# This is a writeup for OWASP Juice Shop

- CTF Name: Oswap Juice

- CTF Level: Easy

- Date: 7/1/2025

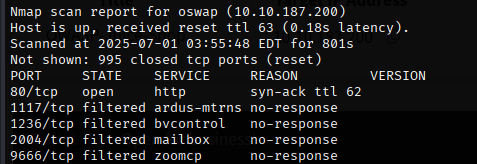
- Platform: THM CTF

- Category: Jeopardy style

- IP: here I used a host called “oswap”

- CTF Description:This room uses the Juice Shop vulnerable web application to learn how to identify and exploit common web application vulnerabilities.

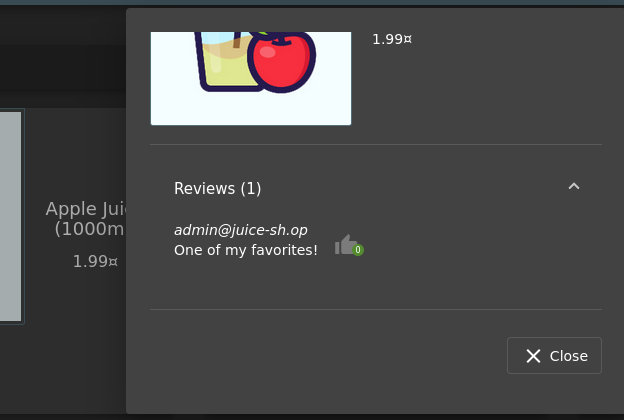
Nmap result: nmap -sV oswap -vvv -o nmap.txt



80/tcp open http

**#1 Question #1: What’s the Administrator’s email address?**

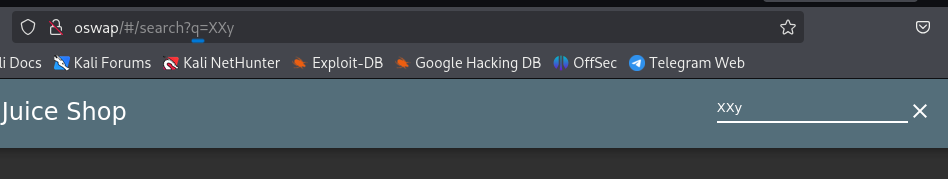
**Let’s** visit website and by clicking on any product, we can find out the admin’s email address.



**ans :** [admin@juice-sh.op](mailto:admin@juice-sh.op)

**#2 Question #2: What parameter is used for searching?**

Click on the magnifying glass in the top right of the application will pop out a search bar and begin inputting any product names. On our URL, there will be updates on the parameter of the request URL immediately after we enter our inputted strings into it.



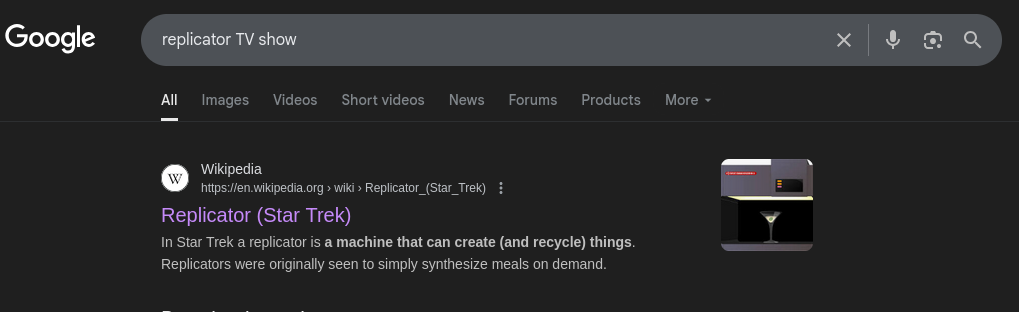
Ans: q

**#3 Question #3: What show does Jim reference in his review?**

Jim did a review on the Green Smoothie product. We can see that he mentions a replicator.



If we google "**replicator**" we will get the results indicating that it is from a TV show called Star Trek



Ans: Star Trek

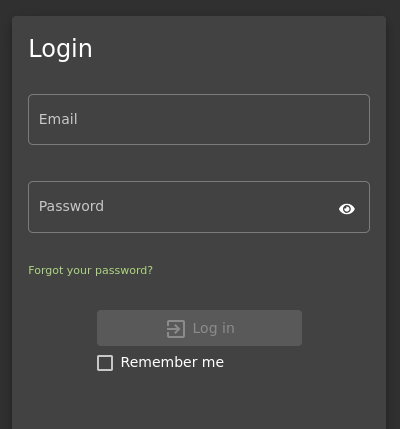
**Task 3 : Inject the juice**

This task will be focusing on injection vulnerabilities. Injection vulnerabilities are quite dangerous to a company as they can potentially cause downtime and/or loss of data. Identifying injection points within a web application is usually quite simple, as most of them will return an error. There are many types of injection attacks, some of them are:

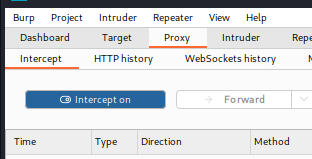
**SQL Injection :** SQL Injection is when an attacker enters a malicious or malformed query to either retrieve or tamper data from a database. And in some cases, log into accounts.  
**Command Injection** : Command Injection is when web applications take input or user-controlled data and run them as system commands. An attacker may tamper with this data to execute their own system commands. This can be seen in applications that perform misconfigured ping tests.   
**Email Injection :** Email injection is a security vulnerability that allows malicious users to send email messages without prior authorization by the email server. These occur when the attacker adds extra data to fields, which are not interpreted by the server correctly.

