GLOBAL RAIN

Artemis Financial Vulnerability Assessment Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	**10/31/2023**	**Joseph Marek**	**This document represents the initial version and serves as an evaluation of potential threats to Artemis Financial. Additionally, it outlines a proactive mitigation plan designed to safeguard against the exploitation of vulnerabilities. The document encompasses comprehensive elements, such as a thorough code review and a dependency assessment, to ensure a holistic approach to security.**

Client

Instructions

Submit this completed vulnerability assessment report. Replace the bracketed text with the relevant information. In the report, identify your findings of security vulnerabilities and provide recommendations for 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 choose to include images or supporting materials. If you include them, make certain to insert them in all 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
Joseph Marek

1. Interpreting Client Needs

Secure Communication –

Securing communication plays a vital role, particularly in software applications dealing with sensitive information. However, the importance of maintaining communication integrity extends beyond personal data security. Take Artemis Financial, a company specializing in customized financial and insurance plans for individual clients. This process involves exchanging information between the company and its clients, who entrust sensitive details like income and investments. Consequently, it's crucial for the company to establish robust data security measures to safeguard this valuable information from unauthorized access.

International Transactions –

Following a thorough assessment of the provided documents and the current codebase of Artemis Financial, there is no overt confirmation of international transactions. Nonetheless, it is crucial to adopt a security-focused approach and acknowledge the potential for international transactions, even if not explicitly mentioned. Abiding by security best practices, it is wise to prepare for such scenarios in light of the lack of express exclusion. Accordingly, Artemis Financial should put in place a robust communication security plan that ensures the safety of both domestic and foreign interactions. This proactive strategy will safeguard data integrity and privacy, regardless of the transaction's origin.

Governmental Restrictions –

Ensuring secure communication is a pivotal aspect, especially in software applications that handle sensitive information. However, the significance of preserving communication integrity goes beyond the security of personal data. Consider Artemis Financial, a company specializing in tailoring financial and insurance plans for individual clients. This operation involves the exchange of information between the company and its clients, who confide sensitive particulars such as income and investments. Consequently, it is imperative for the company to institute robust data security protocols to shield this valuable information from unauthorized access.

External Threats –

In today's digital landscape, external threats are a prominent concern, particularly for Artemis Financial, as it deals with the management of sensitive financial and personal data. The primary focus of potential threats for this company revolves around individual or small-scale attacks, rather than attracting the interest of governments or high-level actors. These threats manifest through various channels, including the input of invalid data, the looming specter of SQL injection, inadvertent data exposure in the browser's history (e.g., usernames and passwords), and vulnerabilities stemming from software dependencies. The deployment of bots further heightens the risk, as they can be meticulously programmed to repetitively attempt logins using an array of usernames and passwords, potentially leading to a denial of service if executed at scale and speed. Furthermore, the risk extends to authorized users striving to access data beyond their designated authorization levels, potentially through privilege escalation, necessitating a fortified system with a robust API to repel potential threats.

Modernization Requirements –

The journey toward modernization is fraught with challenges, and even seemingly dependable open-source libraries and actively maintained software can introduce unforeseen hazards to the system. In the context of Artemis Financial's involvement in financial planning, the integration of compiled stock and bond data from external sources, while advantageous, carries the potential for added security vulnerabilities. This also holds true for the incorporation of third-party software aimed at enhancing the core program. If these external tools lack secure API structures or robust security measures, they pose a risk to the system's integrity.

Furthermore, with the continual evolution of web-application technologies, it becomes imperative to assess their implications for security. While modern web applications offer expanded functionalities, such progress may inadvertently expose the application to vulnerabilities when interactions with other applications are required. Attackers, becoming increasingly sophisticated over time, continuously devise new methods to breach system defenses. As a proactive safeguard, the principle of "default to fail" should be adhered to, giving priority to security in all system interactions and evolutions.

2. Areas of Security

The primary areas of concern in terms of security encompass input validation and APIs:

- Input Validation: The significance of input validation cannot be overstated, especially when dealing with user input in a program. In the case of this program, which accommodates user input, validation plays a critical role in upholding system security. For instance, consider line 12 in the CRUDController.java, where input is received in the form of an expected string. To avert potential failures or the risk of SQL injection, it is of utmost importance to rigorously validate this input. This measure is particularly vital because users are inputting data into the system, and any malicious intent could result in significant harm or unauthorized access to sensitive information.

- APIs: Considering that this application is designed for external use, particularly in web browsers, the need for a robust API cannot be overstated. The API serves as the rulebook that governs interactions between our program and third-party software, dictating what methods are allowed or restricted and which data is deemed acceptable or not. Furthermore, as this project may involve the integration of third-party software that our software depends on, a secure API becomes paramount to mitigate the potential risks stemming from external dependencies.

