# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **9/21/2025** | **Ryan Gilstrap** |  |

## 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

Ryan Gilstrap

**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?

As a financial institution, Artemis has access to its customers' sensitive data. The sensitivity of this data underscores the importance of secure communications from all avenues, whether with customers, other financial institutions, or internally. While no direct mention of international transactions has been made, it is something that should at least be planned for and a secure mindset built for the client. As a financial institution, Artemis is subject to laws governing its communications and processes. The largest threat to Artemis would be data breaches resulting in the loss of client data or funds. Other notable threats include insider threats and human factors. Modernization, such as the use of open-source libraries, would be beneficial for cost and efficiency; however, we must be careful to use trusted libraries or create our own to minimize security vulnerabilities.

**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 is the first step in minimizing vulnerabilities. Implementing input validation for both employee and customer input fields will provide user feedback if an input is incorrect and prevent potentially malicious attacks from occurring. Using and creating quality code is also an important consideration; alongside quality code, proper error handling should also be a priority. Quality coding through best practices minimizes vulnerabilities and ensures code remains easily workable for future updates and fixes. In line with best practices, proper error handling will enable the development team to identify and resolve any bugs or missing code segments easily.

**3. Manual Review**

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

* CRUD Controller uses business name. possible data leak exploit.
* Application takes input through URL instead of POST
* Lack of input validation
* Lack of error handling
* Possible breach exploit with user input due to lack of POST
* Lack of any cryptography
* No API implementation

**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 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. |
| 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 Apache Tomcat 10.0.6 or later |
| 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 honoured the identify encoding; and - Tomcat did not ensure that, if present, the chunked encoding was the final encoding. | - Upgrade to Apache Tomcat 10.0.6 or later |
| 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 bouncycastle to:  Version update to 1.60 |
| 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 current 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. | Upgrade to current 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 hibernate-validator-6.0.20 |
| 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 SnakeYAML Engine. It has a configuration option to restrict aliases for collections (the aliases for scalars cannot grow and they are not restricted) |

**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.

Ensuring all dependencies are updated to the current versions will solve the majority of the issues. Changing the version of snakyaml and restricting collected aliases will help with this one. Aside from that, building a user-friendly API, providing input validation and proper error handling, and implementing some form of cryptography will ensure that significant vulnerabilities are addressed.