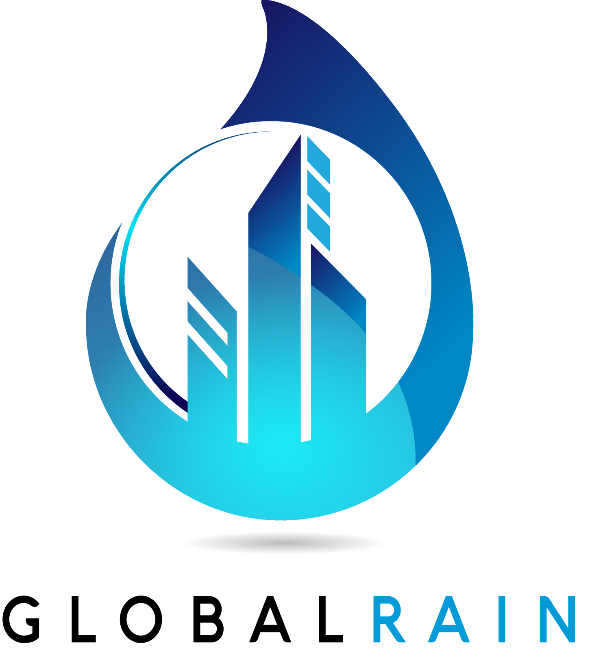
****

# CS 305 Project One

**Artemis Financial Vulnerability Assessment Report**

Table of Contents

[Document Revision History 3](#_Toc32574607)

[Client 3](#_Toc32574608)

[Instructions 3](#_Toc32574609)

[Developer 4](#_Toc32574610)

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

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

[3. Manual Review 4](#_Toc32574613)

[4. Static Testing 4](#_Toc32574614)

[5. Mitigation Plan 4](#_Toc32574615)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **3/16/2022** | **Aaron Shipley** |  |

## 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

Aaron Shipley

## 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 Financial is a financial consulting company that handles customer information about savings, insurance, retirement plans, and other sensitive information. Secure communication is essential for Artemis Financial. Secure communication between the client and business should be of critically high importance. Vulnerabilities in communication can be exploited in various ways, one being SQL injections. According to Manico (2015), “SQL injection is by far the most dangerous vulnerability impacting online applications today (Chapter 2).” Providing secure communication can prevent vulnerabilities within the communication.

Artemis Financials' software is a RESTful web application interface (API), so there is potential for having international clients seeking consultation. Unfortunately, with this potential of international transactions comes more potential threats. The type of threats is not specific to international traffic, but the volume of user access increases with a worldwide distribution of the software system.

There are government restrictions to consider when providing secure communication. The restrictions for the U.S alone should be considered in all aspects when providing secure communication. The Sarbanes Oxley Act (2022) states that particular companies must provide audits of secure data to ensure proper measures are being used to ensure the security of said data. The Federal Trade Commission Gramm-Leach-Biley Act (2002) states that companies who offer consumers financial consultation, loans, or insurance must explain their financial sharing strategies to consumers and safeguard sensitive data. Under the FAST Act (2015), companies meeting specific criteria will not have to comply with the GLBA. The GLBA is looked at as a benchmark for all companies, no matter the mandate to comply with this act.

There are numerous threats in many forms. According to OWASP (2021), the top ten web security threats are as follows:

* Broken Access Control
* Cryptographic Failures
* Injection
* Insecure Design
* Security Misconfiguration
* Vulnerable and Outdated Components
* Identification and Authentication Failures
* Software and Data Integrity Failures
* Security Logging and Monitoring Failures
* Server-Side Request Forgery

Modernization requirements must be considered in developing secure communication between the client and Artemis Financial. Items to consider in modernization ensure that the libraries and plugins used within the software are available in the latest versions. This modernization will allow the software to obtain the latest patches or actions based on typical vulnerabilities or dependencies within the program. Additionally, ensuring the latest versions of the two-factor authorization will help protect the user and business from potential attacks to an outdated system.

## 2. Areas of Security

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

**Input Validation –** In the code, we have a command input function. With this, all input must be severely scrutinized through validation controls. This validation will secure our system and prevent potential injection attacks. According to OWASP (2021), any input validation done on the client must be done on the server as well. In addition, because the client will have the potential to upload sensitive files (401K, 1098 tax form, loan documents, etc.), the system should verify that the file extension is of a valid type and be under a certain maximum size constraint (OWASP, 2021).

