****

# 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** | **1/22/23** | **Justin Lombardi** |  |

## 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

Justin Lombardi

## Interpreting Client Needs

Our client, Artemis Financial, needs a team that can modernize their operations. They create custom financial plans for their customers which include savings, retirement, investments, and insurance. Being that this is a financial institution, there is a lot of personal/sensitive data in scope that could be enticing to attackers. External threats could potentially have the ability of going after personal information and personal bank information for all customers associated with Artemis Financial. The value of secure communication is held high due to their customer’s personal/sensitive data that is in scope. There is also the potential for international transactions to be in scope so Global Rain must be aware of the different laws revolving around security in different jurisdictions. This includes any government regulations and even falls into the cooperation of the RFC 5246. For any threat that is current or in the immediate future, potential attackers could invade and obtain personal data if the API in question is not secure. If it is not secure, this could cause leaks of information and allow attackers to penetrate the API and steal personal data. Some modern ways to consider in protect this information could be the use of Two-Factor Authentication to prevent suspicious login attempts and ensuring we use HTTPS with the appropriate certificates and authentication to protect the transfer of data.

## Areas of Security with Justification

**Input Validation –** If Artemis Financial is having their clients sign in to view their accounts or make decisions, we will need to authenticate and validate the input given to ensure we protect the client’s personal data. This could also be validating whether a client is sure they want to move forward with their decision or if the information they input is all correct with a reCAPTCHA of sorts.

**APIs –** Since Artemis Financial has a RESTful API, there needs to be the absolute positivity that the API is ready to prevent breaches of any kind. We must ensure that there is also an authentication process built into the API itself to prevent potential attacks from external threats.

**Cryptography –** With the highly sensitive data that Artemis Financial holds, encrypting the data via hashing would be a great way to make the sensitive data harder to access and decode from external attackers.

**Client/Server –** Protecting the connection between the client server can be some of the best lines of defense. With features like the correct licensing and headers in our HTTPS requests, we can better handle and better secure the data that is being transferred to the correct storage and server.