**Question #1: Log into the administrator account!**

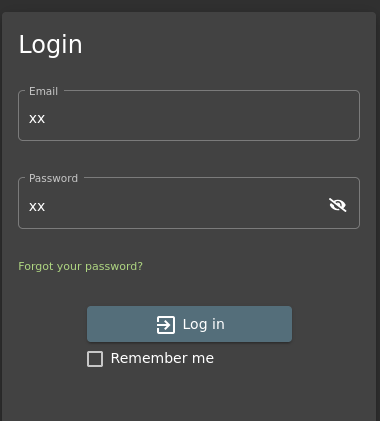
We have to open burp-suite. We need to open the foxy proxy.



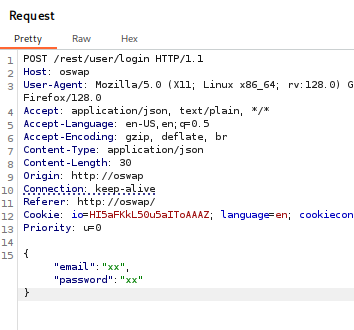
Intercept is on:



Enter email and password something:



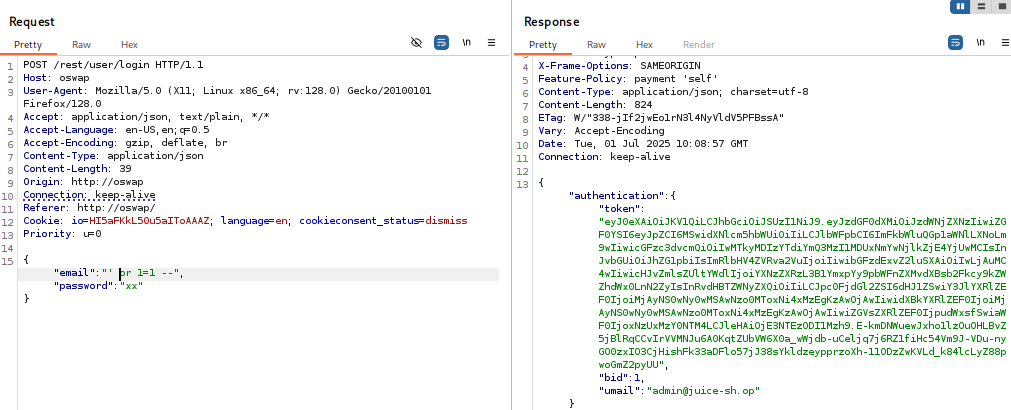
You Will get:

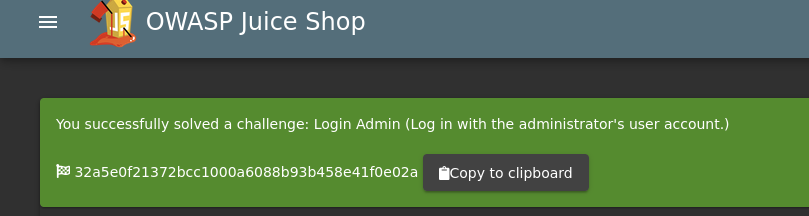


Send to repeater and then, open repeater.

We can focus email:

We can write “ ‘ or 1=1-- “

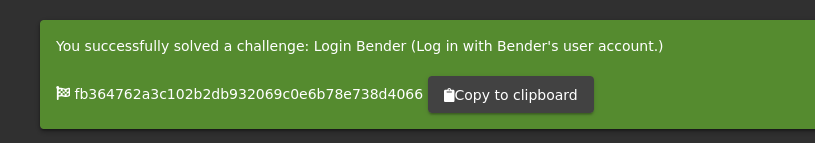


Successful!! 

Ans: 32a5e0f21372bcc1000a6088b93b458e41f0e02a

**#2 Question #2: Log into the Bender account!**

Similar to what we did in **Question #1**, we will now log into Bender's account! Capture the login request again, but this time we will put: **bender@juice-sh.op'--**as the email.



Ans: fb364762a3c102b2db932069c0e6b78e738d4066

**Task 4 : Who broke my lock?!**

In this task, we will look at exploiting authentication through different flaws. When talking aboutflaws within authentication, we include mechanisms that are vulnerable to manipulation. These mechanisms, listed below, are what we will be exploiting.

