****

# CS 305 Project One

**Artemis Financial Vulnerability Assessment Report**

**Dr. Vivian Lyon**

**July 17, 2022**

Table of Contents

[CS 305 Project One 1](#_Toc92292017)

[Document Revision History 2](#_Toc359954504)

[Client 3](#_Toc385716994)

[Instructions 3](#_Toc1847810152)

[Developer 3](#_Toc276724701)

[1. Interpreting Client Needs 4](#_Toc374401231)

[2. Areas of Security 4](#_Toc938094040)

[3. Manual Review 5](#_Toc2018167449)

[4. Static Testing 7](#_Toc474701338)

[5. Mitigation Plan 19](#_Toc735240588)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **7/17/2022** | **Michael Ross** |  |

## 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

Michael Ross

## 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?
* 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?

Artemis Financials is a financial services consulting company. As a financial services company, they handle their client’s money, financial records, personal information and potentially their future. Keeping the client’s information secure is important for the reputation of the brand, and for protecting Artemis Financial from severe fines and loss of capital in the event of a breach.

Being a consulting company, Artemis Financial does not process their customer’s money. They do, however, store a large amount of specific financial information about each customer. As such, they fall under several regulations including the Gramm-Leach-Bliley Act (GLBA) (FTC, 2022), EU-GDPR (EU, 2022), ISO/IEC 27001 (ISO, 2020), and SOX (LII, 2021). These regulations would all impact the types of security needed in the existing REST based interfaces.

The current threat landscape for Financial Services organizations includes global actors attempting to:

* Steal customers PII through targeted attacks.
* Steal customers' financial information.
* Compromise the server to steal the information of other users.

According to Dr Suleyman Ozarslan of Picus Security (Ozarslan, 2022, para. 3), the most common threats facing the financial services and banking firms in 2022 include:

* Ransomware
* Phishing
* Web Application and Vulnerability Attacks
* Denial of Service Attacks
* Insider Threats
* Nation-State sponsored threat actors including Advanced Persistent Threat (APT) groups.

These various threat sources and types must be considered when developing a comprehensive security analysis of the code and any mitigation measures included in the final product.

The open-source community provides a lot of the functionality that goes into modern web-based platforms. Because of the rapid evolution of new frameworks and web technologies, these open-source packages allow developers to implement modern technologies quickly and easily by leveraging proven code. However, this ease of development must be tempered by the needs of security. Recently several very commonly used open-source libraries were compromised (Polasani & Rubin, 2021) and led to the projects utilizing these open-source packages being compromised as well. To mitigate this avenue of attack, only known, trusted, and proven versions of open-source libraries and packages should be utilized.

## 2. Areas of Security

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

The application under review includes several key indicators which will inform the areas of the vulnerability flowchart to include in our analysis. First, the application is a REST based application which provides data to entities external to the datacenter. This means that we should check the areas relevant to **APIs**.

Second, the application has external, untrusted, entities providing input to the application. To ensure that this data is unlikely to cause issues with our application, **input validation** needs to be performed to sanitize and normalize the data coming in to be processed.

Given that this application is part of a financial institution, **cryptography** absolutely needs to be considered. Topics like using HTTPS for secure transfer of client information as well as database and storage encryption need to be addressed. Storage level encryption will keep the data safe if any unauthorized users get access to the storage. Using HTTPS for communication will ensure that data in flight remains secure.

This REST API is designed to be utilized by a client. Without the client to call the information from the server, the data contained in the server is useless. To ensure that communications between the client and the server are secure, the architecture should encompass good **client/server** security best practices.

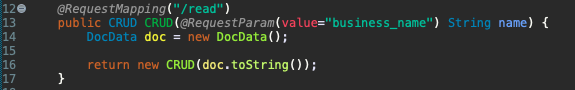
**Code quality** plays into each of the security areas. To ensure that code is utilizing secure coding practices, the code should be checked against code quality standards. Without examining this, each of the previous areas could have obvious code quality issues while at the same time meeting their respective security standards.

This code both logs errors and has rudimentary error detection and correction through try/catch statements. To catch any logical errors that could result in a vulnerability or exploitable issue the code should be reviewed for **Code Errors.**

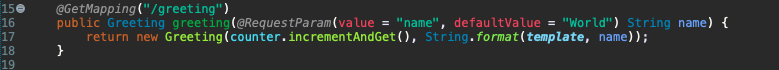
## 3. Manual Review

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

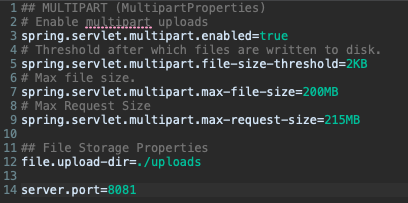
Taking Input Validation first, several code issues were found that needed input validation. Lines 13 through 16 in the CRUDController.java file take a request parameter in and fail to validate the data or process it in any way. This should be processed by a centralized data validation class to evaluate it for common vulnerabilities or to whitelist the contents.



