# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **09/22/2024** | **Landon Phillips** |  |

## Client



## Developer

Landon Phillips

**1. Interpreting Client Needs**

The client, Artemis Financial, a consulting company that develops financial plans for customers, is looking to completely modernize its operations and use the latest and greatest software security to ensure it is protected from external threats. Specifically, Artemis needs to secure its web-based application, which utilizes a RESTful API and handles sensitive financial data.

The value of secure communications is immeasurable, especially when dealing with customer financial data. Secure communication is crucial for protecting data and transactions. If security is breached, financial and reputational damage could be significant, as could user trust.

When dealing with financial companies, it is safe to assume they will most likely make international transactions at some point. However, it was not mentioned that Artemis Financial produces international transactions. Ensuring international transactions are secure, comply with international standards, and meet legal requirements is crucial.

When considering governmental restrictions on secure communications and financial services, it is essential to note that Artemis Financial must comply with various regulations depending on the nature of the transaction. This involves ensuring that data is securely handled and maintaining financial transparency. Additionally, the company's location could entail further implicit regulations. Failure to adhere to these standards could result in financial losses and reputational damage.

External threats are always present, and even well-prepared companies can still fall victim to them. When money is involved, companies become targets for malicious actors. Some immediate threats to be aware of include data breaches due to poor API security, ransomware that could destroy customer data, phishing attacks that can compromise user credentials and account security, as well as evolving threats such as AI attacks, bots attempting brute force attacks, and threats from third-party integrations.

Many modern web applications use open-source libraries. This can introduce vulnerabilities if these libraries are outdated or poorly maintained. As Artemis Financial continues to modernize, new technologies may be introduced to ensure that these libraries stay up-to-date and known vulnerabilities are addressed.

**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:** It is crucial to validate all inputs in a web-based financial services application. Validating inputs helps prevent the entry of malicious or faulty data, which could otherwise lead to vulnerabilities such as SQL injection or cross-site scripting. Artemis Financial handles sensitive financial information, and ensuring all user inputs are validated can help prevent malicious actors from injecting harmful code and compromising the system.
* **API Security:** In Artemis Financial's application, API Security is a must because the application relies on a RESTful API, which exposes various endpoints handling critical user inputs and data, making them potential targets for attackers seeking to bypass security or access restricted functionality. To mitigate these risks, robust authentication, authorization, and encryption mechanisms must be in place to ensure that only verified users can access the API, control which users can perform specific actions, and protect data during transmission. Sensitive information should never be transmitted through request headers, as these can be intercepted or altered by intermediaries such as proxies. Securing the API is vital to protecting customer data, ensuring system integrity, and preventing unauthorized access, making it a top priority in the ongoing modernization of Artemis Financial's web application.
* **Cryptography:** When it comes to storing and transmitting sensitive data, it is imperative to encrypt the data to protect against interception or corruption. It is crucial for applications using RESTful APIs to implement proper encryption. All sensitive data within the system should be encrypted to ensure security. By utilizing strong encryption methods, sensitive data can be safeguarded even if intercepted.
* **Code Error:** It's crucial to have secure error handling in place to avoid leaking sensitive information in error messages, which could be exploited by malicious actors to identify vulnerabilities. This is particularly important in financial services applications. During development, developers often display error messages, so it's vital to make sure that these messages don't expose internal application details. The application should show generic messages to users while still providing useful information to the developers.
* **Code Quality:** It is crucial to ensure that the Artemis Financial application utilizes secure coding practices and proper patterns throughout its development. The application should be built with security in mind, incorporating object-oriented programming principles such as encapsulation to prevent the exposure of commands or data. This approach leads to higher code quality and reduces the risk of vulnerabilities.

**3. Manual Review**

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

* **Vulnerability - Input Validation:** In GreetingController.java and CrudController.java, inputs from users are accepted without any validation or sanitization.
* **Vulnerability –** Secure API – No authentication or authorization is applied in the /greeting or /read endpoints found in GreetingController.java and CrudController.java, allowing anyone to access these endpoints.