Weak passwords in high privileged accounts

Forgotten password pages

**#1 Question #1: Bruteforce the Administrator account’s password!**

For the payload, we will be using the **best1050.txt from Seclists**. (Which can be installed via: **apt-get install seclists**)

*You can load the list from: /usr/share/wordlists/SecLists/Passwords/Common-Credentials/best1050.txt*

Once the file is loaded into Burp, start the attack. You will want to filter for the request by status.

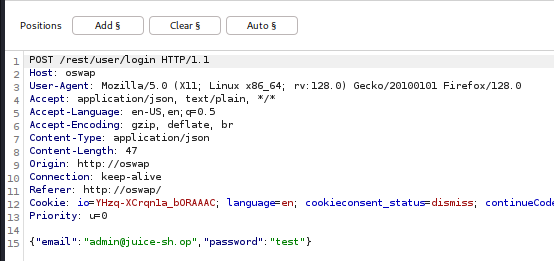
A **failed**request will receive a **401 Unauthorized**   https://i.imgur.com/HcUs6eW.png

Whereas a **successful**request will return a **200 OK**. https://i.imgur.com/q5jcfIA.png

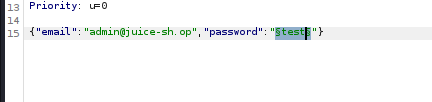
Once completed, login to the account with the password.

We going to intruder.

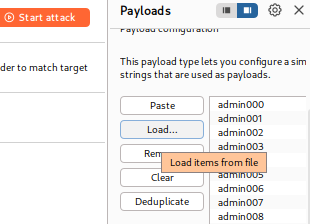
Click “clear”:



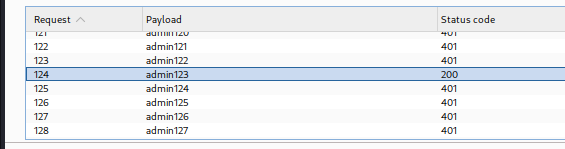
Select the password and click Add:



Load the wordlist:

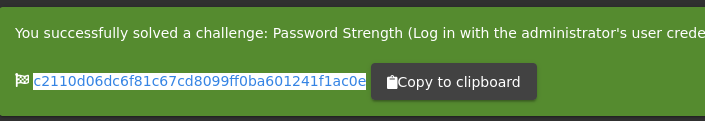


start the attack:



We find password.

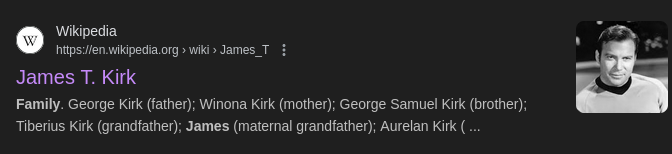
Success!



Ans: c2110d06dc6f81c67cd8099ff0ba601241f1ac0e

**#2 Question #2: Reset Jim’s password!**

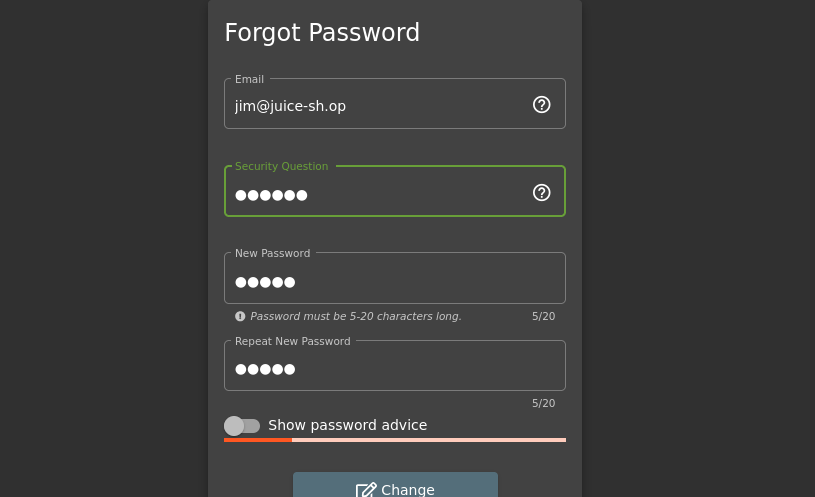
When inputted into the email field in the Forgot Password page, Jim’s security question is set to “Your eldest siblings middle name?”. In Task 2, we found that Jim might have something to do with Star Trek. Googling “Jim Star Trek” gives us a wiki page for Jame T. Kirk from Star Trek. **Looking through the wiki page we find that he has a brother.**

****

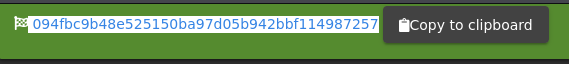
Looks like his brother's middle name is **Samuel**

Inputting that into the Forgot Password page allows you to successfully change his password.

