# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **[07/21/2024]** | **[Taylor Stapus]** |  |

## Client



## Developer

Taylor Stapus

**1. Interpreting Client Needs**

Artemis Financial prioritizes the value of secure communication. The company will be handling personal/sensitive information to provide financial plans for customers including savings, retirement, investments and insurance. This will mean that the company will have access to confidential information such as business/ personal names and account balances. This type of information is sought after by attackers who are looking for security vulnerabilities to exploit. Using secure communication will ensure that there are no data breaches that could potentially compromise sensitive material.

By being a financial institution, Artemis Financial will be obligated to follow government regulations and restrictions when it comes to information communicated throughout the website application they implement. All threats, current and potential will more than likely be seeking confidential information that can be manipulated elsewhere for financial gain. This could include an attack on the clients as well as an attack on the company itself.

Modern requirement that must be considered will include a secure API, two factor authentication for users looking to use the application and communications should be down through a HTTP because it this is the most secure way to share information from company to person currently.

**2. Areas of Security**

The Vulnerability Assessment Process Flow includes even areas to check for security vulnerabilities. This list includes Input Validation, APIs, Cryptography, Client/Server, Code Error, Code Quality and Encapsulation.

The areas of security that applies specifically to Artemis Financials’ software application are input validation, API’s, code errors and code quality. Every project needs to reflect current best practices to minimize potential errors and security hazards. Also ensuring the quality of code does not include unnecessary line additions is essential. After a quick look at the packages used for this project, it looks like Artemis is using RESTful API. This API will need to be verified for secure communication. APIs also take user inputs so measures will need to be taken to ensure input from users are cleared and validated.

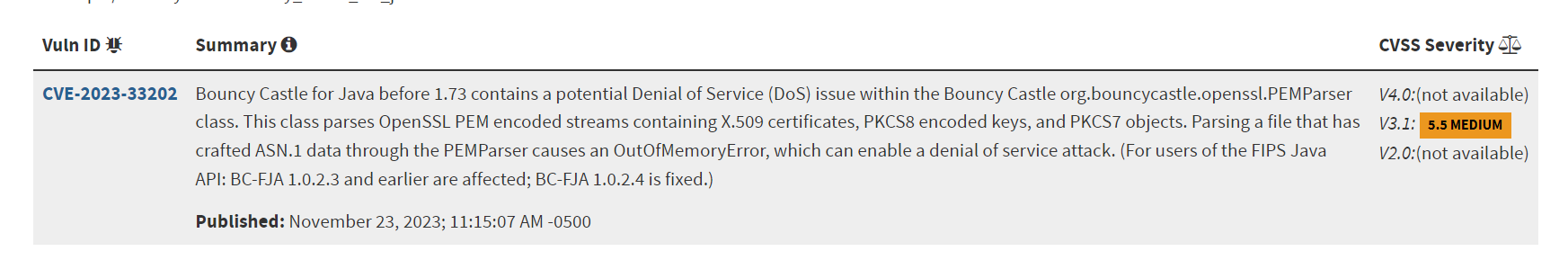
**3. Manual Review**

* Inputs are not validated which can leave the system vulnerable to outside attacks
* There is no authentication system in place to verify users
* The application does not use HTTPS which is recommended when it comes to sharing sensitive information
* Business names are being used as the request parameters within the CRUD Controller class. This can be seen as a leak in personal information if companies are not wanting to disclose that they use Artemis.
* The dependency-check-maven is outdated and cannot run a dependency vulnerability report.
* In the DocData class, the database location is presentable as an output to the user.
* There is a code error where the local variable ‘con’ is not used
* Issue in customer.java where the account balance is not specified as private and could potentially be leaked

