****

# 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** | **1/28/24** | **Kat Gilliland** | **Conducted vulnerability assessment and reported findings/mitigation plan** |

## Client



## Developer

Kat Gilliland

## Interpreting Client Needs

Artemis Financial, a consulting company, develops individualized financial plans for their clients. The financial plans they offer include savings, retirement, investment, and insurance plans. Because Artemis Financial deals with extremely sensitive data (social security numbers or SSNs, contact information, financial documents, etc.,) they have a major responsibility to keep that information and the systems they utilize for their business both protected and secure.

In addition to U.S. based clients, Artemis Financial may service international clients; if this is the case, they will need to enforce an extra layer of security when it comes to handling international transactions as these types of financial arrangements can often be targets of specific fraud and phishing attempts. International financial scams may include but are not limited to identity theft, money laundering, and wire fraud.

Artemis Financial should also be aware of governmental restrictions regarding secure communications. Organizations within the financial service industry, such as Artemis Financial, need to consider the data protection laws and regulations in the areas in which they serve when deciding which security measures are most vital to their online platforms. All financial entities in the United States, for example, must comply with the cybersecurity guidelines established by The Federal Financial Institutions Examination Council (FFIEC). Understanding and adhering to cybersecurity regulations is not only mandatory in most scenarios but also an effective strategy for maximizing an institution’s security (Kost, 2023).

Since Artemis Financial handles a great deal of personally identifiable information (PII), there are various external threats for them to be on the lookout for. Common cyberattacks affecting web-based software include SQL injection attacks, broken authentication, and input validation attacks. SQL injection attacks involve a malicious actor attempting to interfere with the queries sent from an application to its database. In a successful SQL injection attack, the attacker will gain unauthorized access to sensitive data such as personal user details, credit card information, and user login credentials. A broken authentication attack can occur when an application is not able to properly authenticate its users. Hackers will take advantage of an organization’s broken or incorrectly implemented authentication mechanisms to gain access to its user’s private information. In a broken authentication attack, attackers compromise passwords and session tokens to assume a user’s identity and gain access to their personal details. Web-based applications can also fall victim to input validation attacks. Input validation attacks involve the injection of harmful input into a platform’s user input fields. If an application does not inspect and filter through the input that is entered into it, the system could behave unexpectedly, resulting in disastrous events including the previously mentioned SQL injection attacks, buffer overflow, and distributed denial of service (DDoS) attacks. The list of cyber threats and attacks will only continue to grow as companies, like Artemis Financial, become more digital; therefore, it is crucial for these types of organizations to stay vigilant when it comes to identifying vulnerabilities and risks within their web-based systems.

Modernized and digitally transformed applications have a handful of requirements to consider. Modernized application requirements may include using open-source libraries and frameworks to develop software, utilizing cloud computing environments and tools for scalability and adaptability, and implementing encapsulation for more flexibility and control over data. Regardless of the modernization technique used, application modernization is crucial to ensuring that software is relevant, up-to-date, and most importantly, secure.

1. **Areas of Security**

The areas of security for Artemis Financial to focus on are as follows:

**Input Validation** – Input validation is the process of examining input or data entered by a user to ensure it is properly formatted before allowing it to enter the software.

**Justification:** Input validation confirms that before data is processed by an application or software, it is accurate and meets an organization’s requirements for quality. Because Artemis Financial is using a web-based platform for their services, they are at risk for specific kinds of cyber-attacks, such as the previously mentioned SQL injection attack and another attack known as cross-site scripting (XSS). In the case of a financial institution, SQL injection attacks can be used by a bad actor to gain unauthorized access to other’s bank records and financial accounts. Once unauthorized entry into an account has been made, the attacker can engage in malicious activities such as stealing PII, transferring money to an external account, or voiding transactions (Acunetix, 2024). Web-based applications are also targets for cross-site scripting or XSS attacks. In a cross-site scripting attack, an attacker will inject harmful code into otherwise safe and trustworthy web applications to gain access to sensitive data. For example, on a web-based financial platform, an attacker may perform an XSS attack to fraudulently obtain users’ login credentials. The hacker may use the credentials and unauthorized account access to commit acts such as identity theft or financial fraud.

**APIs** – Application programming interfaces (APIs) follow a specific set of rules and guidelines to allow applications the ability to communicate and share information with one another.

**Justification:** In the scenario shared for Artemis Financial, it was noted that they are utilizing a RESTful web application programming interface (REST API); this type of interface is essential to the development of modern technology because it allows various types of software the ability to communicate with one another, such as enabling web-applications to interact with mobile-applications. REST APIs are also beneficial to modern applications because they use standard and simple-to-use HTTP requests to manage data. Methods such as GET, POST, PUT, and DELETE to easily determine which action should be performed on a specific resource. Since APIs revolve around the transferring of data, which in the case of a financial institution would be quite sensitive in nature, it is crucial for Artemis Financial to secure their REST API. If the REST API is not secured, bad actors will seek out vulnerabilities in the API endpoints; if a vulnerability is found, the attacker may use it to gain unauthorized access to private data and perform harmful activities.

**Cryptography** – Cryptography is the process of using encryption and decryption algorithms to keep data protected and private from third parties or external users.

