# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **5/26/2024** | **Karina Aronov** | **Interpreting Client Needs, Areas of Security, Manual Review, Static Testing, Mitigation Plan** |

## Client



## Developer

Karina Aronov

**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?
  + The value of secure communication is essential to Artemis Financial in order to protect sensitive financial and personal data and confidentiality. Any mistake could result in serious damages to one's finances and reputation.
* Are there any international transactions that the company produces?
  + It does not state that Artemis Financial is an organization that is based in the United States or elsewhere, but we can assume that most likely they will deal with international transactions.
* Are there governmental restrictions on secure communications to consider?
  + The organization will have to adhere to several governmental restrictions as the US alone requires compliance with several laws depending on the state, industry, and data storage type (Brands, 2024). If this company is international and will be dealing international transactions, there are other restrictions they would have to adhere to.
* What external threats might be present now and in the immediate future?
  + Potential threats may include Phishing, SQL Injection Attacks, Cross-site Scripting (XSS), Denial of Service (DoS), Man-in-the-Middle Attacks (MitM), Insider Threats, and Zero-Day Attacks (Toohil, 2023).
* What modernization requirements must be considered, such as the role of open-source libraries and evolving web application technologies?
  + Using open-source libraries can speed up development but can introduce vulnerabilities if it is not properly managed and updated. Performing thorough code reviews to detect potential threats is vital.
  + Evolving web application technologies require strong security practices in order to decrease any new types of vulnerabilities. Including secure coding and frameworks that provide built-in security features can help prevent threats.

**2. Areas of Security**

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

* Analyzing the application architecture is important for identifying design-level vulnerabilities and ensuring the overall security framework aligns with best practices.
* By validating inputs, the application can defend against common attacks like SQL injection and XSS, which are significant threats to financial data integrity.
* Since Artemis Financial uses a RESTful web API, ensuring secure API interactions prevents unauthorized access and data breaches.
* Protecting sensitive financial information during storage and transmission is essential. Effective cryptographic practices ensure data confidentiality and ethics.
* Proper error handling can prevent attackers from gaining insights into the system’s weaknesses and helps maintain the application's stability and security.
* High code quality reduces the likelihood of introducing security flaws during development, ensuring the application is strong against attacks.

**3. Manual Review**

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

* **CRUD.java**
* The constructor that sets both ‘content’ and ‘content2’ to the same value is redundant and unnecessary.
* **CRUDController.java**
  + Accepting untrusted input and using it directly in the response without validation can lead to expression language injection. Validating all user inputs is important to prevent injection attacks.
* **Customer.java**
* The ‘account\_balance’ field has package-private access, making it accessible to other classes in the package, potentially leading to unauthorized access. Using method that provide private access modifiers and controlled access can reduce unauthorized access.
* The ‘deposit’ method does not validate the input, allowing for negative values to be deposited, which can cause incorrect account balances. Validating input values to ensure they are within the expected range are vital.
* **DocData.java**
* The ‘read\_document’ method uses unvalidated input for database operations, which can lead to SQL injection attacks.
* **GreetingController.java**
* The ‘name’ parameter is not limited in size, which could be exploited by sending a very large input, potentially leading to a DoS attack.
* **Pom.xml**
* The dependency on ‘bcprov-jdk15on’ version 1.46 is outdated and may have known vulnerabilities. Update to the latest version of the library to ensure all security patches are applied.

**4. Static Testing**

Run a dependency check on Artemis Financials’ 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

A screenshot of a service

Description automatically generated

A screenshot of a computer

Description automatically generated

A black background with colorful text

Description automatically generated with medium confidence

The dependency check report came back with 13 vulnerable dependencies with a total of 86 vulnerabilities having been found. There were some duplicates that I had removed. Below I have listed 10 significant vulnerable dependencies with their description and published vulnerability.

