****

# Artemis Financial Vulnerability Assessment Report

Table of Contents

[Artemis Financial Vulnerability Assessment Report 1](#_Toc125222313)

[Document Revision History 3](#_Toc125222314)

[Client 3](#_Toc125222315)

[Instructions 3](#_Toc125222316)

[Developer 4](#_Toc125222317)

[1. Interpreting Client Needs 4](#_Toc125222318)

[2. Areas of Security 5](#_Toc125222319)

[3. Manual Review 6](#_Toc125222320)

[4. Static Testing 7](#_Toc125222321)

[5. Mitigation Plan 7](#_Toc125222322)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **01/20/2023** | **Sarah Schmidt** | **Vulnerability assessment report to identify any security vulnerabilities.** |

## 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

Sarah Schmidt

## Interpreting Client Needs

1. **What is the value of secure communications to the company?**
   1. Artemis Financial has a high value of secure communications in the company. Since their companies’ mission statement is “Security is everyone’s responsibility”. Artemis Financial is responsible for their clients’ personal information, if there were to be a security breach, they would lose the trust of their clients, ruining their reputation.
2. **Does the company make any international transactions?**
   1. Since this is a financial company, it is safe to assume that this company is dealing with international transactions. They work with businesses, entrepreneurs, and the government.
3. **Are there governmental restrictions about secure communications to consider?**
   1. There are plenty of governmental restrictions Artemis Financial will need to consider with secure communications. First off, they work with the government. Any vulnerabilities this website may have could be detrimental to their government clients. There are certain Cybersecurity laws that financial agencies must follow.
4. **What external threats might be present now and in the immediate future?**
   1. External threats would be those who would be seeking personal information from the employees or looking into the API of the company. Some individuals may be looking for technological flaws in the web application that could lead to seeing information stored within the computer. Other issues to bring up would be those inside the company who may be blackmailed or bribed to give information, and certain terrorist groups trying to get information on the government.
5. **What are the modernization requirements that you must consider? For example:**
   1. **The role of open-source libraries**
   2. **Evolving web application technologies**
      1. Some modernized requirements like open-source libraries can create an issue because the source code is made available for anyone to use or modify as that developer sees fit. This could become an issue because if a company like Artemis financial shares its source code, it could open a wide variety of dependencies for others to understand how their code works.
      2. Other items to consider are that the web application will want to be updated regularly. If it’s not someone may be able to get into the source code creating a vulnerability. For Artemis Financial, cross-site Scripting (XSS) is something to keep in mind. Attackers can insert code onto a webpage now that will redirect users from the real site to a malicious one. If the user enters their username and password, the attacker now has the credentials to get into their personal information. If this user has a similar password for all their accounts, this could affect the 2-step identification process as well.

## Areas of Security

1. **Use what you’ve learned in step 1 and refer to the Vulnerability Assessment Process Flow Diagram provided. Think about the functionality of the software application to identify which areas of security apply to Artemis Financials’ web application. Document your findings in your vulnerability assessment report and justify why each area is relevant to the software application.**
   1. APIs: Application Program Interface, APIs help prevent malicious attacks on web applications. Companies are ever-changing, and with this, we will need to have developers working on updating the security of web applications to continue to achieve an up-to-date secure web application for Artemis Financials.
   2. Cryptography: We need to make sure that the Artemis Financials cryptography that is implemented isn’t going to be compromised. If there is a cryptography failure on their web application this can lead to a vulnerability that can give out sensitive information such as passwords, account information, credit card information, and so forth.
   3. Input Validation: Any input that is given to the user or application should be tested. Input validation will help prevent incorrect data from entering the system. For instance, if someone needs to enter their account number in the Artemis Financials web application, the input validation will make sure that number doesn’t contain anything it’s supposed to like letters. This helps with making sure that each item entered is correct.
   4. Code Quality: Code quality and secure coding practices are essential for a web application like Artemis Financials. You can have certain access of control for each user, this helps ensure that the code and information from each account cannot be accessed by just anyone.

