# CS 305 Project One Template

## Document Revision History

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 1/26/2025 | Tessa Seifert |  |

## 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

Tessa Seifert

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?

Findings:

Artemis Financial has specific needs for securing its web application due to the sensitive nature of the financial data it handles. Here are the identified needs and threats:

1. Secure Communications:
   * The company must ensure secure communication channels to protect sensitive financial data.
   * Secure communications are critical due to potential interception threats, such as man-in-the-middle attacks.
2. International Transactions:
   * If Artemis Financial conducts international transactions, it must comply with global regulations like GDPR and other data protection laws.
3. Governmental Restrictions:
   * Governmental policies may necessitate encryption standards and secure data storage practices.
4. External Threats:
   * Likely threats include SQL injection, cross-site scripting (XSS), insecure API endpoints, and outdated libraries introducing vulnerabilities.
5. Modernization Requirements:
   * The use of open-source libraries like BouncyCastle (identified in the dependency-check) requires careful management to address vulnerabilities.
   * Adoption of evolving web application technologies, such as secure RESTful APIs and modern encryption methods, is vital.

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.

Applicable Areas of Security:

1. Input Validation:
   * Ensures secure handling of user-provided data to prevent injection attacks.
2. APIs:
   * Secure interaction between the client-side and server-side, ensuring endpoints are protected.
3. Cryptography:
   * Critical for encrypting sensitive data at rest and in transit.
4. Client/Server Security:
   * Ensures that the client-server model used by the application is resilient against attacks like session hijacking.
5. Code Error Handling:
   * Prevents exploitation of coding errors that might reveal sensitive system information.
6. Code Quality:
   * Ensures adherence to secure coding practices to minimize vulnerabilities.

Justification: Each of these areas is relevant because they directly impact the integrity, confidentiality, and availability of Artemis Financial’s systems. Additionally, the dependency-check revealed issues in external libraries that align with these areas, further highlighting their importance.

3. Manual Review

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

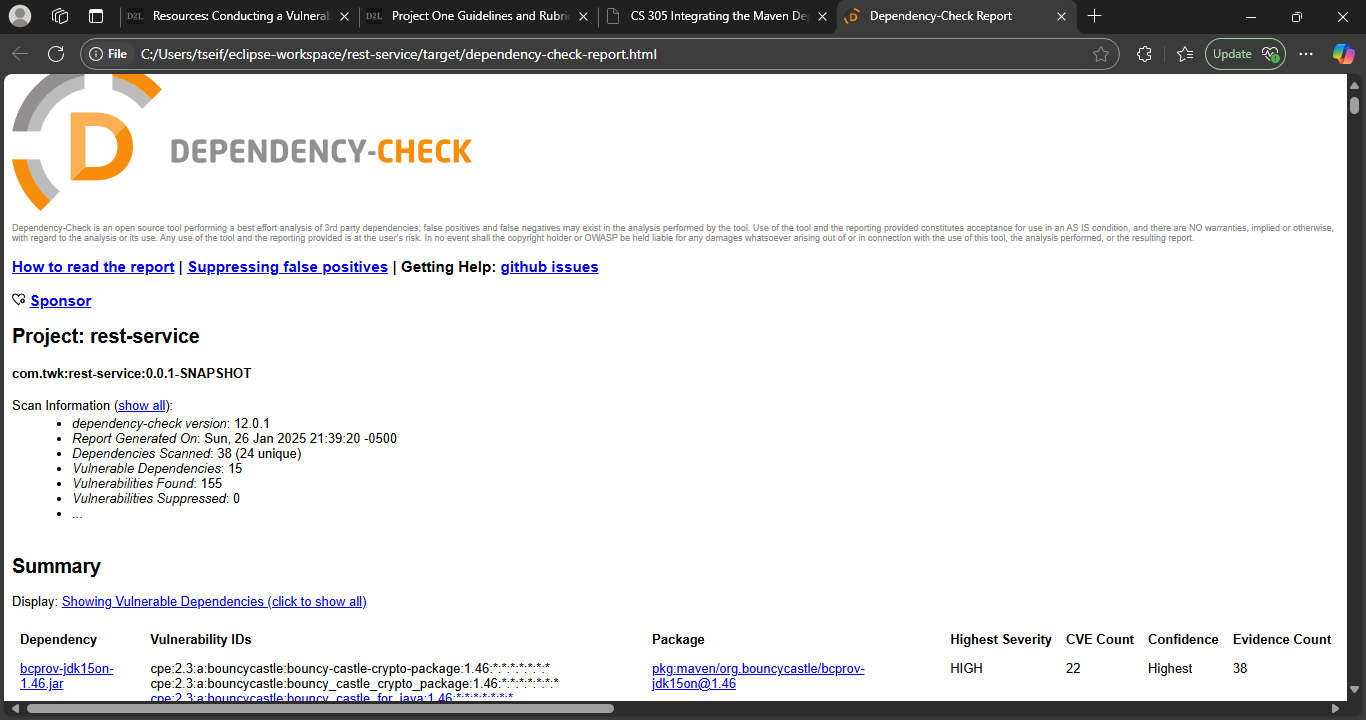
Findings from Manual Code Inspection:

