# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **01-20-25** | **Carlos Rodriguez** | **Project One** |

## 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

Carlos Rodriguez

**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?

The client’s, Artemis Financials, needs are they want to modernize and use some of the latest and most effective software security to protect their customers from possible cyber-attacks and security vulnerabilities of their current API. The value of secure communications to the company is very important because of the sensitive information and data their customer’s hold. They take care of the financial plans of customers that range from savings, retirement, investments and insurance so they must heavily focus on security threats that may arise. Establishing an effective HTTPS is crucial to ensure that communications between servers and clients do not get attacked and breach their data. There could be possible international transactions that the company can produce so they must follow and adhere to their local regulations and standards. It should have an established transaction encryptions and secure API endpoints to make sure no unauthorized users obtain this information. There are governmental restrictions on secure communications to consider like the Gramm-Leach-Bliley Act (GLBA) and other data privacy laws that are required to comply to. The external threats that might be present now are Man-in-the-Middle (MIM) attacks, SQL injections, weak API endpoints and future threats that may occur are vulnerabilities found in third-party libraries or weak updated versions of the software. Open-source libraries are important for modernization requirements because it helps update and add certain features and functionalities to stay up to date with security threats. Evolving web application technologies help with cross-functionality over various devices, better maintainability and help with reduced costs and efficiency. These applications help to create more security measures and defenses to help performance run smoothly.

**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.

Based off the vulnerability assessment process flow diagram, Artemis Financial’s software application areas of security that apply are input validation, APIs, cryptography, client/ server, and code quality. Input validation would be needed because the user must put their account information into the application to be able to access it. It is relevant because there could be injection attacks to the account, and we must be aware of that security threat and be able to make sure only authorized users can access the corresponding account. APIs are relevant because they are already using RESTful web APIs to help prevent the unauthorized users and keep their data secure. We must check their API to make sure it is running efficiently and doing its necessary purpose. Cryptography is relevant because it is a safeguard to use to help transfer sensitive data and data at rest. The application will be having important financial information from the customer’s so it would need to be able to protect balance transfers and the balance itself. The client/ server security is relevant because the architecture needs to have secure communication, like HTTPS, to keep data transfers encrypted and make sure the data does not reach someone else. Lastly, the code quality must be consistent, efficient and up to industry standards to prevent/ minimize vulnerabilities and attacks to the application. The code should be well-defined and have in-text comments to understand what is being done. If any vulnerabilities arise, then checking the code quality would give a better idea of why it has been attacked.

**3. Manual Review**

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

[Include your 7–10 findings here.]

1. In DocData.java on DriverManager.getConnection(), there appears to be a database connection that is using a hardcoded login information (username/ password), that can lead to possible exposure and vulnerability.

2. In DocData.java, there is a vulnerability because there is no exception handling, which can cause the system to go down.

3. In DocData.java, there is no method of logging which can make it difficult to track the activity and adjustments made on the software.

4.. In CRUDController.java and GreetingController.java, the API endpoints of /read and /greeting appear to not have authentication to prevent attackers from entering.

5. In CRUDController.java on CRUD() method, there appears to be no input sanitization which can lead to a possible SQL injection onto the software

6. In customer.java, there was no encryption added to the customer’s data and can lead to exposure of their information.

7. In myDateTime.java on setMyDateTime() and retrieveDateTime(), it appears to have no real purpose to the code and can lead to code errors and confusion.

**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

A screenshot of a computer

Description automatically generated

There are 14 dependencies on the report:  
1. 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.

[bcprov-jdk15on-1.46.jar](file:///Users/crodr038/Downloads/rest-service/target/dependency-check-report.html#l2_991c96a4e31e6c19e2b9136c8955bd423f2dc4c7)

[**CVE-2013-1624**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-1624)

This implementation does not consider timing side-channel attacks on a noncompliant MAC check, causing attackers to be able to do plaintext-recovery attacks through statistical analysis of timing data. The recommended solution would be to update to the latest version of the cryptography. It was documented on 2013 and the source was MITRE.

