****

# CS 305 Project One

**Artemis Financial Vulnerability Assessment Report**

Table of Contents

[Document Revision History 3](#_Toc32574607)

[Client 3](#_Toc32574608)

[Developer 4](#_Toc32574610)

[1. Interpreting Client Needs 4](#_Toc32574611)

[2. Areas of Security 4](#_Toc32574612)

[3. Manual Review 4](#_Toc32574613)

[4. Static Testing 5](#_Toc32574614)

[5. Mitigation Plan](#_Toc32574615) 8

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **07/11/2022** | **Brandon Coulter** |  |

## Client



## Developer

Brandon Coulter

## 1. Interpreting Client Needs

Artemis Financial, a financial firm that deals with personal financial information for their cliental, requires top notch security to ensure their clients as well as the company remain secure and safe from outside threats. Given Artemis Financial is using a web application, it can be assumed that transactions can and will take place globally. Because of the web based nature of this application, secure communications are an invaluable asset; this in part due to the safety of the customers and their personally identifiable information, but also for the company’s data and reputation. Additionally, it is important to understand and follow any restrictions with relation to secure communications imposed by the government, whether federal or state. As described by the National Conference of State Legislatures, “An increasing number of laws also require specific measures to protect sensitive information from unauthorized access, destruction, use, modification, or disclosure. The measures include required training for state employees, periodic security audits or assessments, development of standards and guidelines, and other provisions.” (*Data security laws: State government,* 2020) These are only a few of the requirements that may be imposed on Artemis Financial with regards to secure communications and their financial application. Through the nature of their business, Artemis Financial and thus their cliental will be at severe risk of attacks from external entities. The most predictable of these would be a man in the middle situation, under which non secure information could be stolen or compromised. As a result, it is absolutely critical to ensure all communications and the application in general are secure, encrypted, and safe from those types of attacks. Lastly, as time moves forward, aspects of the applications such as utilized libraries and various other dependencies will update. It is pertinent to maintain and validate that the libraries and dependencies are as up to date as possible. This will ensure known vulnerabilities are less of a risk as these vulnerabilities are fixed and patched with new releases.

## 2. Areas of Security

The security areas of most concern are input validations, APIs, cryptography, code error and code quality. These areas are of concern because they directly affect the quality and security of the application at hand. Take for example, input validations, this area of security is important for ensuring that the input given to the system is not only valid but not malicious as well. This is done by validating the input given and handling errors if there are any. With a web application like Artemis Financial has, the application is utilizing a RESTful API to pass data. Because of this, it is imperative to make certain that the API is securely set up and operated as this will help safeguard the clients and the company. With sensitive information on the line from both domestic and foreign accounts, Artemis Financial will need to encrypt their communications utilizing proper cryptography and HTTPS protocol to ensure data cannot be viewed if intercepted. The importance of code error and proper error handling is to prevent malicious attacks from erroneous errors that stem from user inputs. This portion will work hand in hand with the input validation. As this web application will be handling important user and company information, code quality and coding best practices are a must to ensure that this program is secure and built to modern security standards.

## 3. Manual Review

Manually inspecting the code has left a few dashing concerns, of which API stands out distinctly. The APIs utilized in the given code are all multiple versions behind the most current version. For example, the spring boot dependency version is 2.2.4.RELEASE and is outdated. This can be found in pom.xml line 8. Again, the dependency from bouncycastle bcprov-jdk15on which is version 1.46 is quite outdated as well. This is found in pom.xml line 30. In fact, the version of java in use is also older as the code uses version 1.8 which is found again in pom.xml line 18. Validation of inputs does not appear to be accomplished in either of the input areas in the files CRUDController.java or GreetingController.java. This could lead to attacks that cause memory overloading. As for cryptography, the RestServiceApplication.java lacks a set up for security within the application. Additionally, there does not appear to be any error handling to prevent malicious executable log entries or sensitive information in the logs. This is a must as the lack of security for error handling could leak sensitive information to a potential attack if the error is not handled securely. Last but not least is code quality; of which the application should be checked for best practices and secure coding means. This comes into play for instance with the input validation mentioned above; to prevent memory overload, quality code would limit the number of characters to be input. This is not done in either of the above mentioned files.

## 4. Static Testing

The static testing of Artemis Financial’s software application has identified the following dependency vulnerabilities:

Dependency: bcprov-jdk15on-1.46.jar:

[CVE-2016-1000338](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000338): This version does not fully validate the encoding of a e-signature.

