**3-1 Project One: Vulnerability Assessment Report**

Patrick S. Coyne

Global Campus, Southern New Hampshire University

CS 305: Software Security

Instructor: Barbara Gladney

July 19, 2025

# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **July 19, 2025** | **Patrick S. Coyne** |  |

## Client



## 

## Developer

Patrick S. Coyne

**1. Interpreting Client Needs**

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

Securing communication is a necessary step for all web-based applications. For Artemis Financial, secure communication is crucial to protect customers’ sensitive data, including passwords, account numbers, payment methods, social security numbers, and addresses. Interception of customer data could lead to identity and monetary theft.

*Are there any international transactions that the company produces?*

Whether Artemis Financial operates internationally is unclear from the scenario; however, given the nature of its business, it is likely. If Artemis Financial accepts international clients and conducts international transactions, its software may require additional compliance, such as the European Union’s General Data Protection Regulation (GDPR).

*Are there governmental restrictions on secure communications to consider?*

Yes. At the very minimum, software must comply with the Gramm-Leach-bliley Act (GLBA). The GLBA is a United States regulation designed to protect customer financial data. GLBA requires compliance with various coding practices, including encrypting communications and implementing other security controls.

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

Present threats include API injection, distributed denial of service (DDoS) attacks, phishing scams (for clients and Artemis Financial employees), and man-in-the-middle interception of sensitive data. In the immediate future, outdated dependencies create the most significant threat. Keeping dependencies up to date mitigates many exploitable vulnerabilities and keeps software secure from attackers.

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

As mentioned, vulnerabilities are frequently found in open-source libraries. Therefore, checking for updated versions of dependencies is critical to software security. Additionally, web applications evolve, and new technologies should be adopted as they become available. For example, Spring Security can be used for API authentication, and HTTPS (instead of HTTP) can be used for encrypting communications.

**2. Areas of Security**

Input Validation: The application has two endpoints that accept input (/greeting and /read). All input should be validated to prevent injection attacks and application crashes that can occur when unexpected input is entered.

APIs: The application uses the REST API for its endpoints. Authentication should always be used for secure API interactions.

Cryptography: Whenever data is in transit (the code base has endpoints), encryption is necessary.

Client/Server: The application utilizes HTTP requests; therefore, Client/Server security is a concern.

Code Error: Secure error handling should be implemented in all applications to avoid crashes and unexpected behavior that could be exploited.

Code Quality: As with secure error handling, using secure coding practices/patterns is a necessary practice to keep applications from being exploitable, and certainly applies to the Artemis Financial web-based application.

Encapsulation: Protecting sensitive data, such as financial information, using encapsulation is critical. Encapsulation prevents unauthorized data modifications from happening.

**3. Manual Review**

**Vulnerability 1 (Input Validation):**

In CRUDController.java, the /read endpoint takes user input without validating it. The DocData.java file contains comments indicating that future implementation will use the input from the /read endpoint to query an SQL database. Therefore, an attacker could use the /read endpoint for an SQL injection attack if the input is not validated.

**Vulnerability 2 (Input Validation):**

In GreetingController.java, the /greeting endpoint takes user input without validating it. Currently, the input is reformatted as a string and returned to the user, which is safe. However, if the input is used for other purposes in the future (e.g., in an SQL query), an attacker could exploit the endpoint as a means of an injection attack.

**Vulnerability 3 (APIs):**

The /greeting and /read endpoints (GreetingController.java and CRUDController.java, respectively) do not use authentication, which allows unauthorized access. The lack of authentication enables anyone to access sensitive data. Additionally, unauthorized access opens the door to DDoS attacks.

**Vulnerability 4 (Cryptography):**

The RestServiceApplication.java does not use HTTPS, resulting in all communications being transmitted in plain text and making them vulnerable to man-in-the-middle attacks.

**Vulnerability 5 (Code Error):**

In DocData.java, a try/catch block is used to handle an SQL Exception. However, the catch statement prints the stack trace. Therefore, when an exception is caught, the user may see the stack trace in the HTTP response, revealing details that could help an attacker develop a malicious injection.

**Vulnerability 6 (Encapsulation):**

In customer.java, the public deposit method accepts a parameter (a), which is not validated. Although an integer is required, it is possible to pass the deposit method a negative integer, which can cause unexpected results.

Vulnerability 7 (Code Quality):

In DocData.java, the username and password to access the database are hardcoded. Anyone who has access or gains access to the code base will have the credentials. Hardcoding usernames and passwords is an anti-pattern and can lead to compliance violations.

