# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **1/30/25** | **Devin Bashaw** | **First Rendition** |

## 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

Devin Bashaw

**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 is a financial consulting company so they handle personal information of their clients. This info can include their financial data like bank accounts, retirement accounts, stocks, etc. It also includes their insurance details, birth dates, social security numbers, full names, etc. A security lapse in communications between client to employee, employee to employee, employee to insurance, employee to bank accounts can occur and put the client or company at risk of breach. Financial fraud, identity theft, and the companies reputation is on the line.
* If the company chooses to operate globally then yes there are international transactions that are involved. Thus the company needs to follow strict protocols and laws in place when performing internationally, such as the General Data Protection regulation of the European Union which is in place to keep user data safe.
* Governmental restrictions on secure communication include: U.S. Compliance with Gramm-Leach-Biley Act, SEC Cybersecurity, and the EU Standard Contractual Clauses. If the company does not adhere to these regulations, substantial fines may be imposed or worse---complete company disbandment and prison time/law suits.
* External threats present now and in the future include the current vulnerabilities within the Dependency Report:
  1. Apache Tomcat Websocket 9.0.30 that is in critical severity for its outdated version that needs upgraded to the latest secured version.
  2. Bouncy Castle Cryptography with high severity for its weakness in encryptions---needs updated to the latest patched version.
  3. Jackson Databind with high severity for its remote code execution in deserialization---needs upgraded to the latest patched version.
  4. Logback Logging Framework with high severity that is susceptible to remote code execution exploitation---needs upgraded to a more secure version, which could be the latest.
  5. Spring Framework MVC and Boot that has the threats of Spring4Shell Exploit and remote code execution by deserialization---Spring Framework and Spring Boot need upgraded to the latest version.

**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
  + If this is treated poorly, SQL Injection, cross-site scripting, and command injection attacks can occur. This can be prevented by implementing server-side validation and filtering.
  + Secure the input with encoding and schema validation to prevent unencrypted responses from leaking.
* Project Setup
  + Server configs, data flow, and API security are essential to keeping the customers information private. Secure API gateways and enforce TLS encryption to prevent data breaches.
* Error Handling
  + Ensure code follows the standards for that language (such as pep8 for Python) to prevent unnecessary bugs and errors. OWASP and NIST, code linting, and try except handling techniques will ensure errors are properly handled for and cleared.
* Secure Authentication
  + Implement OAuth, JWT tokens, and request validation so the software itself is not hacked.

**3. Manual Review**

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

* SQL Injection
  + The code directly embeds database credentials and there is no input validation.
* Authentication
  + No authentication on API endpoints. Anyone can access these endpoints.
* Authorization
  + No authorization on API endpoints. Anyone can access these endpoints.
* Poor Input Handling
  + Input is directly passed between functions without validation.
* Hardcoded Credentials
  + Database credentials are hardcoded, which can allow attackers to extract it from decompiled code.
* No Exception handling or Logging
  + Leaks database errors and system details in logs.
* Poor Access Modifiers
  + Some aspects of the code should be private to enforce controlled access.
* Missing Encapsulation
  + Exposes raw data and violates encapsulation principles.
* Poor API response handling
  + No data masking or filtering here
  + This can leak sensitive information
* Directory Traversal
  + File paths need validated before accessing files
  + Parameterized queries can prevent bad access.

**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
* Apache Tomcat Websocket: CVE-2020-13935
  1. Invalid payload lengths could trigger an infinite loop. Multiple requests with invalid payload lengths could lead to a denial of service.
* Jackson Databind: CVE-2020-25649
  1. A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity.
* Bouncy Castle Cryptography: CVE-2020-15522
  1. Bouncy Castle BC Java before 1.66, BC C# .NET before 1.8.7, BC-FJA before 1.0.1.2, 1.0.2.1, and BC-FNA before 1.0.1.1 have a timing issue within the EC math library that can expose information about the private key when an attacker is able to observe timing information for the generation of multiple deterministic ECDSA signatures.
* Logback Logging Framework: CVE-2021-42550
  1. In logback version 1.2.7 and prior versions, an attacker with the required privileges to edit configurations files could craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers.
* Spring Framework: CVE-2022-22965
  1. Product: VMware Spring Framework
  2. Name: Spring Framework JDK 9+ Remote Code Execution Vulnerability
  3. Date Added: 2022-04-04
  4. Description: Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding.
  5. Required Action: Apply updates per vendor instructions.
     1. A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. The specific exploit requires the application to run on Tomcat as a WAR deployment. If the application is deployed as a Spring Boot executable jar, i.e. the default, it is not vulnerable to the exploit. However, the nature of the vulnerability is more general, and there may be other ways to exploit it.

**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.

* Apache Tomcat Websocket: CVE-2020-13935
  + Upgrade Apache Tomcat to 9.0.37 or later
* Jackson Databind: CVE-2020-25649
  + Upgrade version to 2.12.2 or later
  + Use @JsonIgnoreType to prevent deserialization of risky classes.
* Bouncy Castle Cryptography: CVE-2020-15522
  + Upgrade to version 1.68 or later
  + Use NIST-approved encryption standards
* Logback Logging Framework: CVE-2021-42550
  + Upgrade to version 1.2.9 or later
  + Restrict log message input to prevent command injection.
* Spring Framework: CVE-2022-22965
  + Upgrade to version 2.6.6 or later
  + Restrict classes loaded by setting disallowedFields in DataBinder.
* Other resolutions:
  + Implement JWT-based authentication
  + Use Spring Security
  + Apply Role-Based Access Control
  + Use parameterized queries
  + Remove hardcoded credentials
  + Apply database least privilege access
  + Store credentials in environment variables or Spring Boot config files
  + Use Spring Cloud Config Server for secure key management
  + Implement audit logging
  + Implement rate limiting and API throttling

Dependency report ss:

A screenshot of a computer

Description automatically generated