****

# 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 6](#_Toc32574612)

[3. Manual Review 7](#_Toc32574613)

[4. Static Testing 8](#_Toc32574614)

[5. Mitigation Plan 10](#_Toc32574615)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **11/10/2022** | **John Brungard** | **This document evaluates the security vulnerabilities that Artemis Financial’s RESTful Web API has or will have as they prepare to modernize their operations. It also introduces mitigation plans to include a code review and dependency assessment.** |

## 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

John Brungard

## Interpreting Client Needs

* What is the value of secure communications to the company?
  + Artemis Financial seeks to have the most current and effective software security for many secure interactions and communications within their system. These involve customers’ personal information, financial plans, savings, retirement, investments, and insurance. Particularly, they want to ensure that they are protected from external and unauthorized threats. This company, like many others, wishes to ensure the security of their client to avoid breaches that could cause loss of money, information, and clientele.
* Does the company make any international transactions?
  + The scenario and code base presented do not admit to any international transactions, but at the same time, do not mention of being purely domestic. Therefore, it would be practical to consider this a possibility and only exclude it if communication from the client says to do so. At the end of Global Rain, this may mean reanalyzing requirements or having a follow-up conversation with a representative from Artemis Financial.
* Are there governmental restrictions about secure communications to consider?
  + “The Gramm-Leach-Bliley Act requires financial institutions – companies that offer consumers financial products or services like loans, financial or investment advice, or insurance – to explain their information-sharing practices to their customers and to safeguard sensitive data.” “Financial institutions covered by the Gramm-Leach-Bliley Act must tell their customers about their information-sharing practices and explain to customers their right to "opt out" if they don't want their information shared with certain third parties.
    - Reference: *Gramm-Leach-Bliley Act.* (2022, February 11). Federal Trade Commission. https://www.ftc.gov/business-guidance/privacy-security/gramm-leach-bliley-act
  + Under the Red Flag’s Rule, a financial institution needs to conduct a risk assessment report to analyze its offering of covered accounts as well as uphold an Identity Theft Prevention Program. This program will find, stop, and solve identity theft in connection to a covered account. The Financial company must also alert customers of communications that did not end up being secure and create mitigation plans pertaining to the situation.
    - Reference: *Code of Federal Regulations*. (2022, November 8). National Archives and Records Administration.<https://www.ecfr.gov/current/title-16/chapter-I/subchapter-F/part-681#681.1>)
  + If the company is related to any management of a client’s credit card, they will have to abide by the PCI-DSS. This includes maintaining a secure network for transactions to occur, encrypting personal information on a public network, and using the most updated anti-virus anti-spyware, and anti-malware software.
    - Reference*: PCI DSS (Payment Card Industry Data Security Standard)*. (2020, May 27). TechTarget SearchSecurity. https://www.techtarget.com/searchsecurity/definition/PCI-DSS-Payment-Card-Industry-Data-Security-Standard
* What external threats might be present now and in the immediate future?
  + One external threat that can be compatible with all companies is the weather. Power outages can cause software problems including loss of data. The weather can also destroy the physical components of the company which can lead to a decline in revenue within the company.
  + Another external threat includes politics. Unconventional political participation, such as riots, can also destroy property. In terms of software, a law or sanction could be introduced that conflicts with the ideology of a group that may cause hackers to specifically target banks or financial institutions.
  + An additional threat that focuses more narrowly on the web-based application itself, is the security misconfiguration involving an associated third-party of the application. This can introduce Malware that can compromise secure communications. Weak principles of security can also introduce injection attacks through input. Moreso, hackers will attempt to use brute force attacks to get into customers account as well as create phishing emails to acquire information they are unauthorized to see.
    - Reference: Berge, A. (2022, September 6). *5 Common Threats to Web Applications and How to Avoid Them*. Geekflare. https://geekflare.com/common-web-application-threats/
* What are the “modernization” requirements that must be considered to include the role of open-source libraries and evolving web application technologies?
  + With the ever-increasing presence of technology and its capabilities in web applications, it is important that applications and their technologies are scalable. This is because new systems are constantly being added or being built upon in evolving web applications. We want to ensure the application can increase their clients without replacing the as-is system.
  + Applications and technologies (including open-source ones) must also be regulatory. This means they must meet the software standards applicable to the code or designs they implement. For example, like previously mentioned, any code involving financial management must create a secure environment on public networks. This will also aid in the security of these technologies.
  + To complement the errors, defects or failures in technologies and applications, we also want maintainability or the ability to repair failures in an efficient manner. Any piece of code is never defect free. Instead, like the principles of scientific testing, it has continuously been proven that defects have not compromised the integrity of the product. Therefore, we need a proactive and reactive solution to when failures become evident.
    - Reference: Harding, R. (2022, January 3*). 5 Application Modernization Considerations*. EQengineered. https://www.eqengineered.com/insights/5-application-modernization-considerations

