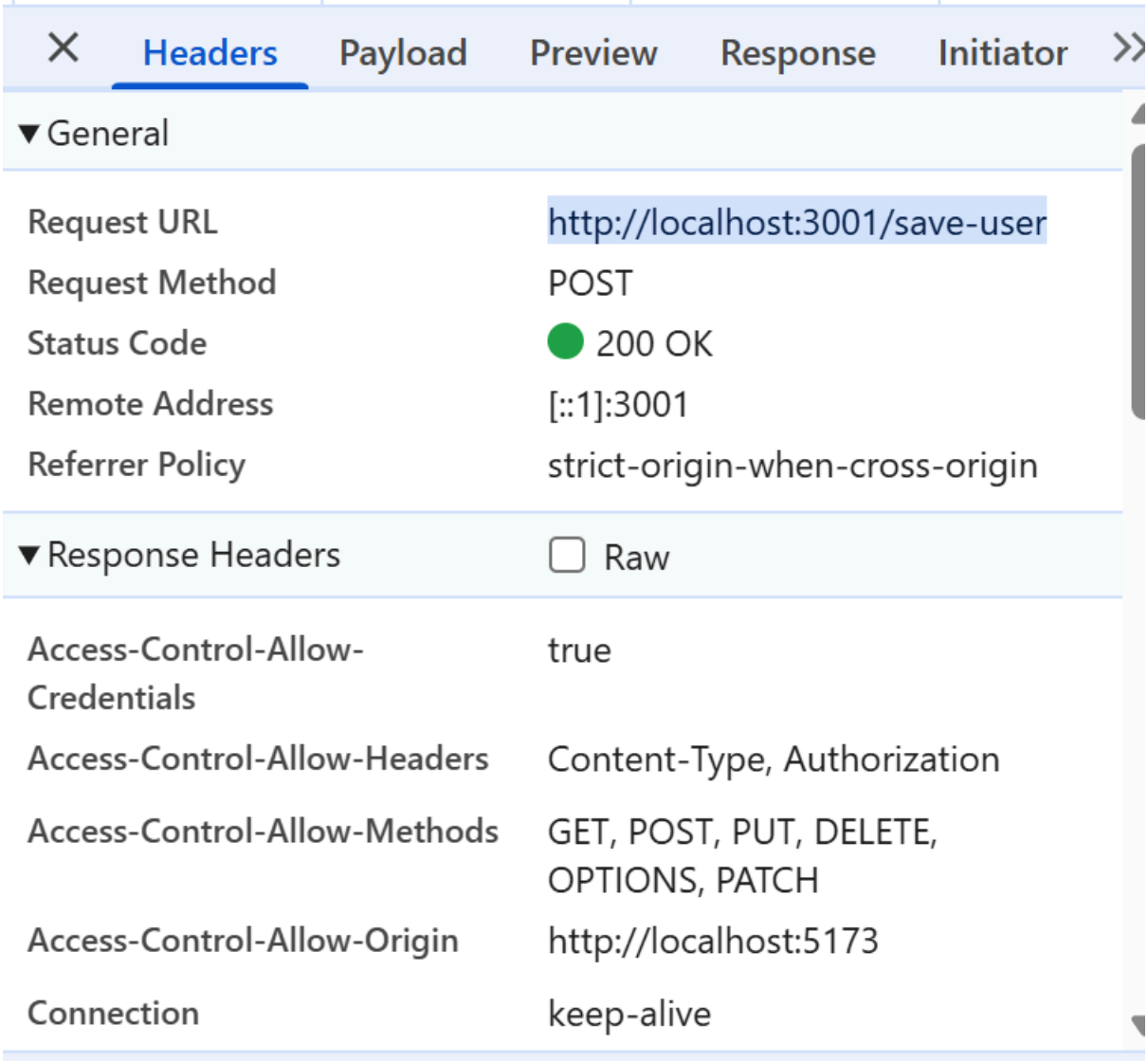
After doing so. Fill in the Registration form and send it



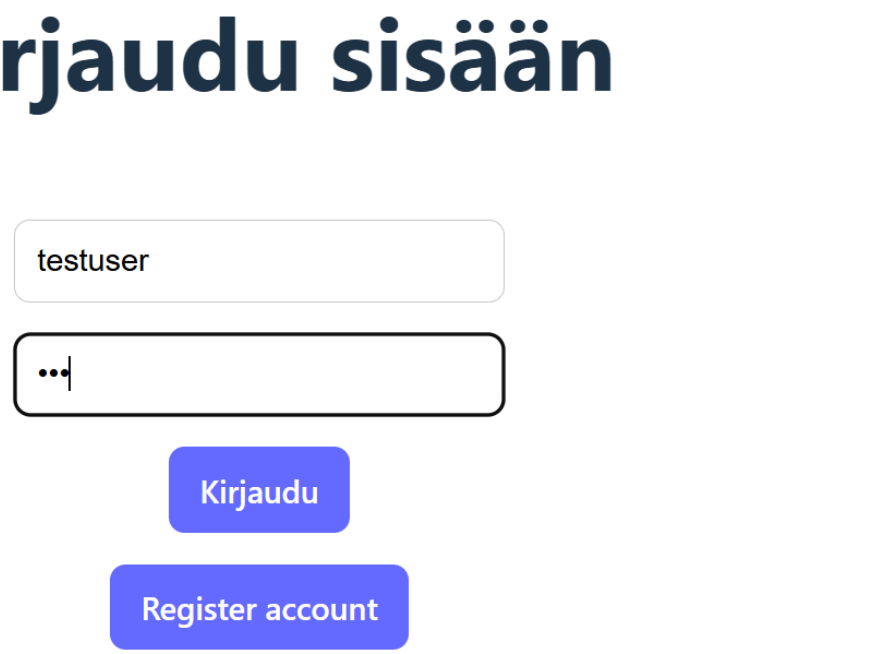
Then go to the network tab, and click the “save-user” event



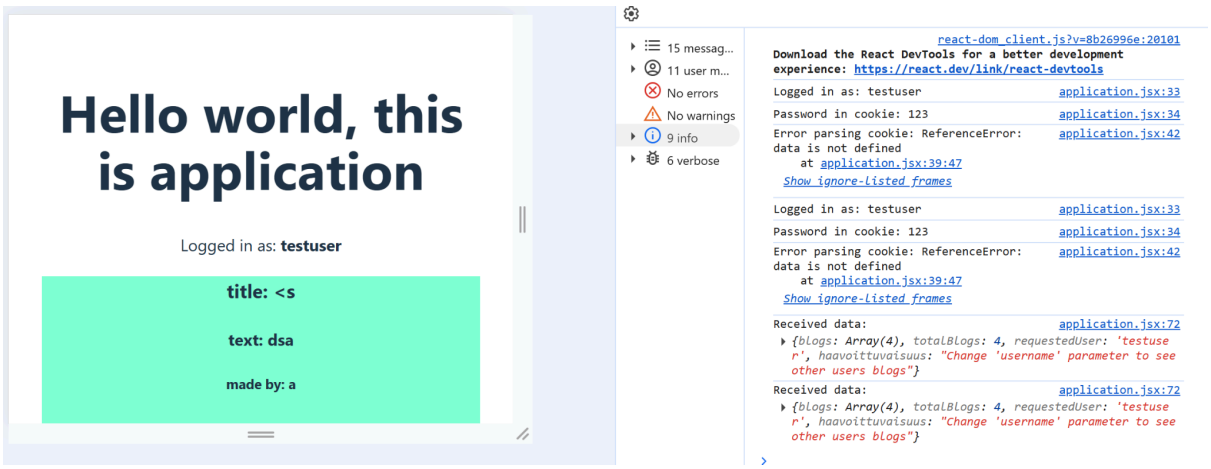
By clicking this event, you can clearly see the payload that the registration sends.
As seen from the image. The whole user information, including the password, is there in plain text. And the request url is: http://localhost:3001/save-user



Do not get this wrong, this itself is not a problem and a typical behavior of many web applications, since they usually go by HTTPS connection, but the real problem is that these passwords would be saved as plain text in the storage, and onto a cvs file. You will see how this affects later on, when the passwords are shown in plain text in cookies, too.
And that is not all.
Lets login with the newly made user.



1.

