# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **07/21/24** | **Jonathan C. Sanchez** | **Vulnerability Assessment** |

## 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

Jonathan C. Sanchez

**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?

**Secure Communications:**

Artemis Financial will handle customer and internal data daily. Therefore, secure communication between systems and processes is vital to the company and its customers. By using REST APIs, we can take advantage of the HTTPS protocols the APIs use to communicate. We must also secure communication by implementing strict authentication and authorization for users.

**International Transactions:**

Based on the information provided, I cannot identify whether or not the company produces international transactions. If the company produces international transactions, we will need to become familiar with international laws and regulations. For example, the General Data Protection Regulation (GDPR) is a European regulation that requires companies to follow numerous requirements when they process user data.

**Governmental Restrictions:**

The United States does not have very stringent regulations on the use and security of user data. The Gramm Leach Bliley Act (GBLA), does not ensure the protection of user data. However, the GBLA does require financial services to disclose how their data is used, and a user can then opt out. I mentioned a stricter regulation above, but this only applies to companies with European customers.

**External Threats:**

External Threats that may present now, and/or in the immediate future can vary based on third-party dependencies, and implementation. External threats that may be present now could be phishing attacks, data loss, and/or malware. Future threats could come up due to vulnerabilities in third-party dependencies and as technology becomes more available new “hacking” techniques may become used.

**Modernization Requirements:**

Modernization requirements are important to ensure the software can compete and still support the customers’ use case in the future. It is hard to predict how technology will change three years from now. However, we can plan for a year from now. Ensuring that the code has modularity and is easily maintainable is a way to handle future innovations. When using open-source/third-party libraries, we have to consider the need for regular updates to implementation and check for vulnerabilities.

**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.

It is hard to say that a certain area(s) of the vulnerability assessment applies to Artemis Financial. A single vulnerability across the system can cause an array of issues. If the entire system is not secure, what is the point in ensuring a single portion of the system is? Therefore, I think all seven areas are good to focus on. However, cryptography and encapsulation can ensure that if an attack occurs it is not successful in harvesting data.

**Input Validation:** Implementation and testing of input validation can prevent buffer overflows, SQL injection, and cross-site scripting vulnerabilities.

**APIs:** Proper functionality and security in APIs can protect data from being accessed by non-authorized users and allow for robust and efficient communication.

**Cryptography:** Cryptography can protect sensitive data from being accessed by users without authorities. It can also prevent data leaks and decryption attempts.

**Client/Server:** A client/server model provides centralization, which promotes greater security, better functionality, and easier maintenance. We can also use APIs that use HTTPS to ensure data is protected during transmission.

**Code Error:** Code Errors, or bugs, can lead to vulnerabilities such as null pointers, buffer overflows, and flaws in the implementation of requirements.

**Code Quality:** Good code quality reduces the chances of releasing vulnerable software. We can reduce these chances by testing and participating in code reviews.

**Encapsulation:** Encapsulation ensures that certain components and objects are only accessible by authorized classes. Doing so can prevent unauthorized access and separate functionality.

**3. Manual Review**

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

**CrudController.java:**

* The RequestParam takes input from the user that can lead to a malicious injection command.

**Customer.java**

* The customer class does not require authentication to access a user’s account\_number through the showInfo() method.
* The deposit(int a) method accepts any integer ‘a’. This will allow a user to deposit any positive or negative integer.

**DocData.java**

* The database credentials for the database are hardcoded in the code.
* Code quality. The code is not complete.
* If the read\_document method takes the String key, and String value parameters and uses it to query data from the database. Then this would allow an attacker to use SQL injection.

**myDateTime.java**

* Code quality, the code is not complete.
* The variables should be private. To ensure they are not accessible to outside classes.
* I am unsure, if I am missing something, but line 11 returns an integer with a value of 3. That does not look correct.

**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

The dependency check returned 135 vulnerabilities from 13 dependencies. I also ran a Red Hat Dependency Check that returned 55 unique vulnerabilities from two dependencies.

**A screenshot of a computer

Description automatically generatedA diagram of a number of people

Description automatically generated with medium confidence**

**5 Critical Vulnerabilities:**

|  |  |  |  |
| --- | --- | --- | --- |
| Dependency | CVE Count | Severity | Mitigation |
| tomcat-embed-websocket-9.0.30.jar | 27 | CRITICAL | Upgrade to a more recent version |
| tomcat-embed-core-9.0.30.jar | 26 | CRITICAL | Upgrade to a more recent version |
| spring-web-5.2.3.RELEASE.jar | 14 | CRITICAL | Upgrade to a more recent version |
| spring-webmvc-5.2.3.RELEASE.jar | 11 | CRITICAL | Upgrade to a more recent version |
| spring-core-5.2.3.RELEASE.jar | 11 | CRITICAL | Upgrade to a more recent 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 manual review and static testing report resulted in finding approximately nine code issues/vulnerabilities and approximately 135 vulnerabilities. The code in the project lacks completeness, encapsulation, validation, and security. To increase the overall security of the application the company should implement strict authorization and authentication protocols. I would recommend that input validation is added where there is any type of user input. In addition to implementing input validation, the company should set instance variables to private, so they cannot be accessed outside of expected classes. The company should also refrain from hardcoding credentials into the database parameters. My recommendation is to store the credentials separately in a config file. Lastly, the company should update all of the third-party dependencies and libraries to fix the majority of the vulnerabilities found during the dependency scan. The current state of the application is unsafe for the company and its users.