CS 305 Project One Template

Document Revision History

Version	Date	Author	Comments
1.0	9/21/2025	Kevin Lenihan	

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

Kevin Lenihan

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?

Artemis Financial Services needs to protect customer information while modernizing its technology. They handle sensitive data like account numbers and investments, which must be safe from hackers. The company wants to use cloud services and modern tools, but without creating security risks. Their system must work well for both advisors and clients, including international customers.

All financial data must be encrypted when sent online using strong protocols like HTTPS with TLS. Without this, hackers could steal information in transit. Artemis must also follow data protection laws, such as GDPR for European customers and the Gramm-Leach-Bliley Act for U.S. customers.

The company faces many cyber threats. Hackers may inject malicious code, steal login credentials, or exploit weak passwords. Automated tools constantly search for vulnerabilities. Risks also come from insecure third-party software, employee mistakes, and poorly secured APIs that expose too much data.

To address these issues, Artemis needs strict security measures as they update their systems. This includes reviewing all third-party software for vulnerabilities and securing APIs with authentication, usage limits, and data protection. They must also safeguard passwords and secrets in containerized and automated systems. Finally, all new software should undergo thorough security testing throughout development to catch problems before going live.

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.

Based on the vulnerability assessment framework, several critical security areas need attention for this financial web application. Authentication and session management are the foundation of security, as they protect user accounts and administrative functions from unauthorized access. Access control and authorization also play a key role, ensuring that users only see and use what they are supposed to. This prevents situations where regular users might gain admin privileges or access another person's account.

Input validation and injection protection are also essential, since hackers often try to insert malicious code through forms, search boxes, or API calls. Without strong safeguards, attackers could launch SQL injection attacks that expose the entire database or use other forms of code injection to compromise the server. Data protection ensures that sensitive information is kept safe, both when stored on servers and when transmitted over the internet. This requires proper encryption and secure key management to prevent data theft.

Another important area is dependency and third-party management. External software libraries and components can introduce vulnerabilities if not carefully monitored, so they must be scanned regularly and updated when flaws are found. Error handling and logging are equally important, since systems should not reveal sensitive information in error messages. At the same time, proper logging must be maintained to support monitoring and security investigations.

Configuration and deployment security focuses on setting up systems safely from the beginning. This includes using secure default settings and ensuring passwords or other secrets are handled properly during deployment. While physical security and business continuity planning are important overall, the focus here is on protecting the code and technical systems themselves. These areas are especially critical because they directly safeguard financial data and address the most common weaknesses found in REST APIs that financial services often rely on.

3. Manual Review

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

1. CRUD.java
    a. No validation – low severity
        i. Sanitize input/output
2. CRUDController.java
    a. Unvalidated parameters, no authorizations – medium severity
        i. Validate parameters, secure end point
3. Customer.java

    a. No encapsulation, invalid deposits – medium severity
        i. Validate input, follow conventions

4. DocData.java
    a. Hard coded creds, SQL injection risk, no TLS – high severity
        i. Use env vars, SSL, secure logging
5. Greeting.java
    a. Potential XXS – low severity
        i. Sanitize output
6. GreetingController.java
    a. XXS via name, no validation – medium severity
        i. Encode user input
7. MyDateTime.java
    a. No validation, poor encapsulation – low severity
        i. Add validation, fix encapsulation
8. RestServiceApplication.java
    a. Risk of exposed default endpoints – medium severity
        i. Secure actuator, configure spring security

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

[Include your findings here.]

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.

After reviewing the application code, several security issues were identified that require immediate attention. The plan to fix these problems focuses on three main areas: protecting sensitive data, ensuring that user input is handled safely, and configuring the system correctly to prevent accidental exposure of private information.

The most urgent fixes involve adding proper checks to all user input and making sure any data displayed back to users is safely formatted. This will prevent attackers from injecting malicious code or scripts that could steal information or compromise user accounts. Hardcoded passwords and secret keys found in the code must also be removed and replaced with a secure system for storing secrets outside of the main application.

Database queries need to be rewritten using parameterized statements, a secure coding practice that protects against SQL injection attacks. All communication between users and the server should be encrypted with HTTPS/TLS to keep data secure in transit. The Spring Security framework must also be properly configured to protect API endpoints, and any diagnostic endpoints that could reveal sensitive system details should be disabled.

The application requires a systematic approach to managing third-party software. External libraries and dependencies should be regularly updated and scanned for known vulnerabilities. Although automated security scanning could not be run in the current environment, it is assumed that some outdated or insecure components are present and must be addressed.

To prevent similar problems in the future, the development team should integrate OWASP Dependency-Check into the build process. This tool automatically identifies insecure dependencies during both development and deployment, allowing the team to resolve issues before they reach production systems.