A screen shot of a computer

Description automatically generatedA screen shot of a computer

Description automatically generated

* **Vulnerability – Cryptography** – No use of encryption for sensitive data like account numbers or balances in the customer class exposing sensitive information.

A screenshot of a computer program

Description automatically generated

* **Vulnerability – Secure Error Handling** – In the read\_document method in DocData.java catches SQLException and prints the stack trace. This could reveal sensitive information.

A screen shot of a computer program

Description automatically generated

* **Vulnerability—Secure Error Handling—**GreetingController.java, CRUDController.java, customer.java, and myDateTime.java all lack error handling, which is crucial for handling edge cases or unexpected behavior across the application.
* **Vulnerability—Code Quality —** myDateTime class has placeholder methods with no implementations.

A screen shot of a computer program

Description automatically generated

* **Vulnerability— Code Quality —** Java Version should be updated to Java 17 and Spring Boot can also be upgraded to 3.1.4

**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 | Code | Description |
| [bcprov-jdk15on-1.46.jar](#l1_991c96a4e31e6c19e2b9136c8955bd423f2d) | [**CVE-2023-33202**](https://nvd.nist.gov/vuln/detail/CVE-2023-33202) | Bouncy Castle for Java before 1.73 contains a potential Denial of Service (DoS) issue within the Bouncy Castle org.bouncycastle.openssl.PEMParser class. This class parses OpenSSL PEM encoded streams containing X.509 certificates, PKCS8 encoded keys, and PKCS7 objects. Parsing a file that has crafted ASN.1 data through the PEMParser causes an OutOfMemoryError, which can enable a denial of service attack. (For users of the FIPS Java API: BC-FJA 1.0.2.3 and earlier are affected; BC-FJA 1.0.2.4 is fixed.) |
| [hibernate-validator-6.0.18.Final.jar](#l3_7fd00bcd87e14b6ba66279282ef15efa30dd) | [**CVE-2020-10693**](https://nvd.nist.gov/vuln/detail/CVE-2020-10693) | 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-databind-2.10.2.jar](#l5_0528de95f198afafbcfb0c09d2e43b6e0ea6) | [**CVE-2023-35116**](https://nvd.nist.gov/vuln/detail/CVE-2023-35116)  [**CVE-2021-46877**](https://nvd.nist.gov/vuln/detail/CVE-2021-46877)  [**CVE-2022-42004**](https://nvd.nist.gov/vuln/detail/CVE-2022-42004)  [**CVE-2022-42003**](https://nvd.nist.gov/vuln/detail/CVE-2022-42003)  [**CVE-2020-36518**](https://nvd.nist.gov/vuln/detail/CVE-2020-36518)  [**CVE-2020-25649**](https://nvd.nist.gov/vuln/detail/CVE-2020-25649) | Jackson-databind through 2.15.2 allows attackers to cause a denial of service or other unspecified impact via a crafted object that uses cyclic dependencies. NOTE: the vendor's perspective is that this is not a valid vulnerability report, because the steps of constructing a cyclic data structure and trying to serialize it cannot be achieved by an external attacker. |
| [log4j-api-2.12.1.jar](#l10_a55e6d987f50a515c9260b0451b4fa217dc) | [**CVE-2021-44832**](https://nvd.nist.gov/vuln/detail/CVE-2021-44832)  [**CVE-2021-45105**](https://nvd.nist.gov/vuln/detail/CVE-2021-45105)  [**CVE-2021-45046**](https://nvd.nist.gov/vuln/detail/CVE-2021-45046)  [**CVE-2021-44228**](https://nvd.nist.gov/vuln/detail/CVE-2021-44228)  [**CVE-2020-9488**](https://nvd.nist.gov/vuln/detail/CVE-2020-9488) | Apache Log4j2 versions 2.0-beta7 through 2.17.0 (excluding security fix releases 2.3.2 and 2.12.4) are vulnerable to a remote code execution (RCE) attack when a configuration uses a JDBC Appender with a JNDI LDAP data source URI when an attacker has control of the target LDAP server. This issue is fixed by limiting JNDI data source names to the java protocol in Log4j2 versions 2.17.1, 2.12.4, and 2.3.2. |
