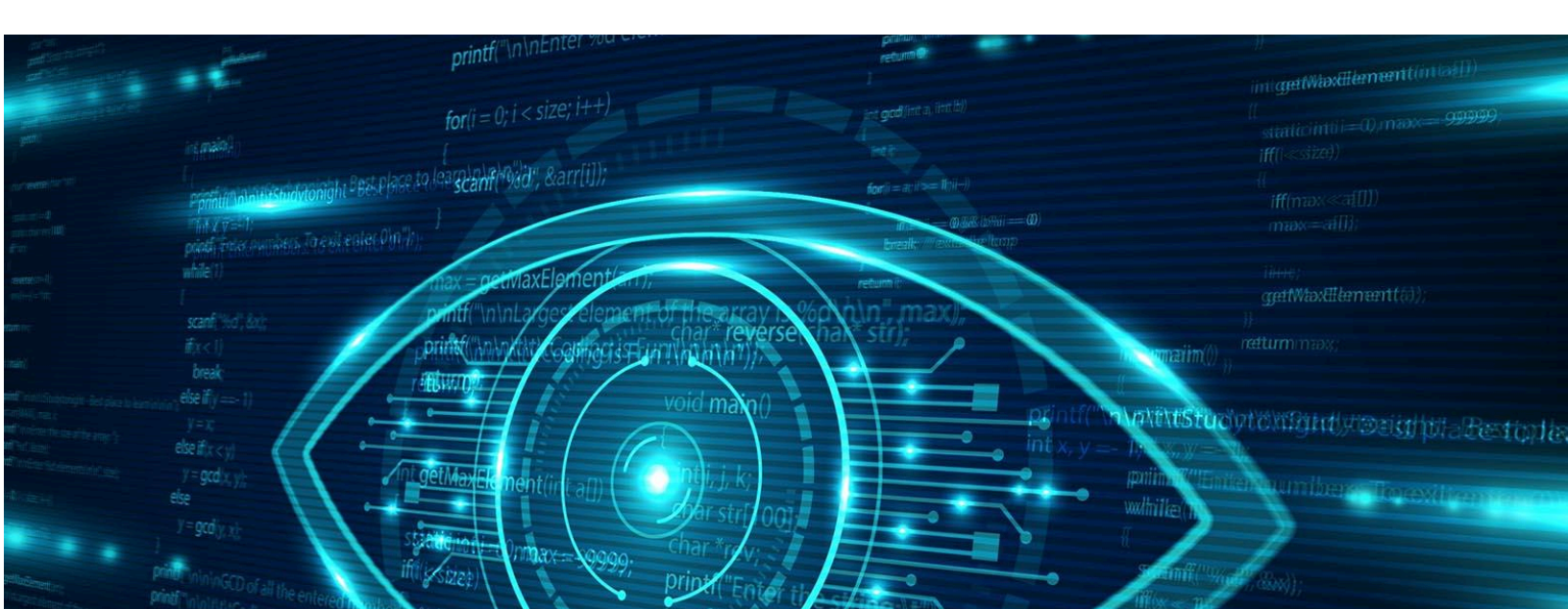


# Cyber Security

## Babakhanyan Tamara

# Mastering Web Vulnerabilities and network scanning

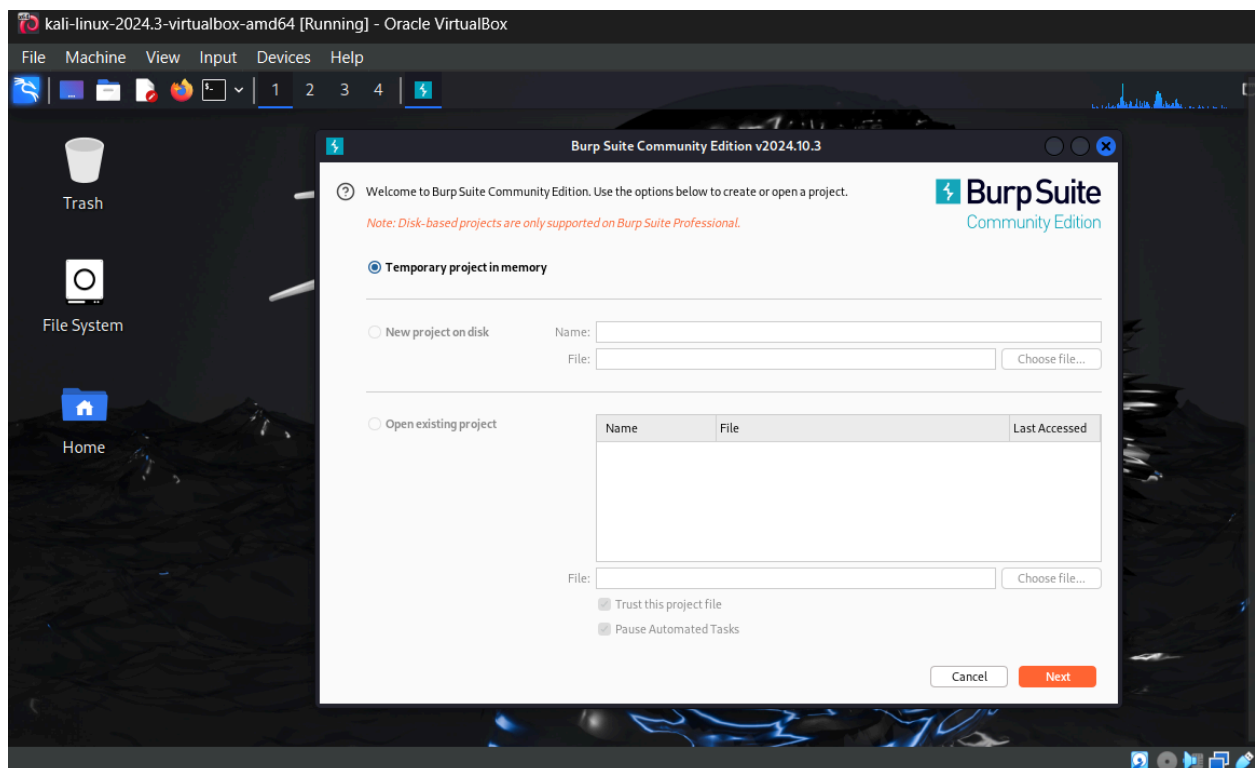


## Description

This project focuses on practical cybersecurity training by exploring web application vulnerabilities, writing professional reports, solving Capture The Flag (CTF) challenges, and mastering network scanning. It includes hands-on tasks with OWASP Juice Shop and TryHackMe's "Root Me" CTF, enhancing both offensive and defensive skills. Additionally, it incorporates developing a custom Python-based Nmap tool, fostering scripting and network analysis expertise.

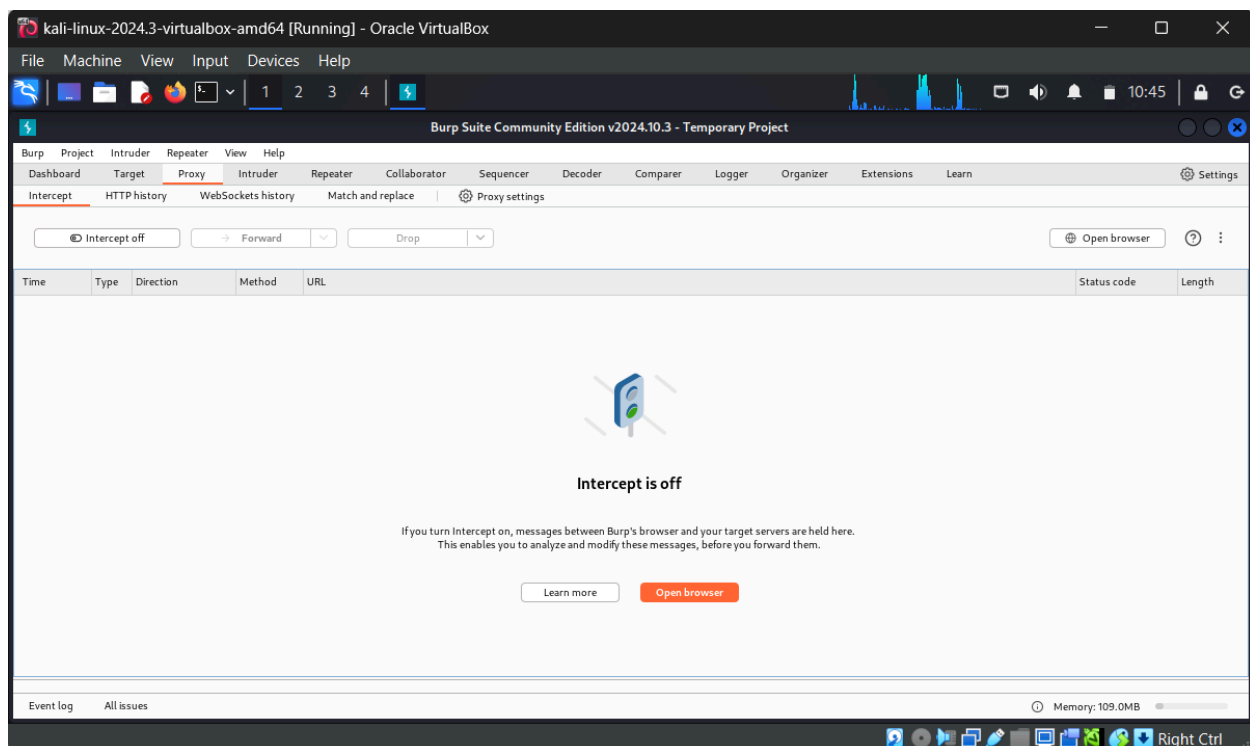
In this document, I will examine and analyze five critical security vulnerabilities within the OWASP Juice Shop application, a deliberately insecure web application designed for security testing and learning. The objective of this exercise is to uncover these vulnerabilities and understand their implications on web application security. To achieve this, I will be using Burp Suite, a comprehensive and widely recognized web application security testing tool created by PortSwigger. Burp Suite is an essential resource for penetration testers and security professionals, offering a range of features such as intercepting HTTP/S requests, scanning for vulnerabilities, and performing both automated and manual security testing.