**Vulnerability 8 (Encapsulation):**

In customer.java, account\_balance is not defined as private. Therefore, account\_balance can be modified outside of the customer class.

**4. Static Testing**

### The following is a summary of the Dependency-Check report (July 13, 2025):

### **bcprov-jdk15on-1.46.jar**: 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.

### **CVE Codes:** CVE-2013-1624, CVE-2015-6644, CVE-2015-7940, CVE-2016-1000338, CVE-2016-1000341, CVE-2016-1000342, CVE-2016-1000343, CVE-2016-1000345, CVE-2016-1000346, CVE-2018-5382, CVE-2020-0187, CVE-2023-33201, CVE-2023-33202, CVE-2024-29857, CVE-2024-30171, CVE-2024-34447

Descriptions of Vulnerabilities: This version of The Bouncy Castle Crypto package is vulnerable to timing side-channel attacks, invalid curve attacks, data injection, weak key generation, padding oracle attacks, denial-of-service attacks, excessive CPU consumption, and various other flaws. Every vulnerability can be resolved by upgrading to the newest version (1.8).

**Attribution:** Identified by Dependency-Check, National Vulnerability Database, and Sonatype OSS Index.

spring-boot-2.2.4.RELEASE.jar: Spring Boot.

**CVGE Codes:** CVE-2016-1000027, CVE-2018-11039, CVE-2018-11040, CVE-2018-1257, CVE-2020-5421, CVE-2022-22950, CVE-2022-22965, CVE-2022-22968, CVE-2022-22970, CVE-2022-27772, CVE-2023-20861, CVE-2023-20873, CVE-2023-20883, CVE-2024-22259

Descriptions of Vulnerabilities: This version of Spring Boot is vulnerable to remote code execution, Cross-Site Tracing (XST) attacks, denial-of-service attacks, reflected file download (RFD) attacks, directory hijacking, and server-side request forgery (SSRF) attacks. All vulnerabilities can be resolved by updating to the newest version (3.5.3).

**Attribution:** Identified by Dependency-Check and the National Vulnerability Database.

logback-classic-1.2.3.jar: Logback Classic module.

**CVE Codes**: CVE-2023-6378, CVE-2021-42550

Descriptions of Vulnerabilities: This version of Logback Classic is vulnerable to remote code execution (RCE) and denial-of-service (DoS) attacks. Upgrading to version 1.5.18 will resolve the issues.

**Attribution:** Identified by Dependency-Check and the National Vulnerability Database.

logback-core-1.2.3.jar: Logback Core module.

**CVE Codes:** CVE-2023-6378, CVE-2021-42550, CVE-2024-12798, CVE-2024-12801

Descriptions of Vulnerabilities: This version of Logback Core is vulnerable to remote code execution (RCE), denial-of-service attacks, and server-side request forgery (SSRF) attacks. Upgrading to the newest version (1.5.18) resolves the issues.

**Attribution:** Identified by Dependency-Check, National Vulnerability Database, and Sonatype OSS Index.

log4j-api-2.12.1.jar: The Apache Log4j API.

**CVE Codes**: CVE-2020-9488, CVE-2021-44228, CVE-2021-44832, CVE-2021-45046, CVE-2021-45105

Descriptions of Vulnerabilities: This version of the Apache Log4j API is vulnerable to man-in-the-middle attacks, remote code execution (RCE), information leakage, and denial-of-service attacks. The newest version (2.25.1) resolves the vulnerabilities.

**Attribution:** Identified by Dependency-Check and the National Vulnerability Database.

snakeyaml-1.25.jar: SnakeYAML.

**CVE Codes:** CVE-2022-1471, CVE-2021-4235, CVE-2022-3064, CVE-2027-18640, CVE-2022-25857, CVE-2022-38749, CVE-2022-38751, CVE-2022-38752, CVE-2022-41854, CVE-2022-38750

Descriptions of Vulnerabilities: This version of SnakeYAML is vulnerable to excessive CPU usage, denial-of-service attacks, and remote code execution (RCE). Upgrading to the newest version (2.0) resolves most issues. Additionally, for CVE-2022-1471, it is recommended to use SnakeYaml’s SafeConstructor when parsing untrusted content to restrict deserialization

**Attribution:** Identified by Dependency-Check and the National Vulnerability Database. CVE-2022-1471 mitigation recommended by Google Security Research.

jackson-databind-2.10.2.jar: General data-binding functionality for Jackson: works on core streaming API.

