Experiment 5: Cross-Site Scripting (XSS) Attacks

Scenario:

The OWASP Juice Shop allows user-generated content. The security team suspects there is an XSS flaw that could lead to user session hijacking or credential theft.

Tasks:

- Inject a malicious JavaScript payload via a form or comment section using Burp Suite Community Edition or OWASP ZAP to intercept and modify requests.
- Demonstrate that the payload executes in a victim's browser (e.g., by producing an alert or stealing cookies).

Deliverable:

A screenshot of the XSS payload executing and a short explanation of the potential consequences.

XSS attacks:

A Cross-Site Scripting (XSS) attack happens when someone puts harmful content, like a message or code, into a website that allows user input, such as a search box or comment section. If the website does not check the input properly, the harmful content can run on other users' screens when they visit the page. This can be used to show fake messages, trick users, or even steal personal information like usernames or passwords. XSS is one of the most common and dangerous problems found in websites today.

INTRODUCTION TO BURP SUITE

Burp Suite is a tool used to test the security of websites. It helps users see and control the data that goes between their web browser and the website. This is useful for finding security problems like XSS or broken logins. Burp Suite has many features, but one of the most useful is the built-in browser and the intercept tool, which lets you pause and change requests before they go to the website. It's a popular tool used by ethical hackers and security testers.

OWASP Juice Shop

OWASP Juice Shop is a website made for learning about web security. It looks like a real online shopping site, but it's designed to have many security flaws on purpose. Users can try to find and test these weaknesses to understand how attacks like XSS, SQL injection, and others work. It is often used by students, security learners, and professionals to practice ethical hacking in a safe and legal environment.

OWASP Juice Shop Setup Instructions

Step 1: Install Docker

sudo apt install docker.io

Step 2: Start and Enable Docker

sudo systemctl start docker

sudo systemctl enable docker

Step 3: Pull

sudo docker pull bkimminich/juice-shop

Step 4: Run

sudo docker run --rm -p 3000:3000 bkimminich/juice-shop

Step 5: Access Juice Shop

http://localhost:3000

Step 6: Use Burp Suite's Built-in Browser

- In Burp Suite, go to "Proxy" and click on "Open Browser".
- In the browser, navigate to the OWASP Juice Shop URL:

  http://localhost:3000

Step 7: Locate an Input Field (e.g., Search Bar)

In OWASP Juice Shop, locate an input field where you can enter text. This can be:

- Search Bar: Located on the homepage.

