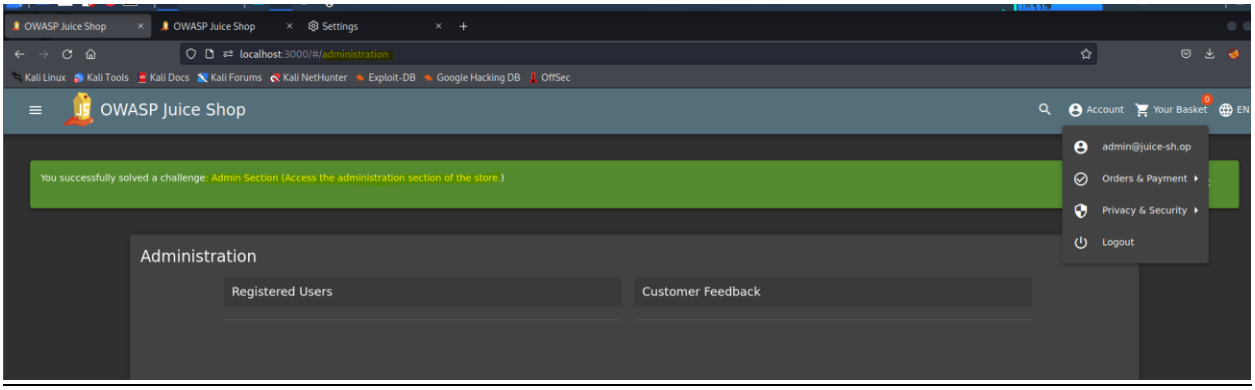
OWASP juice Box Level2:

Installation: follow the below link

https://www.golinuxcloud.com/install-owasp-juice-shop-kali-linux/#Step_1_Download_OWASP_Juice_Shop

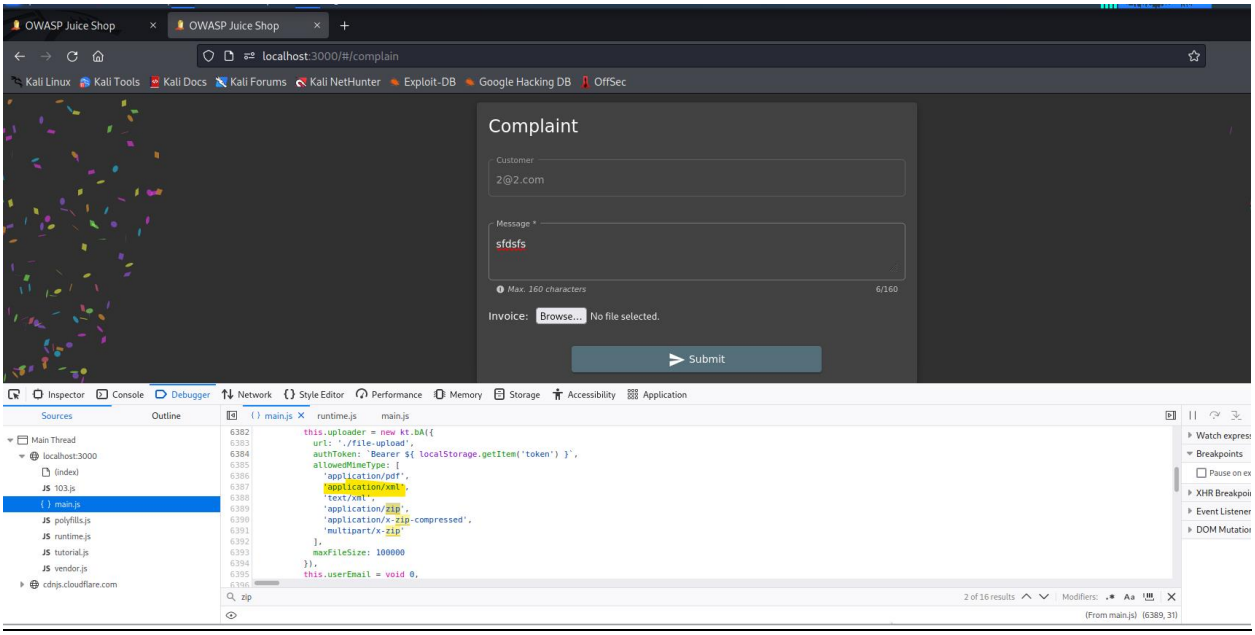
---

Findings 1: Access the administration section of the store (Admin Section)