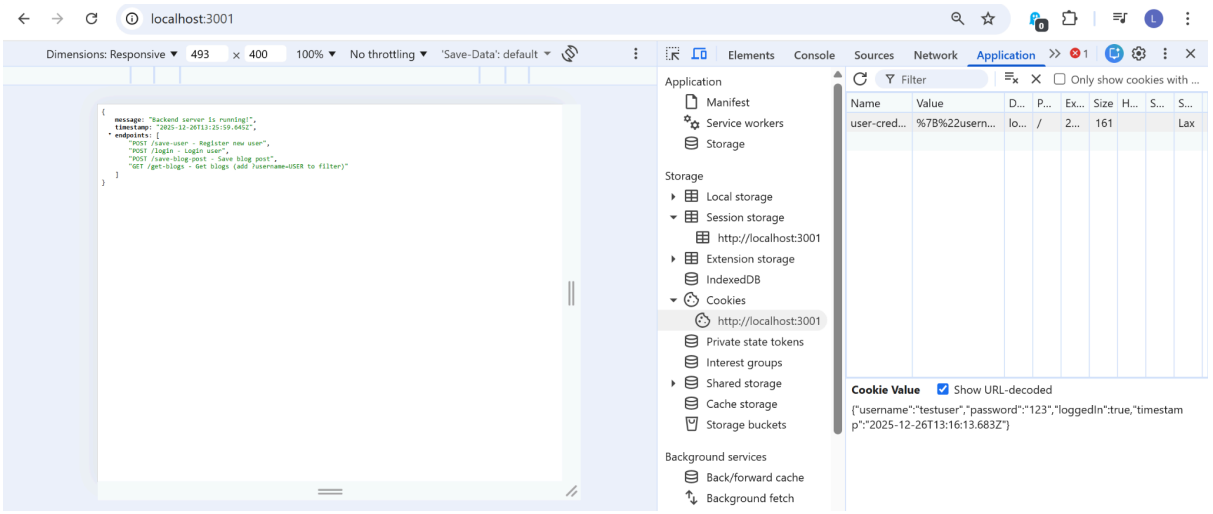


2. A05:2021 - Security Misconfiguration

There are already some messed-up development-time console logs in the console. The console logs give out too much data.
Lets continue with the previous one cryptographic failures.

But even if those console logs disappeared, the real vulnerability resides in the backend. Once logged in, let's navigate to the address of the backend. Which in this case is: http://localhost:3001/

After doing so, go to developed tools->Application and click cookies



Select the cookie from the backend's address (localhost:3001) and click the URL-decoded cookie value.
As shown in the image, the user data, a non-decrypted password (cryptographic failure), and the timestamp logged are displayed.

3. A07:2021 - Identification and Authentication Failures

There is a Lack of Session Validation (this makes horizontal privilege escalation possible in this case)

Lets register 2 users.

1. Jonne with password jonnePass
2. Liina with password liinaPass

Login as Liina and create a few Private posts

Hello world, this is application

Logged in as: **liina**

title: Liina's privat oist

text: this is private

made by: liina

secret? true

title: Liina's second private psot

text: no one can see this

made by: liina

secret? true

title: public to show example

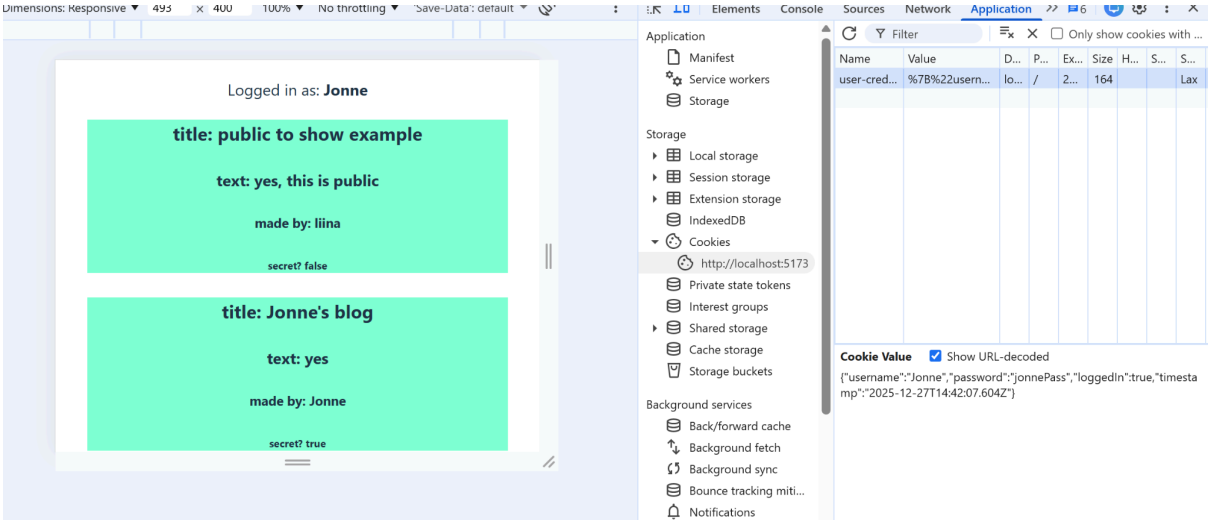
text: yes

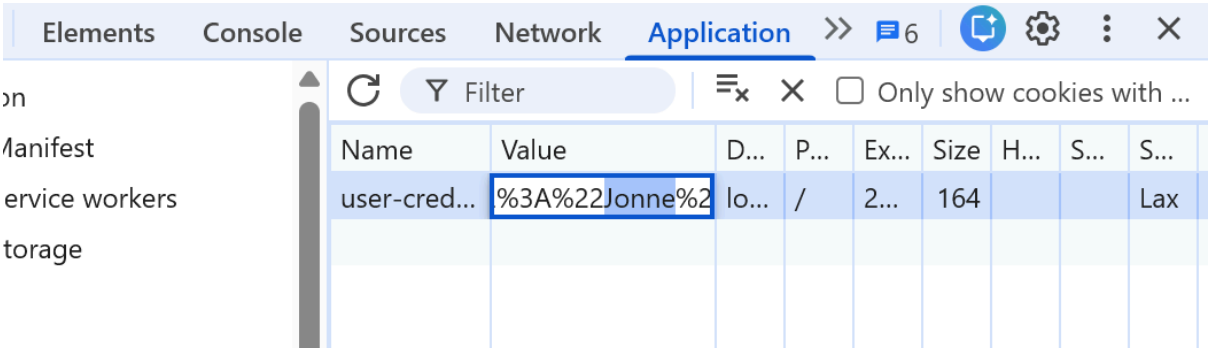
made by: liina

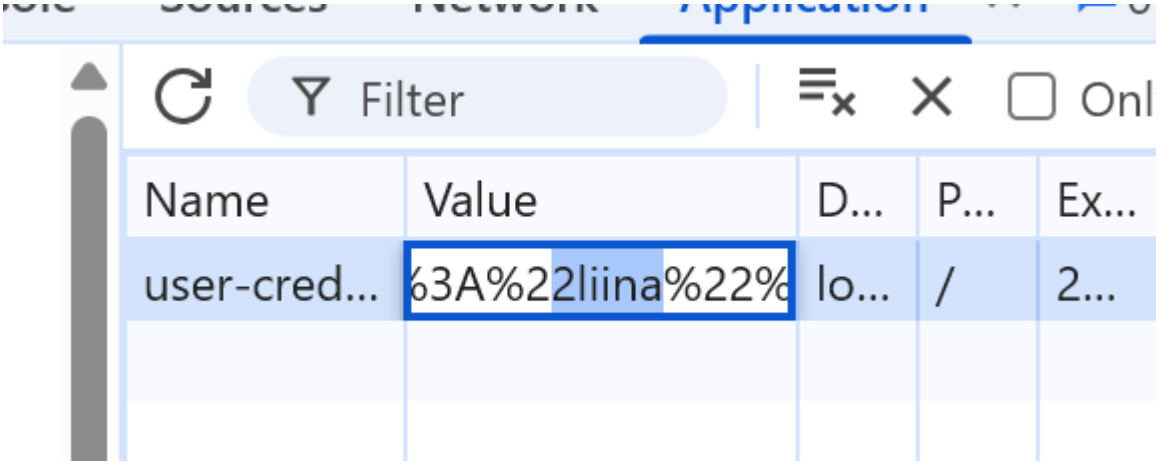
secret? this is public

Logout, and then
Log in with Jonne and note his cookie.
Without logging out, go to the developer tools -> Application -> Cookies
Manually edit the user-credentials cookie:

- Change “username”:”Jonne” to “username”:”liina”
- Keep everything else the same

