FUTURE INTERNS

TASK 1: Web Application Security Testing

✅ What You'll Do

Set up and explore a test web app (like DVWA or OWASP Juice Shop)

Use scanning tools like OWASP ZAP, Burp Suite, or Nikto

Test for common vulnerabilities like SQL injection, XSS, and CSRF

Map the vulnerabilities to OWASP Top 10 threats

Document findings with screenshots, impact level, and remediation steps

Compile a Security Assessment Report (PDF format)

📝 Final Deliverables

PDF Security Report with risk rating, screenshots, and suggestions

OWASP Top 10 Compliance checklist

Tool logs (ZAP scan reports, Burp Suite issues, etc.)

(Optional) Video walkthrough of your findings]

Introduction

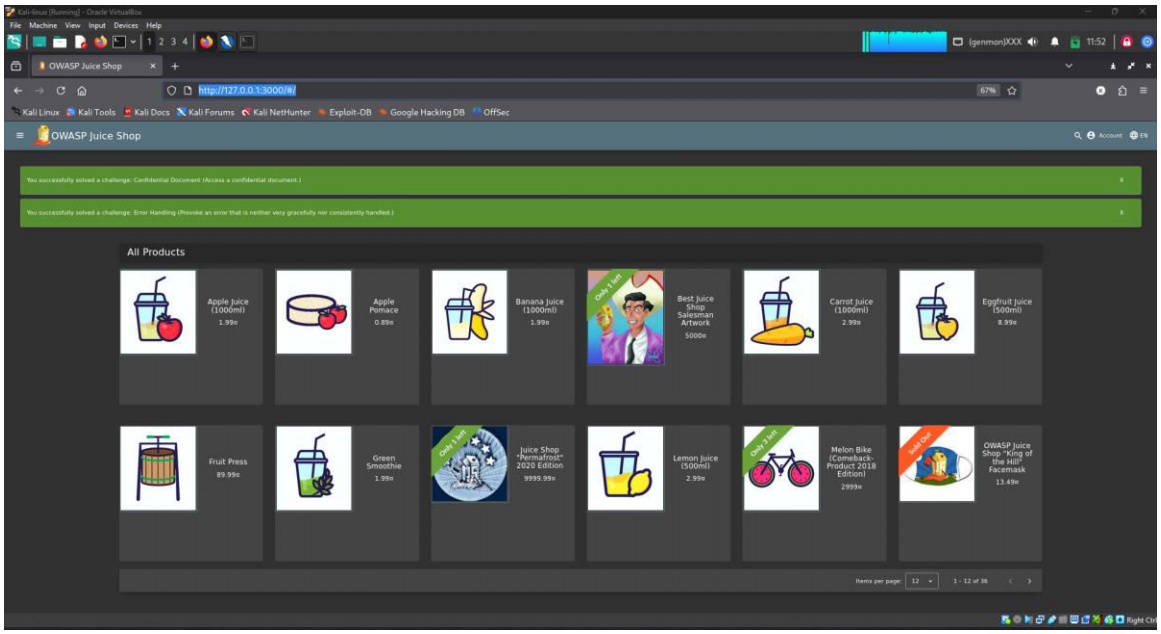
As part of my Cyber Security Internship at Future Interns, I **conducted web application security testing** on OWASP Juice Shop. The goal was to identify common web vulnerabilities like - SQL Injection (SQLi), Cross-Site Scripting (XSS), Brute Force Authentication flaws, etc.

This report details my findings, screenshots, and recommended mitigations.

TASK 1: Set up and explore a test web app (OWASP Juice Shop)

- Before starting anything, I installed docker.io and then used it to lauch the Juice-shop.