**API –** Artemis Financial uses a RESTful API. Due to this, outside access is allowed. We will need to ensure no unintended access at the API level to prevent potential attacks. It may be possible to work with input validation at this level. RedHat (2019) discusses the importance of API security in a REST system using a TLS. The TLS will keep internet connections private and ensure that data transferred between the client and server is encrypted.

**Cryptography –** Due to the nature of the data transferred, it is required of Artemis Financial to ensure data is encrypted. According to the FFIEC Handbook (2022), financial data stored by financial institutions must be encrypted.

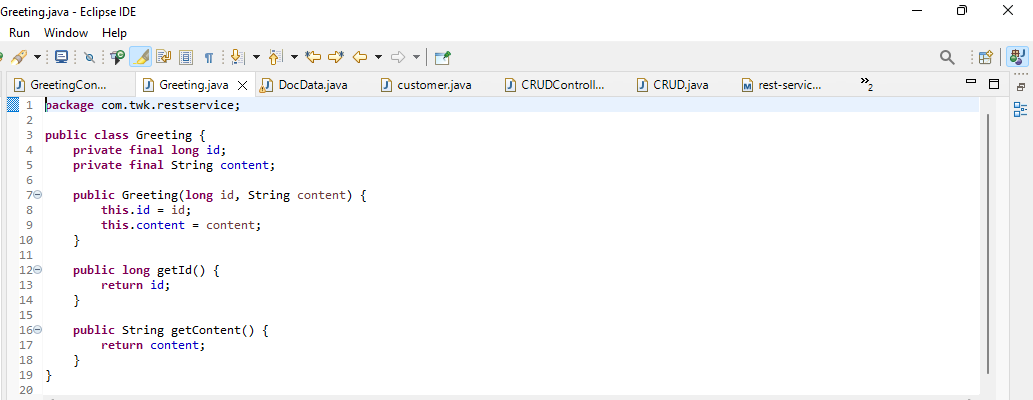
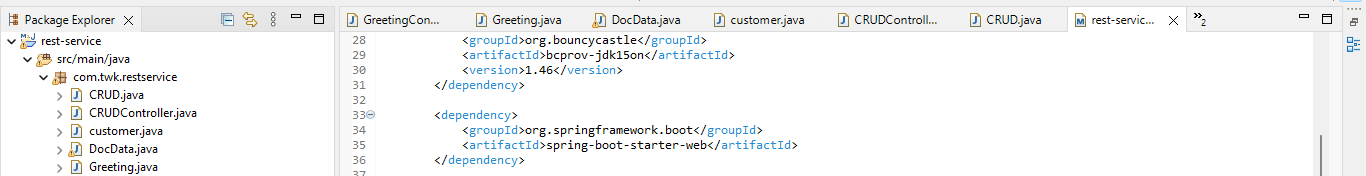
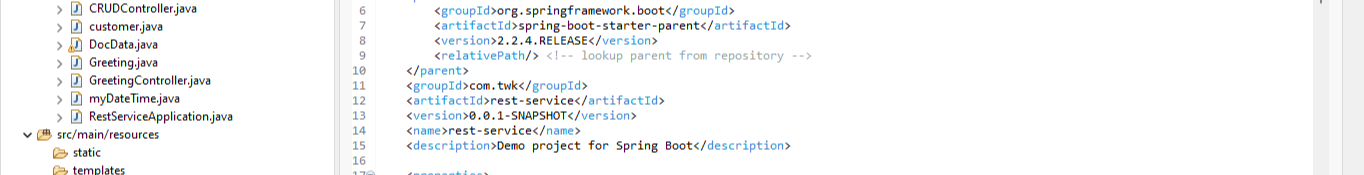
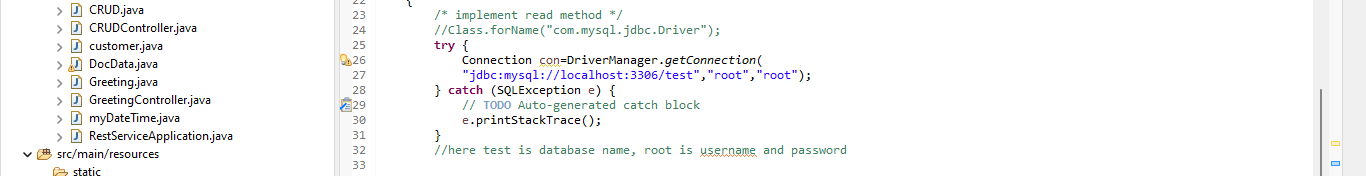
**Client/Server -** Due to API having outside access, proper security between the client and server is crucial to protect this data. Having valid certificates to protect the client and server during the HTTP request will help prevent potential attacks such as supply chain attacks and injection-based attacks.