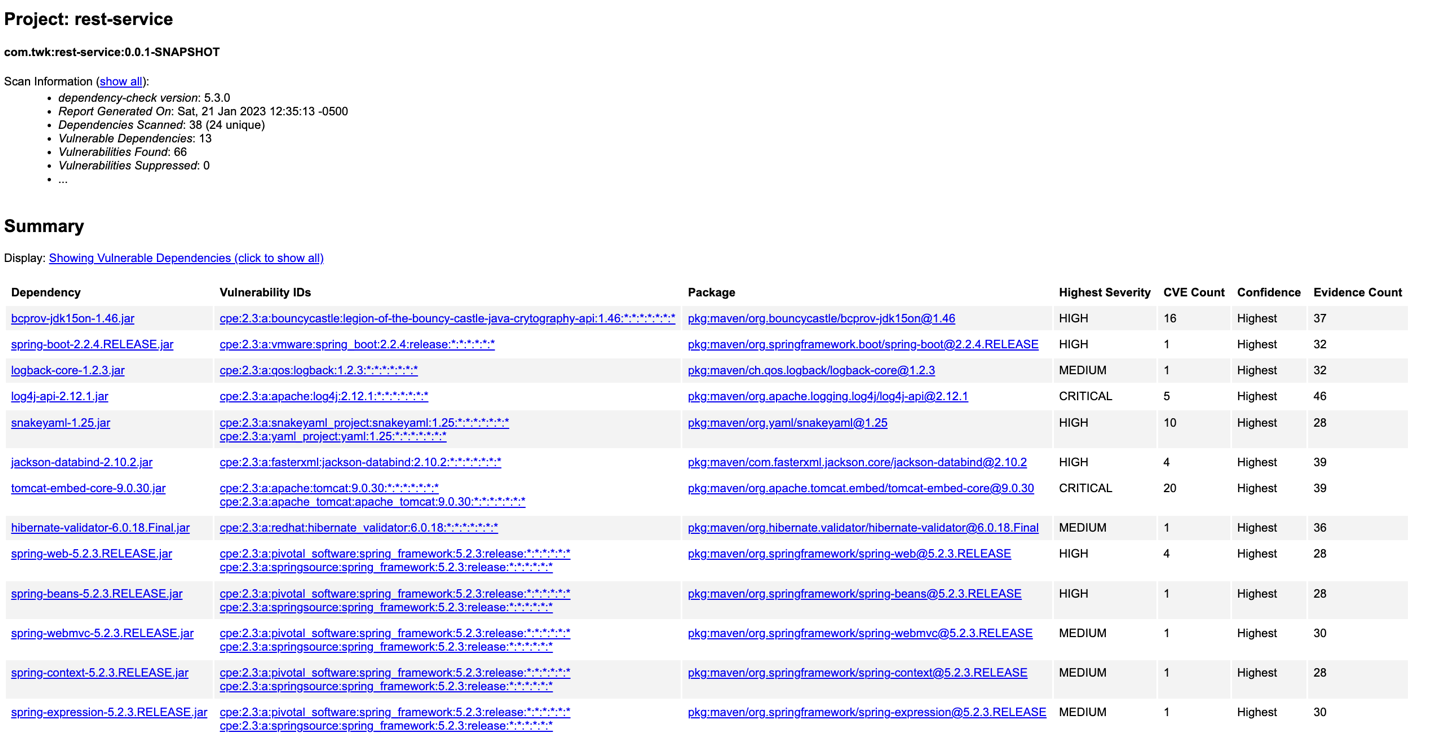
**Code Error –** Handling code errors correctly will be able to ensure that the clients of Artemis Financial or attackers to Artemis Financial cannot proceed with actions if the specific authentications or validation are not met. This could include locked accounts due to too many sign-in attempts. This could be error handling stack tracing as well.

## Manual Review

There are a few areas that seem to raise some red flags regarding what we pointed out in the Client Needs and Areas of Security:

* I do not see any code that requests validation for the CRUD.java and CRUDController.java files. This could be problematic because we are not validating and securing our requests.
* In the CRUDController.java file, I noticed we are returning the business name value which could reveal sensitive customer data without any encryption or validation.
* In the DocData.java file, we see the implementation of the of the read method. The connection is from a local server it seems with no HTTPS protection. It also will show the database name, username, and password in the Stack Trace without encryption.
* There is no sign of validation, encryption, or authentication when dealing with requests.

## Static Testing



[bcprov-jdk15on-1.46.jar](#l2_991c96a4e31e6c19e2b9136c8955bd423f2d)

1. The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms. This jar contains JCE provider and lightweight API for the Bouncy Castle Cryptography APIs for JDK 1.5 to JDK 1.7.
2. It is recommended to halt the server by stopping the JBoss Application  
   Server process before installing this update, and then after installing the  
   update, restart the server by starting the JBoss Application Server process. (https://access.redhat.com/errata/RHSA-2014:0371.html)
3. [**CVE-2013-1624**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-1624)**, CVE-2015-7940,** [**CVE-2016-1000338**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000338)**,** [**CVE-2016-1000339**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000339)**,** [**CVE-2016-1000341**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000341)**,** [**CVE-2016-1000342**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000342)**,** [**CVE-2016-1000343**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000343)**,** [**CVE-2016-1000344**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000344)**,** [**CVE-2016-1000345**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000345)**,** [**CVE-2016-1000346**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000346)**,** [**CVE-2016-1000352**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000352)**,** [**CVE-2017-13098**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-13098)**,** [**CVE-2018-5382**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2018-5382)**, CVE-2020-0187, CVE-2020-26939**

[spring-boot-2.2.4.RELEASE.jar](#l3_225a4fd31156c254e3bb92adb42ee8c6de81)