## Manual Review

1. **Refer to the seven security areas outlined in the Vulnerability Assessment Process Flow Diagram. Use what you’ve learned in steps 1 and 2 to guide your manual review. Identify all vulnerabilities in the Project One Code Base, linked in Supporting Materials, by manually inspecting the code. Document your findings in your vulnerability assessment report. Be sure to include a description that identifies where the vulnerabilities are found (specific class file, if applicable).**
   1. **Customer.java**
      1. Public void deposit (int a) {

account\_balance = account\_balance + a;

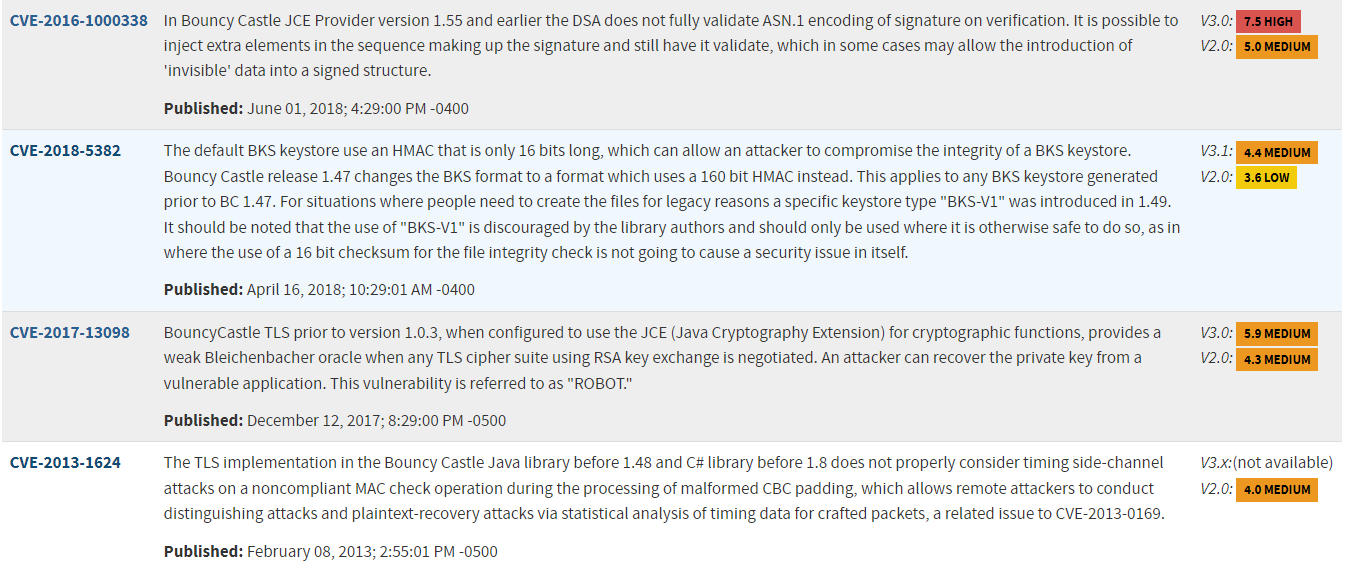
}

1. This code reflects depositing the amount into an account. What I see should be done is some sort of validation of how much is a? what if someone is trying to deposit a negative amount? Is there an amount that wouldn’t be acceptable?
   1. Greeting.java
      1. Requests are not being validated by logging in, there is no authentication system in even if there was a verification process.
   2. CRUDController
      1. The business names are being passed as request parameters, although this is something that’s done within an API, this is usually a response when you want to know everything about the entity being studied. We should be doing a login identifying each client base within that business. Not everyone in business needs access to everything. This can also lead to vulnerability by again no validation of the business and can lead to an attack very quickly.

