# CS 305 Project One

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **September 22, 2024** | **Robert Eigenfeld** |  |

## 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

Robert Eigenfeld

**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 recognizes the importance of secure communications as they modernize their operations and improve software security. This helps build trust with customers who rely on sensitive financial data. They should consider potential international transactions and any governmental restrictions on data protection. The company also needs to prepare for external threats like cyberattacks and risks from third-party integrations. In their modernization efforts, they should focus on securely using open-source libraries and adopting new web technologies to enhance security and maintain a good user experience.

**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.

1. Authentication & Authorization
2. Input Validation
3. Data Protection
4. Secure Communication

**3. Manual Review**

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

|  |  |  |
| --- | --- | --- |
| **File Name** | **Line #** | **Reason** |
| CRUD.java | 5 | Sensitive Data in fields content & content2. These fields being exposed could lead to access to sensitive information. |
| CRUD.java | 9 | There is no input validation for both content1 & content2. This could cause major issues if the either fields contained malicious data or invalid values. |
| CRUDController.java | 6 | Like line 9 in CRUD.java the name parameter is taken without any validation. This could lead to issues such as injection attacks. |
| CRUDController.java | 9 | Just as line 5 in CRUD.java there is also possible sensitive data in DocData. |
| customer.java | 11 | Lack of input validation for the deposit method as it directly adds the value to the account balance without validation. |
| DocData.java | 21 | In the read document method, there is no input validation making it at risk for SQL Injection. |
| Greeting.java | 7 | No input validation for the id and content parameters. |

**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

|  |  |  |
| --- | --- | --- |
| **#** | **Finding** | **RISK** |
| **CVE-2020-1938** | Serious vulnerability in Apache Tomcat that lets attackers misuse the Apache JServ Protocol (AJP). It treats AJP connections as more trusted than regular HTTP connections, allowing attackers to access files and run malicious code. | **CRITICAL** |
| **CVE-2022-1471** | Vulnerability in the Apache HTTP Server that could allow an attacker to bypass authentication restrictions. This occurs when the server is configured to use specific authentication methods, and an attacker can manipulate certain request headers to gain unauthorized access to protected resources. | **CRITICAL** |
| **CVE-2016-1000027** | Allows potential remote code execution (RCE) if an attacker accesses the API. This vulnerability arises from improper validation of user input in certain API requests. | **CRITICAL** |
| **CVE-2022-22965** | Vulnerability in the Spring Framework that lets attackers execute remote code on Java applications. It affects Spring apps running on JDK 9 or higher due to improper input handling. | **CRITICAL** |
| **CVE-2022-1471** | Vulnerability in the Apache HTTP Server that allows an attacker to bypass authentication controls. This can occur if the server is configured with certain authentication methods that do not properly validate user input. | **CRITICAL** |
| **CVE-2024-3447** | Vulnerability in the Spring Framework that lets attackers execute remote code if certain applications are misconfigured. This can happen when using specific features in vulnerable versions of the framework. | **HIGH** |

**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 enhance security in Artemis Financials’ software application, several key steps should be taken. First, sensitive data in CRUD.java should be encrypted, and access should be restricted based on user roles. Next, strict input validation must be implemented across various files to prevent injection attacks, with validation checks included in state-modifying methods like deposit. It is also crucial to update outdated libraries and frameworks to address critical vulnerabilities and review configuration settings to align with security best practices. Additionally, conducting regular security audits and providing developer training on secure coding practices will further strengthen security. Finally, deploying a Web Application Firewall (WAF) will help protect against common attacks. By following these measures, Artemis Financial can significantly improve the security posture of its application.