**4. Static Testing**

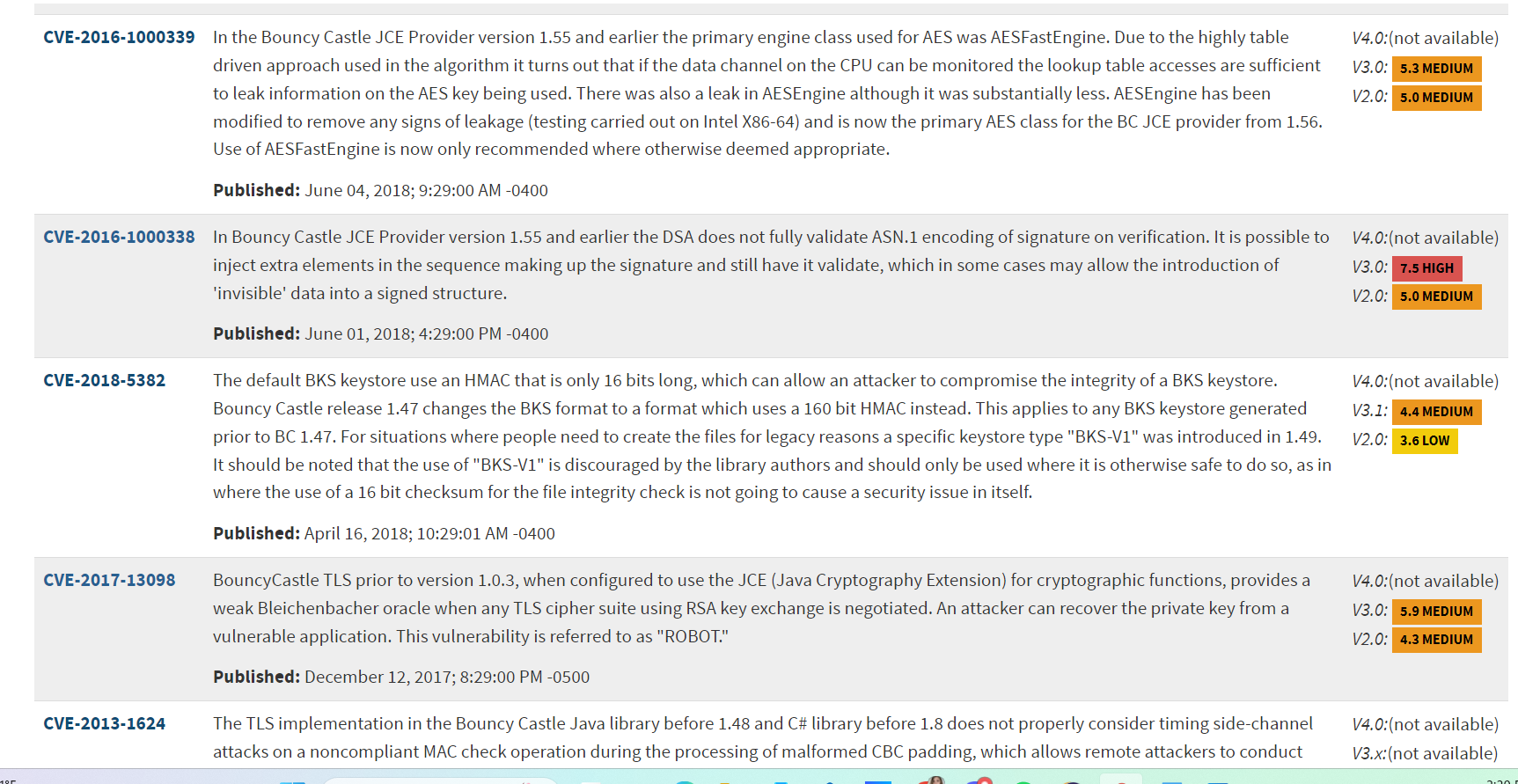
After running static testing, the dependency-check report found 13 vulnerable dependencies which totaled into 135 vulnerabilities detected within them. Below will include screenshots of the dependency issue was located with a description of the vulnerability found, the level of severity and a publication date of when the specific issue was discovered with potential corrections. This information is given through using the National Vulnerability Database (NVD NIST). The reference provides potential solutions by clicking the ID.