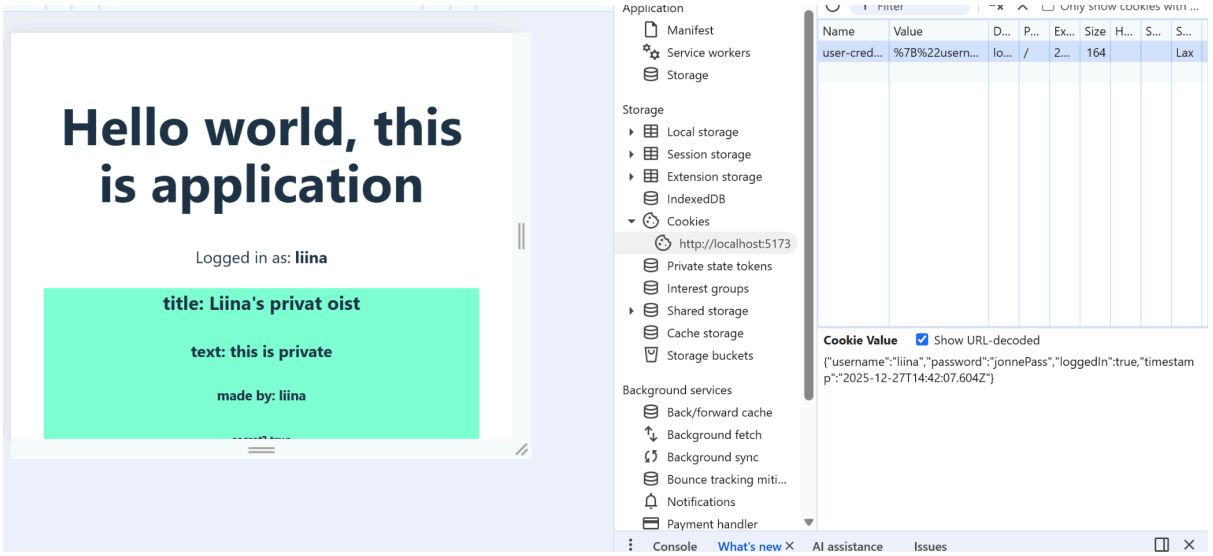


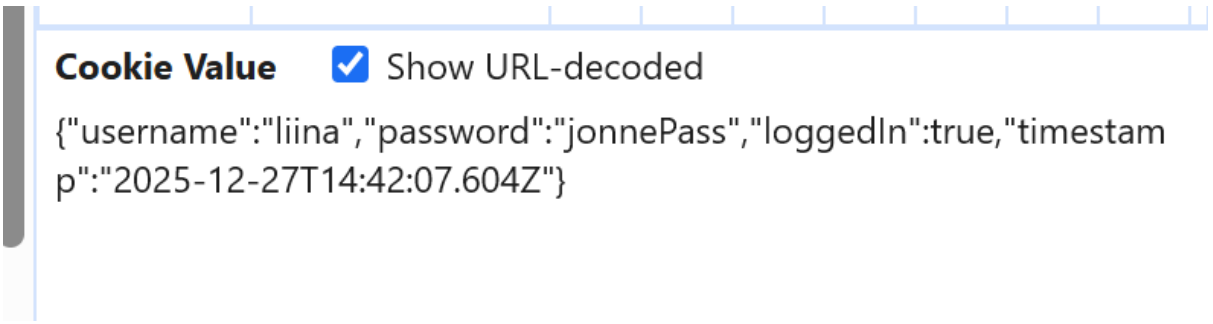




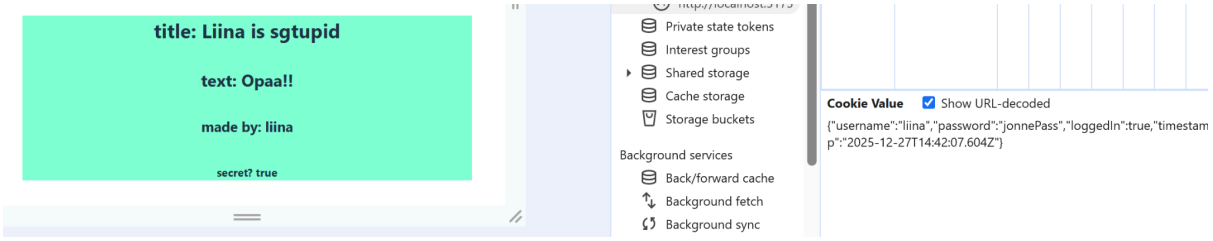
Press enter

Then refresh the page to be Logged in as liina

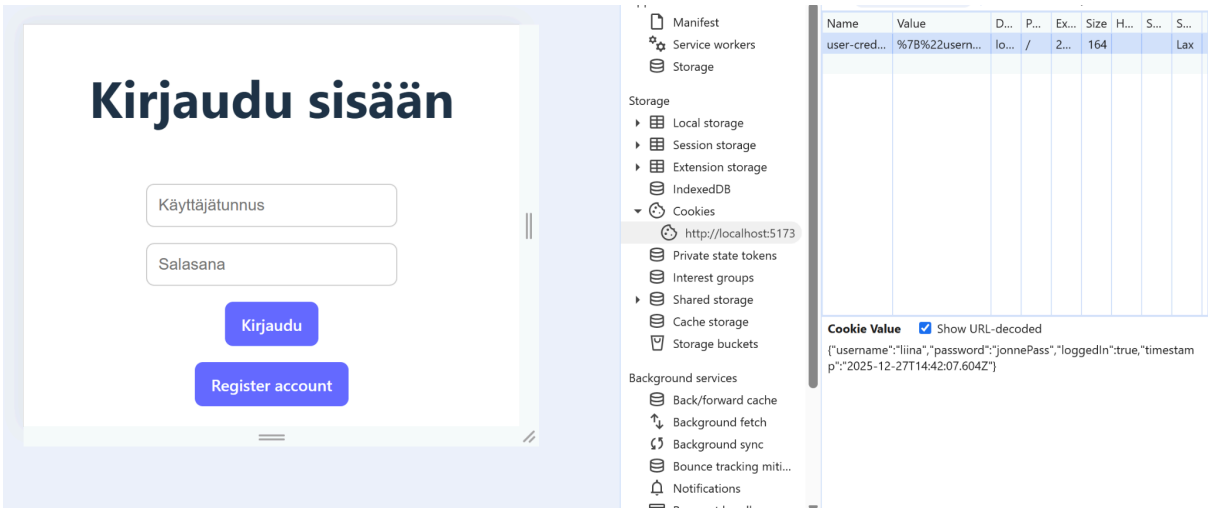




You can now also send new blogs disguised as the other person, too.

