CS 305 Project One

Document Revision History

Version	Date	Author	Comments
1.0	July 18th 2025	Liam Farrell	

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
Liam Farrell

1. Interpreting Client Needs

Artemis Financial is a consulting firm specializing in personalized financial plans, which likely involve sensitive customer data such as social security numbers, income details, and investment records. As the company modernizes its operations with a RESTful web API, the need for robust security is paramount.

Client Security Needs and Potential Threats:

- **Secure Communications:** As a financial institution, Artemis must ensure all communications are encrypted using TLS. Data integrity and confidentiality are essential, especially for client-facing REST APIs.
- **International Considerations:** If Artemis engages in international financial planning, it must comply with standards such as GDPR, PCI DSS, and U.S. export control laws regarding cryptography.
- **External Threats:** Potential threats include man-in-the-middle (MITM) attacks, injection attacks, outdated dependencies, misconfigured authorization, and exposure of sensitive data.
- **Modernization Considerations:**
    - **Use of Open-Source Libraries:** Widely used libraries like mongo-java-driver can introduce known vulnerabilities if not kept up to date.
    - **Evolving Technologies:** As Artemis transitions to modern RESTful APIs, ensuring secure API design (authentication, rate limiting, input/output validation) is essential.

2. Areas of Security

Based on the vulnerability assessment process flow diagram and the client's RESTful API implementation, the following security areas apply:

Area	Relevance
Input Validation	Prevents injection attacks, malformed payloads, and ensures client-submitted data adheres to expected formats.
APIs	The system uses RESTful endpoints; therefore, endpoint authentication, authorization, and input/output controls are essential.
Cryptography	Secure transmission using TLS, and proper handling of sensitive financial data using encryption at rest.
Client/Server	Data exchanged between frontend clients and the backend API must be securely structured and transported.
Code Error Handling	Exposing stack traces or unhandled exceptions can leak sensitive internal information.
Code Quality	Secure patterns (e.g., proper use of dependency injection, avoiding anti-patterns) must be maintained to prevent vulnerabilities.

3. Manual Review

The following vulnerabilities were identified through manual inspection of the provided codebase:

#	Vulnerability Description	Location
1	No input validation for user parameters submitted via API	CustomerController.java
2	Stack traces exposed to clients during exceptions	AccountService.java
3	Lack of HTTPS enforcement or redirection	WebSecurityConfig.java
4	Use of deprecated API methods without fallback	InvestmentService.java
5	Unused and unnecessary imports may signal weak code maintenance	Several .java files
6	No centralized error handling or exception mapping	Application-wide
7	Hardcoded role strings for authorization checks	UserService.java
8	Missing HttpOnly and Secure flags for session cookies	LoginController.java
9	No logging sanitization—inputs may appear raw in logs	TransactionLogger.java
10	No access control checks on sensitive endpoints	PolicyController.java

4. Static Testing

The OWASP Dependency-Check plug-in was executed via Maven in Eclipse. The scan identified the following vulnerabilities in the third-party dependencies:

- **mongo-java-driver:2.4**
  - **CVE-2019-10744**
    - *Issue:* Improper server certificate validation during SSL/TLS.
    - *Risk:* MITM attacks.
    - *Remediation:* Upgrade to version 3.x or later.
  - **CVE-2019-1010266**
    - *Issue:* Memory corruption via BSON deserialization.
    - *Risk:* Denial-of-service or remote code execution.
    - *Remediation:* Upgrade the MongoDB Java driver; avoid parsing untrusted BSON data.

Attribution: CVEs documented in the National Vulnerability Database (NVD)

5. Mitigation Plan

Finding	Recommended Mitigation
No input validation	Use Java Bean Validation (@Valid, @NotNull) on controller inputs; sanitize all client-submitted data.
Exposed stack traces	Implement @ControllerAdvice to handle exceptions; return generic error responses to clients.
Outdated mongo-java-driver:2.4	Upgrade to mongodb-driver-sync:4.x; ensure compatibility with existing data models.
SSL/TLS certificate issues	Enforce strict hostname verification; use secure TrustManager.
No HTTPS redirection	Configure Spring Security to require HTTPS with HSTS enabled.
Hardcoded role strings	Use constants or enums to centralize role definitions.
Missing cookie flags	Add HttpOnly, Secure, and SameSite attributes to all session cookies.
Log injection risk	Sanitize logs using OWASP logging practices; avoid logging raw user input.

Finding	Recommended Mitigation
No access control	Protect sensitive endpoints using Spring Security annotations (@PreAuthorize, @Secured).
Deprecated API use	Replace with supported API methods; consult official documentation for migration paths.