### Southern New Hampshire University

Project One: Vulnerability Assessment Report

Monticia Dunn

Dr. Vivian Lyon

CS 305 – Software Security

9/17/22

****

# Artemis Financial Vulnerability Assessment Report

Table of Contents

[Document Revision History 4](#_Toc114392648)

[Client 4](#_Toc114392649)

[Developer 5](#_Toc114392650)

[1. Interpreting Client Needs 5](#_Toc114392651)

[2. Areas of Security 5](#_Toc114392652)

[3. Manual Review 5](#_Toc114392653)

[4. Static Testing 6](#_Toc114392654)

[5. Mitigation Plan 8](#_Toc114392655)

[References 9](#_Toc114392656)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **9/15/2022** | **Monticia Dunn** |  |

## Client



## Developer

Monticia Dunn

## Interpreting Client Needs

The client is a financial company and financial companies have a lot of sensitive data about their customers, this could make them a target for attackers to try to obtain that information. Though it is not explicitly stated in the scenario, the company could potentially handle international transactions since it is web-based software, which would make the tighter security even more important. There are typically government policies when handling financial records for example, banks are now required to inform the federal regulator about any incidents that have occurred that can affect the viability of their operations or their ability to deliver services and products (Duncan, 2022). So far, I think the biggest external threats are hackers trying to steal data, as well as viruses that could accidentally get through from phishing/whaling attempts. Some modern requirements to help counter this are things like MFA (Multi-Factor Authentication), HTTPS (Hypertext Transfer Protocol Secure) and TSL (Transport Layer Security).

## Areas of Security

The biggest areas of security to watch for in this application are APIs, Client/Server, Code Quality and Cryptography.

APIs is probably the most important area since the client Artemis has a RESTful web application programming interface (API). With the RESTful architecture there is a standard way for programs to interact with the service making it easier to apply OAuth (Open Authorization) and securely enable users who want to access the API without having to share their passwords, so that’s less sensitive information being passed around (Scott & Neray, 2021).

Cryptography will be used to access and transfer information to and from the API, we must secure the data that is being transmitted by following a process known as encryption so the data cannot be easily compromised by external threats. One of the ways we could do this is through TSL, which will protect the information the API sends by encrypting them while they’re in transit (Scott & Neray, 2021).

Client/Server architecture typically involves breaking down tasks between the clients and servers that reside on the same system or are linked by a network (Terra, 2022). The users will be interacting on the client-side to either send or retrieve information from the API on the server-side, this architecture lends itself to security since there are more layers between the request and the database. Most of the security will need to be on the server-side to ensure the appropriate access depending on the request. This goes together with Cryptography as we’ll want to secure information as it travels from the client-side to the server-side.

Code Quality is applied to the entire system as a whole since regardless because we still want to utilize good secure coding practices as that will lead to less bugs and vulnerabilities. Some of these secure coding practices include input validation, default deny, and watching compiler warnings (Seacord, 2018).

## Manual Review

In the *DocData.java* file, on lines 25-27, it appears that we’re trying to establish a connection with the database but the way the username and password is being sent is not very secure. First, the username and password are just “root” which is an easily discernable password for attackers to guess so I would recommend setting the username and password to something harder to guess. Also, the username and password are hardcoded in the file which will make it easier for an attacker to access the database if the application is compromised since they won’t have to figure out the credentials, this vulnerability falls under the Cryptography area of security.



Also there doesn’t seem to be a lot of validation for inputs, for instance in the GreetingController.java file, the Greeting function takes in a parameter but doesn’t validate the input it was given, and this is common with several functions in the application. This vulnerability falls under the Code Quality area of security.

## Static Testing

Of the 38 dependencies scanned, 11 vulnerable dependencies were found with a total of 83 vulnerabilities. These include:

