# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **5/25/2025** | **Heath Davis** |  |

## 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

Heath Davis

**1. Interpreting Client Needs**

Artemis Financial’s RESTful API handles sensitive financial data, requiring robust security. Key considerations:

* Secure Communications: Essential for compliance (GDPR, PCI-DSS) and client trust.
* Threat Landscape: Risks include API attacks (RCE, CSRF), dependency vulnerabilities, and data breaches.
* Modernization: Requires updates to:
  + Spring Boot 3.x (current LTS)
  + Java 17+ (for security patches)
  + HTTPS/TLS 1.3 (disable deprecated protocols).

**2. Areas of Security**

Relevant to Artemis Financial’s application:

1. Dependency Security (Critical): Outdated libraries with known CVEs.
2. API Protection: Missing CSRF tokens, rate limiting.
3. Data Integrity: Weak crypto (Bouncy Castle 1.46).
4. Logging Risks: Vulnerable Log4j/SnakeYAML.

**3. Manual Review**

| **#** | **Vulnerability** | **Location** | **Risk Level** | **Details** |
| --- | --- | --- | --- | --- |
| 1 | Outdated Java Version (1.8) | pom.xml (Line 12) | High | JDK 1.8 lacks modern security patches; upgrade to Java 17+. |
| 2 | No HTTPS Enforcement | Missing in application.properties | High | API accepts HTTP traffic; risk of MITM attacks. |
| 3 | Hardcoded Database Credentials | application.yml | Critical | Plaintext credentials in config file. |
| 4 | Missing CSRF Protection | REST Controller classes | High | No CSRF tokens for state-changing requests. |
| 5 | Excessive Error Details | GlobalExceptionHandler.java | Medium | Stack traces exposed in production errors. |
| 6 | No Input Validation | UserController.java (POST methods) | High | Endpoints accept raw/unfiltered user input. |
| 7 | Insecure CORS Configuration | WebSecurityConfig.java | Medium | Wildcard (\*) allowed for origins. |
| 8 | Deprecated Encryption (SHA-1) | PasswordUtil.java | High | Uses insecure hashing algorithm; upgrade to bcrypt. |
| 9 | Missing Rate Limiting | N/A (Not implemented) | Medium | API vulnerable to brute force/DDoS. |
| 10 | Insecure File Uploads | FileUploadController.java | High | No file type/size validation. |

**Key Justifications**

* **Critical Risks (#3, #6): Directly expose sensitive data or allow RCE.**
* **High Risks (#1, #2, #4, #8, #10): Could lead to unauthorized access or data breaches.**
* **Medium Risks (#5, #7, #9): Impact availability or leak debugging info.**

**4. Static Testing**

**Critical Vulnerabilities**

| **Dependency** | **CVE ID** | **Fix** |
| --- | --- | --- |
| spring-boot:2.2.4 | CVE-2022-22965 | Upgrade to ≥ 3.1.6 |
| log4j-api:2.12.1 | CVE-2021-44228 | Replace with Log4j ≥ 2.17.0 |
| tomcat-embed:9.0.30 | CVE-2020-1938 | Upgrade to ≥ 10.1.0 |

**High-Risk Vulnerabilities**

| **Dependency** | **CVE ID** | **Fix** |
| --- | --- | --- |
| bcprov-jdk15on:1.46 | CVE-2016-1000341 | Upgrade to ≥ 1.70 |
| snakeyaml:1.25 | CVE-2022-1471 | Upgrade to ≥ 2.0 |

**5. Mitigation Plan**

**Immediate Actions (Critical)**

1. **Patch Spring Boot**:

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.1.6</version>

</parent>

2. **Replace Log4j**: Use Logback (already in Spring Boot 3.x).

3.**Upgrade Bouncy Castle**:

<dependency>

<groupId>org.bouncycastle</groupId>

<artifactId>bcprov-jdk18on</artifactId>

<version>1.77</version>

</dependency>

**Long-Term Strategies**

* **Automate Dependency Checks**: Integrate OWASP scans into CI/CD.
* **API Hardening**: Add Spring Security with OAuth2.
* **Monitoring**: Deploy WAF (Web Application Firewall).

A screenshot of a computer

AI-generated content may be incorrect.