The second Input Validation issue is on lines 16 and 17 of the GreetingController.java file. A request parameter is being pulled from an untrusted source and is used in the Greeting object. The value of this parameter should be run through a centralized input validation class to ensure that it is appropriate and sanitized before committing it to the Greeting object.



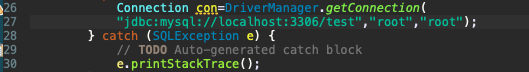
In examining the REST application for security issues, several were found. First, the API server should be run on port 443 only and utilize HTTPS for all client communication with the server. The application.properties file determines the configuration of the running Spring server. The current contents of the application.properties file are as follows:



To allow for secure HTTPS communication, a “server.ssl” section would need to be added after line 14 and an appropriate certificate uploaded to the server.

The second problem in the category of APIs is that there is no authentication in the application. Anyone with access to the URL the application is running on can access the application. We need to be able to control who can access sensitive customer data, and so authentication and authorization needs to be added. To add authentication and authorization to the application, Spring Security module can be added. Each class which exposes a REST endpoint should check for proper authentication and authorization against a central authorization and authentication class to eliminate the possibility of unauthorized users accessing the server.

While HTTPS communication sits in the sphere of API security, it also fits into the realm of cryptography and all the same remediations also apply here. Additionally, the connection to the database in the DocData.java file on line 27 does not require an SSL connection to the database.

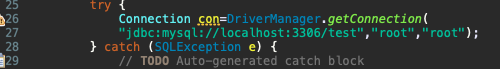


To make this more secure, the connection string should include ‘requireSSL=true’ (Oracle, 2022). This addition will force the connection string to use encryption when communicating with the database.

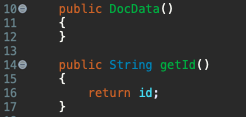
The customer.java class is vulnerable to an integer overflow exploit. An arbitrary number is being added to the account balance. If the total sum overflows the bounds of an integer, it could cause the program to crash causing a denial of service or DOS. This can be seen on lines 12-14 of the code.



One example of a secure coding violation is the inclusion of the password for the jdbc data connection directly in the source code. This can be seen on line 27 of the DocData.java class file.



If there is an error connecting, the username and password can be output as part of the stack trace which will be logged. This represents a serious violation of secure coding standards. In the same file, the id variable is never properly initialized through the constructor. The getId method allows retrieving the value of the id variable in an uninitialized state. This can result in a crash or unintended data leak. It can be found on lines 10-17.



Finally, examining the pom.xml file reveals that older versions of the Spring Framework are being referenced. This should be updated to the latest GA version if possible (version 5.3.22 as of this writing). (Spring.io, 2022)

## 4. Static Testing

Run a dependency check on Artemis Finacial'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