For this report, Burp Suite has been installed and configured within a Kali Linux environment, which will serve as the platform for testing. By acting as an intermediary proxy between the browser and the OWASP Juice Shop application, Burp Suite enables me to intercept, analyze, and manipulate the traffic between the client and server. This allows me to identify and exploit potential security weaknesses in the application, simulating real-world attack scenarios. The findings from these tests will be detailed in the following sections, alongside recommendations for mitigating the identified vulnerabilities and securing the web application.



To begin the process, I first navigate to the **Proxy** tab in Burp Suite. With the **Intercept** feature turned off, I open the browser configured to work with Burp Suite and access the OWASP Juice Shop application. This setup allows me to start analyzing the web traffic between the browser and the application without manually intercepting each request.

The OWASP Juice Shop is a purposely vulnerable web application developed by the OWASP community for the sole purpose of security testing and training. It is designed to simulate real-world security issues that developers, penetration testers, and security professionals are likely to encounter. As an educational tool, Juice Shop mirrors the complexities of common vulnerabilities such as Cross-Site Scripting (XSS), SQL Injection, Broken Authentication, and more, while aligning with the OWASP Top 10 security risks. It provides an interactive platform for users to practice identifying and exploiting these weaknesses in a controlled and safe environment.



---

## 1. IDOR (Insecure Direct Object Reference)

The first vulnerability identified in the OWASP Juice Shop application is **Insecure Direct Object Reference (IDOR)**. This vulnerability occurs when an application allows users to access or modify objects (such as files, records, or user-specific data) directly through user-controlled input, such as URLs or HTTP requests, without proper access control checks. As a result, an attacker can manipulate these inputs to access objects they are not authorized to view or modify.

In the case of OWASP Juice Shop, I begin by inspecting the application's traffic to identify any potential instances where user input, such as session IDs or object identifiers, might be exposed. Using Burp Suite's proxy feature, I capture and examine requests made between the browser and the server. I pay particular attention to URL parameters, headers, or any other data passed in the requests that could reference internal objects or user-specific resources.

For instance, a common scenario for IDOR could involve accessing another user's shopping basket or profile information by altering the object reference in the URL or request body. If the application does not properly verify the user's authorization to access these objects, an attacker could exploit the vulnerability to access, modify, or delete data belonging to other users.

### Example:

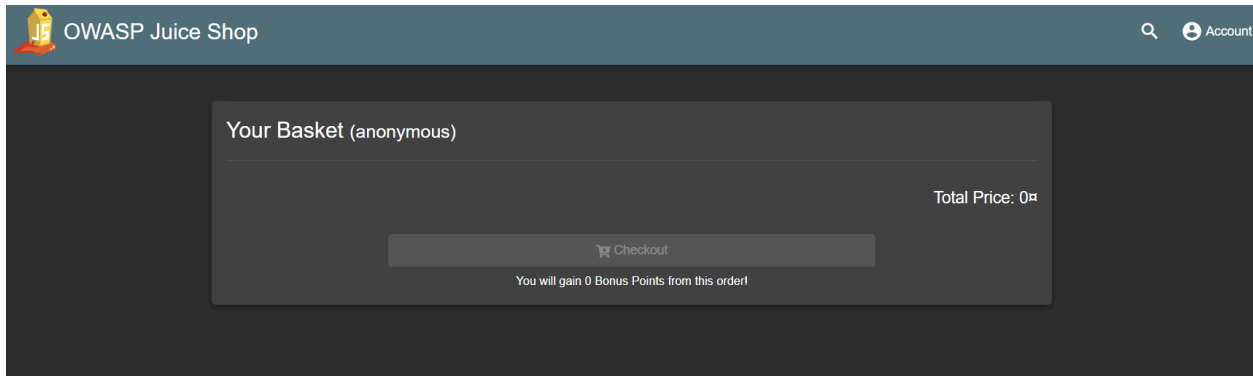
Consider a URL like:

<https://juice-shop.herokuapp.com/#/basket?itemId=12345>

If the application does not verify whether the user is authorized to access the item with `itemId=12345`, an attacker could modify the `itemId` to that of another user's item (e.g., `itemId=54321`) and gain unauthorized access to that user's basket.

### Impact:

The potential impact of IDOR in the Juice Shop application could be severe, as it may allow attackers to view or manipulate sensitive user data, compromise user privacy, or interfere with the integrity of the application's functionality.