Step 8: Inject Malicious XSS Payload

In the search bar

<img src=x onerror=alert('XSS')>

Or

<iframe src="javascript:alert('XSS 1')">

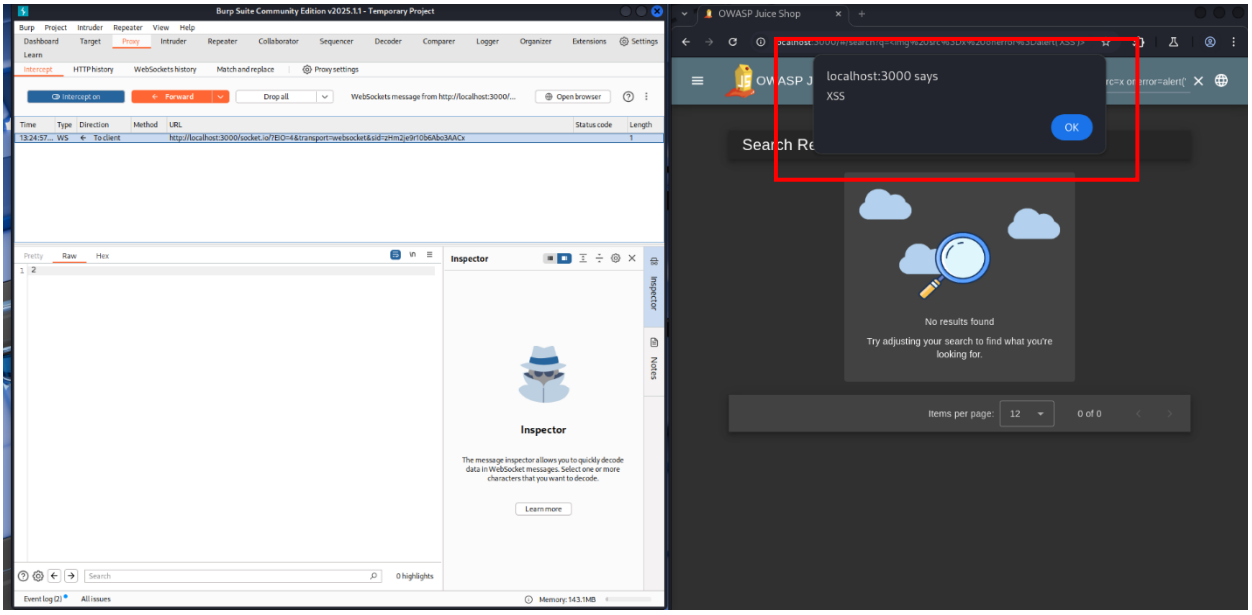
Step 9: Intercept

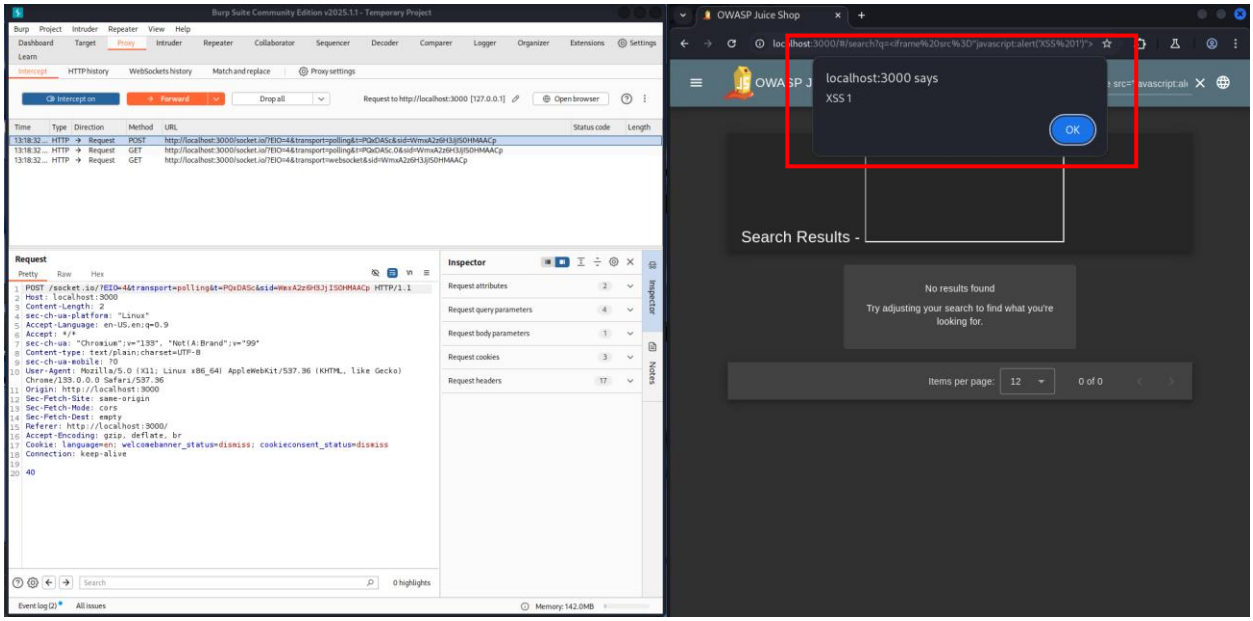
- Go back to Burp Suite and check if the request is intercepted.
- forward it to the server.

Step 10: XSS Popup Displayed in Browser

- After forwarding the request from Burp Suite, open the Juice Shop application in the browser.
- The malicious payload injected earlier will execute in the browser.

Output:


Burp Suite Community Edition v2025.1.1 - Temporary Project
Inspector
The message inspector allows you to quickly decode data in WebSocket messages. Select one or more characters that you want to decode.
Learn more
OWASP Juice Shop
localhost:3000 says
XSS
OK
No results found
Try adjusting your search to find what you're looking for.
Items per page: 12
0 of 0


Burp Suite Community Edition v2025.1.1 - Temporary Project
Request
Inspector
OWASP Juice Shop
localhost:3000 says
XSS 1
OK
Search Results -
No results found
Try adjusting your search to find what you're looking for.
Items per page: 12
0 of 0

Conclusion:

In this experiment, I successfully triggered an XSS (Cross-Site Scripting) attack by injecting harmful input, which was then executed in the browser. This shows that the application doesn’t properly check or sanitize user input, allowing attackers to inject malicious scripts. If not fixed, this could lead to serious security issues, such as stealing sensitive user data, hijacking user sessions, or performing unwanted actions without the user's knowledge. This experiment highlights the importance of proper input validation, output encoding, and security measures to protect users from XSS attacks.

****