

# Artemis Financial Vulnerability Assessment Report

Table of Contents

[Artemis Financial Vulnerability Assessment Report 1](#_Toc1502403144)

[Document Revision History 2](#_Toc1275313266)

[Client 3](#_Toc28607884)

[Developer 3](#_Toc1326978576)

[1. Interpreting Client Needs 4](#_Toc1973723916)

[2. Areas of Security 5](#_Toc1220140369)

[3. Manual Review 6](#_Toc96859661)

[4. Static Testing 7](#_Toc1444069267)

[ bcprov-jdk15on-1.46.jar 7](#_Toc1438373747)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **7/10/2023** | **Zachory Behrensmeyer** | **First implementation of document** |

## Client



## Developer

Zachory Behrensmeyer

## Interpreting Client Needs

1. **What is the value of secure communications to the company?**

First, the value of secure communications comes in the form of protecting yourself from hackers stealing your data and doing bad things with it. In some cases, this data could be personal data of a customer and cause them issues in their personal lives. This can lead to lawsuits and other costs that will hurt the company. Spending a little more on security is worth the while because it will reduce the chances of a data breach, or some kind of cyber-attack. Even the smallest vulnerability could be detrimental to the business. Especially since Artemis Financial is a financial company they will likely have personal details, such as payment information or even Social Security numbers. Additionally, this will affect the company's reputation. If people see that they have been hacked or been through a data breach due to negligence, people will avoid working with the company.

1. **Does the company make any international transactions?**

It does not immediately appear in the code that there are international transactions occurring, however a website if made public can be accessed virtually from anywhere so it seems entirely possible that international transactions could occur. We should be prepared for this.

1. **Are there governmental restrictions about secure communications to consider?**

It does not seem that there is anything out there that is clear cut, and each individual state within the United States makes their own laws, and each country has their own rules. I did find something for the United States that appears to be a blanket coverage to help keep communications secure. The United States has the Federal Information Processing Standards that requires federal agencies and vendors to implement cryptographic modules to protect sensitive information.

1. **What external threats might be present now and in the immediate future?**

There are many ill-willed people and hackers in the world that want sensitive data for many different reasons, to sell, extort the company, or use themselves and so on. Most commonly they will attempt to get this information through phishing emails and hacking into people's domain profiles that have easily cracked passwords. The reason this is a threat is because there is financial and personal data at risk for the business's clientele. Phishing is a form of social engineering to get someone to give up some sensitive data, or to get someone to open a file/link that gives the hacker backdoor access to the network. Password cracking is when a hacker tries to guess a potential password for a domain account. The longer passwords are the harder it is to crack. This leaves the business at risk of a possible civil lawsuit costing more money than it would have to implement higher security to prevent this from happening in the first place. To be less specific there are a lot of things in general that hackers will do if they find they can such as DDOS attacks, DNS attacks, gaining unauthorized access to your network, and other types of injection-based attacks.

1. **What are the modernization requirements that you must consider? For example:**
2. **The role of open-source libraries**
3. **Evolving web application technologies**

It's important to carefully consider using open-source libraries and carefully select them when using them. The problem with open-source libraries is that they will get many updates to enhance functionality and replace bugs. Also, anyone can view the source code. If the IT department of a company does not keep these libraries up to date it can leave them open to many vulnerabilities introducing many new security risks. It is imperative that these libraries are updated regularly. In addition to this the web has become a mainstream source of communication. Most people these days can’t imagine life without the internet. It is important to make sure that secure coding is done. To keep customers safe, we must program secure APIs, encrypt messages, train employees to get them to recognize phishing emails, validate user input, and so on. All these things will deter hackers as it makes it harder for them to hack into your network making it more time costly to hack.

## Areas of Security

**Input Validation** – Input validation is required to make sure that only properly formatted data is entered into the system. This helps guide users to correctly input data into the system as they will incorrectly enter data a lot. This will also work in preventing hackers from entering special characters into the system which can be recognized as code. Input validation will prevent Buffer Overflows, SQL injection attacks, and Cross Site Scripting. If input validation is not implemented, it is easy for hackers to run automated programs to try and get the system to break.

