# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **5/23/2025** | **Roger Fisher** |  |

## 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

Roger Fisher

**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?

Because Artemis Financial works with sensitive data about savings, retirement and insurance, they must rely on secure forms of communication. The loss of that data can lead to problems for their customers and a decrease in the company’s reputation. Doing everything you can to keep that information protected should be the main concern.

Artemis Financial does not say i it conducts international transactions, but since they’re moving towards modern systems and web tools, that may come. In such cases, they must ensure both communication systems and data storage meet international safety standards as well. For instance, U.S. laws for financial institutions demand using strong security to protect customers’ information. Artemis Financial needs to ensure users are always using safe encryption, solid passwords and guarded connections.

Among the critical risks Artemis Financial must handle now, we also have hackers stealing private information, bad actors adding harmful software and other people trying to access accounts through loose login controls. They have to ensure their site is protected from frequent problems such as SQL injection and cross-site scripting.

Because they are upgrading their technologies, they’ll likely rely on open-source libraries. They are useful, but there are dangers if they haven’t been updated or are no longer maintained. For this reason, you should update those often and check them for any reported weaknesses. Reacting to technological changes and keeping security as they update the company’s software is necessary.

**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.

Artemis Financial's software goes through several parts of the vulnerability assessment process. Because it relies on REST and is most likely accessed from a web browser, the key concerns are input checks, the API, cryptography, server and client processed, code quality and encapsulation.

You should use input validation because the user’s input comes to the system through forms or requests. To prevent SQL injection or XSS issues, all the data should be examined and cleaned.

RESTful services are a major component on their platform thanks to APIs. It is important to secure API interactions so no one outside can use or gain access to confidential information.

Having strong cryptography protects everything from financial data to login information. When sending or saving data, strong encryption is required between various parties.

This also means we need to pay attention to client/server security. Because people may access Artemis’s software from browsers or on mobile applications, all data moving between the client and server needs to be protected from being seen or changed by others.

Having adequate code quality reduces the chances of encountering either bugs or insecure behaviors (such as data leaks). Good, safe programming habits make the app perform well and are easy to take care of.

With encapsulation, the system’s sensitive areas are hidden from exposure. Artemis should use secure interfaces to hide their user data and business logic.

Each of these areas needs to be considered during the vulnerability assessment for Artemis Financial's system to remain safe, secure and reliable.

**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. Because the account\_balance field in customer.java is public, all the details of customer financial data are now directly exposed to other programs. People outside the bank can reach and adjust customers’ account balances. Meanwhile, it should be made private, and its value must be accessed or changed through proper methods.
2. DocData.java is set up to connect to the database using hardcoded credentials root, root. Keeping usernames and passwords in unencrypted format is unsafe and might result in serious security problems. Move the credentials to either a secure configuration file or environment variables and try to use encryption.
3. The SQL connection in DocData.java is not prepared and it fails to clean data that comes from users. Therefore, if values are added later from outside sources, the site might be attacked by SQL injection. Attack prevention requires checking inputs and creating queries that use parameter formats.
4. When using DocData.jav, any SQLException found is only printed without any helpful action or log entry. Without feedback, there is no protection, should anything fail. To do better, you should record the errors and provide useful information to the user or manage the exception in the right way.
5. The method /read in CRUDController.java accepts a business\_name parameter but does not check or clean it. The string could be exploited by being used in SQL if it is entered later by the application. Always verify and clean your inputs before starting application work.
6. In GreetingsController.java, the application constructs a greeting using the data sent by the user without escaping it. Rendering the input in a web page can result in an XSS attack. It's important to put special characters in user input into harmless form before showing the output.
7. For some classes, like Greeting.java and CRUD.java, final fields are used and no checks for correct values are done. Even though immutability is valuable, values that put users at risk could still be kept and returned. Make sure to add input validation before you set values during construction.
8. Both retrieveDateTime and setMyDateTime are empty methods in the myDateTime.java class. This might cause a runtime bug or mix up for others who use the software. If a method isn’t in use, it should either be completed or taken out.
9. The application.properties listens at port 8081 by default, but nothing shows it using HTTPS. If a website isn’t using HTTPS, all your data could easily be snooped on. It's necessary to use a valid SSL certificate to enable HTTPS in order to make communication safe.

