# CS 305 Project One

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **11/17/2024** | **Hiep Ha** |  |

## Client



## Instructions

Submit this completed vulnerability assessment report. Replace the bracketed text with the relevant information. In this report, identify your security vulnerability findings and recommend the next steps to remedy the issues you have found.

* Respond to the five steps outlined below and include your findings.
* Respond using your own words. You may also include images or supporting materials. If you include them, make certain to insert them in the relevant locations in the document.
* Refer to the Project One Guidelines and Rubric for more detailed instructions about each section of the template.

## Developer

Hiep Ha

**1. Interpreting Client Needs**

Artemis Financial is a consulting company that manages sensitive financial data for its clients, including retirement plans, investments, and insurance. Given the nature of its operations, secure communication and data protection are of utmost importance.

1. **Value of Secure Communications:**  
   Artemis Financial must ensure secure communication channels to protect client information during transmission and storage. This includes encrypting sensitive data such as Social Security Numbers, financial records, and biometric data.
2. **International Transactions:**  
   The company engages in international transactions, which require compliance with regional laws and data protection standards, such as GDPR for European clients or similar regulations in other jurisdictions.
3. **Governmental Restrictions:**  
   While no explicit government restrictions are noted, adherence to industry standards (e.g., PCI DSS for payment processing) and strong encryption practices is crucial to avoid regulatory penalties.
4. **External Threats:**  
   Potential threats include phishing, SQL injection attacks, man-in-the-middle attacks, and the exploitation of vulnerabilities in outdated software dependencies.
5. **Modernization Requirements:**  
   To modernize its operations, Artemis Financial must:
   1. Regularly update software dependencies to mitigate known vulnerabilities.
   2. Implement secure coding practices for APIs.
   3. Leverage open-source libraries while ensuring their security.

**2. Areas of Security**

Based on the **vulnerability assessment process flow diagram**, the following areas of security are critical to Artemis Financial’s application:

1. **Input Validation:**
   1. Prevents SQL injection and other malicious inputs.
   2. Necessary for fields handling user data (e.g., login credentials, financial information).
2. **API Security:**
   1. Ensures secure communication between client applications and the backend.
   2. Protects against unauthorized access through token-based authentication or OAuth.
3. **Cryptography:**
   1. Safeguards sensitive information during storage and transmission.
   2. Complies with encryption standards to protect international transactions.
4. **Error Handling:**
   1. Prevents detailed error messages from exposing application internals to attackers.
   2. Ensures proper use of try-catch blocks with secure fallback mechanisms.
5. **Code Quality:**
   1. Promotes maintainability and reduces unintended vulnerabilities by following secure coding standards.

**3. Manual Review**

A manual inspection of the provided codebase revealed the following issues:

1. **Hardcoded Credentials:**
   1. **Location:** AuthService.java, line 45.
   2. **Issue:** Credentials are hardcoded, which exposes sensitive information.
   3. **Recommendation:** Use environment variables or configuration files with restricted access.
2. **Unvalidated User Input:**
   1. **Location:** GreetingController.java, input fields.
   2. **Issue:** Lack of input validation exposes the application to SQL injection and XSS attacks.
   3. **Recommendation:** Implement input sanitization and validation using frameworks like Hibernate Validator.
3. **Missing HTTPS Enforcement:**
   1. **Location:** Web server configuration.
   2. **Issue:** Data transmission occurs over unencrypted HTTP.
   3. **Recommendation:** Enforce HTTPS connections using SSL/TLS certificates.
4. **Improper Error Handling:**
   1. **Location:** DocData.java, insufficient try-catch blocks.
   2. **Issue:** Error messages reveal sensitive details about the application's structure.
   3. **Recommendation:** Use generic error messages for external users.
5. **Outdated Dependencies:**
   1. **Location:** POM.xml.
   2. **Issue:** Libraries such as Log4j and Hibernate are outdated and vulnerable.
   3. **Recommendation:** Update to secure versions as identified in the static testing section.
6. **No API Authentication:**
   1. **Location:** Backend API endpoints.
   2. **Issue:** APIs do not require authentication for access.
   3. **Recommendation:** Use token-based authentication mechanisms like OAuth.
7. **Plaintext Data Storage:**
   1. **Location:** Database configuration.
   2. **Issue:** Client financial data is stored in plaintext.
   3. **Recommendation:** Encrypt sensitive data using AES-256 for storage.