(Booth et. al., 2015)

[CVE-2016-1000342](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000342): Version 1.55 and earlier do not validate encoding of e-signatures.

(Booth et. al., 2015)

[CVE-2016-1000343](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000343): The key generator could generate a weak key given default values. (Booth et. al., 2015)

[CVE-2016-1000344](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000344): This allows for an unsafe mode no longer supported to be used. (Booth et. al., 2015)

[CVE-2016-1000352](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000352): Version 1.55 and earlier utilized ECB mode, which is unsafe. (Booth et. al., 2015)

[CVE-2016-1000341](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000341): Generation of DSA signatures were susceptible to timing attacks. (Booth et. al., 2015)

[CVE-2016-1000345](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000345): This version was sensitive to padding oracle attacks. (Booth et. al., 2015)

[CVE-2017-13098](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-13098): This version was vulnerable to attackers recovering weak ciphers. (Booth et. al., 2015)

[CVE-2020-15522](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-15522): Susceptible to timing information issues, an attacker could steal private keys. (Booth et. al., 2015)

CVE-2020-0187: Because of incomplete comparisons, potential incorrect cryptographic algorithms could be selected. (Booth et. al., 2015)

[CVE-2016-1000339](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000339): This version could potentially leak AES keys being utilized. (Booth et. al., 2015)

CVE-2020-26939: Due to improper error handling, attacker could view sensitive data. (Booth et. al., 2015)

[CVE-2015-7940](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-7940): This version is vulnerable to an invalid curve attack. (Booth et. al., 2015)

[CVE-2018-5382](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2018-5382): This version allowed the cripple the reliability of BKS keystore. (Booth et. al., 2015)

[CVE-2013-1624](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-1624): Attackers could use timing data to conduct various attacks. (Booth et. al., 2015)

[CVE-2016-1000346](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000346): Dh keys were not fully validated in this version. (Booth et. al., 2015)

CVE-2015-6644: Attackers could utilize special applications to gain sensitive data. (Booth et. al., 2015)

Dependency: hibernate-validator-6.0.18.FINAL.jar

[CVE-2020-10693](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-10693): A bug in the interpolation processing allowed for invalid expression to be validated. (Booth et. al., 2015)

Dependency: jackson-databind-2.10.2.jar

[CVE-2020-25649](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-25649): A high threat of data integrity came from XML external entity attacks. (Booth et. al., 2015)

[CVE-2020-36518](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-36518): Denial of service attacks were possible due to nested objects. (Booth et. al., 2015)

Dependency: log4j-api-2.12.1.jar

[CVE-2020-9488](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9488): A certificate validation flaw allowed for man-in-the-middle attacks. (Booth et. al., 2015)

Dependency: logback-core-1.2.3.jar

[CVE-2021-42550](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-42550): Attackers could access privileges through malicious configurations. (Booth et. al., 2015)

Dependency: snakeyaml-1.25.jar

[CVE-2017-18640](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-18640): Denial-of-service attackers were possible because of XXL attacks. (Booth et. al., 2015)

Dependency: spring-boot-2.2.4.RELEASE.jar

[CVE-2022-27772](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-27772): Temporary hijacking of the directory was possible with this version. (Booth et. al., 2015)

Dependency: spring-core-5.2.3.RELEASE.jar

[CVE-2022-22965](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965): This version is vulnerable to remote code execution by data binding. (Booth et. al., 2015)

[CVE-2021-22118](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22118): Multiple versions were susceptible to privilege escalation. (Booth et. al., 2015)

[CVE-2020-5421](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-5421): Prior protections against RFD attacks could be bypassed. (Booth et. al., 2015)

[CVE-2022-22950](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22950): Denial-of- service attacks were possible based on a given user input. (Booth et. al., 2015)

[CVE-2022-22971](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22971): Authenticated users could create Denial-of-service situations. (Booth et. al., 2015)

[CVE-2022-22968](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22968): Due to case sensitivity, some input fields weren’t properly protected. (Booth et. al., 2015)

[CVE-2022-22970](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22970): File upload handling could potentially cause denial-of-service attacks. (Booth et. al., 2015)

[CVE-2021-22060](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22060): Log entries were susceptible to malicious user inputs. (Booth et. al., 2015)

[CVE-2021-22096](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22096): Vulnerable to malicious user inputs, additional log entries could be (Booth et. al., 2015)made.