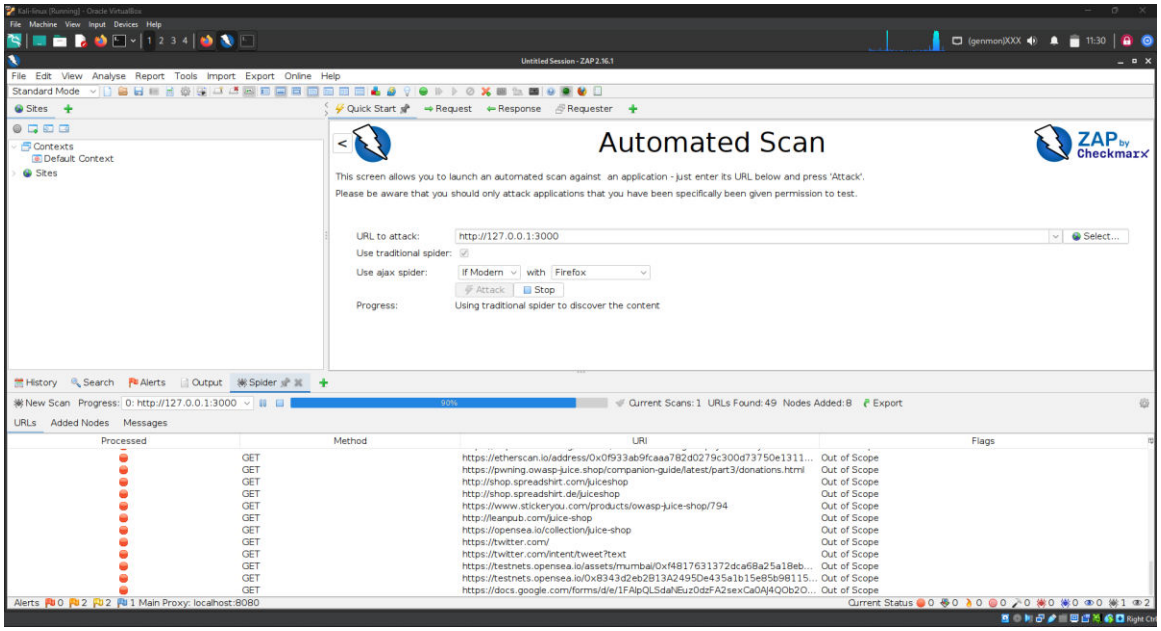
- I already have OWASP ZAP installed, so i just lauched it.



TASK 2: Use scanning tools like OWASP ZAP, Burp Suite, or Nikto (I used OWASP ZAP)

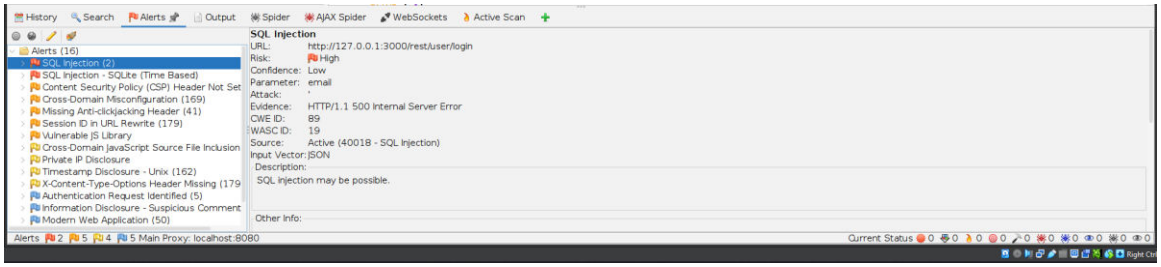
I used OWASP ZAP (2.16.1) on Kali Linux to scan the OWASP Juice-shop application by inputing it's local URL to **URL TO ATTACK** field and running the automated scan by clicking the **ATTACK button**. ZAP first crawled the site to discover available pages and then actively tested those pages for vulnerabilities showing the output in the ALERT tabs with certain ratings . I confirmed the findings by copying the suspicious URLs into my browser to see if the issue could be exploited.

