# CS 305 Module Two Written Assignment Template

Lukas Singer

Southern New Hampshire University

CS-305 Software Security

Dr. Albanie Bolton

July 13, 2025

## 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

The four most pertinent areas of security to keep an eye on for this project will be APIs, Input Validation, Code Quality, and Code Error. These areas were selected due to the application’s architecture, which relies on automatic REST endpoint exposure, user-supplied command input, and Spring Expression Language (VMware). By focusing on these areas of security, we can minimize risks and improve how user interactions are handled, how inputs are processed, how the system responds to failures, and how the overall codebase maintains secure behavior.

## Areas of Security Justification

* APIs
  + API security is an important area as the program’s use of “spring-data-rest-webmvc” means that RESTful endpoints are vulnerable. This may reveal or permit sensitive operations, the kind we are trying to avoid. These endpoints must be reviewed and secured to prevent unauthorized access, especially when tied to dynamic or user-driven functionality (VMware). API security directly impacts the safety and integrity of application data.
* Input Validation
  + Input validation is critical because the application allows user-supplied command input, which may be parsed and executed using Spring Expression Language. Without strict validation, attackers could inject malicious expressions or manipulate code at runtime. By utilizing input validation only, safe, expected input, is processed which helps prevent injection vulnerabilities and unauthorized code execution.
* Code Quality
  + Due to the complexity of the code using features like Spring, we must ensure that code quality is maintained diligently. Sloppy or inconsistent coding will often hide bugs and/or security issues. By writing good code we can prevent future issues and any that do arise can be snuffed out.
* Code Error
  + Improper error handling can give insight to potential attackers such as system attributes, server names, or other sensitive internal information. Handling errors the right way will keep the application safe and prevent leaks and security vulnerabilities.

## Code Review Summary

The most glaring issues come from the outdated nature of the code. First, by looking at the pom.xml file I can gather that the program is using Spring Boot Framework 2.2.4.RELEASE. This version is extremely out of date and contains many known vulnerabilities. We can patch up a bunch of vulnerabilities by simply updating this. Also, our version of MongoDB Java Driver is set to 2.4 out of date by several years as well. Again, by updating this we can patch up quite a few vulnerabilities and issues with the code. The third and maybe most vulnerable aspect of the code currently would be that in GreetingController.java our user input is directly executed without any form of input validation. This in turn creates an opportunity for attackers to inject malicious code. Combined, these three issues make the application highly vulnerable to an attack and need to be addressed quickly.

## Mitigation Plan

Our mitigation plan will be quite simple, first we will address the outdated versions of Spring Boot and MongoDB Java Driver. Based on my research it appears the latest stable version of Spring Boot is 3.5.x. However, if there are any issues with transitioning away from a 2.x.x version then the best available option would be 2.7.x. These versions of Spring Boot will have enterprise support until 2032-06-30 and 2029-06-30 respectively (VMware). As for MongoDB Java Driver, the most recent stable version appears to be 5.5.1 (MVNRepository). If we have issues for some reason with this version, we can slowly walk our versions back until we have a working application again. Updating to these newer versions will immediately resolve many known security vulnerabilities, improve overall performance, and ensure better compatibility with modern tools and libraries. It also puts us on supported versions with long-term maintenance, reducing future risks and making future updates easier to manage. After updating, we should verify that the applications still work as expected to. Finally, the direct execution of user input using Spring needs to be removed or controlled. We should start by avoiding evaluating raw user input and instead sanitize or whitelist allowed inputs to prevent any potential injection attacks. Fixing these three areas will greatly reduce our exposure to common exploits and bring the application up to a much safer baseline.

References

MVNRepository. (n.d.). MongoDB Driver*. MongoDB.*

<https://mvnrepository.com/artifact/org.mongodb/mongodb-driver-sync>.

VMware. (n.d.). Spring Boot – Support. *VMware.*

<https://spring.io/projects/spring-boot#support>.

VMware. (n.d.). Building REST services with Spring. *VMware.*

<https://spring.io/guides/tutorials/rest>.