**Justification:** Web-based applications, like Artemis Financial, should employ cryptography techniques in their software to ensure sensitive information such as financial records and login credentials cannot be accessed by unauthorized parties. Encryption such as Transport Layer Security (TLS) encryption is important for securing a REST API, the architecture model being used by Artemis Financial, as it will protect any data in transit. Essentially, TLS encryption enforces private communication between systems, thus adding an extra layer of security to an application’s data.

**Code Error –** Secure error handling is vital to software applications as it minimizes the possibility of a system’s internal architecture being exposed to attackers. In addition to preventing a variety of cyber-attacks, utilizing successful error handling techniques will increase software’s reliability and stability and boost its overall user experience.

**Justification:** Graceful error-handling will enhance the quality of a RESTful API; the architecture model being used by Artemis Financial for their web-based platform. Since Artemis Financial is utilizing the Spring framework for their application, they have a ton of features for error handling at their disposal, many of which could greatly contribute to their system’s security and user satisfaction. For example, Artemis Financial can take advantage of the Spring Framework’s @ControllerAdvice and @ExceptionHandler annotations to effectively handle exceptions and provide useful responses to their users (Iroshan, 2024). The exception handling tools offered by the Spring Framework will offer meaningful responses to the system’s users when something unexpected happens instead of affecting system performance or crashing it entirely.

**Code Quality** – Software’s quality is typically measured by how well-structured and maintainable its code is. Making sure a program’s code is high-quality will reduce the likelihood of bugs and defects and minimize the amount of security vulnerabilities.

**Justification:** Artemis Financial is responsible for safeguarding very sensitive information, therefore, they must have strong security measures in place for their software. One way Artemis Financial can ensure their software is protected is by consistently testing their code for quality. Static testing techniques, such as the static test conducted for this analysis, are useful for automatically verifying that code is up to standard, checking for bugs, and identifying possible security vulnerabilities.

## Manual Review

In manually reviewing the code, the following vulnerabilities have been identified:

* In the CRUDController.java class, on line 13, the variable “business\_name” is sent as a request parameter (@RequestParam). Query parameters such as this one can be dangerous as the data in the request parameter can be saved in the user’s browsing history and become the target for malicious attacks that can result in leaked passwords, tokens, etc.
* The program does not use HTTPS to transfer data. It looks as though no cryptography methods are used in this software.
* In the GreetingController.java class, there are no input validation methods put in place to ensure the data entered is formatted correctly and secure before allowing it to enter the system.

## Static Testing

A screenshot of a service

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| **Known Vulnerability Name** | **Vulnerability Code(s)** | **Description** | **Solution** |
| Bouncy Castle Crypto Package | **CVE-2015-6644** (OSSINDEX) | Bouncy Castle in Android before 5.1.1 LMY49F and 6.0 before 2016-01-01 allows attackers to obtain sensitive information via a crafted application, aka internal bug 24106146. Exposure of information to unauthorized actor. | To avoid information from being leaked to unauthorized individuals, proper cryptography methods need to be put in place. To resolve the cryptography issues in this program, the Bouncy Castle Crypto package should be updated to its most recent version. |
| Hibernator’s Bean Validator | [**CVE-2020-10693**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=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. This software is using improper methods of input validation. | To prevent input validation attacks such as SQL injection attacks and cross-site scripting (XSS) attacks, validator should be updated to the most recent version and proper input validation methods should be implemented. |
| Core Tomcat Implementation | [**CVE-2020-1938**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1938) | 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. | It is recommended to update Apache Tomcat, however, with the number of issues stemming from this HTTP connection, it is in Artemis Financial’s best interest to switch to using a more secure, HTTPS connection. |

## Mitigation Plan

* Instead of using request body parameter for the “business\_name” variable in the CRUDController.java class, the information should be sent as a header. If this data is sent as a request instead of passed through as a header, the information may be visible to attackers or outside users who are able to access the system’s request histories. Using a header instead will add an extra layer of security to this part of the software.
* Artemis Financial should secure their API by using hashing algorithm for password security and HTTPS for data transmission as HTTPS encrypts data. HTTPS is more secure than HTTP and vital to institutions that manage such sensitive and personally identifying information. By using HTTPS, Artemis Financial can protect all forms of communication and customer data stored within its database. HTTPS secures connections by using a public-private key pairing to safely transfer data between a browser and server (Chipeta, 2023).
* In the GreetingController.java class, input validation should be put in place to guarantee that the data being entered by the user aligns with the format allowed by the system.
* Per the results from the static test, the Bouncy Castle Crypto package and the Hibernator’s Bean Validator should be updated to the most recent versions for optimal software security.

**References**

Acunetix. (2024). What is SQL injection (SQLi) and how to prevent it. *Acunetix by Invicti*, 1. <https://www.acunetix.com/websitesecurity/sql-injection/#:~:text=For%20example%2C%20in%20a%20financial,a%20database%2C%20even%20drop%20tables>.

Chipeta, C. (February 7, 2023). What is HTTPS? How it works and why it’s so important. *UpGuard*, 1. https://www.upguard.com/blog/what-is-https

Iroshan, A. (January 27, 2024). Exception handling – spring boot REST API. *Medium*, 1. https://medium.com/thefreshwrites/exception-handling-spring-boot-rest-api-c2656b575fee

Kost, E. (June 27, 2023). Top 8 cybersecurity regulations for financial services. *UpGuard*, 1. https://www.upguard.com/blog/cybersecurity-regulations-financial-industry