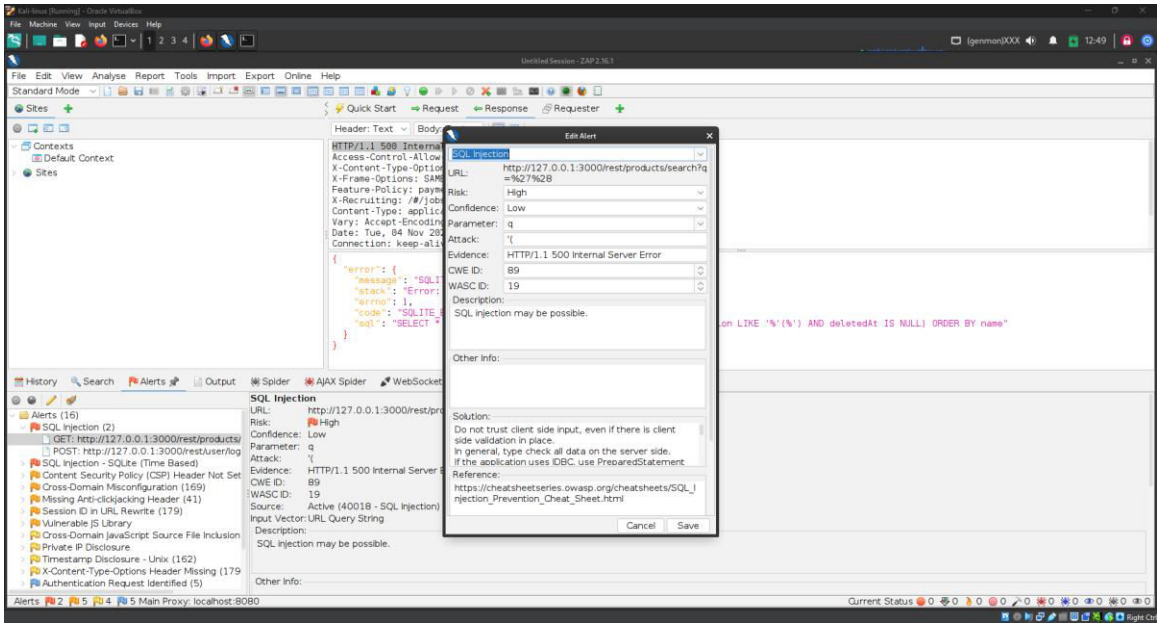
For each confirmed issue, i took screenshots, assessed the harm level and noted key solutions in my report.



As you can see, the **ZAP** is scanning the **OWASP Juice-shop** web application to identify an weaknesses.

The picture below shows the complete scan . ZAP managed to identify all the website weaknesses, though not all is true, you must confirm the weaknesses to see if it's true or false. this can be done by going towards the alert setion and duble clicking on the vulnerability.





