Secure Software Systems (IE3042)

Secure Audit Report

Sri Lanka Institute of Information Technology

Lecturer: Chethana Liyanapathirana

Date of Submission: 2024/05/19

IT numbers	Name
IT21826368	Nanayakkara Y.D.T. D
IT21828348	Dissanayaka K.D.A.R. A

Contents

1 Executive Summary

The IoT device management software project's security audit was carried out in accordance with OWASP best practices and guidelines. This project is a full platform with a Node.js and Express backend API, a PostgreSQL database for data storage, and a React-based frontend with machine learning incorporated. It is intended to be used for remote management and monitoring of IoT devices. Critical vulnerabilities were found during the audit, such as weak JWT token management with no expiration or revocation controls, SQL injection flaws caused by insufficient input validation, and data exposure vulnerabilities from sending and storing sensitive data without encryption. Furthermore, vulnerabilities related to cross-site scripting were discovered, which may facilitate the execution of malicious scripts. The security and integrity of the application are seriously jeopardized by these vulnerabilities.

Implemented security measures were assessed to determine their effectiveness in mitigating these risks. The audit revealed shortcomings in authentication mechanisms, data encryption practices, and input validation procedures. Specifically, JWT tokens lacked necessary expiration and revocation controls, and sensitive data was transmitted and stored without encryption, potentially exposing it to unauthorized access. Insufficient input validation further exposed the application to SQL injection and cross-site scripting attacks. Recommendations were developed to address these vulnerabilities, including implementing robust input validation and output encoding, enhancing JWT token management with proper expiration and revocation controls, encrypting sensitive data both in transit and at rest, and improving error handling practices.

The security audit concluded by finding major vulnerabilities in the IoT device management software project that need to be fixed right away. The project may greatly improve its security posture, safeguard sensitive data, and reduce risks by putting the recommended practices into implementation. A secure and robust application that complies with business standards and guards against emerging threats requires regular security audits and preventative security measures.

2 Introduction

The rapid expansion of the Internet of Things (IoT) has brought about significant advancements in how we interact with devices and environments. With the increasing deployment of IoT devices across various industries, ensuring their security has become a paramount concern. The complexity and diversity of IoT ecosystems introduce numerous potential vulnerabilities that can be exploited by malicious actors, leading to unauthorized access, data breaches, and operational disruptions. Therefore, conducting a thorough security audit is essential to identify and mitigate these risks.

This document outlines the security audit conducted on our IoT Device Management System, utilizing the OWASP (Open Web Application Security Project) framework as a guideline. The primary objectives of this audit are to identify and assess potential security vulnerabilities, including injection flaws, authentication and authorization issues, data exposure, and other critical security weaknesses. The audit also aims to evaluate the effectiveness of the implemented security measures, suggest necessary improvements, and propose remediation strategies to enhance the overall security posture of the system. By adhering to established security frameworks and best practices, this audit seeks to ensure that our IoT Device Management System remains secure, reliable, and resilient against emerging threats. The findings and recommendations provided herein will serve as a foundation for continuous improvement in securing our IoT infrastructure, thereby protecting sensitive data and maintaining the trust of our users.

3 Scope

The security audit covered both backend and frontend components of the IoT device management system. Focused on vulnerabilities as above,

Vulnerability	Description	Example
Authentication Vulnerabilities	Weaknesses in how users prove their identity to a system. Attackers exploit these to gain unauthorized access.	Attempting to access the IoT device management system using stolen login credentials.
	Flaws in how users are granted permission to access specific resources or functionalities.	An unauthorized user gains access to device monitoring data due to misconfigured access controls.
Data Validation Vulnerabilities	Issues where user-provided data is not properly checked before being processed. Attackers inject malicious code or manipulate data to compromise the system.	SQL injection attack where an attacker injects malicious code through an input form to steal device data.
Secure Communication Vulnerabilities	Weaknesses in how data is transmitted between systems. Attackers intercept or modify data in transit.	Sending sensitive device management API tokens over unencrypted channels (HTTP instead of HTTPS).
Error Handling Vulnerabilities	Flaws in how errors are reported by the system. Attackers exploit these to gain insights into system functionality or sensitive information.	Revealing detailed error messages that expose internal system workings or database structures.
Database Security Vulnerabilities	Weaknesses in how databases are secured. Attackers gain access to sensitive data stored within.	Using weak database passwords or missing access controls for IoT device monitoring system database users.
API Security Vulnerabilities	Flaws in how APIs are designed and implemented. Attackers exploit these to gain unauthorized access to data or functionality.	An API endpoint without proper authentication allowing unauthorized access to add IoT devices to the system.