| [logback-core-1.2.3.jar](#l12_864344400c3d4d92dfeb0a305dc87d95367) | [**CVE-2023-6378**](https://nvd.nist.gov/vuln/detail/CVE-2023-6378)  [**CVE-2021-42550**](https://nvd.nist.gov/vuln/detail/CVE-2021-42550) | 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. |
| [snakeyaml-1.25.jar](#l14_8b6e01ef661d8378ae6dd7b511a7f2a33fa) | [**CVE-2022-1471**](https://nvd.nist.gov/vuln/detail/CVE-2022-1471)  [**CVE-2022-41854**](https://nvd.nist.gov/vuln/detail/CVE-2022-41854)  [**CVE-2022-38752**](https://nvd.nist.gov/vuln/detail/CVE-2022-38752)  [**CVE-2022-38751**](https://nvd.nist.gov/vuln/detail/CVE-2022-38751) | 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](#l15_225a4fd31156c254e3bb92adb42ee8c6de8) | [**CVE-2023-20883**](https://nvd.nist.gov/vuln/detail/CVE-2023-20883)  [**CVE-2023-20873**](https://nvd.nist.gov/vuln/detail/CVE-2023-20873)  [**CVE-2022-27772**](https://nvd.nist.gov/vuln/detail/CVE-2022-27772) | In Spring Boot versions 3.0.0 - 3.0.6, 2.7.0 - 2.7.11, 2.6.0 - 2.6.14, 2.5.0 - 2.5.14 and older unsupported versions, there is potential for a denial-of-service (DoS) attack if Spring MVC is used together with a reverse proxy cache. |
| [spring-boot-starter-web-2.2.4.RELEASE.jar](#l16_ec75d01d212b5229c16d872fb127744c0ed) | [**CVE-2023-20883**](https://nvd.nist.gov/vuln/detail/CVE-2023-20883)  [**CVE-2023-20873**](https://nvd.nist.gov/vuln/detail/CVE-2023-20873)  [**CVE-2022-27772**](https://nvd.nist.gov/vuln/detail/CVE-2022-27772) | In Spring Boot versions 3.0.0 - 3.0.6, 2.7.0 - 2.7.11, 2.6.0 - 2.6.14, 2.5.0 - 2.5.14 and older unsupported versions, there is potential for a denial-of-service (DoS) attack if Spring MVC is used together with a reverse proxy cache. |
| [spring-core-5.2.3.RELEASE.jar](#l17_3734223040040e8c3fecd5faa3ae8a1ed6d) | [**CVE-2023-20863**](https://nvd.nist.gov/vuln/detail/CVE-2023-20863)  [**CVE-2023-20861**](https://nvd.nist.gov/vuln/detail/CVE-2023-20861)  [**CVE-2022-22965**](https://nvd.nist.gov/vuln/detail/CVE-2022-22965) | In spring framework versions prior to 5.2.24 release+ ,5.3.27+ and 6.0.8+ , it is possible for a user to provide a specially crafted SpEL expression that may cause a denial-of-service (DoS) condition. |
| [spring-expression-5.2.3.RELEASE.jar](#l18_d0c6bb10758805b2153c589686b8045554b) | [**CVE-2023-20863**](https://nvd.nist.gov/vuln/detail/CVE-2023-20863)  [**CVE-2023-20861**](https://nvd.nist.gov/vuln/detail/CVE-2023-20861)  [**CVE-2022-22965**](https://nvd.nist.gov/vuln/detail/CVE-2022-22965) | In spring framework versions prior to 5.2.24 release+ ,5.3.27+ and 6.0.8+ , it is possible for a user to provide a specially crafted SpEL expression that may cause a denial-of-service (DoS) condition. |
| [spring-web-5.2.3.RELEASE.jar](#l19_dd386a02e40b915ab400a3bf9f586d2dc4c) | [**CVE-2023-20863**](https://nvd.nist.gov/vuln/detail/CVE-2023-20863)  [**CVE-2023-20861**](https://nvd.nist.gov/vuln/detail/CVE-2023-20861)  [**CVE-2022-22965**](https://nvd.nist.gov/vuln/detail/CVE-2022-22965) | In spring framework versions prior to 5.2.24 release+ ,5.3.27+ and 6.0.8+ , it is possible for a user to provide a specially crafted SpEL expression that may cause a denial-of-service (DoS) condition. |