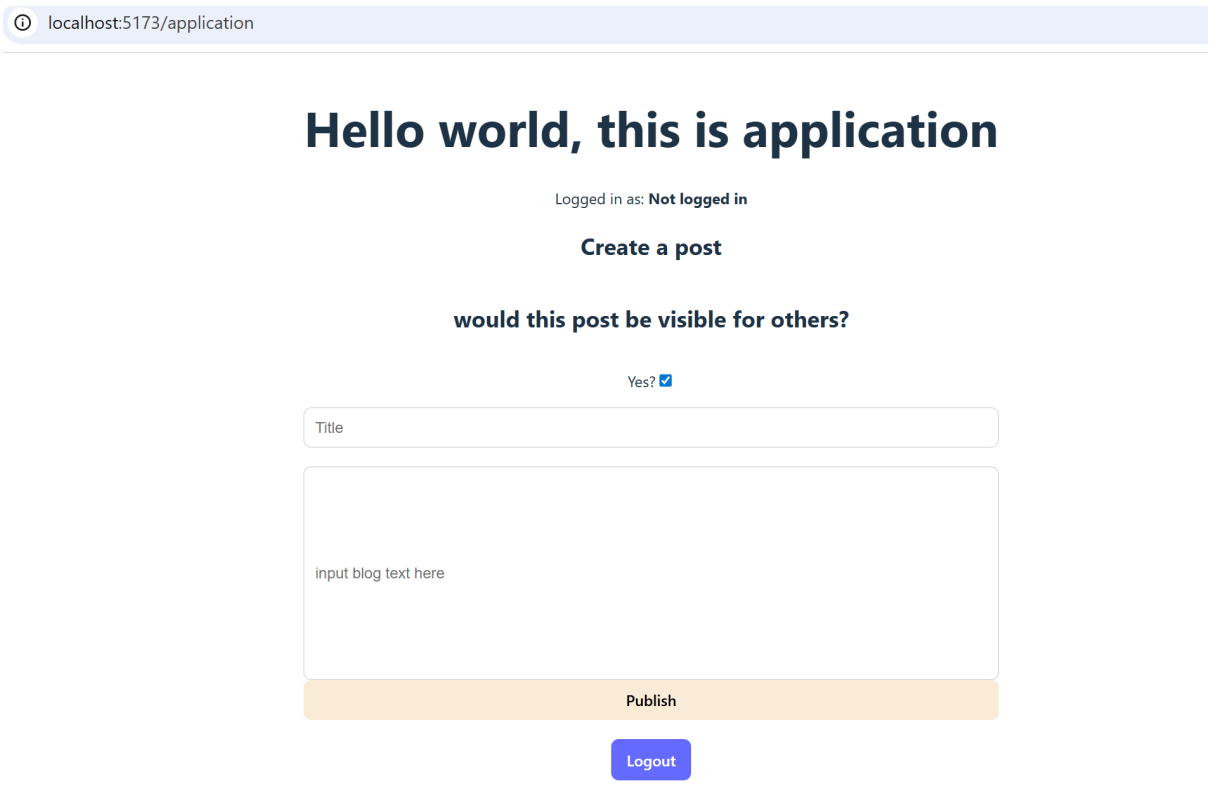


There is also another flaw, which is: No Session Expiration/Insecure Logout.
When logging out:
The previous cookie is till present:

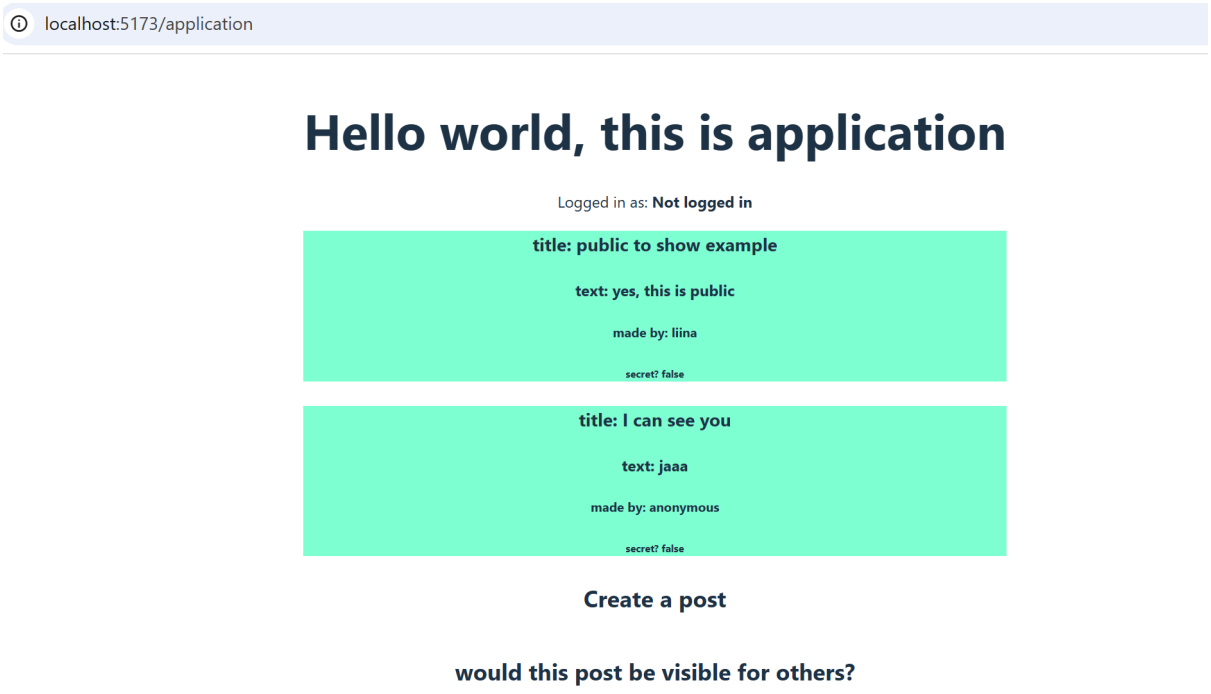


4. A01:2021 – Broken Access Control

You can access the application without being logged in.
Just by going through the route /application



This anonymous user can create his own blogs, and see others public posts by creating a post himself.

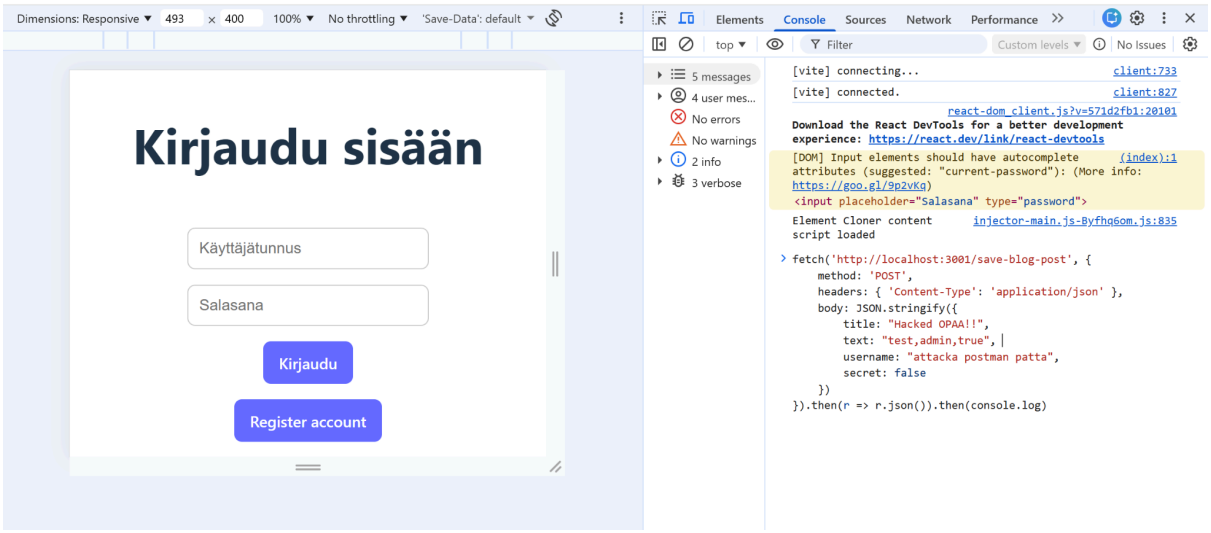


Note, that to see the posts he wrote and to have them displayed for others, you must select the private check box off.

5. A01:2021 – Broken Access Control

Another exploit on this topic.
The input validation is lacking. The save-blog-post route is vulnerable, anyone who would put onto the browser terminal:

```
fetch('http://localhost:3001/save-blog-post', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
        title: "Hacked OPAA!!",
        text: "test,admin,true",
        username: "attacka postman patta",
        secret: false
    })
}).then(r => r.json()).then(console.log)
```



No need to even log in, just type it into the terminal, and press enter.

```
> fetch('http://localhost:3001/save-blog-post', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
          title: "Hacked OPAA!!",
          text: "test,admin,true",
          username: "attacka postman patta",
          secret: false
      })
  }).then(r => r.json()).then(console.log)
<· ▶ Promise {<pending>}
   ▶ {message: 'Blog saved!'}
>
```

You can either go in as anonymous by using the broken access control or just by logging in as a normal user to see the hacked post.

Anyone could make posts and disguise themselves as anyone, despite the “person” not being in the system.

