****

# Artemis Financial Vulnerability Assessment Report

Table of Contents

[Artemis Financial Vulnerability Assessment Report 1](#_Toc118937193)

[Document Revision History 3](#_Toc118937194)

[Client 3](#_Toc118937195)

[Instructions 3](#_Toc118937196)

[Developer 4](#_Toc118937197)

[1. Interpreting Client Needs 4](#_Toc118937198)

[2. Areas of Security 6](#_Toc118937199)

[3. Manual Review 9](#_Toc118937200)

[4. Static Testing 11](#_Toc118937201)

[5. Mitigation Plan 14](#_Toc118937202)

[References 17](#_Toc118937203)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **11/7/2020** | **Jonathan Kleven** | **Initial Report** |

## 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

Jonathan Kleven

## Interpreting Client Needs

To provide the best benefit to the client, a web-based software application needs to be useful and provide value and ensure that value cannot be compromised by unwanted malicious threats. The client Artemis Financial has a similar goal. Artemis Financial seeks to modernize its operations and provide its customers a more practical and valuable tool. As a financial company, secure communications ensure confidence among their customers that they are working to their benefit. Finances within the organization, as well as among the customers, are privileged and sensitive information. Information that needs to be transmitted to and from their customers.

Artemis Financial creates financial plans for its customers. Many of these plans will deal with investments, both foreign and domestic. The field of financial institutions has many federal regulations and guidelines that outline how Artemis Financial can conduct its business. Some rules and policies must be followed regarding the customer's sensitive information and how that information is handled within the organization and communicated to the customers.

The "OWASP Secure Coding Practices Quick Reference Guide" (2010) asserts, "it is much less expensive to build secure software than to correct security issues in a completed software package, not to mention the costs that may be associated with a security breach." Artemis Financial working with Global Rain, understands that recovering from a threat will be more expensive, both in financial losses and lost confidence and reputation. External threats prevalent today and likely in the future are the acquisition of business and customer information. Business information gathered can compromise its customers or business operations. Most financial institutions handle large sums of money. How they handle that money regarding the customer's financial plans can be helpful information to an attacker. Customer information can be used to compromise their identity, gain access to other customers, or access sensitive business information.

To work towards Artemis Financials' goals to modernize their operations and ensure they receive a web-based software application with current and adequate security, Global Rain must ensure we are working with the latest knowledge of coding practices and known vulnerabilities. The landscape is constantly evolving, with threats and vulnerabilities being discovered frequently. Global Rain's relationship with Artemis Financial must continue to provide value and security. As vulnerabilities are found, it is essential to set guidelines on how Global Rain will work towards shoring up those potential areas of threat with updates and patches while not compromising Artemis Financials' useability and value to their customers.

There are many avenues to explore to ensure the highest level of security for the Artemis Financial web-based software application. Many of them involve open-source libraries and evolving web application technologies. Open-source libraries are free libraries that offer a variety of methods to enhance a software application, from backend and server support to user interfaces and communication to and from servers.

The technologies used in applications are constantly evolving and improving. This may be in the form of an organization-specific application for mobile devices or in development tools that Global Rain can use to better serve its clients.

## Areas of Security

Areas of security that are important to identify for the Artemis Financial web-based software application are Input Validation, APIs, Cryptography, Client/Server, Code Error, and Code Quality. Threats can occur during the interaction with each of these areas or in a combination of these areas. In addition, they offer many avenues of value for communication from the customer to Artemis Financials' application but can also act as fields for malicious activity. Therefore, focusing on these areas and balancing valid user interactions with understanding and parsing methods and APIs within these areas often exploited by attackers are important security focuses for web-based applications.

**Input Validation**

"Performing validation early can be beneficial, as it will reject invalid input sooner and reduce exposure to malformed data." (Secure Coding Guidelines for Java SE, 2022). Input validation is essential for many reasons, not just usernames and passwords. Customers will interact with the application to send and receive highly sensitive information. Understanding the format, this data is in will help in application security. "Maliciously crafted inputs may cause problems, whether coming through method arguments or external streams" (Secure Coding Guidelines for Java SE, 2022). Additionally, input validation helps confirm the data requested and transferred was appropriately formatted and from potentially trusted sources.

**API**

Artemis Financial uses a RESTful application programming interface or API. Understanding the behavior of the client's API will help ensure that trusted data is received and transferred to trusted sources. "Verifying the API behavior is especially important when validating untrusted data." (Secure Coding Guidelines for Java SE, 2022).

