# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **7/15/2024** | **Shae Machlus** | **First draft** |

## 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

Shae Machlus

**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?
* Artemis Financial provides individualized financial plans to customers for savings, retirement, investments, and insurance. Security is extremely important to the company since their services contain sensitive information like account numbers, routing numbers, social security numbers, and answers to security questions.
* Though not mentioned explicitly, Artemis Financial does not seem limited from making international transactions.
* The government regulates the financial industry because it is the source of inflation, and therefore can influence interest rates.
* The stock market is unpredictable, so investments in stocks or mutual funds are at risk of losing customer money depending on the performance of the assets. Additionally, disaster may strike and if the customer is caught without an insurance plan they may lose money, whether the disaster incurs property damage or unexpected health issues.
* Since software is constantly evolving, the agile scrum team should stay up to date on their versions of open-source libraries and web frameworks. The updates of these tools should be implemented on a continuous basis to ensure bugs of older editions are resolved and do not affect the software product.

**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.

* Secure API Interactions: the database and backend should correspond securely to the frontend where sensitive information is displayed. The API should be robust to man-in-the-middle attacks which could eavesdrop on these communications and steal customer information.
* Cryptography: within the database, the data should be hidden using cryptography. For example, an SQL injection attack could allow a hacker to gain access to the database. They could steal data, but would be unable to decrypt it, ensuring the safety of the sensitive information.
* Client/Server: the network connection between the client and the server should be secure and robust against distributed denial of service (DDoS) attacks. If a client were able to overwhelm the server with malicious code, the server would need to shut down and/or be in a vulnerable state. The hacker would more easily be able to infiltrate the server to destroy or steal information.

**3. Manual Review**

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

* Input validation: RestServiceApplication.java takes in String[] args for the main function, but these inputs are not validated.
* API’s: no API’s were found in any of the .java files.
* Cryptography: no cryptography was used in any of the .java files.
* Client/Server: no client/server interactions take place in the code
* Code Error: the code runs without error.
* Code Quality: the code is efficient and well documented. The code quality is excellent.
* Encapsulation: the account number is appropriately a private int in the customer class. The ID and content in the class Greeting are both private, too. In the GreetingController, the template welcome string is private as is the AtomicLong object named counter.

**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:

|  |  |  |
| --- | --- | --- |
| Dependency | Description | Recommended Solutions |
| bcprov-jdk15on-1.46.jar | The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms. This jar contains JCE provider and lightweight API for the Bouncy Castle Cryptography APIs for JDK 1.5 to JDK 1.7. | Check that the certificate provided by the host is actually associated with the host. |
| hibernate-validator-6.0.18.Final.jar | Hibernate's Bean Validation (JSR-380) reference implementation. | Implement a stricter input sanitization routine so that attackers cannot inject malicious data. |
| jackson-databind-2.10.2.jar | General data-binding functionality for Jackson: works on core streaming API | Secure the XML document and ensure data security by using cryptography. |
| log4j-api-2.12.1.jar | The Apache Log4j API | Upgrade to Apache Log4j 2.12.3 or 2.13.1 |
| logback-core-1.2.3.jar | logback-core module | Upgrade to a version after 1.4.11 to avoid a DoS vulnerability. |
| snakeyaml-1.25.jar | YAML 1.1 parser and emitter for Java | Upgrade to version 2.0+ |
| spring-boot-2.2.4.RELEASE.jar | Spring Boot | 2.7.x users should upgrade to 2.7.11+ and 3.0.x users should upgrade to 3.0.6+. |
| spring-boot-starter-web-2.2.4.RELEASE.jar | Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container | 2.7.x users should upgrade to 2.7.11+ and 3.0.x users should upgrade to 3.0.6+. |
| spring-core-5.2.3.RELEASE.jar | Spring Core | Apply updates per vendor’s instructions. |
| spring-web-5.2.3.RELEASE.jar | Spring Web | Upgrade to a version after 5.3.16. |
| spring-webmvc- 5.2.3.RELEASE.jar | Spring Web MVC | Apply updates per vendor’s instructions. |
| tomcat-embed-core-9.0.30.jar | Core Tomcat implementation | Apply updates per vendor instructions. |
| tomcat-embed-websocket-9.0.30.jar | Core Tomcat implementation | Apply updates per vendor instructions. |

**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 code which was manually reviewed had some missing pieces. There was no input sanitization, no API’s, no cryptography, and no information about networking. The code that was provided was excellent, just incomplete. Encapsulation was used to keep some sensitive variables like user information and account numbers secure. From the static testing, the general message was to stay up to date on the releases of the dependencies used in the software. With dated versions of dependencies, the system is susceptible to DoS attacks, external entity attacks, man-in-the-middle attacks, and others. These are all avoidable vulnerabilities if the dependencies are kept up to date. The mitigation plans for each dependency are presented in the third column of the previous section.