# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **1/25/2025** | **Nichole St.Clair** |  |

## 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

Rose Nichole St.Clair

**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 provides tailored financial plans for individuals, covering savings, retirement, investments, and insurance. Due to the highly sensitive nature of the data they manage—such as Social Security Numbers, tax records, and other confidential client information—secure communication is of utmost importance. Although there is no direct indication that Artemis Financial operates exclusively in the U.S., it is likely the company engages in international transactions.

A significant regulatory consideration involves safeguarding trade secrets and ensuring they are not exposed. To protect client data, implementing strong encryption protocols is essential to prevent unauthorized access. Additionally, regular system updates, including bug fixes and addressing security vulnerabilities, are critical for maintaining a modern and secure operational framework.

**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**: Artemis Financial implements input validation to confirm the authenticity of information ownership, ensuring user protection. This involves verifying inputs, often processed as strings, to maintain data integrity.
* **Code Quality**: High code quality allows for precise access control over methods based on user roles. For example, users can only view and interact with their own information, preventing access to other users’ data or sensitive server resources.
* **APIs**: The development of APIs is essential for managing internal and external operations. APIs define and regulate permissible data access, ensuring secure and efficient communication between systems.
* **Error Handling**: Effective error handling identifies and addresses issues within the API, ensuring problematic areas are resolved. This minimizes the risk of user data exposure and strengthens the overall security framework.
* **Cryptography**: The implementation of cryptography is vital to protect user information, particularly when dealing with international transactions and multiple currencies. This ensures sensitive data remains secure against unauthorized access globally.

**3. Manual Review**

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

As part of the vulnerability assessment, I began by reviewing input validation. I checked the POM.XML file for the presence of an Apache Validator but found no indication of its use. Next, I examined the GreetingController and observed that input validation was not implemented. Without any output to verify, this remains inconclusive.

I then evaluated the API functionality and found no operational API. However, the program could still access data, albeit in an unsecured manner. It retrieves data through the URL instead of a POST method, which introduces security risks as sensitive information could be exposed in browser history. Although the program accepts input via the URL without displaying results, this approach remains exploitable. Additionally, the lack of an API makes it difficult for end users to interact with the program without accessing the underlying code. A RESTful API requires a clear and structured mechanism for user interaction, which is currently missing.

Moving on to cryptography, I found no evidence of data encryption in the code. Artemis Financial must implement robust encryption to secure stored information and support international transactions while complying with global regulations.

Regarding error handling, I reviewed the DocData.java class and found only basic try and catch blocks, with no comprehensive error-handling mechanisms in place. Other parts of the code were not evaluated for error handling.

While the overall code quality was strong, the absence of an API, insufficient input validation, and reliance on URL-based input handling significantly undermine its functionality and security. Using the URL instead of a POST method creates vulnerabilities, such as potential data leaks in browser history. Addressing these issues is critical to improving the program's usability and security.

**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 | Vulnerability | Description | Solution |
| log4j-api-2.12.1.jar | |  |  | | --- | --- | |  | cpe:2.3:a:apache:log4j:2.12.1:\*:\*:\*:\*:\*:\*:\* | | 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. | Upgrade to version 2.13.2 to gain support for this feature. For earlier versions, you can globally enable hostname verification for SMTPS connections by setting the system property mail.smtp.ssl.checkserveridentity to true.  . |
| tomcat-embed-core-9.0.30.jar | cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:apache\_software\_foundation:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* | Apache Tomcat 10.0.0-M1 to 10.0.6, 9.0.0.M1 to 9.0.46 and 8.5.0 to 8.5.66 did not correctly parse the HTTP transfer-encoding request header in some circumstances leading to the possibility to request smuggling when used with a reverse proxy. Specifically: - Tomcat incorrectly ignored the transfer encoding header if the client declared it would only accept an HTTP/1.0 response; - Tomcat honoured the identify encoding; and - Tomcat did not ensure that, if present, the chunked encoding was the final encoding. | Upgrade to the latest Apache Tomcat |
| tomcat-embed-websocket-9.0.30.jar | cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:apache\_software\_foundation:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* | Apache Tomcat 10.0.0-M1 to 10.0.6, 9.0.0.M1 to 9.0.46 and 8.5.0 to 8.5.66 did not correctly parse the HTTP transfer-encoding request header in some circumstances leading to the possibility to request smuggling when used with a reverse proxy. Specifically: - Tomcat incorrectly ignored the transfer encoding header if the client declared it would only accept an HTTP/1.0 response; - Tomcat honored the identify encoding; and - Tomcat did not ensure that, if present, the chunked encoding was the final encoding. | Update to the latest Apache Tomcat |
| bcprov-jdk15on-1.46.jar | 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:legion-of-the-bouncy-castle-java-crytography-api:1.46:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:bouncycastle:the\_bouncy\_castle\_crypto\_package\_for\_java:1.46:\*:\*:\*:\*:\*:\*:\* | Legion of the Bouncy Castle Legion of the Bouncy Castle Java Cryptography APIs 1.58 up to but not including 1.60 contains a CWE-470: Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection') vulnerability in XMSS/XMSS^MT private key deserialization that can result in Deserializing an XMSS/XMSS^MT private key can result in the execution of unexpected code. This attack appear to be exploitable via A handcrafted private key can include references to unexpected classes which will be picked up from the class path for the executing application. This vulnerability appears to have been fixed in 1.60 and later. | Update bouncy castle to the latest update. |
| jackson-databind-2.10.2.jar | |  |  | | --- | --- | |  | [cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Afasterxml&cpe_product=cpe%3A%2F%3Afasterxml%3Ajackson-databind&cpe_version=cpe%3A%2F%3Afasterxml%3Ajackson-databind%3A2.10.2) cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:\*:\*:\*:\*:\*:\*:\* | | A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity. | Update to the latest Version |
| spring-aop-5.2.3.RELEASE.jar | 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:vmware:springsource\_spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* | In Spring Framework versions 5.2.0 - 5.2.8, 5.1.0 - 5.1.17, 5.0.0 - 5.0.18, 4.3.0 - 4.3.28, and older unsupported versions, the protections against RFD attacks from CVE-2015-5211 may be bypassed depending on the browser used through the use of a jsessionid path parameter. | Update to the latest version |
| hibernate-validator-6.0.18.Final.jar | cpe:2.3:a:redhat:hibernate\_validator:6.0.18:\*:\*:\*:\*:\*:\*:\* | A flaw was found in Hibernate Validator version 6.1.2.Final. A bug in the message interpolation processor enables invalid EL expressions to be evaluated as if they were valid. This flaw allows attackers to bypass input sanitation (escaping, stripping) controls that developers may have put in place when handling user-controlled data in error messages. | Upgrade to the latest hibernate-validator |
| snakeyaml-1.25.jar | cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\* | The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564.  Published: December 11, 2019; 10:15:10 PM -0500 | Migrate to the SnakeYAML Engine, which provides a configuration option to restrict aliases for collections. Note that while the aliases for collections can be limited, the aliases for scalars remain unrestricted and cannot be controlled. |

**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.

Most of these vulnerabilities can be addressed by upgrading to the latest versions. Furthermore, switching the Snakeyaml version and restricting aliases for collections can provide added security.