****

# 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** | **1/9/2023** | **Ahmad Omar** |  |

## 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

Ahmad Omar

## Interpreting Client Needs

Review the scenario to determine your client’s needs and potential threats and attacks associated with their application and software security requirements.Document your findings in your vulnerability assessment report. Consider the scenario information and the following questions regarding how companies protect against external threats:

* 1. What is the value of secure communications to the company?
  2. Does the company make any international transactions?
  3. Are there governmental restrictions about secure communications to consider?
  4. What external threats might be present now and in the immediate future?
  5. What are the modernization requirements that you must consider? For example:
     1. The role of open-source libraries
     2. Evolving web application technologies

Our client Artemis Financial is a company that creates individualized financial plans for them, the finance plans include savings, retirement, investments, and insurance. Artemis Financial wants to modernize their operations and use a RESTful web API and wanted to make sure that their company is protected from external threats.

The reason that Artemis Financial will value secure communications is that since they deal with customers information that will most likely have sensitive information that will need to be secured since that information can potentially include social security, tax information, addresses, emails, and payment information. This makes secure communication vital for Artemis Financial due to the sensitive information that they host and move around.

Due to missing information, I am not sure where the client is based and what they do transactions on, so I will go with the assumption that they do international transactions and make sure that the information that they potential move around will be secured to international standards.   
  
Due to missing information, we are unable to assume where Artemis Financial is hosted in, if they were in the United States, we would want to make sure to comply with any state and federal laws for secure communications but the most important would be to ensure that no insider trading occurs. If they are in a different country, we would want to ensure that they follow the regulations for secure communications.

The potential external threats that Artemis Financial will deal with a third party accessing its information since they will be hosting financial data of those customers. They will need to make sure that any of the client information is encrypted, and that any information they send is secured as well as encrypted to prevent access due to potential breaches that can accrue.

Modernization that Artemis Financial would consider is making sure that they are on the latest versions for both software and hardware they use which helps to reduce chance of security breaches, having a dedicated IT team to ensure that employees have restricted access in which information they can send and access and this will also in turn help in the first point and having the IT team do regular maintenance check to check for any potential problems. The use of open-source libraries helps with security since multiple people that use the library can investigate the source code, point out, and fix any potential security problems since they are using the library themselves and want it to be as secure as possible the more popular an open-source library the more likely it’s secure.

## Areas of Security

Use what you’ve learned in step 1 and refer to the Vulnerability Assessment Process Flow Diagram provided. Think about the functionality of the software application to identify which areas of security apply to Artemis Financials’ web application. Document your findings in your vulnerability assessment report and justify why each area is relevant to the software application.

**Input Validation:** Since the users will be inputting information to validate themselves, we will want to make sure that the inputs are validated to protect the users, but we can also ensure that no injection attacks can occur by having validated inputs.   
  
**APIs:** Since Artemis Financial is using API calls, we will want to make sure that the API’s are secure and that the API is only able to access certain types of information when being used.   
  
**Cryptography:** We will want cryptography to ensure that we are encrypting the information for Artemis Financial to ensure that any data passed around needs the correct decryption key to access it but also helps in the case of a data breach in which the information is encrypted making it highly unlikely for anyone to access the customer data.

**Code Error:** We will want to have error validation in place to ensure that any errors found can be easily fixed since potential issues with the code can lead to bugs which in turn leads to potential security issues.

## Manual Review

Refer to the seven security areas outlined in the Vulnerability Assessment Process Flow Diagram. Use what you’ve learned in steps 1 and 2 to guide your 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).

I have found a few things that could be an issue even though my Java is rusty looking through the code I found three issues which was the DocData.Java, CRUDController.Java, and the GreetingController.Java.

DocData.Java

I noticed that we are calling the username and password which are set to root which someone can easily guess using a brute force method which then allows them to gain access to the system.

CRUDController.Java & GreetingController.Java

Both of these call for parameters that then are changed to strings which allows someone to manipulate them and gain access to systems that they should not gain access to.

Other

I’ve noticed a few things that we can include to increase security, looking through most of the code we don’t do any encryption which we would want to include to help secure any data we pass through, we don’t have any input validation when look through the code which as I stated before can help with any injection attacks that could be attempted, and I did see a few instances of try catch error handling but not as much which can help secure the software more by giving more information when issues happen in the software we are using.

## Static Testing

Integrate the dependency-check plug-in into Maven by following the instructions outlined in the Integrating the Maven Dependency-Check Plug-in tutorial provided in Supporting Materials. Run a dependency check on Artemis Financial’s software application to identify all security vulnerabilities in the code. Specifically, identify all vulnerabilities in the code base by analyzing results from running the code through a static test.Include these items from the dependency-check report in your vulnerability assessment report:

* 1. The names or vulnerability codes of the known vulnerabilities
  2. A brief description and recommended solutions that are found in the dependency-check report
  3. Attribution (if any) that documents how this vulnerability has been identified or how it was documented in the past
* 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.  
  **License:** Bouncy Castle Licence: <http://www.bouncycastle.org/licence.html>

**Attribution:** 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.   
**Evidence:** [NVD - CVE-2016-1000338 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2016-1000338)