* bcprov-jdk15on-1.46.jar
  + [CVE-2016-1000352](https://nvd.nist.gov/vuln/detail/CVE-2016-1000352) – a bug that allows the use of a deprecated mode leaving the application open to various cryptographic issues
  + [CVE-2016-1000346](https://nvd.nist.gov/vuln/detail/CVE-2016-1000346)
  + [CVE-2016-1000345](https://nvd.nist.gov/vuln/detail/CVE-2016-1000345)
  + [CVE-2016-1000344](https://nvd.nist.gov/vuln/detail/CVE-2016-1000344)
  + [CVE-2016-1000343](https://nvd.nist.gov/vuln/detail/CVE-2016-1000343)
  + [CVE-2016-1000342](https://nvd.nist.gov/vuln/detail/CVE-2016-1000342)
  + [CVE-2016-1000341](https://nvd.nist.gov/vuln/detail/CVE-2016-1000341)
  + [CVE-2016-1000339](https://nvd.nist.gov/vuln/detail/CVE-2016-1000339)
  + [CVE-2016-1000338](https://nvd.nist.gov/vuln/detail/CVE-2016-1000338)
  + [CVE-2018-5382](https://nvd.nist.gov/vuln/detail/CVE-2018-5382)
  + [CVE-2017-13098](https://nvd.nist.gov/vuln/detail/CVE-2017-13098)
  + [CVE-2013-1624](https://nvd.nist.gov/vuln/detail/CVE-2013-1624)
* hibernate-validator-6.0.18.Final.jar
  + [CVE-2020-10693](https://nvd.nist.gov/vuln/detail/CVE-2020-10693) – a bug that potentially allows attackers to bypass input sanitation controls
* jackson-databind-2.10.2.jar
  + [CVE-2020-36518](https://nvd.nist.gov/vuln/detail/CVE-2020-36518) – a bug that potentially allows a Java StackOverflow exception and denial of service via a large depth of nested objects.
  + [CVE-2020-25649](https://nvd.nist.gov/vuln/detail/CVE-2020-25649) – a bug that potentially allows vulnerability to XML external entity (XXE) attacks
* log4j-api-2.12.1.jar
  + [CVE-2021-44832](https://nvd.nist.gov/vuln/detail/CVE-2021-44832) – a bug that potentially allows a remote code execution (RCE) attack when a configuration uses a JDBC Appender with a JNDI LDAP data source URI when an attacker has control of the target LDAP server.
  + [CVE-2021-45105](https://nvd.nist.gov/vuln/detail/CVE-2021-45105) – a bug that potentially allows for uncontrolled recursion from self-referential lookups
  + [CVE-2021-45046](https://nvd.nist.gov/vuln/detail/CVE-2021-45046) – a bug that potentially allows attackers to craft malicious input data using a JNDI Lookup pattern resulting in an information leak and remote code execution in some environments and local code execution in all environments.
  + [CVE-2021-44228](https://nvd.nist.gov/vuln/detail/CVE-2021-44228) – a bug that potentially allows an attacker who can control log messages or log message parameters to execute arbitrary code loaded from LDAP servers when message lookup substitution is enabled.
  + [CVE-2020-9488](https://nvd.nist.gov/vuln/detail/CVE-2020-9488) – a bug that potentially allows an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender.
* logback-core-1.2.3.jar
  + [CVE-2021-42550](https://nvd.nist.gov/vuln/detail/CVE-2021-42550) – a bug that potentially allows an attacker with the required privileges to edit configurations files to craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers.
* snakeyaml-1.25.jar
  + [CVE-2022-38751](https://nvd.nist.gov/vuln/detail/CVE-2022-38751) – a bug that potentially allows for Denial-of-Service attacks (DOS) and an attacker may supply content that causes the parser to crash by stackoverflow
  + [CVE-2022-38750](https://nvd.nist.gov/vuln/detail/CVE-2022-38750) – Same as above
  + [CVE-2022-38749](https://nvd.nist.gov/vuln/detail/CVE-2022-38749) – Same as above
  + [CVE-2022-25857](https://nvd.nist.gov/vuln/detail/CVE-2022-25857) – a bug that potentially allows for Denial of Service (DoS) due missing to nested depth limitation for collections.
  + [CVE-2017-18640](https://nvd.nist.gov/vuln/detail/CVE-2017-18640) – a bug that potentially allows for entity expansion during a load operation
* spring-boot-2.2.4.RELEASE.jar
  + [CVE-2022-27772](https://nvd.nist.gov/vuln/detail/CVE-2022-27772) – spring-boot versions prior to version v2.2.11.RELEASE was vulnerable to temporary directory hijacking. This vulnerability only affects products and/or versions that are no longer supported by the maintainer.
* spring-core-5.2.3.RELEASE.jar
  + [CVE-2022-22971](https://nvd.nist.gov/vuln/detail/CVE-2022-22971) – a bug that potentially allows a denial-of-service attack by an authenticated user.
  + [CVE-2022-22970](https://nvd.nist.gov/vuln/detail/CVE-2022-22970)
  + [CVE-2022-22968](https://nvd.nist.gov/vuln/detail/CVE-2022-22968)
  + [CVE-2022-22965](https://nvd.nist.gov/vuln/detail/CVE-2022-22965)
  + [CVE-2022-22950](https://nvd.nist.gov/vuln/detail/CVE-2022-22950)
  + [CVE-2021-22060](https://nvd.nist.gov/vuln/detail/CVE-2021-22060)
  + [CVE-2021-22096](https://nvd.nist.gov/vuln/detail/CVE-2021-22096)
  + [CVE-2021-22118](https://nvd.nist.gov/vuln/detail/CVE-2021-22118)
  + [CVE-2020-5421](https://nvd.nist.gov/vuln/detail/CVE-2020-5421)
  + [CVE-2016-1000027](https://nvd.nist.gov/vuln/detail/CVE-2016-1000027)
* spring-web-5.2.3.RELEASE.jar
  + Same as *spring-core-5.2.3.RELEASE.jar*
* tomcat-embed-core-9.0.30.jar
  + [CVE-2022-34305](https://nvd.nist.gov/vuln/detail/CVE-2022-34305) – a bug that potentially exposes a XSS vulnerability
  + [CVE-2022-29885](https://nvd.nist.gov/vuln/detail/CVE-2022-29885)
  + [CVE-2021-41079](https://nvd.nist.gov/vuln/detail/CVE-2021-41079)
  + [CVE-2021-33037](https://nvd.nist.gov/vuln/detail/CVE-2021-33037)
  + [CVE-2021-30640](https://nvd.nist.gov/vuln/detail/CVE-2021-30640)
  + [CVE-2021-25329](https://nvd.nist.gov/vuln/detail/CVE-2021-25329)
  + [CVE-2021-25122](https://nvd.nist.gov/vuln/detail/CVE-2021-25122)
  + [CVE-2021-24122](https://nvd.nist.gov/vuln/detail/CVE-2021-24122)
  + [CVE-2020-17527](https://nvd.nist.gov/vuln/detail/CVE-2020-17527)
  + [CVE-2020-13943](https://nvd.nist.gov/vuln/detail/CVE-2020-13943)
  + [CVE-2020-13935](https://nvd.nist.gov/vuln/detail/CVE-2020-13935)
  + [CVE-2020-13934](https://nvd.nist.gov/vuln/detail/CVE-2020-13934)
  + [CVE-2020-8022](https://nvd.nist.gov/vuln/detail/CVE-2020-8022)
  + [CVE-2020-11996](https://nvd.nist.gov/vuln/detail/CVE-2020-11996)
  + [CVE-2020-9484](https://nvd.nist.gov/vuln/detail/CVE-2020-9484)
  + [CVE-2020-1938](https://nvd.nist.gov/vuln/detail/CVE-2020-1938)
  + [CVE-2020-1935](https://nvd.nist.gov/vuln/detail/CVE-2020-1935)
  + [CVE-2019-17569](https://nvd.nist.gov/vuln/detail/CVE-2019-17569)
* tomcat-embed-websocket-9.0.30.jar
  + Same as *tomcat-embed-core-9.0.30.jar*