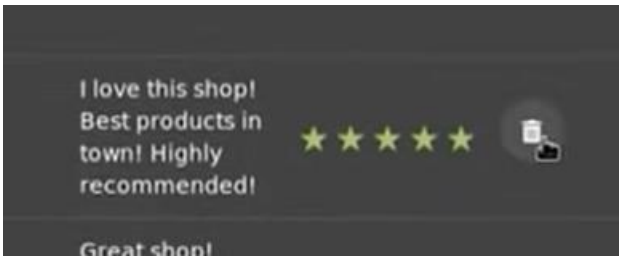
Findings 2: Use a deprecated B2B interface that was not properly shut down

Though it shows we can only upload pdf and .zip files when we inspect the source code, we can see it is also allowing xml files so I have uploaded a file with .xml extension and it worked



Findings 3: Get rid of all 5-star customer feedback.

Delete customer who gave 5 star review



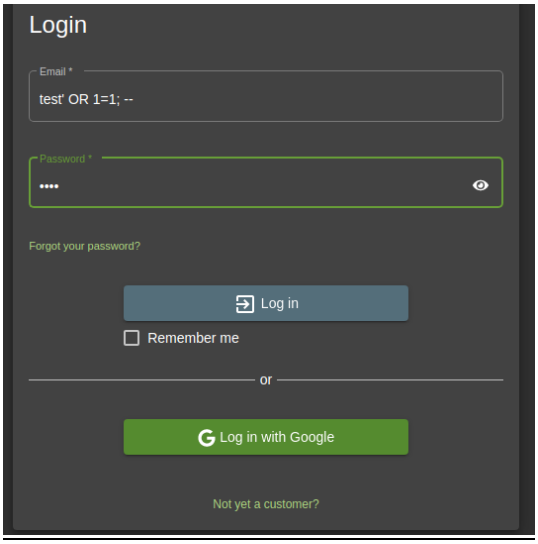
Findings 4: Log in with the administrator's user account.

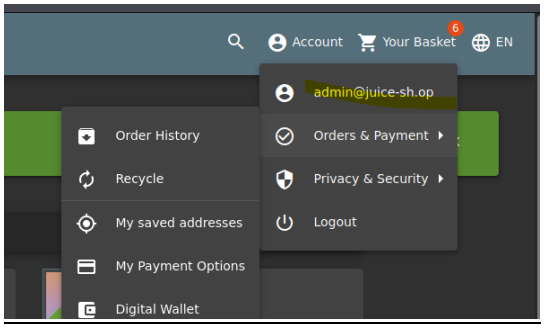
I performed SQL injection attack where I have used this in login test' OR 1=1; --

Which simply means select * from users where emailid = 'test' and password = 'some value'

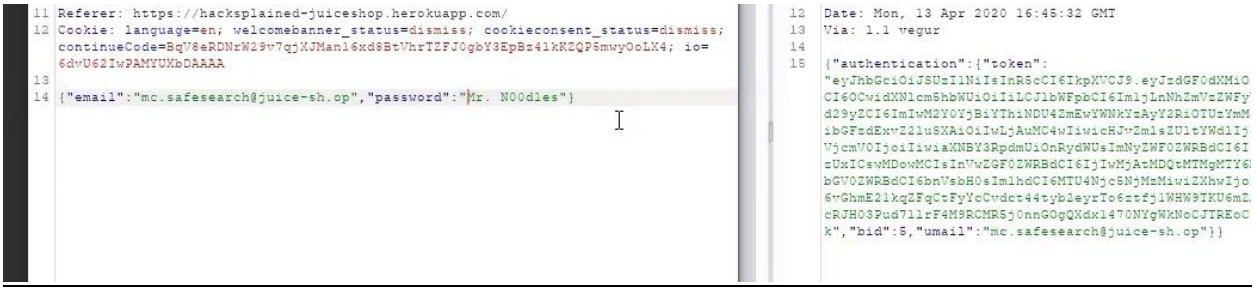
Which simply means select * from users where emailid = 'test' OR 1=1; --**' and password = 'some value'**

