****

# 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 5](#_Toc32574613)

[4. Static Testing 5](#_Toc32574614)

[5. Mitigation Plan 11](#_Toc32574615)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **01-25-2024** | **Samantha Durr** | **This is the initial documentation.** |

## 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

Samantha Durr

## Interpreting Client Needs

Artemis Financial wants a web-based application to help them create their personal plans for their client’s financial needs. They require that the application is secure and uses a RESTful API. As a part of this, there are several areas that need to be looked.

Secure communications: since the company provides their own clients with financial plans, there needs to be a level of integrity and confidentiality in the communications and data. Since there can be international communications, this is highly important. Such transactions can add an extra level of security risk to the system, as well as other regulatory requirements.

Government Restrictions: there is an impact from what the government allows in terms of secure communications. With this in consideration, there is an impact to the ability of the company to protect the data and the systems themselves.

External Threats: there are threats currently present. These threats can include phishing, malware, denial-of-service attacks, and ransomware. There are also internal threats as well, which include employees that can cause security breaches both intentionally and unintentionally.

After looking at the levels of communications, restrictions, and threats to the company itself, the next is the modernization of the software that Artemis Financial wants. One of the considerations should be the use of Open-Source libraries. These are often used in modern software. There are vulnerabilities that can be present with the use of open-source libraries, but mostly when they are not properly managed. Since, these are often used in web-based applications, we must look at how they are always evolving. This means it is important to stay up-to-date on the latest versions and developments that are released. This ensures the company has robust and adaptable security.

## Areas of Security

There are 7 main areas of security that need to be taken into account. This ensures that Artemis Financial’s new application is as secure as possible. These are: Input Validation, APIs, Cryptography, Client/Server, Code Error, Code Quality, and Encapsulation.

Input Validation: The importance of this to Artemis Financial is to validate the individuals that are accessing the system. This provides protection to the users and is a way to help ensure that their data remains secure. One of the ways this can be done is to write the validation as strings.

APIs: This is a necessary feature for the application for running both internally and externally. This is especially true when outside users, AKA the customers, are accessing their data. This allows for the data to be accessible and also secure when being transmitted externally.

Cryptography: This is a crucial part that Artemis Financial would need for accessing their system. Since this is about financial data, there needs to be a level of security to ensure that the information is not exposed to the outside world for everyone to see. This means encrypting the data.

Client/Server: This is another layer of protect that provides security between a client machine and a server. This requires specific data, such a private key, for the data to be accessed. Since this is often used over the web, having this feature would allow for data stored on the server to not be accessed, except by the client that has the matching information. These secure communications between the two would help to reduce the risk of just anyone accessing certain data. Use of a VPN would help in this, especially for any Artemis Financial employees that are accessing this data from outside of the main network.

Code Error: Implementing error handling creates an understanding of where the API could have vulnerabilities within the code that needs to be fixed. This helps to close these holes and allow for Artemis Financial to not have to worry about their data being exposed.

Code Quality: The code needs to be at a level of quality that allows Artemis Financial to control access to the different methods based on the user. This would be a way of ensuring that users do not have access to data that they shouldn’t have. An example of this is non-systems administrators not having access to user passwords.