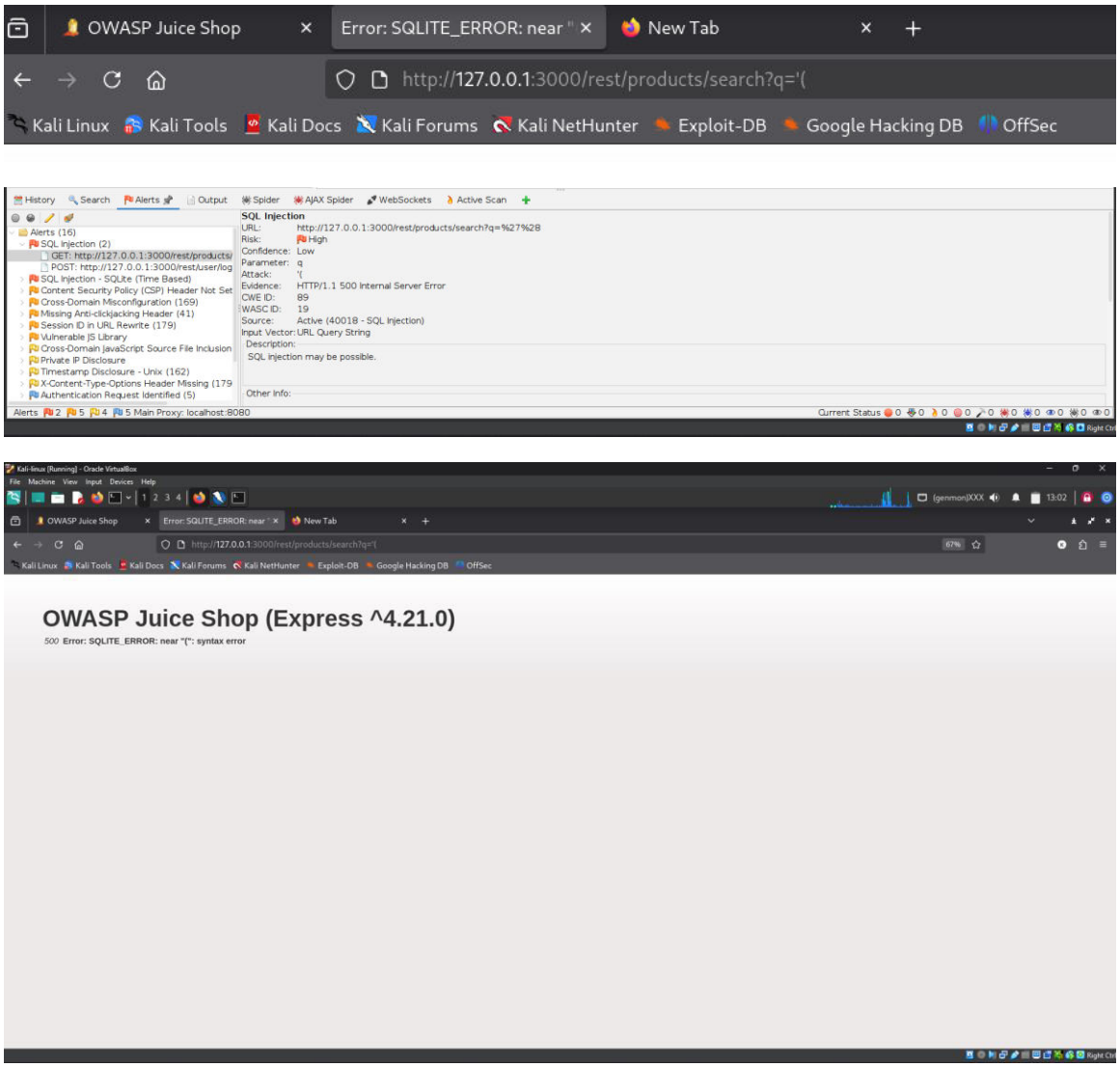
TASK3: Test for common vulnerabilities like SQL injection, XSS, and CSRF

Vulnerability 1 - SQL Injection

Risk Rating: High

Description: SQL injection may be possible

Evidence: I have confirmed this by visiting the page below in firefox browser.



Mitigation: Do not trust client side input even if there's client side validation in place. Applying the principle of least privilege by using the least priviledge databased user possible. Grant the minimum database access that is necessary for the application.

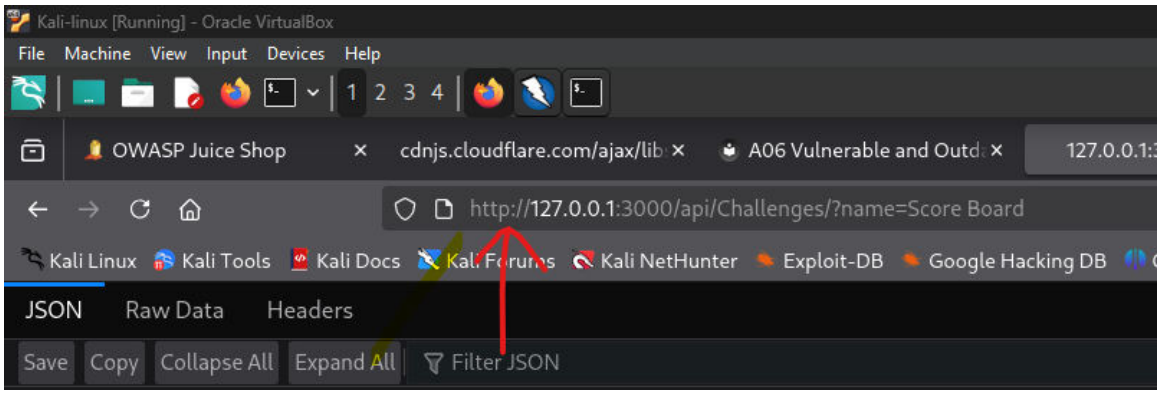
Vulnerability 2 - Cross domain misconfiguration