|  |  |  |  |
| --- | --- | --- | --- |
| CVE | Description | Resolution | Attribution |
| **Bouncy Castle** |  |  |  |
| [**CVE-2016-1000338**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000338) | This vulnerability impacts software running DSA ASN.1 validation. Because the ASN.1 validation does not check the length of the signature, extra data can be added into the signature and still be processed as valid. | This vulnerability only affects versions of bouncy-castle JCE provider versions 1.55 and earlier. Upgrading to a version greater than 1.55 will resolve this issue. | None |
| [**CVE-2016-1000342**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000342) | This vulnerability impacts software running ECDSA ASN.1 validation. Because the ASN.1 validation does not check the headers of the signature properly, extra data can be added to the signature and still be processed as valid. | This vulnerability only impacts versions of bouncy-castle JCE provider versions 1.55 and earlier. Upgrading to a version greater than 1.55 will resolve this issue. | None. |
| [**CVE-2016-1000343**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000343) | In versions of Bouncy Castle JCE provider 1.55 and below, the defaults for the DSA key pair generator are not secure. If no explicit input is provided, then the generator will utilize a 1024-bit key size. | Provide additional values to the DSA key pair generator, specifically a higher bit key size, for versions below 1.55 or upgrade to a version of Bouncy Castle greater than 1.55 |  |
| [**CVE-2016-1000344**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000344) | Electronic Code Block (ECB) encryption is an inherently insecure method of encrypting data. Versions of Bouncy Castle JCE 1.55 and below allow ECB encryption to be used when implementing DHIES. | Do not utilize ECB encryption when using versions of Bouncy Castle JCE at or below v1.55 and using DHIES. To eliminate the issue upgrade to a version higher than 1.55 to disable the ability to use ECB altogether. | None |
| [**CVE-2016-1000352**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000352) | Electronic Code Block (ECB) encryption is an inherently insecure method of encrypting data. Versions of Bouncy Castle JCE 1.55 and below allow ECB encryption to be used when implementing ECIES. | Do not utilize ECB encryption when using versions of Bouncy Castle JCE at or below v1.55 and using ECIES. To eliminate the issue upgrade to a version higher than 1.55 to disable the ability to use ECB altogether. | None |
| [**CVE-2016-1000341**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000341) | The generation of DSA keys is subject to a timing attack. This is because there is no blinding enabled on versions prior to 1.55. Without blinding, an attacker with the precise time the DSA key was generated can generate the same key and decrypt information sent between the two endpoints | Versions greater than 1.55 have resolved this issue by implementing a randomizer function in the DSA key generation. | None |
| [**CVE-2016-1000345**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000345) | In versions 1.55 and below of the Bouncy Castle JCE provider the provider is vulnerable to an oracle padding vulnerability. This happens when the server leaks information on whether a particular message is properly padded. With enough attempts, the attacker can decipher the key. | Versions greater than 1.55 have resolved this issue. | None |
| [**CVE-2017-13098**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-13098) | This entry appears to be a false positive as it lists Bouncy Castle versions before 1.03 and version 1.55 is in use | No remediation necessary | None |
| [**CVE-2020-15522**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-15522) | Affects versions of Bouncy Castle below 1.66. When generating an ECDSA key, if an attacker has access to closely monitor the timings, they can eventually obtain information about the private key. | Tightly controlling who has access to the timings will help to remediate this issue. Upgrading to version 1.67 or above of Bouncy Castle will also remediate this issue. | "Yet another GCD based inversion side-channel affecting ECC implementations" by Nir Drucker and Shay Gueron |
| **CVE-2020-0187** | This appears to be a false positive as it only affects android versions 10 or below. | No remediation necessary | None |
| [**CVE-2016-1000339**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000339) | In versions of Bouncy Castle 1.55 or below, the AesFastEngine leaks data via the lookup table used for Aes encryption. An attacker who has access to the server and can monitor CPU activity can get the keys in use. | Tightly controlling who has access to the servers using Least Privilege models can help to remediate this issue. Upgrading to 1.56 or above where the default method for Aes operations has been changed to AesEngine which does not exhibit the leak will also resolve the vulnerability. | None |
| **CVE-2020-26939** | This vulnerability impacts versions of Bouncy Castle below 1.61. By timing the response to an invalid cyphertext submission, the OAEP Decoder can throw an early exception. This can leak information about the private exponent of the RSA private key performing the encryption. | Upgrading to version 1.61 of the Bouncy Castle package resolves this issue. A workaround for situations where the version cannot be upgraded is to perform a raw RSA decryption for the purposes of checking the length of the payload and then decrypting a known, second payload, and then returning a success or failure based on the length. This ensures that the timing is the same as for a payload with a valid length. | https://github.com/bcgit/bc-java/wiki/CVE-2020-26939 |
| [**CVE-2015-7940**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-7940) | Versions of Bouncy Castle below 1.51 allow an “invalid curve” exploit to be performed. This is where an attacker can consistently guess points outside the Diffie Hellman curve to help them decrypt the private key. | Upgrading to versions 1.51 or above will resolve this issue. | None |
| [**CVE-2018-5382**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2018-5382) | In versions of Bouncy Castle below 1.46 the default HMAC key is only 16 bits long. This is sufficiently small for an attacker to decrypt the key and compromise the server. Versions 1.46 and above change the default size of the HMAC key to 160 bits. | Manually specifying a larger HMAC key can resolve the issue. Upgrading to version 1.46 or above will change the default to 160 bits and resolve the issue as well. | https://nvd.nist.gov/vuln/detail/CVE-2018-5382 |