4 Methodology

The audit utilized a combination of automated tools and manual testing methods. OWASP guidelines and best practices were followed to identify and assess security risks. Key areas evaluated included:

- Authentication and Authorization Mechanisms
- Input Validation and Output Encoding
- Secure Communication Channels
- Error and Exception Handling
- Database Configuration and Access Controls
- API Security and Rate Limiting

For them we used the tools and methods.

Phase	Tools	Description
Planning and Preparation	MS word	For tracking tasks, findings, and remediation efforts.
		Documentation tools for documenting audit scope, objectives, and team assignments.
Information Gathering	Draw.io	Diagramming tools for reviewing system architecture and data flow diagrams.
	Grep, custom scripts	Tools for searching configuration files and source code for sensitive information.
Vulnerability Assessment	OWASP ZAP, Burp Suite, Nmap.	Automated scanning tools for identifying web application vulnerabilities and network scanning.
	Postman.	Manual testing tools for API testing, SQL injection, and penetration testing.
Reporting and Documentation	MS word	Reporting tools for compiling audit reports.
Framework	OWASP Framework	Guidelines and best practices for identifying and mitigating security vulnerabilities in web applications.
Technologies Native to the Application	Node.js, React, Postgres, OpenSSL	Technologies and platforms used in the application

5 Application Details

Component	Description	Functionalities	Security Features
Backend	Node.js & Express-based server for handling business logic and APIs.	- User authentication and authorization. - Device management (add, update, delete devices) - Sensor data collection and storage. - Anomaly detection and threat prediction using ML. - Secure communication with IoT devices.	Parameter validation using express validator. JWT for secure authentication. Rate limiting using express-rate-limit. Centralized error handling. Secure cookie flags.
Database	PostgreSQL database for storing device data, user information, and sensor data.	- Store user credentials securely. - Store device details and configurations. - Store sensor data logs. - Store machine learning model data.	- Data encryption at rest and in transit. - Strong password policies. - SQL injection prevention.
Frontend	React-based web application for user interaction and management of IoT devices.	- User registration and login. - Dashboard displaying device status - Add and manage IoT devices. - View device details and sensor data. - Configure API integrations.	- Secure communication with backend using HTTPS. - JWT-based authentication. - Role-Based Access Control (RBAC). - Content Security Policy (CSP).
Security	Implementation of security features to protect data and communications.	- Parameter validation using express-validator. - JWT for secure authentication. - CORS setup for secure cross-origin requests. - Password hashing with bcryptjs. - Role-Based Access Control (RBAC). - Rate limiting using express-rate-limit. - Secure cookie flags.	- Comprehensive implementation across all components to ensure data integrity and protection against common attacks.
Machine Learning	Anomaly detection and threat prediction using machine learning algorithms.	- Analyze sensor data for anomalies. - Predict potential threats based on device behavior. - Generate alerts and notifications for abnormal activities.	- Secure handling of training data and models. - Regular updates and validation of ML models. - Access control for ML data and analytics.
API Integrations	RESTful APIs for communication between frontend and backend, and between backend and IoT devices.	- Expose endpoints for device management. - Secure endpoints with authentication and authorization. - Handle data from IoT devices. - Integrate with external services for additional functionalities.	- API security using OAuth2 or JWT. - Input validation and sanitation. - Secure API design to prevent unauthorized access. - Monitoring and logging of API usage.
Deployment and Monitoring	Deployment on a cloud platform (e.g., AWS, Azure, Google Cloud) and monitoring tools for maintenance.	- Deploy backend and database on cloud servers. - Deploy frontend on a web hosting service. - Monitor application performance and security - Regularly update dependencies and security patches.	- Secure cloud configuration (IAM roles, VPCs, security groups) - Continuous monitoring and alerting. - Regular security audits and compliance checks. - Automated patch management.

6 Findings

The audit identified several critical security vulnerabilities in the IoT device management software project:

Vulnerability	Description	Explanation
Injection Flaws	The application is vulnerable to SQL injection attacks due to lack of parameterized queries.	Injection flaws occur when untrusted data is sent to an interpreter as part of a command or query. Attackers can exploit these flaws to execute malicious queries.
Authentication and Authorization	JWT tokens lack expiration and revocation mechanisms, and session timeouts are not enforced.	Without expiration and revocation, JWT tokens remain valid indefinitely, increasing the risk of unauthorized access if tokens are compromised.
Cross-Site Scripting (XSS)	The application is susceptible to XSS attacks due to inadequate input validation and output encoding.	XSS vulnerabilities allow attackers to inject malicious scripts into web pages viewed by other users, potentially leading to data theft or session hijacking.
API Security	APIs lack rate limiting and are vulnerable to brute force and denial-of-service (DoS) attacks.	Without rate limiting, APIs can be abused by attackers to perform brute force attacks or overwhelm the server with requests, causing a denial-of-service.

7 Recommendations

Based on the findings, the following recommendations are proposed to improve the security of the IoT device management software with the help of OWSAP alignments.