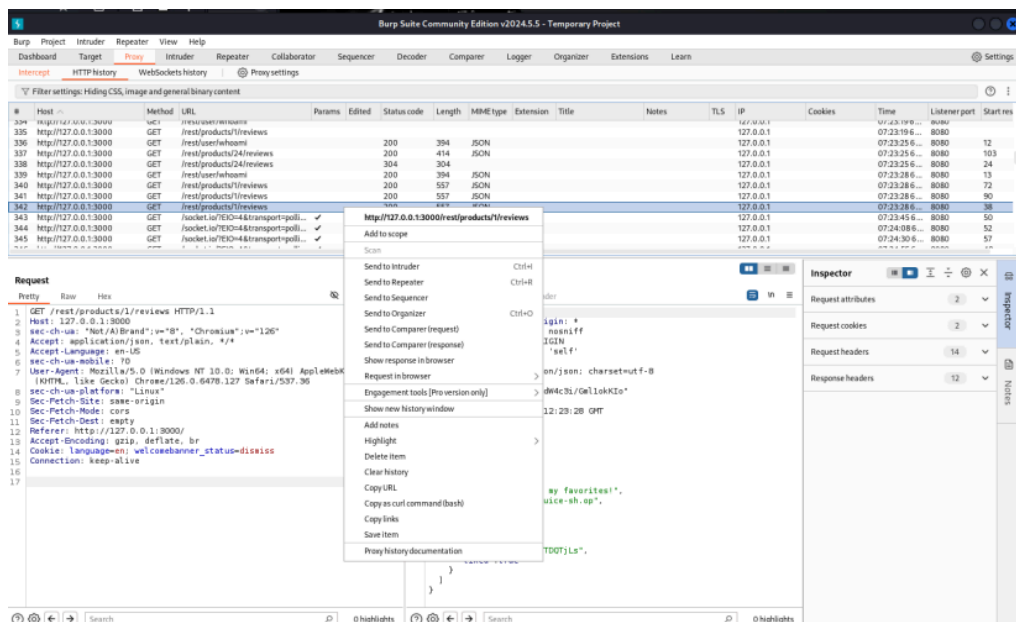


1."I begin by viewing the first product in the OWASP Juice Shop application, then enable the Intercept feature in Burp Suite to capture HTTP requests."

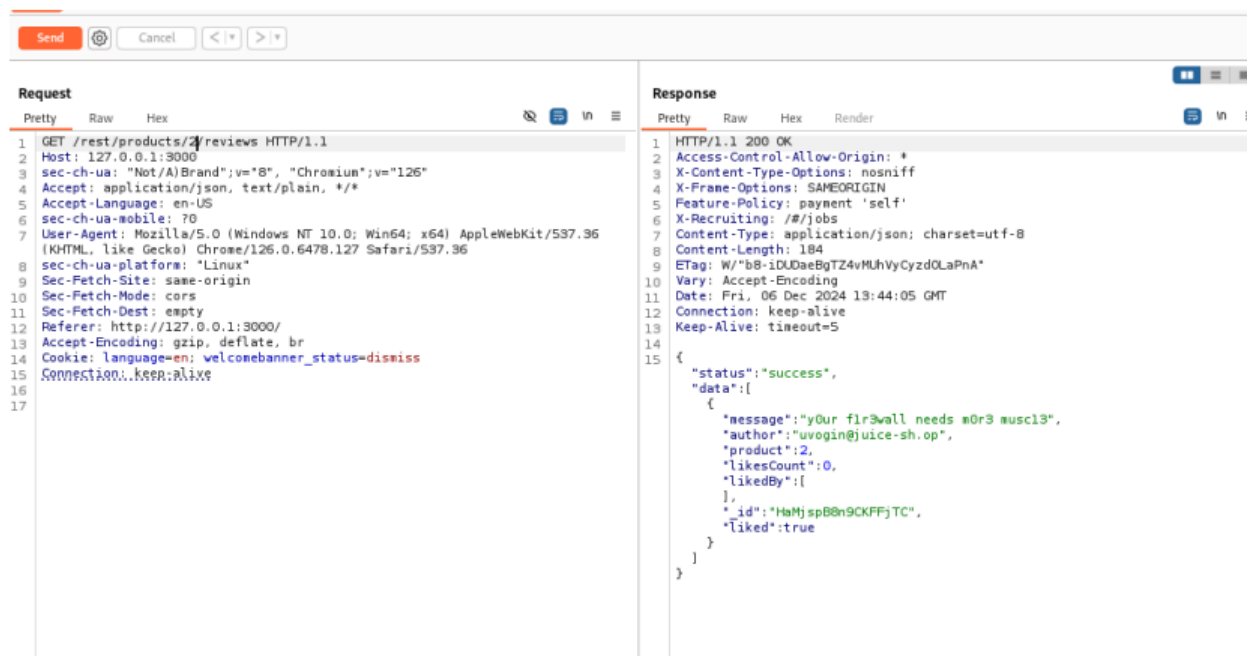
2."Once Intercept is turned on, I navigate to the HTTP History tab in Burp Suite to locate the specific request associated with the first product."