You can change it to anything you want!



security question : Samuel



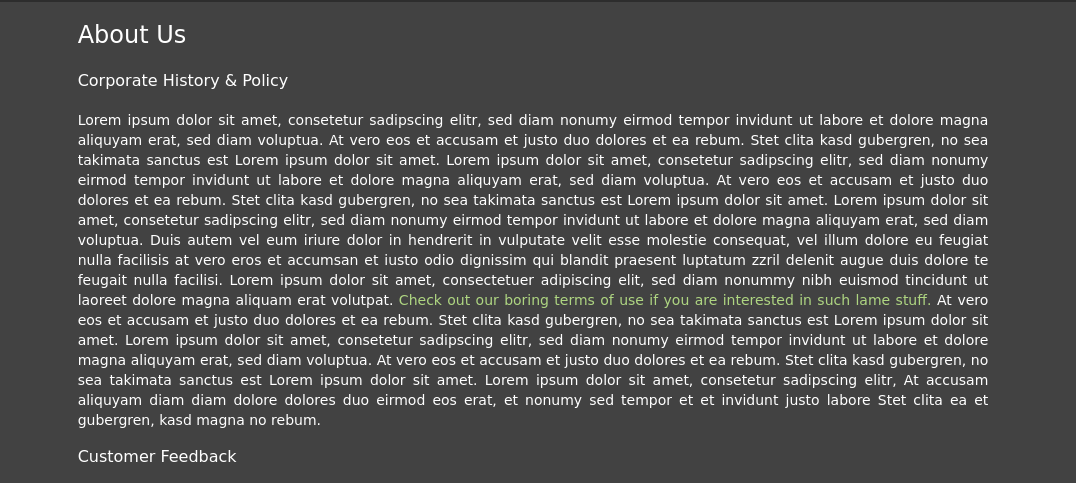
Ans: 094fbc9b48e525150ba97d05b942bbf114987257

**Task 5 : AH! Don’t look!**

A web application should store and transmit sensitive data safely and securely. But in some cases, the developer may not correctly protect their sensitive data, making it vulnerable.

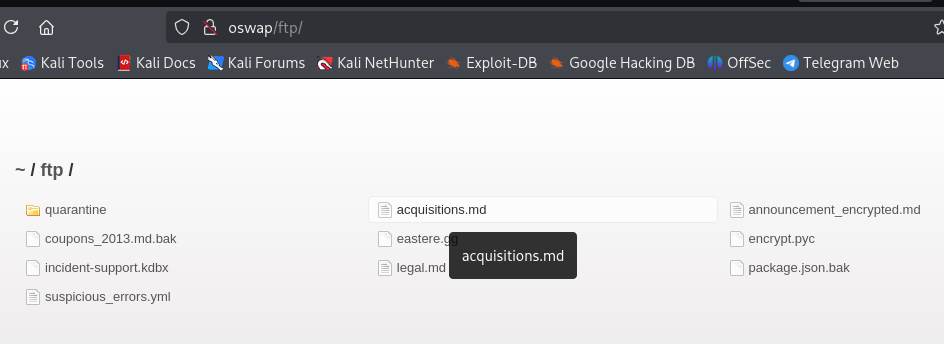
Most of the time, data protection is not applied consistently across the web application making certain pages accessible to the public. Other times information is leaked to the public without the knowledge of the developer, making the web application vulnerable to an attack.

**#1 Question #1: Access the Confidential Document!**

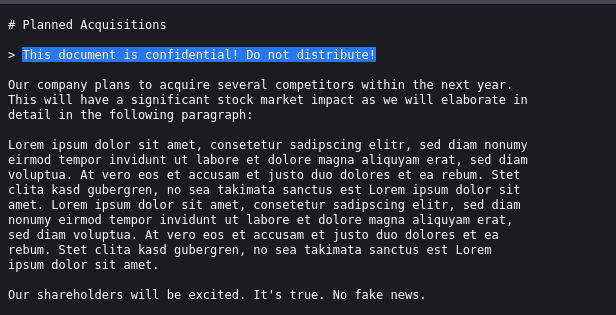


If we click the colored text, We can see /ftp/legal.md

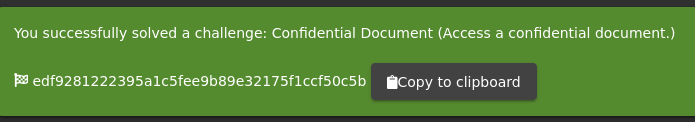
We will download the acquisitions.md and save it. It looks like there are other files of interest here as well.



Download acquisitions.md



After downloading it, navigate to the home page to receive the flag!

