****

# CS 305 Project One

**Artemis Financial Vulnerability Assessment Report**

**Table of Contents**

[Document Revision History 3](#_heading=h.gjdgxs)

[Client 3](#_heading=h.30j0zll)

[Instructions 3](#_heading=h.1fob9te)

[Developer 4](#_heading=h.3znysh7)

[1. Interpreting Client Needs 4](#_heading=h.2et92p0)

[2. Areas of Security 4](#_heading=h.tyjcwt)

[3. Manual Review 4](#_heading=h.3dy6vkm)

[4. Static Testing 4](#_heading=h.1t3h5sf)

[5. Mitigation Plan 4](#_heading=h.4d34og8)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **7/18/22** | **Kendrick Free** | **First Edit, all categories filled out** |

## Client



## Instructions

Deliver this completed vulnerability assessment report, identifying your findings of security vulnerabilities and articulating recommendations for next steps to remedy the issues you have found.

Respond to the five steps outlined below and include your findings. Replace the bracketed text on all pages with your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

Kendrick Free

## 1. Interpreting Client Needs

Determine your client’s needs and potential threats and attacks associated with their application and software security requirements. Consider the following regarding how companies protect against external threats based on the scenario information:

* What is the value of secure communications to the company?

A successful company is only as successful as its security. Maintaining secure connections ensures that all forms of communication are secure and in turn generates more revenue. Lack of security can lead to things such as data breach and theft.

* Are there any international transactions that the company produces?

Artemis Financial deals with the finances of its clients. All information, especially financial information, should be always secure. Ensuring this security ensures the company's prosperity as well as its client.

* Are there governmental restrictions about secure communications to consider?

The only restrictions the government has on communication security is that the companies must share their information sharing methods which is backed by the Gramm-Leach-Bliley Act. Also, the government does have the right to access any forms of communications such as emails, social media, and stored information if a warrant is provided. Company security is left in the hands of the company.

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

Just as programming is always improving and changing, so are the methods of the people who attack companies. Security is and always will be an ongoing battle and the only way to maintain security is to constantly maintain and upgrade software to be as secure as possible.

* What are the “modernization” requirements that must be considered, such as the role of open source libraries and evolving web application technologies?

Open-source libraries allow users to adapt software to meet their specific needs. This will also allow future developers to add onto or change the software entirely should the need arise.

## 2. Areas of Security

Referring to the Vulnerability Assessment Process Flow Diagram, identify which areas of security are applicable to Artemis Financial’s software application. Justify your reasoning for why each area is relevant to the software application.

**Input Validation**: Since the company communicates worldwide an input validation would be critical to the security and functionality of the software. Also, since the input is validated, it ensures that the correct information is entered and that it is not potentially harmful to the software.

**API:** Ensuring the API is always up to date and secure will ensure that all communication within the company and externally is protected. Since the API is website based it is always accessible and needs to be constantly monitored.

**Cryptography:** Cryptography will ensure that the data in transit is encrypted and secure as well as whomever interacts with the website's data.

**Client/Server:** The clients who interact with the website will need to access the information they need, and the RESTful API will ensure they have access to any data they are permitted to. The server will receive client requests from the website API and will return data accordingly.

**Code Error:** The code will be reviewed, and any errors will be dealt with.

**Encapsulation:** This will ensure that the data accessed by clients cannot be changed or tampered with unless authorized. Tampering with a database can corrupt it entirely.

## 3. Manual Review

Continue working through the Vulnerability Assessment Process Flow Diagram. Identify all vulnerabilities in the code base by manually inspecting the code.

* In the pom.xml file the version of spring boot being utilized is version 2.2.4 which is outdated. The most current version is 2.6.6. Updating this is critical so that attackers can’t take advantage of outdated software.

## 4. Static Testing

Run a dependency check on Artemis Financial’s software application to identify all security vulnerabilities in the code. Record the output from dependency check report. Include the following:



1. The names or vulnerability codes of the known vulnerabilities

### bcprov-jdk15on-1.46.jar

### hibernate-validator-6.0.18.Final.jar

### jackson-databind-2.10.2.jar

### log4j-api-2.12.1.jar

### logback-core-1.2.3.jar

### snakeyaml-1.25.jar

### spring-boot-2.2.4.RELEASE.jar

### spring-core-5.2.3.RELEASE.jar

### spring-web-5.2.3.RELEASE.jar

### tomcat-embed-core-9.0.30.jar

### tomcat-embed-websocket-9.0.30.jar

1. A brief description and recommended solutions provided by the dependency check report (In the same order as the codes above).

Since there are 11 vulnerabilities the descriptions will be in the same order as the list above.

* 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.
* Hibernate's Bean Validation (JSR-380) reference implementation.
* General data-binding functionality for Jackson: works on core streaming API
* The Apache Log4j API
* logback-core module
* YAML 1.1 parser and emitter for Java
* Spring Boot
* Spring Core
* Spring Web
* Core Tomcat implementation
* Core Tomcat implementation

1. Attribution (if any) that documents how this vulnerability has been identified or documented previously (Same order as vulnerability codes in part A).

* In Bouncy Castle JCE Provider version 1.55 and earlier the DSA does not fully validate ASN.1 encoding of signature on verification. It is possible to inject extra elements in the sequence making up the signature and still have it validate, which in some cases may allow the introduction of 'invisible' data into a signed structure.
* 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.
* A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity.
* Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender. Fixed in Apache Log4j 2.12.3 and 2.13.1
* 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.
* The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564.
* \*\* UNSUPPORTED WHEN ASSIGNED \*\* spring-boot versions prior to version v2.2.11.RELEASE was vulnerable to temporary directory hijacking. This vulnerability impacted the org.springframework.boot.web.server.AbstractConfigurableWebServerFactory.createTempDir method. NOTE: This vulnerability only affects products and/or versions that are no longer supported by the maintainer.
* A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. The specific exploit requires the application to run on Tomcat as a WAR deployment. If the application is deployed as a Spring Boot executable jar, i.e. the default, it is not vulnerable to the exploit. However, the nature of the vulnerability is more general, and there may be other ways to exploit it.
* Pivotal Spring Framework through 5.3.16 suffers from a potential remote code execution (RCE) issue if used for Java deserialization of untrusted data. Depending on how the library is implemented within a product, this issue may or not occur, and authentication may be required. NOTE: the vendor's position is that untrusted data is not an intended use case. The product's behavior will not be changed because some users rely on deserialization of trusted data.
* 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. A number of changes were made to the default AJP Connector configuration in 9.0.31 to harden the default configuration. It is likely that users upgrading to 9.0.31, 8.5.51 or 7.0.100 or later will need to make small changes to their configurations.
* 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. A number of changes were made to the default AJP Connector configuration in 9.0.31 to harden the default configuration. It is likely that users upgrading to 9.0.31, 8.5.51 or 7.0.100 or later will need to make small changes to their configurations.

## 5. Mitigation Plan

After interpreting your results from the manual review and static testing, identify the steps to remedy the identified security vulnerabilities for Artemis Financial’s software application.

* Ensure that the system data as well as anyone who interacts with the system’s data is secure.
* Implement input validation for all input.
* Update Spring Framework boot to avoid any version errors as well as attackers exploiting outdated software

Work Cited

Manico, J., & Detlefsen, A. (2015). *Iron-clad java building secure web applications*. McGraw-Hill Education.

Hodges, S. (2013). Examining the Gramm-Leach-Bliley Act’s Opt-out Method for Protecting Consumer Data Privacy Rights on the Internet. Information & Communications Technology Law, 22(1), 60–85.