# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **24, January 2025** | **Jacquelin Kitcher** |  |

## 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

Jacquelin Kitcher

**1. Interpreting Client Needs**

* 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?

Secure communication is valuable to the company because as a financial company, they are responsible for their client’s personal information, including identifying information and private financial details, which could include bank account numbers, social security numbers, and so on. Although unspecified, since Global Rain works with companies from all over the world, it is possible that Artemis Financial is not only based in another country, but could also perform international transactions. Governmental restrictions do not appear to apply, but could in the way of banking laws, and international business transactions. Artemis wants to modernize the process, to include the newest software security tools.

**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.

* API’s are relevant to the application, since they use a RESTful API.
* Input validation is important, since they are trying to protect their application from external threats, which could gain access to unintentional data if proper input validation is not in place.
* Output encoding is also important to prevent external threats, to make sure that the information is only provided to the intended audience, by way of password management, MFA, encryption, etc.
* Secure error handling is essential to prevent external attacks on their program.

**3. Manual Review**

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

Findings:

1. For input validation, I noticed that the code does include the SQL library which I think could help prevent SQL injection.
2. The customer class has private fields for the account number, but not account balance.
3. I also don’t see input validation anywhere in the customer class, although I am not sure if that is handled elsewhere.
4. It also looks like in multiple classes, output encoding is not present.
5. Since there is a long id, and string input in the Greeting class, there needs to be input validation.
6. In the greeting class, there isn’t user authentication.
7. Input validation needed in CRUD class.

**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

**CVE-2020-10693: Allows attackers to bypass input sanitation.**

[**CVE-2020-25649**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-25649): Data integrity vulnerability due to XXE attacks.

[**CVE-2020-9488**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9488): SMTPS susceptible to man-in-middle attack.

[CVE-2023-6378](https://cve.org/CVERecord?id=CVE-2023-6378): Susceptible to denial-of-service attacks.

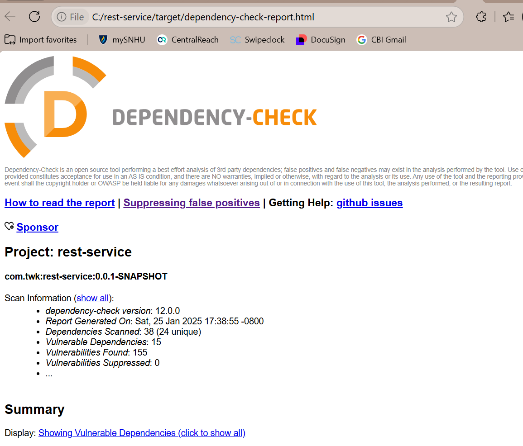
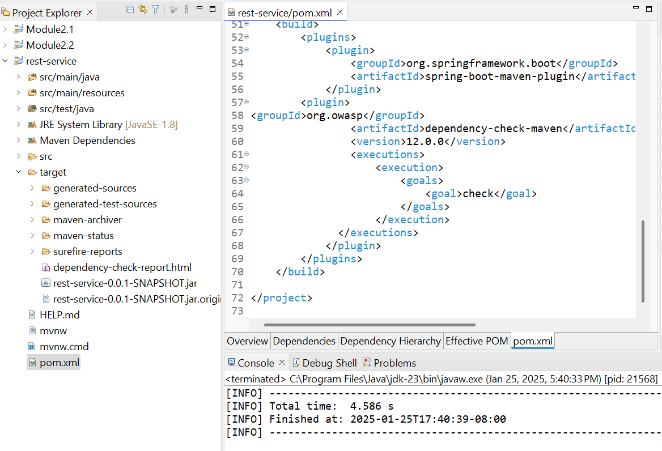
[**CVE-2022-1471**](https://nvd.nist.gov/vuln/detail/CVE-2022-1471)**: Susceptible to deserialization**

**CVE-2023-20883: Susceptible to denial-of-service attacks**

**CVE-2023-20863: Susceptible to cause denial of service condition**

**CVE-2023-46589: Input validation vulnerability (request smuggling)**

I’ve included the information provided when running the test, and the dependency check info in the screenshots below:



**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.

Input validation appears to be a particularly large issue for this program. I noticed it when doing the code review and it was the basis of several of the Maven Vulnerability Dependency Test vulnerabilities. I think the use of an input validation library could help strengthen the security of the program. I am not sure if it already has one, but if it does, it doesn’t seem to be strong enough or working properly. Maybe the libraries are out of date and need to be updated or just verified that they are the most up to date versions. Sanitizing input/output is something that could help as well. We need to verify and choose or adjust current serialization practices. We could add a WAF for controlling and detecting DOS patterns. Update libraries that handle HTTP requests, parsing to prevent request smuggling. Most of the CVE’s suggest updating libraries to the most current versions, and/or to include tools that address some of the main issues.

References

Dependency-Check Report <file:///C:/rest-service/target/dependency-check-report.html>

*NVD - vulnerabilities*. (n.d.). https://nvd.nist.gov/vuln

*OWASP Secure Coding Practices-Quick Reference Guide | OWASP Foundation*. (n.d.). https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/