1. \*\* UNSUPPORTED WHEN ASSIGNED \*\* spring-boot versions prior to version v2.2.11.RELEASE was vulnerable to temporary directory hijacking.
2. This vulnerability impacted the org.springframework.boot.web.server.AbstractConfigurableWebServerFactory.createTempDir method. NOTE: This vulnerability only affects products and/or versions that are no longer supported by the maintainer.
3. [**CVE-2022-27772**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-27772)

[log4j-api-2.12.1.jar](x-apple-ql-id2://D97A37E7-C2B2-4796-94BE-4DAE0C56EE31/x-apple-ql-magic/#l9_a55e6d987f50a515c9260b0451b4fa217dc5)

1. Improper validation of certificate with host mismatch in Apache Log4j SMTP appender.
2. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender. Fixed in Apache Log4j 2.12.3 and 2.13.1
3. [**CVE-2020-9488**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9488)**,** [**CVE-2021-44228**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-44228)**,** [**CVE-2021-44832**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-44832)**,** [**CVE-2021-45046**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-45046)**,** [**CVE-2021-45105**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-45105)

[logback-core-1.2.3.jar](#l4_864344400c3d4d92dfeb0a305dc87d953677)

1. In logback version 1.2.7 and prior versions, an attacker with the required privileges to edit configurations files could craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers.
2. Updating to the latest versions should help to prevent any attacks
3. [**CVE-2021-42550**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-42550)

[snakeyaml-1.25.jar](#l8_8b6e01ef661d8378ae6dd7b511a7f2a33fae)

1. SnakeYaml's Constructor() class does not restrict types which can be instantiated during deserialization. Deserializing yaml content provided by an attacker can lead to remote code execution.
2. We recommend using SnakeYaml's SafeConsturctor when parsing untrusted content to restrict deserialization.
3. [**CVE-2017-18640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-18640)**,** [**CVE-2021-4235**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-4235)**,** [**CVE-2022-25857**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-25857)**,** [**CVE-2022-3064**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-3064)**,** [**CVE-2022-38749**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-38749)**,** [**CVE-2022-38750**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-38750)**,** [**CVE-2022-38751**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-38751)**,** [**CVE-2022-38752**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-38752)**,** [**CVE-2022-41854**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-41854)

[spring-web-5.2.3.RELEASE.jar](x-apple-ql-id2://D97A37E7-C2B2-4796-94BE-4DAE0C56EE31/x-apple-ql-magic/" \l "l18_dd386a02e40b915ab400a3bf9f586d2dc4c)

1. Pivotal Spring Framework through 5.3.16 suffers from a potential remote code execution (RCE) issue if used for Java deserialization of untrusted data. Depending on how the library is implemented within a product, this issue may or not occur, and authentication may be required. NOTE: the vendor's position is that untrusted data is not an intended use case. The product's behavior will not be changed because some users rely on deserialization of trusted data.
2. To reduce the likelihood of code injection, use stringent allowlists that limit which constructs are allowed.
3. **CVE-2016-1000027, CVE-2020-5421, CVE-2021-22096, CVE-2021-22118**

[spring-webmvc-5.2.3.RELEASE.jar](x-apple-ql-id2://D97A37E7-C2B2-4796-94BE-4DAE0C56EE31/x-apple-ql-magic/" \l "l19_745a62502023d2496b565b7fe102bb1ee22)

1. A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. The specific exploit requires the application to run on Tomcat as a WAR deployment. If the application is deployed as a Spring Boot executable jar, i.e. the default, it is not vulnerable to the exploit. However, the nature of the vulnerability is more general, and there may be other ways to exploit it.
2. To reduce the likelihood of code injection, use stringent allowlists that limit which constructs are allowed.
3. **CVE-2021-22060**

