# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **May 25, 2024** | **Trangdai Huynh** |  |

## 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

Trangdai Huynh

**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 comprehensive financial planning services, helping individuals navigate everything from savings and retirement strategies to investments and insurance. Given the highly sensitive nature of client information, secure communication is paramount. This includes data such as Social Security Numbers (SSNs) and tax records. While the text doesn't explicitly state Artemis Financials’ location, its global reach is likely due to the mention of international transactions. This necessitates compliance with various government regulations regarding secure communication. A key concern would be preventing the exposure of trade secrets. The sheer volume of sensitive client information demands a robust security posture. This includes heavy encryption for all data accessible from outside the company. Additionally, maintaining a rigorous update schedule for software and systems is crucial to address vulnerabilities and emerging security threats.

**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: To safeguard user data, Artemis Financial prioritizes robust input validation. This process ensures that only authorized users can access and modify sensitive information. Input validation typically involves verifying data types and formats (e.g., ensuring Social Security Numbers follow a specific format).
* Code Quality: One aspect of code quality that directly impacts security is access control. Well-written code ensures users can only access their information. This means preventing unauthorized access to other users’ data, the server itself, or any other restricted areas.
* API: Artemis Financials’ operations likely involve both internal and external systems. An API (Application Programming Interface) would serve as a secure intermediary, facilitating communication and data exchange between these systems. The API would strictly control what data is accessible, ensuring only authorized information is shared.
* Code Error: Implementing robust error handling within the API is crucial. This process identifies and addresses potential issues within the code. By effectively handling errors, Artemis Financial can minimize the risk of vulnerabilities that could expose sensitive user information.
* Cryptography: For a company like Artemis Financial, which deals with international transactions and potentially diverse currencies, implementing robust cryptography is paramount. Cryptography ensures that user information remains secure throughout the entire financial process, regardless of location. This protects sensitive data from unauthorized access, even across international borders.

**3. Manual Review**

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

My review of the code, which included a vulnerability assessment, revealed several areas for improvement in security:

**Input Validation**

* **POM.xml:** While the Apache Validator wasn't explicitly found in the POM.xml file, further investigation is recommended to determine if other input validation dependencies are present.
* **Greeting Controller:** A critical vulnerability exists due to the lack of input validation in the Greeting Controller. This exposes the system to potential harm from malicious data. Implementing input validation will ensure user input adheres to expected formats and prevent such attacks.

**Code Quality and Error Handling**

Overall code quality was acceptable, but error-handling mechanisms were not implemented. Error handling is essential for a robust system, as it allows the system to gracefully handle unexpected situations and avoid potential crashes. Integrating error-handling routines throughout the codebase is recommended.

**API Security**

The API exhibits several security vulnerabilities that require attention:

* **User Input Exposure:** User input wasn't sent through a secure method, likely not a POST request. This could lead to data breaches if sensitive information is intercepted. Implementing secure methods like POST for user input transmission will mitigate this risk.
* **Missing Cryptography:** The absence of cryptography leaves user data vulnerable to unauthorized access, especially for a system dealing with sensitive financial information. Implementing robust encryption mechanisms within the API is crucial for protecting user 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

**A screenshot of a computer

Description automatically generated**

### 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. In Bouncy Castle JCE Provider version 1.55 and earlier the DSA does not fully validate ASN.1 encoding of signature on verification. It is possible to inject extra elements in the sequence making up the signature and still have it validate, which in some cases may allow the introduction of 'invisible' data into a signed structure.

### classmate-1.5.1.jar

Library for introspecting types with full generic information including resolving of field and method types.

### hibernate-validator-6.0.18.Final.jar

Hibernates Bean Validation (JSR-380) reference implementation. 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.

### jackson-core-2.10.2.jar

Core Jackson processing abstractions (aka Streaming API), implementation for JSON. Upgrading to the current version.

### jackson-databind-2.10.2.jar

General data-binding functionality for Jackson: works on core streaming API. 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.

### jakarta.annotation-api-1.3.5.jar

Jakarta Annotations API. Upgrading to the current version.

### jakarta.validation-api-2.0.2.jar

Jakarta Bean Validation API. Upgrading to the current version.

### jboss-logging-3.4.1.Final.jar

The JBoss Logging Framework. Upgrading to the current version.

### jul-to-slf4j-1.7.30.jar

JUL to SLF4J bridge. Upgrading to the current version.

### log4j-to-slf4j-2.12.1.jar

The Apache Log4j binding between Log4j 2 API and SLF4J. Upgrade to the current version.

### logback-core-1.2.3.jar

logback-core module. 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. Upgrading to 2.13.2.

### slf4j-api-1.7.30.jar

The slf4j API. Upgrading to the current version.

### snakeyaml-1.25.jar

YAML 1.1 parser and emitter for Java. 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.

### spring-boot-2.2.4.RELEASE.jar

Spring Boot. 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+.

