# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **7/21/24** | **Bee Best** |  |

## 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

Bee Best

**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?

Because Artemis Financial is (as the name states,) a financial company, secure communications are imperative to the livelihood of their business due to the sensitivity of the information being communicated, this information being financial information relating to their savings, their investments, their insurance, and personal information (bank info, social security, full name, etc.) relating to the clients. Artemis Financial runs their business “around the world” and as such will need to be complying with all major security laws present around the globe. The external threats present would be any data breaches targeted towards client information, as such the data should be kept securely and encrypted in such a way that a data breach would not allow attackers to get any sensitive data. Open-Source libraries and keeping everything up to date would be beneficial as open source things tend to be the most up to date when it comes to fixing security vulnerabilities.

**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: Input validation applies to Artemis Financial because they require their inputs to be validated to protect against attacks, essentially it provides more security.

APIs: An API is necessary because the software can be used by people that aren’t internal to the company so having a way to ensure that users are only allowed access to the information that they should have access to is important.

Cryptography: This is heavily important because of how sensitive the data involved is, a breach in raw data could result in extremely large losses and as such cryptography must be applied to ensure that should any data breach occur, the data gained will provide little to no value to the attacker.

Client/Server: We need to ensure that data is protected, for example with HTTP requests, we have to ensure that the data of our system is protected.

Code Error: This is necessary for Artemis Financial because all of the code involved must be reviewed and created in a way such that errors are not an issue present in the resulting software.

Code Quality: All code must be designed in a secure manner from the beginning to prevent oversights.

Encapsulation: All information in the system must be protected via encapsulation, essentially, we do not want the data accessed from the incorrect channels as that is a security risk.

**3. Manual Review**

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

The things I noticed from looking through the code base were that in the customer class, account\_balance is left public, in GreetingColtroller, the string input is not validated, in myDateTime the time is not private (leaving things, even simple things like this as public is bad practice), and the dependency check itself is outdated and as such it likely will not cover every vulnerability that is currently known. There were things as well that were simply not present throughout the code base such as error handling and cryptography (although it is possible that I just did not see the cryptography portions somehow).

**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

|  |  |  |
| --- | --- | --- |
| **Dependency** | **Vulnerability IDs** | **Description……………………………………………………………………………………………………………………..** |
| **bcprov-jdk15on-1.46.jar** | **cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java crytography-api:1.46:\*:\*:\*:\*:\*:\*:\*** | **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.** |
| **spring-boot-2.2.4.RELEASE.jar** | **cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\*** | **In Spring Boot versions 3.0.0 - 3.0.6, 2.7.0 - 2.7.11, 2.6.0 - 2.6.14, 2.5.0 - 2.5.14 and older unsupported versions, there is potential for a denial-of-service (DoS) attack if Spring MVC is used together with a reverse proxy cache.** |
| **logback-core-1.2.3.jar** | **cpe:2.3:a:qos:logback:1.2.3:\*:\*:\*:\*:\*:\*:\*** | **A serialization vulnerability in logback receiver component part of logback version 1.4.11 allows an attacker to mount a Denial-Of-Service attack by sending poisoned data.** |
| **log4j-api-2.12.1.jar** | **cpe:2.3:a:apache:log4j:2.12.1:\*:\*:\*:\*:\*:\*:\*** | **Apache Log4j2 versions 2.0-beta7 through 2.17.0 (excluding security fix releases 2.3.2 and 2.12.4) are vulnerable to a remote code execution (RCE) attack when a configuration uses a JDBC Appender with a JNDI LDAP data source URI when an attacker has control of the target LDAP server. This issue is fixed by limiting JNDI data source names to the java protocol in Log4j2 versions 2.17.1, 2.12.4, and 2.3.2.** |
| **snakeyaml-1.25.jar** | **cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\***  **cpe:2.3:a:yaml\_project:yaml:1.25:\*:\*:\*:\*:\*:\*:\*** | **SnakeYaml's Constructor() class does not restrict types which can be instantiated during deserialization. Deserializing yaml content provided by an attacker can lead to remote code execution. We recommend using SnakeYaml's SafeConsturctor when parsing untrusted content to restrict deserialization. We recommend upgrading to version 2.0 and beyond.**  **Parsing malicious or large YAML documents can consume excessive amounts of CPU or memory.** |
| **jackson-databind-2.10.2.jar** | **cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\*** | **jackson-databind through 2.15.2 allows attackers to cause a denial of service or other unspecified impact via a crafted object that uses cyclic dependencies. NOTE: the vendor's perspective is that this is not a valid vulnerability report, because the steps of constructing a cyclic data structure and trying to serialize it cannot be achieved by an external attacker.** |
| **tomcat-embed-core-9.0.30.jar** | **cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\***  **cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*** | **Generation of Error Message Containing Sensitive Information vulnerability in Apache Tomcat.This issue affects Apache Tomcat: from 8.5.7 through 8.5.63, from 9.0.0-M11 through 9.0.43. Users are recommended to upgrade to version 8.5.64 onwards or 9.0.44 onwards, which contain a fix for the issue.** |
| **hibernate-validator-6.0.18.Final.jar** | **cpe:2.3:a:redhat:hibernate\_validator:6.0.18:\*:\*:\*:\*:\*:\*:\*** | **A flaw was found in Hibernate Validator version 6.1.2.Final. A bug in the message interpolation processor enables invalid EL expressions to be evaluated as if they were valid. This flaw allows attackers to bypass input sanitation (escaping, stripping) controls that developers may have put in place when handling user-controlled data in error messages.** |
| **spring-web-5.2.3.RELEASE.jar** | **cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\***  **cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*** | **No Description given** |
| **spring-beans-5.2.3.RELEASE.jar** | **cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\***  **cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*** | **No Description given** |
| **spring-webmvc-5.2.3.RELEASE.jar** | **cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\***  **cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*** | **No Description given** |
| **spring-context-5.2.3.RELEASE.jar** | **cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\***  **cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*** | **No Description given** |
| **spring-expression-5.2.3.RELEASE.jar** | **cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\***  **cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*** | **No Description given** |

**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 best way to fix many of the vulnerabilities would be to simply update the dependencies that are outdated.