**Code Error –** Artemis Financial must know how the system will handle unexpected inputs. Having checked on input validation and API will help check that all previous methods to the input will be met. This check will allow our system to handle numerous input command types.

**Code Quality -** Industry standards must be followed when creating the code. Precise notation and following standards within the Spring framework will be necessary to allow the team to understand and interpret the code being created clearly.

## 3. Manual Review

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

* In the greeting.java file, the input is not being validated prior to being added to string "content" on line 17. This formatting issue can lead to unauthorized access to the string. This issue will also allow potentially harmful information to be included with the string. 
* On line 16 of the Greeting.java file, the Get method is used to transfer data. However, this method is the less secure way of transferring the data. With GET, according to Manico (2015), the data is stored within the URL, which can leak into the browser history. 
* Line 30 of the rest-service file has the version of bouncy castle being used as 1.46. This is an outdated version, with the latest version, according to BouncyCastle.org (Wich, 2021), is 2.1.1. This update was released in November of 2021. 
* The Spring boot version in use in the rest-service file is 2.2.4. This version can be seen on line 8 of the file. However, this version is outdated and will need to be updated to the latest version, according to Spring framework (2022), which is 2.6.4.
* On line 27 of the DocData file, we see the root username and root password are using the root user. This opens up vulnerabilities. The root password could be manipulated and guessed through brute force. The data access method in this file is to access the data containing the location of the data, which is the username and password. 

