# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **20May24** | **Gavin Bish** |  |

## 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

Gavin Bish

**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 specializes in creating comprehensive financial plans for individuals, encompassing savings, retirement, investments, and insurance. The importance of secure communication cannot be overstated, as the company deals with sensitive client information, such as Social Security numbers and tax data. Although there is no specific information indicating that Artemis Financial operates exclusively within the United States, it is reasonable to assume that the company may engage in international transactions. A key governmental restriction for the company involves preventing the exposure of trade secrets. Given the critical need to safeguard all forms of information, protecting client data from external threats is paramount, necessitating extensive encryption measures. Regular maintenance checks to address bug fixes and security vulnerabilities are essential for maintaining robust security standards at Artemis Financial. Avoiding common open source libraries would be key to maintaining security. If utilizing web services then ensuring vulnerabilities are always being scanned both on the front and backend of processes to ensure safety of data.

**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 is the first thing I want to talk about in the flow diagram. Input validation verifies the identity of information owners. This measure enhances user protection. The input validation process would be implemented using strings. Code Quality is important because ensuring code quality allows you to control method access based on the user. For instance, a user would have access only to their own information, without the ability to access another user's information or the server. The next area is API’s. Developing an API is essential for both internal and external operations. It would regulate and determine which data access is permissible. Cryptography is probably an extremely important part due to the security measure it implements. Implementing cryptography at Artemis Financial is crucial to ensure user information remains secure, especially given the involvement of different currencies from various regions around the world. The last one is Secure Client/Server. I feel as this is the most important here because even if the program isn’t working as well as you want it, if you take the time to verify the backend and frontend securities you protect a lot. The server is the place for storing all the information dealing with the process and it should be the most secure so no one obtains this information. The frontend is the software side for the client, and making sure no virus, third party software, or hardware can change the program then you increase the likely-hood of not getting malicious content and loosing 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.

Upon reviewing the code and conducting a Vulnerability Assessment, I analyzed both the POM.XML file and the Greeting Controller. In the XML file, I sought to verify the presence of an Apache Validator. In the Greeting Controller, I observed a lack of input validation, which I noted as a feature to address. While the code quality was generally acceptable, I identified a significant issue with the absence of error handling.

Regarding the API, I found several deficiencies. There was a potential breach exposing user input because it was not implemented using a POST method. Finally, I attempted to verify the use of cryptography but found no evidence of its implementation.

**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

| Static Testing Table | | | |
| --- | --- | --- | --- |
| **Dependency:** | **Vulnerability ID’s:** | **Description:** | **Solution:** |
| bcprov-jdk15on-1.46.jar | cpe:2.3:a:bouncycastle:bouncy-castle-crypto-package:1.46:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:bouncycastle:bouncy\_castle\_crypto\_package:1.46:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:bouncycastle:bouncy\_castle\_for\_java:1.46:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:bouncycastle:the\_bouncy\_castle\_crypto\_package\_for\_java:1.46:\*:\*:\*:\*:\*:\*:\*  pkg:maven/org.bouncycastle/bcprov-jdk15on@1.46 | Legion of the Bouncy Castle Legion of the Bouncy Castle Java Cryptography APIs 1.58 up to but not including 1.60 contains a CWE-470: Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection') vulnerabilityin XMSS/XMSS^MT private key deserialization that can result in Deserializing an XMSS/XMSS^MT private key can result in the execution of unexpected code. This attack appear to be exploitable via A handcrafted private key can include references to unexpected classes which will be picked up from the class path for the executing application. This vulnerability appears to have been fixed in 1.60 and later. | Update version of Bouncy Castle Crypto. Ensure that the API is working properly for the downloaded version by running test. Ensure that proper Host validation occurs as it has been noted in the past to cause a mismatch. |
| 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 Hibernate to the newest version. Ensure input validation to ensure that no invalid EL expressions are found and passed. This will increase security. |
| jackson-databind-2.10.2.jar | cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:\*:\*:\*:\*:\*:\*:\* | 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 to current version of software. This will fix issue with security and increase the data integrity. |
| 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. Fixed in Apache Log4j 2.12.3 and 2.13.1. | Upgrade to the newest version available of Apache. Newer version. Support that required features. |
| logback-core-1.2.3.jar | cpe:2.3:a:qos:logback:1.2.3:\*:\*:\*:\*:\*:\*:\* | 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. | Updated to the newer version of the program. New versions have fixed the receiver component denying attackers to the database through back door interpretations of the logs. This fixes the issue backwards processing information to gain access to the system through the logs, securing your software more. |
| snakeyaml-1.25.jar | cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\* | The Alias feature in SnakeYAML before 1.26 allows entity expansion during a load operation, a related issue to CVE-2003-1564. | Migrate to SnakeYAML Engine. It has a configuration option to restrict aliases for collections. The second option is update SnakeYAML but constant monitoring would be required to ensure process is fixed. Going to the engine seems more reliable and cost effective over a long period of time. |
| spring-core-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:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* | 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 the safest and newest version. Verify that Tomcat is not having any vulnerabilities because these two programs work hand and hand. Which may induce friction and more vulnerabilities. |
| tomcat-embed-core-9.0.30.jar | cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* | 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. | Upgraded to the newest version Tomcat solves this problem. |
| tomcat-embed-websocket-9.0.30.jar | cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* | 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. | Upgraded to the newest version Tomcat solves this problem. |

**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.

A quick remedy for the identified security vulnerabilities at Artemis Financial would be to update to the latest versions of Snakeyaml, Hibernate Validator, Apache Tomcat, and BouncyCastle. Then looking at the documentation for the newest versions to see what was fixed. Identified possible issues that were not fixed. Reread the documentation for proper implementations and known solutions to issue identified by the company and other user. Test out the new updates and code to make sure it is up to industry standard. Utilize a long test period with multiple test and hire outside resources to try and penetrate the system. Gather data from this test period and begin to identify new issues. Fix any issues found during test. Then run a light test phase before pushing product to market and allow clients to run reports and give feedback on further issues identified or requirements needed.