•**APIs**- APIs are used to transfer data. Artemis has very little API functionality but as time goes on, they will likely need to create APIs that transfer financial data and other sensitive data that will need to be protected so that hackers cannot hijack the payload and steal the data. Some common issues related to improperly programmed APIs include Broken Access Control, Broken Authentication Issues, Injection Attacks, Excessive Data Exposure, Issues with Rate Limiting, and Insecure Direct Object References.

•**Cryptography**- Cryptography is the act of encrypting a message so that only the intended recipient can view it. Artemis Financial will want to include this functionality as sensitive data will be included in the messages they are sending. Cryptography is important when communicating between server and client, storing data, and even storing passwords. Encrypting this type of information makes it more difficult for hackers to hijack messages and decrypt the message to get the data.

•**Client/Server**- It is imperative that Artemis Financial has properly implemented client/server security measures as they are storing financial information of the customer. We do not want just anyone to be able to login and access, change, or even withdraw from someone's accounts. If something like this happened, it would damage the company's reputation and likely lead to a large lawsuit. To prevent this, they need to implement certificates and proper authorization to prevent the wrong person from seeing and doing things they should not.

•**Code Error**- Error messages must properly be implemented. If they are not, it is possible that when the error is displayed to the user too much information about your system is given away. For example, detailed error messages can include stack traces, data dumps, and error codes can be displayed. This reveals different aspects about how the system was implemented giving hackers clues to potential flaws. It is also considered bad practice for your users to see error messages like this.  
  
Please note: Not all seven areas of security in the Vulnerability Assessment Process Flow Diagram apply to the company’s software application.

## 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).

After reviewing the code here are some of my findings:

* The DocData class has a method called read\_document(). This method requires two variables key and value. Currently they are unused but when they are implemented there should be validation wrapped around these. Someone could easily pass anything in they wanted, and this could lead to altered flow of a process, arbitrary control, or arbitrary code execution.
* There is a try-catch block in the read\_document() method however it spits out a stack trace which would give an error to the user. This is bad because it gives hints to the person browsing the site that could lead them to finding out how they could perform a SQL injection attack.
* Another issue in the DocData class is it directly calls the jdbc using DriverManager.getConnection method but should be using TLS or SSL. It is not good to use an unencrypted connection between MySQL Server and the application. This can cause data to be exposed when data is being sent over the network. In addition to encrypting the data TLS also offers an identity verification feature.
* The Greeting Controller Class has no input validation. It currently returns a greeting that includes a name that is passed in. As it stands any string can be input into the name variable leaving the site vulnerable to different types of attacks.
* Additionally, in the GreetingController Class there is no error handling so if someone manages to encounter an error it's possible too much information will be displayed on the view.
* As it stands the API section in the greeting controller class is generally underdeveloped and there is a lot of work to be done. There is no encryption that takes place, there are not any HTTP Requests such as GET or POST requests, and there is not any type of Authorization to prevent unwanted visitors from accessing the site's content. If the site is to be used for more than just greetings, they will want to implement this in other APIs.

