# CS 305 Module Two Code Review and Mitigation Plan Assignment

## Instructions

Replace the bracketed text with your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

**Role**: Senior software developer

**Application:** A complex web application using the Spring framework.

**Task**: Implement an expressive command input function for the application

**Framework**: Version 2.6.5 of the spring-data-rest-webmvc in the Spring framework

**Language**: Spring Expression Language, Java

***Preface*:** Our growing web application uses Spring Initializr with Spring Boot frameworks for database access. API access to commands from our input function shall also be implemented, and we may require a REST API service with Spring Boot

## Areas of Security

* Input Validation / Secure Input and Representations
  + As we are implementing an expressive command input function, we require validation on all input for the function. Particularly, verify the number of SQL results using JDBC4ResultSet class’s getUpdateCount method, and incorporate Type safety by forcing output into a formal data type when using SELECT. In addition, use query parameterization for all queries and variables, escaping variables, and input validation to prevent SQL injection attacks.
* APIs / Secure API interactions
  + We will build a RESTful service using Spring Boot and Initializr which will allow certain input functions to be accessible outside of our system. This will require API-level Input validation, as well as limiting database connection access (Principle of Least Privilege).
* Cryptography / Encryption Use and Vulnerabilities
  + Cryptography and hashes should be used on sensitive user data; in addition, secure coding practices such as using Stored Procedures and query parameterization for form authentication in place of mixing query language fragments with untrusted data.
* Client / Server / Secure Distributed Composing
  + Ensure HTTPS (Hypertext Transfer Protocol Secure) encryption is used between client and server during transfer requests. Ensure the certificate used for HTTP is valid, and the information in the certificate is correct to prevent man-in-the-middle attacks (or use Let’s Encrypt). In addition, use only POST requests to transmit authentication credentials as per OWASP standards.
* Code Error / Secure Error Handling
  + Use Junit tests, boundary value analysis, and input validation on all command input functions, and all API-level functions, aiming for high test coverage.
* Encapsulation / Secure Data Structures
  + Always use good Object oriented principles including Principle of Least Privilege and Encapsulation when programming data structures. Only allow the necessary data to be accessed by the Spring command input functions with use of private instance data and public methods.

## Code Review Summary

* Upon running the software, noted Spring version v2.2.4 was in use rather than the newest 2.4.3. Consider updating to mitigate any security vulnerabilities.
* Our mongo-java-driver is out of date at version 2.4, the current version is 3.12.8. Consider updating to mitigate vulnerabilities.
* Input validation should be used when using requesting and parsing our name parameter, within file GreetingController. The expression can be formatted or tampered with to cause abnormal functionality.

## Mitigation Plan

* We will update our Spring version to 2.4.3 along with any dependencies.
* We will update our mongo-java-driver to the latest version, 3.12.8.
* Add input validation to our GreetingController when parsing unknown inputs, such as length and size validation as well as type safety. In accordance with REST security, we should reject unexpected/illegal content, make use of any sanitation/validation libraries in Spring, log input validation failures, and define appropriate request size limit so that we can reject requests exceeding the limit with correct HTTP response.