# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **1/24/2024** | **Caleb McManus** |  |

## 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

[Insert your name here.]

**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?

The client is Artemis Financial, who are a financial planning firm that many use for savings, retirement, investments, and insurance. They are looking to update their systems of operation to improve their efficiency. They need custom software with effective security due to the sensitivity of their data. The software is not going to be used for a global market but will still need outside protection as it could easily become a target. There must be a high level of encryption due to the passing of data such as social security numbers, financial papers, names, and license. These need to be none visibly passed and need to be secured. As far as legal and government regulations are concerned, that will be up to the local and federal law of the country/ state they operate in, in other words, not enough information is not given to say and a lawyer or someone more specialized should clarify. Although I imagine it’ll have to be strictly secure as earlier mentioned that the company will be handling many government documentations for many clients. As such, the company will need to have constant security specialist help develop the software every update to check for issues, along with keeping it up to date and monitoring the database in live service to prevent a leak of data. Open-source libraries can be a security risk if not handled correctly due to them being publicly available, which in turn leaves attacks direct access to figuring out how a system may work and be able to get around all the counters to its risks.

**2. Areas of Security**

Refer to the vulnerability assessment process flow diagram. Identify which areas of security apply to Artemis Financials’ software application. Justify your reasoning for why each area is relevant to the software application.

* Input Validation- This is needed to avoid any string variable exploits within the software application from the user.
* API- API is important as the system operates internally and externally and will need a secure way to interact with the database securely.
* Cryptography- As mentioned in question 1, there is going to be a threat of attackers wanting to get a hold of people’s information that could be used to steal their identity; therefore, the data is going to need to be encrypted from outside threats.
* Code Error- Making sure that potential issues with the code, discovered in extensive testing, should be able to handle exploitive ways or errors properly.
* Code Quality- This is important, yet different from code errors. This one focus more on the way the code is structured to prevent issues and breaches rather then the handling of errors.

**3. Manual Review**

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

I do not know if it is intentional when I opened the code base, I had GreetingsController, CrudController, RestServiceApplication, and the Pom.xml file all show errors, so far, an error in code quality. Looking in the “Greeting” files, I noticed a lack of and validation process nor a way to prevent overflow and unauthorized strings to be sent through, which is a security breach. I also noticed that in the “myDateTime” there are uncomplete class calls which for a final product, is unacceptable, even if the time and date is not necessarily attached to the goal of the financial service. In the “CRUD” files, there are similar issues of un-validated strings being passed. Same with the “customer” file, not only that, but there is also not encryption nor private access to this personal data, it allows you to just access the information if you call the class. Overall, there are many issues in validations, but also there is a issue with absolutely no error handling in none of the files and therefore can cause issues for users along with allowing attackers to get a understanding of the file system and breach.

**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

Here is the screen shoot of the vulnerabilities provided by the tool:

A screenshot of a computer

Description automatically generated

Going by the list:

* [bcprov-jdk15on-1.46.jar](file:///C:\Users\CalebDev.System\eclipse-workspace\rest-service\target\dependency-check-report.html#l1_991c96a4e31e6c19e2b9136c8955bd423f2dc4c7): This issue is a package implementation of the Cryptography Api. This can be solved by updating the .jar to the latest version of 1.60.
* [hibernate-validator-6.0.18.Final.jar](file:///C:\Users\CalebDev.System\eclipse-workspace\rest-service\target\dependency-check-report.html#l3_7fd00bcd87e14b6ba66279282ef15efa30dd2492): There seems to be a issue where the .jar will mis validate invalid EL expressions. This can be remedied by upgrading to 6.0.20.
* [jackson-databind-2.10.2.jar](file:///C:\Users\CalebDev.System\eclipse-workspace\rest-service\target\dependency-check-report.html#l5_0528de95f198afafbcfb0c09d2e43b6e0ea663ec): It was discovered that in the Databind, the data was not secured properly leaving it open to XML external attacks. Simply update the version to patch it out.
* [log4j-api-2.12.1.jar](file:///C:\Users\CalebDev.System\eclipse-workspace\rest-service\target\dependency-check-report.html#l10_a55e6d987f50a515c9260b0451b4fa217dc539cb): Unsecure and mismatch of certification can lead to being intercepted log messages. You have the option to upgrade to the current version, or downgrade to a previous version as this issue is only in 2.12..
* Logback: Both logback are tied together as they are both susceptible to the remote execution of code though someone who has access to the logs (see log4 issue) and is unsecure. These are fixed by upgrading both log4 and logback to latest version and limiting the JNDI data source names.
* [snakeyaml-1.25.jar](file:///C:\Users\CalebDev.System\eclipse-workspace\rest-service\target\dependency-check-report.html#l15_8b6e01ef661d8378ae6dd7b511a7f2a33fae1421): There is an issue where the Constructor class() does not properly restrict the types which are instantiated during the deserialization process. Easy fix is to update to at least version 2.0.
* The Spring framework is filled with many vulnerabilities and should be honestly kept up to date to the most up-to-date version to keep many of the issues from being exploited. Another solution is to migrate to another framework.
* [tomcat-embed-core-9.0.30.jar](file:///C:\Users\CalebDev.System\eclipse-workspace\rest-service\target\dependency-check-report.html#l22_ad32909314fe2ba02cec036434c0addd19bcc580): There is a known issue with tomcat core not properly parsing the HTTP transfer encoding leaving it vulnerable to HTTP attacks and hijacks. Simply update to the current version to patch this out.
* [tomcat-embed-websocket-9.0.30.jar](file:///C:\Users\CalebDev.System\eclipse-workspace\rest-service\target\dependency-check-report.html#l24_33157f6bc5bfd03380ebb5ac476db0600a04168d): Same as with the core.

**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.

I had caught many issues visually that were also reiterated by the vulnerability tool, which is that there is a distinct lack of error handling and validation within the code. If we were to add both of those into the code, the data would be better off. As well as needing to make sure cryptography is added to the data when the data is passed to ensure that the system is secure from possible prying eyes. Another thing is to make sure all dependencies are up to date to avoid common security issues that are found for the older version that are currently in use. After all that is done, need to do further testing on the updated security, possibly hire a legal team of hackers to try to infiltrate the system and report back to further develop the plan of defense.