6. A06:2021 - Vulnerable and Outdated Components

```
Leone@LAPTOP-5OFSB5M1 MINGW64 ~/cyberhaava/my-noob-react-app (master)
$ npm audit
# npm audit report

cookie  <0.7.0
cookie accepts cookie name, path, and domain with out of bounds characters - htt
ps://github.com/advisories/GHSA-pxg6-pf52-xh8x
fix available via `npm audit fix`
node_modules/cookie-parser/node_modules/cookie
  cookie-parser  1.0.1 - 1.4.6
  Depends on vulnerable versions of cookie
  node_modules/cookie-parser

2 low severity vulnerabilities

To address all issues, run:
  npm audit fix
```

It is clear that the current cookie library is outdated and vulnerable, as seen from the npm audit.

```
Leone@LAPTOP-5OFSB5M1 MINGW64 ~/cyberhaava/my-noob-react-app (master)
$ npm list cookie-parser
my-noob-react-app@0.0.0 C:\Users\Leone\cyberhaava\my-noob-react-app
└── cookie-parser@1.4.5
```

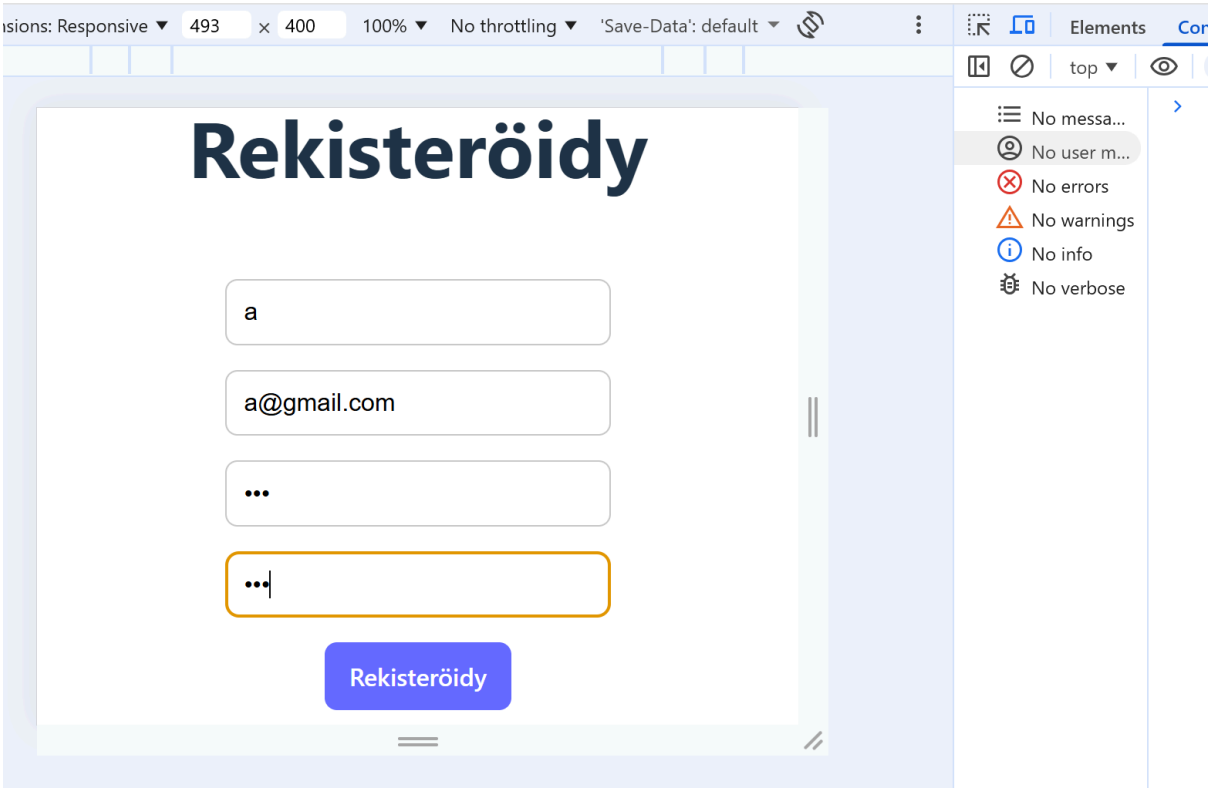
Luckily this can be fixed just by upgrading the cookie parser version higher than the range of vulnerable versions shown in the npm audit report.

```
Leone@LAPTOP-5OFSB5M1 MINGW64 ~/cyberhaava/my-noob-react-app (master)
$ npm install cookie-parser@latest

changed 2 packages, and audited 245 packages in 3s

55 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

```
Leone@LAPTOP-5OFSB5M1 MINGW64 ~/cyberhaava/my-noob-react-app (master)
$ npm list cookie-parser
my-noob-react-app@0.0.0 C:\Users\Leone\cyberhaava\my-noob-react-app
└── cookie-parser@1.4.7
```

7. A08 - Software Integrity Failures

Flaw: No rate limiting on login



Lets register a new user, with the username “a”, and a password 123

Then we open up the browser terminal and copy and paste onto it:

// Paste in console:

console.log(" Starting brute force attack on user 'a'...");

const passwords = [

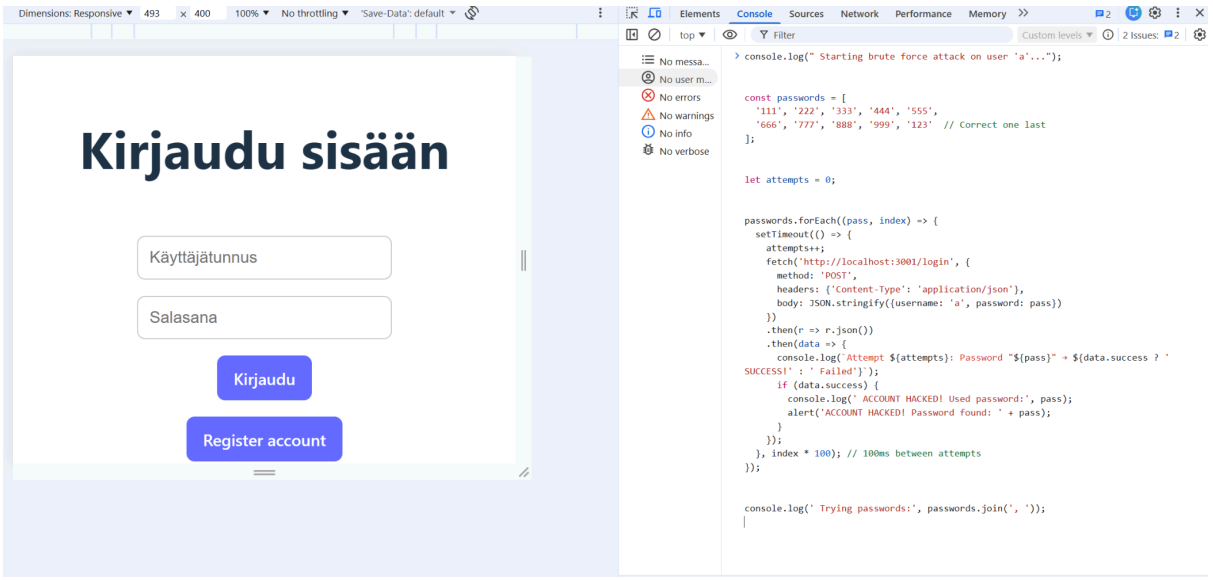
  '111', '222', '333', '444', '555',

  '666', '777', '888', '999', '123' // Correct one last

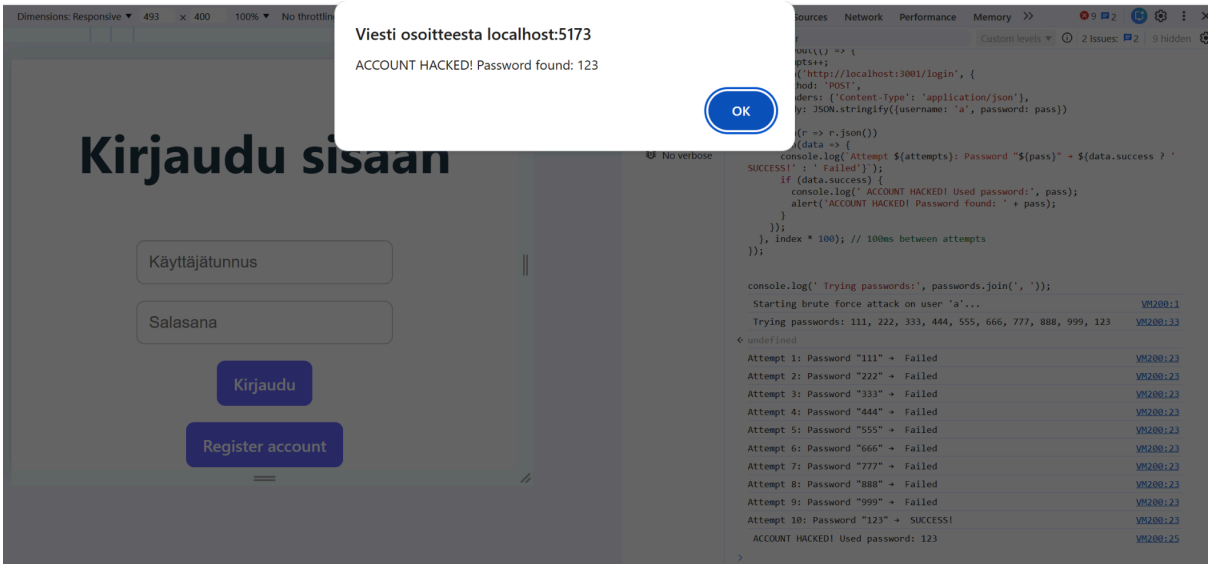
];

let attempts = 0;