## 4. Static Testing

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

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



* [**bcprov-jdk15on-1.46.jar**](file:///C:\Users\Aaron\OneDrive\Desktop\rest-service\target\dependency-check-report.html#l1_991c96a4e31e6c19e2b9136c8955bd423f2dc4c7)
* **(Severity: Unkown)**
* The following list is the CVE codes listed under the dependency according to the dependency check above:
* [**CVE-2013-1624**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-1624)**, CVE-2015-6644,**[**CVE-2015-7940**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-7940)**,** [**CVE-2016-1000338**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000338)**,** [**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-2016-1000342**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000342)**,** [**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-2020-15522**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-15522)
* The vulnerability listed pertains to Bouncy Castle Java implementation. Bouncy Castle is a collection of APIs used in cryptography and algorithms within Java.
* The solution for fixing this vulnerability would be to update to the latest version of Bouncy Castle. The latest version, according to Wich (2021), is Bouncy Castle 2.1.1. The version in use is 1.46.
* [**hibernate-validator-6.0.18.Final.jar**](file:///C:\Users\Aaron\OneDrive\Desktop\rest-service\target\dependency-check-report.html#l3_7fd00bcd87e14b6ba66279282ef15efa30dd2492)
* **(Severity: Medium)**
* The CVE for the vulnerability listed is as follows:
* [**CVE-2020-10693**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-10693)
* The vulnerability listed pertains to Hibernate Validator and the issue with input sanitation issues within the version being used. Hibernate Validator is an input validator comparing the input against constraints put in place. The solution for fixing this vulnerability is to update Hibernate Validator to the latest version. According to Hibernate (2022), the latest version is 7.0.3. The version being used in the code is version 6.0.18.
* [jackson-databind-2.10.2.jar](file:///C:\Users\Aaron\OneDrive\Desktop\rest-service\target\dependency-check-report.html#l5_0528de95f198afafbcfb0c09d2e43b6e0ea663ec)
* **(Severity : High)**
* The following CVE codes are for the vulnerability listed above. The CVE codes were acquired from the dependency check above.
* [**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)
* The vulnerability listed pertains to the Jackson data bind implementation. According to NVD database (2021), the Jackson data bind is a general data binding functionality used in the Spring framework. The vulnerability listed the issue as the entity expansion was not secured properly. The to this vulnerability is to update to the latest version of Jackson data bind. According to MVN repository (2022), the latest version is 2.13.2. the version being used in the code is 2.10.2.
* [log4j-api-2.12.1.jar](file:///C:\Users\Aaron\OneDrive\Desktop\rest-service\target\dependency-check-report.html#l10_a55e6d987f50a515c9260b0451b4fa217dc539cb)
* **(Severity : Low)**
* The CVE for the vulnerability listed is as follows:
* [**CVE-2020-9488**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9488)
* The vulnerability listed pertains to Apache’s log4 API library. Log4 API is used to log information in order to help applications run smoothly and to determine what is happening within the application. The vulnerability pertains to an invalid certificate due to an older version of the library being used. According to MVN repository (02/2022), the latest version released is 2.17.2. The version being used in the code is 2.12.1
* [snakeyaml-1.25.jar](file:///C:\Users\Aaron\OneDrive\Desktop\rest-service\target\dependency-check-report.html#l14_8b6e01ef661d8378ae6dd7b511a7f2a33fae1421)
* **(Severity : High)**
* The CVE for the vulnerability listed is as follows:
* [**CVE-2017-18640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-18640)
* The vulnerability in question pertains to the snakeyaml. This is used to parse a mapped object to a custom java object after reading a YAML file. The issue listed allows entity expansion. This issue can be fixed by updating to the latest version of snakeyaml. According to MVN repository (2021), the latest version is 1.30. The version used in the code is 1.25.
* [spring-aop-5.2.3.RELEASE.jar](file:///C:\Users\Aaron\OneDrive\Desktop\rest-service\target\dependency-check-report.html#l15_9cdd9a1dd636331767fffcc7fe49a7bb00e7b34b)
* **(Severity: High)**
* The following list is the CVE codes listed under the dependency according to the dependency check above:
* [**CVE-2020-5421**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-5421)**,** [**CVE-2021-22060**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22060)**,** [**CVE-2021-22096**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22096)**,** [**CVE-2021-22118**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22118)
* The vulnerability listed pertains to the Spring AOP in the Spring framework. It is focused on the Aspect-Oriented Programming in Spring. The issue within the spring-aop is due to the version being outdated enabling the version to be open for RFD attacks (NVD, 2022). The solution to this vulnerability is to update to the latest version of spring-aop. According to MVN repository (03/2022), the latest version available is 5.3.17. The version being used is 5.2.3.
* [spring-core-5.2.3.RELEASE.jar](file:///C:\Users\Aaron\OneDrive\Desktop\rest-service\target\dependency-check-report.html#l17_3734223040040e8c3fecd5faa3ae8a1ed6da146b)
* **(Severity: High)**
* The following list is the CVE codes listed under the dependency according to the dependency check above:
* [**CVE-2020-5421**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-5421)**,** [**CVE-2021-22060**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22060)**,** [**CVE-2021-22096**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22096)**,** [**CVE-2021-22118**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22118)
* The spring core dependency listed pertains to the Spring core framework. The main container of the Spring framework, it helps provide core infrastructure to Java projects. The issue listed in the dependency pertains to the version of spring core being outdated. The latest version, according to Spring (2022) is 5.3.16. The version in use in the code is 5.2.3.
* [tomcat-embed-core-9.0.30.jar](file:///C:\Users\Aaron\OneDrive\Desktop\rest-service\target\dependency-check-report.html#l18_ad32909314fe2ba02cec036434c0addd19bcc580)
* **(Severity: Critical)**
* The following list is the CVE codes listed under the dependency according to the dependency check above:
* [**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-2020-9484**](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-2021-42340**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-42340)
* The vulnerability within the dependency listed pertains to the embedded tomcat library. The embedded tomcat will help run the web application like a normal java application. The issue with the embedded tomcat core has invalid transfer encoding due to an outdated version of tomcat. According to MVN repository (2022), the latest version of tomcat embed core is 10.1. The version in use is 9.0.30.
* [tomcat-embed-websocket-9.0.30.jar](file:///C:\Users\Aaron\OneDrive\Desktop\rest-service\target\dependency-check-report.html#l20_33157f6bc5bfd03380ebb5ac476db0600a04168d)
* **(Severity: Critical)**
* The following list is the CVE codes listed under the dependency according to the dependency check above:
* [**CVE-2019-17569**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2019-17569)**,** [**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-2021-42340**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-42340)
* The tomcat embed web socket dependency listed has issues pertaining to the version being used within the code. The version being used is 9.0.30. According to the MVN repository (2022) the latest version available is 10.1. The solution for resolving this issue is to upload the latest version.

