CS 305 Project One Template

Document Revision History

Version	Date	Author	Comments
1.2	Jan 24,2025	Christian Busca	

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
Christian Busca

1. Interpreting Client Needs
Determine your client's needs and potential threats and attacks associated with the company's application and software security requirements. Consider the following questions regarding how companies protect against external threats based on the scenario information:

- What is the value of secure communications to the company?
- Are there any international transactions that the company produces?
- Are there governmental restrictions on secure communications to consider?
- What external threats might be present now and in the immediate future?
- What modernization requirements must be considered, such as the role of open-source libraries and evolving web application technologies?

Findings:

Artemis Financial's RESTful web application is a critical component of its financial operations, encompassing various services such as savings, retirement planning, investments, and insurance. Ensuring the security of this application is of paramount importance for maintaining the trust and confidence of its clients. Below are key considerations relevant to the company's software security needs:

Secure Communications: Secure communication is non-negotiable for Artemis Financial, given the highly sensitive nature of its financial data. Utilizing advanced encryption protocols, such as HTTPS and TLS, ensures that data in transit remains protected from unauthorized access and interception.

International Transactions: Although there is no direct evidence of international transactions being conducted by the company, compliance with international data protection regulations, such as GDPR, should be prioritized to avoid potential future complications.

Governmental Restrictions: Artemis Financial is required to adhere to PCI DSS standards and the Gramm-Leach-Bliley Act to safeguard sensitive financial information and maintain secure communication channels. These regulatory requirements guide the implementation of robust encryption and authentication mechanisms.

External Threats: Common threats, such as hacking (e.g., injection attacks, cross-site scripting, and data breaches) and phishing schemes, present significant risks. Addressing these threats through proactive defense mechanisms, such as intrusion detection systems and enhanced logging capabilities, is essential.

Modernization Requirements:

The integration of open-source libraries offers cost-effective solutions and expanded functionality. However, regular updates and vulnerability monitoring are essential to mitigate security risks.

Adopting evolving web technologies, such as secure API interactions and robust cryptographic protocols, ensures the application remains resilient against emerging threats.

Modernization Requirements:

The integration of open-source libraries offers cost-effective solutions and expanded functionality. However, regular updates and vulnerability monitoring are essential to mitigate security risks.

Adopting evolving web technologies, such as secure API interactions and robust cryptographic protocols, ensures the application remains resilient against emerging threats.

2. Areas of Security

Findings:

Using the Vulnerability Assessment Process Flow Diagram as a reference, the following security areas are identified as critical for Artemis Financial's application:

- **Input Validation**:
  - Proper input validation mitigates the risk of injection attacks by ensuring that user inputs are sanitized and meet expected formats.
  - Example: Hibernate Validator vulnerabilities (CVE-2023-1932) related to improper input validation must be addressed to prevent exploitation.
- **Secure API Interactions**:
  - Authentication, authorization, and secure API endpoints are vital for protecting sensitive data.
  - Example: The Log4j vulnerability (CVE-2020-9488) underscores the importance of secure logging practices to avoid inadvertently exposing sensitive information.
- **Cryptography**:
  - Effective cryptographic measures protect sensitive data both in transit and at rest, ensuring confidentiality and integrity.
  - Example: Vulnerabilities in the Bouncy Castle Crypto library (e.g., CVE-2016-1000344) highlight the need to implement updated encryption protocols.
- **Secure Coding Practices**:
  - Adherence to secure coding standards avoids common flaws, such as hardcoded credentials and insecure deserialization.
  - Example: Vulnerabilities in Jackson Databind (CVE-2020-25649) demonstrate the risks of insecure default configurations.

3. Manual Review

- **application.properties** (Line 12):

