# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **3/22/2025** | **Connor Holohan** |  |

## Client



## Developer

Connor Holohan

**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?

Artemis financial is an investment consulting company, and as such would have access to clients personal and banking information, along with access to investment portfolios. Maintaining the maximum level of security is a top concern from a legal, ethical, and company image standpoint.

* Are there any international transactions that the company produces?

From the information given I am uncertain. It wasn't specified if the company invests in non US stock markets. It also fails to mention if it only accepts US clients. If Artemis Financial did allow for Non US clients, considerations would need to be made for local laws and restrictions involving the banking systems of the clients home country.

* Are there governmental restrictions on secure communications to consider?

Data protection and privacy laws, each of which would need to be examined for compliance. Some states have specific rulings(California comes to mind), that would also need to be investigated.

* What external threats might be present now and in the immediate future?

Nefarious actors attempting to steal information regarding clients or system information for financial gain. Potentially state organized and funded groups considering attacking the US financial sector as an act of cyber warfare.

* What modernization requirements must be considered, such as the role of open-source libraries and evolving web application technologies?

Staying up to date with system updates and patches involving known or potential system vulnerabilities. Ensuring that encryption standards are met or surpassed. In terms of open source libraries, they can provide a robust testing ground, however care should be taken to utilize only those that are active and heavily supported.

**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-

Customers will be logging into and accessing secured information. Two factor identification and encrypted passwords would be expected.

Cryptography-

Sensitive information must be secured in accordance with state and federal laws. The encryption standards must meet or exceed the recommendation.

Coding Error-

Not revealing exploitable information in the event of a system error or malfunction. Proper documentation and remediation of logged errors.

Coding quality/Encapsulation-

All of the above mentioned security concerns will be addressed in the way and quality of the written code. Encapsulation and ensuring specific access is of great importance for anything dealing with sensitive information.

Secure API interactions-

This is partially covered in cryptography and input validation, but encryption and proper authentication and authorization is required. RESTful api utilized for communication.

**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. There is an unsecured database connection in the DocData Class, the read\_document method reveals the credentials root. Exposing sensitive credentials is an exploitable security risk.

2. In the customer class account\_balance is not private.

3. The Customer class deposit method has no input validation, it would accept unlimited or negative quantities.

4. The Greeting Controller uses a GET request rather than a POST request, this could expose sensitive information in logs.

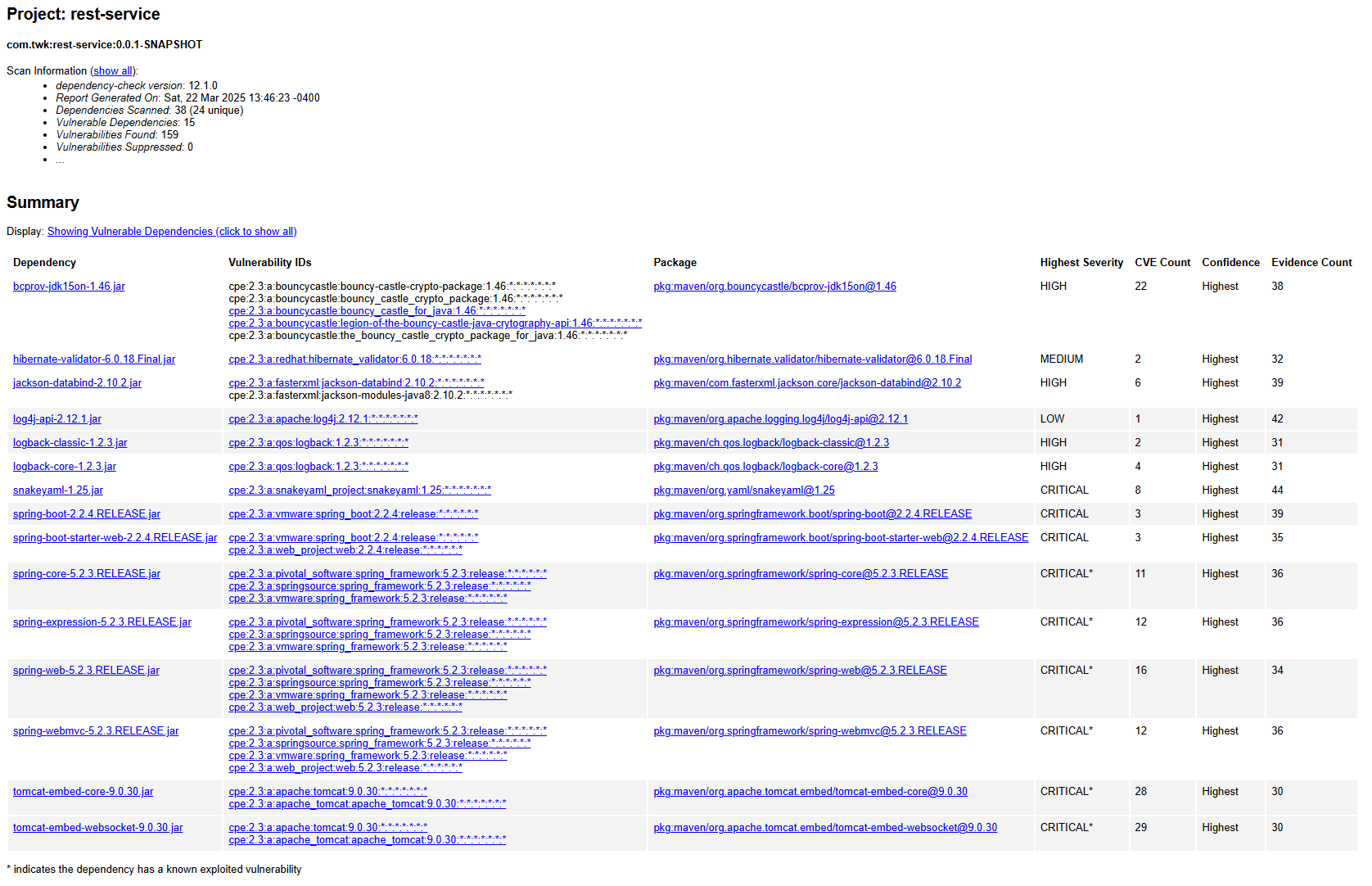
5. There is no security framework for the REST api, there is no user authentication.