2. Spring Boot

[spring-boot-2.2.4.RELEASE.jar](file:///Users/crodr038/Downloads/rest-service/target/dependency-check-report.html#l3_225a4fd31156c254e3bb92adb42ee8c6de812714)

[**CVE-2022-27772**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-27772)

This version was vulnerable to directory hijacking and the recommended solution is to update or upgrade the product to the latest version. The ones who remain affected are the maintainers that are using a product or version that is not supported. This issue was known/ shared through GitHub.   
  
3. logback-classic module

[**logback-classic-1.2.3.jar**](file:///Users/crodr038/Downloads/rest-service/target/dependency-check-report.html#l4_7c4f3c474fb2c041d8028740440937705ebb473a)

[**CVE-2021-42550**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-42550)

This version and prior ones caused the attacker to edit the configuration files that could craft malicious configurations to execute a code from LDAP servers. The recommended solution would be to get the latest version of logback. This issue was documented by the Switzerland Government Common Vulnerability Program

4. logback-core module

[logback-core-1.2.3.jar](file:///Users/crodr038/Downloads/rest-service/target/dependency-check-report.html#l5_864344400c3d4d92dfeb0a305dc87d953677c03c)

[**CVE-2021-42550**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-42550)

This section has the same vulnerabilities as #3.

5. The Apache Log4j API

[log4j-api-2.12.1.jar](file:///Users/crodr038/Downloads/rest-service/target/dependency-check-report.html#l6_a55e6d987f50a515c9260b0451b4fa217dc539cb)

[**CVE-2020-9488**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9488)

This vulnerability made an improper validation of certificate with the host mismatch in Apache appender that can cause the connection to be attacked by a man-in-the-middle and can leak the messages in the appender. The recommended solution is to download the latest version of Apache Log4j. This issue was documented by Apache Software Foundation.

6. YAML 1.1 parser and emitter for Java

[snakeyaml-1.25.jar](file:///Users/crodr038/Downloads/rest-service/target/dependency-check-report.html#l9_8b6e01ef661d8378ae6dd7b511a7f2a33fae1421)

[**CVE-2017-18640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-18640)

This vulnerability is that the Alias feature in this program makes the entity expansion during a load operation. To resolve this issue, it is recommended to get the version after 1.26 to get rid of this issue. This issue was documented by another similar detected vulnerability on a different case, CVE-2003-1564. The source of this issue was made by MITRE.

7. General data-binding functionality for Jackson: works on core streaming API

[jackson-databind-2.10.2.jar](file:///Users/crodr038/Downloads/rest-service/target/dependency-check-report.html#l10_0528de95f198afafbcfb0c09d2e43b6e0ea663ec)

[**CVE-2020-25649**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-25649)

The issue that was identified was that the entity expansion was not properly secured which led to XML external entity (XXE) attacks. This attack would affect the data integrity. The expected solution is to make sure that the program is up to date with the latest version. This vulnerability was identified by Red Hat, Inc and was found in the FasterXML Jackson Databind.

8. Core Tomcat implementation

[tomcat-embed-core-9.0.30.jar](file:///Users/crodr038/Downloads/rest-service/target/dependency-check-report.html#l14_ad32909314fe2ba02cec036434c0addd19bcc580)

[**CVE-2019-17569**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2019-17569)

This vulnerability introduced a regression, and it was invalid Transfer-encoding headers that had been incorrectly processed and can lead to a possible HTTP Request smuggling if the Tomcat was located in a reverse proxy. The solution to this was that the regression was considered to be unlikely and the issue was documented by Apache Software Foundation.