3."After finding the relevant HTTP request, I right-click on it and select 'Send to Repeater' to send the request for further analysis."

4."In the Repeater tab, I modify the HTTP request parameters to test for potential vulnerabilities, such as Insecure Direct Object Reference (IDOR) or Cross-Site Scripting (XSS)."



In the Repeater tab, I modify the URL by changing the product ID from '1' to '2'. After making this change, I click on 'Send' to resend the altered request. The server responds positively, indicating that the operation was successful and executed without any access control restrictions.



I received an HTTP 200 OK response, confirming the modification was successful. This indicates the presence of an IDOR (Insecure Direct Object Reference) vulnerability.

IDOR allows unauthorized users to access or modify resources by changing resource IDs in the URL without proper authorization checks. This flaw can lead to unauthorized access to sensitive data.



---

## 2. XSS (Cross-Site Scripting)

Another instance of XSS (Cross-Site Scripting) in the OWASP Juice Shop application (<https://juice-shop.herokuapp.com/#/>) was identified in the feedback form.

To test this, I submitted the following payload in the feedback comment field:

```
php
```

[Copy code](#)

```
<script>console.log('XSS vulnerability detected');</script>
```

After submitting the form, I opened the **Console** tab in the browser's developer tools. The message 'XSS vulnerability detected' was logged in the console, confirming that the input was executed as JavaScript. This indicates that the application does not sanitize or validate user input properly, allowing malicious scripts to be executed in the user's browser. Attackers can exploit this vulnerability to execute arbitrary commands in the context of the user's session.

### XSS (Cross-Site Scripting)

The next vulnerability I identified is **XSS (Cross-Site Scripting)** in the **OWASP Juice Shop** application, accessible at <https://juice-shop.herokuapp.com/#/>. To test for this, I right-clicked on the webpage, selected **Inspect** to open the developer tools, and navigated to the **Console** tab. I then entered the JavaScript command `alert('XSS')`. Upon execution, a pop-up appeared with 'XSS', confirming the vulnerability. This demonstrates that the application fails to properly sanitize or validate user input, allowing the injected script to execute in the browser.

# Guide to Exploit SQL Injection Using Burp Suite

## 1. Set Up Burp Suite Proxy

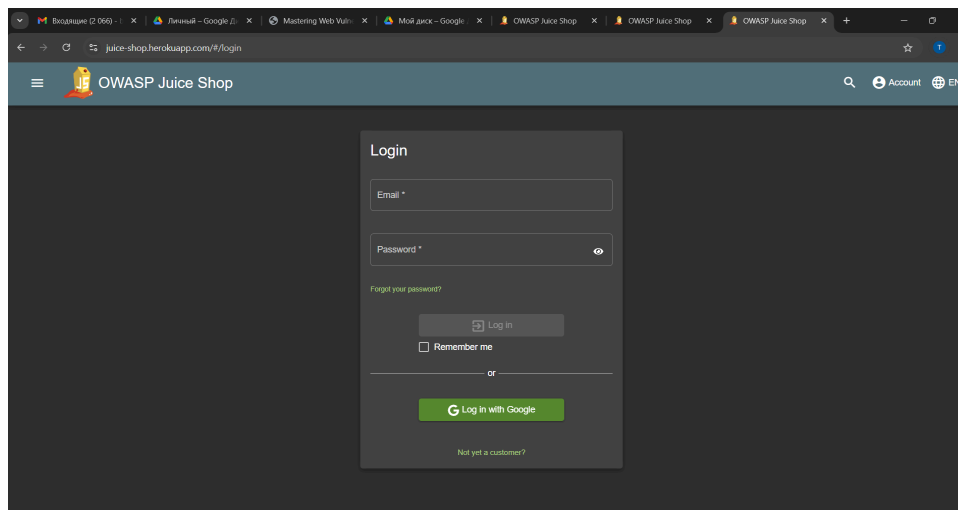
- **Configure your browser** to route traffic through Burp Suite's proxy. By default, Burp Suite listens on `127.0.0.1:8080`. In your browser settings, set the proxy to this address.
- In **Burp Suite**, go to the **Proxy** tab and enable **Intercept**. This allows Burp Suite to capture HTTP requests and modify them before they reach the server.

## 2. Locate the Product Comment Section

- Navigate to the **OWASP Juice Shop** product page, and find the **comment section** where users can submit feedback.
- Submit a comment (such as a test comment) on a product. You should observe an HTTP request being sent when the form is submitted.

