# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **3/30/2025** | **Brad Peterson** |  |

## 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

Brad Peterson

**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 is a financial service provider, so security is one of their foremost concerns. Trust is one of the most important aspects for any company handling someone else’s money so before they work on any of the business problems they are facing they need to ensure their platform won’t have money stolen or information leaked. A failure in this aspect will crumble their business since at best customers will lose trust in them and pull their funds out and, at worst, they could have all the money entrusted to them stolen and lost to unknown people across the globe.

Modern finances have very global ties so international transactions are a necessary challenge to allow Artemis to grow to their full potential. But this brings new challenges as working across borders means working with different sets of rules and currencies. For instance, if they want to expand to the EU then they must comply with their privacy laws that tend to be more strict than those found in the USA. This could also mean different requirements on how you store and protect that data as well as differing accessibility to that data for individuals to pull.

Another aspect to keep in mind is encryption standards. As the world continues to add more powerful technology new standards for minimum levels of encryption are raising to stay ahead. Today, NIST recommends using 2048 bit-keys for encryption and even that won’t last forever. There are actual laws requiring the use of encryption for financial institutions like Artemis so it’s absolutely a baseline level requirement for them to implement.

In addition to encryption, one of the best safety measures is training personnel how to avoid phishing attacks. This can come in the form of trying to get someone inside the network to download malware through fake links and attachments. Or by trying to fake their identity and tricking someone into giving them access to information they don’t have rights to. These attacks are very common and serious as they pose a large risk for a breach that attackers can use to steal money with.

One last topic to consider is the use of open-source libraries. They can be incredibly powerful and useful but fundamentally they will always pose something of a risk as you can’t control who is updating them and what that might mean. It’s certainly important to keep all dependencies updated to stay current. But in the case of open-source more care must be taken to review what is potentially being installed into the code base as it could be malicious.

**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**

This is baseline functionality in secure programming that if you have inputs to your system they should be cleaned and validated before being fed into the program to protect against injection attacks.

**Cryptograpy**

Not only is it vitally important to secure the databases holding all the customers’ information as it’s very sensitive and can lead to the loss of actual funds if compromised, but it’s also legally required for financial and medical records.

**Client/Server**

Communications between client and server are vulnerable to man-in-the-middle type attacks so keeping these information packets secure is another avenue to protect from malicious attackers from.

**Encapsulation**

The controlled access of variables is important to prevent misuse of information and keep it contained only to where it is needed within the code base.

**Code Error/Code Quality**

As with any application following best practices and cleaning up errors that may cause vulnerabilities is important to produce secure code.

**3. Manual Review**

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

* Customer class has a public variable to hold *account\_balance*
* User inputs are not being validated for security
* GreetingController class is not making use of get/set functions for its private data
* CRUD class has public get/set functions
* Customer class *showInfo* method is essentially a get function that is public
* Greeting class uses public get/set functions
* No evidence of encryption

**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
* **bcprov-jdk15on-1.46.jar**
  + cpe:2.3:a:bouncycastle:bouncy-castle-crypto-package:1.46
  + Bouncy Castle is used for encryption so using the most up to date version is vital to keep up with current standards as well as avoiding bugs
* **spring-boot-2.2.4.RELEASE.jar**
  + cpe:2.3:a:vmware:spring\_boot:2.2.4:release
  + Spring boot is an open source framework for Java and the Spring platform
* **hibernate-validator-6.0.18.Final.jar**
  + cpe:2.3:a:redhat:hibernate\_validator:6.0.18
  + This tool is used for input validation and should be kept up to date to account for any new bugs or discoveries in the tool that would render it ineffective in doing it’s job

**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.

To fix the errors found during the dependency check the solution seems to always be upgrade that dependency to the most current version. This will protect against bugs in those packages that may introduce a vulnerability.

For all the problems found during manual review, taking a moment to do code reviews specifically with the purpose of resolving these should be done.

Finally, there is much functionality yet to add and part of that process should be to build out the encryption and input validation capabilities.