Dependency: spring-web-5.2.3.RELEASE.jar

[CVE-2016-1000027](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000027): Dependent on library configuration, remote code execution attacks could potentially happen. (Booth et. al., 2015)

[CVE-2022-22965](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965): This version is vulnerable to remote code execution by data binding. (Booth et. al., 2015)

[CVE-2021-22118](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22118): Multiple versions were susceptible to privilege escalation. (Booth et. al., 2015)

[CVE-2020-5421](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-5421): Prior protections against RFD attacks could be bypassed. (Booth et. al., 2015)

[CVE-2022-22950](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22950): Denial-of- service attacks were possible based on a given user input. (Booth et. al., 2015)

[CVE-2022-22971](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22971): Authenticated users could create Denial-of-service situations. (Booth et. al., 2015)

[CVE-2022-22968](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22968): Due to case sensitivity, some input fields weren’t properly protected. (Booth et. al., 2015)

[CVE-2022-22970](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22970): File upload handling could potentially cause denial-of-service attacks. (Booth et. al., 2015)

[CVE-2021-22060](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22060): Log entries were susceptible to malicious user inputs. (Booth et. al., 2015)

[CVE-2021-22096](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22096): Vulnerable to malicious user inputs, additional log entries could be made. (Booth et. al., 2015)

Dependency: tomcat-embed-core-9.0.30.jar:

[CVE-2020-1938](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1938): Due to potential remote code execution, Apache JServe protocol users should be concerned of inbound connections. (Booth et. al., 2015)

[CVE-2020-11996](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-11996): With specialized sequences, an attacker could cause a denial-of-service situation. (Booth et. al., 2015)

[CVE-2020-13934](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13934): Due to memory leakage, a denial-of-service attack could occur. (Booth et. al., 2015)

[CVE-2020-13935](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13935): Unvalidated payload length could create infinite loops or Dos attacks. (Booth et. al., 2015)

[CVE-2020-17527](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-17527): This version allowed potential information leakage with reused HTTP request headers. (Booth et. al., 2015)

[CVE-2021-25122](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25122): Malicious users could see parts of other users request body leaking information. (Booth et. al., 2015)

[CVE-2021-41079](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-41079): Due to unvalidated inbound TLS packets, infinite loops and Dos attacks could occur. (Booth et. al., 2015)

[CVE-2022-29885](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-29885): Misguided user information was given for Encryptinterceptor. (Booth et. al., 2015)

[CVE-2020-9484](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9484): Threat of remote code execution is possible with a few preexisting conditions. (Booth et. al., 2015)

[CVE-2021-25329](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25329): Given a specific, yet highly improbable configuration, remote code execution is possible. (Booth et. al., 2015)

[CVE-2021-30640](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-30640): Variations of a valid login could allow malicious users to gain access. (Booth et. al., 2015)

[CVE-2022-34305](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-34305): Not filtering user supplied data allowed attackers to utilize cross site scripting in this version of Apache Tomcat. (Booth et. al., 2015)

[CVE-2021-24122](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-24122): Potential JSP source code leakage could occur while transferring assets of the NTFS file system. (Booth et. al., 2015)

[CVE-2021-33037](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-33037): Incorrect parsing of HTTP request headers lead to request smuggling. (Booth et. al., 2015)

[CVE-2019-17569](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2019-17569): Improper handling of invalid transfer-encoding headers lead to request smuggling attacks. (Booth et. al., 2015)

[CVE-2020-1935](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1935): Parsing errors allowed request smuggling to happen when the parsed header would pass as a valid header. (Booth et. al., 2015)

[CVE-2020-13943](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13943): Inaccurate responses could occur from surpassed simultaneous streams limits. (Booth et. al., 2015)

Dependency: tomcat-embed-websocket-9.0.30.jar

[CVE-2020-1938](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1938): Due to potential remote code execution, Apache JServe protocol users should be concerned of inbound connections. (Booth et. al., 2015)

[CVE-2020-8022](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-8022): Inaccurate default permissions left Apache Tomcat terminals vulnerable. (Booth et. al., 2015)

[CVE-2020-11996](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-11996): With specialized sequences, an attacker could cause a denial-of-service situation. (Booth et. al., 2015)

[CVE-2020-13934](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13934): Due to memory leakage, a denial-of-service attack could occur. (Booth et. al., 2015)

[CVE-2020-13935](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13935): Unvalidated payload length could create infinite loops or Dos attacks. (Booth et. al., 2015)

