# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **06/01/2025** | **Eric Schooler-Davison** |  |

## 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

Eric Schooler-Davison

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?

Artemis financial is a company that advises clients on individualized financial plans. These include savings, retirement, investments, and insurance. The company values secure communications, opting to use only the latest and most effective software. The company will be advising entrepreneurs, businesses, and government agencies around the world. The software that Artemis Financial needs must be secure and encrypted as they will be completing long distance communications of confidential information such as social security numbers, account numbers, and personal info of clients and governments here and overseas. In order for governments or clients to trust Artemis Financial, they need to know that the company with protect their data absolutely and with utmost care. Transferring existing data from their previous software will need to be handled with care as an unsecured transfer without encryption may make data vulnerable. The main concern is external cyber attacks which are expected due to the wealth of information a hacker could obtain. Modernization will mean keeping everything up to date with the latest open-source libraries as cybersecurity is always evolving. Maintenance checks will be necessary to fix bugs/vulnerabilities causing things like denial of service attacks. Best security practices will need to be maintained to avoid things such as SQL injection and URL interpretation.

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.

* Input Validation: This is important for setting and protecting user accounts for the clients that use their application. This also helps prevent DoS attacks that can arise from overwhelming the system with continuous inputs or brute force attacks where the hacker tries various passwords till they gain entry.
* Cryptography: Data will be moving from A to B via communications with the client and the financing company. Ensuring that all data is encrypted whilst moving between the two will prevent data breaches during communication by ensuring only the intended recipient can interpret the information.
* Code Quality: Best Practices in secure coding will prevent the most dangerous vulnerabilities such as SQL injection and URL interpretation. Errors can create breaches that would be hackers can exploit to gain access to privileged information.
* Encapsulation: Using proper public and private variables instead of carrying variables from one function to another prevents data being accessible or manipulated. Declaring secure private variables class to class can go a long way in preventing data access.

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. Maven and Spring are using outdated versions.
2. The class customer is public as well as functions showInfo() and deposit()
3. Variables in class customer are public except for account\_number, but account\_balance was left public.
4. Function deposit() in customer has a freshly declared variable stand in for a deposit adjustment. This setup allows SQL injection to be completed, putting another plus at the end to add any amount they want to the account balance.
5. Many classes are public when they do not need to be, coding best practices should result in things being public only if needed.
6. Public class DocData has a public string DocData() function that is currently empty.
7. Setters and getters are spread out across multiple classes such as DocData and CRUD.
8. Greeting and DocData both contain a public function getId().

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** | **Description** | **Solution** |
| bcprov-jdk15on-1.46.jar | 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. | Update Bouncy Castle Cryptography from version 1.46 to version 1.80 |
| hibernate-validator-6.0.18.Final.jar | A bug in the message interpolation processor enables invalid EL expressions to be evaluated as if they were valid. | Implement proper input validation as ‘isValid’ method can be bypassed. |
| jackson-databind-2.10.2.jar | General data-binding functionality for Jackson: works on core streaming API | Jackson-databind is outdated leading to denial of service vulnerabilities and XXE attacks. Update to version above 2.13 (likely 2.15) |
| log4j-api-2.12.1.jar | Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender. Fixed in Apache Log4j 2.12.3 and 2.13.1 | Upgrade to 2.13.2 which supports this feature. Previous versions can set the system property mail.smtp.ssl.checkserveridentity to true to globally enable hostname verification for SMTPS connections. |
| logback-classic-1.2.3.jar | A serialization vulnerability in logback receiver component part of logback version 1.4.11 | Data needs verification that it is trusted before the application deserializes untrusted data. |
| logback-core-1.2.3.jar | In logback version 1.2.7 and prior versions, an attacker with the required privileges to edit configurations files could craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers. | Update logback to a newer version. |
| snakeyaml-1.25.jar | SnakeYaml's Constructor() class does not restrict types which can be instantiated during deserialization. Deserializing yaml content provided by an attacker can lead to remote code execution. | Recommend using SnakeYaml's SafeConsturctor when parsing untrusted content to restrict deserialization. Recommend upgrading to version 2.0 and beyond. |
| spring-boot-2.2.4.RELEASE.jar | In Spring Boot versions 3.0.0 - 3.0.5, 2.7.0 - 2.7.10, and older unsupported versions, an application that is deployed to Cloud Foundry could be susceptible to a security bypass. | Upgrade to either version 3.0.6 or 2.7.11 |
| spring-boot-starter-web-2.2.4.RELEASE.jar | In Spring Boot versions 3.0.0 - 3.0.5, 2.7.0 - 2.7.10, and older unsupported versions, an application that is deployed to Cloud Foundry could be susceptible to a security bypass. | Upgrade to either version 3.0.6 or 2.7.11 |
| spring-context-5.2.3.RELEASE.jar | A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. The specific exploit requires the application to run on Tomcat as a WAR deployment. | Update to 5.2.15 will resolve some vulnerabilities but most require 5.4 or above. Recommend updating to the newest version. |
| spring-core-5.2.3.RELEASE.jar | A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. The specific exploit requires the application to run on Tomcat as a WAR deployment. | Update to 5.2.15 will resolve some vulnerabilities but most require 5.4 or above. Recommend updating to the newest version. |
| Tomcat-embed-core-9.0.30.jar | Tomcat out of date | Update tomcat-embed/core |

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.

All but two vulnerabilities were resolved by updating the .jar. After updating to newer versions another test should be run to sniff out any other issues that may arise with newer versions. Solutions to each issue are included in the above chart.