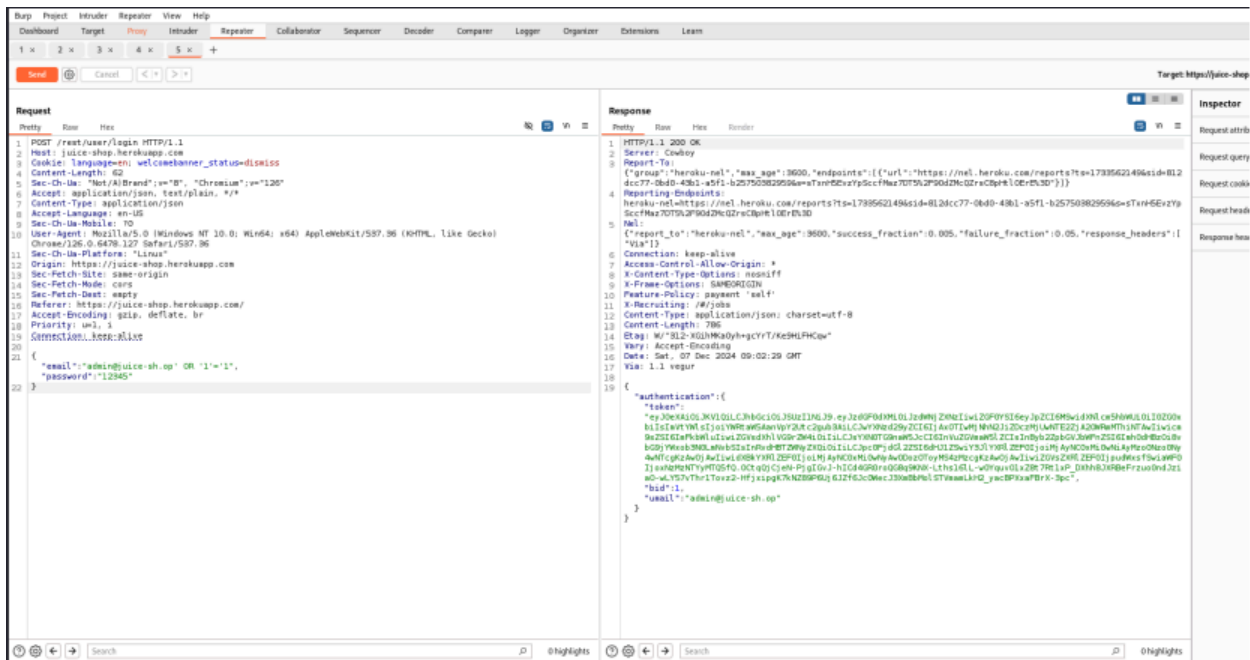
## 3. Capture the HTTP Request

- Burp Suite will capture the HTTP request related to the comment submission.
- Go to the **Proxy** → **Intercept** tab to see the captured request. You should see the POST request sent to the server, and it will include parameters such as `email` and `comment`.



Here, the `email` and `comment` fields are user inputs, which are potentially vulnerable to SQL injection if they are not properly sanitized.





To exploit **Insecure Direct Object Reference (IDOR)** in the **Juice Shop** application, the idea is to manipulate a URL or request parameter that references an object (such as a shopping basket), with the goal of accessing or modifying data belonging to other users. This can happen when the application doesn't properly verify the user's authorization before granting access to such resources.

## Steps for Exploiting IDOR in the Shopping Basket

### 1. Log into Juice Shop

- Visit the Juice Shop login page: Juice Shop Login.
- Log in with your user credentials, or if you're testing as an attacker, you might use any valid user credentials or even the default ones if you're testing in a controlled environment.

## 2. Identify the Basket URL

Once logged in, go to your shopping basket.  
<https://juice-shop.herokuapp.com/#/basket/12345>

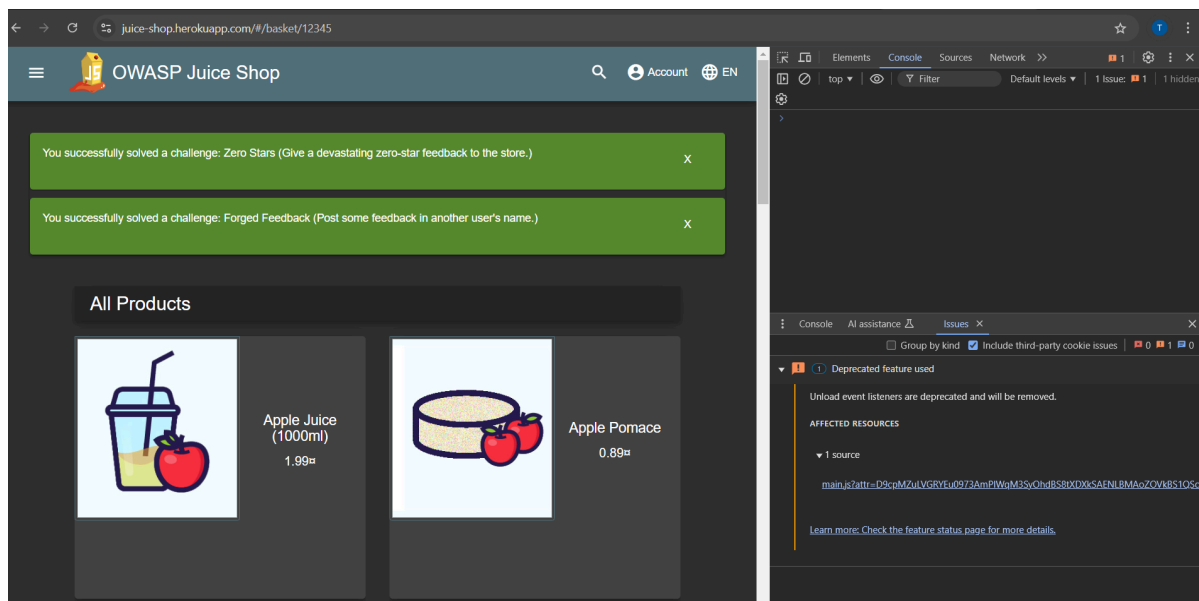
Here, **12345** is a unique identifier for your shopping basket. The parameter in the URL, such as **basket\_id**, is what we're interested in, as it's being used to reference the shopping basket.