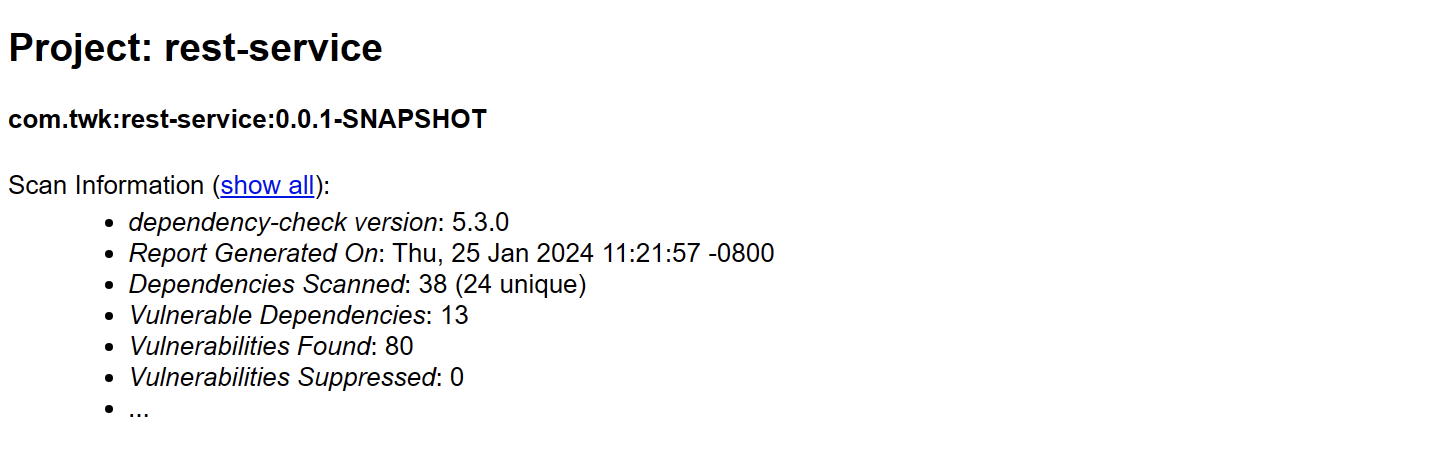
Encapsulation: Encapsulating the code and the data allows for a more secure means of keeping the data protected. As with the code quality allowing for writing high quality code, using encapsulation allows for limiting what has access to different methods and data. Encapsulating the different elements means that data is only accessed by the necessary methods and not by any part of the program that should not. An example would be calculating monthly budgets. That data should not be accessed by the change password component.

## Manual Review

In a manual review of the code, there are two files that are important to look through, the POM.XML and the Greeting Controller. From the POM.XML file, I needed to verify that there was an Apache Validator within the code. From the Greeting Controller, I wanted to find code for input validation. I did not find this code. This is something to make note of, as this is likely to be an important aspect needed by Artemis Financial. The quality of the code looked to be at an acceptable level, though error handling was missing and should be added.

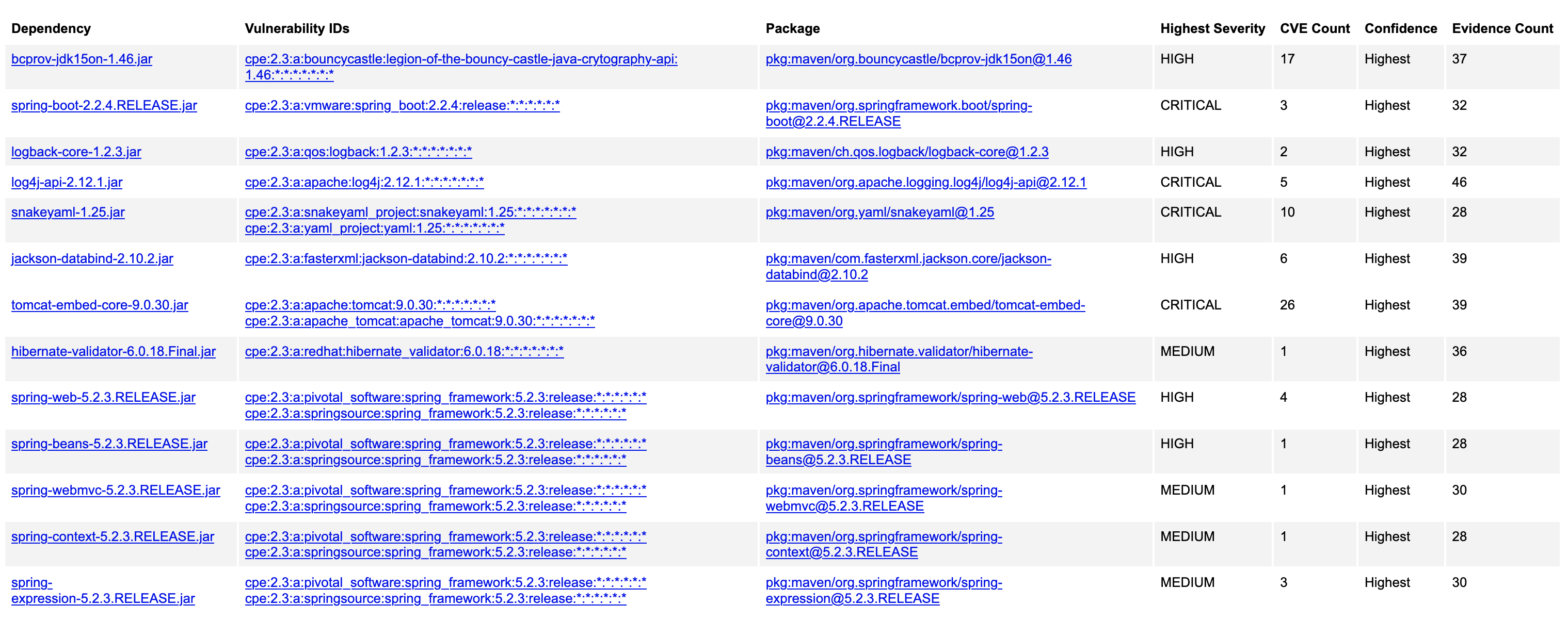
Next was checking the API. The user input was exposed, as it wasn’t written through POST. This exposes the user’s data as well. This also relates into client/server interactions as there was an exposure of the user data with this lack of post. Finally, there didn’t appear to be any form of cryptography taking places. These are all areas that I took note of to improve and add to.