* Hibernate-validator-6.0.18.Final.Jar  
  **Description**: Hibernate's Bean Validation (JSR-380) reference implementation. **License:** http://www.apache.org/licenses/LICENSE-2.0.txt **Attribution:** 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.  
  **Evidence:** [NVD - CVE-2020-10693 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2020-10693)
* Jackson-databind-2.10.2.jar  
  **Description:** General data-binding functionality for Jackson: works on core streaming API **License:** <http://www.apache.org/licenses/LICENSE-2.0.txt>  
  **Attribution:** 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. **Evidence:** [NVD - CVE-2020-25649 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2020-25649)
* Log4j-api-2.12.1.Jar  
  **Description:** The Apache Log4j API **License:** [**https://www.apache.org/licenses/LICENSE-2.0.txt**](https://www.apache.org/licenses/LICENSE-2.0.txt) **Attribution:** 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 **Evidence:** [NVD - CVE-2020-9488 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2020-9488)
* Logback-core-1.2.3.Jar  
  **Description:** logback-core module **License:** http://www.eclipse.org/legal/epl-v10.html, <http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html>  
  **Attribution:** 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. **Evidence:** [NVD - CVE-2021-42550 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2021-42550)
* Snakeyaml-1.25.Jar  
  **Description**: YAML 1.1 parser and emitter for Java  
  **License:** Apache License, Version 2.0: <http://www.apache.org/licenses/LICENSE-2.0.txt>  
  **Attribution:** 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. We recommend using SnakeYaml's SafeConsturctor when parsing untrusted content to restrict deserialization **Evidence:** [[CVE-2022-1471] CWE-502: Deserialization of Untrusted Data - Sonatype OSS Index](https://ossindex.sonatype.org/vulnerability/CVE-2022-1471?component-type=maven&component-name=org.yaml%2Fsnakeyaml&utm_source=dependency-check&utm_medium=integration&utm_content=8.0.1)
* Spring-boot-2.2.4.RELEASE.Jar  
  **Description:** Spring Boot **License:** Apache License, Version 2.0: <https://www.apache.org/licenses/LICENSE-2.0>  
  **Attribution:** \*\* 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.  
  **Evidence:** [NVD - CVE-2022-27772 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2022-27772)
* Spring-boot-starter-web-2.2.4.RELEASE.Jar  
  **Description**: Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container **License:** Apache License, Version 2.0: <https://www.apache.org/licenses/LICENSE-2.0>  
  **Attribution:** \*\* 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. **Evidence:** [NVD - CVE-2022-27772 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2022-27772)
* Spring-core-5.2.3.RELEASE.Jar  
  **Description**: Spring Core **License**: Apache License, Version 2.0: <https://www.apache.org/licenses/LICENSE-2.0>  
  **Attribution:** 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. **Evidence:** [NVD - CVE-2022-22965 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2022-22965)
* Spring-web-5.2.3.RELEASE.Jar  
  **Description**: Spring Web **License:** Apache License, Version 2.0: <https://www.apache.org/licenses/LICENSE-2.0>  
  **Attribution:** 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. **Evidence:** [NVD - CVE-2016-1000027 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2016-1000027)
* Spring-webmvc-5.2.3.RELEASE.Jar

**Description**: Spring Web MVC  
**License**: Apache License, Version 2.0: <https://www.apache.org/licenses/LICENSE-2.0>  
**Attribution:** 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. **Evidence:** [NVD - CVE-2022-22965 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2022-22965)

* Tomcat-embed-core-9.0.30.Jar  
  **Description:** Core Tomcat implementation  
  **License:** Apache License, Version 2.0: <http://www.apache.org/licenses/LICENSE-2.0.txt>  
  **Attribution:** 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. A number of 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. **Evidence:** [NVD - CVE-2020-1938 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2020-1938)
* Tomcat-embed-websocket-9.0.30.Jar  
  **Description:** Core Tomcat implementation **License:** Apache License, Version 2.0: <http://www.apache.org/licenses/LICENSE-2.0.txt>  
  **Attribution:** 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. A number of 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.  
  **Evidence:** [NVD - CVE-2020-1938 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2020-1938)

## Mitigation Plan

There are quite a few things we can do to increase the security for our client Artemis Financial in which the first easy thing we would want to do is update to the latest version of the libraries that I documented in which we had 13 vulnerable dependencies and I noted the name of the library and why it’s a security issue with evidence by the national vulnerability database these are the easiest to fix by making sure we are using the most current version of them.   
  
After updating our dependency versions we will want to address the areas of concern that I noted in the manual review in which I noticed 3 areas in the code that we will want to update due to the security issues these part of the code had by making sure we address those issues we are limiting the chance of a security breach, after that I gave recommendations on how to improve the overall security of the code in which I referenced the areas of security that we would want to include due to the nature of Artemis Financial using API calls.

Another aspect of security that Artemis Financial can pursue that does not deal with the software itself since security is much more than just securing the software, and this is due to the fact that no matter how secure we make the software we also have to account for both people working for Artemis Financial who can fall to social engineering or having outdated hardware that also have security issues. I would recommend that Artemis Financial have a IT team that can ensure that people and hardware are secured in which they can’t fall to any attacks since in today’s world attacks occur on both software, hardware, and people.

By addressing these areas of issues, we can ensure that we are ensuring that our client Artemis Financial has a greater protection in the modern world which is something they will need due to the data they handle for clients in which the greater the security they have allows their customers to have greater peace of mind.