## 3. Manipulate the Basket ID

Once you have the shopping basket ID from your own basket, try to change the **basket\_id** parameter in the URL to different values and see if you can access another user's shopping basket.

- If your basket is at **basket/12345**, try changing it to:
  - **basket/12346**
  - **basket/12347**
  - Or any other number

If you are able to access someone else's basket, that means the application is vulnerable to **Insecure Direct Object Reference (IDOR)** because it is not properly validating user access rights to the resources.



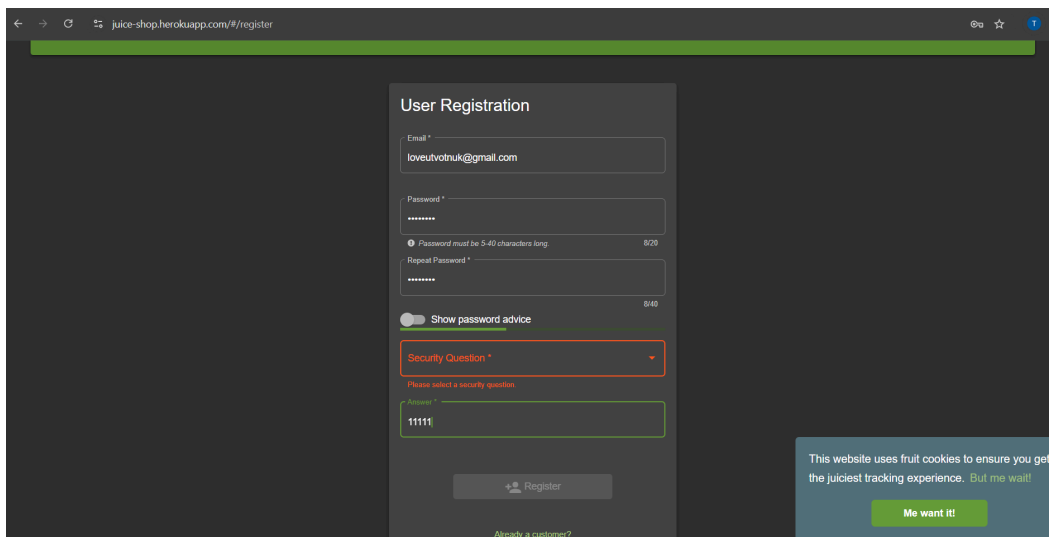
---

**Broken Access Control** is a type of vulnerability where the application doesn't correctly enforce restrictions on what authenticated or unauthorized users are allowed to do. This can lead to unauthorized access to sensitive resources, modification of data, or actions that should only be permitted to specific users.

<https://juice-shop.herokuapp.com/#/>

## Exploiting Broken Access Control in Juice Shop by Manipulating User Reviews

In **Juice Shop**, users can write reviews for products. If **Broken Access Control** exists, it might allow you to manipulate or delete reviews of other users or perform actions that are meant to be restricted to authorized users (such as admin).



The screenshot shows the 'User Registration' form in the Juice Shop application. The form is centered on a dark background. It includes fields for 'Email \*' (containing 'loveutvotruk@gmail.com'), 'Password \*' (masked with dots), 'Repeat Password \*' (also masked), and a 'Security Question \*' dropdown menu. Below the password fields is a 'Show password advice' toggle. The 'Answer' field for the security question contains '11111'. At the bottom of the form is a 'Register' button. A link 'Already a customer?' is located below the register button. On the right side of the form, there is a cookie consent banner that reads: 'This website uses fruit cookies to ensure you get the juiciest tracking experience. But me wai!'. A 'Me want it!' button is at the bottom of the banner.

### 1. Register a New User

- Go to the **Juice Shop registration page**: Juice Shop Register.
- Register as a new user by providing your details (username, email, password).
- After successfully registering, you will be logged in as a regular user.

---

## 2. Submit a Product Review

Once you're logged in as a regular user, select a product and leave a review:

- Browse the available products and select one.
- In the product page, you should find an option to write a review.
- Submit a review with a rating and a comment (e.g., "Great product!").