**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

Dependency: Example (level of security issue)

* Vulnerability IDs- Example.release.1

Description- Here is where an example of the description would be as well as a solution for the issue

Attribution- Here is where an example of an attribution would be

Dependency: spring-boot-starter-web-2.2.4.RELEASE.jar (Critical)

* cpe:2.3:a:vmware:spring\_boot:2.2.4:release::\*:\*:\*:\*:\*:\*
* cpe:2.3:a:web\_project:web:2.2.4:release:\*:\*:\*:\*:\*:\*

Spring Boot version 2.2.4 is affected by multiple vulnerabilities including potential denial-of-service risks due to poor validation. A potential solution would be to update the spring boot to the most current version.

Identified by OWASP Dependency check using NVD CVE data (CVE-2023-20873)

Dependency: spring-boot-2.2.4.RELEASE.jar (Critical)

* cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\*

Spring Boot version 2.2.4 is affected by multiple vulnerabilities including potential denial-of-service risks due to poor validation. A potential solution would be to update the spring boot to the most current version.

Identified by OWASP Dependency check using NVD CVE data (CVE-2023-20873)

Dependency: snakeyaml-1.25.jar (Critical)

* cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\*

This version of SnakeYAML has several security issues that allow attackers to run harmful code remotely. To fix this, upgrade to a newer version of SnakeYAML 2.0+

Identified by OWASP Dependency check using NVD CVE data and OSS Index (CVE-2022-1471)

Dependency: spring-core-5.2.3.RELEASE.jar (Critical)

* cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
* cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
* cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*

This version of Spring Framework may let attackers run harmful code remotely by injecting malicious data. Solution to this is by updating to a newer version of Spring Framework.

Identified as a CISA Known Exploited Vulnerability under CVE-2022-22965

Dependency: spring-webmvc-5.2.3.RELEASE.jar (Critical)

* cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
* cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
* cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
* cpe:2.3:a:web\_project:web:5.2.3:release:\*:\*:\*:\*:\*:\*

This version of Spring Framework may let attackers run harmful code remotely by injecting malicious data. The solution to this is, is to try updating to a newer version of Spring Framework.

Identified as a CISA Known Exploited Vulnerability under CVE-2022-22965

Dependency: spring-expression-5.2.3.RELEASE.jar (Critical)

* cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
* cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
* cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*

This version of Spring Framework may let attackers run harmful code remotely by injecting malicious data. The solution to this is, is to try updating to a newer version of Spring Framework.

Identified as a CISA Known Exploited Vulnerability under CVE-2022-22965

Dependency: spring-context-5.2.3.RELEASE.jar (Critical)

* cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
* cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
* cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*

This version of Spring Framework may let attackers run harmful code remotely by injecting malicious data. The solution to this is, is to try updating to a newer version of Spring Framework.

Identified as a CISA Known Exploited Vulnerability under CVE-2022-22965

Dependency: spring-web-5.2.3.RELEASE.jar (Critical)

* cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
* cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
* cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
* cpe:2.3:a:web\_project:web:5.2.3:release:\*:\*:\*:\*:\*:\*

This version of Spring Web has an issue where untrusted data may allow attackers to run malicious code remotely. A solution to this is to update your Spring Web to a newer version.

Identified by OWASP Dependency check using NVD CVE data and OSS Index (CVE-2016-1000027)

Dependency: tomcat-embed-core-9.0.30.jar (Critical)

* cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*
* cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*

This version of Tomcat allows attackers to exploit the AJP connector. If exposed to the internet, it can be used to upload or run malicious files remotely. The solution to this is to update Tomcat to a later version or disable it if unneeded.

Identified by OWASP Dependency check using NVD CVE data (CVE-2020-1938)

Dependency: tomcat-embed-websocket-9.0.30.jar (Critical)

* cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*
* cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*

This version of Tomcat allows attackers to exploit the AJP connector. If exposed to the internet, it can be used to upload or run malicious files remotely. The solution to this is to update Tomcat to a later version or disable it if unneeded.

Identified by OWASP Dependency check using NVD CVE data (CVE-2020-1938)

Dependency: bcprov-jdk15on-1.46.jar (High)

* cpe:2.3:a:bouncycastle:bouncy-castle-crypto-package:1.46:\*:\*:\*:\*:\*:\*:\*
* cpe:2.3:a:bouncycastle:bouncy\_castle\_crypto\_package:1.46:\*:\*:\*:\*:\*:\*:\*
* cpe:2.3:a:bouncycastle:bouncy\_castle\_for\_java:1.46:\*:\*:\*:\*:\*:\*:\*
* cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46:\*:\*:\*:\*:\*:\*:\*
* cpe:2.3:a:bouncycastle:the\_bouncy\_castle\_crypto\_package\_for\_java:1.46:\*:\*:\*:\*:\*:\*:\*

This version has a certificate validation vulnerability. It does not properly verify if the certificate was tied to the host its from and can allow attackers to trick the system with fake certificates. To fix this issue, update to a newer version of Bouncy Castle.

