****

# CS 305 Project Two

**Practices for Secure Software Report**

**Santiago Ramirez**

Table of Contents

[Document Revision History 3](#_Toc100887727)

[Client 3](#_Toc100887728)

[Instructions 3](#_Toc100887729)

[Developer 4](#_Toc100887730)

[1. Algorithm Cipher 4](#_Toc100887731)

[2. Certificate Generation 5](#_Toc100887732)

[Cmd certificate: 5](#_Toc100887733)

[6](#_Toc100887734)

[3. Deploy Cipher 7](#_Toc100887735)

[4. Secure Communications 7](#_Toc100887736)

[5. Secondary Testing 7](#_Toc100887737)

[6. Functional Testing 9](#_Toc100887738)

[7. Summary 13](#_Toc100887739)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **04/15/2022** | **Santiago Ramirez** |  |

## Client



## Instructions

Deliver this completed Practices for Secure Software Report documenting your process for writing secure communications and refactoring code that complies with software security testing protocols.

Respond to the steps outlined below and replace the bracketed text with your findings in your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

Santiago Ramirez Jimenez

## 1. Algorithm Cipher

The appropriate encryption algorithm cipher that I would choose for this project will be SHA-256. The reason SHA-256 cipher is the best candidate to tackle Artemis Financial is because it’s a reliable algorithm that is being used by multiple technology leaders, it has an unlikely chance to encounter collision since it has a 2 to the power of 256 to arrive to a collision which is basically a higher number then the number of atoms in the universe and it will maintain a secure hash value even if the original information has a small change it will automatically change the hash value instantly to maintain its integrity (Callaghan,2020). SHA 256 is one of many options that derived from the secure hash algorithm 2 which was created in 2001 by the National Security Agency, Sha-256 is a 256 bit string that is used to secure password and data to be transfer in a proper and secure manner(N-able, 2019).

The hash function as mention before is long and has around 256 bit so the benefits of these factors will allow the hash to be less affected by collision and less likely to be broken, “the longer the message digest, the more difficult it is to collide” (Wang, Liu, Chen & Wang 2021). Additionally, lets explained the difference between symmetric, Asymmetric and random numbers, symmetric is a data encryption technique that will use one single secret cryptographic key that will be use both for encrypting and decrypting while asymmetric uses “linked-public and private keys to encrypt and decrypt sensitive data”(Daniel, 2021). Meanwhile their is the random number which as the name states it generate a random number when encrypting a hash and this is very important factor in encryption because it maintains a form of unpredictability.

The history of cryptography, to make things briefly in the year 600 B.C Spartans use to send messages to each other with the aid of scytale which is a device that will send secret message to Spartans and it will be decrypted by the recipient having the right size rod. Later on, Julius ceaser, Giovan Battista Bellaso and Charles Wheatstone started using proper encryption keys that will later evolve to make it more complex over time. In the 1970s is when we first started to get closer into using actually encryption algorithms which was based on a “crypto group”, named the IBM’s team which purpose was to protect data and so they called the encryption, Data Encryption Standard(DES) which was the first cipher encrypted algorithm that later was cracked in 1997. The current state of encryption has landed in which encryption has become harder and harder for attackers to be able to infiltrate or steal data and that’s why there are so many ciphers that are recommended to make encrypting reliable(Thales, 2022)