## Static Testing

### 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.
  + I believe the following issues will be resolved by a version upgrade. We can go all the way to version 1.7, it does not appear any vulnerabilities have been found - <https://mvnrepository.com/artifact/org.bouncycastle/bcprov-jdk15on>
  + A good number of these issues are shown as resolved by updates - <https://access.redhat.com/errata/RHSA-2018:2669>
    - [**CVE-2013-1624**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-1624)
    - **CVE-2015-6644** (OSSINDEX)
      * Calls out android. Not a problem for Artemis.
    - **CVE-2015-7940** (OSSINDEX)
    - [**CVE-2016-1000338**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000338)
    - [**CVE-2016-1000342**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000342)
    - [**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-2016-1000343**](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** (OSSINDEX)
    - **CVE-2020-26939** (OSSINDEX)
    - **CVE-2023-33201** (OSSINDEX)
* **spring-boot-2.2.4.RELEASE.jar**
  + According to IBM, “Java Spring Framework is a popular, open source, enterprise-level framework for creating standalone, production-grade applications that run on the Java Virtual Machine.”
  + Seems a version upgrade would resolve these. We can upgrade all the way to 3.1 - <https://mvnrepository.com/artifact/org.springframework.boot/spring-boot>
    - [**CVE-2022-27772**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-27772)
      * This vulnerability only affects products and/or versions that are no longer supported by the maintainer.
    - [**CVE-2023-20883**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-20883)
* **log4j-api-2.12.1.jar**
  + The Apache Log4j API which is used for application logs.
  + Once again, an update would be good to consider as all the below mention they were fixed in patches, we could go all the way to 2.20.0
    - [**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)
* **snakeyaml-1.25.jar**
  + YAML 1.1 parser and emitter for Java
  + It appears an update will resolve most of these issues if not all by an update as we are decently out of date. <https://mvnrepository.com/artifact/org.yaml/snakeyaml>
  + Another source to help prove upgrade is good - <https://security-tracker.debian.org/tracker/source-package/snakeyaml>
  + Additionally seems there are many issues working with user input.
    - [**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-2022-1471**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-1471)
    - [**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)
    - [**CVE-2023-2251**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-2251)
* **jackson-databind-2.10.2.jar**
  + General data-binding functionality for Jackson: works on core streaming API
  + Once again updates are recommended at least 3 issues are resolved in one patch - <https://www.debian.org/security/2022/dsa-5283>
    - [**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-2021-46877**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-46877)
    - [**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).
    - [**CVE-2023-35116**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-35116)
* **tomcat-embed-core-9.0.30.jar**
  + According to GeeksforGeeks “Apache Tomcat is a web container. It allows the users to run Servlet and JAVA Server Pages that are based on the web-applications.”
  + Some of these issues go all the way back to 2019. Upgrade as the latest vulnerability from 2023 is covered here - <https://lists.apache.org/thread/hdksc59z3s7tm39x0pp33mtwdrt8qr67>
    - [**CVE-2019-17569**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2019-17569)
    - [**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)
    - [**CVE-2023-28708**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-28708)
* **hibernate-validator-6.0.18.Final.jar**
  + Hibernate's Bean Validation (JSR-380) reference implementation.
  + An update is available that corrects this - <https://access.redhat.com/errata/RHSA-2020:3464>
    - [**CVE-2020-10693**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-10693)
* **spring-web-5.2.3.RELEASE.jar**
  + According to Tutorialspoint “The Spring Web Services project facilitates contract-first SOAP service development, provides multiple ways to create flexible web services”
  + There are many spring articles that indicate an update is needed. Ex: <https://spring.io/security/cve-2021-22118>
    - **CVE-2016-1000027** (OSSINDEX)
      * Do not expose endpoints to untrusted clients
    - **CVE-2020-5421** (OSSINDEX)
    - **CVE-2021-22096** (OSSINDEX)
    - **CVE-2021-22118** (OSSINDEX)
* **spring-beans-5.2.3.RELEASE.jar**
  + Spring bean implementation
  + A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding.
  + 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.
    - **CVE-2022-22965** (OSSINDEX)
* **spring-webmvc-5.2.3.RELEASE.jar**
  + Spring Web MVC implementation
  + Update Available for this - <https://spring.io/security/cve-2021-22060>
    - **CVE-2021-22060** (OSSINDEX)
* **spring-context-5.2.3.RELEASE.jar**
  + Spring Context Implementation
  + An update was made - <https://spring.io/blog/2022/04/13/spring-framework-data-binding-rules-vulnerability-cve-2022-22968>
    - **CVE-2022-22968** (OSSINDEX)
* **spring-expression-5.2.3.RELEASE.jar**
  + Spring Expression Language (SpEL)
  + Updates were made to patch all these issues.
  + <https://spring.io/blog/2022/03/17/spring-framework-6-0-0-m3-and-5-3-17-available-now>
  + <https://spring.io/security/cve-2023-20863>
    - **CVE-2022-22950** (OSSINDEX
    - **CVE-2023-20861** (OSSINDEX)
    - **CVE-2023-20863** (OSSINDEX)

