****

# 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** | **9/13/21** | **Benjamin Verrill** |  |

## 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

Ben Verrill

## 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 consulting company that provides customized financial plans for their patrons. They make use of a REST API for their global customers and want to update their security practices to ensure protection against external threats. Providing secure communications to companies such as Artemis Financial is critical as they host very sensitive PII for their clients. Such information includes client information, account information, and account transfers. Interactions with their software need to be protected as this information being stolen could lead to negative consequences for Artemis Financial’s clients as well as the company’s revenue and brand.

There are governmental restrictions about secure communications to consider as well. If Artemis Financial does more than $25 million in revenue, they are subject to The California Consumer Privacy Act of 2018 if they do business with clients in California (California State Government, 2018). This requires them to encrypt data and provide reasonable security procedures or they’re liable to be sued. Another concerns the European Banking Authority who requires any organizations who store, process, or transmit sensitive payment data to provide end to end encryption for such communications (European Banking Authority, 2014). The European Union also has a General Data Protection Regulation law that states that anyone doing business with people within the EU must provide safeguards, including encryption, for the data they collect otherwise they are subject to fines and penalties (European Union, 2019). There are many other laws to consider, but here is a subset to think about.

External threats can come from a variety of vectors. These can include nation states, groups, or individuals who work to achieve goals that can be monetary, political, or just to cause chaos. As the IT landscape evolves, new ways open for these threat actors to take advantage of to achieve their goals. These include things like migrating infrastructure to the cloud, the use of open-source libraries, or dispersed work forces. These areas expose organizations to additional risk that they may have little to no control over. Open-source libraries can introduce vulnerabilities in code from lax security implementation. Misconfigurations in cloud environments can allow the wrong people access to areas of the infrastructure they shouldn’t have permissions to. A dispersed workforce can make phishing and other, similar styles of attacks easier as there are less interactions with people within the organization face to face. All of these things need to be taken into account as companies modernize their cyber security strategies.

## 2. Areas of Security

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

* Secure API – The application relies on API calls to retrieve and update data for Artemis Financials’ clients. Because of this, the API needs to be configured so as to maintain the safety and integrity of the client’s data. This includes areas such as ensuring only authorized and authenticated users are making calls to solely their own data, validating any input to prevent unauthorized use of characters, and limiting the data exposed by the calls. These types of preventative measures can limit the risk of injection attacks such as SQL Injection, which is untrusted data sent as queries to web applications that can harvest sensitive information.
* Input Validation – Since the application relies on input from the users for making API calls, input needs to be vetted before used within the app to lower risk of injection attacks. This includes limiting the length of input based on use case, as well as limiting the characters and even expressions allowed as input.
* Cryptography – Much of the information presented in the application is sensitive to the user. This includes personal information, bank information, and even money transfers. This data needs to be encrypted so as to prevent unauthorized viewers from accessing it. Earlier, we also mentioned specific governmental agencies that require this type of information to be encrypted.
* Code Quality - Code needs to be clear, concise, and provide only the functionality that input for API calls require so as to not introduce additional vulnerabilities. Code should also be error free with comments supplied for context.

## 3. Manual Review

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

* No authentication into the app. This exposes the API to input from anyone.
* No input validation for API requests. I was able to use a multitude of different characters at various lengths that always worked.
* Because of no authentication, the application’s database can easily be exposed through the CRUDController, which creates and sends database information through the DocData class.
* Customer class and myDateTime class are created but not used. Customer class exposes constructor despite holding sensitive information.
* Despite the BouncyCastle dendency being added to the application, it is never used. Therefore, there is no encryption being done within the application.
* There are no permission restrictions. Any user can perform any action within the app. Least privilege is not applied.

