# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **5/21/2024** | **Frederick Brehm** |  |

## 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

Frederick Brehm

**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?

Artemis Financial is a company who develops individual financial plans for its clients. The financial plans include savings, retirement, investments, and insurance. The company wants to continue to use its web-based applications to provide great service to their customers and in doing so they would like to modernize their operations. Their motto is “Security is everyone’s responsibility.” When working with individuals in finance it is imperative that the company can provide security for their users in every way possible. This could be from API’s, user input validations, continuous integration, and even DevSecOps to name a few. Utilizing these methods can ensure that security is being implemented and tested throughout the entire process of the company’s modernization.

* Are there any international transactions that the company produces?

As there was no mention of the company only dealing with information that is maintained within the US it is safe to assume that international transactions will take place.

* Are there governmental restrictions on secure communications to consider?

As a company that deals with finances it is safe to assume that there would be governmental restrictions that they would need to consider. The company deals directly with investment stocks, meaning they would need to ensure the safety of trade secrets. They would also need to ensure that they are protected from within their own company. Embezzlement and fraud are common crimes and there would need to be secure communications that prevent this from happening.

* What external threats might be present now and in the immediate future?

There are external threats that will need to be considered in the present and future. The information that is being gathered from clients can include SSN, personal information, bank information, and so on. It is crucial that Artemis Financial prevents hackers from gaining access to this information. This can be accomplished by ensuring that the information being communicated is securely encrypted.

* What modernization requirements must be considered, such as the role of open-source libraries and evolving web application technologies?

There are many options that Artemis Financial can investigate to modernize their web application. This can include utilizing API’s, input validation, regular maintenance, and testing. Continuous integration will be the best option when looking for a method to utilize within the development team and process. This will ensure that the development team is continuously testing the software to ensure proper performance but also to find and fix any bugs and defects that may jeopardize the safety of the company and its clients.

**2. Areas of Security**

Input Validation:

Input validation will ensure the validation of the owner’s information. This will provide a first line of defense for users as they are in control of their validation. This validation would be done in the way of Username/Email and password confirmation. This could also include a 2-factor authentication for extra safety measures.

API’s:

The web application is going to be running internally and externally so the use of an Application Programming Interface would be necessary.

Cryptography:

This is crucial to the security of the web applications as with any type of currency it would have to be questioned if Artemis Financial only uses US dollars or also other foreign currency. Either case, cryptography will help ensure that user information is not going to be compromised while traveling throughout the world.

Client/Server:

I do not think client/server is necessary in this case, but I do believe it would be crucial in my point of preventing any type of crime. The client/server security would keep records of everything going through the servers, which could be beneficial in preventing this. It could also be another defense in the fact that the clients can utilize a type of “firewall” to further the security of the information going to and from servers.

Code Error:

During the implementation and testing phase code error handling is important in weeding out mistakes that are within the API. Through CI developers will be able to catch mistakes and fix them in real time to help keep the clients safe from interception of their information.

Code Quality:

Code quality will fall in the same category as code error but will focus more on the readability and dependency of the code. This would include the development team using code error to improve the quality of the code, ensuring that there are no mistakes or defects that have been missed. Code quality can also help in any future improvements to the code as developers should follow good practice by including comments, creating a clean and understandable code for any future developers who work on the product.

**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 using the VAPFD and reviewing the code there are a few opportunities off the back where I could see issues in security. The Pom.XML file, it should be notated the versions Apache 4.0, spring.framework.boot version 2.2.4, rest-service version 0.0.1, java version 1.8, bouncycastle version 1.46, and dependency check maven version 5.3.0. It is necessary to check and ensure these versions are up to date. In the greetingController file, I noticed that there was no effective code for input validation from the user. The API seems like it is acceptable from first look, but I believe user validation could be jeopardized as I did not see a POST method. I did not see any code written for any type of cryptography which would jeopardize the security of the transfer of communication. The code quality is lacking, throughout some of the code there was no readability embedded in the code, so a future developer working on this product may not know what the purpose of the code is. There were some instances of comments but again, it was not specific in summarizing what the purpose of that portion of code was. In DocData.java, it seems as if there is an issue with the id variable showing there needs to be improved error handling.

From the manual review I see vulnerabilities in the code and files that would be a concern to the overall security of the product. However, these findings do seem to be miniscule errors that would be fixed through continuous integration and delivery.

**4. Static Testing**

Run a dependency check on Artemis Financials’ 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

1.

CVE -2024-34447

Highest Severity: High

An issue was discovered in Bouncy Castle Java Cryptography APIs before BC 1.78. When endpoint identification is enabled in the BCJSSE and an SSL socket is created without an explicit hostname (as happens with HttpsURLConnection), hostname verification could be performed against a DNS-resolved IP address in some situations, opening a possibility of DNS poisoning.

Analysis: Update to the newest version of Bouncy Castle Java Cryptography API’s

2.

CVE-2020-10693

Highest Severity: Medium

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.

Analysis: Update to newest version of Hibernate Validator

3.

CVE-2020-25649

Highest Severity: High

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.

Analysis: Update to latest version of FasterXML Jackson Databind

4.

CVE-2020-9488

Highest Severity: Low

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

Analysis: Update to latest version

5.

CVE-2023-6378

Highest Severity: High

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.

Analysis: Update to latest version

6.

CVE-2022-1471

Highest Severity: Critical

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.

Analysis: Update to latest version

7.

CVE-2023-20873

Highest Severity: Critical

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+.

Analysis: Update to latest version

8.

CVE-2023-20873

Highest Severity: Critical

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+.

Analysis: Update to latest version

9.

CVE-2022-22965

Highest Severity: Critical

POST requests to /web/mvc in GX Software XperienCentral version 10.36.0 and earlier were not blocked for uses that are not logged in. If an unauthorized user can bypass other security filters, they are able to post unauthorized data to the server because of CVE-2022-22965.

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.

Analysis: Update to latest version

10.

CVE-2016-1000027

Highest Severity: Critical

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.

Analysis: Update to latest version

11.

CVE-2022-22965

Highest Severity: Critical

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.

Analysis: Update to latest version

12.

CVE-2020-1938

Highest Severity: Critical

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. Several 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.

Analysis: Update to latest version and make configurations

13.

CVE-2020-1938

Highest Severity: Critical

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. Several 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.

Analysis: Update to latest version and make 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 Financials’ software application.

Throughout my manual review I noticed issues with the current pom.xml file and some of the .java source files when looking over. Some of the findings that were noticed were vulnerabilities due to API’s, spring book framework, Java, maven, etc. versions that were not up to date. There as also A notice of there not being a post method through the API which could cause harm due to their being a lack of input validation.

After looking at the static tests you can see that my manual review and static test were accurate when referencing each. Most of the vulnerable dependencies are easily fixed by updating to the current version of the software. The other issue seemed to be that there were a few vulnerabilities where the lates version was being utilized but there were specific configurations that had to be made in order for them to work properly. Last, there was an issue with the #9 dependency CVE-2022-22965 where you can see that the post was allowing access to any user. When summarizing the dependencies of this test, these are simple issues that may have been overlooked and could be fixed in miniscule as most have to do with using an updated version.