1. [bcprov-jdk15on-1.46.jar](file:///C:\Users\16232\eclipse-workspace\rest-service\target\dependency-check-report.html#l1_991c96a4e31e6c19e2b9136c8955bd423f2dc4c7)

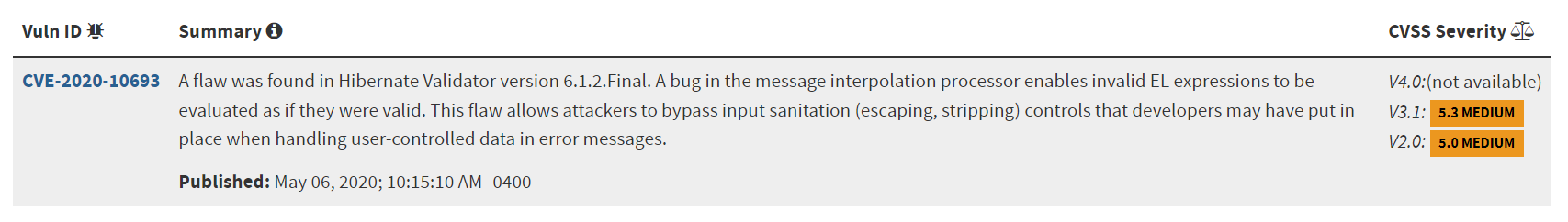


A screenshot of a computer

Description automatically generated



2. [hibernate-validator-6.0.18.Final.jar](file:///C:\Users\16232\eclipse-workspace\rest-service\target\dependency-check-report.html#l3_7fd00bcd87e14b6ba66279282ef15efa30dd2492)