## Areas of Security

* Think about the functionality of the software application to identify which areas of security apply to Artemis Financial’s web application. Document your findings in your vulnerability assessment report and justify why each area is relevant to the software application.

Based on the scenario and code base, I believe the following areas of security are applicable to Artemis Financial:

* Input Validation
  + The code base does allow input as shown on line 12 of the CRUDController class. When dealing with individualized plans, Artemis Financial may seek to have users login through the web application. When dealing with input, validation through many techniques such as whitelisting and blacklisting may be useful to this application. For these techniques, it may be hard to define what is acceptable without gathering information on what input Financial Artemis depends on using. The same can then be applied to blacklisting. Another useful security mechanism for a web application is creating security to avoid Cross Site Request Forgery. We can lower the chances of this attack by applying synchronizer tokens to sensitive transactions.
* APIs
  + Because the application is intended to run externally to include end user web browsers, an API to define how each user will interact with the application if appropriate. The API will evaluate the methods and data acceptable as well as unacceptable. This may include the use of third-party open-source libraries or software.
* Cryptography
  + Because we are dealing with sensitive data, it is important to encrypt data in transit and at rest. We also need to choose an appropriate cipher mode and algorithm to apply to Artemis Financial. Sensitive data may include passwords which may require encryption keys. Also, depending on if international business will be conducted with Artemis Financial may dictate the encryption techniques used because of domestic and international regulations.
* Code Error
  + When dealing with errors, defects, and/or failures in the system, we want secure error handling to contain the problem, pinpoint its cause(s) and resolve it through appropriate mitigation. Input from the user opens up many more opportunities for errors in the code to occur, especially when it is mutated, accessed, or dependent upon many other components of the system. For this particular situation, we want to put in place proper exceptions and tests to deal with errors in a concise manner.
* Code Quality
  + When dealing with complex systems and many other people working towards the same project, it is important that proper conventions and patterns set forth by procedure and regulations are followed. Particularly, because Artemis Financial deals with sensitive information of their clients, they must ensure that the code written gives the correct amount of privilege and privacy to each user and their respective data.

## Manual Review

* Identify all vulnerabilities in the Project One Code Base, linked in Supporting Materials, by manually inspecting the code. Document your findings in your vulnerability assessment report. Be sure to include a description that identifies where the vulnerabilities are found (specific class file, if applicable).