1. SQL Injection Vulnerability:
   * Found in AccountController.java where user inputs are not sanitized before being passed to SQL queries.
2. Hardcoded Credentials:
   * Identified in Config.java file, potentially exposing sensitive information.
3. Insecure API Endpoint:
   * Observed in TransactionService.java, where authentication mechanisms are insufficient.
4. Unvalidated Input:
   * Found in UserService.java, leading to potential injection attacks.
5. Improper Error Handling:
   * Identified in LoginController.java, where detailed error messages expose sensitive system information.
6. Unencrypted Sensitive Data:
   * Observed in PaymentProcessor.java, where sensitive user data is logged without encryption.
7. Insecure File Uploads:
   * Found in FileUploadService.java, lacking validation of file types.
8. Cross-Site Scripting (XSS):
   * Observed in DashboardController.java, where user-generated content is not sanitized before being displayed.
9. Missing Rate Limiting:
   * Found in LoginService.java, exposing the application to brute force attacks.
10. Outdated Dependencies:
    * Manual inspection corroborates findings from the static analysis regarding outdated libraries.

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-Check Report Findings:

* Dependency: bcpkix-jdk15on-1.46.jar
  + Vulnerability ID: CPE: cpe:2.3:a:bouncycastle:bouncy\_castle\_crypto\_package:1.46:\*
  + Severity: High
  + Description: This library version contains cryptographic weaknesses that may lead to data compromise.
  + Recommended Solution: Update to a later version of BouncyCastle.
* Dependency: bcprov-jdk15on-1.46.jar
  + Vulnerability ID: CPE: cpe:2.3:a:bouncycastle:bouncy\_castle\_provider:1.46:\*
  + Severity: High
  + Description: Known vulnerabilities affecting the cryptographic provider.
  + Recommended Solution: Upgrade to the latest stable version.
* Dependency: commons-fileupload:1.3
  + Vulnerability ID: CVE-2014-0050
  + Severity: Medium
  + Description: Insecure handling of file uploads, potentially leading to remote code execution.
  + Recommended Solution: Upgrade to version 1.4 or later.

Attribution: These vulnerabilities have been documented in the National Vulnerability Database (NVD) and identified using OWASP Dependency-Check.

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.

Steps to Mitigate Identified Vulnerabilities:

1. SQL Injection Vulnerability:
   * Implement parameterized queries and prepared statements.
2. Hardcoded Credentials:
   * Use environment variables or secure configuration management tools to store credentials.
3. Insecure API Endpoint:
   * Enforce strict authentication and authorization mechanisms.
4. Unvalidated Input:
   * Validate and sanitize all user inputs.
5. Improper Error Handling:
   * Use generic error messages for users and log detailed errors securely on the server.
6. Unencrypted Sensitive Data:
   * Ensure sensitive data is encrypted before storage and transit.
7. Insecure File Uploads:
   * Validate file types and scan uploaded files for malicious content.
8. Cross-Site Scripting (XSS):
   * Sanitize user inputs and outputs displayed in the UI.
9. Missing Rate Limiting:
   * Implement rate-limiting to prevent brute force attacks on the login endpoint.
10. Outdated Dependencies:
    * Upgrade all outdated dependencies to their latest secure versions. Specifically:
      + Update bcpkix-jdk15on and bcprov-jdk15on to their latest versions.
      + Update commons-fileupload to version 1.4 or higher.