Identified by OWASP Dependency check using NVD CVE data and OSS Index (CVE-2024-34447)

Dependency: logback-classic-1.2.3.jar (High)

* cpe:2.3:a:qos:logback:1.2.3:\*:\*:\*:\*:\*:\*:\*

This version has a vulnerability where an attacker can crash the software by sending specialized data to cause a denial-of-service. The solution to this is to upgrade the Logback version

Identified by OSS Index (CVE-2023-6378)

Dependency: logback-core-1.2.3.jar (High)

* cpe:2.3:a:qos:logback:1.2.3:\*:\*:\*:\*:\*:\*:\*

This version has a vulnerability where an attacker can crash the software by sending specialized data to cause a denial-of-service. The solution to this is to upgrade the Logback version

Identified by OSS Index (CVE-2023-6378)

Dependency: jackson-databind-2.10.2.jar (High)

* cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\*
* cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:\*:\*:\*:\*:\*:\*:\*

This version has a security flaw that allows attackers to access sensitive data by using specially made XML input. The solution to this is to upgrade the Databind software to a newer version. Identified by OSS Index (CVE-2020-25649)

Dependency: hibernate-validator-6.0.18.Final.jar (Medium)

* cpe:2.3:a:redhat:hibernate\_validator:6.0.18:\*:\*:\*:\*:\*:\*:\*

This version of Hibernate Validator has a bug in a method that allows it to be tricked by invalid HTML. If a user leaves off the closing character of an HTML tag, it can allow attacks to occur. The solution is to upgrade the Hibernate Validator to a newer version with this patch.

Identified by OSS Index (CVE-2023-1932)

Dependency: log4j-api-2.12.1.jar (Low)

* cpe:2.3:a:apache:log4j:2.12.1:\*:\*:\*:\*:\*:\*:\*

This version improperly checks the identity of certificates for SMTP connections, allowing attackers to intercept log messages. The solution is to upgrade the Log4j to a newer version

Identified by OWASP Dependency check using NVD CVE data (CVE-2020-9488)

**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.

To begin with, it's important to start by upgrading or updating Artemis Financial’s outdated and insecure dependencies. This includes updating Log4j, Hibernate Validator, Jackson Databind, Logback (both core and classic), Bouncy Castle, Tomcat, SnakeYAML, Spring Boot, and several parts of the Spring Framework (like Spring Web and Spring Core). A number of these libraries had vulnerabilities, such as input validation, remote code execution and certificate verification. However, their newer versions avoid these vulnerabilities. One of the easiest and best ways to improve security, is to maintain these libraries regularly.

As well as upgrading the dependencies, the review of the code showed that several parts of it could be improved. An instance of this is setting the account\_balance field in the customer.java to public means that anyone outside your code can access customer funds. Managing access to that field by making it private and controlling the changes using the proper getter/setter methods. The database credentials are also written clearly in DocData.jav, which leaves the system very vulnerable to hackers. Encryption should be implemented whenever possible, and you should save these things in a secure configuration file or as an environment variable.

There are also a few problems tied to how the input is managed. Because the inputs for SQL are not paramaterized in the application, it is open to SQL injection attacks. The queries above should instead be organized with prepared statements. Those two controllers, CRUDController and GreetingsController, do not remove dangerous characters from the user’s input data. You must make sure they are validated and escaped to stop both XSS attacks and unintended use of SQL commands. In addition, some fields that are marked final don't receive validation, so any input that looks correct, but is dangerous for use, can still go through unchecked.

This last bit involves fixing up some bits of code. For example, retrieveDateTime and setMyDateTime in myDateTime.java do nothing and should either be developed or taken out respectively to avoid misunderstandings. As for the application, it’s set up to run on port 8081 even though it doesn’t force the use of HTTPS. Thats really risky because any attacker can grab unencrypted data. If you set up SSL and redirect all traffic to HTTPS, both your user’s information and the system’s will be much safer.