**Cryptography**

Whether customers using the Artemis Financial application are located outside the United States (US) or if customers are making financial decisions within the application outside the US, cryptography is essential to safeguard data if intercepted. "Encrypt highly sensitive stored information, like authentication verification data, even on the server side." (OWASP Secure Coding Practices Quick Reference Guide, 2010). "When handling sensitive data, you need to encrypt that data both in transit and at rest." (Secure Coding Guidelines for Java SE, 2022)

**Client/Server**

The web-based software application will communicate with the Artemis Financial server and its database to provide information relevant to each user. "…[A] web browser is outside of the system for a web server. Equally, a web server is outside of the system for a web browser. Therefore, web browser and server software should not rely upon the behavior of the other for security." (Secure Coding Guidelines for Java SE, 2022). So, understanding the security implemented on the Artemis Financial server is vital for the interaction of the web-based application.

**Code Error**

"Exceptions may occur for a number of reasons: bad inputs, logic errors, misconfiguration, environmental failures (e.g., network faults), and so forth. While it is best to prevent or avoid situations that cause exceptions in the first place, secure code should still assume that any exception may occur at any time." (Secure Coding Guidelines for Java SE, 2022). Although errors can and will happen, understanding and preparing for them ahead of time can improve the usefulness of an application. Still, it can hinder an attacker from taking advantage of these errors. "Unexpected input and error conditions may cause native code to behave unpredictably." (Secure Coding Guidelines for Java SE, 2022). It is in this unpredictable behavior that vulnerabilities can develop. Ensuring the application can handle errors and redirect, log or restart will help make these unexpected behaviors from opening the system to threat.

**Code Quality**

"Secure systems need to make effective use of these mechanisms in order to achieve their desired quality, security, and robustness goals." (Secure Coding Guidelines for Java SE, 2022). While code errors can be expected, ensuring the application only behaves as intended is more practical. "It is important for applications to minimize exceptions by utilizing robust resource management, and also by eliminating bugs that could result in exceptions being thrown." (Secure Coding Guidelines for Java SE, 2022). We must ensure that any resources being utilized are specific as a form of access control but that once that resource is no longer needed, it is closed or returned to rest. If any part of the application fails in these circumstances, it may be vulnerable to attacks such as denial of service.

"However, since exceptions may also be thrown due to unforeseeable or unavoidable conditions, secure systems must also be able to safely handle exceptions whenever possible." (Secure Coding Guidelines for Java SE, 2022)

## Manual Review

A review of the given code was conducted and resulted in several findings. Among those findings were within the JAVA IDE console: "Tomcat started on port(s): 8081 (HTTP) with context path ''. This indicates that the application is running on HTTP rather than HTTPS. Furthermore, "HTTP, or Hypertext Transfer Protocol, is a simple, stateless, request and response protocol that drives the vast majority of web traffic." (J. Manico & A. Detlefsen,2014). In contrast, "HTTPS, or Hypertext Transport Protocol Secure, provides a variety of different encryption services to protect user data during transport, such as protecting data while it's traveling over a wired or wireless network." (J. Manico & A. Detlefsen,2014).

Input validation and code error handling were some issues discovered in the CRUDController.Java and GreetingController.Java classes. However, within these classes, it is possible to pass a payload value to interact with the application returning a successful status code of 200 OK.

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

When interacting with the browser and attempting to pass a payload variable of "name," an error occurred within the application that noted the error with the URL. "Required String parameter 'business\_name' is not present."

Graphical user interface, text, application

Description automatically generated

This shows the viewer how to correct the error despite a successful request. This example clearly indicates that better error handling is needed as these errors provide an attacker more insight into the inner workings of the application's source code and how to revise the URL to get a successful request.

Within the DocData.Java class, there appears to be unfinished code. For example, this class is called from the CRUDController class. Additionally, there is a warning within the IDE: "the value of the local variable 'con' is not used."

Text, letter

Description automatically generated

Another note is some classes were not being called by the application. Customer.Java and myDateTime.java appear to aid in providing the application's customer and time and date information. Incorporating these classes into the application will require additional security to ensure access to these classes protects the data from being called from untrusted sources. Customer.Java is noteworthy as it provides a means to return customers' account numbers. This sensitive information should be partially or entirely hidden from transmission to the web application as that information can otherwise leak to an untrusted source.

Text

Description automatically generated Text

Description automatically generated

## Static Testing

Using the Maven Dependency-Check plugin with the provided code base identified 11 dependencies and their respective vulnerabilities from the National Vulnerability Database (NVD). The dependencies and their vulnerability codes are as follows:

**bcprov-jdk15on-1.46.jar**

Several vulnerabilities were identified that indicate issues with versions before 1.55. Therefore, this dependency should be updated to the latest version. The current version is 1.70 (MVN Repository, n.d.).