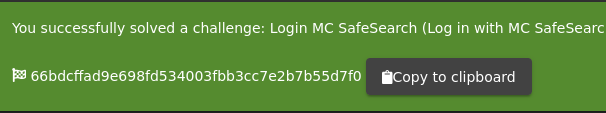


Ans: edf9281222395a1c5fee9b89e32175f1ccf50c5b

**#2 Question #2: Log into MC SafeSearch’s account!**

<https://www.youtube.com/watch?v=v59CX2DiX0Y&embeds_widget_referrer=https%3A%2F%2Fex0a.medium.com%2F&embeds_referring_euri=https%3A%2F%2Fcdn.embedly.com%2F&embeds_referring_origin=https%3A%2F%2Fcdn.embedly.com>

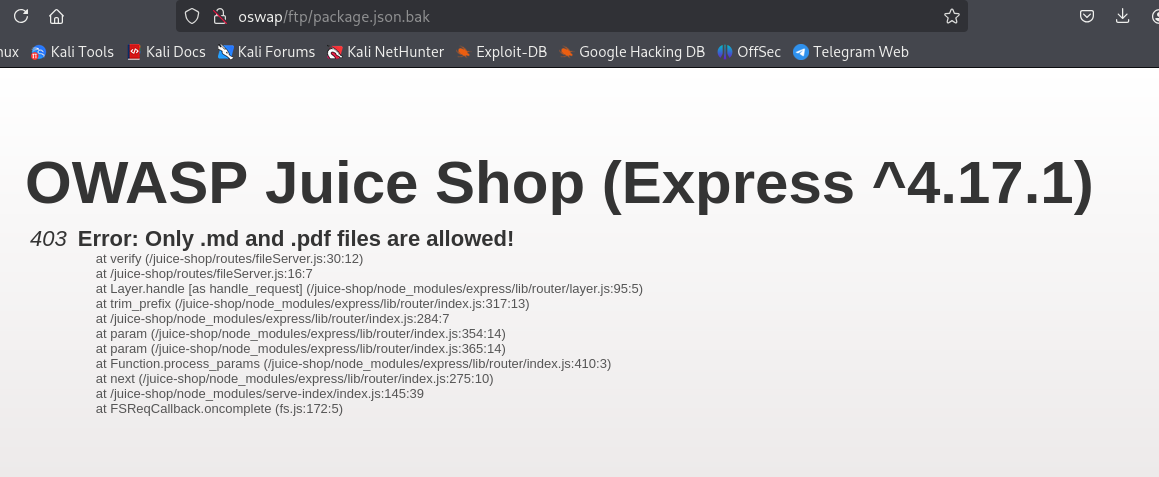
After watching the video, a few parts of the song stood out. He mentions that his password is "Mr. Noodles," but he changed some of the vowels to zeros—specifically, he replaced the "o"s with "0"s. So, the password for the mc.safesearch@juice-sh.op account is “Mr. N00dles.



Ans: 66bdcffad9e698fd534003fbb3cc7e2b7b55d7f0

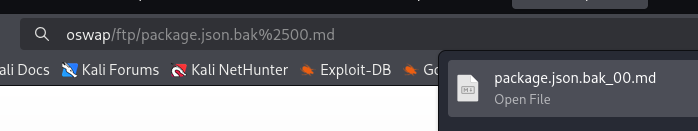
**#3 Question #3: Download the Backup file!**

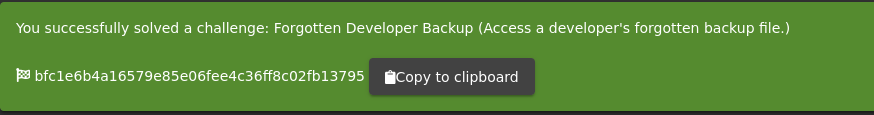
Now we’ll return to the FTP folder and attempt to download the package.json.bak file. However, we encounter a 403 error, indicating that only .md and .pdf files are allowed for download.



To bypass this restriction, we can use a technique known as the “Poison Null Byte,” which is represented as %00.

Since we’re downloading via a URL, we need to URL-encode it. When encoded, the Poison Null Byte becomes %2500. By appending this followed by .md to the filename, we can trick the system and successfully bypass the 403 error.





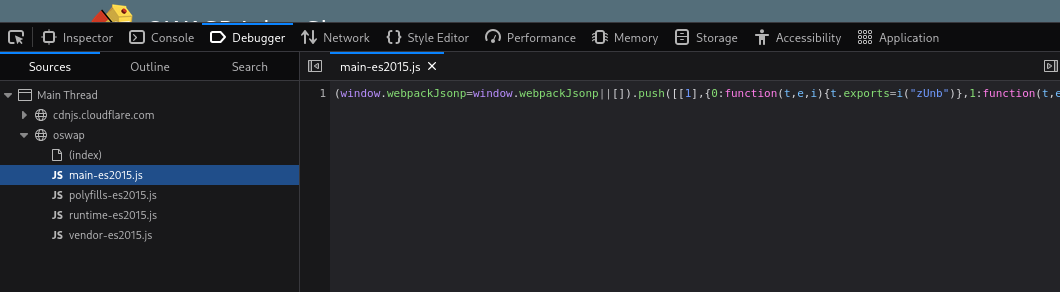
**Task 6 : Who’s flying this thing?**

Modern-day systems will allow for multiple users to have access to different pages. Administrators most commonly use an administration page to edit, add and remove different elements of a website. You might use these when you are building a website with programs such as Weebly or Wix.