## Static Testing



(screenshot of dependency report Scan Info, 2024)

There were 13 dependencies identified using the version of the dependency-check that was built into the POM.XML file. Here is a list of the following Dependencies and their vulnerability IDs:



(Screenshot of Dependency Report Dependency List, 2024)

For the dependencies, there are specific vulnerability codes attached to each. All of these codes were identified by the National Vulnerability Database. What follows is the dependency and the specific vulnerability IDs that belong to them:

**Bcprov-jdk15on-1.46.jar**

* CVE-2016-1000354: This vulnerability was removed by the provider as unsafe allowed for “the use of ECB mode” (*National Vulnerability Database,* 2024).
  + Recommended solution is to update to a version after 1.55 of this dependency.
* CVE-2016-1000346: This vulnerability did not validate DH public keys.
  + Recommended solution would be to update to a newer version, at least 1.56, as this has been fixed in those versions.
* CVE-2016-1000345: This vulnerability left the system open to padding oracle attacks.
  + Recommended solution: update to a version past 1.55, as that version and older are vulnerable.
* CVE-2016-1000344: In versions 1.55 and earlier, this allowed for “implementation allowed the use of ECB mode” (*National Vulnerability Database,* 2024).
  + Recommended action is to remove this version, as it is considered unsafe.
* CVE-2016-1000343: This vulnerability would generate weak private keys.
  + Recommended action is that in version before, and including, 1.55 by “explicitly passing parameters to the key pair generator” (*National Vulnerability Database,* 2024).
* CVE-2016-1000342: This version did not completely valid ASN.1 encoding. This allowed for injection into the sequence.
  + Recommended action is to update to a version after 1.55.
* CVE-2016-1000341: This vulnerability allowed for timing attacks in the DSA signature.
  + Recommended action is to update to a version after 1.55.
* CVE-2016-1000339: This vulnerability used an AES engine that allowed for leaks in the key from monitoring the lookup table.
  + Recommended action is to update the AES to the base engine instead of the fast engine that is used in this version.
* CVE-2016-1000338: This version does not validate the DSA’s ASN.1 encoding. This allows for injection attacks.
  + Recommended action is to go to a version after 1.55.
* CVE-2018-5382: This vulnerability used BKS Keystore that only went to 16bits and allowed for an attacker to compromise the key.
  + Recommended action is to update to at least 1.47, as this was dealt with in that version.
* CVE-2017-13098: This vulnerability provided a weak Bleichenbacher oracle. This allowed attackers to access a private key.
  + Recommended action is to update to a more current version.
* CVE-20123-1624: This vulnerability is that the library does not handle side-channel attacks properly.
  + Recommended action is to update to at least 1.48.

**Spring-boot-2.2.4.RELEASE.jar**

* CVE-2023-20883: This version of the spring boot is no longer supported and can allow for denial-of-service attacks.
  + Recommended action is to update to a current version.
* CVE-2023-20873: This version “could be susceptible to a security bypass” (*National Vulnerability Database,* 2024).
  + Recommended action is to update to a current version.
* CVE-2022-27772: This version is unsupported and showed to allow directory hijacking.
  + Recommended action is to update to a current version.