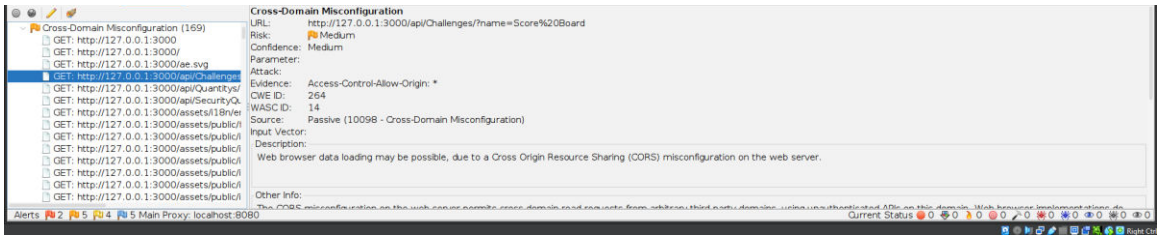
Risk Rating: Medium

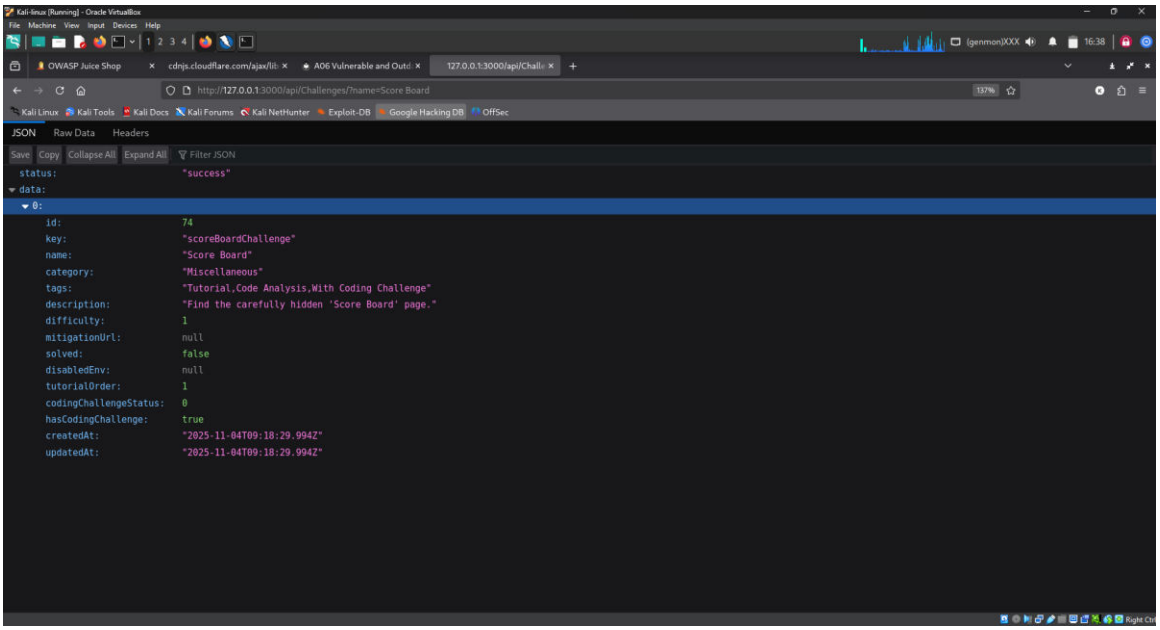
Description: There's a Cross Origin Resource Sharing (CORS) misconfiguration on the web server. The CORS misconfiguration on the webserver permits cross domain read requests from arbitrary third party domains, using unauthenticated APIs on this domain.

Impact: This misconfiguration could be used by an attacker to access data that is unavailable in an unauthenticated manner.