When Broken Access Control exploits or bugs are found, it will be categorised into one of two types:  
**Horizontal Privilege Escalation :** Occurs when a user can perform an action or access data of another user with the same level of permissions.  
**Vertical Privilege Escalation** : Occurs when a user can perform an action or access data of another user with a higher level of permissions.

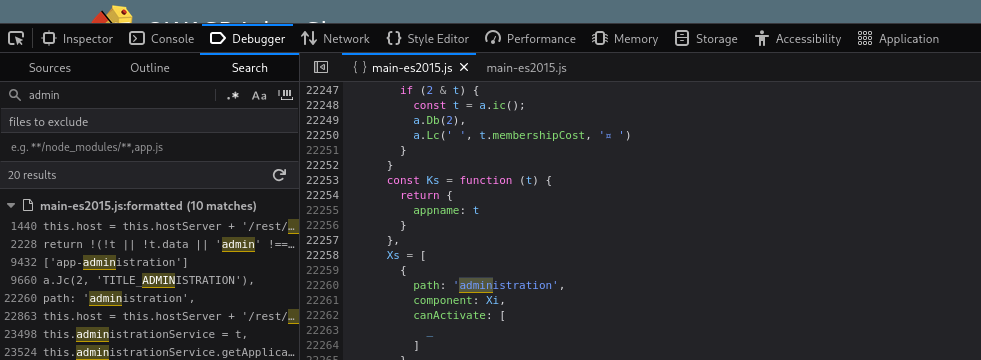
**#1 Question #1: Access the administration page!**

We are then going to refresh the page and look for a javascript file for main-es2015.js

To get this into a format we can read, click the { } button at the bottom

Now search for the term "admin"

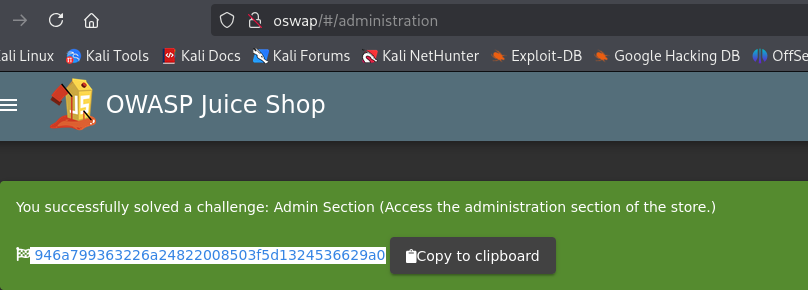
You will come across a couple of different words containing "admin" but the one we are looking for is "path: administration"



This hints towards a page called "**/#/administration**" as can be seen by the **about** path a couple lines below, but going there while not logged in doesn't work.

As this is an Administrator page, it makes sense that we need to be in the **Admin account** in order to view it.

A good way to stop users from accessing this is to only load parts of the application that need to be used by them. This stops sensitive information such as an admin page from been leaked or viewed.

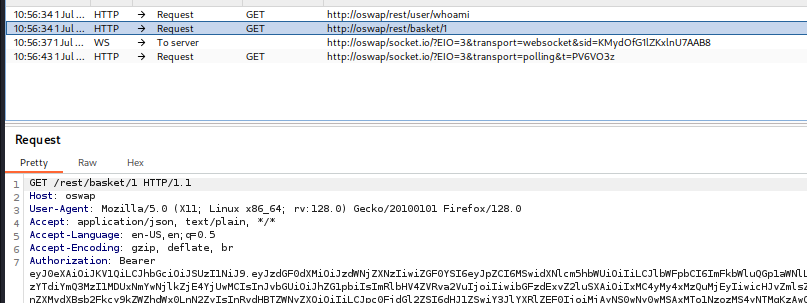


Ans: 946a799363226a24822008503f5d1324536629a0

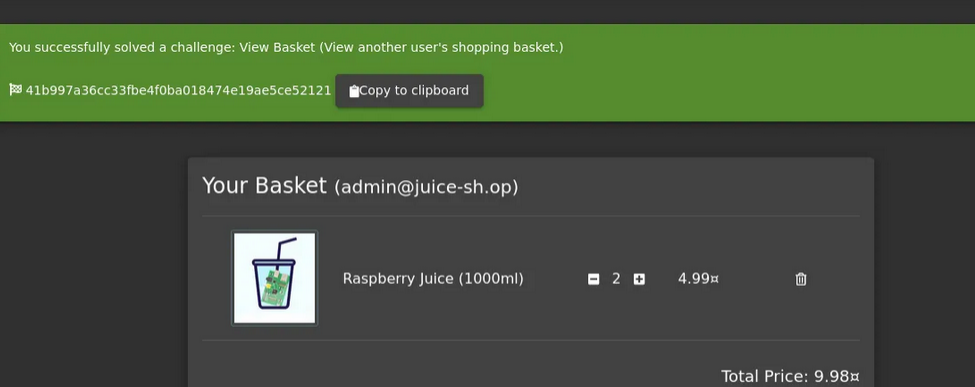
**#2 Question #2: View another user’s shopping basket!**

Login to the Admin account and click on ‘Your Basket’. Make sure Burp is running so you can capture the request!

