### Southern New Hampshire University

2-1 Written Assignment: Code Review and Mitigation Plan

David Vega

### Dr. Vivian Lyon

### CS-305 – Software Security

01/15/2023

# CS 305 Module Two Written Assignment Template

## Areas of Security

The application code-base shows that the application is utilizing Spring MVC version 2.6.5 in the Spring framework and SpEL (“Spring Expression Language”). The areas of security that I feel are relevant to this application are Input Validation, APIs, and Code Error.

## Areas of Security Justification

Input validation is crucial because you want to ensure that a client cannot pass in nefarious code. Since the main goal of this project’s sprint is to implement an expressive command input function, having appropriate checks in place is a good way to protect your application from potentially devastating bugs. Such constraints can be used to check if the input allows numbers or restrict special characters, words, or perhaps remain a certain length. Securing APIs is important as with requesting data via HTTP can be intercepted by a third-party. Hackers can and will find a way to exploit the code structure through an applications URL. In addition, the code base should also have adequate error handling and return this to the user.

## Code Review Summary

In reviewing the provided code base, I can see that the Greetings.java file defines the class Greeting but I don’t see where the method is ever called. The Application.java file is the entry point into the application and appears to hard code the message “Hello World” into the parser. There is also no error validation being done in any of the files. As you can see when hitting the endpoint “/greeting” there is a built-in fallback but nothing that is explicit to the actual error. I also modified the default value in the GreetingsController.java file for highlighting that the default value was provided to the client just by going to the endpoint. If this piece of data proved to be valuable to a hacker, he can exploit this.

Graphical user interface, text, application, email

Description automatically generated

The “/number/{id}” endpoint also reveals information about the static array (See Below).

Graphical user interface, website

Description automatically generated

Graphical user interface, text

Description automatically generated

As you can see from the figures above that the “id” and the “element” of the array was exposed by going to the endpoint “/number” and entering in an arbitrary number that represents the index of the array.

The below figure is a fallback error when the input is “7” because the array is out of bounds. The index starts at 0 and ends at 6, therefore, determining the array’s length.

A picture containing text

Description automatically generated

In the pom.xml file, I noticed the version of Spring being implemented is 2.6.5. There could be an issue with making this application RESTful as there was no support for REST prior to version 3.0 (Debnath, 2018).

## Mitigation Plan

To mitigate some of these vulnerabilities, I would focus on input validation. Validating input ensures that no client can introduce potentially harmful code. Depending on the complexity of the application, I would also revisit access controls and whether they should be role-based or attribute-based. For smaller applications, it might make sense to implement a role-based access control so that only users with those assigned roles can access the requested page(s) (Jim Manico, 2014). Finally, I would run a dependency check and see if there might be any dependencies that are out-of-date (Spring Boot Maven Plugin, 2023). As mentioned earlier, upgrading Spring to 3.0 might provide some additional features and support such as REST architectural support.

# References

*Boot Features Developing Web Applications*. (2023, January 15). Retrieved from Spring: https://docs.spring.io/spring-boot/docs/2.2.4.RELEASE/reference/htmlsingle/#boot-features-developing-web-applications

Debnath, M. (2018, February 26). Exploring REST APIs with Spring MVC.

Jim Manico, A. D. (2014). Iron-Clad Java: Building Secure Web Applications.

*Maven Documentation*. (2023, January 15). Retrieved from Apache Maven Project: https://maven.apache.org/guides/index.html

*Spring Boot Maven Plugin*. (2023, January 15). Retrieved from Spring: https://docs.spring.io/spring-boot/docs/2.2.4.RELEASE/maven-plugin/