# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **1/26/25** | **Jennifer Swinton** |  |

## 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

Jennifer Swinton

**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?

Secure communications are absolutely a priority for Artemis Financial. They regularly deal with confidential client information including SSN, individual’s investments, bank account information, financial information and insurance details. Secure channels are critical to prevent unauthorized access and increase customer confidence. Based on the limited information given, it is safe to assume that there are international transactions that the company produces, and in the financial world, there are usually heavy government restrictions and limitations from HIPAA, kickback laws, insider trading laws and more will need to be kept in mind as well as any and all data inscription regulations. External threats include direct threats like hacking, but less direct such as phishing attacks, malicious user attacks, unauthorized access data breaches, and disruption of servers to disturb business. For modernization, the company would need ongoing and regular updates. They would need to ensure all the systems are on the most recent available versions and that they constantly scan for vulnerabilities. They would want to ensure encryption techniques are the most modern and that the API security is as recent and tightened down as possible, including input validations and authentications mechanisms are in place.

**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.

Reviewing Artemis Financial areas of security, we can see several potential vulnerabilities.

* This program allows input. Therefore, validation is essential to avoid points of failure.
* Since there is both internal and external points, a well-developed API is essential. The API will define end user interaction so it should determine methods available for data access.
* Cryptography – Cryptography is necessary as international transfers with proprietary customer information will be included with the transfers. The data should be secured in a fashion that can comply to both North American regulation and any regulation to its destination country.
* Working in combination with the API and Input validation should be Code Error, or error handling. This will be critical in understanding what areas need repair.
* Code quality is important when working to ensure that data exposure is prevented. It will keep the unauthorized access to a minimum but also ensure that the latencies from poor processing is reduced.

**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. **Architecture Review:**
   * In RestServiceApplicationTests the contextLoads() method is empty. This limits the ability to thoroughly test if the Spring Boot application context loads without issues.
2. **Input Validation:**
   * In GreetingController the greeting() method doesn't perform any validation on the name parameter, which could lead to issues.
3. **Input Validation:**
   * In CRUDController the CRUD() method is also not validating the business\_name parameter.
4. **APIs:**
   * In CRUDController (file: CRUDController.java), the CRUD method doesn't expose a terministic URI for RESTful principles, which might cause issues in the API design.
5. **Cryptography:**
   * No cryptographic handling can be seen.
6. **Client/Server Security:**
   * In DocData the database connection details are hardcoded, which could expose sensitive information if the source code is exposed.
7. **Code Error:**
   * In DocData there is an unhandled SQLException, which could cause crashes or inconsistent behavior.
8. **Code Quality:**
   * The class customer does not follow standard Java naming conventions (i.e., Customer instead of customer).
9. **Code Quality:**
   * Methods in myDateTime are not implemented, which might lead to incomplete or incorrect functionality.

**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

|  |  |  |  |
| --- | --- | --- | --- |
| Names/ vulnerability codes | Brief description | Recommended solutions | Attribution if applicable |
| bcprov-jdk15on-1.46.jar | Various vulnerabilities in Bouncy Castle library versions. | Update to the latest version of Bouncy Castle. | NVD |
| spring-boot-2.2.4.RELEASE.jar | Several critical vulnerabilities affecting Spring Boot. | Upgrade to the latest stable release of Spring Boot. | NVD |
| logback-classic-1.2.3.jar | High severity vulnerabilities in Logback Classic library. | Update to the latest version of Logback. | NVD |
| logback-core-1.2.3.jar | Multiple vulnerabilities in Logback Core library. | Upgrade to the latest version of Logback. | NVD |
| log4j-api-2.12.1.jar | Critical vulnerabilities in Log4j API. | Update to the most recent version of Log4j. | NVD |
| snakeyaml-1.25.jar | Vulnerabilities in SnakeYAML library. | Upgrade SnakeYAML to the latest version. | NVD |
| jackson-databind-2.10.2.jar | High risk vulnerabilities in Jackson Databind library. | Update Jackson Databind to the latest stable version. | NVD |

**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.

The primary and fastest fix would be to update the items from the dependency check to their latest versions as indicated in the table above. This will probably result is code adjustments to be done where some of the other fixes can be made, such as

* Ensure test methods like contextLoads() in RestServiceApplicationTests are implemented fully to validate the Spring Boot application.
* Add the proper input validations to prevent injection attacks.
* Ensure the CRUDController follows RESTful API design.
* Implement proper cryptographic handling wherever sensitive data is processed or transferred and remove all hardcoded details in DocData to manage sensitive information.
* Ensure all exceptions are properly handled to avoid application crashes or inconsistent behavior.
* Ensure the classes follow Java naming conventions, and fully implement methods in myDateTime to ensure complete and correct functionality.

References:

National Institute of Standards and Technology (NIST). (n.d.). National Vulnerability Database (NVD). U.S. Department of Commerce. Retrieved from <https://nvd.nist.gov>.