**Logback-core-1.2.3.jar**

* CVE-2023-6378: published in November of 2023, this showed a vulnerability that allowed for “Denial-Of-Service attack[s] by sending poisoned data” (*National Vulnerability Database,* 2024).
  + Recommended action is to update to a newer version.
* CVE-2021-42550: This allowed for attackers to change any configurations, if they had the right privileges, and “craft a malicious configuration” (*National Vulnerability Database,* 2024).
  + Recommended Action would be to update to a newer version.

**Log4j-api-2.12.1.jar**

* CVE-2021-44832: This allowed for remote code execution that let attackers control their target server.
  + Recommended action is to update to at least version 2.17.1.
* CVE-2021-45105: This vulnerability allowed for “uncontrolled recursion from self-referential lookups” (*National Vulnerability Database,* 2024).
  + Recommended solution is to update to at least 2.17.0.
* CVE-2021-45046: This vulnerability “was incomplete in certain non-default configurations” (*National Vulnerability Database,* 2024). This allowed for attackers to take control of MDC.
  + Recommended action is to update to at least 2.16.0.
* CVE-2021-44228: This vulnerability did not stop LDAP and JNDI controlled attacks.
  + Recommended action is to update to at least 2.16.0.
* CVE-2020-9488: The vulnerability did not properly validate certificates. This allowed for SMTPS connections.
  + Recommended action is to update to at least 2.12.3.

**Snakeyaml-1.25.jar**

* CVE-2022-1471: This vulnerability does not restrict types. This allows for instantiated types.
  + Recommended action is to update to at least version 2.0
* CVE-2022-41854: This vulnerability allows for parsing untrusted YAML files and can allow for denial-of-service-attacks.
  + Recommended action is to not use snakeYAML.
* CVE-2022-38752: This vulnerability is similar to the previous one in that it allows for parsing untrusted YAML files.
  + Recommended action is to not use snakeYAML.
* CVE-2022-38751: This vulnerability also allows for parsing untrusted YAMLs.
  + Recommended action is to not use snakeYAML.
* CVE-2022-38750: As before, this allows for parsing untrusted YAML files.
  + Recommended action is to not use snakeYAML.
* CVE-2022-38749: As before, this allows for parsing untrusted YAML files.
  + Recommended action is to not use snakeYAML.
* CVE-2022-25857: This vulnerability is that snakeYAML has a vulnerability in regards to Denial-of-service-attacks from “missing to nested depth limitation for collections” (*National Vulnerability Database,* 2024).
  + Recommended action is to not use snakeYAML.
* CVE-2017-18640: This vulnerability “entity expansion during a load operation, a related issue to CVE-2003-1564” (*National Vulnerability Database,* 2024).
  + Recommended action is to not use snakeYAML.
* CVE-2022-3064: This allowed for “parsing malicious or large YAML documents” (*National Vulnerability Database,* 2024).
  + Recommended action is to not use snakeYAML.
* CVE-2021-4235: This allowed for malicious YAML files to use up resources on servers.
  + Recommended action is to not use snakeYAML.

**Jackson-databind-2.10.2.jar**

* CVE-2023-35116: This allowed for denial-of-service-attacks through cyclical dependencies.
  + Recommended action is to not use this version if possible as the company does not feel this is a valid issue.
* CVE-2021-46877: This allowed for denial-of-service-attacks through JsonNode serialization.
  + Recommended action is to update to a version after 2.12.6.
* CVE-2022-42004: This was caused by “a check in BeanDeserializer.\_deserializeFromArray to prevent use of deeply nested arrays” (*National Vulnerability Database,* 2024). This allowed for a vulnerability in certain deserialization.
  + Recommended action is to update to a version after 2.13.4.
* CVE-2022-42003: This vulnerability allowed for deserializers from the lack of checks.
  + Recommended action is to update to a version after 2.13.4.1.
* CVE-2020-36518: This allowed for denial-of-service-attacks from nested objects through the Java StackOverflow exception.
  + Recommended action is to update to a version after 2.13.0.
* CVE-2020-25649: This vulnerability “did not have entity expansion secured properly” (*National Vulnerability Database,* 2024). This allowed for XML external entity attacks.
  + Recommended action is to not use FasterXML Jackson Databind.

**Tomcat-embed-core-9.0.30.jar**

CVE-2024-21733: This vulnerability causes error messages that can contain sensitive information.

