# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **11/17/2024** | **[Your name]** |  |

## 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

Joshua Shults

**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?

Secure communications are vital to protecting any sensitive data, which could include financial data and records, data on businesses that utilize Global Rain, and sensitive government data. By having all data be secure and encrypted while either at rest or in transit to a client, the risk of data being retrieved can be mitigated, as well as encryption will reduce any leaks if data is retrieved as it will not be written in plain language that can be easily interpreted.

* Are there any international transactions that the company produces?

As stated in Global Rain’s scenario, they work with agencies and businesses around the world, this implies that international transactions will be produced. International transactions have to follow certain rules, such as GDPR that is mandated in the EU.

* Are there governmental restrictions on secure communications to consider?

Yes, as stated above international transactions are subject to rules on secure communications. These rules can be different depending on the region the transaction is a part of, but they contain rules pertaining to consent, notifications upon any breaches, as well as being audited by the enforcers of the restriction. Any non-compliance with these rules and regulations could result in substantial fines, as well as loss of business in the region that the breach occurred in.

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

Threats that are present and at a potential to occur in the future include phishing, SQL injection, malicious code injection, abuse of permissions with APIs, and threats are ever evolving.

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

There are several modern techniques that must be considered, as noted above first would be utilizing open source libraries that are updated frequently about new threats, breaches, and vulnerabilities along with their causes. Another requirement would be the frequent updating and patching of the server that Global Rain is operating on, by staying up to date with third party tools as well as any proprietary tools this will make it harder for malicious sources to keep up with Global Rain and take their time to find vulnerabilities. Finally, APIs should be heavily vetted and understood to identify where their end points into Global Rain’s system could lead to injection or other malicious attacks.

**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.

1. Identity and access management. A role based access system will ensure that unauthorized users cannot access or perform administrative commands.
2. Data Security. Keeping data secure and encrypted at rest and in transit will help prevent, or mitigate, the risk of leaking sensitive data.
3. Dependency Management. Any open-source utilities that will be used for Global Rain should be kept up to date and scanned for vulnerabilities.
4. APIs. Any API used should be restricted at their end point into the system to prevent any abuse such as injections.
5. Architecture Review. The entire code base should be reviewed by testers to find any bugs that could be taken advantage of in the case of a malicious user.

**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. Under “DriverManager.getConnection” there is a line where credentials can be exposed. During error handling this should be handled to where a generic error message displays instead of potentially exposing a root directory.
2. In “customer.java” there is a field called account\_balance that is not a protected variable.
3. Also in “customer.java” the deposit method does not have a validator.
4. In “GreetingController.java” there is another unvalidated parameter “name.”
5. In “DocData.java” the root directory is also used again, potentially exposing credentials.
6. Under “GreetingController.java” there is an endpoint where a user inputs the variable “name.” This is not rate limited, which could be used for Denial of Service.
7. Under “customer.java” account\_number is also unprotected.
8. Account\_number and account\_balance are also able to be accessed by a public method, this could be abused to directly change the values stored within the two variables.

**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
  + - 1. bcprov-jdk15on-1.46.jar
      2. [spring-boot-2.2.4.RELEASE.jar](file:///C:\Users\Jtshu\OneDrive\Desktop\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l3_225a4fd31156c254e3bb92adb42ee8c6de812714)
      3. [logback-core-1.2.3.jar](file:///C:\Users\Jtshu\OneDrive\Desktop\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l4_864344400c3d4d92dfeb0a305dc87d953677c03c)
      4. [log4j-api-2.12.1.jar](file:///C:\Users\Jtshu\OneDrive\Desktop\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l5_a55e6d987f50a515c9260b0451b4fa217dc539cb)
      5. [snakeyaml-1.25.jar](file:///C:\Users\Jtshu\OneDrive\Desktop\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l8_8b6e01ef661d8378ae6dd7b511a7f2a33fae1421)
      6. [jackson-databind-2.10.2.jar](file:///C:\Users\Jtshu\OneDrive\Desktop\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l9_0528de95f198afafbcfb0c09d2e43b6e0ea663ec)
      7. [tomcat-embed-core-9.0.30.jar](file:///C:\Users\Jtshu\OneDrive\Desktop\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l13_ad32909314fe2ba02cec036434c0addd19bcc580)
      8. [hibernate-validator-6.0.18.Final.jar](file:///C:\Users\Jtshu\OneDrive\Desktop\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l16_7fd00bcd87e14b6ba66279282ef15efa30dd2492)
      9. [spring-web-5.2.3.RELEASE.jar](file:///C:\Users\Jtshu\OneDrive\Desktop\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l19_dd386a02e40b915ab400a3bf9f586d2dc4c0852c)
      10. [spring-beans-5.2.3.RELEASE.jar](file:///C:\Users\Jtshu\OneDrive\Desktop\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l20_0250c8c641433dc06b1b44e4563fa08a2fbf8954)
      11. [spring-webmvc-5.2.3.RELEASE.jar](file:///C:\Users\Jtshu\OneDrive\Desktop\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l21_745a62502023d2496b565b7fe102bb1ee229d6b7)
      12. [spring-context-5.2.3.RELEASE.jar](file:///C:\Users\Jtshu\OneDrive\Desktop\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l22_7750c95c96c7a1885c8b1b503ba915bc33ca579a)
      13. [spring-expression-5.2.3.RELEASE.jar](file:///C:\Users\Jtshu\OneDrive\Desktop\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l23_d0c6bb10758805b2153c589686b8045554bfac2d)
* A brief description and recommended solutions provided by the dependency-check report

The majority of the solutions recommended by the dependency-check report is to update the referenced tools to newer versions that do not contain the aforementioned vulnerabilities.

* Any attribution that documents how this vulnerability has been identified or documented previously

<https://spring.io/blog/2024/11/15/spring-framework-cve-2024-38828-published>

<https://spring.io/security/cve-2021-22060>

<https://spring.io/blog/2022/04/13/spring-framework-data-binding-rules-vulnerability-cve-2022-22968>

<https://spring.io/security/cve-2024-38808>

**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 and foremost critical vulnerabilities should be fixed by making the upgrades recommended within the dependency check to new versions of the tools utilized. Any tools that still cause issues could be changed out for other libraries that are open source with an active support site. Continue the code review and make necessary changes, such as in making variables private and disallowing public methods to access them.