# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **5/26/2024** | **Dan Kleiner** |  |

## 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

Daniel Kleiner

**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?

As a financial institution Artemis Financial must handle some of the customers’ most sensitive data. This includes their payment, contact, financial, and all other data that would be needed for identity theft. While international transactions may not be too much of a concern, we may have made sure we follow international regulations as well as the PCI-DSS for card payments and The Graham-Leach-Bliley Act for all financial transactions.

When modernizing these systems, we need to consider all technologies in an ever-changing cyber landscape. Using open-source libraries so that we can evolve just as fast the threats as well as keeping up to date with the latest firewalls and programing techniques to help thwart these threats before they become a breach.

**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.

The first area of concern I would like to address would be input validation, as this application accepts input. This could lead to different types of injection attacks if the data is not sanitized before it is input into the system.

Next, we do have the use of Application Program Interfaces (APIs), if the API is not kept up to date this could create an increase in our attack surface by providing another vector for the attacker to go through.

Since we are using banking and financial information, we need to make sure that all data that is going between the user and the system and vice versa needs securely encrypted so that if there is a breach and the attacker tries to intercept data with a man in the middle attack, they will not be able to read the data without having to decrypt it. Making sure that both the client and the server ends are both secured and free from coding errors that could be exploited.

Last, making sure we follow secure coding practices to ensure that errors are kept to a minimum and that we can encapsulate portions of the code so that portions of the code are not visible or able to be interacted with by the users.

**3. Manual Review**

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

Java version is 1.8 and it should be no less than 18.

The REST API could be a vulnerability.

The Spring API could be a vulnerability.

DocData.java has username and password of root and root in code.

JSON could be a vulnerability.

myDateTime.java has incomplete accessor method.

myDateTime.java has public variables that should be encapsulated as private.

**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](file:///C:\Users\dklei\OneDrive\Documents\SNHU\CS305\Module%203\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l1_991c96a4e31e6c19e2b9136c8955bd423f2dc4c7)

Bouncy Castle has potential for a Denial of Service (DoS) that causes an OutOfMemoryError. Can be resolved by updating the dependency.

[hibernate-validator-6.0.18.Final.jar](file:///C:\Users\dklei\OneDrive\Documents\SNHU\CS305\Module%203\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l3_7fd00bcd87e14b6ba66279282ef15efa30dd2492)

A flaw found in Hibernate Validator has a bug that allows attackers to bypass input sanitation. Can be resolved by updating dependency.

[jackson-databind-2.10.2.jar](file:///C:\Users\dklei\OneDrive\Documents\SNHU\CS305\Module%203\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l5_0528de95f198afafbcfb0c09d2e43b6e0ea663ec)

jackson-databind allows attackers to cause DoS via object that uses cyclic dependencies. The vendor perspective is that the vulnerability is not valid because the steps cannot be achieved by an external attacker.

[log4j-api-2.12.1.jar](file:///C:\Users\dklei\OneDrive\Documents\SNHU\CS305\Module%203\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l10_a55e6d987f50a515c9260b0451b4fa217dc539cb)

Apache Log4j2 can be vulnerable to remote code execution (RCE) when attack configuration uses JDBC Appender with a JNDI LDAP data source URI when attacker has control of the target LDAP server. This can be fixed by limiting JNDI data source names to the java protocol in Log4j2 as well as updating the version.

[logback-core-1.2.3.jar](file:///C:\Users\dklei\OneDrive\Documents\SNHU\CS305\Module%203\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l12_864344400c3d4d92dfeb0a305dc87d953677c03c)

A serialization vulnerability in logback receiver can allow attackers to perform DoS when sending poisoned data. Can be resolved by updating dependency.

[snakeyaml-1.25.jar](file:///C:\Users\dklei\OneDrive\Documents\SNHU\CS305\Module%203\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l14_8b6e01ef661d8378ae6dd7b511a7f2a33fae1421)

Snakeyaml’s Constructor() class does not restrict object types that can be instantiated during deserialization. This can lead to RCE by attackers. It is recommended to use SnakeYaml’s SafeConstructor when parsing untrusted content. It is also recommended to upgrade dependency.

[spring-boot-2.2.4.RELEASE.jar](file:///C:\Users\dklei\OneDrive\Documents\SNHU\CS305\Module%203\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l15_225a4fd31156c254e3bb92adb42ee8c6de812714)

Spring Boot version is unsupported, there is potential for DoS attack if Spring MVC is used together with reverse proxy cache. It is recommended to upgrade the dependency.

[spring-boot-starter-web-2.2.4.RELEASE.jar](file:///C:\Users\dklei\OneDrive\Documents\SNHU\CS305\Module%203\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l16_ec75d01d212b5229c16d872fb127744c0ed46ed8)

Spring Boot version is unsupported, there is potential for DoS attack if Spring MVC is used together with reverse proxy cache. It is recommended to upgrade the dependency.

[spring-core-5.2.3.RELEASE.jar](file:///C:\Users\dklei\OneDrive\Documents\SNHU\CS305\Module%203\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l17_3734223040040e8c3fecd5faa3ae8a1ed6da146b)

Spring framework version allows for specially crafted SpEL expressions to cause DoS. Can be resolved by upgrading the dependency.

[spring-web-5.2.3.RELEASE.jar](file:///C:\Users\dklei\OneDrive\Documents\SNHU\CS305\Module%203\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l18_dd386a02e40b915ab400a3bf9f586d2dc4c0852c)

Spring framework version allows for specially crafted SpEL expressions to cause DoS. Can be resolved by upgrading the dependency.

[spring-webmvc-5.2.3.RELEASE.jar](file:///C:\Users\dklei\OneDrive\Documents\SNHU\CS305\Module%203\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l19_745a62502023d2496b565b7fe102bb1ee229d6b7)

Spring framework version allows for specially crafted SpEL expressions to cause DoS. Can be resolved by upgrading the dependency.

[tomcat-embed-core-9.0.30.jar](file:///C:\Users\dklei\OneDrive\Documents\SNHU\CS305\Module%203\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l20_ad32909314fe2ba02cec036434c0addd19bcc580)

Improper Input Validation can lead to sensitive data being leaked in an Error Message. It is recommended to upgrade the dependency.

[tomcat-embed-websocket-9.0.30.jar](file:///C:\Users\dklei\OneDrive\Documents\SNHU\CS305\Module%203\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l22_33157f6bc5bfd03380ebb5ac476db0600a04168d)

Improper Input Validation can lead to sensitive data being leaked in an Error Message. It is recommended to upgrade the dependency.

**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.

Our mitigation plan will start with upgrading all the dependencies to eliminate those vulnerabilities that were highlighted in the dependency check. We can then update the java library to make sure that we are using the most up-to-date library. We can then go in a clean-up places where the code in areas such as incomplete methods, as well as taking out the data that could give someone unauthorized access to parts of the system.