# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **2025/9/Sep** | **Chris Howard** |  |

## 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

[Insert your name here.]

**1. Interpreting Client Needs**

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

* What is the value of secure communications to the company?
* Are there any international transactions that the company produces?
* Are there governmental restrictions on secure communications to consider?
* What external threats might be present now and in the immediate future?
* What modernization requirements must be considered, such as the role of open-source libraries and evolving web application technologies?

**Value of secure communications:** Artemis Financial handles sensitive PII and financial data (accounts, balances, plans). Confidentiality, integrity, and availability are essential to protect clients, meet regulatory expectations, and preserve brand trust. End-to-end protection is required for all client API calls and admin operations.

**International transactions:** Financial consulting can involve international clients, cross-border communications, and data residency concerns. Assume some data transits international networks; enforce strong cryptography and review data-transfer requirements.

**Governmental restrictions:** Compliance with U.S. privacy/financial regs and possibly GDPR/other cross-border laws. Use modern TLS, vetted cipher suites, and secure key management.

**External threats (now/near future):** Credential stuffing, API abuse, injection attacks (SQL/SpEL/NoSQL), deserialization issues, dependency supply-chain vulnerabilities, misconfiguration, and data exfiltration via overly permissive endpoints.

**Modernization requirements:**

**Open-source libraries:** Keep Spring, Jackson, DB connectors, and transitive deps current; use SBOM and automated dependency scanning.

**Evolving web tech:** Harden REST endpoints, implement least privilege, observability, and secure CI/CD with SAST/DAST and dependency checks.

**2. Areas of Security**

Refer to the vulnerability assessment process flow diagram. Identify which areas of security apply to Artemis Financial’s software application. Justify your reasoning for why each area is relevant to the software application.

**Input Validation:** Multiple endpoints accept query parameters (e.g., name, business\_name); must sanitize/validate.

**APIs:** Public REST endpoints require authentication/authorization, rate limiting, and error-handling standards.

**Cryptography:** Enforce TLS, secure cookie/headers, secret management (no hardcoded credentials).

**Code Error:** Avoid stack-trace leakage; centralized exception mapping; safe logging.

**3. Manual Review**

Continue working through the vulnerability assessment process flow diagram. Identify all vulnerabilities in the code base by manually inspecting the code.

Hardcoded DB credentials & no encryption – DocData.read\_document: DriverManager.getConnection. Secrets in code, weak creds, no SSL.  
*Risk:* Credential theft, lateral movement.

No resource cleanup – DocData.read\_document: Connection never closed; no try-with-resources.  
*Risk:* Leaks/exhaustion DoS.

Information disclosure via toString() – CRUDController.CRUD() returns new CRUD(doc.toString()). Default Object.toString() reveals class + hash, internal details.  
*Risk:* Recon aid, unstable output in API.

Missing input validation – /read business\_name and /greeting name are unsanitized.  
*Risk:* Injection/XSS if later rendered or used in queries.

No authentication/authorization on endpoints – All controllers (/read, /greeting) are public.  
*Risk:* Data exposure/abuse by unauthenticated clients.

Stack trace exposure – DocData.read\_document uses e.printStackTrace().  
*Risk:* Reveals internals to logs/console; can leak DB info in some setups.

Outdated driver reference pattern – Commented Class.forName("com.mysql.jdbc.Driver") hints at the legacy MySQL driver; combined with hardcoded URL suggests potential use of outdated DB driver when integrated.

**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 the dependency-check report. Include the following items:

* The names or vulnerability codes of the known vulnerabilities
* A brief description and recommended solutions provided by the dependency-check report
* Any attribution that documents how this vulnerability has been identified or documented previously

Dependency: bcprov-jdk15on-1.46.jar (Bouncy Castle JCE Provider)

1. CVE-2024-34447
   * Description: Improper validation of certificate with host mismatch. The software does not properly ensure that the presented certificate is associated with the expected host.
   * CWE: CWE-297 Improper Validation of Certificate with Host Mismatch
   * Severity: HIGH (7.7)
   * Remediation: Update to the latest fixed version of Bouncy Castle (> 1.78).
   * NVD Link: CVE-2024-34447
2. CVE-2016-1000338
   * Description: DSA does not fully validate ASN.1 encoding of signatures on verification. This allows injection of extra elements into a signed structure, potentially introducing invisible data.
   * CWE: CWE-347 Improper Verification of Cryptographic Signature
   * Severity: HIGH (7.5)
   * Remediation: Upgrade Bouncy Castle JCE Provider to version > 1.56.
   * NVD Link: CVE-2016-1000338

**5. Mitigation Plan**

Interpret the results from the manual review and static testing report. Then identify the steps to mitigate the identified security vulnerabilities for Artemis Financial’s software application.

Upgrade Vulnerable Libraries

* Bouncy Castle (bcprov-jdk15on-1.46): Multiple HIGH severity vulnerabilities (e.g., CVE-2016-1000338, CVE-2024-29857) affect cryptographic validation and key generation.
  + Action: Upgrade to the latest secure version (> 1.78).
  + Impact: Ensures strong cryptographic validation and prevents denial-of-service or signature forgery.
* Jackson (jackson-databind-2.10.2 & jackson-core-2.10.2): Several HIGH severity deserialization flaws (remote code execution).
  + Action: Upgrade to > 2.15.x.
  + Impact: Eliminates dangerous gadget chains that attackers can exploit.
* Logback (logback-classic and logback-core 1.2.3): Known HIGH severity vulnerabilities in logging configuration.
  + Action: Upgrade to > 1.2.13 or 1.3.x depending on compatibility.
  + Impact: Prevents logging-related remote attacks.

Implement Continuous Monitoring

* Integrate OWASP Dependency-Check into the CI/CD pipeline.
* Automate scanning to detect new CVEs immediately.

Adopt a Secure Update Policy

* Track vendor security advisories and NVD feeds.
* Apply security patches within defined SLAs (CRITICAL within 7 days, HIGH within 14 days).

Conduct Regression Testing

* After dependency upgrades, run integration and unit tests to confirm compatibility and prevent functionality breaks.