Cyber Security Internship

Week # 02-Documentation

Project Title: Enhancing Security Measures in a Node.js-Based Web App

Intern: ayesha khan

Tool Stack: Node.js, Express.js, MongoDB, OWASP ZAP, Browser Developer Tools



**Week 2 – Vulnerability Mitigation & Feature Hardening**

**WEEK # 02 -Vulnerability Mitigation & Feature Hardening**

1. **Input Validation and Sanitization**

To address the input-related vulnerabilities identified in Week 1, I integrated the validator npm package. This allowed me to sanitize and validate user inputs on both frontend and backend.

if (!validator.isAlphanumeric(username)) {

return res.status(400).send('Invalid characters in username');

}

Result:

Prevented malicious input, including XSS payloads

Ensured that usernames contain only safe characters



**2. Password Hashing Validation**

The original project already used bcrypt for hashing passwords. I reviewed the logic and verified it was implemented correctly in the registration controller.

Code Verification:

const hashedPassword = await bcrypt.hash(password, 10);

Additionally, using MongoDB Compass, I confirmed that no plaintext passwords were stored.

✅ Result: Passwords are securely hashed, preventing credential theft on DB breach.



**3. Token-Based Authentication Using JWT**

Implemented secure session management using JWT after successful login. The token contains user-specific data and is set to expire after 1 hour.

Code Example:

const token = jwt.sign(

{ id: user.\_id, username: user.username },

'ifra\_secret\_key\_1234',

{ expiresIn: '1h' }

);



Result:

Token successfully created and logged

Prepared for secure access to protected routes

**4. Enforcing Secure HTTP Headers with Helmet**

Integrated the helmet middleware for setting multiple HTTP security headers.

Code Implementation:

const helmet = require('helmet');

app.use(helmet());

✅ Result: Browser DevTools confirmed headers like:

X-Content-Type-Options

X-Frame-Options

Strict-Transport-Security

