# CS 305 Module Two Written Assignment Template

## Instructions

Replace the bracketed text with the relevant information in your own words. If you choose to include images or supporting materials, make certain to insert them in all the relevant locations in the document.

## Areas of Security

**Input Validation** – Secure Input and Representations: This area of security is crucial for the web application because it deals with data provided by the user. Any vulnerability provided in code that can be manipulated by a user needs to be validated upon input, as well as sanitized to ensure the data is not compromised to manipulate the program’s intended output.

**API** – As applications implement APIs for functionality, special care must be taken to ensure that proper access is allowed during the API’s construction. APIs can often leave vulnerable points of entry for unauthorized access by attackers through various endpoints.

**Cryptography** - When an application might handle communication methods such as HTTP/HTTPS proper cryptographic methods should be implemented to ensure secure communication since sensitive data is often transmitted across servers and applications.

**Code Error –** Maintaining proper code guidelines can aid in preventing unsecure errors in code. Exception handling in code is an important method of vulnerability assessment as exceptions can provide information regarding an applications detail like the specific framework, language, and version in use. All of which can be accessible by an attacker to be used for information gathering.

**Code Quality** – Ensuring secure Coding Practices/Patterns is essential to secure development. Secure code quality helps keep an application’s internal workings safe from attackers.

## Areas of Security Justification

**Input Validation** – Inadequate input validation can lead to severe security vulnerabilities, thus allowing attackers the chance to inject malicious information, compromising the program’s intended functionality and security. Secure input and representations are a vital component of an applications assessment to increase the defense against attackers.

**API** – Unsecured APIs can lead to unauthorized access across an application, making areas vulnerable such as sensitive features, data, and control. Proper secure API control is a crucial aspect of secure software development.

**Cryptography –** Encrypting data is a top priority of secure development as it ensures that sensitive data and communication cannot be accessed or listed to by unauthorized users. Utilizing proper encryption methods can help in ensuring that data is not intercepted or manipulated.

**Code Error –** Errors in code are inevitable as they allow the programmer to deduce problems in code logic and fix mistakes. However, they can leave an applications information prone to tactics by attackers that take advantage of errors in code. Errors can often lead to vulnerabilities like improper input validation, SQL injection, XSS attacks, all of which can be the result of improper secure code development.

Code Quality – Secure coding practices/patters are essential to secure code development as applications can often be vulnerable through the patterns they use, in which attackers are looking to take advantage of.

## Code Review Summary

**Input Validation –** The code base demonstrates some common vulnerabilities such as not properly implementing input validation in its development. For example, in the “GreetingController” class, the “/greeting” endpoint uses a ‘name’ parameter is vulnerable to SQL injection attacks and data manipulation.

**API –** The application does not implement proper authentication and authorization methods. It currently allows unrestricted access to the ‘/greeting’ and ‘/number’ endpoints in the “GreetingController” class. This can lead to unauthorized access by attackers, inevitably leading to data breaches and stolen information.

**Secure Coding** – There is an issue in the “Greeting number” function with the parameter of ‘id’. This function does not properly validate the ‘id’ parameter when accessing the array. This can potentially lead to an out of bounds issue and possibly cause a runtime error or application crash.

**Dependencies** – The code utilizes the Spring Framework version 2.6.5, which is an outdated version of the Sprint Boot Framework. Proper dependencies should be implemented to ensure that the software versions are up to date as vulnerabilities may be present and known within threat actor communities. S

## Mitigation Plan

**Input Validation –** Implement proper input validation techniques and data sanitization that ensures that user-input data is safe and does not contain any malicious information.

**API** – Enhancing the applications user authentication techniques and role-based access control that ensure only users with necessary access to the applications endpoints can help in preventing unrestricted access to data.

**Secure Coding** – To ensure that proper that proper secure coding practices are followed and to prevent any errors that may lead to unsecure access or application failures, there should be proper input validation for parameters like the ‘id’ parameter so that an array cannot be accessed out of bounds. I recommend using various “checkIndex, checkFromToIndex, and checkFromIndexSize” methods from the java.util.Objects class (*Secure Coding Guidelines for Java SE*, n.d.). Which can help in preventing integer overflows.

**Dependencies –** Based on the published report of the static dependency check any Sprint Boot versions 3.0.5 and older may be susceptible to security bypass attacks. It’s recommended that the application be updated to the most recent version of either 3.0.6+ or 2.7.11+ to prevent any potential DoS (Denial-of-Service) attacks on the application. (*CVE - CVE-2023-20873*, n.d.)

*CVE - CVE-2023-20873*. (n.d.). https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-20873

Manico, J. (n.d.). *Iron-Clad java*. O’Reilly Online Learning. https://learning.oreilly.com/library/view/iron-clad-java/9780071835886/ch01.html#page\_18

*Secure Coding Guidelines for Java SE*. (n.d.). https://www.oracle.com/java/technologies/javase/seccodeguide.html