| [**CVE-2013-1624**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-1624) | The TLS implementation in the Bouncy Castle Java library before 1.48 does not consider timing side-channel attacks on a noncompliant MAC check operation during the processing of malformed CBC padding, which allows remote attackers to conduct distinguishing attacks and plaintext-recovery attacks via statistical analysis of timing data for crafted packets. | Upgrading to Bouncy Castle version 1.48 or above to resolve the issue. | https://nvd.nist.gov/vuln/detail/CVE-2013-1624 |
| [**CVE-2016-1000346**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000346) | In versions 1.55 and earlier of Bouncy Castle, the remote keys for Diffie Hellman transactions are not fully validated. This can allow an attacker to craft special requests and reveal information about the private key in use. | Upgrading to a version of Bouncy Castle above 1.55 will resolve the issue. | https://nvd.nist.gov/vuln/detail/CVE-2016-1000346 |
| **CVE-2015-6644** | This appears to be a false positive | No remediation necessary | None |
| **Hibernate-validator** |  |  |  |
| [**CVE-2020-10693**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-10693) | Version 6.0.18 of the hibernate-validator dependency has an input validation vulnerability. The module does not properly validate EL expressions and so allows an attacker to bypass the user created input validation and sanitization measures implemented by the programmer. This is under CVE-2020-10693. | The issue has been resolved in Spring Boot version 2.2.6. The version this application is using is 2.2.4. Upgrading to a version of the Spring Boot framework that is equal to or higher than 2.2.6 will resolve the issue. | https://bugzilla.redhat.com/show\_bug.cgi?id=CVE-2020-10693 |
| **Jackson-databind** |  |  |  |
| [**CVE-2020-25649**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-25649) | The vulnerability is due to improper security in the way it handles entity expansion. This vulnerability makes it susceptible to XML external entity attacks. This could lead to data integrity issues if the vulnerability is exploited. | To resolve the issue, we need to specify a newer version of the jackson-databind module in the pom.xml file. Versions 2.10.5.1 or 2.11.0 and later will work to resolve the vulnerability. | https://github.com/FasterXML/jackson-databind/issues/2589 |
| [**CVE-2020-36518**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-36518) | The second vulnerability affects any versions of jackson-databind below 2.13.0. This vulnerability allows a stack overflow which could potentially cause a denial of service to the application because of the substantial number of nested objects. | Updating the pom.xml file to use jackson-databind version 2.14 or above will resolve this issue. It is also worth noting that this vulnerability only occurs with exceptionally large nested (100k levels) JSON objects. If upgrading is not possible, a limit could be set on the JSON objects being deserialized to prevent this exploit from being used. | None |
| **Jakarta-annotation** |  |  |  |
| [**CVE-2022-31569**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-31569) | This is a false positive | No remediation necessary | None |
| **Log4j-api** |  |  |  |
| [**CVE-2020-9488**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9488) | Version 2.12.1 of the log4j-api module has a vulnerability in the validation of certificates which allows an attacker to be able to intercept SMTPS traffic and perform a man-in-the-middle exploit. | None of the code in our application utilizes SMTPS currently, so there is no danger of the existing code being exploited. However, if it were necessary to utilize SMTPS in the future, upgrading to version 2.13.2 would resolve the issue. | <https://issues.apache.org/jira/browse/LOG4J2-2819> |
| **Logback-core** |  |  |  |
| [**CVE-2021-42550**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-42550) | Version 1.2.3 and any versions including and below version 1.2.7 allows an attacker to craft a malicious configuration which allows the attacker to execute arbitrary code from LDAP servers. | The existing code does not include any references to the scan attribute which is required to exploit this vulnerability or utilize this functionality. To resolve this issue, avoid having an arbitrary or user-defined path for any uploads sent to the server. | https://github.com/cn-panda/logbackRceDemo |
| **Snakeyaml** |  |  |  |
| [**CVE-2017-18640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-18640) | Version 1.25 of the snakeyaml module has an exploitable vulnerability in the way that it expands during a load operation. | It can be resolved by using version 1.26 of the snakeyaml module. This can be enabled by inserting the dependency code in the pom.xml file | None |
| **Spring-boot** |  |  |  |
| [**CVE-2022-27772**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-27772) | Version 2.2.4 of the spring-boot module contains vulnerability CVE-2022-27772. This vulnerability is caused by a race condition in creating a temporary folder for various files needed by the web server. If an attacker that has access to the server can create a necessary folder before the spring-boot process can create it, the process fails silently, and the attacker now has control over this folder to read any files placed there and to write any files they want to the location. This can lead to data being exposed or further vulnerabilities being exploited. | The fix for this vulnerability is to upgrade to spring-boot version 2.2.11 or later. Version 2.2.4 is no longer supported, and the version should be upgraded to 2.2.11 or later supported version. | None |
| **Spring-Core** |  |  |  |
| [**CVE-2022-22965**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965) | This vulnerability requires that the associated MVC or Spring WebFlux application be running on JDK 9 or higher, using Apache Tomcat as the servlet container, and be packaged as a WAR file. | Upgrading to Spring Framework 5.2.20 or higher will resolve the vulnerability. | None |
| [**CVE-2021-22118**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22118) | This vulnerability allows a privilege escalation attack against vulnerable versions of the spring-core module. By rewriting files in temporary directories utilized by spring-core, an attacker can generate malicious multi-part form data. | Upgrading to Spring Framework 5.2.15 and above will resolve this vulnerability. | None |
