# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **09/22/2024** | **Jerimey Burnside** | **Initial content** |

## 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

Jerimey Burnside

**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?

Importance of Secure Communication: Artemis Financial is a financial firm that handles both internal and external priority information for their clients. Given the sensitive nature of this data, secure communication is crucial for the company. This includes ensuring that all communications, whether internal or external, are protected to prevent any potential leaks of customer information.

Handling Domestic and International Transactions: As a financial firm, Artemis Financial processes transactions from both domestic and international sources. Although there are currently no state or federal restrictions specifically mandating secure communications, it is imperative for Artemis Financial to maintain high security standards to protect client data.

Data Protection Requirements: Artemis Financial deals with various types of sensitive information, including biometric data, social security numbers, account details, and proprietary client information. To safeguard this data, it is essential to implement measures that mask information during storage and transmission. Additionally, any trade secrets held by Artemis Financial must be securely protected.

Modernization and Library Updates: To stay ahead of potential security threats and bugs, Artemis Financial must ensure that their software libraries are regularly updated. Keeping these libraries current will help in implementing the latest fixes and maintaining robust security protocols within their applications.

**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.

After conducting a thorough evaluation of the security measures at Artemis Financial, I have pinpointed several areas that may be susceptible to vulnerabilities.

**Input Validation** – Ensuring the validation of user input is critical. The program must validate input strings to prevent potential failures or SQL injection attacks.

**APIs** – Given that the application will operate both internally and externally, such as on end users’ web browsers, a robust API is essential. This API should define acceptable methods for data access and ensure secure interactions, especially when integrating with third-party software.

**Cryptography** – Cryptographic measures are necessary for securing international transfers that include proprietary customer information. The data must be protected in compliance with North American regulations and those of the destination country.

**Code Error Handling** – Effective error handling should complement the API and input validation processes. Proper management of errors is crucial, particularly in input validation scenarios, to prevent unauthorized access or privilege violations.

**3. Manual Review**

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

Input Validation- I began the assessment by examining input validation mechanisms. Initially, I checked the POM.XML file to see if any Apache validator was being used. Subsequently, I reviewed the greeting controller and noticed that the input here did not seem to undergo any validation. Due to the absence of output, I couldn’t confirm whether validation was implemented.

API Examination- I searched for an API but found none that were functional. Despite this, the program could still access data insecurely. It retrieves data via URLs instead of using the POST method, which can expose sensitive information in browser history and make it susceptible to exploitation. Although no output is displayed, the program still accepts raw user input through URLs, posing a potential security risk. Without an API, end users have no clear way to interact with the program unless they access its codebase. A well-designed RESTful API should provide a clear interaction method for users.

Cryptography- Moving on to cryptography, I discovered that there was no data encryption implemented at all. For compliance with international regulations and secure handling of both stored information and international transactions, Artemis Financial needs to develop robust data encryption mechanisms.

Error Handling- When reviewing error handling practices, I found that the DocData.java class lacked proper error handling despite containing try and catch blocks. No other aspects of error handling were evaluated during this assessment.

Code Quality-Lastly, while the overall code quality was excellent, the absence of an API rendered the program not fully functional and significantly reduced its user-friendliness.

**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 Name | Vulnerability ID | 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 the latest version of Log4j (2.13.x) or use a different logging library if possible. |
| 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 |
| logback-core-1.2.3.jar | cpe:2.3:a:qos:logback:1.2.3:\*:\*:\*:\*:\*:\*:\* | 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. | upgrade to the latest version of Logback (1.14.x) or use a different logging library if possible. |
| bcprov-jdk15on-1.46.jar | cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46:\*:\*:\*:\*:\*:\*:\* | The TLS implementation in the Bouncy Castle Java library before 1.48 and C# library before 1.8 does not properly consider timing side-channel attacks on a noncompliant MAC check operation during the processing of malformed CBC padding, which allows remote attackers to conduct distinguishing attacks and plaintext-recovery attacks via statistical analysis of timing data for crafted packets, a related issue to CVE-2013-0169. | upgrade the version update to atleast 1.8 |
| jackson-databind-2.10.2.jar | cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\* | This is a popular library for serialization and deserialization in Java applications. It has been reported to have several vulnerabilities, including CVE-2018-5976 and CVE-2018-5977, which can be exploited by attackers to execute arbitrary code or access sensitive information. | upgrade to the latest version of Jackson (2.13.x) or use a different library for serialization and deserialization if possible. |
| spring-beans-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:springsource\_spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* | This is a dependency for Spring Boot, which is a popular framework for building web applications with Java. It has been reported to have several vulnerabilities, including CVE-2018-5976 and CVE-2018-5977, which can be exploited by attackers to execute arbitrary code or access sensitive information. To address these vulnerabilities, it is recommended to upgrade to the latest version of Spring Boot (2.3.x) or use a different framework if possible. | upgrade to the latest version of Spring Boot (2.3.x) or use a different framework if possible. |
| 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 the latest version of Hibernate Validator (6.1.x) |
| 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 | upgrade to the latest version of SnakeYAML (1.36.x) or use a different library for YAML parsing and serialization if possible. |

**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.

Upgrade Dependencies

Make Sure Everything is Up-to-Date: To make sure your project is secure, you should update all the libraries you use to their latest versions. For example, get the newest versions of Hibernate Validator (6.1.x), Jackson (2.13.x), Log4j (2.13.x), SnakeYAML (1.36.x), and Spring Boot (2.3.x). This way, any problems with older versions will be fixed.

Check User Input

Input validation:

Validate What Users Enter: To stop hackers from messing with your app through bad data, make sure you check everything users type in. You can use special libraries like OWASP ESAPI or write your own rules to check the data.

Use Safe Logging Tools

Pick a Secure Logging Library: Since some logging tools like Log4j and Logback have had security issues, it’s better to use safer options like SLF4J with Logstash or ELK Stack for logging. These tools offer better security and help you keep track of logs in one place.

Hide Your Code

Obfuscate Your Code: You can protect your app from being hacked by changing the names of your classes and functions, getting rid of information that helps people understand how it works, and using a secret code to keep things hidden.