# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **3/22/2025** | **Shayne Greene** |  |

## 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

Shayne Greene

**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 holds secure communications to the company highly. Security is vital to the success of their web-applications. Since the company handles financial transactions, there will be immediate threats to sensitive data being transmitted back and forth. Other external threats are DDOS attacks that could take down not only the application, but also the institution that is hosting the applications data. When dealing with financial transactions, they will have governmental restrictions for the use and security of their web applications. Some modernizations to be considered could be to migrate to use a gov cloud to host the API. Also, the use of authentication for access to the API should be considered.

**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.

Areas of security for this web application are:

**Code Errors -** Errors can lead to vulnerabilities within the syntax of the source code.

**API’s** - The web app utilizes a RESTful API.

**Input validation** - Within the use of the RESTful API, there will be input that should be validated for malicious code injections.

**Code Quality** – There will be a need for secure structured code to validate and authenticate users credentials.

**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 are no authentication processes for the use of the API.

- Hardcoded password for SQL database in DocData class. Which is also documented in the comments: (root, root)

- Class myDateTime has all public fields that are not set to final.

- myDateTime is missing constructor method. (Even though Java will automatically generate one for you)

- GreetingController class has no input validations for the get: “/greeting” name parameter.

- CrudController class has no input validations for the get: “/read” name parameter.

- In DocData, id field is not set to final.

- DocData class has no accessors to set the “id” field for the instance.

- DocData constructor is missing the id declaration

**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 - CVE-2024-34447**

The software communicates with a host that provides a certificate, but the software does not properly ensure that the certificate is actually associated with that host.

Upgrade to version 1.70

**hibernate-validator-6.0.18.Final.jar - CVE-2023-1932**

A flaw was found in hibernate-validator's 'isValid' method in the org.hibernate.validator.internal.constraintvalidators.hv.SafeHtmlValidator class, which can be bypassed by omitting the tag ending in a less-than character. Browsers may render an invalid html, allowing HTML injection or Cross-Site-Scripting (XSS) attacks.

Upgrade to version 9.0.0.CR1

**jackson-databind-2.10.2.jar** - **CVE-2020-25649**

A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks.

Upgrade to version 2.18.3

**log4j-api-2.12.1.jar - CVE-2020-9488**

Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender.

Upgrade to version 2.24.3

**logback-classic-1.2.3.jar - CVE-2023-6378**

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.

Upgrade to version 1.5.18

**logback-core-1.2.3.jar - CVE-2023-6378**

In logback version 1.2.7 and prior versions, an attacker with the required privileges to edit configurations files could craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers.

Upgrade to version 1.5.18

**snakeyaml-1.25.jar - CVE-2022-1471**

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.

Upgrade to version 2.4

**spring-boot-2.2.4.RELEASE.jar - CVE-2023-20873**

An application that is deployed to Cloud Foundry could be susceptible to a security bypass.

Upgrade to version 3.4.4

**spring-boot-starter-web-2.2.4.RELEASE.jar - CVE-2023-20873**

An application that is deployed to Cloud Foundry could be susceptible to a security bypass.

Upgrade to version 3.4.4

**spring-core-5.2.3.RELEASE.jar - CVE-2022-22965**

A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding.

Upgrade to version 7.0.0

**spring-expression-5.2.3.RELEASE.jar - CVE-2022-22965**

A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding.

Upgrade to version 7.0.0

**spring-web-5.2.3.RELEASE.jar - CVE-2016-1000027**

A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding.

Upgrade to version 7.0.0

**spring-webmvc-5.2.3.RELEASE.jar - CVE-2022-22965**

A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding.

Upgrade to version 7.0.0

**tomcat-embed-core-9.0.30.jar - CVE-2020-1938**

When using the Apache JServ Protocol (AJP), care must be taken when trusting incoming connections to Apache Tomcat. Tomcat treats AJP connections as having higher trust than, for example, a similar HTTP connection. If such connections are available to an attacker, they can be exploited in ways that may be surprising

Upgrade to version 11.0.5

**tomcat-embed-websocket-9.0.30.jar -**  **CVE-2020-1938**

When using the Apache JServ Protocol (AJP), care must be taken when trusting incoming connections to Apache Tomcat. Tomcat treats AJP connections as having higher trust than, for example, a similar HTTP connection. If such connections are available to an attacker, they can be exploited in ways that may be surprising (

upgrade to version 11.0.5

A screenshot of a phone service

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**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.

First step would be to add authentication protocols in order to access the API. This should be done for both the CrudController and GreetingController.

Input validation should also be placed anywhere the API accepts data inputs. We want to check for signs of malicious code injections.

Remove the hard coded SQL database credentials.

In the DocData class, update the fields to be private final and create public accessors for the variables.

In myDateTime class, set all fields to private and final. Update the class to include a default constructor, even if it is empty.

For each of the dependencies, please update the latest stable versions: Below are the dependencies to add to the pom.xml.

"<dependency>

<groupId>org.bouncycastle</groupId>

<artifactId>bcprov-jdk15on</artifactId>

<version>1.70</version>

</dependency>"

"<dependency>

<groupId>org.hibernate.validator</groupId>

<artifactId>hibernate-validator</artifactId>

<version>9.0.0.CR1</version>

</dependency>"

"<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.18.3</version>

</dependency>"

"<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-api</artifactId>

<version>2.24.3</version>

</dependency>"

"<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.5.18</version>

</dependency>"

"<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-core</artifactId>

<version>1.5.18</version>

</dependency>"

"<dependency>

<groupId>org.yaml</groupId>

<artifactId>snakeyaml</artifactId>

<version>2.4</version>

</dependency>"

"<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot</artifactId>

<version>3.4.4</version>

</dependency>"

"<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot</artifactId>

<version>3.4.4</version>

</dependency>"

"<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>7.0.0-M3</version>

</dependency>"

"<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-expression</artifactId>

<version>7.0.0-M3</version>

</dependency>"

"<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>7.0.0-M3</version>

</dependency>"

"<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>7.0.0-M3</version>

</dependency>"

"<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-core</artifactId>

<version>11.0.5</version>

</dependency>"

"<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-websocket</artifactId>

<version>11.0.5</version>

</dependency>"