****

# 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** | **3/20/2022** | **Shawn Neal** |  |

## Client



## Developer

Shawn Neal

## 1. Interpreting Client Needs

As a financial firm, secure communications should be a top priority of Artemis Financial. With clients trusting the firm to manage their savings, retirement, investments and insurance, preserving the security of all data and communications is a priority both in the interest of their customers and in the terms of federal and international regulations.

I would assume that international markets would be part of any financial firm that manages investments and insurance, though I could also see a company that focuses on domestic investments only.

The only government restrictions might be in the form of the Fair Credit Reporting Act and the Gramm-Leach-Biley Act. The FCRA manages the accuracy, fairness and privacy of information that consumer credit bureaus file, whereas the Gramm-Leach-Biley Act.

Financial Institutions are amongst the top targets for cyber-attacks. Not only are the actual financial transactions a target, but the firm also stores a vast array of sensitive confidential information, from personal information such as names, birthdates, social security numbers, financial histories, addresses, and tax information.

Maintaining an up-dated system, including open-source libraries and frameworks can be a critical part of maintaining a secure and protected application. Regular maintenance can ensure that identified vulnerabilities can be monitored, addressed and fixed. New options can be considered and applied if needed. And system lifespan can be evaluated as well.

## 2. Areas of Security

* Input validation – The application must ensure that any access to the system is only done by authorized users and only in means that authorized users intend to.
* API’s - The use of third-party API’s means that specific steps must be taken to ensure proper and secure interactions, and regular maintenance must be applied to ensure that the most stable and secure version is being used.
* Cryptography The sensitive nature of both the transactions and customer information involved means that data encryptions must be applied, and any potential vulnerabilities be addressed as soon as possible.
* Client/Server - Given the online nature of the application, the application must be able to speak both to the client-side interface (meaning an app or website) and the firm's servers in a secure manner from anywhere the client has internet access.
* Code Error – The application needs to have the proper error handling in place, with options for the client to correct or bypass error, or contact the firm for additional support, and the appropriate notification sent to the application administrator.
* Code Quality – High quality code that follows secure coding standard will ensure vulnerabilities are minimized and unnecessary errors are avoided.
* Encapsulation – Given that data security is a top priority, information should be protected and only accessible in an authorized manner by authorized users.

## 3. Manual Review

By reviewing the code, I found multiple vulnerabilities in place when looking for signs of the seven vulnerabilities categories. First an older version of spring is being used. The application uses no input validation, and the API is not implemented yet. The Code is of good quality, but the code error handling is lacking, though there is one partially implemented instance of this in the DocData file. There is no attempt at cryptography, or encapsulation. The application accesses data via url, which leaves leaks into the browser history, rather than being implemented via the POST method.

## 4. Static Testing

|  |  |  |  |
| --- | --- | --- | --- |
| Dependency | Vulnerability | Description | Solution |
| hibernate-validator-6.0.18.Final.jar | cpe:2.3:a:redhat:hibernate\_validator:6.0.18:\*:\*:\*:\*:\*:\*:\* | 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. | Update RedHat OpenTack platform to at least 7.3 |
| jackson-databind-2.10.2.jar | cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\* | jackson-databind before 2.13.0 allows a Java StackOverflow exception and denial of service via a large depth of nested objects. | Update jackson package to 2.13.2 |
| og4j-api-2.12.1.jar | cpe:2.3:a:apache:log4j:2.12.1:\*:\*:\*:\*:\*:\*:\* | Apache Log4j2 versions 2.0-beta7 through 2.17.0 (excluding security fix releases 2.3.2 and 2.12.4) are vulnerable to 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. This issue is fixed by limiting JNDI data source names to the java protocol in Log4j2 versions 2.17.1, 2.12.4, and 2.3.2. | Update Apache log to 2.17.1 |
| logback-core-1.2.3.jar | cpe:2.3:a:qos:logback:1.2.3:\*:\*:\*:\*:\*:\*:\* | In logback version 1.2.7 and prior versions, an attacker with the required privileges to edit configurations files could craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers. | Update logback to 1.2.7 |
| snakeyaml-1.25.jar | cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\* | The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564. | Update snakeyaml to version 1.26 |
| spring-aop-5.2.3.RELEASE.jar | cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* | In Spring Framework versions 5.2.0 - 5.2.8, 5.1.0 - 5.1.17, 5.0.0 - 5.0.18, 4.3.0 - 4.3.28, and older unsupported versions, the protections against RFD attacks from CVE-2015-5211 may be bypassed depending on the browser used through the use of a jsessionid path parameter. | Update Spring framework to 5.2.9 |
| tomcat-embed-core-9.0.30.jar | cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* | The fix for bug 63362 present in Apache Tomcat 10.1.0-M1 to 10.1.0-M5, 10.0.0-M1 to 10.0.11, 9.0.40 to 9.0.53 and 8.5.60 to 8.5.71 introduced a memory leak. The object introduced to collect metrics for HTTP upgrade connections was not released for WebSocket connections once the connection was closed. This created a memory leak that, over time, could lead to a denial of service via an OutOfMemoryError. | Update tomcat to 9.0.43-2 |

## 5. Mitigation Plan

All the identified vulnerabilities can be addressed by upgrading their associated components. The vulnerabilities in the manual review can be addressed by continuing to implement the API, apply encryptions and input validation, correct server access from URL to POST method and apply error handling standards.