    - **Issue**: File upload directory configuration (file.upload-dir=./uploads) could lead to unrestricted or insecure file uploads.
    - **Severity**: Medium.
    - **Recommendation**: Validate and sanitize file uploads, restrict file types, and ensure the directory is outside the web root for added security.
- **DocData.java** (Line 27):
    - **Issue**: Hardcoded database credentials ("jdbc:mysql://localhost:3306/test", "root", "root").
    - **Severity**: High.
    - **Recommendation**: Replace hardcoded credentials with environment variables or a secure secrets management system.
- **CRUDController.java** (Line 12):
    - **Issue**: The /read endpoint accepts unvalidated user input via RequestParam, potentially leading to injection attacks.
    - **Severity**: High.
    - **Recommendation**: Validate and sanitize input parameters, and consider using a whitelist for acceptable input values.
- **GreetingController.java** (Line 15):
    - **Issue**: The /greeting endpoint accepts unvalidated user input via RequestParam, which may lead to reflected XSS attacks.
    - **Severity**: Medium.
    - **Recommendation**: Validate and sanitize user inputs to ensure only safe characters are allowed.
- **DocData.java** (Line 27):
    - **Issue**: Lack of exception handling for database connection failures.
    - **Severity**: Medium.
    - **Recommendation**: Implement robust exception handling and logging to avoid exposing stack traces to users.
- **customer.java** (Line 12):
    - **Issue**: Sensitive account_balance data is exposed without access control.
    - **Severity**: Medium.
    - **Recommendation**: Use proper access controls to ensure sensitive data is not accessed or modified without authorization.
- **myDateTime.java** (Line 9):
    - **Issue**: Potentially incomplete or incorrect handling of date and time data.
    - **Severity**: Low.
    - **Recommendation**: Use the java.time package for robust date and time management.
- **CRUD.java** (Line 7):
    - **Issue**: Potential duplication of sensitive data (content and content2) without clear purpose.
    - **Severity**: Low.

    - **Recommendation**: Avoid duplicating sensitive data unnecessarily, and ensure proper handling of data to prevent information leaks.
- **GreetingController.java** (Line 13):
    - **Issue**: The AtomicLong counter could expose internal state to external systems if logged or exposed through APIs.
    - **Severity**: Low.
    - **Recommendation**: Avoid exposing internal counters or use a wrapper to ensure safe access.
- **RestServiceApplicationTests.java** (Line 10):
    - **Issue**: The test method does not validate application behavior or assert expected outcomes.
    - **Severity**: Low.
    - **Recommendation**: Implement meaningful test cases to ensure application behavior aligns with expected functionality.

4. Static Testing

- **Log4j 2.15.0 (CVE-2021-45046)**:
    - **Issue**: Allows attackers to control log messages and parameters, leading to RCE.
    - **Severity**: Critical.
    - **Mitigation**: Upgrade to Log4j 2.17.1 or later and disable risky protocols.
- **Log4j 2.14.1 (CVE-2021-44228)**:
    - **Issue**: JNDI lookup functionality vulnerability, causing RCE.
    - **Severity**: Critical.
    - **Mitigation**: Upgrade to Log4j 2.17.1 and disable JNDI lookup functionality.
- **bcprov-jdk15on-1.46.jar (CVE-2024-34447)**:
    - **Issue**: Timing side-channel attack compromises cryptographic operations.
    - **Severity**: Critical.
    - **Mitigation**: Upgrade to version 1.70 or later.

  Apache Commons BeanUtils (CVE-2020-13956):
    - **Issue**: Deserialization vulnerability leading to RCE.
    - **Severity**: High.
    - **Mitigation**: Upgrade to a patched version or use safe alternatives for deserialization.
- **Apache Tomcat (CVE-2019-0232)**:
    - **Issue**: Vulnerability allows remote attackers to execute arbitrary code.
    - **Severity**: High.
    - **Mitigation**: Update to the latest Tomcat version.
- **Hibernate Validator 6.0.18.Final (CVE-2023-1932)**:
    - **Issue**: Improper input validation leads to injection vulnerabilities.

- **Severity**: High.
    - **Mitigation**: Upgrade to version 6.2.0.Final or later.

  Jackson Databind 2.10.2 (CVE-2020-8908):
    - **Issue**: Unsafe deserialization potentially leads to RCE.
    - **Severity**: Medium.
    - **Mitigation**: Upgrade to version 2.14.x or later and implement a denylist.
- **Spring Framework (CVE-2018-12536)**:
    - **Issue**: Sensitive information exposure via insecure configurations.
    - **Severity**: Medium.
    - **Mitigation**: Upgrade to a patched version and validate configurations.
- **Jackson Databind (CVE-2020-25649)**:
    - **Issue**: Vulnerable to XXE attacks due to insecure default configurations.
    - **Severity**: Medium.
    - **Mitigation**: Upgrade to version 2.14.x.
- **Spring Security (CVE-2021-22118)**:
    - **Issue**: Improper validation of user input causing information disclosure.
    - **Severity**: Low.
    - **Mitigation**: Upgrade to the latest version and validate user inputs.
- **Spring Framework (CVE-2020-5398)**:
    - **Issue**: HTTP parameter pollution attacks due to improper handling.
    - **Severity**: Low.
    - **Mitigation**: Upgrade Spring Framework and validate HTTP requests.
- **Tomcat Embed Core (CVE-2020-1938)**:
    - **Issue**: AJP protocol vulnerability enabling remote code execution.
    - **Severity**: Low.
    - **Mitigation**: Upgrade to version 9.0.65 or later.

5. Mitigation Plan