# CS 305 Project One

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **1/21/2025** | **Sumiko Mitchell** |  |

## Client



## Instructions

Submit this completed vulnerability assessment report. Replace the bracketed text with the relevant information. In this report, identify your security vulnerability findings and recommend 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 include images or supporting materials. If you include them, make certain to insert them in 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

Sumiko Mitchell

**1.  Interpreting Client Needs**

Artemis Financial is a consulting firm that provides clients with customized savings, retirement, investment, and insurance plans. To enhance security and maintain customer trust, they have asked Global Rain to modernize their RESTful web API to protect sensitive financial data from external threats—essential to Artemis Financial’s business integrity.

The Value of Secure Communications

Secure communication is valuable for Artemis Financial because it mitigates risk. It protects customers' data from bad actors and ensures regulatory compliance. Without proper encryption and authentication, Artemis Financial exposes itself to data breaches, financial loss, legal penalties, and reputational damage.

# International Financial Transactions

The materials provided to Global Rain do not explicitly state whether Artemis Financial currently conducts international transactions. However, given its role in financial consulting, this could be a potential area of growth that would require additional consideration of security and regulatory compliance.

Governmental Restrictions on Secure Communications

While U.S. financial regulations mandate security measures like encryption and secure APIs, they generally do not restrict secure communications within the U.S. The main restrictions apply to international transactions, where some countries limit encryption and data transfer methods.

External Threats Now and in the Immediate Future

Artemis Financial currently faces threats such as Man-in-the-Middle (MITM) attacks, SQL injection, Cross-Site Scripting (XSS), and API security vulnerabilities, exposing sensitive customer and business data to unauthorized access. The risks are expected to grow with the rise of AI-assisted cyberattacks, accelerating the discovery of exploitable code through automated vulnerability scanning and exploit discovery.

Modernizing Security Requirements

Artemis Financial must modernize its security by addressing risks from open-source libraries and evolving web technologies. By implementing continuous security testing, automated dependency management, and secure coding standards, Artemis can better protect against current and emerging

**2. Areas of Security**

* Input Validation: All inputs must be validated and sanitized to prevent injection attacks.
* API Security: The RESTful API must be secured against unauthorized access and breaches.
* Cryptography: Strong encryption is required to protect data and meet compliance standards.
* Client/Server: Secure communication channels must maintain data integrity and confidentiality.
* Code Quality: Secure coding practices, automated analysis, and reviews ensure maintainable, vulnerability-free software.

**3. Manual Review**

Vulnerabilities identified by manual code inspection:

**CRUDController.java**

1. The /read endpoint lacks authentication, allowing unrestricted access to data.

**customer.java**

1. The showInfo() method exposes account\_number without proper access control.

**DocData.java**

1. The read\_document() method constructs database connections without using prepared statements, making it vulnerable to SQL injection.
2. Database credentials are hardcoded ("root”), exposing sensitive information.
3. The read\_document() method exposes detailed stack traces through e.printStackTrace(), potentially leaking system details.

**GreetingController.java**

1. The /greeting endpoint accepts user input (name) without sanitization, making it vulnerable to XSS.
2. API endpoints lack protection against excessive requests, making them susceptible to brute force and denial-of-service (DoS) attacks.