After submitting, the review will be associated with your user account, and it might look something like this:

Product: Juice

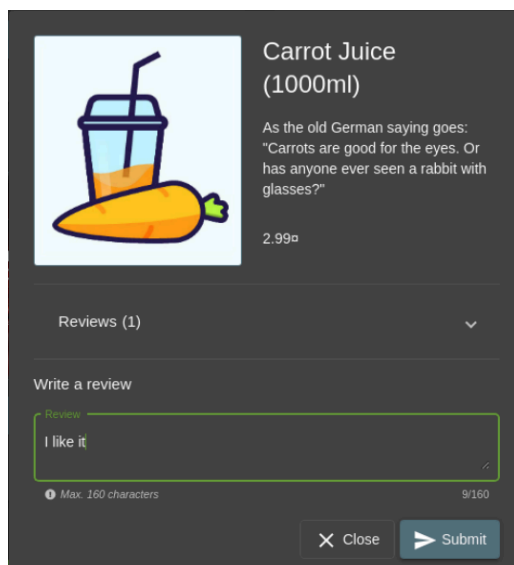
Review: "Great product!"

User: [Myuser]

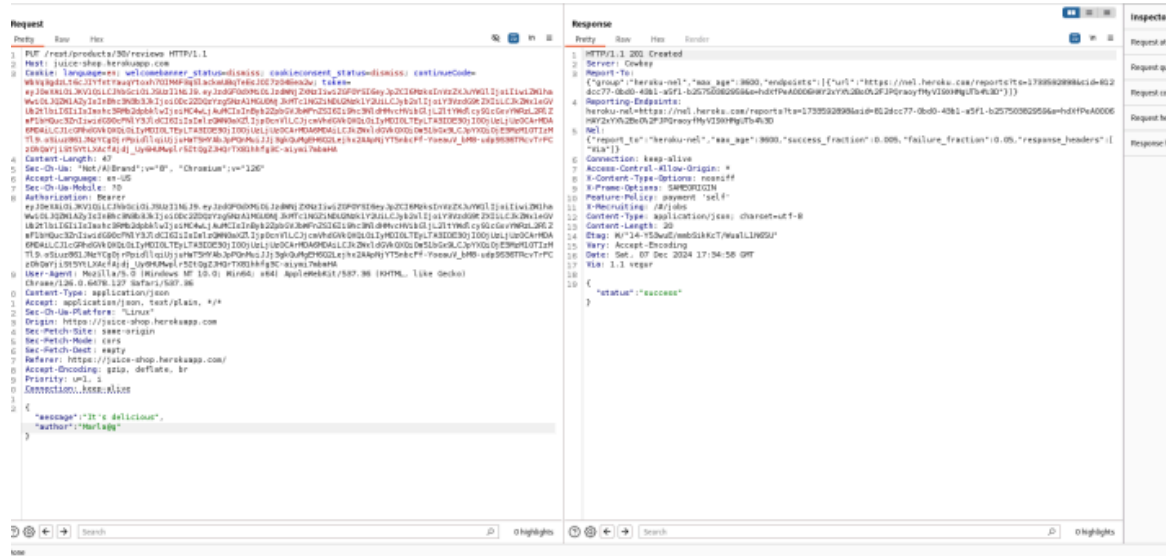
## 3. Inspect the Review Request

To check for **Broken Access Control**, you'll want to inspect the request made when submitting or viewing a review. You can do this using a tool like **Burp Suite** or **browser developer tools**.

- Open the **Developer Tools** in your browser (usually accessible with **F12**).
- Go to the **Network** tab and submit a review. Look for the HTTP request that was sent to the server.







### 3. Forward Modified Request:

- ```
o Forward the modified request to the server and observe if the
  review is updated or altered for another user's product.
```

#### 4. Verify Exploitation:

- o Check the review page to see if the unauthorized review modification or deletion was successful. This confirms **Broken Access Control**.

---

## Conclusion

This project has provided invaluable hands-on experience in cybersecurity by exploring web application vulnerabilities, network scanning techniques, and exploitation challenges. Throughout the task, I used a variety of tools and methodologies to complete key components of the project, which have significantly enhanced my offensive and defensive cybersecurity skills.

### Tools and Techniques Used:

- I deployed and tested **OWASP Juice Shop** locally and in the cloud to identify and exploit vulnerabilities such as **SQL Injection**, **Cross-Site Scripting (XSS)**, **Broken Access Control**, and **Insecure Direct Object References (IDOR)**. Tools like **Burp Suite** and **browser developer tools** were instrumental in intercepting and analyzing requests, enabling me to identify weaknesses in the application.
- I successfully utilized **Nmap** for network scanning and vulnerability detection on various target systems, including the **Metasploitable machine** and **TryHackMe's "Root Me" CTF challenge**. This allowed me to identify open ports, running services, and potential vulnerabilities that I could exploit.
- I also developed a **custom Python-based Nmap scanner** that replicates basic port scanning functionality, giving me a deeper understanding of scripting and network analysis.