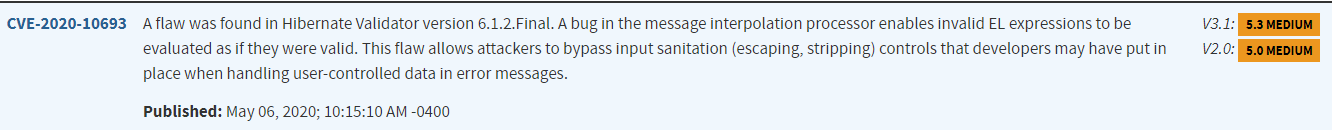
## Static Testing

1. **Static Testing: Integrate the dependency-check plug-in into Maven by following the instructions outlined in the Integrating the Maven Dependency-Check Plug-in tutorial provided in Supporting Materials. Run a dependency check on Artemis Financials’ software application to identify all security vulnerabilities in the code. Specifically, identify all vulnerabilities in the code base by analyzing results from running the code through a static test. Include these items from the dependency-check report in your vulnerability assessment report:**
   1. **The names or vulnerability codes of the known vulnerabilities**
   2. **A brief description and recommended solutions that are found in the dependency-check report.**
   3. **Attribution (if any) that documents how this vulnerability has been identified or how it was documented in the past.**
2. [**Dependency Check Report Link**](file:///C:\Users\sarah\Downloads\CS%20305%20Project%20One%20Code%20Base%20(1)\rest-service\target\dependency-check-report.html)
   1. **Scan Information** 
      1. **Dependencies scanned: 28 (22 unique)**
      2. **Vulnerable Dependencies: 13**
      3. **Vulnerabilities Found: 103**
   2. **Bcprov-jdk15on-1.46.jar**
      1. **The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms. This jar contains JCE provider and lightweight API for the Bouncy Castle Cryptography APIs for JDK 1.5 to JDK 1.7.**
      2. [**National Vulnerability Database**](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)

**Graphical user interface

Description automatically generated with medium confidence**

* 1. **Hibernate-validator-6.0.18.Final.jar**
     1. **Hibernate's Bean Validation (JSR-380) reference implementation.**
     2. [**National Vulnerability Database**](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)

****

* 1. **Jackson-databind-2.10.2.jar**
     1. **General data-binding functionality for Jackson: works on core streaming API**
     2. [**National Vulnerability Database**](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)

**Text, application

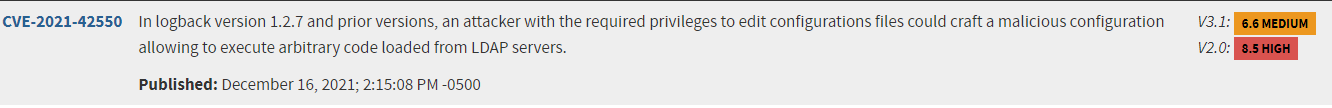
Description automatically generated**

* 1. **Log4j-api-2.12.1.jar**
     1. **The Apache Log4jAPI**
     2. [**National Vulnerability Database**](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)

**Graphical user interface, text, application, Word

Description automatically generated**

* 1. **Logback-core-1.2.3.jar**
     1. **Logback-core module**
     2. [**National Vulnerability Database**](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aqos&cpe_product=cpe%3A%2F%3Aqos%3Alogback&cpe_version=cpe%3A%2F%3Aqos%3Alogback%3A1.2.3)

****

* 1. **Snakeyaml-1.25.jar**
     1. **YAML 1.1 parser and emitter for Java**
     2. [**National Vulnerability Database**](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)