Evidence: The ZAP scanner flagged this as **'Cross Domain-Misconfiguration'** alert. The URL that was affected by this was is found below;







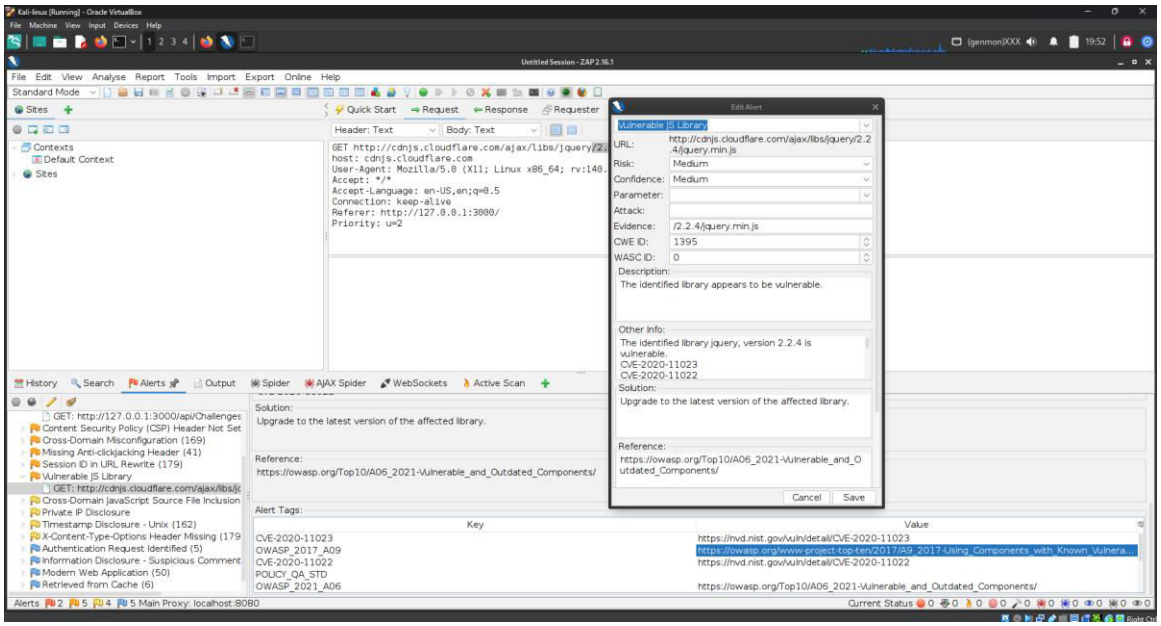
Mitigation: Ensure that sensitive data is unavailable in an unauthenticated (Using IP address white-listing). Configure the **"Access Control Allow Origin"** HTTP header to a more restrictive set of domains or remove all CORS headers entirely to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner.

