**CS 305 Module Two Code Review and Mitigation Plan Assignment**

**Stephen Cardone**

**Project**

* "A complex web application that uses the Sprint framework: implementing an expressive command input function for the application"
* ( It was unclear if we would be implementing API access to our command input function or not. I've written the next section assuming we might implement API access. )

**Areas of Security**

* Input Validation: Since we are implementing a command input function, it will be important for us to strictly validate all input in to the function. Successfully validating input will promote a secure system and prevent any injection attacks.
* APIs: If our command input function needs to be accessible outside of our system, we may need to implement a RESTful api to allow access to the functionality. In this case, we will need to ensure that the api is created in a way that does not allow unexpected access to our systems. We could also do some input validation at the api level.
* Cryptography: In the case we are going to use an API, we should ensure proper cryptography is used to encrypt and protect our data and customer data.
* Client/Server: If we are to implement API access to our function, we need to ensure the proper certificates are used to ensure data is safe during transfer via https requests.
* Code Error: We will need to code review all command input functions as well as any api access layer code.
* Encapsulation: If our command input function needs to access data stored in our system, we need o ensure we are using it properly are not messing up our data structures.

**Code Review Summary**

* spring-data-rest-webmvc version: In our pom.xml, we have a dependency on sprint data rest webmvc. The version we are dependant on is out of date. (2.6.5.RELEASE). The current live version is 3.3.0. We should upgrade to the latest version since the latest version of the dependency could have many security updates since 2.6.5.
* In GreetingController.java line 27-30 we take in data from parser, and put it directly in to a string value that is then added to our template as plain text. This is risky as someone might be able to format a string that can cause a problem in our system down the line. We should at least limit the size (perhaps length of the string) that can be taken in.

**Mitigation Plan**

* We will upgrade our sprint data rest webmvc dependency version to the latest 3.3.0 version.
* We will add input validation on our GreetingController parser input. At the very least we will look for length of data to prevent massive sizes of data to be passed in. We will also look into parsing input to look for any problematic string patterns (or perhaps find a library to do this for us as they exist).