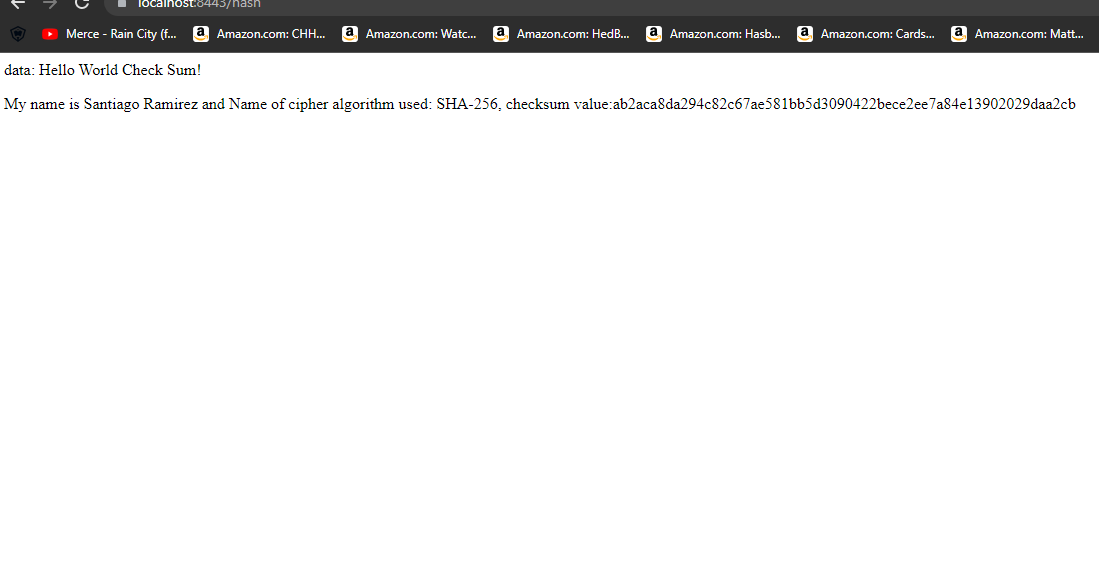
## 2. Certificate Generation

## Cmd certificate:

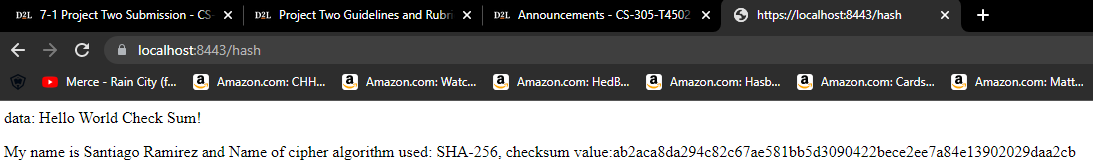
## 

Certificate:

## 3. Deploy Cipher

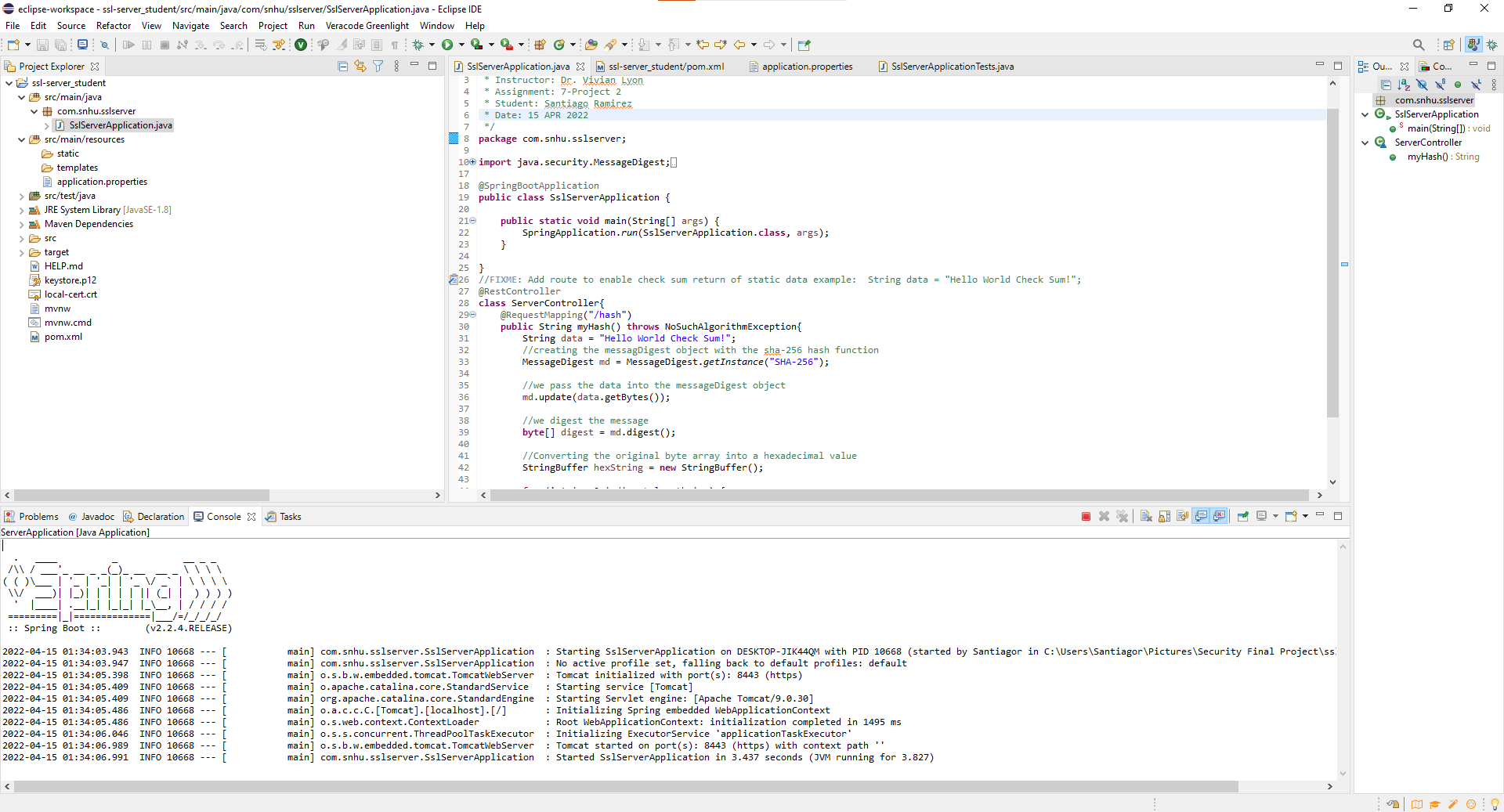


## 4. Secure Communications

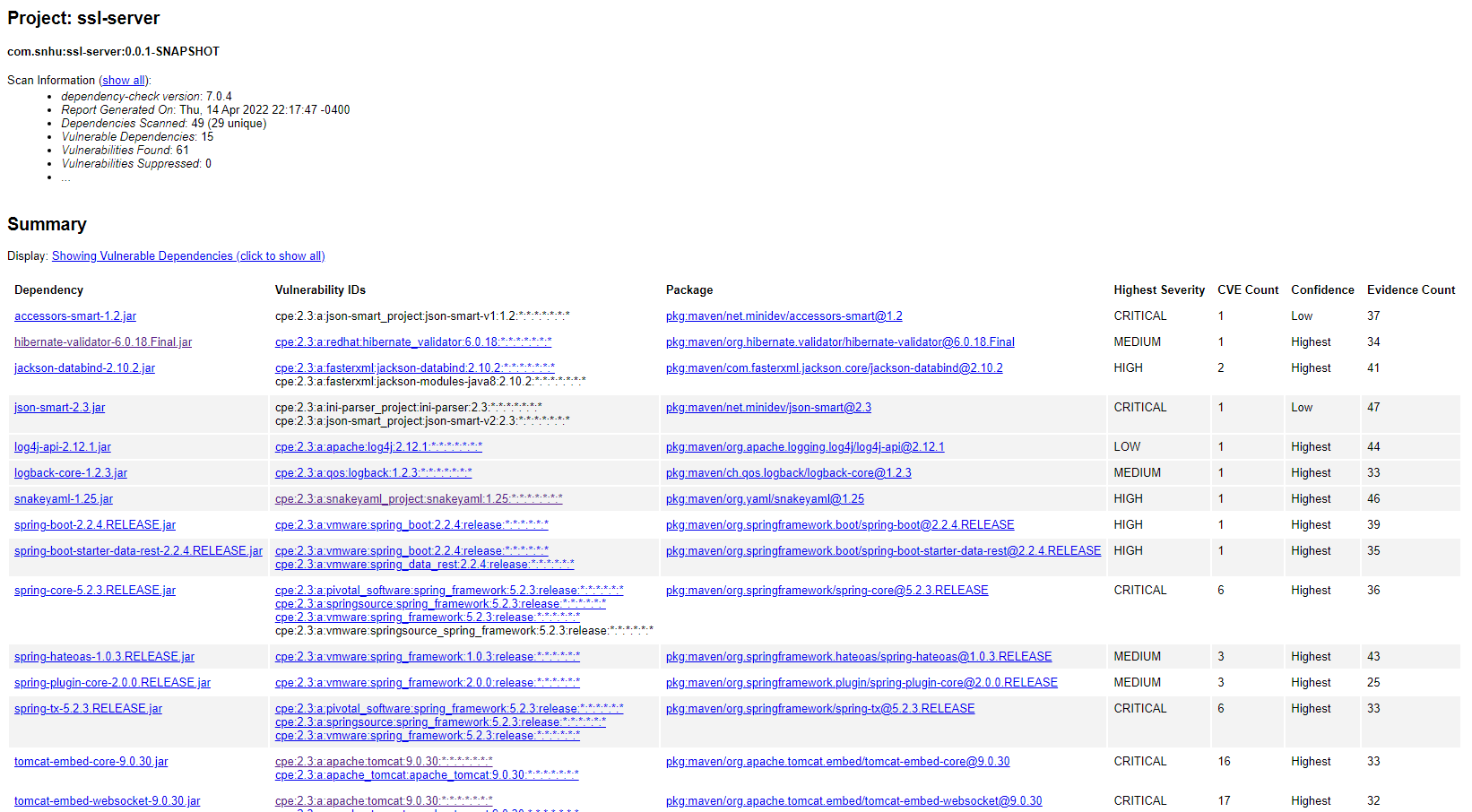