* + Recommended action is to use version 9.0.44.
* CVE-2023-46589: This vulnerability does not have correct parsing of HTTP trailers.
  + Recommended action is to update to at least 9.0.83.
* CVE-2023-45648: This vulnerability had improper validation on the Apache Tomcat. It would allow people to smuggle in requests.
  + Recommended action to is at least update to 9.0.81.
* CVE-2023-42795: This vulnerability had an incomplete cleanup and would skip parts of recycling. This could lead to information leaks.
  + Recommended action is to update to at least 9.0.81.
* CVE-2023-44487: This allowed for denial-of-service-attacks on the HTTP/2 protocol from cancellation of requestions.
  + Recommended actions are unavailable at the current time.
* CVE-2023-41080: This vulnerability had an issue with redirection of untrusted sites. Limited to root applications.
  + Recommended action is to update to not use root applications.
* CVE-2023-28708: This vulnerability did not have a security attribute for reverse proxy.
  + Recommended action is to now allow reverse proxy.
* CVE-2022-42252: This vulnerability allowed for ignoring invalid HTTP headers and not reject the requests. This allowed for smuggling attacks.
  + Recommended action is to ensure validation of all headers and reject those that are invalid.
* CVE-2021-43980: This vulnerability related to Apache Tomcat in that it allowed clients to share HTTP11Processor instance. Requests could go to the wrong client.
  + Recommended action is to not use Tomcat.
* CVE-2022-34305: “[T]he Form authentication example in the examples web application displayed user provided data without filtering, exposing a XSS vulnerability” (*National Vulnerability Database,* 2024).
  + Recommended action is to update the form authentication to not do this.
* CVE-2022-29885: This vulnerability incorrectly allowed clustering over untrusted networks. This allowed for all types of risks.
  + Recommended action is to add code to ensure that only trusted networks are allowed.
* CVE-2021-41079: This vulnerability did not properly validate TLS packets. This could cause infinite loop denial-of-service-attacks.
  + Recommended action is to add proper validation.
* CVE-2021-33037: This vulnerability allowed for incorrect parsing of HTTP encoding requests. This allowed for smuggling attacks.
  + Recommended fix is to update to a newer version of the API.
* CVE-2021-30640: This vulnerability allowed attackers to bypass protections using valid users.
  + Recommended actions is to not use Tomcat.
* CVE-2021-25329: This vulnerability relates to an incomplete fix of CVE-2020-9484.
  + Recommended action is to not use Tomcat.
* CVE-2021-25122: Tomcat allows for duplicate header requests and allow users to get the wrong one.
  + Recommended action is to validate user headers.
* CVE-2021-24122: This affects NTFS file systems and allowed for susceptibility of JSP source code and cause inconsistent behavior.
  + Recommended action is to update to a newer version.
* CVE-2020-17527: The system reuses HTTP header values and cause connection closure.
  + Recommended action is to update to a more current version.
* CVE-2020-13943: This allowed for exceeding number of concurrent connections in an HTTP/2 client.
  + Recommended action is to update to a newer version.
* CVE-2020-13935: This vulnerability did not properly validate payload length in the WebSocket frame.
  + Recommended action is to update to a newer version.
* CVE-2020-13934: this vulnerability did not release the process of HTTP/1 after updating to HTTP/2. Can lead to denial-of-service-attacks.
  + Recommended action is to update to a newer version.
* CVE-2020-8022: This had an incorrect default permission that allowed for issues with Linux machines.
  + Recommended action is to update to a newer version.
* CVE-2020-11996: This allowed for triggering high CPU usage and cause a server to be unresponsive.
  + Recommended action is to update to a newer version.
* CVE-2020-9484: This vulnerability allowed someone to attack the server that controls contents and make the PersistanceManager configure null.
  + Recommended action is to update to a newer version that handles this or disable PersistanceManager.
* CVE-2020-1938: This allows for a higher trust in AJP connections and allows for exploiting this trust.
  + Recommended action is to update to a newer version, at least 9.0.31.
* CVE-2020-1935: Uses an end of line approach in validation. This can lead to smuggling attacks.
  + Recommended action is to update to the latest version of Tomcat.
* CVE-2019-17569: This is caused by a refactoring in the regression and allowed for invalid transfer-encoding that could be handled incorrectly.
  + Recommended action is to update to the latest version.

