# CS 305 Project One

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **1/22/25** | **Logan Boyd** | **Introduction information based on customer needs.** |

## 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

Logan BOyd

**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 is a financial services firm in which they must prioritize secure communications to protect sensitive client data such as biometric information, social security numbers, and account details. Ensuring the confidentiality of internal and external communications is vital to maintain client trust and comply with legal obligations. Given the company's international transactions, it must also navigate varying global regulations on data privacy, retention, and secure communications, making compliance with international and domestic laws.

The company faces a range of external threats, from DoS attacks to sophisticated data breaches that could result in financial losses and reputational damage. High-level encryption must be implemented to secure data storage and transmission, as processing advancements could render weaker encryption obsolete. Additionally, Artemis should remain vigilant about the use of open-source libraries and evolving technologies, ensuring that software is updated regularly to address vulnerabilities. By prioritizing modernization and robust security measures, Artemis Financial can effectively safeguard its operations and protect its clients.

**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.

The application must address several critical security areas to ensure protection against vulnerabilities. **Input validation** is essential to sanitize user inputs, preventing malicious attacks such as SQL injection or other errant behavior caused by abnormal inputs. This is especially important for handling sensitive data.

**APIs** are another crucial area, as the application likely interacts with end-user browsers and possibly third-party software. A well-designed API will define acceptable methods for accessing data while preventing unauthorized access. This includes using authentication and encryption to secure connections, as these APIs form the backbone of how users and systems interact with the application.

**Cryptography** is cruicial for securing sensitive client data during storage and transmission, especially for international transactions. Using encryption will ensure compliance with regional and global data protection laws while mitigating risks of interception or unauthorized access.

Finally, **code quality and error handling** play significant roles in maintaining security. High-quality code minimizes unintended data exposure, while comprehensive error handling prevents attackers from gaining insight into the application’s structure or 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.

After looking through the code I located a few vulnerabilities that could be of concern to the security and functionality software application. Input validation is a significant concern, as user input is neither sanitized nor validated in multiple methods, such as the GreetingsController and CRUDController. For instance, raw user inputs like id are directly used without checks, making the application susceptible to SQL injection, buffer overflows, or other malicious input-based attacks. Input data should be sanitized, validated for length, and restricted to expected formats.

The application lacks a secure API implementation. Currently, user data is passed through the URL rather than secured methods, leaving sensitive information vulnerable to exposure in browser history.

Cryptographic vulnerabilities were also identified. No encryption is implemented for sensitive data, such as usernames, passwords, or financial information. For example, hardcoded credentials in the DocData class use plain text like "root," which is highly insecure. All passwords should be salted, hashed, and stored securely, and data transmission should be encrypted.

Lastly the error handling and code quality present notable gaps. While some classes include try/catch blocks, they fail to implement meaningful error management. For example, a database connection failure in the read\_document method of the DocData class returns an entire stack trace which would be exposing internal system details that could aid in attacks. Error messages must be sanitized to avoid revealing unnecessary information. Furthermore, methods like setMyDateTime are incomplete, and public class variables like account\_balance need to be encapsulated using proper getter and setter methods.

**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

|  |  |  |
| --- | --- | --- |
| Name | Summary | Solution |
| cprov-jdk15on-1.46.jar | 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.) | Update Bouncy Castle for Java. |
| hibernate-validator-6.0.18.Final.jar | 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. | Update Hibernate Validator. |
| 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. | Update Jackson Databind. |
| 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. |  |
| logback-classic-1.2.3.jar | It was found that the fix to address CVE-2021-44228 in Apache Log4j 2.15.0 was incomplete in certain non-default configurations. This could allows attackers with control over Thread Context Map (MDC) input data when the logging configuration uses a non-default Pattern Layout with either a Context Lookup (for example, $${ctx:loginId}) or a Thread Context Map pattern (%X, %mdc, or %MDC) to craft malicious input data using a JNDI Lookup pattern resulting in an information leak and remote code execution in some environments and local code execution in all environments. Log4j 2.16.0 (Java 8) and 2.12.2 (Java 7) fix this issue by removing support for message lookup patterns and disabling JNDI functionality by default. | Update to Log4j 2.16.0+ |
| logback-core-1.2.3.jar | In logback version 1.2.7 and prior versions, an attacker with the required privileges to edit configurations files could craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers. | Update to current version. |
| 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. | Upgrade to version 2.0+ |
| 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+. | Update to 3.0.5+ |
| 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. 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. | Update to current version. |
| tomcat-embed-core-9.0.30.jar | 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. | Update to 11.0.0+ |

**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.

To address the identified vulnerabilities in Artemis Financial’s software application, several key steps should be implemented to enhance security. Updating all dependent libraries should be the first thing. Outdated libraries can expose the application to known vulnerabilities, so maintaining up-to-date dependencies is critical.

After that we can look to implement input validation across all methods that handle user inputs. This includes validating data types, lengths, and acceptable formats to prevent SQL injection, buffer overflows, and other malicious attacks. Sensitive operations must also adopt secure methods to avoid exposing user data in URLs.

Lastly, conduct code reviews and compile to identify potential vulnerabilities and refine the application’s code quality.