## 5. Mitigation Plan

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

* Update Spring framework to the latest version to 5.3.8. Reference CVE and NVD database for solutions to dependencies. Reference Spring web page to verify the latest version.
* Ensure inputs are validated before formatting it to a string. Confirm the length and character list of the string to fit requirements.
* Use POST method in place of the current GET method within the Greeting.java file. This will keep the data from being stored within the URL and leaking into the browser history. According to Manico (2015), this is the more secure option to use instead of GET.
* Update bouncy castle version from 1.4.6 to the latest version. According to Bouncy Castle (2022), the latest version is 2.1.1, which was released in November of 2021.
* Update Spring boot version from the outdated 2.2.4 version being used. Spring framework (2022) shows the latest Spring boot version to be 2.6.4.
* In order to prevent brute force attacks from having greater success, implementing a strong mixture of letters, numbers, and symbols with constraints on character length will prevent attacks from being as successful. The limitations of character length in the username and password will dwindle the ability to parse in long strings with unauthorized data.
* Incorporate secure coding methods within the code in order to validate input properly. This will eliminate validation issues and errors. It will also limit the authentication errors as well.

**References**

FFIEC. (2020). *Information security*. FFIEC IT Handbook InfoBase. Retrieved March 19, 2022, from <https://ithandbook.ffiec.gov/it-booklets/information-security/ii-information-security-program-management/iic-risk-mitigation/iic19-encryption.aspx>

Hibernate. (2022, March 10). *The Bean Validation Reference Implementation. - hibernate validator*. Hibernate. Retrieved March 25, 2022, from https://hibernate.org/validator/

Leahy, M. (2010, September 15). *S. Rept. 111-290 - data breach notification act - congress.gov*. United States Congress Official Website. Retrieved March 17, 2022, from https://www.congress.gov/congressional-report/111th-congress/senate-report/290/1?s=1&r=43

Manico, J., & Detlefsen, A. (2015). *Iron-clad Java building secure web applications*. New York, Ny Mcgraw-Hill Education.

MVNrepository. (2022, March 2). *Maven Repository: Com.fasterxml.jackson.core " jackson-databind*. MavenRepository. Retrieved March 25, 2022, from <https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind>

MVNrepository. (2022, February 27). *Maven Repository: Org.apache.logging.log4j " log4j-api " 2.17.2*. MavenRepository. Retrieved March 25, 2022, from <https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-api/2.17.2>

MVNrepository. (2022, February 9). *Maven Repository: Org.apache.tomcat.embed " tomcat-embed-core*. MavenRepository. Retrieved March 25, 2022, from https://mvnrepository.com/artifact/org.apache.tomcat.embed/tomcat-embed-core

MVNrepository. (2022, March 16). *Maven Repository: Org.springframework " Spring-aop " 5.3.17*. MavenRepository. Retrieved March 25, 2022, from https://mvnrepository.com/artifact/org.springframework/spring-aop/5.3.17

MVNrepository. (2021, December 7). *Maven Repository: Org.yaml " snakeyaml*. MavenRepository. Retrieved March 25, 2022, from https://mvnrepository.com/artifact/org.yaml/snakeyaml

National Institute of Standards in Technology. (2021, October 1)*.* NVD. Retrieved March 17, 2022, from https://nvd.nist.gov/vuln/search

OWASP. (2021, January 10). *Owasp Top Ten*. OWASP. Retrieved March 18, 2022, from https://owasp.org/www-project-top-ten/

SoxLaw. (2022, March 3). *The sarbanes-oxley (sox) act of 2002: Information & Resources*. SoxLaw. Retrieved March 15, 2022, from <https://www.soxlaw.com/>

Spring. (2021, August 7). 1. introduction to spring framework. Retrieved March 25, 2022, from https://docs.spring.io/spring-framework/docs/3.0.x/spring-framework-reference/html/overview.html

Staff, the P. N. O., & This blog is a collaboration between CTO and DPIP staff and the AI Strategy team. (2022, February 11). *Gramm-Leach-Bliley Act*. Federal Trade Commission. Retrieved March 12, 2022, from https://www.ftc.gov/business-guidance/privacy-security/gramm-leach-bliley-act

Wich, T. (2021, November 29). *The legion of the bouncy castle*. bouncycastle.org. Retrieved March 17, 2022, from https://www.bouncycastle.org/releasenotes.html