```
passwords.forEach((pass, index) => {
  setTimeout(() => {
    attempts++;
    fetch('http://localhost:3001/login', {
      method: 'POST',
      headers: {'Content-Type': 'application/json'},
      body: JSON.stringify({username: 'a', password: pass})
    })
    .then(r => r.json())
    .then(data => {
      console.log(`Attempt ${attempts}: Password "${pass}" → ${data.success ? ' SUCCESS!' : ' Failed'}`);
      if (data.success) {
        console.log(' ACCOUNT HACKED! Used password:', pass);
        alert('ACCOUNT HACKED! Password found: ' + pass);
      }
    });
  }, index * 100); // 100ms between attempts
});

console.log(' Trying passwords:', passwords.join(', '));
```



And press enter.

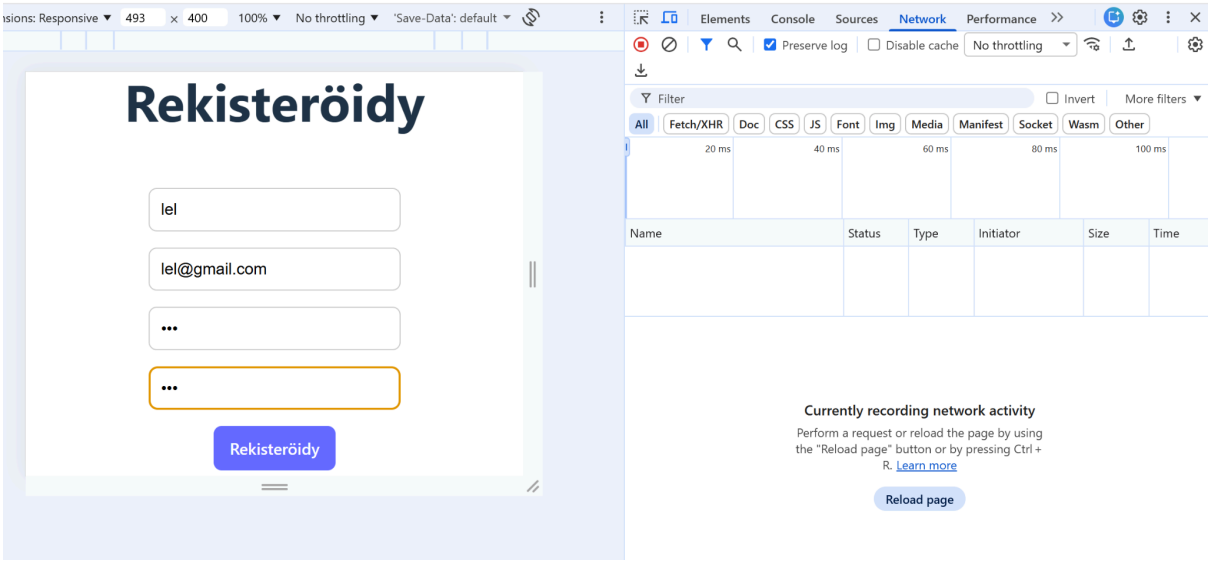


After the fixes:

(the blogs and previous users have been cleared for this
Starting with the

1. Fix for A02:2021 - Cryptographic Failures

Lets make a new user and login with it:



Hello world, this is application

Logged in as: **lel**

Create a post

would this post be NOT visible for others?