6. There doesn't appear to be any form of encryption being utilized with any potentially sensitive information.

7. No utilization of HTTPS

**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:



Known current exploitations listed from top to bottom with recommended solutions

**CISA Known Exploited Vulnerability:**

* Product: VMware Spring Framework
* Name: Spring Framework JDK 9+ Remote Code Execution Vulnerability
* Date Added: 2022-04-04
* Description: Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding.
* Required Action: Apply updates per vendor instructions.
* Due Date: 2022-04-25
* Notes: <https://nvd.nist.gov/vuln/detail/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. The specific exploit requires the application to run on Tomcat as a WAR deployment. If the application is deployed as a Spring Boot executable jar, i.e. the default, it is not vulnerable to the exploit. However, the nature of the vulnerability is more general, and there may be other ways to exploit it.

**CISA Known Exploited Vulnerability:**

* Product: Apache Tomcat
* Name: Apache Tomcat Improper Privilege Management Vulnerability
* Date Added: 2022-03-03
* Description: Apache Tomcat treats Apache JServ Protocol (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.
* Required Action: Apply updates per vendor instructions.
* Due Date: 2022-03-17
* Notes: https://nvd.nist.gov/vuln/detail/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. In Apache Tomcat 9.0.0.M1 to 9.0.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99, Tomcat shipped with an AJP Connector enabled by default that listened on all configured IP addresses. It was expected (and recommended in the security guide) that this Connector would be disabled if not required. This vulnerability report identified a mechanism that allowed: - returning arbitrary files from anywhere in the web application - processing any file in the web application as a JSP Further, if the web application allowed file upload and stored those files within the web application (or the attacker was able to control the content of the web application by some other means) then this, along with the ability to process a file as a JSP, made remote code execution possible. It is important to note that mitigation is only required if an AJP port is accessible to untrusted users. Users wishing to take a defence-in-depth approach and block the vector that permits returning arbitrary files and execution as JSP may upgrade to Apache Tomcat 9.0.31, 8.5.51 or 7.0.100 or later. A number of changes were made to the default AJP Connector configuration in 9.0.31 to harden the default configuration. It is likely that users upgrading to 9.0.31, 8.5.51 or 7.0.100 or later will need to make small changes to their configurations.

NVD-CWE-Other

[**CVE-2022-1471**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-1471) suppress

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.

CWE-502 Deserialization of Untrusted Data, CWE-20 Improper Input Validation

CVSSv3:

* Base Score: CRITICAL (9.8)
* Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:3.9/RC:R/MAV:A

[**CVE-2023-20873**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-20873) suppress

In Spring Boot versions 3.0.0 - 3.0.5, 2.7.0 - 2.7.10, and older unsupported versions, an application that is deployed to Cloud Foundry could be susceptible to a security bypass. Users of affected versions should apply the following mitigation: 3.0.x users should upgrade to 3.0.6+. 2.7.x users should upgrade to 2.7.11+. Users of older, unsupported versions should upgrade to 3.0.6+ or 2.7.11+.

**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.

Static Testing-

Apply Required updates to patch vulnerabilities found during the dependency check. Utilize SnakeYaml’s SafeConstructor when parsing untrusted content to restrict deserialization. After updating and changing the recommended tools, run the dependency report again to search for any items that were overlooked or missed. Repeat process if any are found.

Manual testing-

Hide the disclosed root credentials in the Docdata class. In the customer class make the account balance into a private attribute with appropriate getters and setters. Apply input validation for the deposit class. Apply input validation system wide. Apply encryption standards to sensitive information. Utilize HTTPS for secure communication. Utilize a security framework. Change the greeting controller get request to a post request.