****

# 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** | **[Date]** | **[Your name]** |  |

## Client



## Instructions

Submit this completed vulnerability assessment report. Replace the bracketed text with the relevant information. In the report, identify your findings of security vulnerabilities and provide recommendations for the next steps to remedy the issues you have found.

* Respond to the five steps outlined below and include your findings.
* Respond using your own words. You may also choose to include images or supporting materials. If you include them, make certain to insert them in all the relevant locations in the document.
* Refer to the Project One Guidelines and Rubric for more detailed instructions about each section of the template.

## Developer

Jason Yaeger

## Interpreting Client Needs

1. What is the value of secure communications to the company?

The company Artemis Financial will be working with various customers that will need to interact with the web application in a secure manner. These interactions with the application could involve dealing with retirement funds, savings, investments, and insurance. All of these things are highly sensitive data that must be secure and protected.

1. Does the company make any international transactions?

Customers can be from around the world and international transactions are more than likely. There is not any specific mention of from where they could be, but it can be assumed that these international transactions are possible. Most financial institutions do this sort of thing.

1. Are there governmental restrictions about secure communications to consider?

## In the industry of finance there are many governmental restrictions that need to be considered. There are also various governing agencies to be concerned with as well. The Securities Exchange Commission will likely be involved with any of the investment, savings, and retirements funds side of things as they deal with the stock market and various investments. The U.S. Department of the Treasury is another one to consider. The Federal Reserve Board, Federal Deposit Insurance Corporation, Financial Industry Regulatory Authority, and any state bank or insurance regulators are also worth consideration. Finance is a heavily regulated industry. Regarding communications specifically, these agencies may have no specific restrictions or guidelines, but rather rules or recommendations to follow regarding how to handle customers' data and interact with markets.

1. What external threats might be present now and in the immediate future?

When dealing with people's personal information and especially money. There are always bad actors out there that will want access to the data. These constant external threats can be present now and in the immediate future.

1. What are the modernization requirements that you must consider? For example:
   1. The role of open-source libraries

Open-source libraries are very commonly used in software development. These pieces of code often contain vulnerabilities that are overlooked. Library revisioning can also cause old vulnerabilities to reappear. There are also attacks where a bad actor creates an open-source library that is actually malware or perhaps may be innocent at first, but they update it later to add malicious code.

* 1. Evolving web application technologies

There are more and more web applications being developed every day. The days of native Desktop applications are somewhat in the past. There are also new hardware and devices that web application developers must consider. Things like smart watches are more common and many applications have functionality on multiple devices like watches, phones, desktops, laptops, and even TVs. Using AI is another web application technology that has evolved and gained more traction, especially for things like chatbots and helping customers troubleshoot or navigate the application.