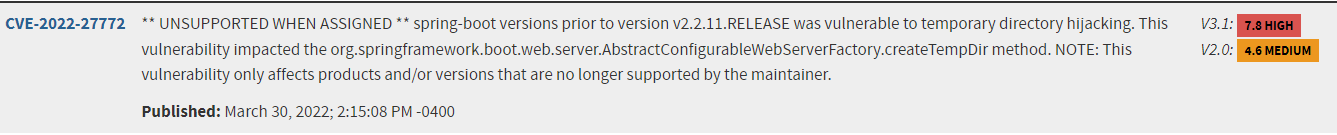
**A picture containing table

Description automatically generated**

**Text, application

Description automatically generated**

* 1. **Spring-boot-2.2.4.RELEASE.jar**
     1. **Spring Boot**
     2. [**National Vulnerability Database**](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_boot&cpe_version=cpe%3A%2F%3Avmware%3Aspring_boot%3A2.2.4)

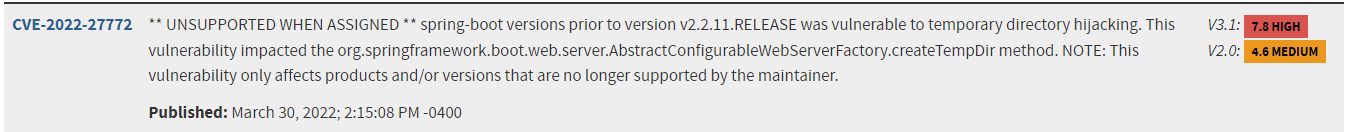
****

## 

* 1. Spring-boot-starter-web-2.2.4.RELEASE.jar
     1. Starter for building web, including RESTful, applications using Spring.

MVC. Uses Tomcat as the default embedded container.

* + 1. [National Vulnerability Database](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_boot&cpe_version=cpe%3A%2F%3Avmware%3Aspring_boot%3A2.2.4)



* 1. Spring-core-5.2.3.RELEASE.jar
     1. Spring Core
     2. [National Vulnerability Database](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)

A picture containing graphical user interface

Description automatically generated

* 1. Spring-web-5.2.3.RELEASE.jar
     1. Spring Web
     2. Vulnerability National Database

Graphical user interface

Description automatically generated with medium confidence

* 1. Spring-webmvc-5.2.3.RELEASE.jar
     1. Spring Web MVC
     2. [National Vulnerability Database](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)

A picture containing graphical user interface

Description automatically generated

* 1. Tomcat-embed-core-9.0.30.jar
     1. Core Tomcat Implementation

* + 1. [National Vulnerability Database](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)

A picture containing graphical user interface

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

* 1. Tomcat-embed-websocket-9.0.30.jar
     1. Core Tomcat implementation
     2. [National Vulnerability Database](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)

## Mitigation Plan

1. **Mitigation Plan: Interpret the results from the manual review and static testing report. Identify steps to mitigate the identified security vulnerabilities by creating an action list that documents how to fix each vulnerability in your vulnerability assessment report.**
   1. **There’s plenty listed in this document to support the action list within the mitigation plan. I believe that the first thing that we need to do is have current versions of all API’s running in Artemis Financials web-application. The next thing that we need to do is create a two-part authentication process for users to log in with. We then need to fix some of the code that doesn’t have validation input within the system. Like depositing or withdrawing from the accounts. I would then like to create a plan with the Scrum Team to figure out what is the highest priority tasks for the vulnerabilities that we found in our dependency checks. We need to figure out how high and critical CVE’s that are listed because we know that this could potentially cause an attack within their web application. While doing this, I would also want to set up future meetings with Artemis Financial to go over what items they may want some cyber-education on. If we can work together on this there may be less vulnerabilities in the future.**

Sources

Engineering, E. (2023, January 13). *Cybersecurity laws and regulations in US 2022*. EES Corporation. Retrieved January 22, 2023, from <https://www.eescorporation.com/cybersecurity-laws-and-regulations-in-us/>

Scorecard, S. (2021, March 25). *41 common web application vulnerabilities explained*. 41 Common Web Application Vulnerabilities Explained. Retrieved January 22, 2023, from <https://securityscorecard.com/blog/common-web-application-vulnerabilities-explained>