In addition to input validation and APIs, several other key areas require attention in the context of security:

- Cryptography: The significance of cryptography comes into focus, especially in the context of international transactions. As previously mentioned, secure and encrypted communication plays a pivotal role in ensuring the safety of international transfers. However, it's worth noting that cryptography is subject to export regulations. Hence, if Artemis Financial embarks on international operations, it becomes imperative to implement data security measures that align with the legal requirements of both the United States and the respective destination country.

- Code Error: Effective error handling is a fundamental pillar of system security. It serves as a crucial complement to input validation, particularly by addressing errors that may arise due to user inputs. Proper error handling is indispensable for preventing privilege escalation and mitigating vulnerabilities, especially those associated with input validation. Ensuring accurate error management is an essential aspect of upholding system security.

- Code quality takes on paramount significance, particularly when APIs and input validation are part of the system. Top-notch code quality is the linchpin for safeguarding sensitive data from unintended exposure and preventing restricted methods from being accessed by unauthorized end users. Essentially, quality code fortifies access controls, assuring that solely authorized users can interact with data and methods in alignment with their specific level of authorization.

3. Code Quality: Manual Review

Upon conducting a manual review of the code base provided for the Artemis Financial project, several important findings have come to light:

- Input Validation: The code's deficiency in input validation is a notable and concerning issue. Input validation serves as a linchpin for upholding system security, and its absence leaves the application vulnerable to an array of potential threats. Notably, the code lacks Apache-type validators, and user input, specifically within the greeting controller, remains unchecked for validation. This void in input validation opens the door to the processing of potentially malicious input without the necessary protective measures in place.

- API: Interestingly, the code base does not exhibit any apparent signs of an API. Despite the system's ability to accept data, the absence of a structured API is conspicuous. The data is accepted directly via the URL, which could result in data being exposed in the browser's history, potentially creating security vulnerabilities. Moreover, the lack of a well-documented and clearly defined method for users to interact with the system hinders its qualification as a RESTful API. Establishing a secure and well-defined API is imperative to enable effective user interaction with the system.

- Cryptography: The absence of cryptographic or encryption mechanisms is notable, especially if international communications become necessary. Artemis Financial should consider implementing encryption practices that conform to both U.S. and international regulations and legal requirements. This proactive approach will ensure that data remains secure and compliant with relevant laws when communicating on a global scale.

- Code Error Handling: Although some classes incorporate try/catch blocks, they frequently lack comprehensive error handling mechanisms. A notable example is the DocData.java class, which features a method for document reading but falls short in terms of effective error handling. Enhancements in error handling practices are necessary to ensure that exceptions and errors are managed efficiently, promoting system stability and security.

- Code Quality: The current codebase is operational but demands significant expansion and refinement to ensure comprehensive functionality. To achieve a fully prepared program, several areas necessitate enhancement. This encompasses optimizing the user-friendliness and security of the API, fortifying input validation procedures, and elevating the overall quality of the code. Furthermore, the system should be architected to support secure international communications, aligning with the company's objectives and regulatory requirements.

These findings emphasize crucial areas of concern within the codebase, emphasizing the necessity for substantial development and security improvements to guarantee the system's robustness and its ability to withstand potential threats and vulnerabilities.

4. Static Testing

The identified dependencies and their associated vulnerability IDs are critical to assess for potential security risks in the software project:

- bcprov-jdk15on-1.46.jar: This particular dependency offers cryptographic algorithms, but it comes with a significant vulnerability (cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-cryptography-api:1.46) that raises substantial concerns, especially concerning cryptography and secure communications. The vulnerability within version 1.46 has the potential to result in security issues where the validation of the second party's Diffie-Hellman (DH) public key is incomplete. This incomplete validation may expose private key details of other parties, jeopardizing the integrity of encryption. Furthermore, even the exposure of the algorithm used could lead to key discovery, rendering secure communications ineffective.

- jackson-databind-2.10.2.jar: This dependency provides essential data-binding functionality, but it is accompanied by vulnerabilities (cpe:2.3:a:fasterxml:jackson:2.10.2 and cpe:2.3:a:fasterxml:jackson-databind:2.10.2) that could allow malicious actors to submit XML data exceeding the intended scope. This unauthorized data submission could trigger error messages that, when exploited, might inadvertently expose sensitive information. These vulnerabilities have the potential to lead to data exposure if exploited effectively, posing a significant security risk.

- log4j-api-2.12.1.jar: This dependency relates to the Apache Log4j API, and it comes with a vulnerability (cpe:2.3:a:apache:log4j:2.12.1) that can compromise SSL security. In certain instances, SSL certificates may not be correctly validated or, in some cases, not validated at all. This could enable software to connect to a potentially malicious host while mistakenly considering it a trusted one. These vulnerabilities have the potential to erode the trustworthiness of SSL connections and mislead users regarding the actual security of their connections.

