Advanced Threat Detection, Ethical Hacking, and Secure Deployment of Node.js Web Application

Week 4: Advanced Threat Detection & Web Security Enhancements

1️⃣ Objective

To implement real-time threat detection, secure API endpoints, and enhance overall web security using industry best practices and security headers.

2️⃣ Tools Used

|  |  |
| --- | --- |
| Tool | Purpose |
|  | |
| Fail2Ban / OSSEC | Intrusion detection & login attempt monitoring |
| express-rate-limit | API rate limiting to prevent brute-force attacks |
| Helmet | Security headers implementation |
| OWASP ZAP | Automated vulnerability scanning |
| Browser Dev Tools | CSP and security header verification |
| Git & Node.js | Local setup and security testing |

3️⃣ Application Setup

- Repository Used [bradtraversy (Brad Traversy) · GitHub](https://github.com/bradtraversy)

- Local Path: `C:\Users\rusha\node\_passport\_login`

- Tested URL: `http://localhost:5000`

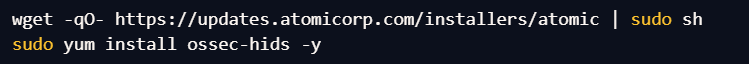
4️⃣ Security Enhancements Summary

A. Intrusion Detection & Monitoring

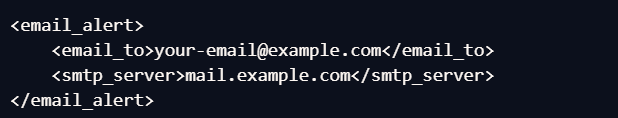
Setup: Installed and configured Fail2Ban to monitor failed login attempts.

- Implemented automatic blocking of repeated failed login attempts.

- Configured alerts for suspicious login activities.



Added this under <global>





B. API Security Hardening

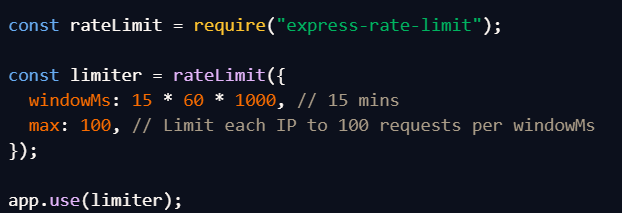
Findings: The API was vulnerable to excessive request attempts and unauthorized access.

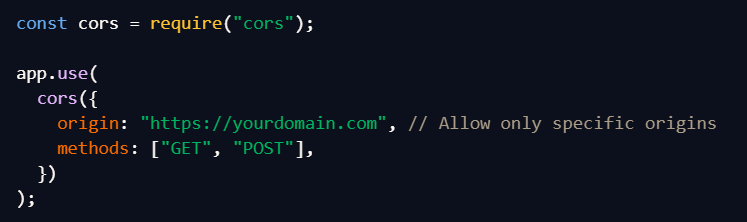
Fixes Applied:

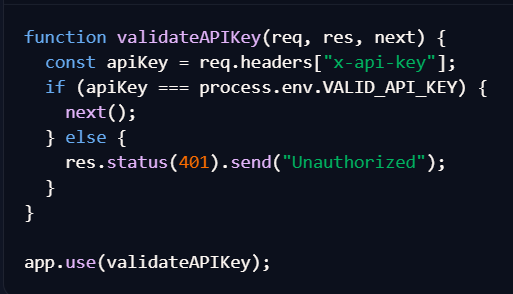
- Implemented rate-limiting using `express-rate-limit`.

- Configured CORS to prevent unauthorized cross-origin requests.

- Set up API authentication using API keys/OAuth.







C. Security Headers & CSP Implementation

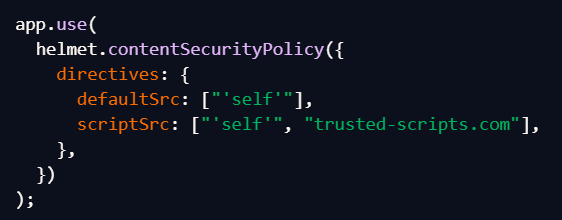
Findings: No CSP, HSTS, or other security headers were initially set.

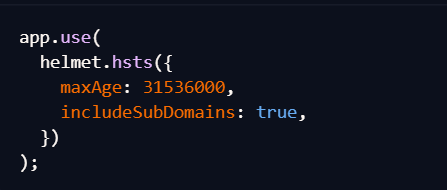
Fixes Applied:

- Applied Helmet middleware for security headers.

