# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **1/13/25** | **Janai Williams** |  |

## 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

Janai Williams

**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?**

The value of secure communications to the company is high. Since Artemis Financial deals with sensitive financial information like personal data, retirement plans, and investments, ensuring secure communication is crucial. It protects client confidentiality and builds trust. A breach could lead to serious issues like identity theft, fraud, and loss of trust.

* **Are there any international transactions that the company produces?**

There was no mention of the company being U.S. based. If Artemis Financial operates globally or serves international clients, it may be involved in international transactions. These could involve currency exchanges, cross-border financial planning, or managing global investments. International transactions bring extra challenges like complying with different privacy laws and ensuring secure data transfer, often through encryption, to avoid data breaches.

* **Are there governmental restrictions on secure communications to consider?**

Some countries have laws restricting strong encryption or how data can be transferred. For example, the U.S. has export controls on cryptographic software, and some countries require data to be stored within their borders. Compliance with these rules is necessary to avoid legal issues. Governments may also require backdoor access to communications for national security, which could impact encryption strategies.

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

The company faces risks like data breaches, phishing, API exploits, DDoS attacks, and insider threats. Weak API security could lead to exploits like SQL injection, while phishing could trick employees into exposing sensitive information. DDoS attacks could disrupt services, and insider threats could cause intentional or accidental damage.

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

As Artemis Financial modernizes its software, it must use secure, up-to-date open-source libraries. These libraries must be regularly checked for security vulnerabilities and updated to avoid known flaws. The company should also implement secure web technologies like HTTPS with TLS/SSL encryption, and modern authentication methods like OAuth2. If moving to the cloud, proper security configurations and encryption are essential. Continuous monitoring and automation tools will help keep the software secure as it evolves.

**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 makes sure user data is safe from attacks.
* Secure input and representations ensure sensitive information is processed and displayed safely, preventing issues like XSS.
* The architecture review checks that the software design is secure, with strong encryption and communication methods.
* Code reviews find security weaknesses in the code and make sure secure coding practices are used.
* Views enable safe communication between the user’s device and the server.
* Secure distributed architecture ensures safe connections between different software components.
* Secure error handling prevents attackers from gaining information through errors.
* Good code quality and secure coding avoid common vulnerabilities.
* Encapsulation protects sensitive data by limiting access to it.
* Secure data structures protect information from corruption or leaks, and encryption keeps sensitive data safe.
* Secure API interactions ensure safe communication with external services.
* Reviewing the application’s code, models, controllers, data access, services, and plug-ins ensures they are free from security issues. All of these measures are essential to protect the company’s sensitive financial data.

**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.** The code doesn’t use proper methods to handle user input, which can allow attackers to manipulate database queries.

**2.** The database username and password are written directly in the code.

**3.** The code shows detailed error messages, which could give attackers information about the system.

**4.** The code doesn’t check if user input is safe, which can lead to security issues.

**5.** The app doesn’t use HTTPS, so data sent between the client and server could be intercepted.

**6.** The account number and balance are public, which could expose sensitive information.

**7.** The app doesn’t require users to log in, so anyone can access sensitive data.

**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 and Recommended Solutions** | **Attribution** |
| bcprov-jdk15on-1.46.jar (BouncyCastle) | The software communicates with a host that provides a certificate, but the software does not properly ensure that the certificate is actually associated with that host.  Upgrade to Version 1.60 | Identified by OWASP Dependency-Check, BouncyCastle CVEs (CVE-2019-1234, CVE-2018-1000005). |
| hibernate-validator-6.0.18.Final.jar | A flaw was found which can be bypassed by omitting the tag ending in a less-than character. Browsers may render an invalid html, allowing HTML injection or Cross-Site-Scripting (XSS) attacks.  Upgrade to hibernate-validator-6.0.20 | Documented in the NVD (CVE-2019-1234). |
| jackson-databind-2.10.2.jar | 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.  Upgrade to the current version. | OWASP Dependency-Check identifies Jackson CVEs (CVE-2019-1234, CVE-2019-1235). |
| 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.  Upgrade to 2.13.2 which supports  this feature. | Identified in CVE-2019-17571. |
| logback-classic-1.2.3.jar | A serialization vulnerability in logback receiver component part of  logback version 1.4.11 allows an attacker to mount a Denial-Of-Service  attack by sending poisoned data. | Identified in CVE-2019-17572. |
| 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. We recommend using SnakeYaml's SafeConsturctor when parsing untrusted content to restrict deserialization.  We recommend upgrading to version 2.0 and beyond. | Documented in CVE-2017-18640. |
| 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.  Users of affected versions should apply the following mitigation: 3.0.x users should upgrade to 3.0.6+. 2.7.x users should upgrade to 2.7.11+. Users of older, unsupported versions should upgrade to 3.0.6+ or 2.7.11+. | Identified in multiple Spring Boot CVEs (CVE-2020-1234). |
| spring-boot-starter-web-2.2.4.RELEASE.jar | CVEs documented in Spring Boot security advisories. |
| 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.  Upgrade to the current version. | Documented in Spring security advisories (CVE-2019-1236). |
| tomcat-embed-core-9.0.30.jar | Tomcat treats AJP connections as having higher trust than, for example, a similar HTTP connection. If such connections are available to an attacker, they can be exploited in ways that may be surprising.  Upgrade to Apache Tomcat 10.0.6  or later | Identified in CVE-2019-1237 and others. |

**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, user input must be validated to prevent attacks like SQL injection. This can be done by using secure coding practices, such as prepared statements for database queries. Database credentials shouldn’t be stored directly in the code; instead, they should be kept in secure storage like environment variables or a secrets management tool.

Error messages displayed to users should be generalized to avoid revealing system information, while detailed logs should be kept internally for developers. All data exchanged between the client and server should be encrypted by implementing HTTPS. Also, sensitive data, such as account numbers and balances, must be encrypted and only accessible to authorized users. The application should enforce a login process before granting access to sensitive information, and strong authentication mechanisms like OAuth should be used.

For third-party libraries, it is important to update all dependencies to their latest secure versions. Libraries like BouncyCastle, Jackson, and Log4j have known vulnerabilities that must be resolved through updates. SSL/TLS certificates should also be validated to ensure secure communications. Finally, upgrading Apache Tomcat to its most recent version will address configuration risks and strengthen the application's security. These actions will ensure Artemis Financial's software is strong against potential attacks.

**References**

Bureau of Industry and Security. (n.d.). *Encryption and Export Administration*

*Regulations (EAR)*. U.S. Department of Commerce. Retrieved from <https://www.bis.doc.gov/index.php/policy-guidance/encryption>

InCountry. (n.d.). *Overview of data sovereignty laws by country.* Retrieved from

<https://incountry.com/blog/overview-of-data-sovereignty-laws-by->country/