## 4. Static Testing

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

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

|  |  |  |
| --- | --- | --- |
| **Dependency** | **Vulnerability IDs** | **Notes** |
| [bcprov-jdk15on-1.46.jar](https://usc-word-edit.officeapps.live.com/we/wordeditorframe.aspx?ui=en%2DUS&rs=en%2DUS&wopisrc=https%3A%2F%2Fsnhu-my.sharepoint.com%2Fpersonal%2Fbenjamin_verrill_snhu_edu%2F_vti_bin%2Fwopi.ashx%2Ffiles%2F42d3590f661e41d28e8931dc2af30679&wdnewandopenct=1631536090186&wdprevioussession=b433f6f7-cf1e-4662-8fb1-376f302a7b9d&wdorigin=OFFICECOM-WEB.MAIN.UPLOAD&wdenableroaming=1&mscc=1&wdodb=1&hid=5A66EF9F-00BC-C000-4C6D-95C2C9485C02&wdhostclicktime=1631536090186&jsapi=1&jsapiver=v1&newsession=1&corrid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&usid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&sftc=1&mtf=1&sfp=1&instantedit=1&wopicomplete=1&wdredirectionreason=Unified_SingleFlush&preseededsessionkey=f04a66d5-39b0-e7a0-a5ba-2736d68316e5&preseededwacsessionid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&rct=Medium&ctp=LeastProtected#l1_991c96a4e31e6c19e2b9136c8955bd423f2dc4c7) | cpe:2.3:a:bouncycastle:bouncy-castle-crypto-package:1.46:\*:\*:\*:\*:\*:\*:\* cpe:2.3:a:bouncycastle:bouncy\_castle\_crypto\_package:1.46:\*:\*:\*:\*:\*:\*:\* [cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Abouncycastle&cpe_product=cpe%3A%2F%3Abouncycastle%3Alegion-of-the-bouncy-castle-java-crytography-api&cpe_version=cpe%3A%2F%3Abouncycastle%3Alegion-of-the-bouncy-castle-java-crytography-api%3A1.46) cpe:2.3:a:bouncycastle:the\_bouncy\_castle\_crypto\_package\_for\_java:1.46:\*:\*:\*:\*:\*:\*:\* | Bouncy Castle is a cryptography API for Java and C#. There are 17 CVEs associated with it. These open the application up to vulnerabilites from TLS misconfigurations, data leaks, improper encoding validation, weak private/public key generation, and more. Mitigation of these vulnerabilites is achieve by updating Bouncy Castle to the most recent version. |
| [hibernate-validator-6.0.18.Final.jar](https://usc-word-edit.officeapps.live.com/we/wordeditorframe.aspx?ui=en%2DUS&rs=en%2DUS&wopisrc=https%3A%2F%2Fsnhu-my.sharepoint.com%2Fpersonal%2Fbenjamin_verrill_snhu_edu%2F_vti_bin%2Fwopi.ashx%2Ffiles%2F42d3590f661e41d28e8931dc2af30679&wdnewandopenct=1631536090186&wdprevioussession=b433f6f7-cf1e-4662-8fb1-376f302a7b9d&wdorigin=OFFICECOM-WEB.MAIN.UPLOAD&wdenableroaming=1&mscc=1&wdodb=1&hid=5A66EF9F-00BC-C000-4C6D-95C2C9485C02&wdhostclicktime=1631536090186&jsapi=1&jsapiver=v1&newsession=1&corrid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&usid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&sftc=1&mtf=1&sfp=1&instantedit=1&wopicomplete=1&wdredirectionreason=Unified_SingleFlush&preseededsessionkey=f04a66d5-39b0-e7a0-a5ba-2736d68316e5&preseededwacsessionid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&rct=Medium&ctp=LeastProtected#l3_7fd00bcd87e14b6ba66279282ef15efa30dd2492) | [cpe:2.3:a:redhat:hibernate\_validator:6.0.18:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aredhat&cpe_product=cpe%3A%2F%3Aredhat%3Ahibernate_validator&cpe_version=cpe%3A%2F%3Aredhat%3Ahibernate_validator%3A6.0.18) | [**CVE-2020-10693**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-10693)addresses a vulnerability in Hibernate’s Validator that could allow attackers to bypass input sanitation controls. Mitigation can be achieved through upgrading the dependency. |