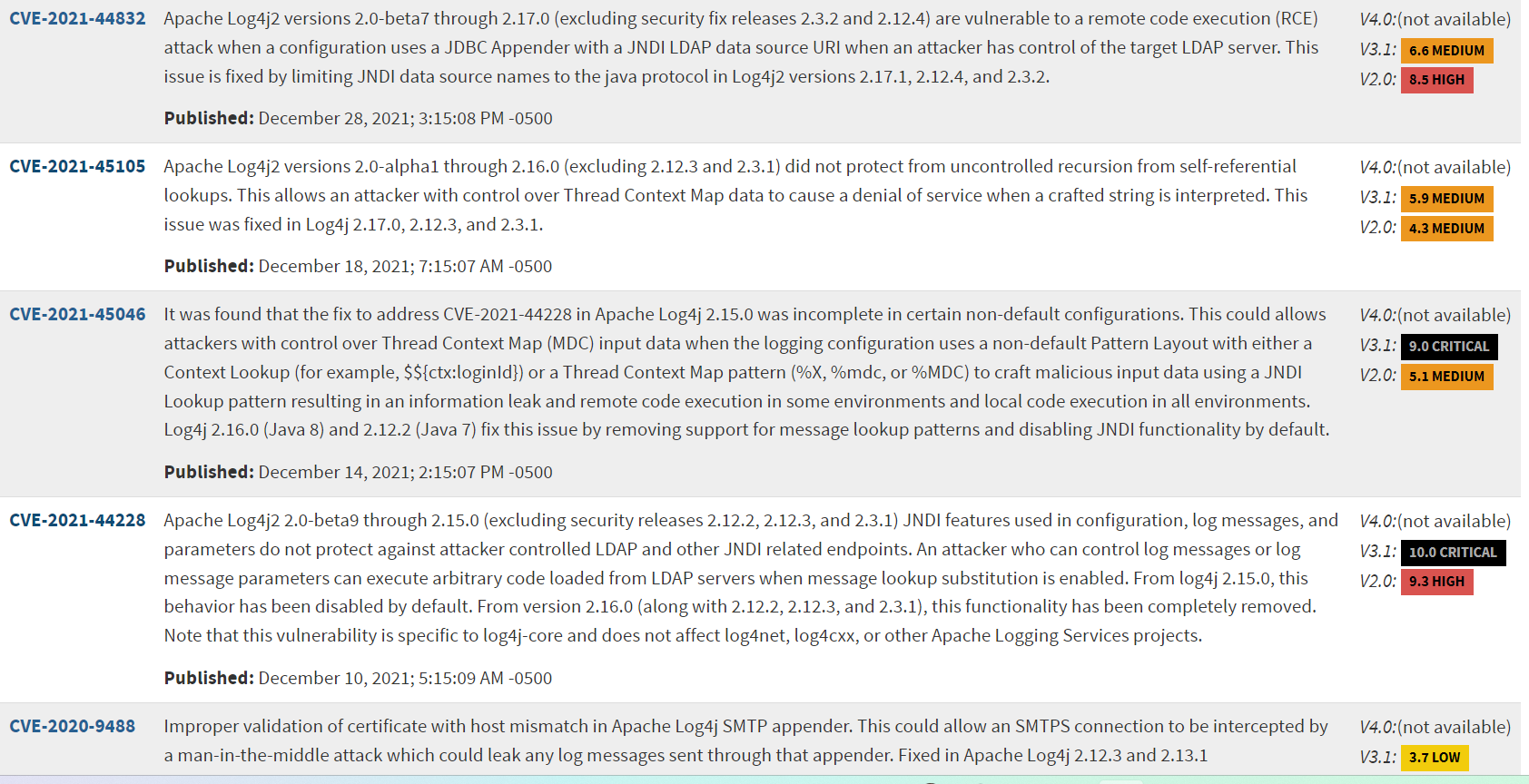


3. [jackson-databind-2.10.2.jar](file:///C:\Users\16232\eclipse-workspace\rest-service\target\dependency-check-report.html#l5_0528de95f198afafbcfb0c09d2e43b6e0ea663ec)