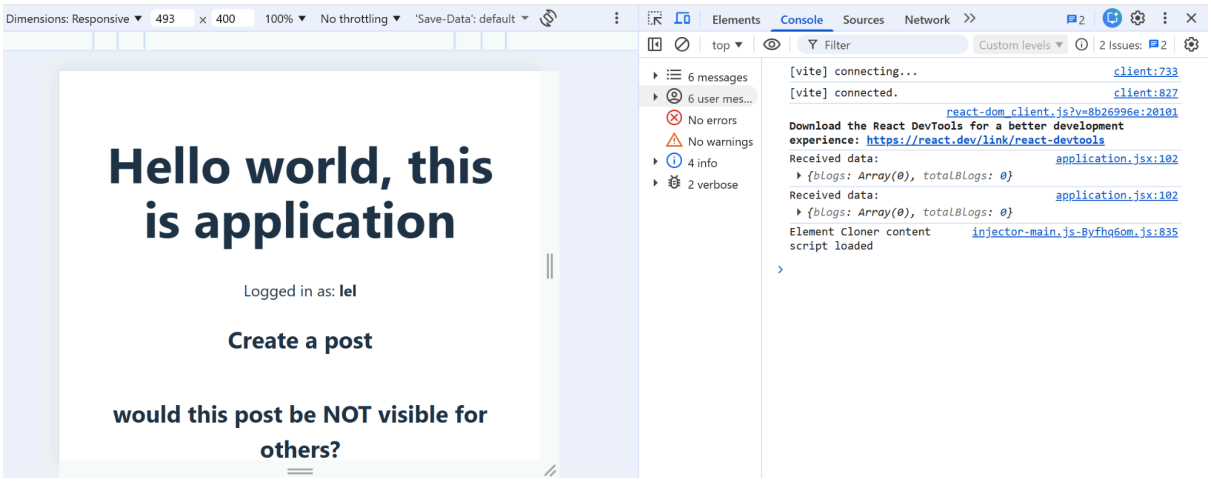
Yes? ☑

Title

input blog text here

Publish

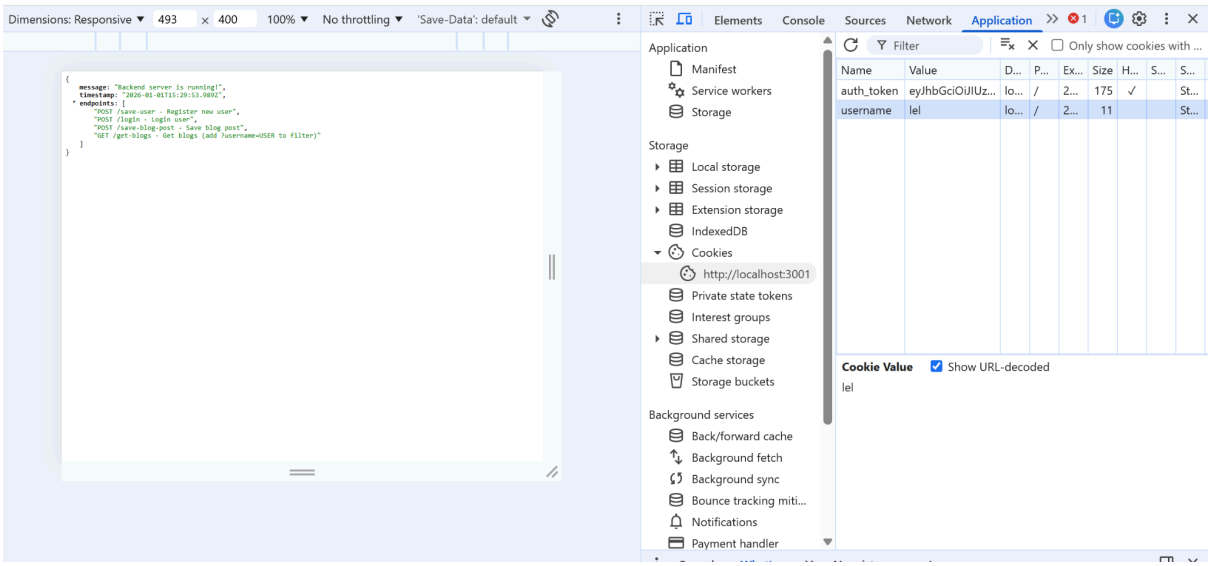
Logout



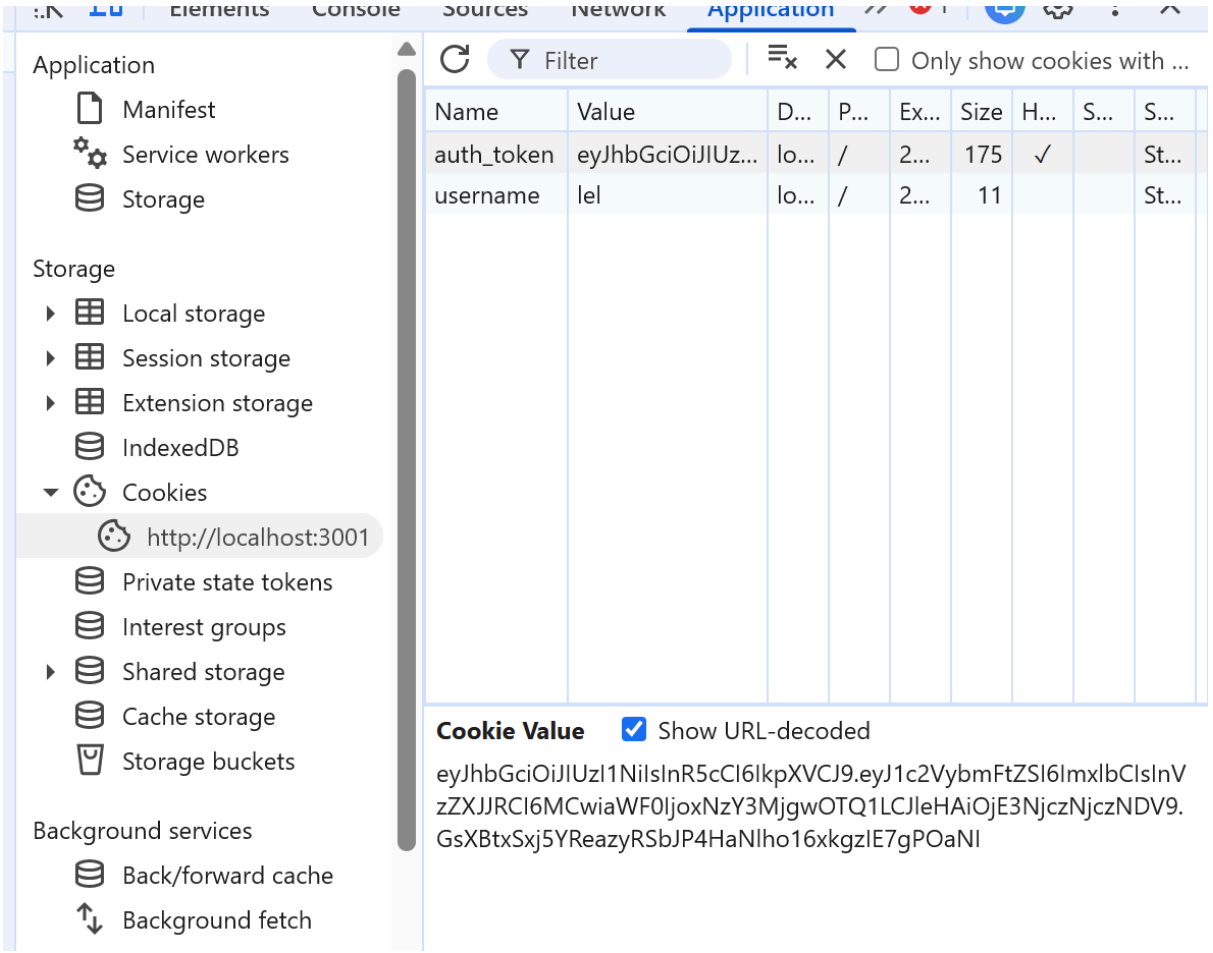
In here, simultaneously fix for: 2. A05:2021 - Security Misconfiguration

No more logs telling the password just like that.

Lets go to localhost 3001, where the server is.



And as we can see. The cookie value doesn't give anything useful to the cookie intermediary.

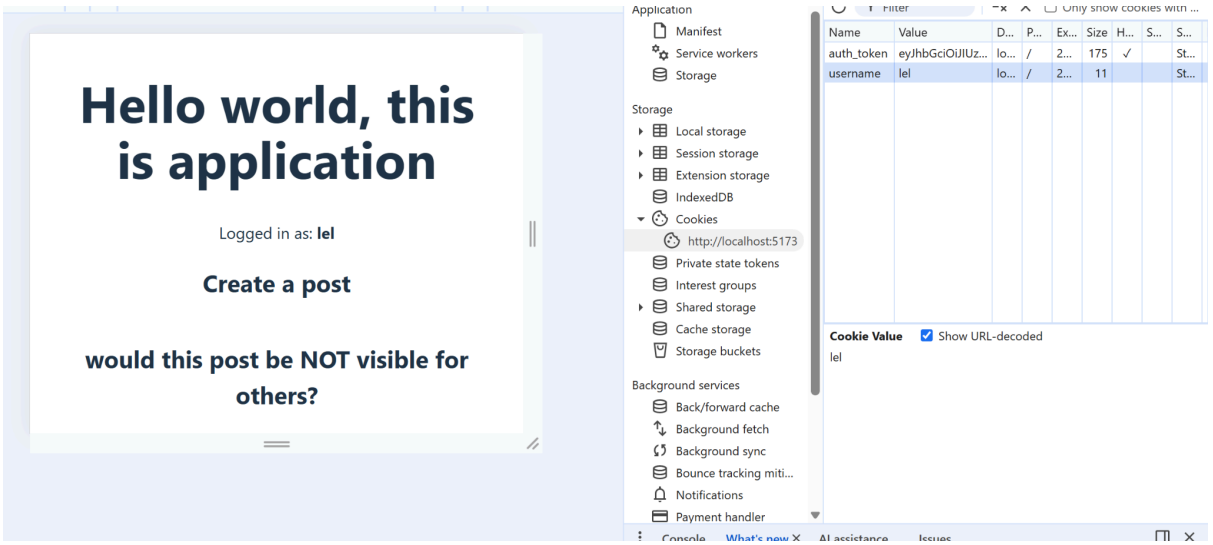


Even the auth token decoded is gibberish.

(previous problem of A02:2021 - Cryptographic Failures) fixed

Here is also a fix for: 3. A07:2021 - Identification and Authentication Failures

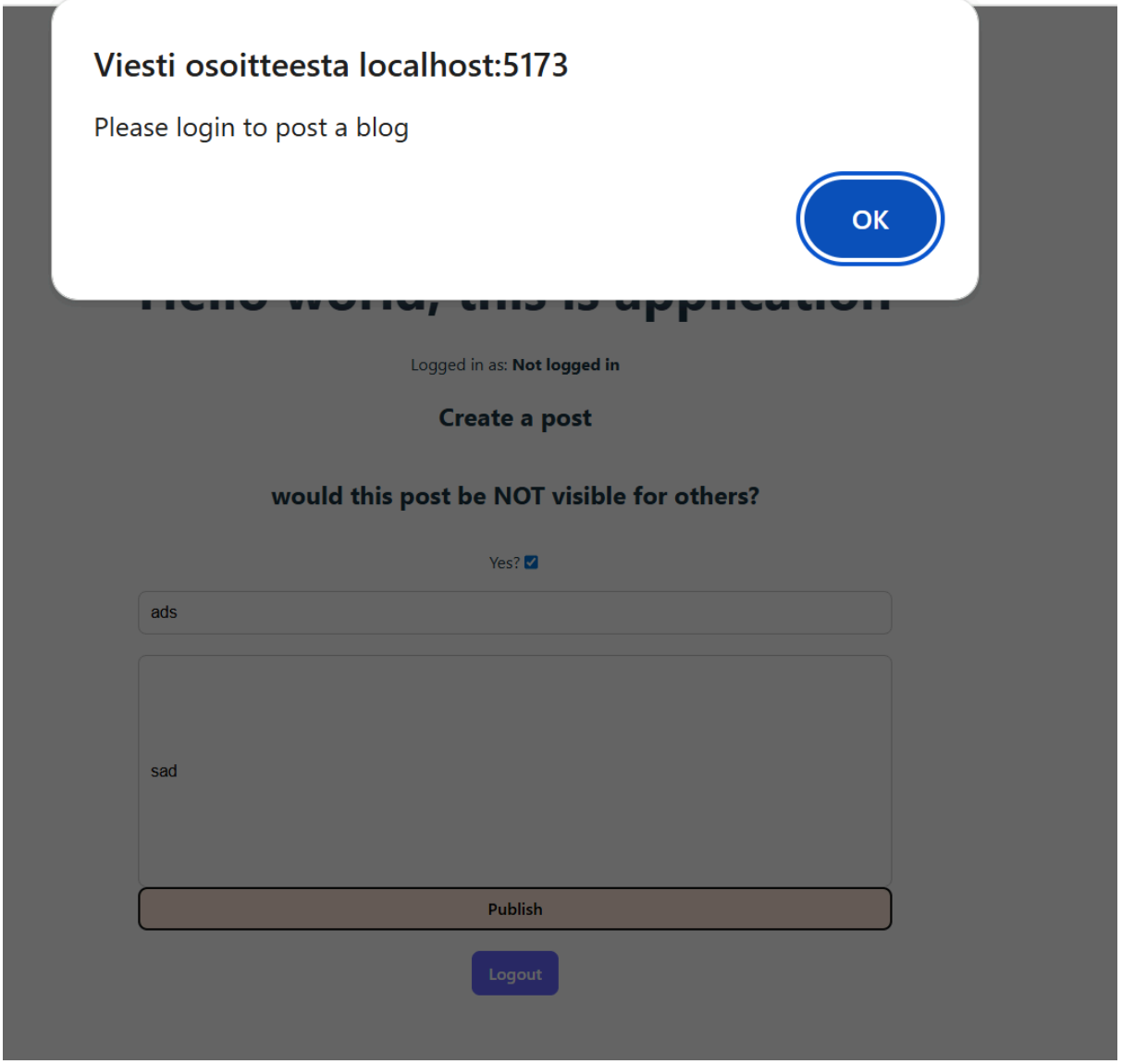
The same goes for trying to inspect cookies on the front end:



Since this specific exploit in my program required problems with the cookies to work

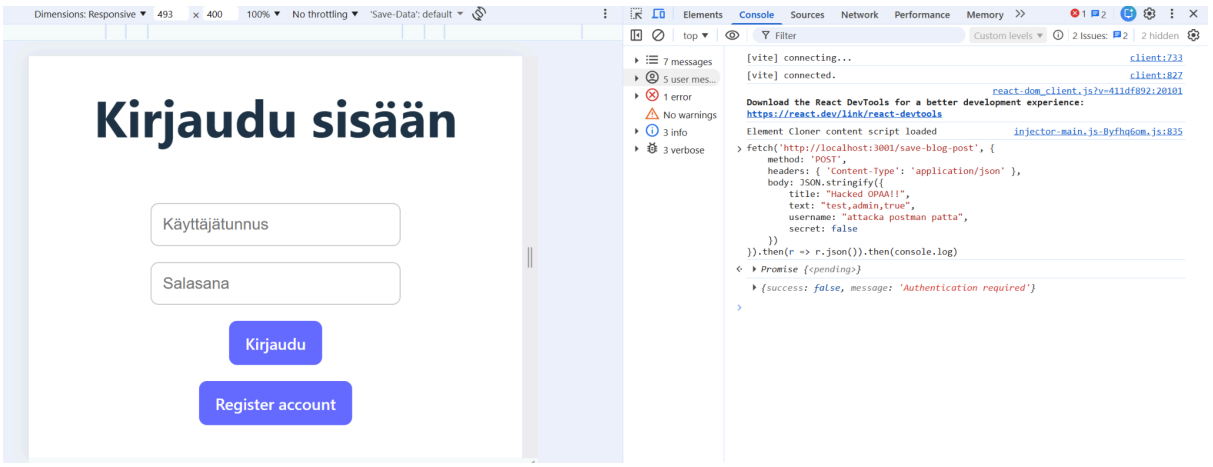
4. Fix for A01:2021 – Broken Access Control

You can no longer make posts as an anonymous user.