**CVE Codes:** CVE-2020-25649, CVE-2020-36518, CVE-2021-46877, CVE-2022-42003, CVE-2022-42004, CVE-2023-35116

Descriptions of Vulnerabilities: This version of Jackson Databind is vulnerable to XML external entity (XXE) attacks, denial-of-service attacks, and resource exhaustion. Upgrading to the newest version (2.19.2) resolves the issues.

**Attribution:** Identified by Dependency-Check and the National Vulnerability Database.

jackson-core-2.10.2.jar: Core Jackson processing abstractions (aka Streaming API), implementation for JSON.

**CVE Codes**: CVE-2025-52999, CVE-2025-49128

**Descriptions of Vulnerabilities:** This version of Jackson core is vulnerable to information exposure and buffer overflow. Upgrading to the newest version (2.19.2) resolves the issues.

**Attribution:** Identified by Dependency-Check, National Vulnerability Database, and Sonatype OSS Index.

tomcat-embed-core-9.0.30.jar: Core Tomcat Implementation.

**CVE Codes:** CVE-2019-17569, CVE-2020-11996, CVE-2020-13934, CVE-2020-13935, CVE-2020-13943, CVE-2020-17527, CVE-2020-1935, CVE-2020-1938, CVE-2020-8022, CVE-2020-9484, CVE-2021-24122, CVE-2021-25122, CVE-2021-25329, CVE-2021-30640, CVE-2021-33037, CVE-2021-41079, CVE-2021-43980, CVE-2022-29885, CVE-2022-34305, CVE-2022-42252, CVE-2023-28708, CVE-2023-41080, CVE-2023-42795, CVE-2023-44487, CVE-2023-45648, CVE-2023-46589, CVE-2024-21733, CVE-2024-23672, CVE-2024-24549, CVE-2024-34750, CVE-2024-38286, CVE-2024-50379, CVE-2024-52316, CVE-2024-54677, CVE-2025-24813, CVE-2025-31651, CVE-2025-46701, CVE-2025-48988, CVE-2025-49124, CVE-2025-49125

Descriptions of Vulnerabilities: This version of Tomcat Core is vulnerable to HTTP request smuggling, excessive CPU usage, denial-of-service attacks, information leakage, remote code execution (RCE), cross-site scripting (XSS) attacks, and authentication bypass. Upgrading to the newest version (11.0.9) resolves the issues.

**Attribution:** Identified by Dependency-Check and the National Vulnerability Database.

hibernate-validator-6.0.18.Final.jar: Hibernate's Bean Validation (JSR-380) reference implementation.

**CVE Codes**: CVE-2025-35036, CVE-2023-1932, CVE-2020-10693

Descriptions of Vulnerabilities: This version of Hibernate’s Bean Validation is vulnerable to HTML injection, cross-site scripting (XSS) attacks, and code injection. Upgrading to the newest version (9.0.0) resolves the issues.

**Attribution:** Identified by Dependency-Check, National Vulnerability Database, and Sonatype OSS Index.

spring-web-5.2.3.RELEASE.jar: Spring Web.

**CVE Codes**: CVE-2016-1000027, CVE-2022-22965, CVE-2024-38809, CVE-2024-22243, CVE-2024-22262, CVE-2024-22259, CVE-2021-22118, CVE-2025-41234, CVE-2020-5421, CVE-2022-22950, CVE-2022-22971, CVE-2023-20861, CVE-2023-20863, CVE-2022-22968, CVE-2022-22970, CVE-2021-22060, CVE-2021-22096, CVE-2024-38828

Descriptions of Vulnerabilities: This version of Spring Web is vulnerable to remote code execution (RCE), reflected file download (RFD) attacks, unauthorized log insertion, privilege escalation, denial-of-service attacks, open redirect attacks, and server-side request forgery (SSRF) attacks. Upgrading to the newest version (7.0.0) resolves the issues.

**Attribution:** Identified by Dependency-Check, National Vulnerability Database, and Sonatype OSS Index.

spring-webmvc-5.2.3.RELEASE.jar: Spring WebMVC.

**CVE Codes**: CVE-2016-1000027, CVE-2022-22965, CVE-2024-38816, CVE-2024-22259, CVE-2021-22118, CVE-2020-5421, CVE-2022-22950, CVE-2022-22971, CVE-2023-20861, CVE-2023-20863, CVE-2022-22968, CVE-2022-22970, CVE-2021-22060, CVE-2021-22096

Descriptions of Vulnerabilities: This version of Spring WebMVC is vulnerable to remote code execution (RCE), reflected file download (RFD) attacks, unauthorized log insertion, privilege escalation, denial-of-service attacks, open redirect attacks, and server-side request forgery (SSRF) attacks. Upgrading to the newest version (7.0.0) resolves the issues.

