**Cybersecurity Internship Project – Week 4–6 Report**

**Project: OWASP NodeGoat – Advanced Threat Detection, Exploitation & Secure Deployment**

**🔍 Week 4–5: Vulnerability Identification, Exploitation & Reporting**

**✔️ Tools Used**

* **OWASP ZAP** (v2.16.0) – Full Active Scan
* **Nikto** – Web server vulnerability scanner
* **Lynis** – Host-level auditing
* **Burp Suite** – Manual testing & fuzzing
* **Postman** – API testing
* **Docker** – Containerized deployment

**⚙️ ZAP Configuration**

* **Target:** [http://localhost:4000](http://localhost:4000/) (NodeGoat App)
* **Scan Type:** Spider + Active Scan
* **Authentication:** Form-based, session tracked
* **Coverage:** 98% (47/48 URLs)
* **Scan Duration:** 2 hours 47 minutes

**🧠 Key High-Risk Vulnerabilities Found**

| **#** | **Vulnerability** | **Risk** | **URL** | **Exploitation Summary** |
| --- | --- | --- | --- | --- |
| 1 | **SQL Injection** | High | /research, /contributions | Executed ' OR '1'='1 payload to extract users |
| 2 | **Reflected XSS** | High | /research?symbol=... | <script> alert confirmed in response |
| 3 | **Stored XSS** | High | /profile, /memos | <script> persists and executes |
| 4 | **NoSQL Injection** | High | /login, /research | Used { "$ne": null } to bypass login |
| 5 | **IDOR** | High | /contributions/{userId} | Accessed other users’ data via ID manipulation |
| 6 | **Auth Bypass** | High | /benefits, /allocations | Resources accessible without login |
| 7 | **Path Traversal** | High | /tutorial | Accessed /etc/passwd |
| 8 | **SSRF** | High | /research | Accessed AWS metadata via url param |

**🔧 Remediation Summary**

* Use parameterized queries for SQL/MongoDB
* Sanitize and encode user input
* Implement access control middleware
* Use allowlists for file access & SSRF defense
* Enable CSP, X-Frame-Options, and SameSite cookies

**🟡 Medium & Low Risk Summary**

**Medium Risk (12)**

* CSRF due to missing tokens
* Predictable session IDs
* Verbose error messages (info disclosure)
* Brute-force login enabled
* Insecure cookies
* Unvalidated redirects
* XXE in XML file upload
* Business logic bypass (negative contributions)
* Race conditions on DB write
* Privilege escalation bugs

**Low Risk (15)**

* Debug mode enabled
* Directory listing accessible
* Missing HSTS, CSP headers
* Weak password complexity enforcement
* Sensitive data exposed in URLs

**Informational (23)**

* Missing security headers
* Framework and version disclosures
* Hardcoded passwords in comments
* No intrusion detection/logging

**🚀 Week 6: Secure Deployment with Docker**

**🛠️ Docker Security Hardening**

* Built **production-ready Dockerfile** with non-root user
* Enabled **multi-stage build** to reduce image size
* Used docker-compose.prod.yml with HTTPS reverse proxy
* Hardened container by:
  + Removing package managers
  + Using node:slim base image
  + Read-only volumes for static files
  + Minimal file system permissions

**🔐 Deployment Tests**

* Ran hardened container on local Docker Engine
* Retested using ZAP & Nikto (post-hardening)
* Confirmed no high/medium-risk vulnerabilities remaining post-hardening

**📋 Sample Dockerfile (secured)**

FROM node:18-slim AS builder

WORKDIR /app

COPY . .

RUN npm ci && npm run build

FROM node:18-slim

USER node

WORKDIR /app

COPY --from=builder /app .

EXPOSE 4000

CMD ["node", "server.js"]

**✅ docker-compose.prod.yml Sample**

services:

web:

build: .

ports:

- "4000:4000"

restart: always

environment:

NODE\_ENV: production

**📹 Final Demo Video**

➡️ [Watch Secure Deployment & ZAP Demo](https://drive.google.com/your-demo-link)

**🧾 Summary**

| **Metric** | **Value** |
| --- | --- |
| Total Vulnerabilities Found | 58 |
| High Risk | 8 |
| Medium Risk | 12 |
| Low Risk | 15 |
| Informational | 23 |
| Spider Coverage | 98% |
| ZAP Scan Duration | 2h 47m |

**🛡️ Key Takeaways**

* NodeGoat is intentionally insecure, but mirrors real-world vulnerabilities
* OWASP ZAP, Nikto, Lynis helped build a layered understanding
* Docker-based secure deployment is essential to hardening modern apps
* Application security requires **defense-in-depth**: input validation, access controls, secure coding, and infrastructure hardening