Dated: 25th March 2025

**Strengthening Security Measures for a Web Application**

Intern: Hassan Nawaz

**Security Audit Report for CRUD React Application**

**1. Overview of the Application**

* **Project Name:** CRUD React App
* **Purpose:** A simple task management application with authentication
* **Technologies Used:** React (Frontend)
* **Security Testing Tools:** OWASP ZAP, Nmap

**2. Security Assessment (Week 1)**

**Vulnerabilities Identified**

✅ **Cross-Site Scripting (XSS)**

* Input fields accepted JavaScript (<script>alert('XSS')</script>) which executed in the browser.
* **Fix:** Implemented DOMPurify to sanitize all user inputs before rendering.

✅ **Missing Content Security Policy (CSP)**

* The app had no Content-Security-Policy header, allowing unsafe inline scripts.
* **Fix:** Implemented react-helmet to enforce strict CSP rules.

✅ **Clickjacking Vulnerability**

* The app was vulnerable to being embedded in an iframe.
* **Fix:** Added X-Frame-Options: DENY to prevent Clickjacking.

✅ **CORS Misconfiguration**

* CORS was allowing all domains (Access-Control-Allow-Origin: \*).
* **Fix:** Restricted CORS to allow only trusted domains.

✅ **Security Headers Missing**

* The application was missing important security headers (X-Content-Type-Options, Strict-Transport-Security).
* **Fix:** Used helmet to add these headers.

**3. Security Enhancements Implemented (Week 2)**

**XSS Prevention**

* **Solution:** Integrated DOMPurify to sanitize user-generated content.

**Content Security Policy (CSP)**

* **Solution:** Implemented react-helmet to enforce a strict CSP.

**Clickjacking Protection**

* **Solution:** Set X-Frame-Options: DENY in HTTP headers.

**CORS Restrictions**

* **Solution:** Restricted API access to trusted domains only.

**Security Logging**

* **Solution:** Configured winston for logging security-related events in security.log.

**4. Final Testing & Results (Week 3)**

**Nmap Scan Results**

✅ No unnecessary open ports detected.

**OWASP ZAP Scan Results**

✅ Major vulnerabilities (XSS, CSP, Clickjacking, etc.) successfully fixed.

**Security Headers Verification**

✅ Confirmed in **Developer Tools → Network Tab**:

* X-Frame-Options: DENY ✅
* X-Content-Type-Options: nosniff ✅
* Content-Security-Policy: default-src 'self' ✅

**Security Logs Generated**

✅ security.log now tracks security events.

**5. Final Submission Details**

* **Video Explanation:** Recorded security audit & fixes.
* **GitHub Repository:** [Insert GitHub Link Here]
* **Security Report:** Attached (this document).

**6. Conclusion**

This security audit successfully identified and fixed vulnerabilities in the CRUD React application. The implemented security measures ensure protection against **XSS, Clickjacking, CORS misconfiguration, and insecure headers**