**Attribution:** Identified by Dependency-Check, National Vulnerability Database, and Sonatype OSS Index.

spring-context-5.2.3.RELEASE.jar: Spring Context.

**CVE Codes:** CVE-2016-1000027, CVE-2022-22965, CVE-2024-22259, CVE-2021-22118, CVE-2020-5421, CVE-2022-22950, CVE-2022-22971, CVE-2023-20861, CVE-2023-20863, CVE-2022-22968, CVE-2022-22970, CVE-2021-22060, CVE-2021-22096, CVE-2025-22233

Descriptions of Vulnerabilities: This version of Spring Context is vulnerable to remote code execution (RCE), reflected file download (RFD) attacks, unauthorized insertion of log entries, privilege escalation, denial-of-service attacks, open redirect attacks, and server-side request forgery (SSRF) attacks. Upgrading to the newest version (7.0.0) resolves the issues.

**Attribution:** Identified by Dependency-Check, National Vulnerability Database, and Sonatype OSS Index.

spring-expression-5.2.3.RELEASE.jar: Spring Expression Language (SpEL).

**CVE Codes:** CVE-2016-1000027, CVE-2022-22965, CVE-2024-22259, CVE-2021-22118, CVE-2020-5421, CVE-2022-22950, CVE-2022-22971, CVE-2023-20861, CVE-2023-20863, CVE-2024-38808, CVE-2022-22968, CVE-2022-22970, CVE-2021-22060, CVE-2021-22096

Descriptions of Vulnerabilities: This version of Spring Context is vulnerable to remote code execution (RCE), reflected file download (RFD) attacks, unauthorized insertion of log entries, privilege escalation, denial-of-service attacks, open redirect attacks, and server-side request forgery (SSRF) attacks. Upgrading to the newest version (7.0.0) resolves the issues.

**Attribution:** Identified by Dependency-Check, National Vulnerability Database, and Sonatype OSS Index.

spring-core-5.2.3.RELEASE.jar: Spring Core.

**CVE Codes:** CVE-2016-1000027, CVE-2022-22965, CVE-2024-22259, CVE-2021-22118, CVE-2020-5421, CVE-2022-22950, CVE-2022-22971, CVE-2023-20861, CVE-2023-20863, CVE-2022-22968, CVE-2022-22970, CVE-2021-22060, CVE-2021-22096

Descriptions of Vulnerabilities: This version of Spring Context is vulnerable to remote code execution (RCE), reflected file download (RFD) attacks, unauthorized insertion of log entries, privilege escalation, denial-of-service attacks, open redirect attacks, and server-side request forgery (SSRF) attacks. Upgrading to the newest version (7.0.0) resolves the issues.

**Attribution:** Identified by Dependency-Check, National Vulnerability Database, and Sonatype OSS Index.

**5. Mitigation Plan**

### The manual code review revealed 8 vulnerabilities in the codebase. The static test report revealed 176 vulnerabilities in 15 dependencies.

**Action List:**

*Mitigate Vulnerabilities Identified During Manual Code Review*

Vulnerability 1 (No input validation in /read endpoint): Add input validation using @Pattern only to allow letters and numbers at the /read endpoint.

**Vulnerability 2 (No input validation in /greeting endpoint):** Similar to mitigating vulnerability 1, add input validation using @Pattern.

Vulnerability 3 (No authentication at endpoints): Implement Spring Security to control access to endpoints.

**Vulnerability 4 (Communications are not encrypted):** In application.properties, add Secure Sockets Layers. In RestServiceApplication.java, use HTTPS.

**Vulnerability 5 (Stack Trace is printed when exception is caught):** Instead of using e.printStackTrace() to print the exception, log it securely.

**Vulnerability 6 (Unvalidated parameter in customer.java):** Make the deposit method private. Additionally, validate the a to ensure it is a non-negative number.

**Vulnerability 7 (Username and password are hardcoded):** Move username and password to application.properties. When they are needed, use @Value to retrieve them.

**Vulnerability 8****(account\_balance can be accessed outside its class):** Set account\_balance as private and create a public method to return its value.

*Mitigate Vulnerabilities Identified During Static Test*

* Upgrade the 15 dependencies identified in the dependency-check report.
* Implement SafeConstructor for snakeYAML.