Project - Security Audit Report

Table of Content

1. Executive Summary

This security audit report presents the findings from our comprehensive security assessment, which included data encryption analysis and web application security testing. The objective was to identify vulnerabilities, assess associated risks, and provide actionable recommendations to enhance the security posture of the application.

2. Introduction

The security audit was conducted to evaluate the effectiveness of existing security measures and identify potential vulnerabilities within the application. This report details the methodologies used, findings from the analysis, identified vulnerabilities and risks, and provides recommendations for mitigation.

3. Methodology

The assessment involved two primary components:

1. Data Encryption Analysis: Evaluating the implementation and effectiveness of data encryption mechanisms.

2. Web Application Security Testing: Conducting a thorough security test using ZAP (OWASP Zed Attack Proxy) to identify common web application vulnerabilities.

4. Findings and Analysis

Data Encryption

Objective: To ensure that sensitive data is adequately protected using robust encryption techniques.

Encryption Algorithm: AES-256-CBC

- Strengths: Strong security with 256-bit key length, widely adopted, and performant.

- Weaknesses: Requires secure key management practices.

Implementation: The application uses AES-256-CBC for encrypting sensitive data. The encryption and decryption functions were tested and verified to work correctly, ensuring data confidentiality.

Web Application Security Testing

Tool Used: ZAP (OWASP Zed Attack Proxy)

- Scope: Comprehensive security scan of the web application.

- Findings: The scan identified multiple vulnerabilities categorized into high, medium, and low severity levels.

5. Vulnerabilities and Risks

High Severity Vulnerabilities

1. PII Disclosure

  - Risk: Exposure of personal identifiable information can lead to identity theft and legal implications.

  - Mitigation: Secure or remove PII from responses.

Medium Severity Vulnerabilities

1. Absence of Anti-CSRF Tokens

  - Risk: Vulnerable to CSRF attacks, allowing unauthorized actions by authenticated users.

  - Mitigation: Implement CSRF tokens.

2. Content Security Policy (CSP) Header Not Set

  - Risk: Increases the risk of XSS attacks.

  - Mitigation: Implement CSP headers.

3. Missing Anti-clickjacking Header

  - Risk: Vulnerable to clickjacking attacks.

  - Mitigation: Implement X-Frame-Options or Content-Security-Policy headers.

4. Cookie No HttpOnly Flag

  - Risk: Cookies without the HttpOnly flag can be accessed via JavaScript, increasing the risk of theft through XSS.

  - Mitigation: Set the HttpOnly flag on cookies to prevent access via JavaScript.

5. Cookie without SameSite Attribute

- Risk: Cookies without the SameSite attribute are vulnerable to CSRF attacks.

- Mitigation: Set the SameSite attribute on cookies to restrict cross-site request contexts.

6. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

- Risk: Disclosure of server information can aid attackers in identifying server technologies and exploiting known vulnerabilities.

- Mitigation: Remove or obscure the X-Powered-By header from responses.

7. Strict-Transport-Security Header Not Set

- Risk: Lack of HSTS headers makes the application vulnerable to SSL stripping attacks.

- Mitigation: Implement HSTS headers to enforce HTTPS connections.

8. Timestamp Disclosure - Unix

- Risk: Disclosure of timestamps can aid attackers in identifying application behaviors and patterns.

- Mitigation: Review and minimize timestamp disclosures where not necessary.

9. X-Content-Type-Options Header Missing

- Risk: Absence of this header can lead to MIME type sniffing vulnerabilities.

- Mitigation: Implement the X-Content-Type-Options: nosniff header to prevent MIME type sniffing.

Low Severity Vulnerabilities

1. Charset Mismatch

- Risk: Mismatch in character sets can lead to rendering issues or security vulnerabilities.

- Mitigation: Ensure consistent character set declarations across the application.

2. Information Disclosure - Suspicious Comments

- Risk: Comments in the code can disclose sensitive information or implementation details.

- Mitigation: Review and remove any sensitive information from comments in the code.

3. Modern Web Application

- Risk: Indicates that the application is modern, which could involve various complexities and newer vulnerabilities.

- Mitigation: Regularly review and update security practices for modern web applications.

4. Re-examine Cache-control Directives

- Risk: Inadequate cache control can lead to sensitive data being cached inappropriately.

- Mitigation: Implement proper cache control directives to ensure sensitive data is not cached.

5. Session Management Response Identified

- Risk: Issues in session management can lead to session hijacking or fixation attacks.

- Mitigation: Review and enhance session management practices, including secure cookie attributes.

6. User Controllable HTML Element Attribute (Potential XSS)

- Risk: User-controllable attributes can lead to XSS attacks.

- Mitigation: Validate and sanitize user inputs to prevent XSS.

6. Recommendations

Immediate Actions

1. Focus on remediating high severity vulnerabilities to safeguard critical data and functionality.

Short-Term Actions

1. Address medium severity vulnerabilities to prevent significant security risks.

Long-Term Actions

1. Resolve low severity vulnerabilities to ensure comprehensive security coverage.

7. Conclusion

By systematically categorizing and prioritizing vulnerabilities, this report provides a clear and actionable roadmap to enhance the application's security posture. Regular security reviews and updates are essential to maintaining a robust defense against potential threats.