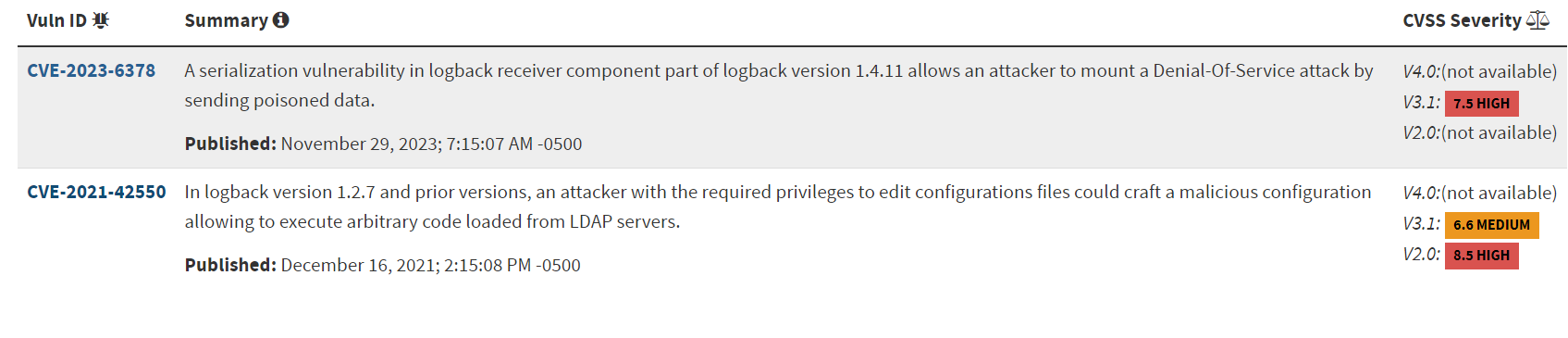
A screenshot of a computer

Description automatically generated

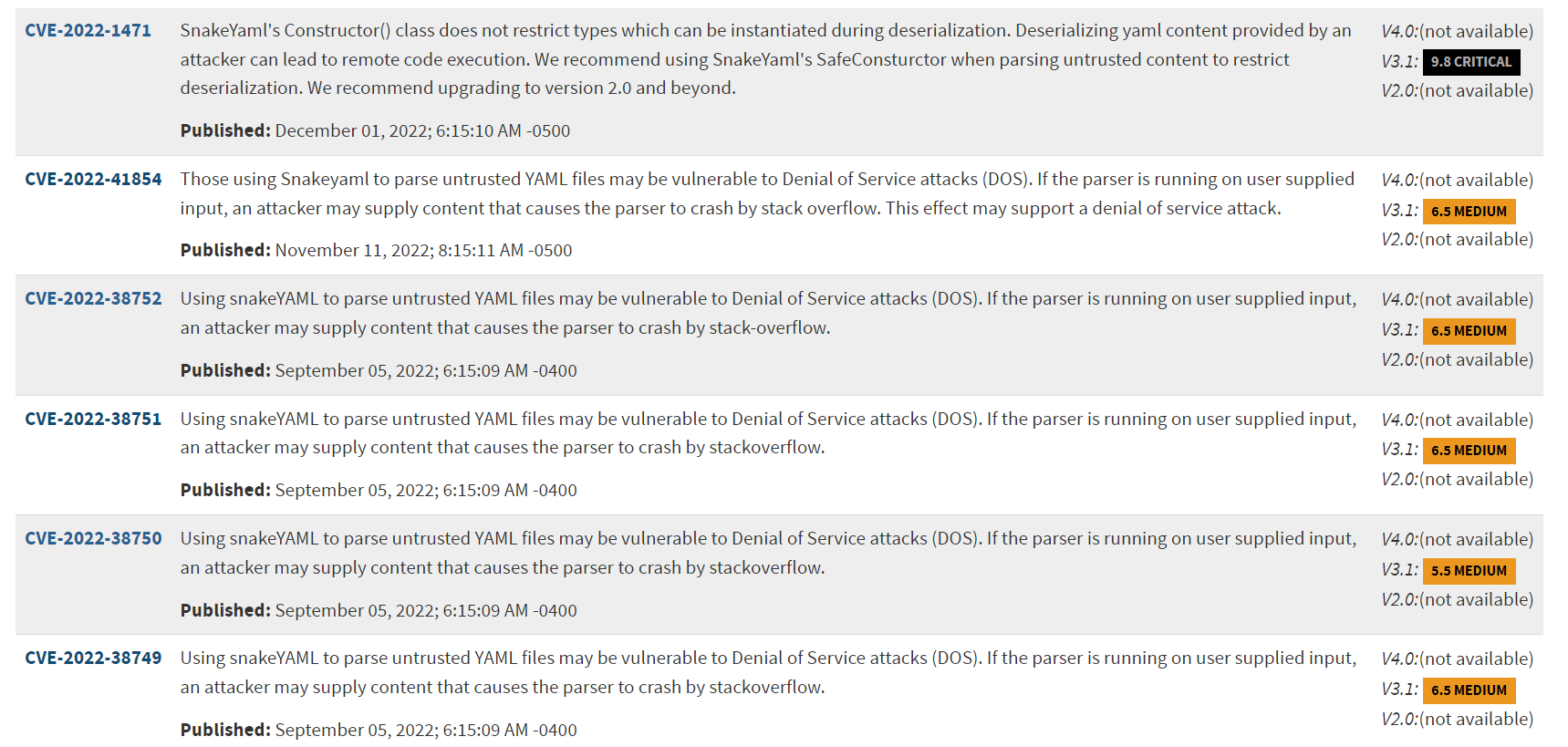
4. [log4j-api-2.12.1.jar](file:///C:\Users\16232\eclipse-workspace\rest-service\target\dependency-check-report.html#l10_a55e6d987f50a515c9260b0451b4fa217dc539cb)



