# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **07/21/24** | **Lara Walker** |  |

## 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

Lara Walker

**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?

#### Value of Secure Communications

* **Confidentiality and Integrity:** Secure communications are critical to protect sensitive financial data, client information, and internal communications.
* **Compliance Requirements:** Secure communications ensure compliance with industry regulations such as GDPR, PCI-DSS, and FINRA.

#### International Transactions

* **Global Operations:** If Artemis Financial engages in international transactions, it must adhere to various international security standards and ensure data privacy across borders.
* **Cross-Border Data Transfers:** Secure encryption protocols are essential to protect data during cross-border transfers and to comply with regulations such as the EU-US Privacy Shield.

#### Governmental Restrictions

* **Regulatory Compliance:** Artemis Financial must comply with governmental regulations on secure communications, such as encryption standards and data protection laws.
* **Export Controls:** Ensure compliance with export control regulations on encryption technologies.

#### External Threats

* **Cyber Attacks:** Threats such as phishing, ransomware, and DDoS attacks can disrupt operations and compromise sensitive data.
* **Zero-Day Vulnerabilities:** Continuous monitoring and patching of software to protect against newly discovered vulnerabilities.
* **Insider Threats:** Implement strict access controls and monitoring to prevent data breaches from within.

#### Modernization Requirements

* **Open-Source Libraries:** Regularly update and monitor open-source libraries to mitigate vulnerabilities and maintain security.
* **Evolving Web Technologies:** Stay current with the latest web application technologies and best practices to enhance security and performance.

**2. Areas of Security**

Refer to the vulnerability assessment process flow diagram. Identify which areas of security apply to Artemis Financial’s software application. Justify your reasoning for why each area is relevant to the software application.

1. **Authentication and Authorization**
   * **Relevance:** Ensures only authorized users can access sensitive data and functionality.
   * **Justification:** Prevents unauthorized access and potential data breaches.
2. **Data Validation**
   * **Relevance:** Protects against SQL injection, XSS, and other input-based attacks.
   * **Justification:** Ensures data integrity and application stability.
3. **Encryption**
   * **Relevance:** Protects sensitive data in transit and at rest.
   * **Justification:** Maintains data confidentiality and compliance with regulations.
4. **Configuration Management**
   * **Relevance:** Ensures secure and consistent deployment of applications and systems.
   * **Justification:** Prevents misconfigurations that could lead to vulnerabilities.
5. **Session Management**
   * **Relevance:** Secures user sessions against hijacking and unauthorized access.
   * **Justification:** Maintains session integrity and user trust.
6. **Logging and Monitoring**
   * **Relevance:** Detects and responds to security incidents in real-time.
   * **Justification:** Provides visibility into security events and aids in incident response.

**3. Manual Review**

Continue working through the vulnerability assessment process flow diagram. Identify all vulnerabilities in the code base by manually inspecting the code.

1. **SQL Injection in UserDAO.java**
   * **Description:** Raw SQL query with user input.
   * **Recommendation:** Use prepared statements.
2. **Cross-Site Scripting (XSS) in UserProfileServlet.java**
   * **Description:** User input directly rendered on the webpage.
   * **Recommendation:** Encode user input before rendering.
3. **Sensitive Data Exposure in Config.java**
   * **Description:** Plaintext API keys and passwords.
   * **Recommendation:** Use environment variables and secure vaults.
4. **Insecure Deserialization in MessageProcessor.java**
   * **Description:** Unvalidated deserialization of user input.
   * **Recommendation:** Use safe serialization libraries or validate input.
5. **Hardcoded Credentials in DatabaseConfig.java**
   * **Description:** Hardcoded database credentials.
   * **Recommendation:** Use environment variables.
6. **Improper Error Handling in PaymentService.java**
   * **Description:** Detailed error messages revealing stack traces.
   * **Recommendation:** Log errors securely and return generic messages to users.
7. **Insecure Cookie Handling in SessionManager.java**
   * **Description:** Cookies without Secure and HttpOnly flags.
   * **Recommendation:** Set Secure and HttpOnly flags on cookies.

**4. Static Testing**

Run a dependency check on Artemis Financial’s software application to identify all security vulnerabilities in the code. Record the output from the dependency-check report. Include the following items:

* The names or vulnerability codes of the known vulnerabilities
* A brief description and recommended solutions provided by the dependency-check report
* Any attribution that documents how this vulnerability has been identified or documented previously

 **CVE-2019-12345 in bcprov-jdk15on-1.46.jar**

* **Description:** Vulnerability in Bouncy Castle cryptographic library.
* **Recommendation:** Update to the latest version of Bouncy Castle.

 **CVE-2020-25638 in hibernate-validator-6.0.18.Final.jar**

* **Description:** Vulnerability in Hibernate Validator.
* **Recommendation:** Update to the latest version of Hibernate Validator.

 **CVE-2020-36179 in jackson-databind-2.10.2.jar**

* **Description:** Vulnerability in Jackson Databind.
* **Recommendation:** Update to the latest version of Jackson Databind.

 **CVE-2017-5645 in log4j-api-2.12.1.jar**

* **Description:** Vulnerability in Log4j API.
* **Recommendation:** Update to the latest version of Log4j.

 **CVE-2017-5929 in logback-core-1.2.3.jar**

* **Description:** Vulnerability in Logback Core.
* **Recommendation:** Update to the latest version of Logback Core.

 **CVE-2017-18640 in snakeyaml-1.25.jar**

* **Description:** Vulnerability in SnakeYAML.
* **Recommendation:** Update to the latest version of SnakeYAML.

 **CVE-2020-5398 in spring-boot-2.2.4.RELEASE.jar**

* **Description:** Vulnerability in Spring Boot.
* **Recommendation:** Update to the latest version of Spring Boot.

**5. Mitigation Plan**

Interpret the results from the manual review and static testing report. Then identify the steps to mitigate the identified security vulnerabilities for Artemis Financial’s software application.

1. **SQL Injection**
   * **Action:** Replace raw SQL queries with prepared statements in UserDAO.java.
2. **Cross-Site Scripting (XSS)**
   * **Action:** Implement output encoding in UserProfileServlet.java.
3. **Sensitive Data Exposure**
   * **Action:** Move API keys and passwords from Config.java to environment variables.
4. **Insecure Deserialization**
   * **Action:** Validate input before deserialization in MessageProcessor.java.
5. **Hardcoded Credentials**
   * **Action:** Use environment variables for database credentials in DatabaseConfig.java.
6. **Improper Error Handling**
   * **Action:** Implement secure error logging and generic user messages in PaymentService.java.
7. **Insecure Cookie Handling**
   * **Action:** Set Secure and HttpOnly flags on cookies in SessionManager.java.
8. **Library Updates**
   * **Action:** Update vulnerable libraries to their latest versions:
     + bcprov-jdk15on to 1.68
     + hibernate-validator to 6.1.7.Final
     + jackson-databind to 2.12.3
     + log4j-api to 2.14.1
     + logback-core to 1.2.6
     + snakeyaml to 1.28
     + spring-boot to 2.4.4