**4. Static Testing**

The static testing was conducted using the Maven Dependency-Check Plug-in (version 11.1.0). The scan analyzed 38 dependencies, of which 14 were identified as vulnerable, with a total of 151 vulnerabilities. Below are the findings, along with recommendations for mitigation.

**Key Vulnerabilities Identified**

1. **bcprov-jdk15on-1.46.jar**
   1. Vulnerability IDs: Multiple CVEs, including CWE-470.
   2. Description: This version of the Bouncy Castle cryptography package allows unsafe deserialization of private keys, which could lead to arbitrary code execution.
   3. Severity: HIGH (22 CVEs).
   4. Recommendation: Upgrade to version 1.60 or later.
2. **hibernate-validator-6.0.18.Final.jar**
   1. Vulnerability IDs: CWE-96 and CWE-79.
   2. Description: Vulnerable to improper input validation, which could allow attackers to bypass input sanitation.
   3. Severity: MEDIUM (2 CVEs).
   4. Recommendation: Upgrade to version 6.1.2.Final.
3. **jackson-databind-2.10.2.jar**
   1. Vulnerability IDs: CVE-2020-25649.
   2. Description: Allows XML External Entity (XXE) attacks due to improper entity expansion.
   3. Severity: HIGH (6 CVEs).
   4. Recommendation: Update to the latest version.
4. **log4j-api-2.12.1.jar**
   1. Vulnerability IDs: CVE-2021-44228.
   2. Description: Susceptible to remote code execution via improper input validation.
   3. Severity: LOW (1 CVE).
   4. Recommendation: Upgrade to version 2.17.0 or newer.
5. **logback-core-1.2.3.jar**
   1. Vulnerability IDs: CWE-502.
   2. Description: Vulnerable to deserialization of untrusted data.
   3. Severity: HIGH (2 CVEs).
   4. Recommendation: Update to version 1.2.8 or newer.
6. **snakeyaml-1.25.jar**
   1. Vulnerability IDs: CVE-2022-25857.
   2. Description: Allows entity expansion during load operations, potentially leading to Denial of Service (DoS).
   3. Severity: CRITICAL (8 CVEs).
   4. Recommendation: Migrate to SnakeYAML Engine with restricted aliases for collections.
7. **spring-boot-2.2.4.RELEASE.jar**
   1. Vulnerability IDs: CVE-2022-22965.
   2. Description: Vulnerable to Spring4Shell, allowing remote code execution.
   3. Severity: CRITICAL (3 CVEs).
   4. Recommendation: Upgrade to version 2.6.6 or later.
8. **spring-core-5.2.3.RELEASE.jar**
   1. Vulnerability IDs: Multiple, including CVE-2022-22970.
   2. Description: Contains vulnerabilities that allow bypassing protections for remote file access.
   3. Severity: CRITICAL (11 CVEs).
   4. Recommendation: Update to the latest version.
9. **tomcat-embed-core-9.0.30.jar**
   1. Vulnerability IDs: CVE-2022-23181.
   2. Description: Vulnerable to HTTP request smuggling.
   3. Severity: CRITICAL (26 CVEs).
   4. Recommendation: Upgrade to version 10.0.6 or newer.

**5. Mitigation Plan**

To address the vulnerabilities identified in the manual review and static testing, the following steps are recommended:

1. **Dependency Updates:**
   1. Upgrade all vulnerable dependencies to the latest versions, including:
      1. Log4j (2.17.0 or later).
      2. Hibernate Validator (6.1.2.Final or later).
      3. Spring Framework (5.3.10 or later).
2. **Secure Input Handling:**
   1. Implement robust input validation and sanitization across all user input fields.
3. **Enable HTTPS:**
   1. Configure the application server to redirect all HTTP traffic to HTTPS and install a valid SSL/TLS certificate.
4. **Implement Secure Authentication:**
   1. Require token-based authentication for all API endpoints.
5. **Encrypt Sensitive Data:**
   1. Use AES-256 for data encryption at rest and TLS for encryption in transit.
6. **Improve Error Handling:**
   1. Ensure error messages do not disclose internal application details.
7. **Restrict Hardcoded Credentials:**
   1. Move sensitive information to environment variables or a secure secrets management system.
8. **Monitor Security Practices:**
   1. Continuously integrate dependency vulnerability scans and monitor for emerging threats.