****

# Artemis Financial Vulnerability Assessment Report

Table of Contents

[Document Revision History 3](#_Toc32574607)

[Client 3](#_Toc32574608)

[Instructions 3](#_Toc32574609)

[Developer 4](#_Toc32574610)

[1. Interpreting Client Needs 4](#_Toc32574611)

[2. Areas of Security 4](#_Toc32574612)

[3. Manual Review 4](#_Toc32574613)

[4. Static Testing 4](#_Toc32574614)

[5. Mitigation Plan 4](#_Toc32574615)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **1/17/2023** | **Austin Palmer** | **Document Creation** |

## 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

Austin Palmer

## Interpreting Client Needs

As a financial institution, Artemis Financial needs to incorporate a high-level of security with their systems due to the importance and stakes of the information. Card info, account info, and other personal information such as a customer’s name, address, DOB, and social security number are typically held by financial institutions and if a hacker gained access to this information, they could cause substantial damage. In addition to this information, Artemis Financial handles thousands of domestic and international transactions a day so security is essential to protect these transactions from being intercepted. Countries utilize the client’s services need to be analyzed to determine any possible international restrictions. Artemis Financial is involved in many types of financial plans which expands the possibility of threats as more private information is stored and long-term investments will be a high-value target.

## Areas of Security

* Input Validation: A RESTful API requires that the user interacts with the system by providing it input. In this application, this includes, but is not limited to, logging into an account, entering in transaction amounts, and creating new or changing financial plans. For this reason, checking what the user inputs will block a potential attacker from messing with the code structure.
* APIs: The Artemis Financial application runs using a RESTful API which links the user and the database of their financial information. API interactions can be intercepted using proxy interceptors and then manipulated to change the request of the user. This can then be used to extract or leak data from the database. Additionally, intercepted API interactions can be used in a denial of service (DoS) attack which could result in users to temporarily lose access to their financial information and could even effect admin users which could create even more problems if the attacker gains admin access.
* Cryptography: Due to the sensitivity of the information in this application, encryption needs to be used to protect the data from possible interceptions. By encrypting passwords, DOB, and social security numbers the information is more difficult to understand to an attacker and it will keep it safe even in an international transaction where Artemis Financial may have less control.
* Client/Server: Since this system uses a RESTful API, there is a distribution of the client and the server. This allows users to interact with the client side which then retrieves the information on the server side. The sensitivity of information and the amount of API interactions that are handled means that a focus should be put on securing the client and server to ensure the functionality and security of all processes.

## Manual Review

* In the GreetingController.java file line 16-17 does not validate the input data of name. This could result in an attacker creating a malicious string that could change the function or output of the data.
* In the customer.java file line 7-9 has the function showInfo() which shows a customer’s account number without any parameters. If a hacker has access to a user’s account info through the means of string manipulation they could call this method and easily gain access to a user’s accounting by accessing the account number.
* There is no form of access control specified. This is a must as keeping track of and knowing a user’s permissions will help keep attackers from gaining access.
* There is no form of session management. Only one session should be allowed to be opened at one point and the session should expire after a short length of time.

## Static Testing

* CVE-2022-42252: Apache Tomcat version was configured to ignore invalid HTTP headers making a request smuggling attack possible. A solution has not been found yet.
* CVE-2021-44832: Apache Log4j2 version was vulnerable to a remote code execution attack by altering the JBAC Appender. A solution was found as a JDBC Appender should use JndiManager when accessing JNDI. Link: <https://issues.apache.org/jira/browse/LOG4J2-3293>
* CVE-2020-10693: Hibernate Validator version had a bug to evaluate invalid EL expression as if they were valid. A solution was found to unwrap the context of HibernateConstraintValidatorContext. Link: <https://bugzilla.redhat.com/show_bug.cgi?id=CVE-2020-10693>
* CVE-2016-1000352: Bouncy Castle JE provider version allowed the use of ECB mode which is regarded as unsafe. This code has been discontinued.
* CVE-2022-42004: FasterXML jackson-databind version could result in resource exhaustion due to the possibility of deeply nested arrays. A solution was found by adding a check into BeanDeserializer.\_deserializeFromArray(). Link: <https://github.com/FasterXML/jackson-databind/issues/3582>
* CVE-2022-27772: Spring-boot version was vulnerable to directory hijacking. A solution was found inadvertently as this was updated as part of patch v2.2.11. Link: <https://github.com/JLLeitschuh/security-research/security/advisories/GHSA-cm59-pr5q-cw85>
* CVE-2022-41854: Using Snakeyaml to parse untrusted YAML files may be vulnerable to a DOS attack. A solution was found as part of an update to version 1.32 but does not list what was fixed. Link: <https://lists.fedoraproject.org/archives/list/package-announce@lists.fedoraproject.org/message/7MKE4XWRXTH32757H7QJU4ACS67DYDCR/>
* CVE-2022-22971: Spring Framework version with a STOMP over Websocket endpoint is vulnerable to a DOS attack. A solution has not been found yet.

## Mitigation Plan

* Input verification is crucial for this application due to the reliance of user input. Malicious string manipulation could result in an attacker gaining control of an account or even gain admin permissions.
* Implementing a role-based access control (RBAC) will help control the access of users and limit access to attackers.
* Implementing a session time and number limit will keep attackers from intercepting a long user session and refreshing it to gain access to that account.
* Using up-to-date versions of 3rd party libraries will reduce dependencies and protect the application from falling victim to common attacks.
* Parameterized functions (especially any used in SQL) will eliminate SQL injection attacks and protect user data from database attackers.
* Hiding the information of an HTTPS request in the body of the request will hide the information from attackers to create more secure client/server interactions.