# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **5-26-2024** | **Aaron Austin** |  |

## 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

Aaron Austin

**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?

My client Artemis Financial works with their customers' money so there is a natural level of trust and security needed in their software solutions. This fact alone puts an immense amount of importance on the clients need for secure communications. Artemis Financial does not specifically mention that they work with international clients, so it is safe to assume currently that international transactions are not needed. My client does not specifically mention that they work with governmental entities. Some of the threats that my client is vulnerable to are injection attacks, phishing attacks, and zero-day exploits where a hacker or scammer could try and exploit bugs before they could be discovered and/or patched. Since we are dealing with sensitive user data, I believe it is important to limit outside dependencies and ensure the ones we are using are secure. We can rely on authentication libraries for verifying identities before granting access to the software. We can also rely on cloud services such as AWS to ensure our data is not stored in easy to steal locations.

**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.

Artemis Financial has a couple areas where vulnerabilities could be more inclined to occur. The first is Input Validations because if the client allows for any kind of user input into their program it should be secured and checked for validity which is easy to sometimes not do. I will also tie Encryption Use and Vulnerabilities into the first one because the customers info should be validated coming in and going out for maximum security. Secure Error Handling is important because errors could be fatal for the client’s software dealing with finances so it's important to have exception handling defined in the program. Secure Coding Practices/Patterns and Secure Data Structures go hand and hand because each deal with encapsulating sensitive data in secure wrappers to protect that data which is relevant for this kind of software.

**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 reviewing the code for my client's application, I noticed several potential security issues within the code. Starting with the CRUD.java there are data validation issues present. In CRUDController.java there is a potential injection vulnerability present, specifically in the “readDocument” method. The inputs should be validated before being passed through. In customer.java the customer information is not properly secured which could lead to exposure of sensitive data within the “showInfo” method. In DocData.java the inputs are not being properly validated which could lead to injection attacks. The Greeting.java and GreetingController.java files also have validation issues. There are methods in the myDateTime.java file that are not properly implemented. The biggest issues involve data validation and authentication which are of utmost importance because they allow not only for data leakage but also unauthorized access and manipulation of the software.

**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

**Name**: Spring Framework JDK 9+ Remote Code Execution Vulnerability (CVE-2022-22965)  
**Description**: Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding.   
**Fix**: Apply updates per vendor instructions.

**Name**: Apache Tomcat Improper Privilege Management Vulnerability (CVE-2020-1938)  
**Description**: Apache Tomcat treats Apache JServ Protocol (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.  
**Fix**: Apply updates per vendor instructions.

**Name**: HTTP/2 Rapid Reset Attack Vulnerability (CVE-2023-44487)  
**Description**: HTTP/2 contains a rapid reset vulnerability that allows for a distributed denial-of-service attack (DDoS).  
**Fix**: This vulnerability affects a common open-source component, third-party library, or a protocol used by different products. Please check with specific vendors for information on patching status. For more information, please see: <https://blog.cloudflare.com/technical-breakdown-http2-rapid-reset-ddos-attack/>

**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.

I found three vulnerabilities that need to be addressed within the vulnerability check. The first two involve simple dependency updates that should patch the vulnerabilities per the dependency check report. The third vulnerability is not as simple to fix on the surface. It creates a risk of DDOS attacks which appear to be a favorite among attackers. The best course of action per the link in the fix is to set up a DDOS mitigation service while vendors address the underlying issue within HTTP/2 to create a permanent fix.

Work Cited

*Owasp dependency-check. OWASP Dependency-Check | OWASP Foundation. (n.d.-a). https://owasp.org/www-project-dependency-check/* [*file:///Users/aaronaustin/Downloads/rest-service/target/dependency-check-report.html#l22\_33157f6bc5bfd03380ebb5ac476db0600a04168d*](file:///Users/aaronaustin/Downloads/rest-service/target/dependency-check-report.html#l22_33157f6bc5bfd03380ebb5ac476db0600a04168d)

*Desgats, L. P., Pardue, L., Desgats, J., Yoachimik, O., Pacheco, J., & Tomé, J. (2024, February 6). HTTP/2 rapid reset: Deconstructing the record-breaking attack. The Cloudflare Blog. https://blog.cloudflare.com/technical-breakdown-http2-rapid-reset-ddos-attack*