| [jackson-databind-2.10.2.jar](https://usc-word-edit.officeapps.live.com/we/wordeditorframe.aspx?ui=en%2DUS&rs=en%2DUS&wopisrc=https%3A%2F%2Fsnhu-my.sharepoint.com%2Fpersonal%2Fbenjamin_verrill_snhu_edu%2F_vti_bin%2Fwopi.ashx%2Ffiles%2F42d3590f661e41d28e8931dc2af30679&wdnewandopenct=1631536090186&wdprevioussession=b433f6f7-cf1e-4662-8fb1-376f302a7b9d&wdorigin=OFFICECOM-WEB.MAIN.UPLOAD&wdenableroaming=1&mscc=1&wdodb=1&hid=5A66EF9F-00BC-C000-4C6D-95C2C9485C02&wdhostclicktime=1631536090186&jsapi=1&jsapiver=v1&newsession=1&corrid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&usid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&sftc=1&mtf=1&sfp=1&instantedit=1&wopicomplete=1&wdredirectionreason=Unified_SingleFlush&preseededsessionkey=f04a66d5-39b0-e7a0-a5ba-2736d68316e5&preseededwacsessionid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&rct=Medium&ctp=LeastProtected#l5_0528de95f198afafbcfb0c09d2e43b6e0ea663ec) | [cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Afasterxml&cpe_product=cpe%3A%2F%3Afasterxml%3Ajackson-databind&cpe_version=cpe%3A%2F%3Afasterxml%3Ajackson-databind%3A2.10.2) cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:\*:\*:\*:\*:\*:\*:\* | [**CVE-2020-25649**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-25649)addresses a vulnerability in the Jackson dependency that exposes the app to data integrity risk through external entity attacks. Mitigation can be achieved by upgrading the dependency. |
| [log4j-api-2.12.1.jar](https://usc-word-edit.officeapps.live.com/we/wordeditorframe.aspx?ui=en%2DUS&rs=en%2DUS&wopisrc=https%3A%2F%2Fsnhu-my.sharepoint.com%2Fpersonal%2Fbenjamin_verrill_snhu_edu%2F_vti_bin%2Fwopi.ashx%2Ffiles%2F42d3590f661e41d28e8931dc2af30679&wdnewandopenct=1631536090186&wdprevioussession=b433f6f7-cf1e-4662-8fb1-376f302a7b9d&wdorigin=OFFICECOM-WEB.MAIN.UPLOAD&wdenableroaming=1&mscc=1&wdodb=1&hid=5A66EF9F-00BC-C000-4C6D-95C2C9485C02&wdhostclicktime=1631536090186&jsapi=1&jsapiver=v1&newsession=1&corrid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&usid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&sftc=1&mtf=1&sfp=1&instantedit=1&wopicomplete=1&wdredirectionreason=Unified_SingleFlush&preseededsessionkey=f04a66d5-39b0-e7a0-a5ba-2736d68316e5&preseededwacsessionid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&rct=Medium&ctp=LeastProtected#l10_a55e6d987f50a515c9260b0451b4fa217dc539cb) | [cpe:2.3:a:apache:log4j:2.12.1:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache&cpe_product=cpe%3A%2F%3Aapache%3Alog4j&cpe_version=cpe%3A%2F%3Aapache%3Alog4j%3A2.12.1) | [**CVE-2020-9488**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9488) addresses a vulnerability with the Apacha Log4J API that exposes the app to man-in-the-middle attacks through improper certificate validation. Mitigation can be achieved by updating the dependency. |
| [snakeyaml-1.25.jar](https://usc-word-edit.officeapps.live.com/we/wordeditorframe.aspx?ui=en%2DUS&rs=en%2DUS&wopisrc=https%3A%2F%2Fsnhu-my.sharepoint.com%2Fpersonal%2Fbenjamin_verrill_snhu_edu%2F_vti_bin%2Fwopi.ashx%2Ffiles%2F42d3590f661e41d28e8931dc2af30679&wdnewandopenct=1631536090186&wdprevioussession=b433f6f7-cf1e-4662-8fb1-376f302a7b9d&wdorigin=OFFICECOM-WEB.MAIN.UPLOAD&wdenableroaming=1&mscc=1&wdodb=1&hid=5A66EF9F-00BC-C000-4C6D-95C2C9485C02&wdhostclicktime=1631536090186&jsapi=1&jsapiver=v1&newsession=1&corrid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&usid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&sftc=1&mtf=1&sfp=1&instantedit=1&wopicomplete=1&wdredirectionreason=Unified_SingleFlush&preseededsessionkey=f04a66d5-39b0-e7a0-a5ba-2736d68316e5&preseededwacsessionid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&rct=Medium&ctp=LeastProtected#l13_8b6e01ef661d8378ae6dd7b511a7f2a33fae1421) | [cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Asnakeyaml_project&cpe_product=cpe%3A%2F%3Asnakeyaml_project%3Asnakeyaml&cpe_version=cpe%3A%2F%3Asnakeyaml_project%3Asnakeyaml%3A1.25) | [**CVE-2017-18640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-18640) addresses a vulnerability to the YAML parser and emitter for Java that exposes the application to a type of DoS attack called the “billion laughs attack”. There are several suggestions for remediating the problem:   * If the YAML is not coming from an untrusted source then it can be ignored as a false positive * Check the quality of the YAML before sending the document to SnakeYAML * Migrate to SnakeYAML Engine * Allow SnakeYAML to use its function of failing early |