Upon review of the code base, the following vulnerabilities were identified:

* As mentioned before, the CRUDcontroller class expects a String on line 12, which does not have any validation used for the input. One specific technique that can be implemented is providing a minimum/maximum length on the string passed in the CRUDController class. This will lower the chances of user input causing harm to the system. Also as mentioned, we could define the acceptable characters for input and reject the rest when more information is gathered on the use of input within the system.
* Additionally, we specified the use of cryptography in the application due to the handling of sensitive data. However, I did not see any mention or use of encryption methods used within the code base. This will be important to incorporate to protect private information and abide by regulations based on the range at which Artemis Financial plans to operate.
* When checking for secure error handling, I did find a try and catch block for reading documents within the code base. From what is within the code base, it appears like this code attempts to connect the user to the database using their credentials identified as “root”. If caught by an SQL exception, it displays a printStackTrace() method to diagnose the exception by displaying where and why the error occurred (though many argue this is not thread safe). However, as the project continues to expand, it will be crucial to include safeguards on other sections of the application to include sensitive user information.
* After reviewing the multiple classes within the system, as well as the Jupiter tests that accompany them, it appears that syntax and logic is followed throughout the code. However, the code is not yet finished enough to be utilized and would need updated to fully launch the program. It would need to add the API and previously mentioned suggestions to the code as well as a few other things to reach an excellent status of coding quality.

## Static Testing

* bcprov-jdk15on-1.46.jar
  + Description: 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.
  + Attributes: In Bouncy Castle JCE Provider version 1.55 and earlier the DSA does not fully validate ASN.1 encoding of signature on verification. It is possible to inject extra elements in the sequence making up the signature and still have it validate, which in some cases may allow the introduction of 'invisible' data into a signed structure.
* hibernate-validator-6.0.18.Final.jar
  + Description: Hibernate's Bean Validation (JSR-380) reference implementation.
  + Attributes: 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.
* jackson-databind-2.10.2.jar
  + Description: General data-binding functionality for Jackson: works on core streaming API
  + Attributes: 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.
* log4j-api-2.12.1.jar
  + Description: The Apache Log4j API
  + Attributes: Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. 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
* logback-core-1.2.3.jar
  + Description: logback-core module
  + Attributes: 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.
* snakeyaml-1.25.jar
  + Description: YAML 1.1 parser and emitter for Java
  + Attributes: The Alias feature in SnakeYAML before 1.26 allows entity expansion during a load operation, a related issue to CVE-2003-1564.
* spring-boot-2.2.4.RELEASE.jar
  + Description: Spring Boot
  + Attributes: \*\* UNSUPPORTED WHEN ASSIGNED \*\* spring-boot versions prior to version v2.2.11.RELEASE was vulnerable to temporary directory hijacking. 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.
* spring-core-5.2.3.RELEASE.jar
  + Description: Spring Core
  + Attributes: 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 vulnerability is more general, and there may be other ways to exploit it.
* spring-web-5.2.3.RELEASE.jar
  + Description: Spring Web
  + Attributes: 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.
* tomcat-embed-core-9.0.30.jar
  + Description: Core Tomcat implementation
  + Attributes: Apache Tomcat 10.0.0-M1 to 10.0.6, 9.0.0.M1 to 9.0.46 and 8.5.0 to 8.5.66 did not correctly parse the HTTP transfer-encoding request header in some circumstances leading to the possibility to request smuggling when used with a reverse proxy.
* tomcat-embed-websocket-9.0.30.jar
  + Description: Core Tomcat implementation
  + Attributes: 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 defence-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. Several 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.

