# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **7/18/2024** | **Andre Freitas** |  |

## 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

Andre Freitas

**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?

Security is paramount for Artemis Financial. As a company that handles customer’s savings, retirements, investments, and insurance there is a lot of sensitive information that the company deals with. Having a breach could be devastating to both their reputation and their customer’s information.

There are no mentions of international transactions in the company blurb, but we should assume they would want to have such a thing included if not mentioned. Ideally we should ask the company if such a feature is needed.

A company would need to follow all governmental restrictions that are in place. These should be taken into account when designing the software.

Bad actors or hackers are always going to be looking for a way to obtain sensitive information. Whether through phishing, direct hacking attempts, or impersonation of clients it is vital that we be aware of how to mitigate and minimize these attempts as much as possible.

We and Artemis Financial must ensure that we continue using the most up-to-date version of whatever web application use. Updates often involve improved security features or fixes. Mobile versions of the site should be considered as the vulnerabilities of someone on a phone are different than an individual on a computer.

**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**: Secure input validation is a key element of protecting a customer’s information. Ensuring that the customer’s information is secure and ensures that they are not protected by weak passwords or info that is easy to guess.

**Code Error:** Having good code is a requirement in any program. If the code breaks, then the company may not be able to continue with the services they provide. This can cause a big reputation hit for Artemis Financial and also cause a great deal of distress to customers who may have needed something urgently done.

**Code Quality:** Being able to read and update code as needed is always necessary when handling live updates. If the code is messy or difficult to parse, it can be hard to see where code errors may occur. Likewise, if an update in one spot breaks another part of the code it can prolong errors and make updates difficult.

**Data Encryption:** Dealing with currencies and sensitive customer information means that encrypting said info is recommended. Encryptions add an extra layer of protection against hackers to keep a customer’s info safe.

**3. Manual Review**

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

* There is no user authentication
* There is no input validation
* Most recent version of Maven not used
* The most recent version of several dependencies are not being used

**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
* **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.
  + We should update this to the latest version rather than stay at 1.46.
* **hibernate-validator-6.0.18.Final.jar**
  + Hibernate's Bean Validation (JSR-380) reference implementation.
  + Allows invalid expressions to be read as valid. Update to latest version
* **jackson-databind-2.10.2.jar**
  + General data-binding functionality for Jackson: works on core streaming API
  + Data Integrity at risk from XML external entity attacks. Update to latest version
* **log4j-api-2.12.1.jar**
  + The Apache Log4j API
  + Host mismatch during validation. Vulnerable to man in the middle attack. Solved by updating to later version.
* **logback-core-1.2.3.jar**
  + logback-core module
  + Serialization vulnerability in version. Update to later version
* **snakeyaml-1.25.jar**
  + YAML 1.1 parser and emitter for Java
  + Does not restrict types that can be instantiated during deserialization. Update to 2.0 or later
* **spring-boot-2.2.4.RELEASE.jar**
  + Spring Boot
  + Application is deployed to Cloud Foundry and susceptible to security bypass. Update to later versions.
* **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
  + Application is deployed to Cloud Foundry and susceptible to security bypass. Update to later versions.
* **spring-core-5.2.3.RELEASE.jar**
  + Spring Core
  + Vulnerable to remove code execution via data binding. Update to more recently supported version
* **spring-web-5.2.3.RELEASE.jar**
  + Spring Web
  + Remote code execution from using untrusted data. Update to more recent version
* **spring-webmvc-5.2.3.RELEASE.jar**
  + Spring Web MVC
  + Remote code execution via data binding. Update to supported version
* **tomcat-embed-core-9.0.30.jar**
  + Core Tomcat implementation
  + If specific mechanism is not disabled, a vulnerability on the IP may occur. Later versions set this to off by default, update to later version.
* **tomcat-embed-websocket-9.0.30.jar**
  + Core Tomcat implementation
  + If specific mechanism is not disabled, a vulnerability on the IP may occur. Later versions set this to off by default, update to later version.

**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.

There are several dependencies that need to be updated to their most recent version. For us to ensure the greatest security, all of these should be updated to their latest version and also be continuously updated as more versions are rolled out. The most recent version of Maven should also be used as if it is not the dependency checks may read incorrect results.