| [spring-aop-5.2.3.RELEASE.jar](https://usc-word-edit.officeapps.live.com/we/wordeditorframe.aspx?ui=en%2DUS&rs=en%2DUS&wopisrc=https%3A%2F%2Fsnhu-my.sharepoint.com%2Fpersonal%2Fbenjamin_verrill_snhu_edu%2F_vti_bin%2Fwopi.ashx%2Ffiles%2F42d3590f661e41d28e8931dc2af30679&wdnewandopenct=1631536090186&wdprevioussession=b433f6f7-cf1e-4662-8fb1-376f302a7b9d&wdorigin=OFFICECOM-WEB.MAIN.UPLOAD&wdenableroaming=1&mscc=1&wdodb=1&hid=5A66EF9F-00BC-C000-4C6D-95C2C9485C02&wdhostclicktime=1631536090186&jsapi=1&jsapiver=v1&newsession=1&corrid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&usid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&sftc=1&mtf=1&sfp=1&instantedit=1&wopicomplete=1&wdredirectionreason=Unified_SingleFlush&preseededsessionkey=f04a66d5-39b0-e7a0-a5ba-2736d68316e5&preseededwacsessionid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&rct=Medium&ctp=LeastProtected#l14_9cdd9a1dd636331767fffcc7fe49a7bb00e7b34b) | [cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Apivotal_software&cpe_product=cpe%3A%2F%3Apivotal_software%3Aspring_framework&cpe_version=cpe%3A%2F%3Apivotal_software%3Aspring_framework%3A5.2.3) [cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aspringsource&cpe_product=cpe%3A%2F%3Aspringsource%3Aspring_framework&cpe_version=cpe%3A%2F%3Aspringsource%3Aspring_framework%3A5.2.3) [cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Avmware&cpe_product=cpe%3A%2F%3Avmware%3Aspring_framework&cpe_version=cpe%3A%2F%3Avmware%3Aspring_framework%3A5.2.3) | The CVEs for Spring Aspect Oriented Programming expose the app to two vulnerabilities:   * Exposure to RFD attacks through the use of jsessiosid path parameters. * Exposure to privilege escalation because of WebFlux.   Both can be mitigated by updating the dependency. |
| [spring-core-5.2.3.RELEASE.jar](https://usc-word-edit.officeapps.live.com/we/wordeditorframe.aspx?ui=en%2DUS&rs=en%2DUS&wopisrc=https%3A%2F%2Fsnhu-my.sharepoint.com%2Fpersonal%2Fbenjamin_verrill_snhu_edu%2F_vti_bin%2Fwopi.ashx%2Ffiles%2F42d3590f661e41d28e8931dc2af30679&wdnewandopenct=1631536090186&wdprevioussession=b433f6f7-cf1e-4662-8fb1-376f302a7b9d&wdorigin=OFFICECOM-WEB.MAIN.UPLOAD&wdenableroaming=1&mscc=1&wdodb=1&hid=5A66EF9F-00BC-C000-4C6D-95C2C9485C02&wdhostclicktime=1631536090186&jsapi=1&jsapiver=v1&newsession=1&corrid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&usid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&sftc=1&mtf=1&sfp=1&instantedit=1&wopicomplete=1&wdredirectionreason=Unified_SingleFlush&preseededsessionkey=f04a66d5-39b0-e7a0-a5ba-2736d68316e5&preseededwacsessionid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&rct=Medium&ctp=LeastProtected#l16_3734223040040e8c3fecd5faa3ae8a1ed6da146b) | [cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Apivotal_software&cpe_product=cpe%3A%2F%3Apivotal_software%3Aspring_framework&cpe_version=cpe%3A%2F%3Apivotal_software%3Aspring_framework%3A5.2.3) [cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aspringsource&cpe_product=cpe%3A%2F%3Aspringsource%3Aspring_framework&cpe_version=cpe%3A%2F%3Aspringsource%3Aspring_framework%3A5.2.3) [cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Avmware&cpe_product=cpe%3A%2F%3Avmware%3Aspring_framework&cpe_version=cpe%3A%2F%3Avmware%3Aspring_framework%3A5.2.3) cpe:2.3:a:vmware:springsource\_spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* | These vulnerabilities are in refernce to the Spring Core, and are the same as the above vulnerability for Spring AOS. Mitigation is achieved by updating the dependency. |