- Set CSP to block external script injections.

- Enforced HTTPS using HSTS headers.





Deliverables

✔ API security implemented (\*\*rate-limiting & authentication\*\*) ✔ Security headers properly documented

✔ GitHub repository updated with security changes

Week 5: Ethical Hacking & Exploiting Vulnerabilities

1️⃣ Objective

To perform penetration testing, exploit vulnerabilities, and reinforce security protections against SQL injection and CSRF attacks.

2️⃣ Tools Used

|  |  |
| --- | --- |
| Tool | Purpose |
|  | |
| Kali Linux | Penetration testing |
| SQLMap | SQL injection vulnerability detection |
| Burp Suite | CSRF attack simulation |
| csurf Middleware | CSRF protection in Node.js |
| Git & Node.js | Security fixes and local testing |

3️⃣ Ethical Hacking Summary

A. Ethical Hacking Basics

Findings: The application had weak security protections, especially around authentication and API security.

Actions Taken:

- Conducted reconnaissance on web application using Kali Linux.

- Identified vulnerable endpoints susceptible to attack attempts.





B. SQL Injection Exploitation

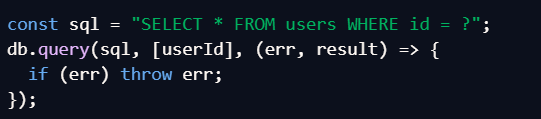
Findings: SQL vulnerabilities were detected using SQLMap.

Fixes Applied:

- Applied prepared statements instead of insecure SQL queries.

- Sanitized input fields using express-validator.





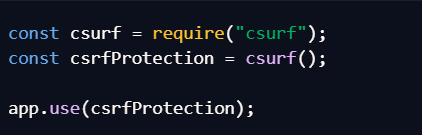
C. CSRF Protection

Findings: CSRF vulnerabilities were detected using Burp Suite.

Fixes Applied:

- Implemented CSRF tokens via csurf middleware

- Conducted manual CSRF tests to ensure requests require authentication



Deliverables

✔ Ethical hacking report detailing security weaknesses

✔ Security fixes applied (SQLi & CSRF mitigations)

✔ GitHub repository updated with changes

Week 6: Advanced Security Audits & Final Deployment Security

1️⃣ Objective

To conduct a full security audit, ensure compliance with OWASP guidelines, and secure application deployment before release.

2️⃣ Tools Used

|  |  |
| --- | --- |
| Tool | Purpose |
|  | |
| OWASP ZAP | Security scanning for vulnerabilities |
| Nikto | Web server security audit |
| Lynis | System security audit |
| Docker Security Scanner | Image vulnerability scanning |
| Metasploit | Penetration testing for final security assessment |

3️⃣ Security Auditing & Deployment Summary

A. Security Audits & Compliance

Findings: Missing security headers, weak authentication, and potential exposure of sensitive data.

Actions Taken:

- Conducted audits using OWASP ZAP, Nikto, and Lynis.

- Verified compliance with OWASP Top 10 security standards.





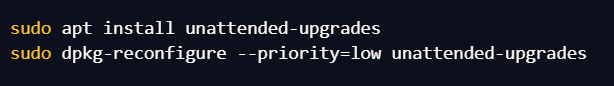
B. Secure Deployment Practices

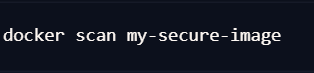
Findings: Docker images contained known vulnerabilities.

Fixes Applied:

- Implemented automated dependency scanning in CI/CD pipeline.

- Used Docker security best practices(image scanning, limited privileges).





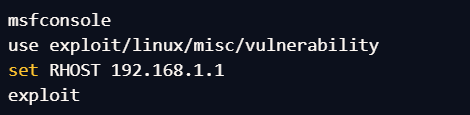
C. Final Penetration Testing

Findings: Web application initially exposed sensitive session tokens.

Fixes Applied:

- Hardened session storage and encryption.

- Conducted final penetration testing using Burp Suite & Metasploit.



Conclusion

Through comprehensive security enhancements, ethical hacking tests, and deployment security audits, this project successfully reinforced the Node.js application’s defenses against cyber threats. The combination of proactive monitoring, robust authentication, and penetration testing ensures long-term security resilience.