****

# CS 305 Project One

**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** | **3/21/21** | **Danica Hesemann** | **Updated sections 1-5** |

## Client



## Instructions

Deliver this completed vulnerability assessment report, identifying your findings of security vulnerabilities and articulating recommendations for next steps to remedy the issues you have found.

Respond to the five steps outlined below and include your findings. Replace the bracketed text on all pages with your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

Danica Hesemann

## 1. Interpreting Client Needs

Determine your client’s needs and potential threats and attacks associated with their application and software security requirements. Consider the following regarding how companies protect against external threats based on the scenario information:

* What is the value of secure communications to the company?
* Are there any international transactions that the company produces? – unlikely, though maybe some queries or something in some cases
* Are there governmental restrictions about secure communications to consider?
* What external threats might be present now and in the immediate future?
* What are the “modernization” requirements that must be considered, such as the role of open source libraries and evolving web application technologies?
* The company may have access to their customers’ personal information, passwords, bank account information, or even things like legal settlements which need to be protected.
* All employees should not be able to access all customer information and the company will need to comply with any laws or regulations to protect Personally Identifiable Information.
* Due to the amount of sensitive information Artemis Financial may have, open source code should be used with caution as it is more easily exploitable.
* Sensitive data will need to be reliably encrypted.
* Caution should be taken to make sure the web application is accessible on older browsers as well as current ones (within reason), especially as it can be exploited on an outdated browser.

## 2. Areas of Security

Referring to the Vulnerability Assessment Process Flow Diagram, identify which areas of security are applicable to Artemis Financial’s software application. Justify your reasoning for why each area is relevant to the software application.

* Input validation may be relevant for validating passwords or checking any input for invalid characters, invalid number of characters, etc.
* APIs may be relevant because secure API transactions are important to keep information secure during data transfer.
* Cryptography is relevant because sensitive information (passwords, account numbers, etc.) should be stored in an encrypted form.
* Code error is relevant because errors can be exploited. Errors or warnings in the code should be addressed, and proper error handling should be a priority, especially when untrusted data like user input or an invalid parameter may cause an error.
* Code quality is relevant because using secure coding practices and patterns can prevent malicious attacks. For example, proper access control measures and a functional program can give attackers less opportunities to exploit.
* Encapsulation is relevant because using classes with primarily private attributes and public methods can make access control easier, keep administration manageable, and prevent unauthorized users from changing data, making it more secure.

## 3. Manual Review

Continue working through the Vulnerability Assessment Process Flow Diagram. Identify all vulnerabilities in the code base by manually inspecting the code.

* Some input or parameters, such as the name and business name in Greeting/GreetingController and CRUD/CRUDController do not seem to be validated, which may pose a problem if there is any length requirements, characters that may cause errors, etc.
* In DocData, the read\_document method logs in as root, providing full access.
* In the read\_document method (DocData), the password to log in is the same as the username.
* In the read\_document method (DocData), the password does not seem to be encrypted.
* In the myDateTime class, the retrieveDateTime method has the potential to cause a memory leak depending hoe often it is used, as there is no command to delete the data.
* The arguments used in the main method (RestServiceApplication) are not checked before they are used in main, which could cause an error if there are any invalid arguments.

## 4. Static Testing

Run a dependency check on Artemis Financial’s software application to identify all security vulnerabilities in the code. Record the output from dependency check report. Include the following:

1. The names or vulnerability codes of the known vulnerabilities
2. A brief description and recommended solutions provided by the dependency check report
3. Attribution (if any) that documents how this vulnerability has been identified or documented previously

