# CS 305 Module Two Code Review and Mitigation Plan Assignment

**Brandon Hobbs**

**CS-305**

**May 11, 2022**

## Areas of Security

The code in question is a Spring-based API. It consists of 2 methods (number and greeting) that are both a Greeting-type. The API attempts to collect user input and then return a value from an array or a personalized greeting string.

With this structure I would focus on Input Validation, Secure API Interactions, Code Errors, and Encapsulation.

Input Validation is important as you need to “sanitize” user inputs, even if from a trusted user. A malformed input, on purpose or not, could cause errant behavior. Securing the API goes one step farther in ensuring the connection (and possibly the user) is trustworthy. There is some code checking built into Spring but this error handling needs to be generic and not provide any signature that a hacker could use to understand more about the code structure. And finally, because we are using methods, the methods should use encapsulation to protect the sensitive data from unwanted changes, i.e., set variables to private and use Get/Set functions.

## Code Review Summary

Methods have Get/Set functions available in the Greeting class and parameters are set to private. GreetingController, however, does not use the get/set functions.

User input to the API is not sanitized. For example, within the number method the value from the array is passed via string splices: *String message = "Element in the given index is :: "+myArray[id]. id* is the raw variable obtained from the user. This could lead to injection.

This method also has no validation. The referenced array is static and of length 7. No validation on the input is seen. If knowing the length of this array was valuable to a hacker the error message would also need to be sanitized. For example, if someone attempting to obtain position 7 should not receive a message stating, “array out of bounds”. One step further would be to make sure that the return length of the number method is only a single value and not multiple.

The greeting method’s input also needs to be sanitized and checked for length, i.e., to prevent buffer overruns.

The pom.xml also shows an outdated Spring framework being invoked and an older version of the Java JDK. These should be updated.

## Mitigation Plan

To mitigate any of the potential security flaws focus should be given to the user inputs. These need to be sanitized and/or validated.

The Greeting class get/set methods could be used to improve the encapsulation.

Error sanitization also needs to be employed to prevent leaking any extra information about the program, i.e., length of *myArray*.

Updates to the Spring controllers and pom should also be considered. If there are any vulnerabilities found in a dependency scan those could be addressed at the same time.