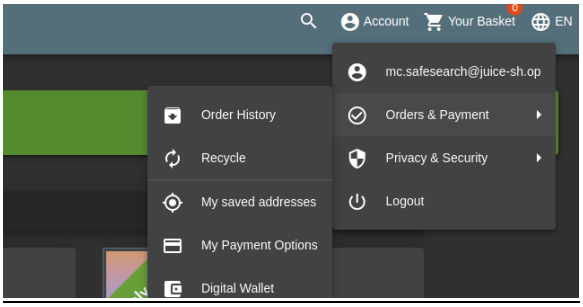
Hence we logged in as administrator





Findings 5: Log in with MC SafeSearch's original user credentials without applying SQL Injection or any other bypass.



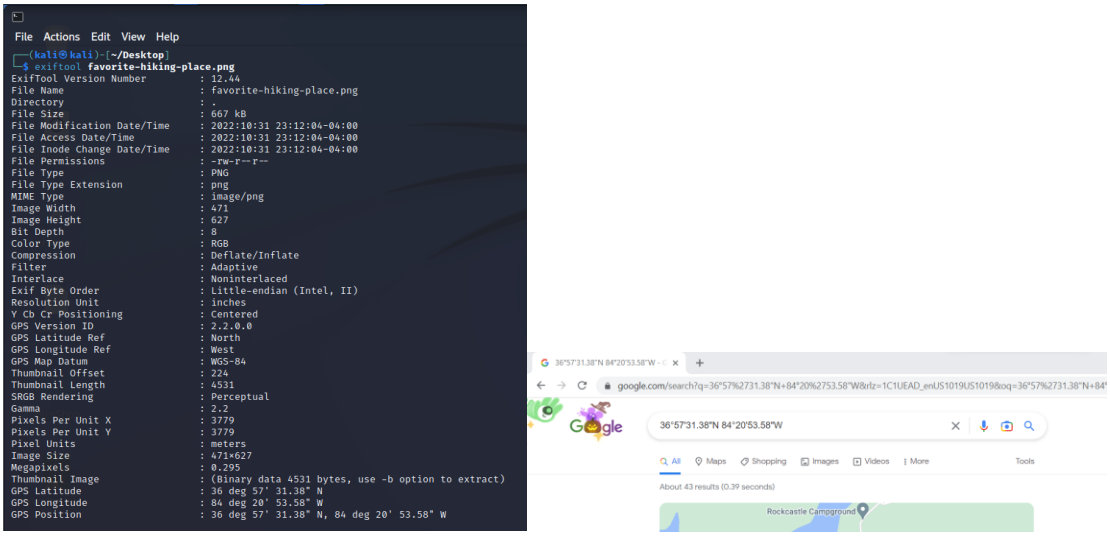


Findings 6: Determine the answer to John's security question by looking at an upload of him to the Photo Wall and use it to reset his password via the Forgot Password mechanism.

I saved the file and used exiftool to view the meta data which contains geo location, latitudes and longitudes which can be used in google maps to get the location which is **Daniel Boone National Forest**.