| [**CVE-2020-5421**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-5421) | This vulnerability allows a bypass of previous protections against RFD which stands for reflected file download. RFD allows an attacker to make a seemingly harmless file from a trusted domain execute local executables or exfiltrate cookies to an attacker’s address. See (https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/reflected-file-download-a-new-web-attack-vector/) for a great explanation of this phenomenon. By using a jsessionid path parameter, an attacker can bypass these RFD protections and execute an RFD attack. | Upgrading to Spring Framework 5.2.9 or above resolves this vulnerability. | None |
| [**CVE-2022-22950**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22950) | This is a vulnerability which impacts SpEL expressions in spring. It allows an attacker to generate an expression which could cause a denial-of-service attack against the vulnerable application. | Upgrading to spring-framework version 5.2.20 or above resolves this vulnerability. | None |
| [**CVE-2022-22971**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22971) | This vulnerability impacts applications utilizing STOP (simple text-oriented messaging protocol) over WebSockets. A denial-of-service attack can be executed for servers running vulnerable versions of the framework (versions before 5.2.22). | Upgrading to version 5.2.22 or later resolves this vulnerability. | None |
| [**CVE-2022-22968**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22968) | This vulnerability impacts applications utilizing disallowedFields on DataBinders. Because the patterns for these filters are case sensitive. Unless the disallowedFields are listed with both the first letter capitalized and uncapitalized, the fields will not be effectively protected. The same goes for any nested fields. | Upgrading to version 5.2.21 or above will resolve this issue. | None |
| [**CVE-2022-22970**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22970) | This vulnerability impacts Spring MVC or WebFlux applications which both handle uploads and rely on data binding for multi-part uploads. A DoS attack can be executed on servers running the vulnerable versions of the software and meeting the conditions above. | Upgrading to version 5.2.22 of the framework resolves this vulnerability. | None |
| [**CVE-2021-22060**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22060) | This vulnerability allows additional log files to be inserted by a user issuing malicious input. | Upgrading to version 5.2.19 of the spring framework resolves this vulnerability. | None |
| [**CVE-2021-22096**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22096) | This vulnerability is similar to CVE-2021-22060 in that it allows malicious input from a user to allow additional log entries to be inserted. | Upgrading to version 5.2.18 or above resolves this vulnerability. | None |
| **Spring-Web** |  |  |  |
| [**CVE-2016-1000027**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000027) | When deserializing untrusted data, this vulnerability allows a potential RCE or remote code execution issue. | The resolution to this issue is to ensure that untrusted data is not being deserialized using this package. | None |
| [**CVE-2022-22965**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965) | Duplicate | No remediation necessary | None |
| [**CVE-2021-22118**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22118) | Duplicate | No remediation necessary | None |
| [**CVE-2020-5421**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-5421) | Duplicate | No remediation necessary | None |
| [**CVE-2022-22950**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22950) | Duplicate | No remediation necessary | None |
| [**CVE-2022-22971**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22971) | Duplicate | No remediation necessary | None |
| [**CVE-2022-22968**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22968) | Duplicate | No remediation necessary | None |
| [**CVE-2022-22970**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22970) | Duplicate | No remediation necessary | None |
| [**CVE-2021-22060**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22060) | Duplicate | No remediation necessary | None |
| [**CVE-2021-22096**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22096) | Duplicate | No remediation necessary | None |
| **Tomcat-embed-core** |  |  |  |
| [**CVE-2020-1938**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1938) | When using AJP (Apache JServ Protocol) in vulnerable versions of the package, an attacker can potentially execute an RCE attack on the server. This happens when file upload is allowed and the AJP port is exposed to non-trusted users. | The recommended fix for this vulnerability is to:   * Block external access to AJP ports where possible. * Limit internal access to AJP ports to trusted nodes. * Upgrade to a newer version that has a more hardened default configuration. 9.0.31 and above. | None |
| [**CVE-2020-11996**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-11996) | HTTP/2 requests sent to the server using a specially crafted sequence can cause a CPU spike. If many of these requests are sent sequentially, a DoS attack can be carried out. | Upgrading to versions 9.0.36 and above will resolve this issue. | None |
| [**CVE-2020-13934**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13934) | When an http2 over cleartext connection is upgraded to http2, the process does not release the processor. If many requests of this type are made sequentially, the memory allocated can be exhausted which leads to an OutOfMemoryException which could lead to a DoS. | Upgrading to a version greater than 9.0.36 will resolve this issue. | None |
| [**CVE-2020-13935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13935) | The length of the payload in a WebSocket frame is not properly validated in vulnerable versions of this package. Because of this, selecting an infinite payload length could trigger an infinite loop. If many such requests are made in sequence, a DoS could occur. | Upgrading to a version greater than 9.0.36 will resolve this issue. | None |
| [**CVE-2020-17527**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-17527) | An information leak can sometimes occur between adjacent streams due to HTTP request header re-use. This will often result in the connection being closed but could sometimes leak session request information. | Upgrading to a version greater than 9.0.39 will resolve the issue. | None |
| [**CVE-2021-25122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25122) | Request headers from adjacent requests can sometimes be duplicated. This could cause users to see each other's requests. | Upgrading to a version greater than 9.0.41 will resolve this issue. | None |