- snakeyaml-1.25.jar: This dependency relates to a YAML parser for Java, and it carries a vulnerability (cpe:2.3:a:snakeyaml_project:snakeyaml:1.25) that exposes the system to potential denial of service (DoS) threats. The vulnerability arises from uncontrolled recursive definitions within document type definitions, which can lead to exponential data expansion when parsing. Such DoS attacks can inundate a server or website, causing it to become overwhelmed and shut down. Effective defense measures are essential to safeguard against these prevalent types of attacks.

- spring-core-5.2.3.RELEASE.jar: This dependency is associated with the Spring Core framework, and it is important to note that vulnerabilities (cpe:2.3:a:pivotal_software:spring_framework:5.2.3, cpe:2.3:a:springsource:spring_framework:5.2.3, and cpe:2.3:a:vmware:springsource_spring_framework:5.2.3) are present. These vulnerabilities impact Spring versions up to 5.2.8 and could potentially allow attackers to bypass protections against reflected file download (RFD) attacks through the jssessionid path. RFD exploits can be quite concerning, as they have the potential to provide attackers with a significant level of control over the targeted system.

- tomcat-embed-core-9.0.30.jar: This dependency is associated with the core implementation of Tomcat. Vulnerabilities (cpe:2.3:a:apache:tomcat:9.0.30, cpe:2.3:a:apache_software_foundation:tomcat:9.0.30, and cpe:2.3:a:apache_tomcat:apache_tomcat:9.0.30) have been identified. These vulnerabilities may present risks related to reflected file download (RFD) attacks, potentially compromising the security of the targeted system.

The identified vulnerabilities within the following dependencies need to be addressed to secure the software project:

- spring-core-5.2.3.RELEASE.jar: This dependency comes with well-documented vulnerabilities, including a major one that could potentially grant unauthorized individuals access to sensitive information. This particular vulnerability arises when the code inserts information directly or indirectly, or when it manages resources containing confidential data that could inadvertently become accessible to users without proper authorization. The risk lies in users gaining access to a method responsible for sensitive data, whether accidentally or through intentional manipulation of the program for unauthorized data retrieval.

- tomcat-embed-websocket-9.0.30.jar: Similarly to the prior dependency, this one also has well-documented vulnerabilities. The specific vulnerabilities are consistent, and the Common Platform Enumeration (CPE) references match those previously discussed. These issues introduce potential risks associated with exposing sensitive information, and they demand prompt attention and resolution

Addressing these vulnerabilities is critical to safeguard the software project from unauthorized data access and potential exploitation. Mitigation efforts could include applying security patches or updates to the affected dependencies, implementing additional security controls, or reviewing and modifying the code to enhance its security. Understanding these vulnerabilities is crucial for the project's security. Mitigating actions such as updating dependencies or implementing additional security measures may be necessary to address these risks effectively.

5. Mitigation Plan

To enhance security for this software project, the mitigation plan involves a shift from a DevOps pipeline to a DevSecOps pipeline. This transition aims to instill a security-oriented mindset in the development team and address potential threats effectively. As the project is still in its nascent stages, this proactive approach ensures security considerations are woven into its fabric from the outset. Here are the detailed mitigation measures for the identified threats:

- bcprov-jdk15on-1.46.jar: To mitigate this threat, it is recommended to use a version numbered 1.56 or later. These later versions include checks on key parameters during agreement calculations, which are vital for secure cryptography. Proper key validation and storage are essential to ensuring the effectiveness of cryptography. Validating keys before any data exchange is crucial.

- jackson-databind-2.10.2.jar: To mitigate this threat, configure the XML parser and validator to disable external entity expansion. This measure prevents attackers from submitting altered DTD files within XML documents by having the parser detect and ignore any malicious code properly.

- log4j-api-2.12.1.jar: Mitigating this threat involves not using certificate pinning or ensuring that all certificate properties are thoroughly validated before pinning, with a special focus on the hostname. Properly enforcing SSL certificates is crucial to maintaining the trust of browsers and end users, as it ensures a genuine sense of security.

- snakeyaml-1.25.jar: To mitigate this threat, prohibit the use of Document Type Definitions (DTDs) or use a parser that limits the expansion of recursive definitions. Limiting the expansion of recursive calls can prevent the generation of large loops, whether accidental or deliberate. This measure safeguards against potential Denial of Service (DoS) attacks.

- spring-core-5.2.3.RELEASE.jar: While there are no specific mitigation measures listed for this vulnerability, one effective approach could be upgrading to the latest version, 5.3.1. Utilizing updated software ensures that the latest patches for known vulnerabilities are applied. However, it's essential to balance this with caution, as the newest software may not yet have real-world exposure to identify vulnerabilities.

- tomcat-embed-core-9.0.30.jar & tomcat-embed-websocket-9.0.30.jar: Mitigating these threats involves implementing the principle of least privilege and creating safe zones within the system. By confining sensitive information within secure structures and enforcing strict authorization mechanisms, the risk of data leakage is significantly reduced.

These mitigation measures will enhance the security of the software project by addressing specific vulnerabilities and potential threats, ensuring that security is an integral part of the development process from the outset.