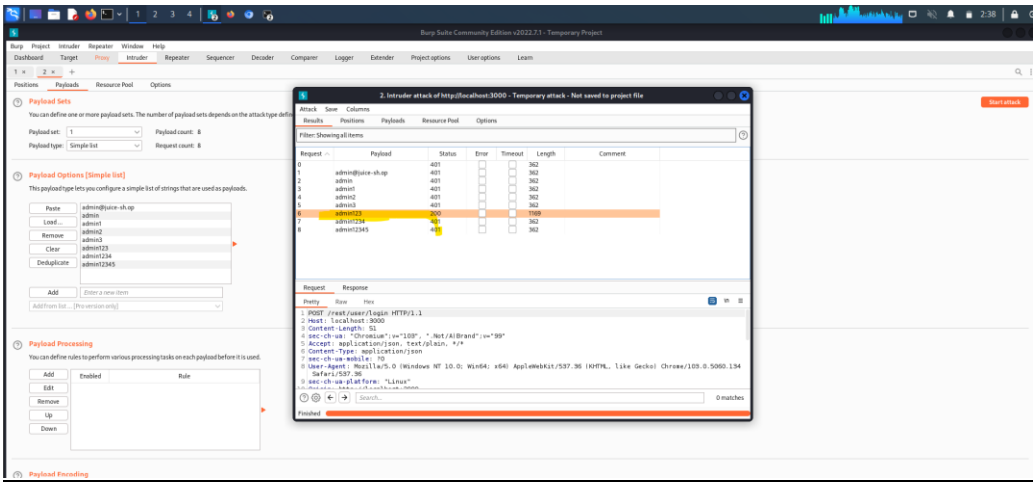
Email is **john@juice-sh.op**

Also the email of john I found it in customer reviews provided



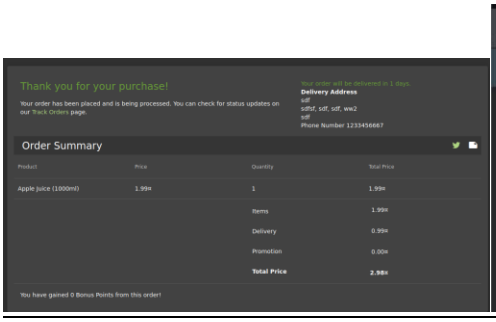
Findings 7: Log in with the administrator's user credentials without previously changing them or applying SQL Injection.

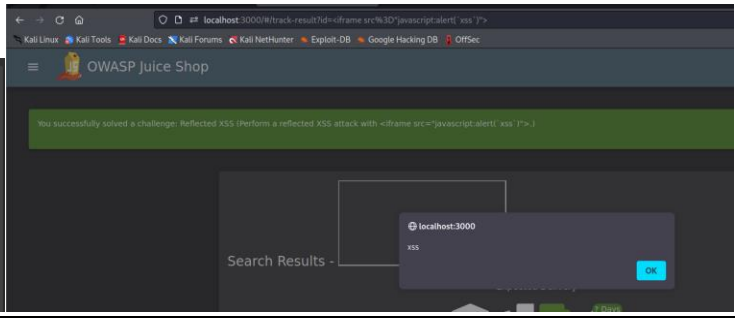
I have used burp suit and used intruder to brute force the password and found the password to be admin123



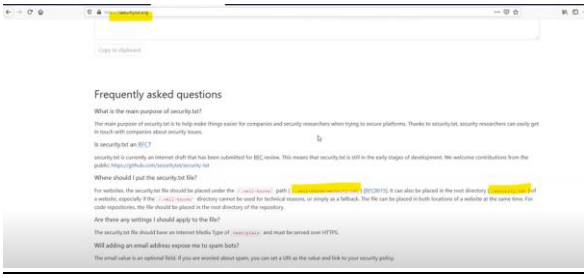
Findings 8: Perform a reflected XSS attack with <iframe src="javascript:alert(`xss`)">.

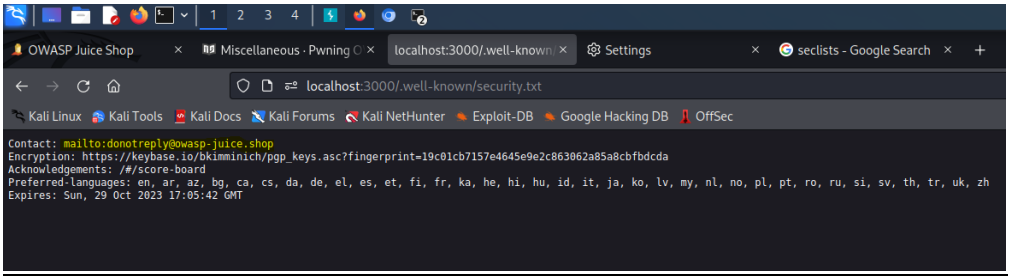
Place an order and route to order history and click track order and set id=<iframe src="javascript:alert(`xss`)"> and reload the page.





