Week 3: Advanced Security and Final Reporting – Report

Student Name: Iqra Azam
Internship: Developers Hub – Cybersecurity
Task: Week 3 – Advanced Security Measures and Final Reporting

1. Overview

In Week 3, we focused on **advanced security measures** and preparing the **final report** for the User Management System.

Goals:

1. Perform basic penetration testing to identify vulnerabilities
2. Set up logging to track important events
3. Create a security checklist
4. Prepare final submission including video, GitHub repository, and report

Tools/Libraries Used:

- Node.js / Express
- Winston (for logging)
- Nmap (for basic penetration testing)
- HTTPS (recommended for secure transmission)

2. Basic Penetration Testing

Purpose:

- Simulate common attacks to see if the system is vulnerable
- Identify weak spots like open ports, XSS, or SQL injection

Implementation / Steps:

1. **Using Nmap (network testing)**
    - Command to scan localhost server:
    - `nmap -p 3000 localhost`
    - Checks if port 3000 is open and listening

    - Confirms backend is running but shows no unnecessary open ports
2. **Browser-based testing**
    - Tested signup and login forms with invalid inputs:
        - `<script>alert('XSS');</script>` → Input rejected by validator
        - SQL injection attempt like `admin' OR '1'='1` → Input rejected / backend secure

Result:

- No vulnerabilities were found using basic penetration testing
- Input validation and password hashing protect against XSS and SQL injection

3. Basic Logging Setup

Purpose:

- Keep track of important events in the backend
- Helps in debugging and monitoring security

Implementation:

1. Install Winston:

```
npm install winston
```

2. Add to `server.js`:

```
const winston = require('winston');

const logger = winston.createLogger({
    transports: [
        new winston.transports.Console(),
        new winston.transports.File({ filename: 'security.log' })
    ]
});

// Example usage
logger.info('Application started');
```

Result:

- Every important event, like server start or errors, is logged in `security.log` file
- Console also shows logs for easy debugging

4. Security Checklist

Purpose:

- Ensure best practices are implemented
- Can be referred to for future projects

Checklist:

Security Measure	Status / Action
Validate all inputs (email, password)	Implemented using `validator`
Hash and salt passwords	Implemented using `bcrypt`
Token-based authentication	Implemented using `jsonwebtoken`
Secure HTTP headers	Implemented using `helmet`
Use HTTPS for data transmission	Recommended (can use self-signed certificate for localhost)
Logging	Implemented using `winston`
Check for vulnerabilities	Tested using browser inputs and Nmap
Avoid storing plain passwords	No plain passwords stored

5. Final Submission Details

1. **Recorded Video**
   - Screen recorded explanation of steps, testing, and security fixes
   - Showed signup, login, hashed passwords, token generation, and logs
2. **GitHub Repository**
   - Repository includes:
     - Backend folder with `server.js`
     - `package.json` with all dependencies
     - `security.log` (generated after running server)
     - README file explaining setup and testing
   - GitHub Link (example for submission): https://github.com/ashiktr/reactjs_user_management
3. **Report**
   - Summarizes all tasks, results, and fixes (this document itself)

6. Observations and Results

- Input validation blocks invalid emails and weak passwords
- Passwords are hashed and stored securely
- JWT token provides secure authentication
- HTTP headers are secured using Helmet
- Basic penetration testing shows no vulnerabilities
- Winston logging helps track security events
- Overall system is secure for a basic User Management System

7. Conclusion

Week 3 successfully completes the **advanced security measures** and prepares the **final submission**.

- Security measures are implemented and tested
- System is protected against basic attacks
- Logging ensures all events are tracked
- Final report, video, and GitHub repository ready for submission