# CS 305 Project One

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **11/16/2024** | **Aqbah Butt** |  |

## 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

Aqbah Butt

**1. Interpreting Client Needs**

Artemis Financial is aware of how critical it I to continue operating in a modern, competitive manner in the quick-paced financial world of today. The use of specialized software to address their unique requirements forms the basis of their approach. To safeguard their sensitive information and business processes, companies are aware of the pressing need for robust software security measures. An essential part of its technological infrastructure is a REST-based online application programming interface(API), and Artemis Financial is eager to use Global Rain’s cybersecurity experience to improve the protection of its clients’ private financial data from external attacks. By collaborating with Global rain, Artemis Financial hopes to improve customer and stakeholder confidence by guaranteeing the availability, confidentiality, and integrity of its systems.

**2. Areas of Security**

I identified several areas which required security:

* APIs: Making sure RESTful APIs are safe and only grant access to authorized databases.
* Input Validation: They need to make sure that the right data types are used to minimize the risks of vulnerabilities. This would increase the chances of Artemis Financials’ software working as intended.
* Code Quality: Software will perform at its best when the best and latest updates are used, together with the best coding practices. The likelihood of potential vulnerabilities decreases with the quality of the software.
* Encryption: Using encryption would decrease the chances of clients’ information being compromised.
* Errors in Code: As a means of protecting against potential vulnerabilities, code quality and minimizing errors work hand in hand.

**3. Manual Review**

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

The lack of necessary restrictions for DocData string IDs and Greeting IDs to guarantee that only valid ID are submitted raises serious concerns about input validation. Security is further compromised by the system’s complete lack of an authentication service and the absence of input verification for authentication.

The system’s protection against data breaches is further weakened by the lack of HTTP for secure internet communication. Furthermore, sensitive data is exposed because there is no cryptography in place for account management. These vulnerabilities are made worse by the inability to recognize encryption services, APIs, and coding errors. To guarantee the security, dependability, and general integrity of the system, these problems must be resolved.

**4. Static Testing**

A screenshot of a computer

Description automatically generated

Stated below are some of the vulnerabilities found when I run the maven dependency check along with fixes:

* CVE-2023-33202: Bouncy Castle for Java before 1.73 contains a potential Denial of Service (DoS) issue within the Bouncy Castle org.bouncycastle.openssl.PEMParser class. This class parses OpenSSL PEM encoded streams containing X.509 certificates, PKCS8 encoded keys, and PKCS7 objects. Parsing a file that has crafted ASN.1 data through the PEMParser causes an OutOfMemoryError, which can enable a denial-of-service attack. (For users of the FIPS Java API: BC-FJA 1.0.2.3 and earlier are affected; BC-FJA 1.0.2.4 is fixed.)
* CVE-2020-10693: A flaw was found in Hibernate Validator version 6.1.2.Final. A bug in the message interpolation processor enables invalid EL expressions to be evaluated as if they were valid. This flaw allows attackers to bypass input sanitation (escaping, stripping) controls that developers may have put in place when handling user-controlled data in error messages.
* CVE-2023-35116: jackson-databind through 2.15.2 allows attackers to cause a denial of service or other unspecified impact via a crafted object that uses cyclic dependencies. NOTE: the vendor's perspective is that this is not a valid vulnerability report, because the steps of constructing a cyclic data structure and trying to serialize it cannot be achieved by an external attacker.
* CVE-2021-44832: Apache Log4j2 versions 2.0-beta7 through 2.17.0 (excluding security fix releases 2.3.2 and 2.12.4) are vulnerable to a remote code execution (RCE) attack when a configuration uses a JDBC Appender with a JNDI LDAP data source URI when an attacker has control of the target LDAP server. This issue is fixed by limiting JNDI data source names to the java protocol in Log4j2 versions 2.17.1, 2.12.4, and 2.3.2.

**5. Mitigation Plan**

The analysis of static testing and manual inspection revealed several crucial areas that needed to be improved to increase system operation and security. First, stricter string restrictions must be added to increase input validation. To close authentication gaps, an authenticator service with appropriate access controls should be developed. To further protect data transfer, it is essential to use encryption services, integrate a functional API, and switch to HTTPS protocols. To make sure the system operates as intended in a variety of scenarios, manual testing should also be expanded.

Every dependency should be updated to the most recent version from the standpoint of static testing. A Maven check needs to be done after updating to find any lingering vulnerabilities. If problems continue, they should be examined and, if they are judged to be non-threatening, suppressed. Together, these actions seek to improve the system’s performance, security and dependability.