|  |  |
| --- | --- |
|  |  |

CS 305 Module Two Code Review and Mitigation Plan Assignment  
Brandon Coulter  
Southern New Hampshire University

# CS 305 Module Two Code Review and Mitigation Plan Assignment

## Areas of Security

* After review of the scenario, based on what I know about web application architecture, the areas of concern relevant to the vulnerability assessment process flow diagram are as follows:
  + Input Validation
  + APIs
  + Cryptography
  + Client/Server
  + Code Quality
* Given the nature of the task, to create an expressive command input function for a complex web application using the spring framework, it will be important to have proper input validation that will ensure user inputs are secure and valid inputs. Additionally, in order to have outside interactions with the application, a RESTFul API is used, thus ensuring the API is secure and properly implemented will be vital to the security of the application. A web application utilizes HTTP protocol and thus secure certificates are required to ensure that data is secure from client to server. As described in *Iron-Clad Java Building Secure Web Applications*, “HTTPS, or Hypertext Transport Protocol Secure, provides a variety of different encryption services to protect user data during transport, such as protecting data while it’s traveling over a wired or wireless network.” (Manico et al., 2015) Because of this, suitable cryptography and client/server configuration are a must in any web application that is intended for the public. Additionally for web applications, in general they will utilize some sort of database and thus another client/server vulnerability is SQL injection. It is important to ensure proper parameterized queries to mitigate those attacks. (Manico et al., 2015) Lastly, the quality of the code, i.e., checking for best coding practices and secure coding practices should be completed to make sure of a quality application with security in mind.

## Code Review Summary

* Code review has led to a few concerning vulnerabilities, of which API is one of the larger concerns. This is due to the team utilizing an outdated version of “spring-data-rest-webmvc”. The code base indicates that 2.6.5 RELEASE is the version currently in use, which is outdated and could lead to potential vulnerabilities that have already been fixed in the newer release versions. Additionally, “spring-boot-starter-parent” is an older version as well. Also of concern is the RESTFul API and whether it will be using the HTTPS protocol POST or not. Utilizing POST will provide a means of securing sensitive data that a GET request would not sense a GET request may leak sensitive information in the URL. A POST request puts vulnerable data into the body of the request; thus, it is secure from various possible leaks. (Manico et al., 2015) Again from *Iron-Clad Java Building Secure Web Applications*, the importance of POST requests is demonstrated in this quote, “Two very important secure coding rules: (1) do not trust anything from the request and (2) only submit sensitive data over HTTPS POST in the body of the POST.” (Manico et al., 2015) Furthermore, this project is using an older version of Java at Java version 1.8. This could lead to vulnerabilities that were patched or fixed in more up to date versions of Java. There seems to be a lack of input validation and thus invalid inputs from external sources could expose the application to injection vulnerabilities and denial of service (DOS) attacks.

## Mitigation Plan

* Of the identified security vulnerabilities, the largest of concerns is the outdated dependencies in the “spring-data-rest-webmvc” found in pom.xml line 42 and spring boot found in pom.xml line 8 both of which are multiple versions behind. Mitigation for this is to update all dependencies to their most current release versions. Given the version of Java is also outdated found in pom.xml line 18; this allows for attacks on susceptibilities within older versions of Java. Again, to mitigate this, updating the Java environment the team is working in would be of best practice. Since the nature of this portion of the application is user inputs, input validation is extremely important to ensure untrusted inputs are not allowed to cause problems with the application. “Defining a wrapper method facilitates installing appropriate security manager checks, validating arguments passed to native code, validating return values, defensively copying mutable inputs, and sanitizing untrusted data. Consequently, every native method must be private and must be invoked only by a wrapper method.” (Mohindra & Rozenau, 2021) As just described by the Carnegie Mellon University’s *SEI CERT Oracle Coding Standard for Java*, a method wrapper is vital for validation of inputs from untrusted sources, thus the mitigation for our lack of input validation would be to define wrappers for methods of the application receiving input i.e., in GreetingControl.java lines 24-30 and 33-38. Theses wrappers would have to validate any and all input to mitigate against DOS attacks. In order to mitigate cryptography vulnerabilities, this application will need to implement HTTPS protocol and use certificates and POST requests.

**References**

Manico, J., Detlefsen, A., & Kenan, K. (2015). *Iron-clad Java*. O'Reilly Online Learning. Retrieved July 6, 2022, from <https://learning.oreilly.com/library/view/iron-clad-java/9780071835886/ch01.html#ch01lev2sec3>

Mohindra, D., & Rozenau, M. (2021, May 18). *JNI00-J. Define wrappers around native methods*. JNI00-J. Define wrappers around native methods - SEI CERT Oracle Coding Standard for Java - Confluence. Retrieved July 8, 2022, from https://wiki.sei.cmu.edu/confluence/display/java/JNI00-J.+Define+wrappers+around+native+methods#:~:text=Defining%20a%20wrapper%20method%20facilitates,only%20by%20a%20wrapper%20method.