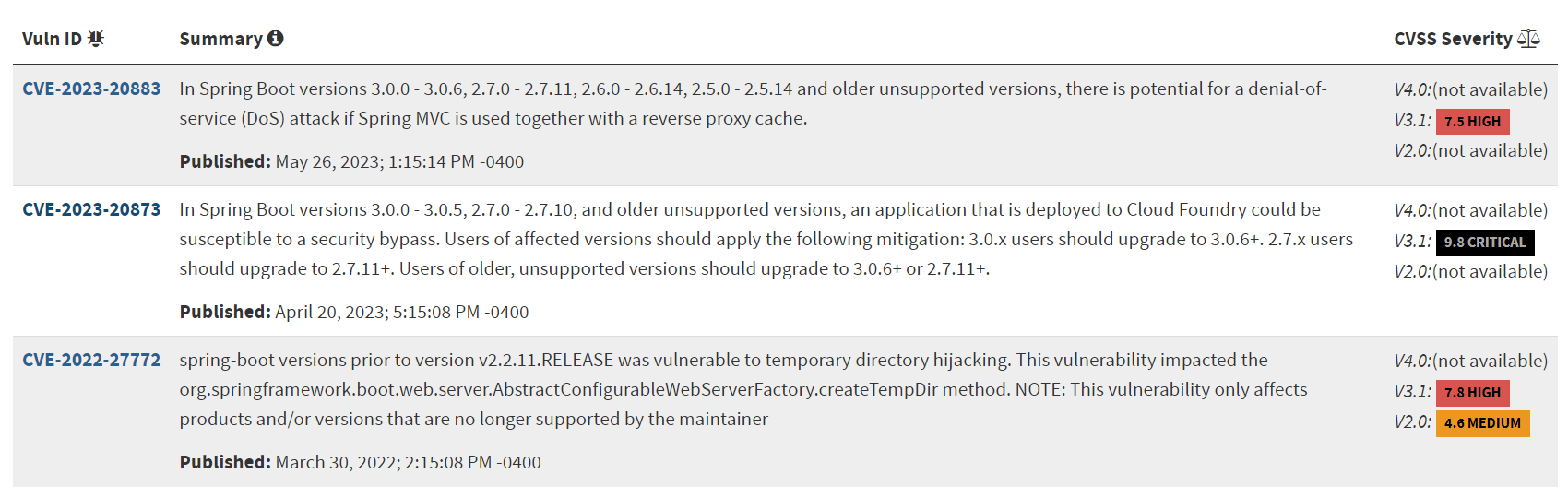
5. [logback-core-1.2.3.jar](file:///C:\Users\16232\eclipse-workspace\rest-service\target\dependency-check-report.html#l12_864344400c3d4d92dfeb0a305dc87d953677c03c)



6. [snakeyaml-1.25.jar](file:///C:\Users\16232\eclipse-workspace\rest-service\target\dependency-check-report.html#l14_8b6e01ef661d8378ae6dd7b511a7f2a33fae1421)



7. [spring-boot-2.2.4.RELEASE.jar](file:///C:\Users\16232\eclipse-workspace\rest-service\target\dependency-check-report.html#l15_225a4fd31156c254e3bb92adb42ee8c6de812714)



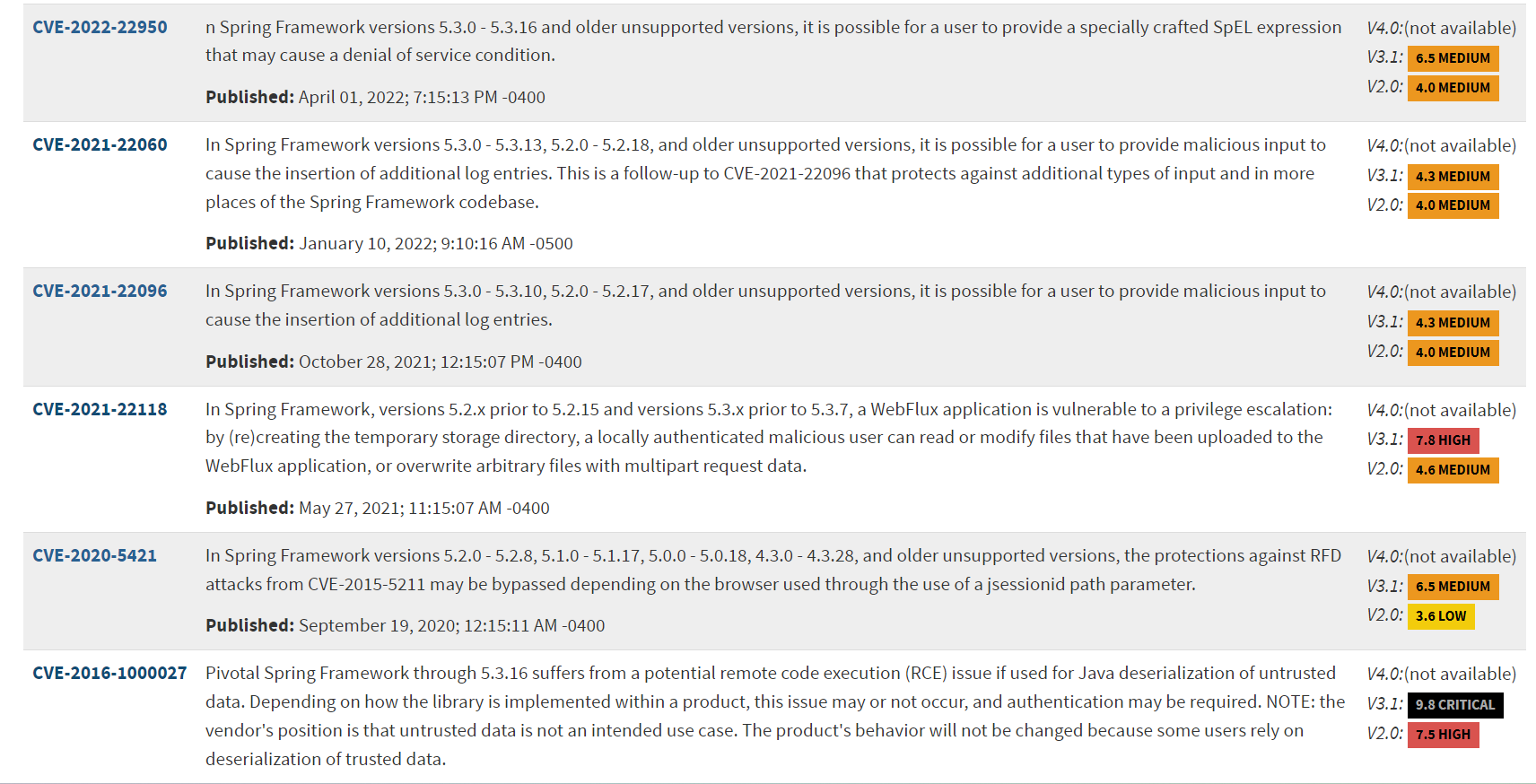
8. [spring-boot-starter-web-2.2.4.RELEASE.jar](file:///C:\Users\16232\eclipse-workspace\rest-service\target\dependency-check-report.html#l16_ec75d01d212b5229c16d872fb127744c0ed46ed8)

