# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **07/20/24** | **Lillian Berry** |  |

## 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

Lillian Berry

**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:** Secure communications are paramount for Artemis Financial to safeguard sensitive financial information, maintain customer trust, and comply with regulatory requirements.

**International Transactions:** If Artemis Financial conducts international transactions, compliance with international standards and regulations such as GDPR for the EU or CCPA for California is necessary. This includes secure data transfer and storage practices.

**Governmental Restrictions:** Different regions impose varying requirements for data protection and secure communications. For example, the EU enforces stringent data protection laws (GDPR), while the US has sector-specific regulations like GLBA for financial institutions. Understanding and adhering to these requirements is critical.

**External Threats:**

* **SQL Injection:** Attackers might exploit vulnerabilities in database queries.
* **Cross-Site Scripting (XSS):** Malicious scripts could be injected into web pages.
* **Cross-Site Request Forgery (CSRF):** Attackers might trick users into performing unintended actions.
* **Man-in-the-Middle Attacks:** Intercepting communications between users and the application.
* **Data Breaches:** Unauthorized access to sensitive financial data.

**Modernization Requirements:**

* **Role of Open-Source Libraries:** Ensure that any open-source libraries used are up-to-date and free from known vulnerabilities.
* **Evolving Web Technologies:** Stay updated with the latest security practices and technologies to protect against emerging threats.

**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.

* **Input Validation:** Ensure secure input and representations to prevent injection attacks.
* **APIs:** Secure API interactions to prevent unauthorized access and data breaches.
* **Cryptography:** Implement robust encryption to protect sensitive data.
* **Client/Server:** Secure distributed components to protect communication channels.
* **Code Error:** Implement secure code handling to prevent vulnerabilities due to coding errors.
* **Code Quality:** Ensure secure coding practices and patterns to maintain a robust codebase.
* **Encapsulation:** Secure data structures to prevent unauthorized access and manipulation.

**3. Manual Review**

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

During the manual review of the codebase, the following vulnerabilities were identified:

1. **SQL Injection Vulnerabilities:** Found in the database query construction within DataAccess.java.
2. **XSS Vulnerabilities:** Present in the ViewHandler.java where user inputs are directly rendered.
3. **CSRF Vulnerabilities:** Lack of anti-CSRF tokens in forms within FormHandler.java.
4. **Insecure Direct Object References:** Found in FileHandler.java, where file paths are constructed based on user input.
5. **Insecure Cryptographic Storage:** Weak encryption algorithms identified in CryptoService.java.
6. **Misconfigured Security Settings:** Default configurations in ServerConfig.java expose sensitive information.
7. **Unvalidated Redirects and Forwards:** Found in RedirectHandler.java, where URLs are constructed based on user input.

**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

Using the Maven Dependency-Check Plug-in, the following vulnerabilities were identified in the dependencies:

**Dependency:** bcprov-jdk15on-1.46.jar

* **Package:** org.bouncycastle
* **Highest Severity:** HIGH
* **CVE Count:** 22
* **Description:** Bouncy Castle is a collection of APIs used in cryptography. Older versions are vulnerable to various security issues
* **Recommended Solution:** Upgrade to a newer version of the Bouncy Castle library.

**Dependency:** hibernate-validator-6.0.18.Final.jar

* **Vulnerability IDs:** CVE-2020-10693
* **Package:** org.hibernate.validator
* **Highest Severity:** MEDIUM
* **CVE Count:** 1
* **Decription:** hibernate validator is used for bean validation. Specific versions are susceptible to issues such as validation bypass
* **Recommended Solution:** Upgrade to a newer version of Hibernate Validator.

**Dependency:** jackson-databind-2.10.2.jar

* **Package:** com.fasterxml.jackson.core
* **Highest Severity:** HIGH
* **CVE Count:** 6
* **Description:** Jackson databind is used for processing JSON. Older versions can be exploiuted to execute arbitrary code
* **Recommended Solution:** Upgrade to a newer version of Jackson Databind.

**Dependency:** logback-core-1.2.3.jar

* **Package:** ch.qos.logback
* **Highest Severity:** HIGH
* **CVE Count:** 2
* **Description:** logback is a logging framework. Certain versions have vulnerabilities that allow remote code execution
* **Recommended Solution:** Upgrade to a newer version of Logback.

**Dependency:** snakeyaml-1.25.jar

* **Package:** org.yaml
* **Highest Severity:** CRITICAL
* **CVE Count:** 8
* **Description:** SnakeYAML is a YAML parser. Older versions can be exploited for code injection attacks
* **Recommended Solution:** Upgrade to a newer version of SnakeYAML.

**Dependency:** spring-boot-2.2.4.RELEASE.jar

* **Package:** org.springframework.boot
* **Highest Severity:** CRITICAL
* **CVE Count:** 3
* **Description:** Spring boot simplifies application configuration. Some versions have vulnerabilities that allow remote code execution
* **Recommended Solution:** Upgrade to a newer version of Spring Boot.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**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.

**Manual Review Vulnerabilities:**

1. **SQL Injection:** Use prepared statements and parameterized queries.
2. **XSS:** Implement output encoding/escaping.
3. **CSRF:** Implement anti-CSRF tokens.
4. **Insecure Direct Object References:** Validate and sanitize file paths.
5. **Insecure Cryptographic Storage:** Use stronger encryption algorithms (e.g., AES-256).
6. **Misconfigured Security Settings:** Review and secure default configurations.
7. **Unvalidated Redirects and Forwards:** Validate and restrict URLs used for redirection.

**Static Testing Vulnerabilities:**

1. **bcprov-jdk15on:** Upgrade to the latest version.
2. **hibernate-validator:** Upgrade to the latest version.
3. **jackson-databind:** Upgrade to the latest version.
4. **logback-core:** Upgrade to the latest version.
5. **snakeyaml:** Upgrade to the latest version.
6. **spring-boot:** Upgrade to the latest version.