## Mitigation Plan

The first step to remedy the issues from the dependency check is to update the vulnerable dependencies that were detected to the most recent version. These version updates typically include patches for security flaws that we don’t need to mitigate manually. After performing this update, a code review will need to be performed to ensure that the updates didn’t break anything else in the process. We will also need to move the username and password to an environment variable or somewhere else more secure than hardcoding it directly into the Controller file and of course, change the username and password to something less easily guessable. I also can’t tell from the files but if it’s not already there then we’ll need to implement HTTPS to help keep requests from getting compromised.

# References

Duncan, C. (2022, September 8). *5 Biggest Threats To Cyber Security In The Banking Industry In 2022*. Retrieved from DeskAlerts: https://www.alert-software.com/blog/cybersecurity-in-banking

Scott, S., & Neray, G. (2021, October 6). *Best practices for REST API security: Authentication and authorization*. Retrieved from Stack Overflow blog: https://stackoverflow.blog/2021/10/06/best-practices-for-authentication-and-authorization-for-rest-apis/

Seacord, R. (2018, May 2). *Top 10 Secure Coding Practices*. Retrieved from Carnegie Mellon University Software Engineering Institute: https://wiki.sei.cmu.edu/confluence/display/seccode/Top+10+Secure+Coding+Practices

Terra, J. (2022, August 25). *What is Client-Server Architecture? Everything You Should Know*. Retrieved from Simplilearn: https://www.simplilearn.com/what-is-client-server-architecture-article