## Mitigation Plan

* bcprov-jdk15on-1.46.jar
  + Remediation: Update to version 1.60 or the most recent version.
  + Reference: Red Hat Customer Portal*.* (2018, September 11). *RHSA-2018:2669 - Security Advisory.* Red Hat Security Advisory. https://access.redhat.com/errata/RHSA-2018:2669
* hibernate-validator-6.0.18.Final.jar
  + Remediation: You can pass user input as an expression variable by unwrapping the context to HibernateConstraintValidatorContext. Otherwise, you may upgrade to the current version.
  + Reference: Red Hat Product Security*.* (2021, December 14). *Bug 1805501 (CVE-2020-10693) - CVE-2020-10693 hibernate-validator: Improper input validation in the interpolation of constraint error messages.* Red Hat Bugzilla. https://bugzilla.redhat.com/show\_bug.cgi?id=CVE-2020-10693
* jackson-databind-2.10.2.jar
  + Remediation: This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.
  + Reference: *CVE-2020-25649 Detail.* (2022, July 25).National Vulnerability Database*.* <https://nvd.nist.gov/vuln/detail/CVE-2020-25649>
* log4j-api-2.12.1.jar
  + Remediation: Upgrade to Apache Log4j 2.12.3 or 2.13.1.
  + Reference: *CVE-2020-9488 Detail. .*(2022, May 12).National Vulnerability Database.<https://nvd.nist.gov/vuln/detail/CVE-2020-9488>
* logback-core-1.2.3.jar
  + Remediation: In response to CVE-2021-42550 and LOGBACK-1591 we have decided to make the following steps: 1) we have disabled all JNDI lookup code in logback until further notice. This impacts ContextJNDISelector and <insertFromJNDI> element in configuration files. 2) we have removed all database (JDBC) related code in the project with no replacement.
  + Reference: *Logback News.* (2021, December 14). Logback. <https://logback.qos.ch/news.html>
* snakeyaml-1.25.jar
  + Remediation: Upgrade snakeyaml to a version without CVE-2017-18640 or current version.
  + Reference: *[atlas] 02/02: ATLAS-3940 : Upgrade snakeyaml to a version without CVE-2017-18640*. (2020, September 16). Apache Pony Mail. <https://lists.apache.org/thread/obr7cqy8lk0dk4gp018pplkqnqsxpkff>
* spring-boot-2.2.4.RELEASE.jar
  + Remediation: Upgrade to version v2.2.11.RELEASE or current.
  + Reference: JLLeitschuh. (2022, February 15). *Temporary Directory Hijacking to Local Privilege Escalation Vulnerability in org.springframework.boot:spring-boot*. GitHub. <https://github.com/JLLeitschuh/security-research/security/advisories/GHSA-cm59-pr5q-cw85>
* spring-core-5.2.3.RELEASE.jar
  + Remediation: Users of affected versions should apply the following mitigation: 5.3.x users should upgrade to 5.3.18+, 5.2.x users should upgrade to 5.2.20+. No other steps are necessary.
  + Reference: *CVE-2022-22965: Spring Framework RCE via Data Binding on JDK 9+.* (2021, March 31). VMware. <https://tanzu.vmware.com/security/cve-2022-22965>
* spring-web-5.2.3.RELEASE.jar
  + Remediation: There is no fix for this vulnerability. Java serialization is unsafe, and all we can do is advise against exposing HTTP Invoker endpoints to untrusted clients in our documentation. That we have done almost 5 years ago when this CVE was first published. However, it does not make the vulnerability go away, nor does it change its score.
  + Reference: gauravdeshmukh612. (2020, January 27). *Sonatype vulnerability CVE-2016-1000027 in Spring-web project*.GitHub. <https://github.com/spring-projects/spring-framework/issues/24434#issuecomment-579669626>
* tomcat-embed-core-9.0.30.jar
  + Remediation: The update for tomcat to version 9.0.31 fixes the following issue.
  + Reference: Opens Use Security. (2020, March 15*). [security-announce] openSUSE-SU-2020:0345-1: important: Security update for tomcat*. Opens Use. <https://lists.opensuse.org/archives/list/security-announce@lists.opensuse.org/message/F3FOVKJAK2YR7UVBYBATR7JKLD5IA6WI/>
* tomcat-embed-websocket-9.0.30.jar
  + Remediation: The update for tomcat to version 9.0.31 fixes the following issue.
  + Reference: Opens Use Security. (2020, March 15*). [security-announce] openSUSE-SU-2020:0345-1: important: Security update for tomcat*. Opens Use. <https://lists.opensuse.org/archives/list/security-announce@lists.opensuse.org/message/F3FOVKJAK2YR7UVBYBATR7JKLD5IA6WI/>