# CS 305 Module Two Written Assignment

## Areas of Security

Considering the given scenario and the VAPFD, the relevant areas of security for this software application are:

* Input Validation
* APIs
* Client/Server
* Code Error
* Code Quality
* Encapsulation

## Areas of Security Justification

* Input Validation: Since the application allows users to input data (such as name), it is necessary to ensure that input validation is performed to avoid potential attacks, such as code injection.
* APIs: The application uses Spring Framework, which may involve interaction with various APIs. Ensuring secure API interactions is crucial for overall application security.
* Client/Server: The application is a web application, which inherently involves client-server communication. It is necessary to secure the distributed components to protect against potential attacks.
* Code Error: The application may contain coding errors that could lead to security vulnerabilities. By reviewing the code for errors and implementing secure code handling practices, the application will be more secure.
* Code Quality: Ensuring the use of secure coding practices and patterns will help prevent security vulnerabilities in the application.
* Encapsulation: The application should implement proper encapsulation techniques to secure data structures and prevent unauthorized access or modification.

## Code Review Summary

Upon manually inspecting the provided code, the following vulnerabilities have been identified:

Potential Vulnerabilities:

* In **GreetingController.java**, the use of Spring Expression Language (SpEL) with user-provided input (name) can lead to Remote Code Execution (RCE) if the input is not validated properly.
* In the **/number/{id}** endpoint, there is no proper input validation or error handling for the **id** path variable, which can lead to an ArrayIndexOutOfBoundsException.

## Mitigation Plan

* Input Validation: Apply input validation techniques to user-provided input (name). For example, use a whitelist of allowed characters or an input validation library like Apache Commons Validator to ensure that only valid input is processed.
* Spring Expression Language (SpEL) Mitigation: Upgrade the Spring Framework version to the latest version to address potential security vulnerabilities. Additionally, disable SpEL evaluation for user-provided input or use an alternative approach to achieve the desired functionality without introducing security risks.
* Error Handling: Implement proper error handling for the **/number/{id}** endpoint to avoid revealing sensitive information through exceptions. For example, catch the ArrayIndexOutOfBoundsException and return a user-friendly error message.
* Code Quality: Ensure that the development team follows secure coding practices and guidelines, such as the OWASP Top Ten Project, to minimize security vulnerabilities in the application.
* Encapsulation: Review the application's data structures and ensure that they are properly encapsulated to prevent unauthorized access or modification.