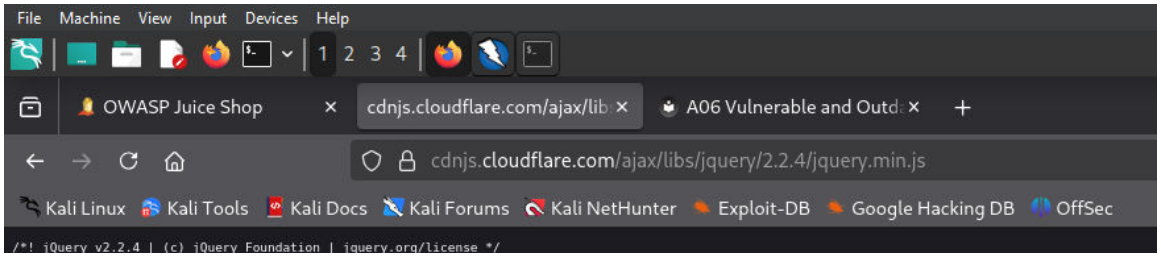
Vulnerability 3 - Vulnerable js Library

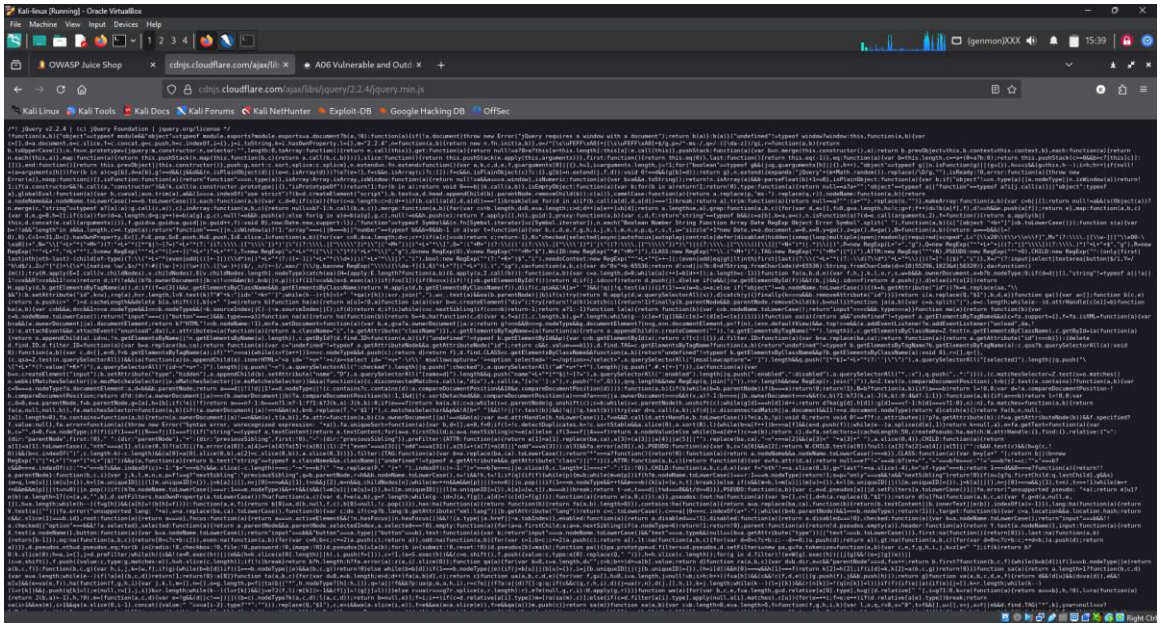
Risk Rating: Medium

Description: The identified Library jquery, version 2.2.4 is vulnerable. CVE-2020-11023, CVE-2020-11022, CVE-2015-9251, CVE-2019-11358.

Evidence: I confirmed this by visiting the page below in firefox browser.







Mitigation: Upgrade to the latest version of the affceted library.

TASK 4: Map the vulnerabilities to OWASP Top 10 threats

NO	VULNERABILITY	EVIDENCE (ALERT/SCREENSHOT)	IMPACT LEVEL	MITIGATION	OWASP TOP 10 CATEGORY
1	SQL Injection	http://127.0.0.1:3000/rest/products/search?q=%27	High	Do not trust client side input even if there's client side validation in place. Applying the principle of least privilege by using the least priviledge databased user possible. Grant the minimum database access that is necessary for the application.	A01:2017 Injection A03:2021 Injection
2	Cross domain misconfiguration	http://127.0.0.1:3000/api/Challenges/?name=Score Board	Medium	Ensure that sensitive data is unavailable in an unauthenticated (Using IP address white-listing). Configure the "Access Control Allow Origin" HTTP header to a more restrictive set of domains or remove all CORS headers entirely to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner.	A01:2021 Broken Access Control A5:2017 Broken Access Control
3	Vulnerable js Library	http://cdnjs.cloudflare.com/ajax/libs/jquery/2.2.4/jquery.min.js	Medium	Upgrade to the latest version of the affceted library	A09:2017 Using Components with Known Vulnerabilities A06:2021 Vulnerable and Outdated Components

Conclusion;

This task strengthened my practical skills in:
- Web application penetration testing
- Identifying real-world vulnerabilities
- Using tools like OWASP ZAP
- Documenting security flaws and proposing fixes

I now better understand how attackers exploit common weaknesses .

Prepared by:

Samuel Emili

Future Interns Cyber Security Intern