[CVE-2020-17527](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-17527): This version allowed potential information leakage with reused HTTP request headers. (Booth et. al., 2015)

[CVE-2021-25122](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25122): Malicious users could see parts of other users request body leaking information. (Booth et. al., 2015)

[CVE-2021-41079](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-41079): Due to unvalidated inbound TLS packets, infinite loops and Dos attacks could occur. (Booth et. al., 2015)

[CVE-2022-29885](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-29885): Misguided user information was given for Encryptinterceptor. (Booth et. al., 2015)

[CVE-2020-9484](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9484): Threat of remote code execution is possible with a few preexisting conditions. (Booth et. al., 2015)

[CVE-2021-25329](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25329): Given a specific, yet highly improbable configuration, remote code execution is possible. (Booth et. al., 2015)

[CVE-2021-30640](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-30640): Variations of a valid login could allow malicious users to gain access. (Booth et. al., 2015)

[CVE-2022-34305](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-34305): Not filtering user supplied data allowed attackers to utilize cross site scripting in this version of Apache Tomcat. (Booth et. al., 2015)

[CVE-2021-24122](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-24122): Potential JSP source code leakage could occur while transferring assets of the NTFS file system. (Booth et. al., 2015)

[CVE-2021-33037](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-33037): Incorrect parsing of HTTP request headers lead to request smuggling. (Booth et. al., 2015)

[CVE-2019-17569](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2019-17569): Improper handling of invalid transfer-encoding headers lead to request smuggling attacks. (Booth et. al., 2015)

[CVE-2020-1935](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1935): Parsing errors allowed request smuggling to happen when the parsed header would pass as a valid header. (Booth et. al., 2015)

[CVE-2020-13943](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13943): Inaccurate responses could occur from surpassed simultaneous streams limits. (Booth et. al., 2015)

## 5. Mitigation Plan

Due to the fact that a majority of the vulnerable areas of concern come from outdated APIs and dependencies, the best mitigation for this would be to ensure the updated version of all dependencies are utilized within this project. Doing so would ensure that all of the known vulnerabilities that stem from the outdated packages would be patched and thus no longer susceptible to such attacks. Any additional vulnerabilities can be verified to be fixed by cross checking the dependencies in the project with the National Vulnerabilities Database, a list of known vulnerabilities. To secure the vulnerable inputs that have no validation, a wrapper method could and should be used around all native methods. This wrapper method ensures only valid arguments and secure checks are made prior to returning the validated values. (Mohindra & Rozenau, 2021) Mitigation for the lack of cryptography, can be accomplished by implementing the HTTPS protocol within the RestServiceApplication.java. Using POST requests rather than GET requests is more secure in nature, restraining sensitive information by passing it in the request body rather than the request header. (Manico et al., 2015) Code error mitigations include preventing executable code, or sensitive dependency information from reaching the code logs, especially if the logs are not encrypted. Lastly, the mitigation for poor code quality, aka poor memory management with the input validation, could be fixed by ensuring the input is within a limited character amount; this will ensure the memory is not overloaded and thus the application is more secure. Implementing these mitigations will help Artemis Financial to have a more robust and secure web application.

References:

Booth, H., Rike, D. and Witte, G. (2013), The National Vulnerability Database (NVD): Overview, ITL Bulletin, National Institute of Standards and Technology, Gaithersburg, MD, [online], https://tsapps.nist.gov/publication/get\_pdf.cfm?pub\_id=915172 (Accessed July 13, 2022)

*Data security laws: State government*. Data Security Laws | State Government. (2020, February 14). Retrieved July 13, 2022, from https://www.ncsl.org/research/telecommunications-and-information-technology/data-security-laws-state-government.aspx

Manico, J., Detlefsen, A., & Kenan, K. (2015). *Iron-clad Java*. O'Reilly Online Learning. Retrieved July 13, 2022, from <https://learning.oreilly.com/library/view/iron-clad-java/9780071835886/ch01.html#ch01lev2sec3>

Mohindra, D., & Rozenau, M. (2021, May 18). *JNI00-J. Define wrappers around native methods*. JNI00-J. Define wrappers around native methods - SEI CERT Oracle Coding Standard for Java - Confluence. Retrieved July 14, 2022, from https://wiki.sei.cmu.edu/confluence/display/java/JNI00-J.+Define+wrappers+around+native+methods#:~:text=Defining%20a%20wrapper%20method%20facilitates,only%20by%20a%20wrapper%20method.