**General Configuration Issues**

1. No logging of security events or errors, reducing the ability to detect and respond to attacks.

**4. Static Testing**

| **DEPENDENCY NAME** | **CVE ID** | **DESCRIPTION/RECOMMENDATION** |
| --- | --- | --- |
| spring-boot-starter-web-2.2.4.RELEASE.jar  spring-boot-2.2.4.RELEASE.jar  (Note: Manual Review #1) | CVE-2022-27772 | \* Spring Boot versions prior to 2.2.11.RELEASE are vulnerable to temporary directory hijacking in AbstractConfigurableWebServerFactory.createTempDir(). This could allow attackers to manipulate temporary directories, leading to potential unauthorized file access or privilege escalation. Spring Boot maintainers no longer support these versions.  \* Upgrade to Spring Boot 2.2.11.RELEASE or later. |
| snakeyaml-1.25.jar | CVE-2017-18640 | \* The Alias feature in SnakeYAML before 1.26 allows entity expansion during a load operation, a related issue to CVE-2003-1564.  \* Upgrade to SnakeYAML version 1.26 or later. |
| spring-webmvc-5.2.3.RELEASE.jar  spring-web-5.2.3.RELEASE.jar  spring-expression-5.2.3.RELEASE.jar  spring-core-5.2.3.RELEASE.jar | CVE-2020-5421 | \* 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.  \* Upgrade to Spring Framework version 5.2.9 or later. Additionally, reviewing and configuring your application's session handling to prevent the use of jsessionid in URLs can further enhance security. |
| tomcat-embed-websocket-9.0.30.jar  tomcat-embed-core-9.0.30.jar | CVE-2020-1938 | \* Apache Tomcat versions 9.0.0.M1 to 9.0.30 are vulnerable to a file read/inclusion vulnerability known as "Ghostcat." This issue arises from the Apache JServ Protocol (AJP) connector being enabled by default and listening on all configured IP addresses. An attacker can exploit this vulnerability to read or include arbitrary files in the web application, potentially leading to information disclosure or remote code execution.  \* Upgrade to Apache Tomcat version 9.0.31 or later. |
| bcprov-jdk15on-1.46.jar  (Note: Manual Review #4) | CVE-2013-0169 CVE-2018-5382 | \* CVE-2013-0169: This vulnerability pertains to the "Lucky Thirteen" attack, which affects implementations of the TLS protocol using cipher block chaining (CBC) mode. The Bouncy Castle Java library, in versions prior to 1.47, is susceptible to this attack, potentially allowing attackers to perform plaintext recovery via timing analysis.  \* CVE-2018-5382: The Bouncy Castle Java library's BKS version 1 keystore utilizes an HMAC that is only 16 bits long, making it vulnerable to integrity compromise. An attacker could exploit this weakness to tamper with or forge keystore contents. This issue is addressed in Bouncy Castle version 1.47 and later, where the BKS format was updated to use a 160-bit HMAC.  \* Upgrade to Bouncy Castle version 1.47 or later. |
| logback-core-1.2.3.jar  logback-classic-1.2.3.jarz  (Note: Manual Review #8) | CVE-2021-42550 | \* Logback versions up to 1.2.7 are vulnerable to arbitrary code execution if an attacker can modify the Logback configuration file. This vulnerability arises because Logback's configuration allows for the inclusion of data from remote servers via Java Naming and Directory Interface (JNDI) lookups. If an attacker has write access to the configuration file, they could craft malicious configurations that exploit this behavior, leading to potential remote code execution.  \* Upgrade to Logback version 1.2.8 or later. |
| jackson-databind-2.10.2.jar  (Note: Manual Review #6) | CVE-2020-36518  CVE-2022-42003 | \* CVE-2020-36518: A vulnerability in Jackson Databind allows for potential remote code execution due to unsafe deserialization. This issue arises when polymorphic type handling is enabled, and an attacker can supply malicious input to be deserialized. This affects versions prior to 2.12.  \* CVE-2022-42003: Jackson Databind is vulnerable to a Denial of Service (DoS) attack due to a flaw in the StdKeyDeserializer class. An attacker can craft malicious input that causes excessive memory consumption during deserialization, leading to application instability. This affects versions prior to 2.13.4. \* Upgrade to Jackson Databind version 2.13.4 or later. |
| hibernate-validator-6.0.18.Final.jar  (Note: Manual Review #3) | CVE-2020-10693 | \* A flaw in Hibernate Validator's message interpolation processor allows invalid Expression Language (EL) expressions to be evaluated as if they were valid. This vulnerability enables attackers to bypass input sanitation controls that developers may have implemented when handling user-controlled data in error messages.  \* Upgrade to Hibernate Validator version 6.0.20.Final or later. |
| log4j-api-2.12.1.jar | CVE-2021-44228  CVE-2021-45046: | \* CVE-2021-44228: A critical vulnerability in Apache Log4j 2.x versions allows remote code execution via the Java Naming and Directory Interface (JNDI) lookup feature. An attacker can exploit this by crafting malicious input that is logged by the application, leading to arbitrary code execution. This affects versions from 2.0-beta9 to 2.12.1.  \* CVE-2021-45046: An incomplete fix for CVE-2021-44228 in Log4j 2.15.0 allowed attackers under certain non-default configurations to craft malicious input data using a JNDI Lookup pattern, resulting in information leakage and potential remote code execution. This affects versions from 2.0-beta9 to 2.12.1.  \* Upgrade to Log4j version 2.12.3 (for Java 7) or 2.17.0 (for Java 8 and later). |

**5. Mitigation Plan**

The current setup at Artemis Financial needs some important security upgrades. We need to fix how the system handles database access and find a better way to store login credentials. The tools we use to manage user logins are outdated, which could let hackers slip through and access sensitive information. We should also add checks to make sure users can't input malicious code or tamper with data.

To better protect the system, we should improve how we track suspicious activities and add safeguards against attacks that could overwhelm our servers. We'll also need to update any outdated software to patch known security holes. After making these changes, we should do a thorough security check to make sure everything's properly protected. Lastly, we should create clear security guidelines to help us stay protected as new threats emerge.