## 5. Secondary Testing

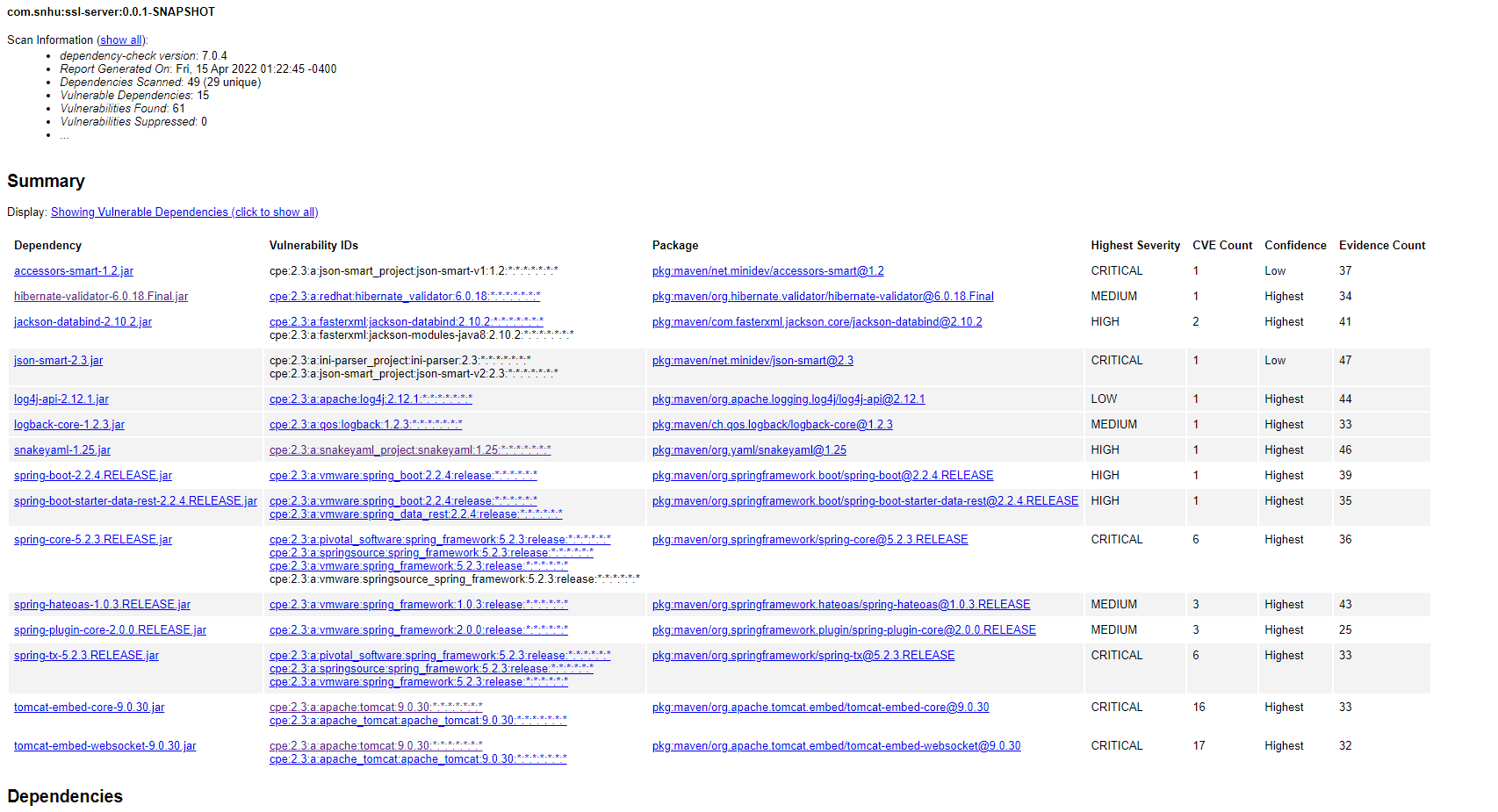
Refactored Code Executed without errors



Dependency before refactor:

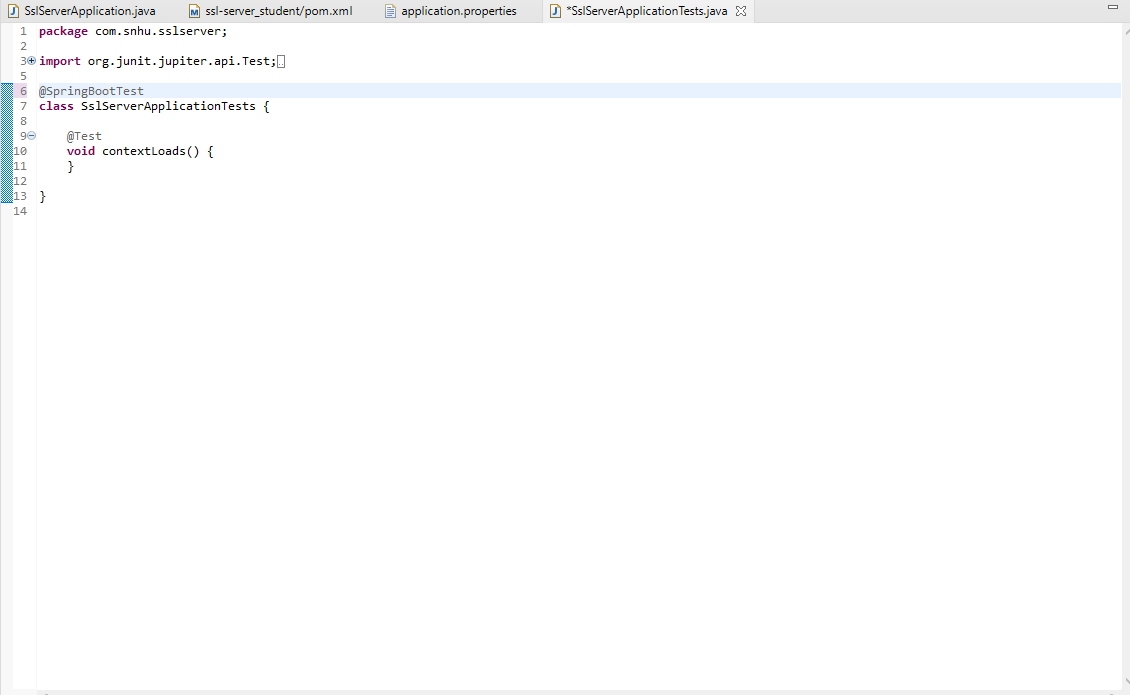


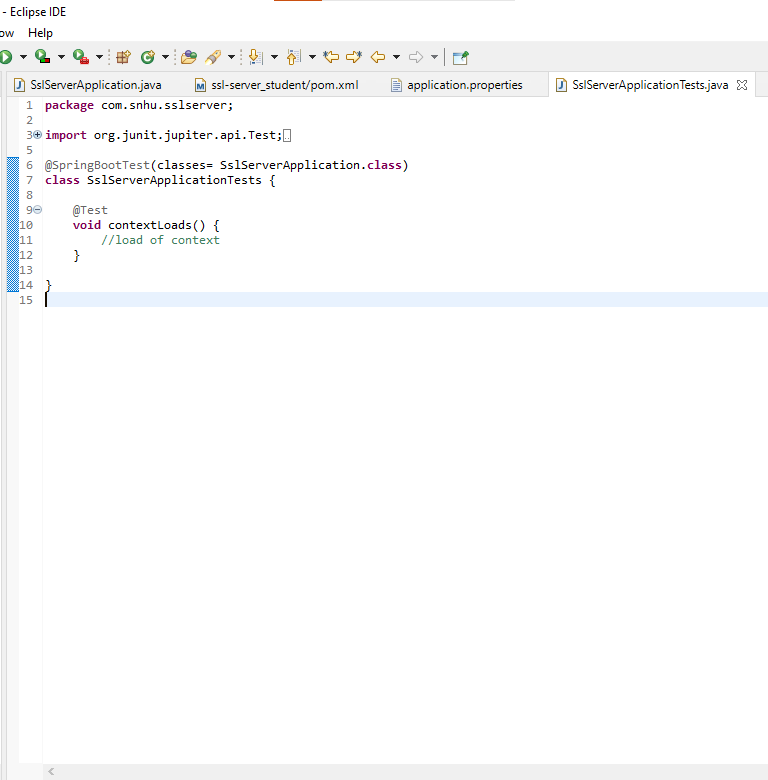
Dependency after refactor:

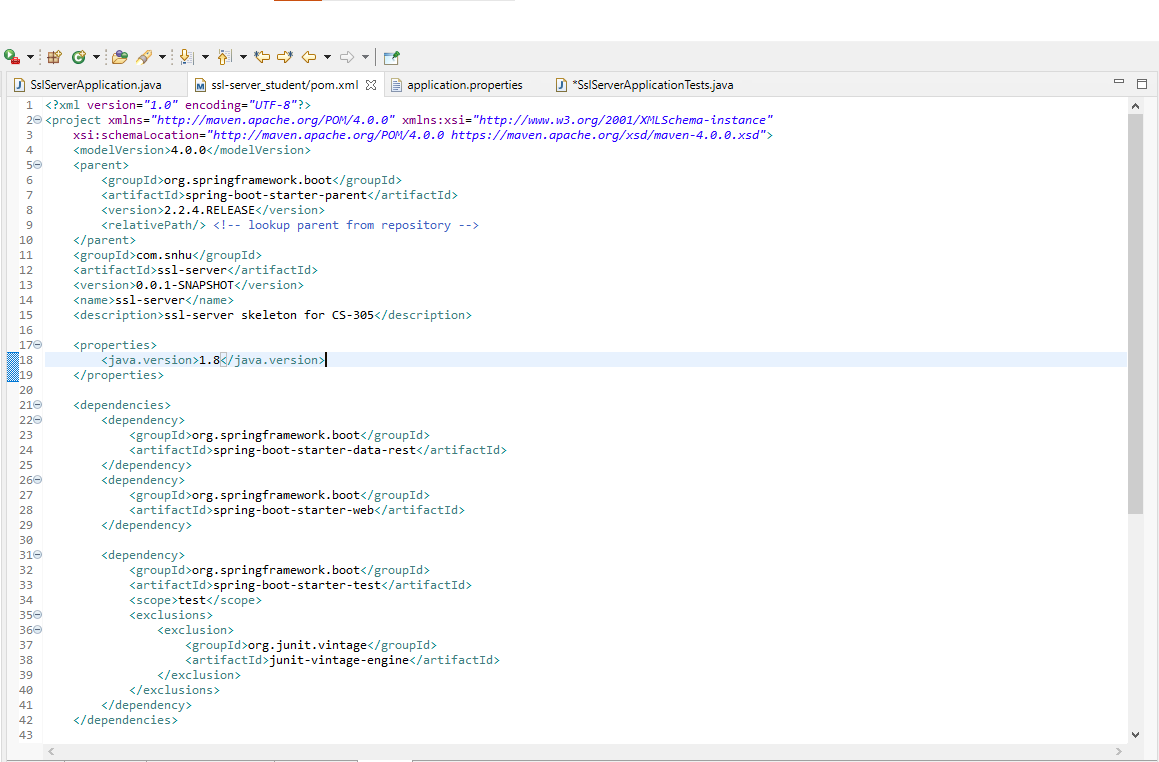


## 6. Functional Testing

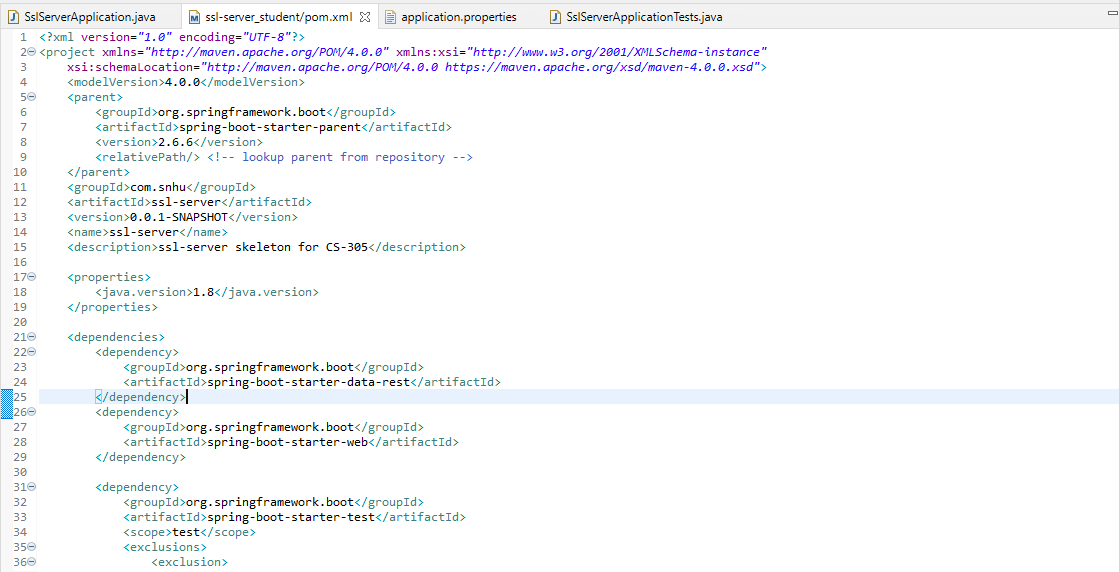
In this screenshot we can determined that in line 6 there is no reference to the class