Vulnerability Identification Code (National Institute of Standards and Technology: National Vulnerability Database, n.d.):

* CVE-2020-26939
* CVE-2020-15522
* CVE-2020-0187
* CVE-2018-5382
* CVE-2017-13098
* CVE-2016-1000352
* CVE-2016-1000346
* CVE-2016-1000345
* CVE-2016-1000344
* CVE-2016-1000343
* CVE-2016-1000342
* CVE-2016-1000341
* CVE-2016-1000339
* CVE-2016-1000338
* CVE-2015-7940
* CVE-2015-6644
* CVE-2013-1624

**hibernate-validator-6.0.18.Final.jar**

The below-identified vulnerability was identified in versions of this dependency before version 6.1.2. Therefore, this dependency should be updated to the latest version. The current version is 7.0.0 (MVN Repository, n.d.).

Vulnerability Identification Code (National Institute of Standards and Technology: National Vulnerability Database, n.d.):

* CVE-2020-10693

**jackson-databind-2.10.2.jar**

The vulnerabilities below were found in the dependency versions before 2.14.0. Therefore, this dependency should be updated to the latest version. The current version is 2.14.0 (MVN Repository, n.d.).

Vulnerability Identification Code (National Institute of Standards and Technology: National Vulnerability Database, n.d.):

* CVE-2022-42004
* CVE-2022-42003
* CVE-2020-36518
* CVE-2020-25649

**log4j-api-2.12.1.jar**

This vulnerability did not verify the hostname matched the SSL/TLS certificate. This dependency should be updated to the latest version. The current version is 2.19.0 (MVN Repository, n.d.).

Vulnerability Identification Code (National Institute of Standards and Technology: National Vulnerability Database, n.d.):

* CVE-2020-9488

**logback-core-1.2.3.jar**

This vulnerability can allow an attacker with appropriate privileges to create or revise configuration files. Therefore, this dependency should be updated to the latest version. The current version is 1.4.4 (MVN Repository, n.d.).

Vulnerability Identification Code (National Institute of Standards and Technology: National Vulnerability Database, n.d.):

* CVE-2021-42550

**snakeyaml-1.25.jar**

This dependency and version are vulnerable to denial of service attacks by supplying content that can cause the application to crash. Therefore, this dependency should be updated to the latest version. The current version is 1.33 (MVN Repository, n.d.).

Vulnerability Identification Code (National Institute of Standards and Technology: National Vulnerability Database, n.d.):

* CVE-2022-38752
* CVE-2022-38751
* CVE-2022-38750
* CVE-2022-38749
* CVE-2022-25857
* CVE-2017-18640

**spring-boot-2.2.4.RELEASE.jar**

A vulnerability was discovered to allow temporary directory hijacking. This dependency should be updated to the latest version. The current version is 2.7.5 (MVN Repository, n.d.).

Vulnerability Identification Code (National Institute of Standards and Technology: National Vulnerability Database, n.d.):

* CVE-2022-27772

**spring-core-5.2.3.RELEASE.jar AND spring-web-5.2.3.RELEASE.jar**

Both dependencies share the below vulnerability identification codes. Among the vulnerabilities was the ability to execute remote code and susceptibility to allow an attacker to escalate the user's privileges. This also allowed an attacker to create or modify temporary storage directories. Therefore, these dependencies should be updated to the latest versions. The current version is 5.3.23 (MVN Repository, n.d.).

Vulnerability Identification Code (National Institute of Standards and Technology: National Vulnerability Database, n.d.):

* CVE-2022-22971
* CVE-2022-22970
* CVE-2022-22968
* CVE-2022-22965
* CVE-2022-22950
* CVE-2021-22118
* CVE-2021-22096
* CVE-2021-22060
* CVE-2020-5421
* CVE-2016-1000027

**tomcat-embed-core-9.0.30.jar AND tomcat-embed-websocket-9.0.30.jar**

These dependencies and their versions are vulnerable to remote code execution and denial of service attacks on the applications that control file uploads. Both dependencies share the below vulnerability identification codes. Therefore, these dependencies should be updated to the latest versions. The current version is 10.1.1 (MVN Repository, n.d.).

Vulnerability Identification Code (National Institute of Standards and Technology: National Vulnerability Database, n.d.):

* CVE-2022-42252
* CVE-2022-34305
* CVE-2022-34305
* CVE-2022-29885
* CVE-2021-43980
* CVE-2021-41079
* CVE-2021-33037
* CVE-2021-30640
* CVE-2021-25329
* CVE-2021-25122
* CVE-2021-24122
* CVE-2020-9484
* CVE-2020-8022
* CVE-2020-1938
* CVE-2020-1935
* CVE-2020-17527
* CVE-2020-13943
* CVE-2020-13935
* CVE-2020-13934
* CVE-2020-11996
* CVE-2019-17569

## Mitigation Plan

Based on the above dependency check, all dependencies should be updated to their latest versions. Of all the dependencies and the known vulnerabilities in the National Vulnerability Database (NVD), the resolution to those vulnerabilities was remedied in later patches or versions. As mentioned in section 2, the areas of focus for improved security within the Artemis Financial Web-bases software application are Input Validation, APIs, Cryptography, Client/Server, Code Error, and Code Quality.