| [tomcat-embed-core-9.0.30.jar](https://usc-word-edit.officeapps.live.com/we/wordeditorframe.aspx?ui=en%2DUS&rs=en%2DUS&wopisrc=https%3A%2F%2Fsnhu-my.sharepoint.com%2Fpersonal%2Fbenjamin_verrill_snhu_edu%2F_vti_bin%2Fwopi.ashx%2Ffiles%2F42d3590f661e41d28e8931dc2af30679&wdnewandopenct=1631536090186&wdprevioussession=b433f6f7-cf1e-4662-8fb1-376f302a7b9d&wdorigin=OFFICECOM-WEB.MAIN.UPLOAD&wdenableroaming=1&mscc=1&wdodb=1&hid=5A66EF9F-00BC-C000-4C6D-95C2C9485C02&wdhostclicktime=1631536090186&jsapi=1&jsapiver=v1&newsession=1&corrid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&usid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&sftc=1&mtf=1&sfp=1&instantedit=1&wopicomplete=1&wdredirectionreason=Unified_SingleFlush&preseededsessionkey=f04a66d5-39b0-e7a0-a5ba-2736d68316e5&preseededwacsessionid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&rct=Medium&ctp=LeastProtected#l17_ad32909314fe2ba02cec036434c0addd19bcc580) | [cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache&cpe_product=cpe%3A%2F%3Aapache%3Atomcat&cpe_version=cpe%3A%2F%3Aapache%3Atomcat%3A9.0.30) [cpe:2.3:a:apache\_software\_foundation:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache_software_foundation&cpe_product=cpe%3A%2F%3Aapache_software_foundation%3Atomcat&cpe_version=cpe%3A%2F%3Aapache_software_foundation%3Atomcat%3A9.0.30) [cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache_tomcat&cpe_product=cpe%3A%2F%3Aapache_tomcat%3Aapache_tomcat&cpe_version=cpe%3A%2F%3Aapache_tomcat%3Aapache_tomcat%3A9.0.30) | There are a multitude of CVEs associated with the Core Tomcat implementation. This includes exposure to risk from:   * HTTP Request Smuggling * DOS attacks from specially crafted HTTP/2 sequences that cause the server to become unresponsive for several seconds * DOS attacks from h2c direct connections * DOS attacks from improper validation of payload lengths in WebSocket frames * Unintentional responses sent to users * Information leaks through the re-use of HTTP request header values * Unintended access for malicious actors through the use of the AJP Connector * Exposure of user data to other users because of a duplication of request headers and a limited amount of request body from one request to another. * Attackers being able to use variations of valid user names to bypass the LockOut Realm to authenticate into the server   All of the above vulnerabilities can be mitigated by upgrading the dependency and downloading an OpenSUSE Security update. |
| [tomcat-embed-websocket-9.0.30.jar](https://usc-word-edit.officeapps.live.com/we/wordeditorframe.aspx?ui=en%2DUS&rs=en%2DUS&wopisrc=https%3A%2F%2Fsnhu-my.sharepoint.com%2Fpersonal%2Fbenjamin_verrill_snhu_edu%2F_vti_bin%2Fwopi.ashx%2Ffiles%2F42d3590f661e41d28e8931dc2af30679&wdnewandopenct=1631536090186&wdprevioussession=b433f6f7-cf1e-4662-8fb1-376f302a7b9d&wdorigin=OFFICECOM-WEB.MAIN.UPLOAD&wdenableroaming=1&mscc=1&wdodb=1&hid=5A66EF9F-00BC-C000-4C6D-95C2C9485C02&wdhostclicktime=1631536090186&jsapi=1&jsapiver=v1&newsession=1&corrid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&usid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&sftc=1&mtf=1&sfp=1&instantedit=1&wopicomplete=1&wdredirectionreason=Unified_SingleFlush&preseededsessionkey=f04a66d5-39b0-e7a0-a5ba-2736d68316e5&preseededwacsessionid=2e13bd40-3860-c7df-a9ad-0ad3d40c52e4&rct=Medium&ctp=LeastProtected#l19_33157f6bc5bfd03380ebb5ac476db0600a04168d) | [cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache&cpe_product=cpe%3A%2F%3Aapache%3Atomcat&cpe_version=cpe%3A%2F%3Aapache%3Atomcat%3A9.0.30) [cpe:2.3:a:apache\_software\_foundation:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache_software_foundation&cpe_product=cpe%3A%2F%3Aapache_software_foundation%3Atomcat&cpe_version=cpe%3A%2F%3Aapache_software_foundation%3Atomcat%3A9.0.30) [cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache_tomcat&cpe_product=cpe%3A%2F%3Aapache_tomcat%3Aapache_tomcat&cpe_version=cpe%3A%2F%3Aapache_tomcat%3Aapache_tomcat%3A9.0.30) | This dependency, also a Core Tomcat implementation, has all the same vulnerabilities as tomcat-embed-core-9.0.20.jar with the addition of [**CVE-2020-8022**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-8022)which exposes the server to incorrect default permissions. This can also be mitigated by upgrading the server. |