| [**CVE-2021-41079**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-41079) | TLS packets are not being validated properly. When using NIO+OpenSSL or NIO2+OpenSSL for TLS an infinite loop can be triggered by using a specially crafted packet. This can lead to a DoS. | Upgrading to a version greater than 9.0.43 will resolve this issue. | None |
| [**CVE-2022-29885**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-29885) | If the following conditions are satisfied:   * An attacker with sufficient access to the server to control the contents and filenames * PersistenceManager with a FileStore is configured on the server. * PersistenceManager is configured with sessionAttributeValueClassNameFilter=”null” * The attacker knows the relative file path from the storage location used by FileStore to the file or folder the user has control over.   Then an attacker can trigger a remote code execution using deserialization of a file under their control | Upgrading to a version greater than 9.0.34 will resolve this issue. | None |
| [**CVE-2020-9484**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9484) | If the following conditions are satisfied:   * An attacker with sufficient access to the server to control the contents and filenames * PersistenceManager with a FileStore is configured on the server. * PersistenceManager is configured with sessionAttributeValueClassNameFilter=”null” * The attacker knows the relative file path from the storage location used by FileStore to the file or folder the user has control over.   Then an attacker can trigger a remote code execution using deserialization of a file under their control | Upgrading to a version greater than 9.0.34 will resolve this issue. | None |
| [**CVE-2021-25329**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25329) | This is to correct an issue with the fix for CVE-202-9484. The vulnerability was still able to be exploited. | Upgrade to a version greater than 9.0.41 to resolve the issue. | None |
| [**CVE-2021-30640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-30640) | When using authentication, an attacker can authenticate using variations of a valid username to bypass protections provided by the LockOut realm. This is caused by a vulnerability in the JNDI Realm. | Upgrading to version greater than 9.0.45 will resolve this issue. | None |
| [**CVE-2022-34305**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-34305) | The examples web form application did not validate form data exposing the application to a Cross Site Scripting attack | Upgrading to version 9.0.65 or above will resolve the vulnerability. | None |
| [**CVE-2021-24122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-24122) | When serving files from a network location utilizing NTFS, a JSP source code disclosure can occur in some situations. | Upgrading to a version greater than 9.0.39 will resolve this issue. | None |
| [**CVE-2021-33037**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-33037) | There is the possibility of HTTP request smuggling due to incorrectly parsing the HTTP transfer-encoding request header. This happens when used with a revers proxy. | Upgrading to a version greater than 9.0.46 will resolve this issue. | None |
| [**CVE-2019-17569**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2019-17569) | There is the possibility of HTTP request smuggling due to a regression which incorrectly handles Transfer-Encoding headers. This can happen when the server is behind a reverse proxy that incorrectly handles the invalid Transfer-Encoding header in a particular way. | Upgrading to 9.0.30 or above will resolve this issue. | None |
| [**CVE-2020-1935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1935) | There is the possibility of HTTP request smuggling if the Tomcat server is located behind a reverse proxy that incorrectly handles the invalid Transfer-Encoding header in a specific way. | Upgrading to a version greater than 9.0.30 will resolve this issue. | None |
| [**CVE-2020-13943**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13943) | Users could see responses from other sessions if they intentionally exceed the agreed maximum number of concurrent streams for a connection. | Upgrading to a version greater than 9.0.37 will resolve this issue. | None |
| **Tomcat-Embed-Websocket** |  |  |  |
| [**CVE-2020-1938**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1938) | Duplicate | No remediation necessary | None |
| [**CVE-2020-8022**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-8022) | Incorrect default permissions allow users of the tomcat group to perform a full local root exploit on the server. | Upgrading to a version greater than or equal to 9.0.35 will resolve this issue. | None |
| [**CVE-2020-11996**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-11996) | Duplicate | No remediation necessary | None |
| [**CVE-2020-13934**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13934) | Duplicate | No remediation necessary | None |
| [**CVE-2020-13935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13935) | Duplicate | No remediation necessary | None |
| [**CVE-2020-17527**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-17527) | Duplicate | No remediation necessary | None |
| [**CVE-2021-25122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25122) | Duplicate | No remediation necessary | None |
| [**CVE-2021-41079**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-41079) | Duplicate | No remediation necessary | None |
| [**CVE-2022-29885**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-29885) | Duplicate | No remediation necessary | None |
| [**CVE-2020-9484**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9484) | Duplicate | No remediation necessary | None |
| [**CVE-2021-25329**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25329) | Duplicate | No remediation necessary | None |
| [**CVE-2021-30640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-30640) | Duplicate | No remediation necessary | None |
| [**CVE-2022-34305**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-34305) | Duplicate | No remediation necessary | None |
| [**CVE-2021-24122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-24122) | Duplicate | No remediation necessary | None |
| [**CVE-2021-33037**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-33037) | Duplicate | No remediation necessary | None |
| [**CVE-2019-17569**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2019-17569) | Duplicate | No remediation necessary | None |
| [**CVE-2020-1935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1935) | Duplicate | No remediation necessary | None |
| [**CVE-2020-13943**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13943) | Duplicate | No remediation necessary | None |