| [spring-webmvc-5.2.3.RELEASE.jar](#l20_745a62502023d2496b565b7fe102bb1ee22) | [**CVE-2023-20863**](https://nvd.nist.gov/vuln/detail/CVE-2023-20863)  [**CVE-2023-20861**](https://nvd.nist.gov/vuln/detail/CVE-2023-20861)  [**CVE-2022-22965**](https://nvd.nist.gov/vuln/detail/CVE-2022-22965) | In spring framework versions prior to 5.2.24 release+ ,5.3.27+ and 6.0.8+ , it is possible for a user to provide a specially crafted SpEL expression that may cause a denial-of-service (DoS) condition. |
| [tomcat-embed-core-9.0.30.jar](#l21_ad32909314fe2ba02cec036434c0addd19b) | [**CVE-2024-21733**](https://nvd.nist.gov/vuln/detail/CVE-2024-21733)  [**CVE-2023-46589**](https://nvd.nist.gov/vuln/detail/CVE-2023-46589)  [**CVE-2023-45648**](https://nvd.nist.gov/vuln/detail/CVE-2023-45648) | Improper Input Validation vulnerability in Apache Tomcat.Tomcat from 11.0.0-M1 through 11.0.0-M10, from 10.1.0-M1 through 10.1.15, from 9.0.0-M1 through 9.0.82 and from 8.5.0 through 8.5.95 did not correctly parse HTTP trailer headers. A trailer header that exceeded the header size limit could cause Tomcat to treat a single request as multiple requests leading to the possibility of request smuggling when behind a reverse proxy. Users are recommended to upgrade to version 11.0.0-M11 onwards, 10.1.16 onwards, 9.0.83 onwards or 8.5.96 onwards, which fix the issue. |
| [tomcat-embed-websocket-9.0.30.jar](#l23_33157f6bc5bfd03380ebb5ac476db0600a0) | [**CVE-2024-21733**](https://nvd.nist.gov/vuln/detail/CVE-2024-21733)  [**CVE-2023-46589**](https://nvd.nist.gov/vuln/detail/CVE-2023-46589)  [**CVE-2023-45648**](https://nvd.nist.gov/vuln/detail/CVE-2023-45648) | Improper Input Validation vulnerability in Apache Tomcat.Tomcat from 11.0.0-M1 through 11.0.0-M10, from 10.1.0-M1 through 10.1.15, from 9.0.0-M1 through 9.0.82 and from 8.5.0 through 8.5.95 did not correctly parse HTTP trailer headers. A trailer header that exceeded the header size limit could cause Tomcat to treat a single request as multiple requests leading to the possibility of request smuggling when behind a reverse proxy. Users are recommended to upgrade to version 11.0.0-M11 onwards, 10.1.16 onwards, 9.0.83 onwards or 8.5.96 onwards, which fix the issue. |
|  |  |  |
|  |  |  |

**5. Mitigation Plan**

To address the vulnerabilities found, we will implement proper input validation across the application, specifically in the GreetingController and CrudController. This will ensure that only correct data enters the system. We will also implement new error-handling logic across the application to ensure proper error messages are displayed, and sensitive data is not exposed to any unauthorized users. It is imperative to secure the API with authentication and authorization to ensure that only authorized users can use endpoints. Additionally, we will integrate cryptography into the application by encrypting sensitive data in the customer.java files. Furthermore, we can improve the code quality by removing unused code and updating project dependencies such as Java and Spring Boot. Lastly, we will need to address other identified vulnerabilities from static testing. Several dependencies also need to be updated. The mitigation plan includes,

* Implementing proper input validation
* Implement proper error handling through the application
* Upgrade Spring framework and Java frameworks
* Upgrade other Dependency found in static testing
* Secure the API with proper Authentication and authorization
* Implement encryption on sensitive data