## 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.

1. Update all dependencies with identified vulnerabilities.
2. Implement authentication to the application. Since the application deals with sensitive information, username and password should not be considered efficient. Include multi-factor authentication.
3. Implement Role Based Access Control. Not all authenticated users should have the ability to do whatever they want within the app. Assign each user a role as they register for the app.
4. Implement Least Privilege. Based on a user’s role, restrict what they’re able to do within the app.
5. Implement input validation for API calls. Currently, the API allows for any input from a browser. This exposes the application to injection attacks. Limit what input the API can receive based on the request being made.
6. Restrict access to database. Currently the API allows direct access to the database through the CRUD class. Implement controls to restrict what calls are allowed to the database.
7. Implement encryption. BouncyCastle has been added to the project but never used. Sensitive user information should be encrypted at rest and in transit.
8. Make unused classes private. Restrict access to classes to prevent exposure to additional avenues of attack by threat actors.

**Resources**

California State Government. (2018, September 24). *Bill text*. Bill Text - SB-1121 California Consumer Privacy Act of 2018. Retrieved September 18, 2021, from <https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180SB1121>.

European Banking Authority. (2014, December 19). FINAL GUIDELINES ON THE SECURITY OF INTERNET PAYMENTS. Retrieved from <https://www.eba.europa.eu/sites/default/documents/files/documents/10180/934179/f27bf266-580a-4ad0-aaec-59ce52286af0/EBA-GL-2014-12%20%28Guidelines%20on%20the%20security%20of%20internet%20payments%29_Rev1.pdf?retry=1>.

European Union. (2019, September 2). *Official legal text*. General Data Protection Regulation (GDPR). Retrieved September 18, 2021, from <https://gdpr-info.eu/>.