Lab 8: DoS Vulnerability Report

Name: Jerome Arsany Mansour Farah
Id: 2305093
Course: Secure Software Development

---

1. Introduction:

In this lab, I tested the security of the OWASP Juice Shop application. My goal was to see if I could crash the server using a "Denial of Service" (DoS) attack. After crashing it, I fixed the code to stop the attack.

---

2. Vulnerabilities Found (OWASP Top 10):

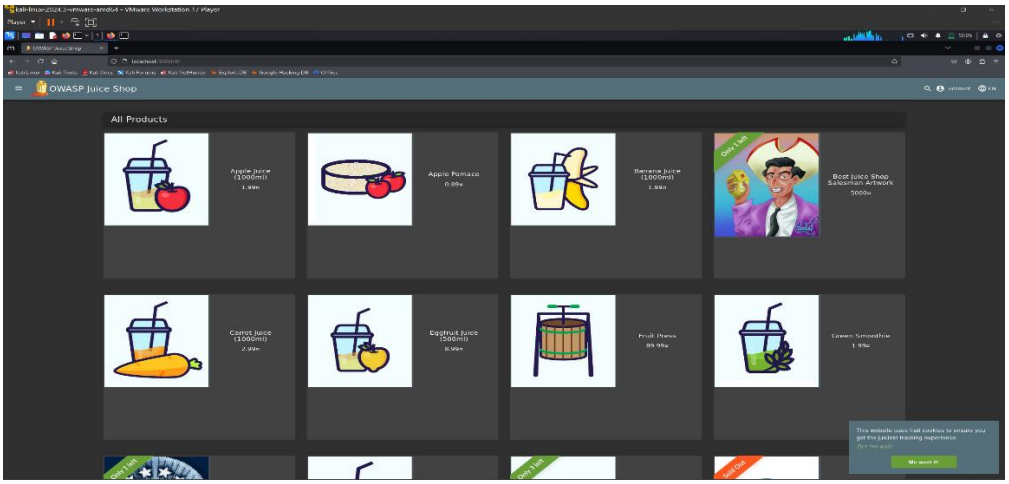
I identified five main security problems in the application:

1. **Broken Access Control (A01):** The server did not limit how many times one person could visit a page. This allowed me to send thousands of requests at once.
2. **Security Misconfiguration (A05):** When the server crashed, it showed detailed error messages and code (stack traces) instead of a simple error page.
3. **Vulnerable Components (A06):** The application uses old software libraries that have known security issues.
4. **Identification Failures (A07):** The login page allowed me to try guessing passwords many times without locking me out.
5. **Injection (A03):** The search bar was very slow when handling complex data, which shows it might be vulnerable to code injection.

---

3. Attack Results (Before Fix):

I used a tool called **Artillery** to send 50 requests every second to the server.
Result: The server could not handle the traffic. It became very slow (latency over 2500ms) and then crashed completely with a "Heap Out of Memory" error.



```
Summary report @ 12:31:14(-0500)

http.codes.401: ..... 1200
http.downloaded_bytes: ..... 31200
http.request_rate: ..... 18/sec
http.requests: ..... 1200
http.response_time:
  min: ..... 10
  max: ..... 3094
  mean: ..... 461.1
  median: ..... 159.2
  p95: ..... 2515.5
  p99: ..... 3011.6
http.response_time.4xx:
  min: ..... 10
  max: ..... 3094
  mean: ..... 461.1
  median: ..... 159.2
  p95: ..... 2515.5
  p99: ..... 3011.6
http.responses: ..... 1200
vusers.completed: ..... 1200
vusers.created: ..... 1200
vusers.created_by_name.0: ..... 1200
vusers.failed: ..... 0
vusers.session_length:
  min: ..... 14.5
  max: ..... 3097.2
  mean: ..... 465.9
  median: ..... 165.7
  p95: ..... 2515.5
  p99: ..... 3011.6
Log file: login-report.json
```

```
Summary report @ 12:32:55(-0500)

http.codes.500: ..... 1800
http.downloaded_bytes: ..... 0
http.request_rate: ..... 30/sec
http.requests: ..... 1800
http.response_time:
  min: ..... 1
  max: ..... 160
  mean: ..... 9.6
  median: ..... 7
  p95: ..... 23.8
  p99: ..... 40.9
http.response_time.5xx:
  min: ..... 1
  max: ..... 160
  mean: ..... 9.6
  median: ..... 7
  p95: ..... 23.8
  p99: ..... 40.9
http.responses: ..... 1800
vusers.completed: ..... 1800
vusers.created: ..... 1800
vusers.created_by_name.0: ..... 1800
vusers.failed: ..... 0
vusers.session_length:
  min: ..... 6.1
  max: ..... 268.4
  mean: ..... 18.3
  median: ..... 14.2
  p95: ..... 43.4
  p99: ..... 79.1
Log file: feedback-report.json
```

4. Mitigation (The Fix):

To fix this, I added a "Rate Limiter" to the server code. This tool counts how many requests a user sends. If they send more than 100 requests in 15 minutes, the server blocks them.

I modified the server.ts file to use the express-rate-limit tool.

---

5. Verification (After Fix):

I ran the same attack again to test the fix.
Result: The server **did not crash**. Instead, it blocked the attack. The report shows that 3,050 requests were rejected with the error code **429 (Too Many Requests)**. The server stayed online for normal users.

```
Summary report @ 14:53:02(-0500)
  ──────────────────────────────

http.codes.200: ..................................................... 100
http.codes.429: ..................................................... 3050
http.downloaded_bytes: .............................................. 128100
http.request_rate: .................................................. 35/sec
http.requests: ...................................................... 3150
http.response_time:
  min: .............................................................. 0
  max: .............................................................. 153
  mean: ............................................................. 3.5
  median: ........................................................... 2
  p95: .............................................................. 5
  p99: .............................................................. 61
http.response_time.2xx:
  min: .............................................................. 14
  max: .............................................................. 153
  mean: ............................................................. 53.3
  median: ........................................................... 54.1
  p95: .............................................................. 94.6
  p99: .............................................................. 138.4
http.response_time.4xx:
  min: .............................................................. 0
  max: .............................................................. 30
  mean: ............................................................. 1.8
  median: ........................................................... 2
  p95: .............................................................. 3
  p99: .............................................................. 7.9
http.responses: ..................................................... 3150
vusers.completed: ................................................... 3150
vusers.created: ..................................................... 3150
vusers.created_by_name.0: ........................................... 3150
vusers.failed: ...................................................... 0
vusers.session_length:
  min: .............................................................. 2.9
  max: .............................................................. 224.3
  mean: ............................................................. 7.5
  median: ........................................................... 5
  p95: .............................................................. 13.6
  p99: .............................................................. 71.5
Log file: products-fixed-final.json
```

---

6. Conclusion:

The lab was successful. I proved that without protection, the application was easy to crash. By adding a simple Rate Limiter in the code, I fully protected the server from this type of DoS attack.