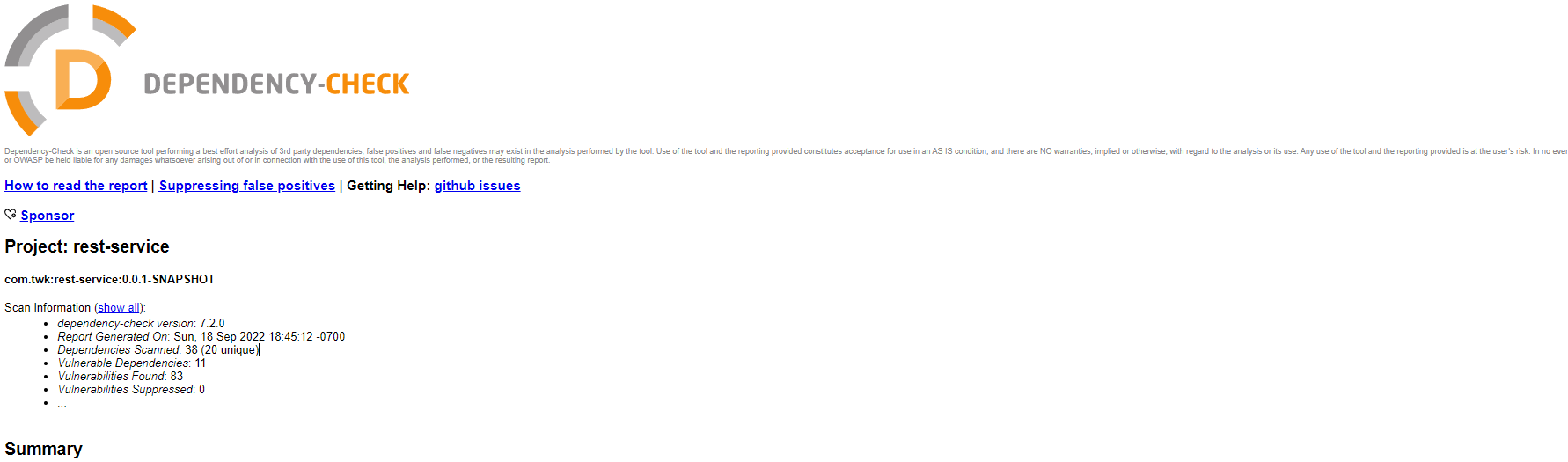
## Areas of Security

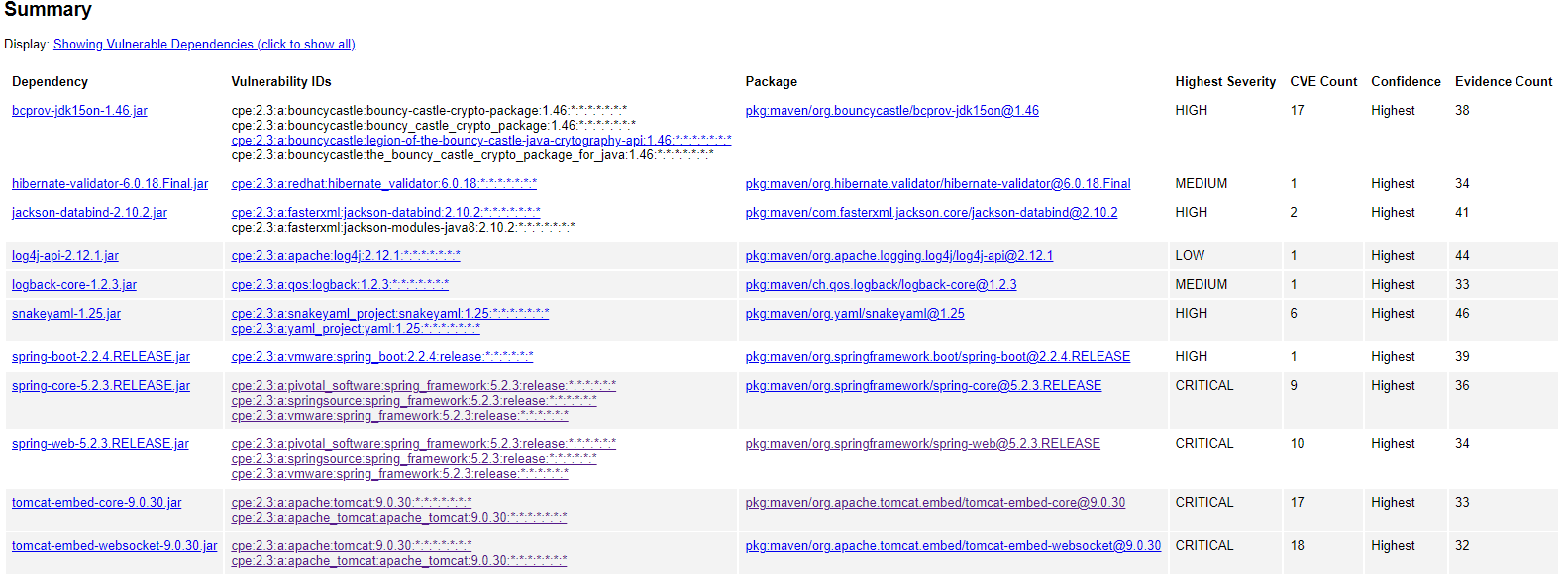
APIs, Cryptography, Code Error, Code Quality, Input Validation, and Client/Server all seemed relevant to the application based on the functionality of the software application. We are using restservice and springframework in the application so secure API interactions and client/server security are important. It seems like the user can deposit via the application so input validation matters here. There is also a try and catch code block used to read documents and do error handling, so code error secure handling is relevant. Lastly, I think code quality is a part of the vulnerability assessment process flow that is applicable to quite literally all applications as it embodies secure coding practices and patterns that should be considered at all times.

## Manual Review

When going over the code base one of the first things that stood out to me was in the customer.java file there is no error handling or input validation on the deposit amount int a. In that same file a user can easily access the account number, perhaps adding another layer of security such as asking for a pin for example would help secure important information in case the account is compromised. The APIs we use are also potentially outdated and this vulnerability will be better addressed after we do the static testing. I notice in the CRUDcontroller file we are using springframework which has had vulnerabilities in the previous module 2 assignment.