9. Hibernate's Bean Validation (JSR-380) reference implementation.

[hibernate-validator-6.0.18.Final.jar](file:///Users/crodr038/Downloads/rest-service/target/dependency-check-report.html#l17_7fd00bcd87e14b6ba66279282ef15efa30dd2492)

[**CVE-2020-10693**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-10693)

The vulnerability included a bug in the message interpolator processor and it made invalid EL expressions appear as valid. This error causes attackers to bypass the input sanitation controls that developers place. The recommended solution is to update to the latest version, and this was documented by the Red Hat, Inc.

10. Spring Web

[spring-web-5.2.3.RELEASE.jar](file:///Users/crodr038/Downloads/rest-service/target/dependency-check-report.html#l20_dd386a02e40b915ab400a3bf9f586d2dc4c0852c)

**CVE-2016-1000027**

This framework has issues on the potential remote code execution that can be used to Java deserialization of untrusted data. The solution for this is that the untrusted data is not an intended use case and it would not be affected because certain users do the deserialization of trusted data. The instance was documented by Pivotal Spring Framework.

11. Spring Beans

[spring-beans-5.2.3.RELEASE.jar](file:///Users/crodr038/Downloads/rest-service/target/dependency-check-report.html#l21_0250c8c641433dc06b1b44e4563fa08a2fbf8954)

**CVE-2022-22965**

This software might be vulnerable to remote code execution by the data binding. The recommended solution is to run it as a Spring Boot executable jar and it would not have this exploit. The issue had occurred on a Spring MVC or Spring WebFlux.

12. Spring Web MVC

[spring-webmvc-5.2.3.RELEASE.jar](file:///Users/crodr038/Downloads/rest-service/target/dependency-check-report.html#l22_745a62502023d2496b565b7fe102bb1ee229d6b7)

**CVE-2021-22060**

In older versions, there is a possibility of the users to have malicious input to make insertions of more log entries. The recommended solution is to update to the latest versions and stop using the unsupported ones. The vulnerability was identified by the Spring Framework.

13. Spring Context

[spring-context-5.2.3.RELEASE.jar](file:///Users/crodr038/Downloads/rest-service/target/dependency-check-report.html#l23_7750c95c96c7a1885c8b1b503ba915bc33ca579a)

**CVE-2022-22968**

The vulnerability is that the older versions had patterns for disallowedFields on a DataBinder that are case sensitive which led to fields not being properly protected. The recommended solution is to update to the latest versions and to be careful with case sensitive patterns. This vulnerability was documented by the Spring Framework.

14. Spring Expression Language (SpEL)

[spring-expression-5.2.3.RELEASE.jar](file:///Users/crodr038/Downloads/rest-service/target/dependency-check-report.html#l24_d0c6bb10758805b2153c589686b8045554bfac2d)

**CVE-2022-22950**

In older version, there is the possibility that the user has provided a unique SpEL expression that can cause denial of service attack. The course of action would be to update to the latest version of the software and to stop using the unsupported version and/ or the outdated versions. This issue was documented by the Spring framework.

**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.

For the manual review, we have identified some of the issues based off the code review and we must look at the steps to take to correct these issues. For the first one, we must either avoid making hardcoded credentials or securing them in environmental variables. For the second one, we must properly do the error handling with a secure error message and letting the user know about the error. For the third one, we have to implement a logging system like logback into the application and make sure that the logs are secured. For the fourth one, we have to make sure that the web server is enforcing the HTTPS into the endpoints and that there is an SSL/TLS certificate on the communication. For the fifth one, the best approach to take would be to implement the input validation on the business name and to sanitize the inputs before having them pass through. For the sixth one, the best action to take would be to add an encryption tool to help protect the customer data and other sensitive information. For the last review, the recommended plan would be to remove these unused methods and implement them when they have a functional purpose.

From the static testing report, the mitigation plan to take would be making sure the software is up to date with their versions and are compatible with the product that is being used. The updated versions have been notified of the vulnerabilities and they now took the necessary steps to eliminate the security concern.