9. [spring-core-5.2.3.RELEASE.jar](file:///C:\Users\16232\eclipse-workspace\rest-service\target\dependency-check-report.html#l17_3734223040040e8c3fecd5faa3ae8a1ed6da146b)

10. [spring-web-5.2.3.RELEASE.jar](file:///C:\Users\16232\eclipse-workspace\rest-service\target\dependency-check-report.html#l18_dd386a02e40b915ab400a3bf9f586d2dc4c0852c)

11. spring-webmvc-5.2.3.RELEASE.jar

The springframework for 8-11 for dependency vulnerability had a few that could be potentially false positives due to there not being a vulnerability ID and description however all had roughly the same issues found in [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) which all have 12 vulnerabilities.



A screenshot of a computer

Description automatically generated

12. [tomcat-embed-core-9.0.30.jar](file:///C:\Users\16232\eclipse-workspace\rest-service\target\dependency-check-report.html#l20_ad32909314fe2ba02cec036434c0addd19bcc580)

13. [tomcat-embed-websocket-9.0.30.jar](file:///C:\Users\16232\eclipse-workspace\rest-service\target\dependency-check-report.html#l22_33157f6bc5bfd03380ebb5ac476db0600a04168d)

There is also a potential false positive with the second dependency ID show through apache tomcat but the first link had 20 issues that were reported and should be addressed. [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)

**5. Mitigation Plan**

After reviewing the code manually, I would suggest Artemis Financial focus on first implementing a validation for user input. I would also recommend having a system but in place where errors involving user input, whether accidental or malicious are handled accordingly to prevent authorized access to the system. Also, I would review the outputs given to the users to ensure that sensitive company information, such as locations, is not given out. Next, I would suggest changing the parameters are not labeled as personal information like business names. Information should be inputted into the system to protect client’s information by assigning each client an ID number and referring to them by the ID number throughout the application. All code errors which could potentially cause the application not to compile and run needs to be addressed before security testing.

The static testing revealed that there is software being used that is very outdated. I recommend looking through the spring framework, apache tomcat and bouncy castle to ensure all software is up to date and to then run another dependency check to verify other potential security hazards.