**Mitigation Plan:**

* First, we should make all the suggested upgrades by the dependency check, this will resolve a lot of the problems we have found. This in general is a good practice to do from time to time as there may be bug fixes, vulnerability patches, and new features available in these libraries.
* We should avoid pairing user input with SnakeYaml or make sure there is very good input validation added to prevent issues.
* In the case of Spring Web we need to make sure we do not expose endpoints to untrusted clients. We should make sure user authorization and authentication occur.
* To resolve issues with Spring Beans we need to make sure the application is deployed as an executable Jar file.
* It would be a good idea to run the dependency check again after these selected libraries have been updated to see if anything new is shown and then fix those as well.
* In the DocData Class we should:
  + Implement Input validation needs to be implemented in the ReadDocument() method
  + Implement code to fail gracefully so users know how to proceed, and in the process hiding information from hackers.
  + Implement an encrypted connection to the database so that data is not exposed when transmitted over the network
* In the GreetingController Class we should:
  + Implement input validation
  + Implement error handling in form of a try-catch to gracefully throw an error
* In the case of all APIs implemented in the application:
  + We need to encrypt the data being sent in each request
  + Properly implement all HTTP Requests properly (GET, POST, PUT)
  + Implement User Authentication and Authorization

**Citations**

7 Pernicious Kingdoms. (2006a, July 19). *Common weakness enumeration*. CWE. <https://cwe.mitre.org/data/definitions/20.html#:~:text=When%20software%20does%20not%20validate,resource%2C%20or%20arbitrary%20code%20execution>.

Allen, N. (2023, January 5). *7 reasons why your organization needs secure communications*. Salt. <https://saltcommunications.com/news/7-reasons-why-your-organisation-needs-secure-communications/#:~:text=By%20using%20a%20secure%20communications,your%20company’s%20reputation%20and%20finances>.

Crane, C. (2021, March 11). *10 data privacy and encryption laws every business needs to know*. Hashed Out by The SSL StoreTM. <https://www.thesslstore.com/blog/10-data-privacy-and-encryption-laws-every-business-needs-to-know/>

Ferragamo, J., Wichers, Bird, J., & Kingthorin. (n.d.). *Improper error handling*. Improper Error Handling | OWASP Foundation. <https://owasp.org/www-community/Improper_Error_Handling>

Nesbo, E. (2023, January 31). *What is input validation and why is it important?*. MUO. <https://www.makeuseof.com/what-is-input-validation/>

*Security Breach - How Businesses May Be Liable*. Hg.org. (n.d.). <https://www.hg.org/legal-articles/security-breach-how-businesses-may-be-liable-44358#:~:text=Legal%20Implications%20of%20a%20Breach,problem%20and%20communicate%20with%20customers>.

*Spring Web Services tutorial*. Tutorialspoint. (n.d.). <https://www.tutorialspoint.com/springws/index.htm#:~:text=The%20Spring%20Web%20Services%20project,Dependency%20Injection%20and%20Configurations%20seamlessly>.

StackHawk. (2022, November 18). *6 serious API security vulnerabilities and how to fix them*. StackHawk. <https://www.stackhawk.com/blog/6-serious-api-security-vulnerabilities-and-how-to-fix-them/#:~:text=API%20vulnerabilities%20can%20occur%20when,gain%20access%20to%20confidential%20information>.

*What is Java spring boot?*. IBM. (n.d.). <https://www.ibm.com/topics/java-spring-boot>