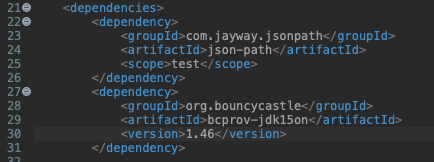
## 5. Mitigation Plan

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

The first step to mitigating vulnerabilities will be to upgrade all packages to non-vulnerable versions. The table below lists the appropriate versions of the packages to include.

|  |  |
| --- | --- |
| Package Name | Minimum Version Required |
| bcprov-jdk15on | 1.67 |
| Jackson-databind | 2.13.0 |
| Log4j-api | 2.13.2 |
| Logback-core | 1.2.8 |
| snakeyaml | 1.2.6 |
| Spring-boot | 2.2.11 |
| Spring-core | 5.2.22 |
| Spring-web | 5.2.22 |
| Tomcat-embed-core | 9.0.65 |
| Tomcat-embed-websocket | 9.0.65 |

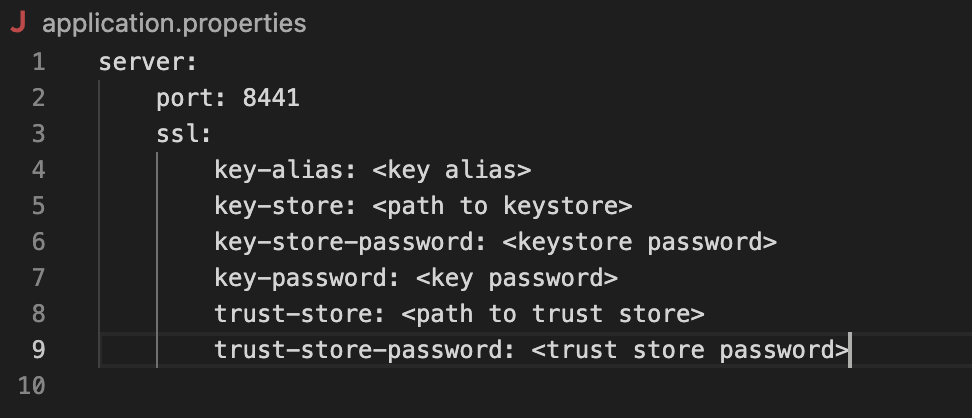
The process for updating the versions of these packages would be similar for each one. In the pom.xml file, a section would be added for each dependency with a version tag. See line 30 for an example of this:



The appropriate versions would be filled in and the Maven install process run again to pull the new versions.

After the new versions have been pulled and integrated into the project, the issues found during the manual inspection can be corrected. For input validation issues, such as the ones seen in lines 13-17 of the CRUDController class and 16-18 of the GreetingController class, a new class should be created. This InputValidation class would house all the methods to validate various inputs used in the project. By housing them in a central class, we can be sure that if new validation needs to be performed, we can change it in one location rather than hunting down the hundreds of discreet locations where input validation is being performed. After a new InputValidation class has been created, adding an input validation check before processing the input data will be performed.