1. **log4j-api-2.12.1.jar**
   * **Description**: The Apache Log4j API
   * **Published Vulnerability: CVE-2020-9488**
   * **Recommended Solutions: Upgrade to version 2.13.2 or later. Ensure that log4j configurations are properly secured and do not log sensitive information.**
2. **snakeyaml-1.25.jar**
   * **Description**: YAML 1.1 parser and emitter for Java
   * **Published Vulnerability: CVE-2017-18640**
   * **Recommended Solutions: Upgrade to version 1.26 or later. Validate and sanitize YAML input to ensure it does not contain malicious content.**
3. **jackson-databind-2.10.2.jar**
   * **Description**: General data-binding functionality for Jackson: works on core streaming API
   * **Published Vulnerability: CVE-2020-25649**
   * **Recommended Solutions: Upgrade to version 2.11.0 or later. Configure Jackson to only allow JSON data to be converted to trusted types.**
4. **tomcat-embed-core-9.0.30.jar**
   * **Description**: Core Tomcat implementation
   * **Published Vulnerability: CVE-2020-17569**
   * **Recommended Solutions: Upgrade to version 9.0.36 or later. Regularly review and update Tomcat configurations to enforce secure defaults.**
5. **hibernate-validator-6.0.18.Final.jar**
   * **Description**: Hibernate's Bean Validation (JSR-380) reference implementation.
   * **Published Vulnerability: CVE-2020-10693**
   * **Recommended Solutions: Upgrade to version 6.1.0 or later. Ensure all input data is properly validated using updated validation constraints.**
6. **spring-web-5.2.3.RELEASE.jar**
   * **Description**: Spring Web
   * **Published Vulnerability: CVE-2016-1000027 (OSSINDEX)**
   * **Recommended Solutions: Upgrade to version 5.2.5.RELEASE or later. Apply secure coding practices for web applications to minimize risks.**
7. **spring-beans-5.2.3.RELEASE.jar**
   * **Description**: Spring Beans
   * **Published Vulnerability: CVE-** **2022-22965 (OSSINDEX)**
   * **Recommended Solutions: Upgrade to version 5.2.20.RELEASE or later. Review and apply recommended security configurations for Spring applications.**
8. **spring-webmvc-5.2.3.RELEASE.jar**
   * **Description**: Spring Web MVC
   * **Published Vulnerability: CVE-** **2021-22060 (OSSINDEX)**
   * **Recommended Solutions: Upgrade to version 5.3.15 or later. Ensure strong input validation and sanitization in web controllers.**
9. **spring-context-5.2.3.RELEASE.jar**
   * **Description**: Spring Context
   * **Published Vulnerability: CVE-** **2022-22968 (OSSINDEX)**
   * **Recommended Solutions: Upgrade to version 5.2.20.RELEASE or later. Regularly review and update Spring security configurations.**
10. **spring-expression-5.2.3.RELEASE.jar**
    * **Description**: Spring Expression Language (SpEL)
    * **Published Vulnerability: CVE-** **2022-22950 (OSSINDEX)**
    * **Recommended Solutions: Upgrade to version 5.3.15 or later. Validate and sanitize expressions to prevent injection attacks.**

**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 Financials’ software application.

1. **Code Changes and Reviews**
   * Remove redundant constructors in CRUD.java.
   * Implement input validation and sanitization in CRUDController.java and customer.java.
   * Use private access modifiers and controlled access methods in customer.java.
   * Use prepared statements in DocData.java to prevent SQL injection.
   * Limit input size in GreetingController.java to prevent DoS attacks.
2. **Dependency Management**

* Regularly update dependencies to the latest versions to apply security patches.
* Use tools like OWASP Dependency-Check to identify and mitigate vulnerable dependencies.

1. **Implement Secure Coding Practices**

* Implement secure coding guidelines.
* Conduct regular code reviews and static code analysis.
* Apply proper error handling and logging mechanisms.

1. **Continuous Monitoring and Testing**

* Regularly perform security audits and testing.
* Monitor application and dependency vulnerabilities using automated tools.

**References:**

Brands, M. (2024, May 6). 2024-05-06-Cybersecurity laws and legislation (2024 update). *ConnectWise*. https://www.connectwise.com/blog/cybersecurity/cybersecurity-laws-and-legislation

Toohil, R. (2023, January 19). *The 21 Latest Emerging Cyber Threats & Attacks (NEW)*. https://www.aura.com/learn/emerging-cyber-threats