Vulnerability	Description	Recommendation	OWASP Top 10 Alignment
Injection Flaws	The application is vulnerable to SQL injection attacks due to lack of parameterized queries.	Implement parameterized queries or an ORM library like Sequelize to mitigate SQL injection vulnerabilities.	A1:2017-Injection
Authentication and Authorization	JWT tokens lack expiration and revocation mechanisms, and session timeouts are not enforced.	Implement JWT token expiration, revocation, and session timeout controls to improve authentication security.	A2:2017-Broken Authentication
Cross-Site Scripting (XSS)	The application is susceptible to XSS attacks due to inadequate input validation and output encoding.	Use proper input validation (e.g., express-validator) and output encoding to prevent XSS vulnerabilities.	A7:2017-Cross-Site Scripting (XSS)
Sensitive Information Disclosure	Error messages reveal sensitive information, including database details and stack traces.	Implement custom error handling to mask sensitive information and provide generic error messages to users.	A6:2017-Security Misconfiguration
API Security	APIs lack rate limiting and are vulnerable to brute force and denial-of-service (DoS) attacks.	Implement rate limiting and additional security measures (e.g., CAPTCHA) to protect APIs from abuse and attacks.	A7:2017-Cross-Site Scripting (XSS)

7.1 Remediation Strategies

To address the identified vulnerabilities and improve the security posture of the IoT device management software project, the following remediation strategies are recommended:

Action	Description	Recommendation	Technology
Patch SQL Injection Vulnerabilities	Addressing SQL injection vulnerabilities is crucial for preventing unauthorized data access and manipulation.	Implement parameterized queries or use ORM libraries like Sequelize to safeguard against SQL injection attacks.	Sequelize, PostgreSQL
Implement Proper JWT Token Management	Proper JWT token management is necessary to secure authentication mechanisms.	Ensure JWT tokens have set expiration times and implement mechanisms for token revocation.	jsonwebtoken
Enhance Input Validation and Output Encoding	Improving input validation and output encoding is key to preventing XSS attacks.	Use input validation libraries like express-validator and ensure proper output encoding.	express-validator
Implement Rate Limiting for APIs	Rate limiting prevents abuse and DoS attacks on APIs.	Implement rate limiting using middleware like express-rate-limit and consider additional security measures like CAPTCHA.	express-rate-limit, CAPTCHA
Train Developers on Secure Coding Practices	Continuous education on secure coding practices helps in mitigating vulnerabilities from the development stage.	Conduct regular security training and workshops to keep developers updated on the latest security practices.	OWASP resources, in-house training
Foster a Security-First Culture	Encouraging a security-first mindset ensures that security is considered at every phase of the software development lifecycle.	Promote security awareness and integrate security considerations into the development process from design to deployment.	Secure Development Lifecycle (SDL)

8 Conclusion

The security audit of our IoT Device Management System has identified critical areas requiring immediate attention, including injection flaws, inadequate authentication mechanisms, and data exposure vulnerabilities. Addressing these issues through immediate fixes such as implementing parameterized queries, proper JWT token management, and data encryption is essential for mitigating immediate threats. Long-term improvements focusing on enhancing input validation, securing API endpoints, and ensuring secure file uploads will fortify the system against a wider range of potential attacks. Additionally, fostering a security-first culture through regular training and adopting secure development practices will contribute to the ongoing security and resilience of the application.

The integration of the OWASP Top 10 security principles has provided a robust framework for evaluating and enhancing our security posture. This systematic approach ensures that we not only patch existing vulnerabilities but also implement proactive measures to prevent future security issues. By prioritizing both immediate and long-term security strategies, our IoT Device Management System will be better equipped to handle emerging threats and maintain the trust of our users.

Overall, this audit underscores the importance of a comprehensive security strategy that includes immediate remediation, continuous improvement, and ongoing education. The recommendations provided, if implemented effectively, will significantly enhance the security and reliability of our IoT Device Management System, ensuring its capability to securely manage IoT devices in various applications and industries.

9 References

- OWASP Top Ten: https://owasp.org/www-project-top-ten/
- "IoT Security: Review of Risks and Prevention Methods" by Kaspersky Lab: A detailed report on the risks associated with IoT devices and best practices for mitigating those risks.
- "State of IoT Security 2020" by Unit 42 (Palo Alto Networks): An annual report that highlights the latest trends and vulnerabilities in IoT security. the latest trends and vulnerabilities in IoT security.
- Node.js Documentation: https://nodejs.org/en/docs/
- Express.js Documentation: https://expressjs.com/
- PostgreSQL Documentation: https://www.postgresql.org/docs/
- React Documentation: https://reactjs.org/docs/getting-started.html
- OpenSSL: https://www.openssl.org/