Forward each request until you see: GET /rest/basket/1 HTTP/1.1

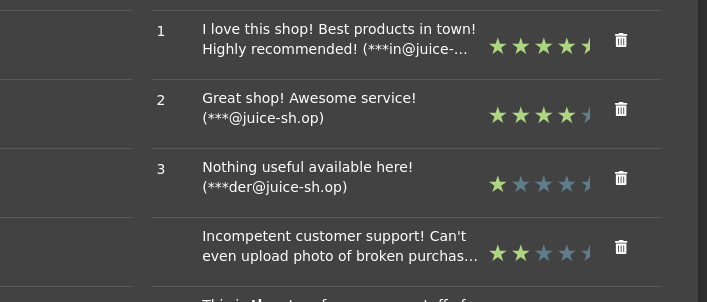


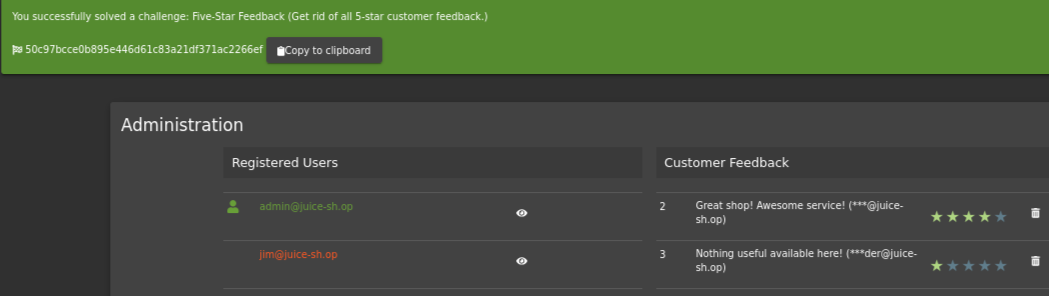
Now, we are going to change the number 1 after /basket/ to 2



**#3 Question #3: Remove all 5-star reviews!**

Navigate to the [http://10.10.63.199/#/administration](http://machine_ip/#/administration) page again and click the bin icon next to the review with 5 stars!





Ans: 50c97bcce0b895e446d61c83a21df371ac2266ef

**Task 7 : Where did that come from?**

XSS or Cross-site scripting is a vulnerability that allows attackers to run javascript in web applications. These are one of the most found bugs in web applications. Their complexity ranges from easy to extremely hard, as each web application parses the queries in a different way.

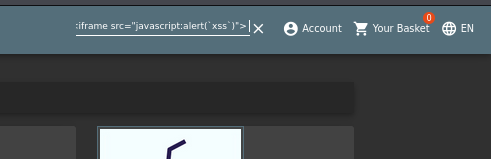
**There are three major types of XSS attacks:**

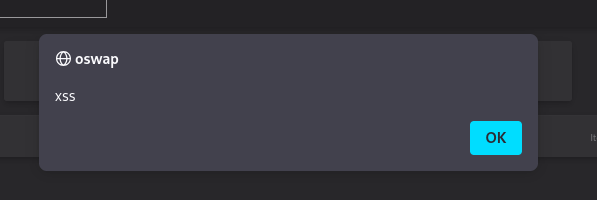
|  |  |
| --- | --- |
| DOM (Special) | **DOM XSS** *(Document Object Model-based Cross-site Scripting)* uses the HTML environment to execute malicious javascript. This type of attack commonly uses the *<script></script>* HTML tag. |
| Persistent (Server-side) | **Persistent XSS** is javascript that is run when the server loads the page containing it. These can occur when the server does not sanitise the user data when it is **uploaded** to a page. These are commonly found on blog posts. |
| Reflected (Client-side) | **Reflected XSS** is javascript that is run on the client-side end of the web application. These are most commonly found when the server doesn't sanitise **search** data. |

**#1 Question #1: Perform a DOM XSS!**

We can use that script : <iframe src=”javascript:alert(`xss`)”>

Inputting this into the **search bar** will trigger the alert.







Ans: 9aaf4bbea5c30d00a1f5bbcfce4db6d4b0efe0bf

**#2 Question #2: Perform a persistent XSS!**

First, login to the **admin** account.

We are going to navigate to the "**Last Login IP**" page for this attack.