* bcprov-jdk15on-1.46.jar – 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. Unless otherwise stated, vulnerabilities affect version 1.55 and earlier.
  + CVE-2013-1624 – Improper considerations of timing side-channel attacks on a noncompliant MAC check operation during the processing of malformed CBC padding, leaving a vulnerability to attacks. Affects Bouncy Castle 1.48 and earlier.
  + CVE-2016-1000338 – The DSA does not fully validate ASN.1 encoding of signature on verification, allowing invisible data into signed structures.
  + CVE-2016-1000339 – Use of AESFastEngine resulted in a leak. Using AESEngine mitigates the issue.
  + CVE-2016-1000341 – DSA signature generation is vulnerable to timing attack.
  + CVE-2016-1000342 – ECDSA does not fully validate ASN.1 encoding of signature on verification.
  + CVE-2016-1000343 – The DSA key pair generator generates a weak private key if used with default values.
  + CVE-2016-1000344 – The DHIES implementation allowed the use of ECB mode, which is regarded as unsafe.
  + CVE-2016-1000345 – The DHIES/ECIES CBC mode vulnerable to padding oracle attack.
  + CVE-2016-1000346 – The other party DH public key is not fully validated.
  + CVE-2016-1000352 – The ECIES implementation allowed the use of ECB mode, which is regarded as unsafe.
  + CVE-2017-13098 – When configured to use the Java Cryptography Extension for cryptographic functions, BountyCastle TLS provides a weak Bleichenbacher oracle when any TLS cipher suite using RSA key exchange is negotiated. An attacker can recover the private key from a vulnerable application. Resolved in version 1.59.
  + CVE-2018-1000613 – Legion of the Bouncy Castle Java Cryptography APIs 1.58 up to but not including 1.60 contains a CWE-470: Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection') vulnerability. Resolved in version 1.60
  + CVE-2018-5382 – Bouncy Castle BKS version 1 keystore (BKS-V1) files use an HMAC that is only 16 bits long, which can allow an attacker to compromise the integrity of a BKS-V1 keystore
* log4j-api-2.12.1.jar – The Apache Log4j API
  + CVE-2020-9488 – Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. Issue resolved in version 2.13.2.
* snakeyaml-1.25.jar – YAML 1.1 parser and emitter for Java
  + CVE-2017-18640 – The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation. resolved in version 1.26.
* jackson-databind-2.10.2.jar – General data-binding functionality for Jackson works on core streaming API
  + CVE-2020-25649 – FasterXML Jackson Databind did not have entity expansion secured properly, allowing vulnerability to XML external entity (XXE) attacks. Resolved in version 2.11.0.
* tomcat-embed-core-9.0.30.jar – Core Tomcat implementation
  + CVE-2019-17569 - The refactoring present in Apache Tomcat 9.0.28 to 9.0.30, 8.5.48 to 8.5.50 and 7.0.98 to 7.0.99 introduced a regression.
  + CVE-2020-11996 - A specially crafted sequence of HTTP/2 requests sent to Apache Tomcat 10.0.0-M1 to 10.0.0-M5, 9.0.0.M1 to 9.0.35 and 8.5.0 to 8.5.55 could trigger high CPU usage for several seconds.
  + CVE-2020-13934 – An h2c direct connection to Apache Tomcat 10.0.0-M1 to 10.0.0-M6, 9.0.0.M5 to 9.0.36 and 8.5.1 to 8.5.56 did not release the HTTP/1.1 processor after the upgrade to HTTP/2.
  + CVE-2020-13935 - The payload length in a WebSocket frame was not correctly validated in Apache Tomcat 10.0.0-M1 to 10.0.0-M6, 9.0.0.M1 to 9.0.36, 8.5.0 to 8.5.56 and 7.0.27 to 7.0.104, an infinite loop risk.
  + CVE-2020-13943 – If an HTTP/2 client connecting to Apache Tomcat 10.0.0-M1 to 10.0.0-M7, 9.0.0.M1 to 9.0.37 or 8.5.0 to 8.5.57 exceeded the agreed maximum number of concurrent streams for a connection (in violation of the HTTP/2 protocol), it was possible that a subsequent request made on that connection could contain HTTP headers from a previous request rather than the intended headers.
  + CVE-2020-17527 – Apache Tomcat 10.0.0-M1 to 10.0.0-M9, 9.0.0-M1 to 9.0.39 and 8.5.0 to 8.5.59 could re-use an HTTP request header value from the previous stream received on an HTTP/2 connection for the request associated with the subsequent stream, potentially leaking information.
  + CVE-2020-1935 – In Apache Tomcat 9.0.0.M1 to 9.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99 the HTTP header parsing code used an approach to end-of-line parsing that allowed some invalid HTTP headers to be parsed as valid.
  + CVE-2020-1938 - 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, potentially leaving vulnerability to attackers.
  + CVE-2020-8022 – An Incorrect Default Permissions vulnerability allows local attackers to escalate from group tomcat to root.
  + CVE-2020-9484 – When using Apache Tomcat versions 10.0.0-M1 to 10.0.0-M4, 9.0.0.M1 to 9.0.34, 8.5.0 to 8.5.54 and 7.0.0 to 7.0.103, attackers can trigger remote code execution via deserialization of a file under certain circumstances.
  + CVE-2021-24122 – When serving resources from a network location using the NTFS file system, Apache Tomcat versions 10.0.0-M1 to 10.0.0-M9, 9.0.0.M1 to 9.0.39, 8.5.0 to 8.5.59 and 7.0.0 to 7.0.106 were susceptible to JSP source code disclosure in some configurations.
  + CVE-2021-25122 – When responding to new h2c connection requests, Apache Tomcat versions 10.0.0-M1 to 10.0.0, 9.0.0.M1 to 9.0.41 and 8.5.0 to 8.5.61 could duplicate request headers and a limited amount of request body from one request to another.
  + CVE-2021-25329 - When using Apache Tomcat 10.0.0-M1 to 10.0.0, 9.0.0.M1 to 9.0.41, 8.5.0 to 8.5.61 or 7.0.0. to 7.0.107 with a configuration edge case that was highly unlikely to be used, the Tomcat instance was still vulnerable to CVE-2020-9494.
* hibernate-validator-6.0.18.Final.jar – Hibernate's Bean Validation (JSR-380) reference implementation
  + A bug in the message interpolation processor enables invalid EL expressions to be evaluated as if they were valid. Fixed in Red Hat JBoss Enterprise Application Platform 7.3.
* spring-core-5.2.3.RELEASE.jar – Spring Core
  + Protections against RFD attacks may be bypassed depending which browser is used. This can be resolved by updating to the latest version.

## 5. Mitigation Plan

After interpreting your results from the manual review and static testing, identify the steps to remedy the identified security vulnerabilities for Artemis Financial’s software application.

* Ensure that all development tools and software are up to date to mitigate the vulnerabilities discovered in the dependency check.
* Create user roles and/or personal accounts that provide the user with only the necessary access and permissions for their role.
* Incorporate a password encryption algorithm.
* Define password requirements, such as a length requirement and three different types of characters.
* Validate any arguments used in main and output error messages for invalid arguments.
* Include a function in the myDateTime class to delete past data generated by the retrieveDateTime method.
* If necessary, add input validation any time a user provides input to ensure input is not out of bounds and check for invalid length or characters.