### spring-boot-starter-web-2.2.4.RELEASE.jar

Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container.

### spring-core-5.2.3.RELEASE.jar

Spring Core. 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. If the application is deployed as a Spring Boot executable jar, i.e. the default, it is not vulnerable to the exploit. However, the nature of the vulnerability is more general, and there may be other ways to exploit it.

### spring-web-5.2.3.RELEASE.jar

Spring Web. Pivotal Spring Framework through 5.3.16 suffers from a potential remote code execution (RCE) issue if used for Java deserialization of untrusted data. Depending on how the library is implemented within a product, this issue may or not occur, and authentication may be required. NOTE: the vendor's position is that untrusted data is not an intended use case. The product's behavior will not be changed because some users rely on deserialization of trusted data.

### spring-webmvc-5.2.3.RELEASE.jar

Spring Web MVC. 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. If the application is deployed as a Spring Boot executable jar, i.e. the default, it is not vulnerable to the exploit. However, the nature of the vulnerability is more general, and there may be other ways to exploit it.

### tomcat-embed-core-9.0.30.jar

Core Tomcat implementation. When using the Apache JServ Protocol (AJP), care must be taken when trusting incoming connections to Apache Tomcat. 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. In Apache Tomcat 9.0.0.M1 to 9.0.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99, Tomcat shipped with an AJP Connector enabled by default that listened on all configured IP addresses. It was expected (and recommended in the security guide) that this Connector would be disabled if not required. This vulnerability report identified a mechanism that allowed: - returning arbitrary files from anywhere in the web application - processing any file in the web application as a JSP Further, if the web application allowed file upload and stored those files within the web application (or the attacker was able to control the content of the web application by some other means) then this, along with the ability to process a file as a JSP, made remote code execution possible. It is important to note that mitigation is only required if an AJP port is accessible to untrusted users. Users wishing to take a defence-in-depth approach and block the vector that permits returning arbitrary files and execution as JSP may upgrade to Apache Tomcat 9.0.31, 8.5.51 or 7.0.100 or later. A number of changes were made to the default AJP Connector configuration in 9.0.31 to harden the default configuration. It is likely that users upgrading to 9.0.31, 8.5.51 or 7.0.100 or later will need to make small changes to their configurations.

### tomcat-embed-el-9.0.30.jar

Core Tomcat implementation. When using the Apache JServ Protocol (AJP), care must be taken when trusting incoming connections to Apache Tomcat. 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. In Apache Tomcat 9.0.0.M1 to 9.0.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99, Tomcat shipped with an AJP Connector enabled by default that listened on all configured IP addresses. It was expected (and recommended in the security guide) that this Connector would be disabled if not required. This vulnerability report identified a mechanism that allowed: - returning arbitrary files from anywhere in the web application - processing any file in the web application as a JSP Further, if the web application allowed file upload and stored those files within the web application (or the attacker was able to control the content of the web application by some other means) then this, along with the ability to process a file as a JSP, made remote code execution possible. It is important to note that mitigation is only required if an AJP port is accessible to untrusted users. Users wishing to take a defence-in-depth approach and block the vector that permits returning arbitrary files and execution as JSP may upgrade to Apache Tomcat 9.0.31, 8.5.51 or 7.0.100 or later. A number of changes were made to the default AJP Connector configuration in 9.0.31 to harden the default configuration. It is likely that users upgrading to 9.0.31, 8.5.51 or 7.0.100 or later will need to make small changes to their configurations.

### tomcat-embed-websocket-9.0.30.jar

Core Tomcat implementation. When using the Apache JServ Protocol (AJP), care must be taken when trusting incoming connections to Apache Tomcat. 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. In Apache Tomcat 9.0.0.M1 to 9.0.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99, Tomcat shipped with an AJP Connector enabled by default that listened on all configured IP addresses. It was expected (and recommended in the security guide) that this Connector would be disabled if not required. This vulnerability report identified a mechanism that allowed: - returning arbitrary files from anywhere in the web application - processing any file in the web application as a JSP Further, if the web application allowed file upload and stored those files within the web application (or the attacker was able to control the content of the web application by some other means) then this, along with the ability to process a file as a JSP, made remote code execution possible. It is important to note that mitigation is only required if an AJP port is accessible to untrusted users. Users wishing to take a defence-in-depth approach and block the vector that permits returning arbitrary files and execution as JSP may upgrade to Apache Tomcat 9.0.31, 8.5.51 or 7.0.100 or later. A number of changes were made to the default AJP Connector configuration in 9.0.31 to harden the default configuration. It is likely that users upgrading to 9.0.31, 8.5.51 or 7.0.100 or later will need to make small changes to their configurations.

**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.

Updating libraries to their latest versions is a good first step. These updates often include critical security patches and bug fixes that address known vulnerabilities. However, it's important to remember that updates alone might not solve all security issues.