SslServerApplication will make it connect to the SslServerApplicationtest as well as there is just a blank space in line 11 for the context load.



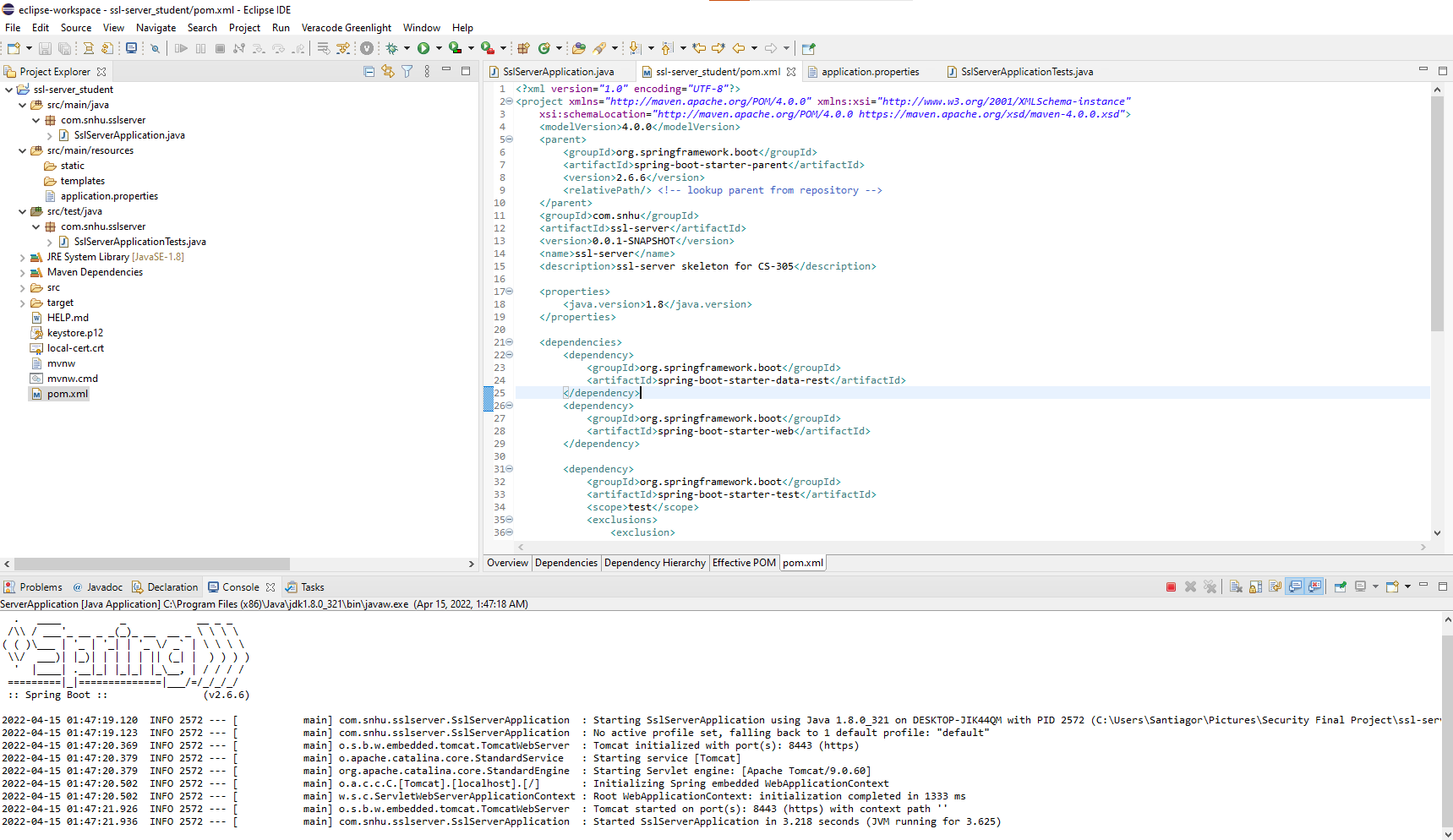
Refactoring:

Line 8 in this code has an older version of what the code should be which is 2.6.6

Refactoring:



No errors:

****

**BONUS:**

Dependency after all refactoring and version correction:



## 7. Summary

Within implementing the factoring code for the code the way I approach fixing the issues in reference to the Vulnerability Assessment Process Flow Diagram is to secure API interactions and I did this by updating the version of the API through the means of updating the spring framework that was in charge of keeping the API secure and also having the correct certificates that were necessary to insure the HTTPS was secure which also helps with the API. Furthermore, the Cryptography was one of the major factors in factoring the code in this project since I had to use checksum that allow to make the encrypting text to detect errors in the encryption as well as implementing the hash-256 which is a reliable hash function against collision. Client/Server was a crucial factor in the mitigation of the code since with the help of the certificates it maintains a secure website server within the HTTPS protocol. Code practice was also important when explaining the code that was change and implementing readability as well as a best practice standard of coding within java.

The process of adding layers to maintain a secure software application which will add to the company’s overall wellbeing is the fact that I first ensure that the encrypted text was secure with the help of hash-256, then the next layer of protection will be to generate a checksum to check for error within the encryption, later the next layer will be the implementation of certificates to ensure that the website was fully secure, then the ratification of the vulnerabilities within the code by using the dependency check and finally the manual observation of logical, syntactical or vulnerabilities that may be within the code itself.

The best practices for maintaining the current security of the software application for my customer for future reference will be to constantly maintain the system updated and this will mean to update the framework, APIs, encryption cipher and certification because most vulnerabilities that are modern have mostly been tackle through the newer versions. Furthermore, keeping a constant vulnerability check is important such as regularly checking on the dependency report to understand if there has been any vulnerabilities that are detected in the system and need to be fixed.

Reference

*A brief history of encryption*. Thales Group. (2021, October 1). Retrieved April 15, 2022, from https://www.thalesgroup.com/en/markets/digital-identity-and-security/magazine/brief-history-encryption

Callaghan, P. (n.d.). *Why you should use SHA-256 in evidence authentication*. Pagefreezer Blog. Retrieved April 15, 2022, from https://blog.pagefreezer.com/sha-256-benefits-evidence-authentication#:~:text=The%20main%20reason%20technology%20leaders,some%20other%20popular%20hashing%20algorithms.

CuongQuay. (2020, September 23). *RNG - the secret of cryptography*. Medium. Retrieved April 15, 2022, from https://medium.com/problem-solving-blog/rng-the-secret-of-cryptography-46d10a405924

*SHA-256 algorithm: N-able*. N. (2021, April 1). Retrieved April 15, 2022, from https://www.n-able.com/blog/sha-256-encryption#:~:text=SHA%2D256%20is%20a%20patented,as%20long%20as%20when%20unencrypted.

Wang, J., Liu, G., Chen, Y., & Wang, S. (2021). *Construction and Analysis of SHA-256 Compression Function Based on Chaos S-Box. IEEE Access, Access*, IEEE, 9, 61768–61777. https://doi-org.ezproxy.snhu.edu/10.1109/ACCESS.2021.3071501