[tomcat-embed-core-9.0.30.jar](x-apple-ql-id2://D97A37E7-C2B2-4796-94BE-4DAE0C56EE31/x-apple-ql-magic/" \l "l20_ad32909314fe2ba02cec036434c0addd19b)

1. When using the Apache JServ Protocol (AJP), care must be taken when trusting incoming connections to Apache Tomcat. Tomcat treats 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 in ways that may be surprising. In Apache Tomcat 9.0.0.M1 to 9.0.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99, Tomcat shipped with an AJP Connector enabled by default that listened on all configured IP addresses. It was expected (and recommended in the security guide) that this Connector would be disabled if not required. This vulnerability report identified a mechanism that allowed: - returning arbitrary files from anywhere in the web application - processing any file in the web application as a JSP Further, if the web application allowed file upload and stored those files within the web application (or the attacker was able to control the content of the web application by some other means) then this, along with the ability to process a file as a JSP, made remote code execution possible. It is important to note that mitigation is only required if an AJP port is accessible to untrusted users. Users wishing to take a defense-in-depth approach and block the vector that permits returning arbitrary files and execution as JSP may upgrade to Apache Tomcat 9.0.31, 8.5.51 or 7.0.100 or later. A few changes were made to the default AJP Connector configuration in 9.0.31 to harden the default configuration. It is likely that users upgrading to 9.0.31, 8.5.51 or 7.0.100 or later will need to make small changes to their configurations.
2. Make sure to update tomcat7packages.
3. **[CVE-2019-17569](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2019-17569" \t "_blank),** [**CVE-2020-11996**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-11996)**,** [**CVE-2020-13934**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13934)**,** [**CVE-2020-13935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13935)**,** [**CVE-2020-13943**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13943)**,** [**CVE-2020-17527**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-17527)**,** [**CVE-2020-1935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1935)**,** [**CVE-2020-1938**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1938)**,** [**CVE-2020-8022**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-8022)**,** [**CVE-2020-9484**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9484)**,** [**CVE-2021-24122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-24122)**,** [**CVE-2021-25122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25122)**,** [**CVE-2021-25329**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25329)**,** [**CVE-2021-30640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-30640)**,** [**CVE-2021-33037**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-33037)**,** [**CVE-2021-41079**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-41079)**,** [**CVE-2021-43980**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-43980)**,** [**CVE-2022-29885**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-29885)**,** [**CVE-2022-34305**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-34305)**,** [**CVE-2022-42252**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-42252)

[hibernate-validator-6.0.18.Final.jar](#l16_7fd00bcd87e14b6ba66279282ef15efa30d)

1. 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.
2. You can pass user input as an expression variable by unwrapping the context to HibernateConstraintValidatorContext. Please refer to the <https://in.relation.to/2020/05/07/hibernate-validator-615-6020-released/> and [https://docs.jboss.org/hibernate/stable/validator/reference/en-US/html\_single/#\_the\_code\_constraintvalidatorcontext\_code](https://docs.jboss.org/hibernate/stable/validator/reference/en-US/html_single/).
3. [**CVE-2020-10693**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-10693)

[jackson-databind-2.10.2.jar](#l9_0528de95f198afafbcfb0c09d2e43b6e0ea6)

1. A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity.
2. Applying a package update could be a solution for this specific issue.
3. [**CVE-2020-25649**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-25649)**,** [**CVE-2020-36518**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-36518)**,** [**CVE-2022-42003**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-42003)**,** [**CVE-2022-42004**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-42004)

## Mitigation Plan

The mitigation plan for Artemis Financial will include the following:

1. It is highly recommended to update all dependencies from our dependency-check report from above.
2. Be sure to implement some form of encryption in the DocData.java, the CRUD.java, and the CRUDController.java to mask vulnerable data that is being transferred from the Server/Client.
3. It would be beneficial to implement a form of two-factor authentication and server validation to ensure proper protection for user authentication.
4. Work on a different implementation of the return values for the business name to protect/encrypt personal data.