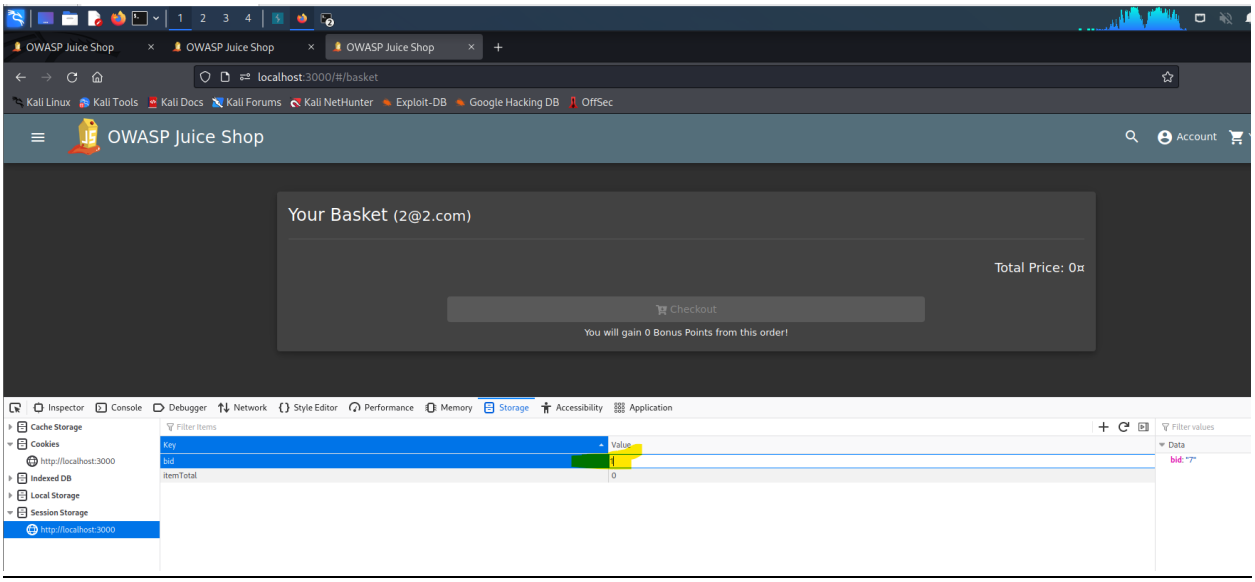
Findings 9: Behave like any "white-hat" should before getting into the action.





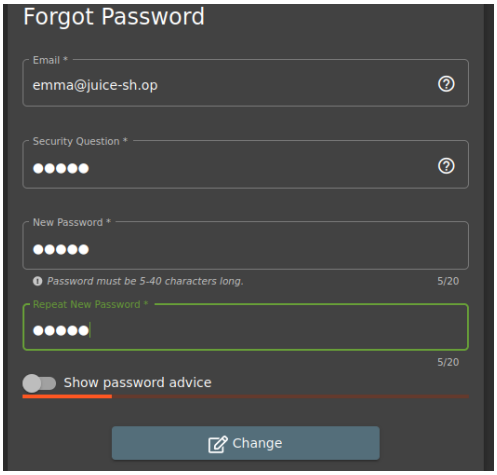
Findings 10: View another user's shopping basket.

Changing the bid value from the storage you have accesses another user's basket, when the value we accessing the cart of the changed value



Findings 11: Determine the answer to Emma's security question by looking at an upload of her to the Photo Wall and use it to reset her password via the Forgot Password mechanism.

I can see a small text written which says ITsec



Findings 12: Inform the shop about an algorithm or library it should definitely not use the way it does.

Md5 is the answer


```
37   tokenOf: (user: UserModel) ⇒ string | undefined
38   from: (req: Request) ⇒ ResponseWithUser | undefined
39   updateFrom: (req: Request, user: ResponseWithUser) ⇒ any
40 }
41
42 exports.hash = (data: string) ⇒ crypto.createHash('md5').update(data).digest('hex')
43 exports.hmac = (data: string) ⇒ crypto.createHmac('sha256', 'pa4qacea4VK9t9nGv7yZtwmj').update(data).digest('hex')
44
```