**Hibernate-validator-6.0.18.Final.jar**

* CVE-2020-10693: This is a flaw that allowed for invalid EL expressioned to be enabled and let attackers bypass input sanitation.
  + Recommended action is to not use this feature or to update to a newer version if available.

**Spring-web-5.2.3.RELEASE.jar**

* CVE-2016-1000027: This shows an issue with remote code execution that cause untrusted data to be used. The vendor does not seem to see an issue with this.
  + Recommended action is to possibly not use Spring Boot.
* CVE-2020-5421: This vulnerability is due to the ability to bypass protections from RFD attacks.
  + The recommended action is to update to the latest version.
* CVE-2021-22096: This allowed for users to input malicious content to the entry logs.
  + Recommended action is to update to the latest version.
* CVE-2021-22118: This allowed for escalation of temporary storage.
  + The recommended action is to update to the latest version.

**Spring-beans-5.2.3.RELEASE.jar**

* CVE-2022-22965: This allowed for remote code execution through data binding.
  + Recommended action is to not allow data binding and to update to the latest version.

**Spring-webmvc-5.2.3.RELEASE.jar**

* CVE-2021-22060: This allowed for users to insert data into the log entries.
  + Recommended action is to update to the newest version.

**Spring-context-5.2.3.RELEASE.jar**

* CVE-2022-22968: This vulnerability allowed for protections on files to not be as effective from disallowedFIelds in the DataBinder.
  + Recommended action is to update to the latest version.

**Spring-expressions-5.2.3.RELEASE.jar**

* CVE-2022-22950: This vulnerability allows a user to craft an SPEL expression that can cause denial-of-service-attacks.
  + Recommended action is to update to the latest version.
* CVE-2023-20861: This vulnerability is the same as CVE-2022-22950.
  + Recommended Action is to possibly not use Spring Framework.
* CVE-2023-20863: This is the same as CVE-2022-20861.
  + Recommended action is to not use Spring Framework.

The attribution to all of these are identified based upon the National Vulnerability Database and the Common Vulnerabilities and Exposures system. The dependency check tool uses these to cross reference this information. To enhance security for Artemis Financial, the recommended solutions after each vulnerability should be applied and regular static testing conducted.

## Mitigation Plan

From the results of looking through the manual review and the static testing, updating to the current versions of the different dependencies is highly recommended. This will fix the majority of these issues. When it comes to minimizing the validation errors, it is best to add extra code to help the validation. This is a must in mitigating issues that may arise.

Another part of mitigating the vulnerabilities is to research and switch from using snakeyaml, as the majority of the vulnerabilities found do not appear to be resolved. One option is to not use yaml at all, but the current version of an alternative called YamlBeans currently does not show any vulnerabilities. Here is the repository link from Maven: <https://mvnrepository.com/artifact/com.esotericsoftware.yamlbeans/yamlbeans>.

**Resources**

Aswani, R. (2023, October 2). *Keep Data Safe: 2023 Guide to Secure Communication Systems*. Kumospace. https://www.kumospace.com/blog/secure-communication

Lim, N. (2020, February 27). *What is a global data security standard?*. AppDirect. https://www.appdirect.com/blog/what-are-standardized-international-data-security-measures-and-how-would-they-work

IT Governance USA. (2024). *Federal Cybersecurity and Data Privacy Laws Directory.* IT Governance USA.

<https://www.itgovernanceusa.com/federal-cybersecurity-and-privacy-laws>

Kimpel, H. (2023, June 19). *Mitigate hidden security risks of open source libraries*. New Relic. https://newrelic.com/blog/how-to-relic/mitigate-open-source-library-security-risks

Durr, S. (2024) *Screenshot of Dependency Report Scan Info* [Photograph].

Durr, S. (2024) *Screenshot of Dependency Report Dependency List* [Photograph].

NVD. (2024). *National Vulnerability Database*. NVD. https://nvd.nist.gov

Common Vulnerabilities and Exposures (2024). *CVE*. CVE. https://cve.mitre.org/

Vermeer, B. (2021, March 30). *Preventing YAML parsing vulnerabilities with snakeyaml in Java*. Snyk. https://snyk.io/blog/java-yaml-parser-with-snakeyaml/