It should say the last IP Address is 0.0.0.0 or 10.x.x.x

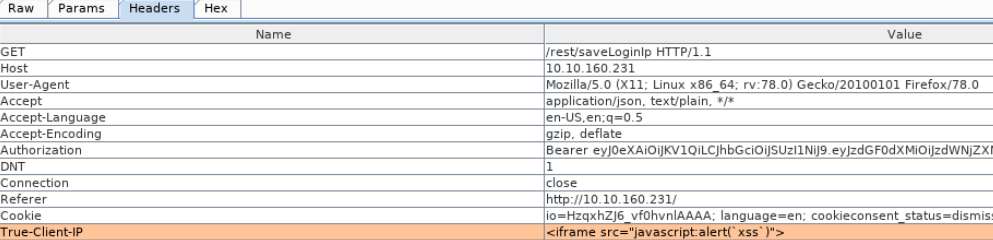
As it logs the 'last' login IP we will now logout so that it logs the 'new' IP.

https://i.imgur.com/4XHHSof.png

Make sure that Burp **intercept is on**, so it will catch the logout request.

We will then head over to the Headers tab where we will add a new header:

|  |  |
| --- | --- |
| *True-Client-IP* | *<iframe src="javascript:alert(`xss`)">* |

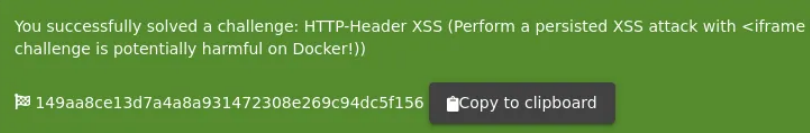


Then forward the request to the server!  
When **signing back into the admin account** and navigating to the Last Login IP page again, we will see the XSS alert!



**Why do we have to send this Header?**

The *True-Client-IP*header is similar to the *X-Forwarded-For*header, both tell the server or proxy what the IP of the client is. Due to there being no sanitation in the header we are able to perform an XSS attack.

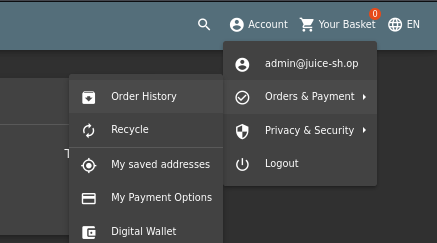


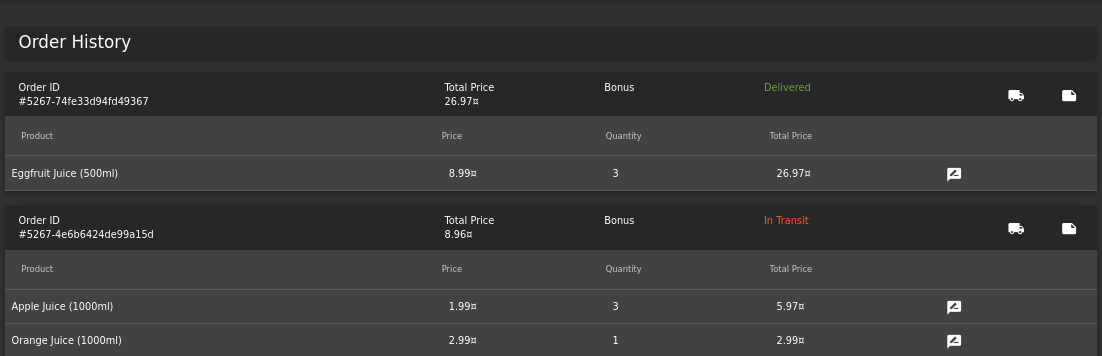
**Ans :** 149aa8ce13d7a4a8a931472308e269c94dc5f156

**#3 Question #3: Perform a reflected XSS!**

First, we are going to need to be on the right page to perform the reflected XSS!

Login into the admin account and navigate to the ‘Order History’ page.





From there you will see a "**Truck**" icon, clicking on that will bring you to the track result page. You will also see that there is an id paired with the order.

https://i.imgur.com/kQdIKyL.png

We will use the iframe XSS, *<iframe src="javascript:alert(`xss`)">,*in the place of the *5267-f73dcd000abcc353* After submitting the URL, refresh the page and you will then get an alert saying XSS!



**Why does this work?**

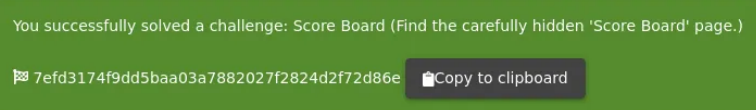
The server will have a lookup table or database (depending on the type of server) for each tracking ID. As the 'id' parameter is not sanitised before it is sent to the server, we are able to perform an XSS attack.

**ans :** 23cefee1527bde039295b2616eeb29e1edc660a0

**Task 8 : Exploration!**

If you wish to tackle some of the harder challenges that were not covered within this room, check out the /#/score-board/ section on Juice-shop. Here you can see your completed tasks as well as other tasks in varying difficulty.

**#1 Access the /#/score-board/ page**



BOOM BOOM BOOM!!



Solomon Tesfaye Ensermu

Ethical Hacker