# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **22 May, 2024** | **Kiersten Grove** | **Initial Draft** |

## 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

Kiersten Grove

**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 provides various financial services to its customers. This means the information they’ll be handling is sensitive and requires high levels of security. Financial companies are at risk of being targeted maliciously to obtain sensitive information illegally or otherwise for the purposes of fraud. Artemis may especially be a target of potential security threats as they do perform transactions and communications internationally, which will require high levels of security as well. Performing international transactions will likely come with different governmental restrictions depending on which countries they are transacting and communicating with. This could include utilizing the proper exchange rates, ensuring the coverage of fees, consideration of time zones, fraud prevention, and other services. Considering that fraud is the most likely issue that Artemis will have to defend against the use of HTTPS and two factor authentication for all account holders/users would be worth considering when requiring the use of 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.

First and foremost, code quality will be of utmost importance. By creating a fully functioning application that meets the needs of Artemis’ users and customers security will be easy to maintain. To ensure code quality frequent testing will be required along with updates and the ability to alter the application as the company’s needs change. Input validation is also of high importance. The applications that will be in use by Artemis and its customers will need to have the ability to verify that the given inputs are valid. Input validation would be important to pair with two factor authentication as well as some system in which credential updates are required after so many failed attempts or specified time i.e. update password every 90 days. Input validation is also important in ensuring the functionality of APIs. Since APIs allow for the transfer of data between software and applications, input validation will be necessary in ensuring only those who are allowed access have access and at the appropriate level depending what type of user they are i.e. a customer may have lower levels of access than a financier. Exception handling needs to also be considered. This can relate to input validation by producing visual errors when the wrong credentials are used, or a user may be attempting to access something they aren’t supposed to. Lastly, encapsulation is essential in OOP and will be for this application as well. Encapsulation allows for the bundling of related data which is important when handling the finances of others. The correct financial information will need to be associated with the correct customer, financial institutions, geographical location etc.

**3. Manual Review**

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

In inspecting the code base, the primary issue I’ve found is the lack of input validation. In the DocData.java file there is no method of input validation to ensure the user accessing the data has entered the correct username and password to do so. This could lead to security threats in which someone may utilize this vulnerability to access the data system and obtain data they don’t have the proper credentials to access.

**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

1. bcprov-jdk15on-1.46.jar: Doesn’t consider proper timing side-channel attacks which can lead to remote attackers conducting attacks via statistical analysis of timing data. This can lead to attackers obtaining sensitive information. The recommended solution would be to update Bouncy Castle to its latest version.
2. spring-boot-2.2.4.RELEASE.jar: Vulnerable to temporary directory hijacking, this only affects products or versions that are no longer supported, the best solution would be to ensure that the product/version being used is up to date and supported.
3. logback-core-1.2.3.jar: In version 1.2.7 and older versions an attacker with the required privileges could craft a malicious configuration allowing arbitrary code to execute. They could also mount a DOS attack by sending poisoned data. The best solution would be to update to the latest version of logback.
4. log4j-api-2.12.1.jar: In Apache Log4j SMTP there was improper validation of certificate with the host allowing a man-in-the-middle attack which could leak log messages sent through that appender. This was fixed in versions 2.12.3 and 2.12.1, it would be best to update to one of these versions.
5. snakeyaml-1.25.jar: Before version 1.26 allows entity expansion during a load operation. Due to unbound alias chasing a maliciously crafted file can cause the system to consume significant resources, parsing user input could lead to a DOS vector. It is recommended upgrading to version 2.0 or higher.
6. jackson-databind-2.10.2.jar: In FasterXML there was a flaw where it didn’t have entity expansion secured properly which allowed for external attacks risking data integrity. It seems most versions have varying vulnerabilities that lead to data integrity, the best option would be to upgrade to a version beyond 2.15.2.
7. tomcat-embed-core-9.0.30.jar: For versions 9.0.28 through 9.0.30, 8.5.48 through 8.5.50, an 7.0.98 to 7.0.99 there was a regression that resulted in invalid transfer-encoding headers being incorrectly processed. This leads to the possibility of HTTP request smuggling. In other versions specially crafted sequences of HTTP/2 requests could trigger high CPU usage. There are also possibilities of endless loops, denial of service, seeing responses for unexpected resources, information leaks, and the return of arbitrary files. It is recommended that users upgrading to 9.0.31, 8.5.51 or 7.0.100 will need to make small changes to their code configurations.
8. hibernate-validator-6.0.18.Final.jar: A bug in the message processor allows for the validation of invalid EL expressions which could lead to attackers having the ability to bypass input sanitation.
9. spring-web-5.2.3.RELEASE.jar: Up through version 5.3.16 there is a potential for remote code execution. This may or may not occur depending on how the library is implemented within a product and authentication may be required.
10. spring-beans-5.2.3.RELEASE.jar: Running on JDK 9+ may be vulnerable to remote code execution via data binding. Running the application as a Spring Boot executable jar mitigates this vulnerability however the nature of the vulnerability is more general and there could be other ways to exploit it.
11. spring-webmvc-5.2.3.RELEASE.jar: It’s possible for a user to provide malicious input to cause the insertion of additional log entries.
12. spring-context-5.2.3.RELEASE.jar: Patterns for disallowedFields are case sensitive which means a field is not effectively protected unless it’s listed with upper- and lower-case letters for the first character of the field, this includes all nested fields.
13. spring-expression-5.2.3.RELEASE.jar: In older and unsupported versions it is possible for a user to provide specially crafter SpEL expressions that can cause a DoS condition. It would be best to upgrade to the latest supported version.

**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.

The majority of the dependencies had vulnerabilities due to outdated or unsupported plug-ins. The first step would be to upgrade all plug-ins that need one as most security risks would start with the targeting of these outdated plug-ins. After reviewing the dependency check it can be concluded that the Springframe boot is an outdated version and will need to be updated. Implementing the use of RESTful APIs and HTTPS communication methods will further improve the protection over the data as it is transferred between clients. Lastly, implementing better input validation methods will also prove beneficial as they are currently lacking.