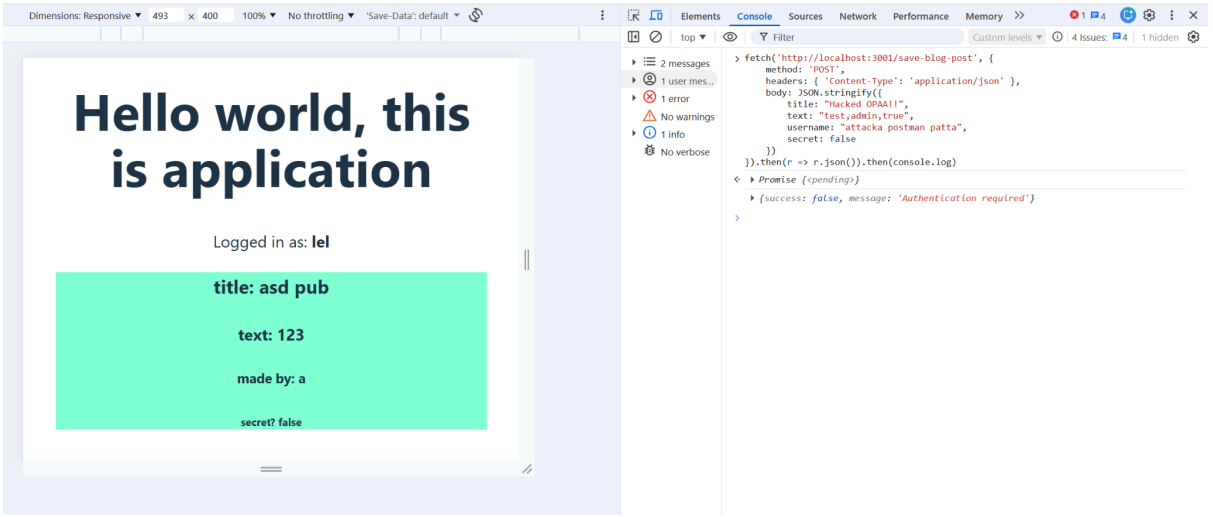


5. Fix for A01:2021 – Broken Access Control

Lets also try the previous post as anyone vulnerability:



And even if you were logged in:



It would not work.

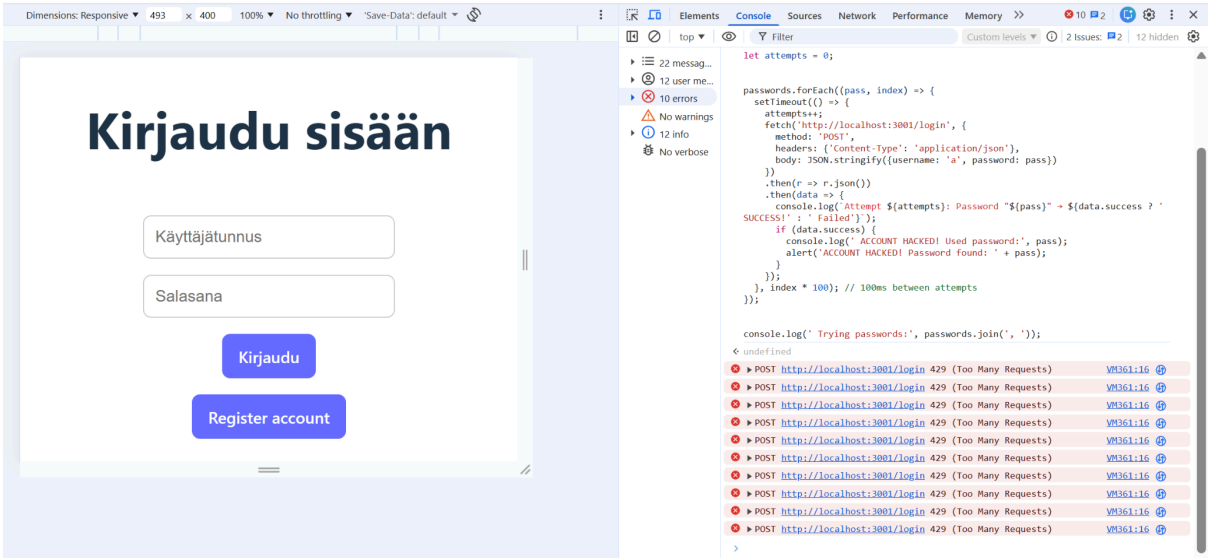
6. Fix for: A06:2021 - Vulnerable and Outdated Components

Luckily, this can be fixed just by upgrading the cookie parser version higher than the range of vulnerable versions shown in the npm audit report.

```
Leone@LAPTOP-5OFSB5M1 MINGW64 ~/cyberhaava/my-noob-react-app (master)
$ npm install cookie-parser@latest

changed 2 packages, and audited 245 packages in 3s

55 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

```
Leone@LAPTOP-5OFSB5M1 MINGW64 ~/cyberhaava/my-noob-react-app (master)
$ npm audit report
found 0 vulnerabilities

Leone@LAPTOP-5OFSB5M1 MINGW64 ~/cyberhaava/my-noob-react-app (master)
$ npm audit
found 0 vulnerabilities
```

7. Fix for A08 - Software Integrity Failures



The brute forcing no longer works, due to the rate limit being set on the login requests.