# CS 305 Module Two Written Assignment Template

## 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

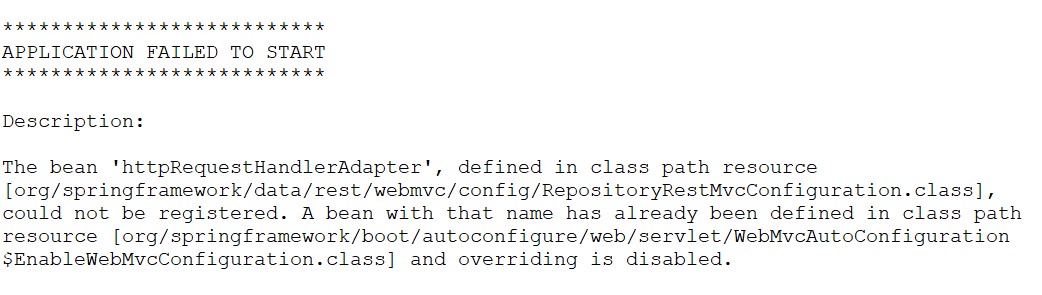
Multiple vectors are of concern with our Spring application, starting with user input validation. As we will be utilizing “expressive” command input functionality for our MVC, it is imperative that we validate user input to ensure input errors don’t cause system instability. If we decide to use a web API for querying data, we need to ensure it performs only the actions it needs to, and does it securely. Cryptographic requirements need to be met for transmitting over https to our end users, as well as verification and protection of sensitive/user data. Our MVC will operate under client and server functionality, and we need to ensure that all our individual providers within our MVC meet a minimum security standard of which together they don’t enhance vulnerability probability. The data structures that will be queried by the Spring Expression Language will need to be encapsulated to prevent unintended access or modification.

## Areas of Security Justification

User input can contain massive security risks for applications. If we intend to have robust user input functionality, we need to ensure that we either restrict what data can be entered, or construct robust error handling for edge cases that could compromise the integrity of our system. We are at risk for code injection vulnerabilities by opening this capability up. A web API can be utilized to return data when a UI isn’t required, and we might want to consider it for faster execution utilizing the Spring Expression Language. Ensuring the API only performs the actions required protects our data structures, and prevents bad actors from using our API to access or modify data outside of the boundaries we set. Cryptography is important for modern web applications – at a minimum we need to issue certificates to our application providers to enable HTTPS communications. Encrypting data-in-transit using HTTPS also helps to secure user data, and prevent attackers from easily stealing user interaction data. To protect a client and server application like we have, we need to create a minimum security standard for each individual provider or client. If an interaction occurs between an aspect of our client and server architecture that creates gaps or vulnerabilities in our system, then it comprises the integrity of the entire system itself. Finally, as we are utilizing the Spring Expression Language (SEL), any objects or datasets that will be queried by SEL need to be encapsulated correctly to prevent unintended data modification, or data exfiltration. Our objects need to utilize private and protected attributes whenever possible, and data endpoints need to have security measures in place to hold data only where it needs to be.

## Code Review Summary

On run time, an error is generated which prevents the application from starting:



The error specifies that 'httpRequestHandlerAdapter', the ‘bean’ or object that manages http requests, is defined in two separate class paths sources. In addition, the version specified for our spring-data-rest-webmvc is at 2.6.5. If we go to the Maven repository, ([spring-data-rest-webmvc/2.6.5.RELEASE](https://mvnrepository.com/artifact/org.springframework.data/spring-data-rest-webmvc/2.6.5.RELEASE)), we can see there’s 58 vulnerabilities associated with this version of the webmvc. By not utilizing an updated version of this framework, we’re opening up our application to a variety of attack vectors.

## Mitigation Plan

To mitigate the first noticed issue, we’ll need to ensure that we are deliberate with our object definitions and links. Having conflicting definitions causes issues like above where the service can fail to start, but it also could create unintended consequences where we provide the wrong class definition for a different dependency in our class paths. To mitigate the larger issue, which entails the outdated framework, I will work with the team in testing out one of the latest version of the MVC, either 6.1.0 or above. After rigorous testing utilizing this version with our current infrastructure, we’ll need implement the change to use the more secure MVC, and rewrite any methods to be compatible with the new package implementations.

**Reference**

1. spring-data-rest-webmvc/2.6.5.RELEASE:

https://mvnrepository.com/artifact/org.springframework.data/spring-data-rest-webmvc/2.6.5.RELEASE