To aid with input validation, it is essential to "define wrappers around native methods," validate user input as well as method output or return values, and "verify API behavior related to input validation" (Secure Coding Guidelines for Java SE, 2022). Input validation will help secure the application by ensuring the correct data is passed, and the accurate information is returned.

"To offer some of these protections during the invocation of native code, do not declare a native method public. Instead, declare it private and expose the functionality through a public Java-based wrapper method." (Secure Coding Guidelines for Java SE, 2022). Checking inputs through public methods to perform the validation and then passing the relevant data, if/when successful, to a private native method provides more secure data handling.

"Whenever possible, processing untrusted input should be avoided." (Secure Coding Guidelines for Java SE, 2022). Suppose the input is not appropriately structured or is provided in a manner that may yield a successful return value. In that case, it should be assumed that a possibility of an untrusted input was provided. Additional checking of the return value with the user session will help to avoid exposing sensitive data to an untrusted source.

Understanding the methods and behavior of the Artemis Financial API is critical to add security to the application. "it is critical to ensure that all authenticated actions do indeed contain a proper and active session." (Manico & Detlefsen, 2014). This can be in the form of validated inputs and ensuring the session is from an authentic user with the proper level of access.

To aid with Cryptography, implementing HTTPS protocol offers methods to secure data from the user to the server, including over wireless networks. "When handling sensitive data, you need to encrypt that data both in transit and at rest." (Manico & Detlefsen, 2014) Most of the data being sent and received by the user and the application should be considered sensitive. Account access and financial information should be encrypted to provide a layer of protection from hijacking such data from an intermediate source, such as a man-in-the-middle attack.

Client/Server mitigation techniques are common to manage client and server resources, i.e., memory, disk space, and processor use. Releasing or closing these resources at the end of a method call or session releases the burden on the system and halts any open processes. Unfortunately, an attacker can use these burdens to disrupt the application by making the hardware work harder than the application was meant to. This can crash the application entirely and prevents the system from working as intended.

Handling errors in the code or how the application takes input and requests ensures that the application continues to function (i.e., not crash) despite the request. These errors are opportunities to inform the user about an error and the data that caused it. The authenticated user can then be allowed to correct and try again. Suppose the error continues or the source is untrusted. In that case, the error can potentially ensure that the application halts the session and requires the user to log in again to verify authenticity. "Utilize locking to prevent multiple simultaneous requests to the application or use a synchronization mechanism to prevent race condition" (OWASP Secure Coding Practices Quick Reference Guide, 2010). In addition, when an error occurs, it should be logged onto the system - identifying the class or function and the manner that caused it. Identifying repeated user errors or a possible attack may be difficult, and the log files should not contain sensitive data. However, logging errors allows the developers to review issues with the application or determine causes, such as if an attack was attempted on the system.

The quality of the code for the application should specifically call out the resources it needs when it comes to memory, disk space, or processing power. Within the code, it should "Double check that the buffer is as large as specified" and "Check buffer boundaries if calling the function in a loop and make sure there is no danger of writing past the allocated space. "(OWASP Secure Coding Practices Quick Reference Guide, 2010) this is one avenue used to plague a system with a denial-of-service attack. Before and after a function, the buffer should be checked, and if the data used exceeds that buffer, it should end and provide an error and log the event.

Finally, "Review all secondary applications, third party code and libraries to determine business necessity and validate safe functionality, as these can introduce new vulnerabilities" (OWASP Secure Coding Practices Quick Reference Guide, 2010). As Artemis Financial already has an API, we need to develop the application with full knowledge of that API's behavior and how the application and users will interact with it. Suppose there are any issues with this interaction or problems with the API that will improve security, performance, and reliability. In that case, those issues should be presented to Artemis Financial for their determination and potential resolution.

## References

J. Manico & A. Detlefsen. (2014) Iron-Clad Java: Building Secure Web Applications, McGraw-Hill Education.

MVN Repository. (n.d.) <https://mvnrepository.com/>

National Institute of Standards and Technology: National Vulnerability Database. (n.d.) <https://nvd.nist.gov/vuln/search>

OWASP Secure Coding Practices Quick Reference Guide. (2010). *The OWASP Foundation*. <https://owasp.org/www-pdf-archive/OWASP_SCP_Quick_Reference_Guide_v1.pdf>

Secure Coding Guidelines for Java SE. (2022). <https://www.oracle.com/java/technologies/javase/seccodeguide.html>