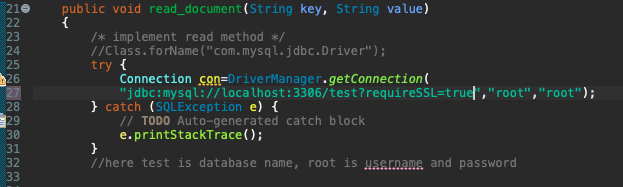
To ensure that the application is using SSL for communication between the client and the server, new lines need to be added to the application.properties file and a certificate needs to be created. For security, a third-party certificate from a trusted certificate authority should be utilized. (Skipper, 2022) The certificate, once generated needs to be installed in a java keystore and the details of the keystore added to the application.properties file. The format is like the code snippet below:



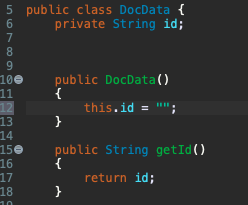
Adding Spring Security would allow the application to include authentication and authorization. First, a new Authorization class file would need to be added with all the authorization logic. This ensures that any change to authorization logic or policy can be updated in one place for all classes which utilize it. In each of the REST classes, a check will be performed before any application logic is performed. If the user is not authenticated, it will bring them to a login screen. If a user is not authorized, it will display the appropriate error message. To add Spring Security a new dependencies line would need to be added to pom.xml to bring in the Spring Security package.

In examining the jdbc implementation in this application in the DocData class, there are several issues to be resolved. The first is the use of the username and password hard coded into the application. This is a security issue as well as an extensibility issue. The password for this account could easily change and having to recompile and release an updated version of the code every time the password changes would be a burden. In addition, the code for this application would be committed to a code repository exposing the password to anyone with access. Storing the username and password in a secure file that the application would reference is the recommended fix. (Oracle, 2022) A connection properties file must be created to store the username and password. That file would then be used to provide the credentials to connect to the database.

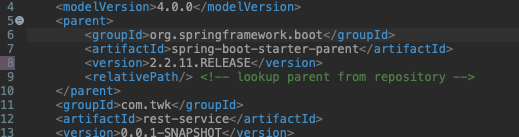
After connecting to the database, the connection needs to be secure. To ensure that only a secure connection will be utilized, “requireSSL=true” needs to be added to the connection string. The result would look like this on line 27:



To resolve issues with the constructor and null values in the DocData class, a default or initialized value for id should be added in the constructor. It should look like line 12 in the code block below:



Finally, updating the pom.xml file so that an appropriate version of the spring boot framework is loaded should be performed. Modify line 8 to reference the appropriate version as in the code block below:



Citations

EU. (2022). *General Data Protection Regulation (GDPR) compliance guidelines*. GDPR.eu. Retrieved July 17, 2022, from <https://gdpr.eu/>

FTC (2022, February 11). *Gramm-Leach-Bliley Act*. Federal Trade Commission. Retrieved July 16, 2022, from <https://www.ftc.gov/business-guidance/privacy-security/gramm-leach-bliley-act>

ISO. (2020, April 3). *ISO/IEC 27001 - information security management*. ISO. Retrieved July 17, 2022, from <https://www.iso.org/isoiec-27001-information-security.html>

Legal Information Institute. (2021, April). *Sarbanes-Oxley act*. Legal Information Institute. Retrieved July 17, 2022, from <https://www.law.cornell.edu/wex/sarbanes-oxley_act>

*MySQL Connector/J 5.1 developer* *guide :: 5.7 connecting securely using SSL*. MySQL. (n.d.). Retrieved July 16, 2022, from <https://dev.mysql.com/doc/connector-j/5.1/en/connector-j-reference-using-ssl.html>

Oracle. (n.d.). *Establishing a Connection*. JDBC Basics. Retrieved July 17, 2022, from <https://docs.oracle.com/javase/tutorial/jdbc/basics/connecting.html>

Ozarslan, D. S. (2022, March 24). *Key threats and cyber risks facing financial services and banking firms in 2022*. Key Threats and Cyber Risks Facing Financial Services and Banking Firms in 2022. Retrieved July 16, 2022, from <https://www.picussecurity.com/key-threats-and-cyber-risks-facing-financial-services-and-banking-firms-in-2022#:~:text=Ransomware%2C%20phishing%2C%20web%20application%20and,financial%20institutions%20face%20in%202022>.

Polasani, G., & Rubin, S. (2021, December 13). *Embedded malware in NPM:* *COA, RC, UA-parser - fossa*. FOSSA. Retrieved July 16, 2022, from <https://fossa.com/blog/embedded-malware-npm-coa-rc-ua-parser/>

Skipper. (n.d.). *Enabling HTTPS*. Spring Boot Documentation. Retrieved July 17, 2022, from <https://docs.spring.io/spring-cloud-skipper/docs/1.0.0.BUILD-SNAPSHOT/reference/html/configuration-security-enabling-https.html>

Spring.io. (n.d.). *Spring Framework*. Spring Documentation. Retrieved July 17, 2022, from <https://spring.io/projects/spring-framework#support>