## Static Testing





The critical dependencies seen in the report are some of the same ones we saw in the module 2 assignment. These are the tomcat embed core, tomcat websocket, spring core, and spring web. I will repost my analysis from the previous assignment as it is relevant here as well, spring-web-5.2.3.RELEASE.jar, tomcat-embed-core-9.0.30.jar, tomcat-embed-websocket-9.0.30.jar, are the three critical dependencies that were found. The first critical issue can be most easily addressed by updated the spring framework to version 5.18x or greater. These newer versions are not vulnerable to the exploit. If for whatever reason updating is not possible, there are some other workarounds. One such potential solution is to disable binding to particular fields by setting disallowedFieldson WebDataBinder globally. Please see https://spring.io/blog/2022/03/31/spring-framework-rce-early-announcement#disallowed-fields for more information if needed.

The last two critical issues dealing with tomcat are described by the NVD as follows “When using the Apache JServ Protocol (AJP), care must be taken when trusting incoming connections to Apache Tomcat. Tomcat treats AJP connections as having higher trust than, for example, a similar HTTP connection. If such connections are available to an attacker, they can be exploited in ways that may be surprising. In Apache Tomcat 9.0.0.M1 to 9.0.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99, Tomcat shipped with an AJP Connector enabled by default that listened on all configured IP addresses. It was expected (and recommended in the security guide) that this Connector would be disabled if not required. This vulnerability report identified a mechanism that allowed: - returning arbitrary files from anywhere in the web application - processing any file in the web application as a JSP Further, if the web application allowed file upload and stored those files within the web application (or the attacker was able to control the content of the web application by some other means) then this, along with the ability to process a file as a JSP, made remote code execution possible. It is important to note that mitigation is only required if an AJP port is accessible to untrusted users. Users wishing to take a defence-in-depth approach and block the vector that permits returning arbitrary files and execution as JSP may upgrade to Apache Tomcat 9.0.31, 8.5.51 or 7.0.100 or later”

As for the other dependencies that need addressing. One of them pertains to the java cryptography extension bouncy castle. The application is using an outdated version of bouncy castle APIs and needs to be updated to be secure. Another issue is the spring boot release version that allowed attackers to temporarily hijack directories. This version is no longer supported or maintained by the provider and will need to be updated. Another issue that was found involves snakeyYAML which is used to read and write YAML files in java and/or parser and emitter. This vulnerability exposes the application to denial-of-service attacks, and it is very important to fix. This can be done by restricting the nested depth for collections, see link here <https://bitbucket.org/snakeyaml/snakeyaml/commits/fc300780da21f4bb92c148bc90257201220cf174> and <https://github.com/snakeyaml/snakeyaml/commit/fc300780da21f4bb92c148bc90257201220cf174>

Another issue found was “A flaw was found in Hibernate Validator version 6.1.2.Final. A bug in the message interpolation processor enables invalid EL expressions to be evaluated as if they were valid. This flaw allows attackers to bypass input sanitation (escaping, stripping) controls that developers may have put in place when handling user-controlled data in error messages.” There was a manual way to fix at first by “pass user input as an expression variable by unwrapping the context to HibernateConstraintValidatorContext”. But now we can simply update the version and the vulnerability has been addressed by the maintainer/Hibernate team.

## Mitigation Plan

We must take several steps to address the identified vulnerabilities. First and foremost, we should update and be on the most recent secure version of all the packages and software we are using. This will mitigate many of the found vulnerabilities such as the critical spring core, spring web, and tomcat vulnerabilities. Next, we will update the hibernate version so attackers cannot bypass any input sanitations controls that are in place and other handling of user-controlled data in error messages. Then some of the manual work that needs to be done is addressing the snakeyYAML issues regarding the potential denial of service vulnerability we are exposed to. We must restrict the nested depth for collections which there is GitHub and bitbucket code to help us with that can be referenced above.

**Citations**

Smet, G. (n.d.). *Hibernate Validator 6.1.5.final and 6.0.20.final released*. In Relation To. Retrieved September 18, 2022, from <https://in.relation.to/2020/05/07/hibernate-validator-615-6020-released/>

*CVE-2020-10693 Detail*. NVD. (n.d.). Retrieved September 18, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2020-10693>

Stoyanchev, R. (2022, March 31). *Spring Framework RCE, early announcement*. Spring. Retrieved September 11, 2022, from <https://spring.io/blog/2022/03/31/spring-framework-rce-early-announcement#disallowed-fields>

NVD. (n.d.). Retrieved September 11, 2022, from

https://nvd.nist.gov/vuln/detail/CVE-2020-1938