****

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[Document Revision History 3](#_Toc33111302)

[Client 3](#_Toc33111303)

[Instructions 3](#_Toc33111304)

[Developer 4](#_Toc33111305)

[1. Algorithm Cipher 4](#_Toc33111306)

[2. Certificate Generation 4](#_Toc33111307)

[3. Deploy Cipher 4](#_Toc33111308)

[4. Secure Communications 4](#_Toc33111309)

[5. Secondary Testing 4](#_Toc33111310)

[6. Functional Testing 5](#_Toc33111311)

[7. Summary 5](#_Toc33111312)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **10/13/2021** | **Lynn McCargar** | **Final Revision** |

## 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

Lynn McCargar

## 1. Algorithm Cipher

Determine an appropriate encryption algorithm cipher to deploy given the security vulnerabilities, justifying your reasoning. Be sure to address the following:

* Provide a brief, high-level overview of the encryption algorithm cipher.
* Discuss the hash functions and bit levels of the cipher.
* Explain the use of random numbers, symmetric vs non-symmetric keys, and so on.
* Describe the history and current state of encryption algorithms.

The appropriate encryption algorithm cipher for this project is Advanced Encryption Standard 256 (AES-256). AES is a symmetric block cipher which data is divided into four by four arrays containing 16 bytes. AES hash is a secure hash function, meaning it takes an arbitrary bit string as input and returns a fixed length (in our case, 256 bit) string as output.

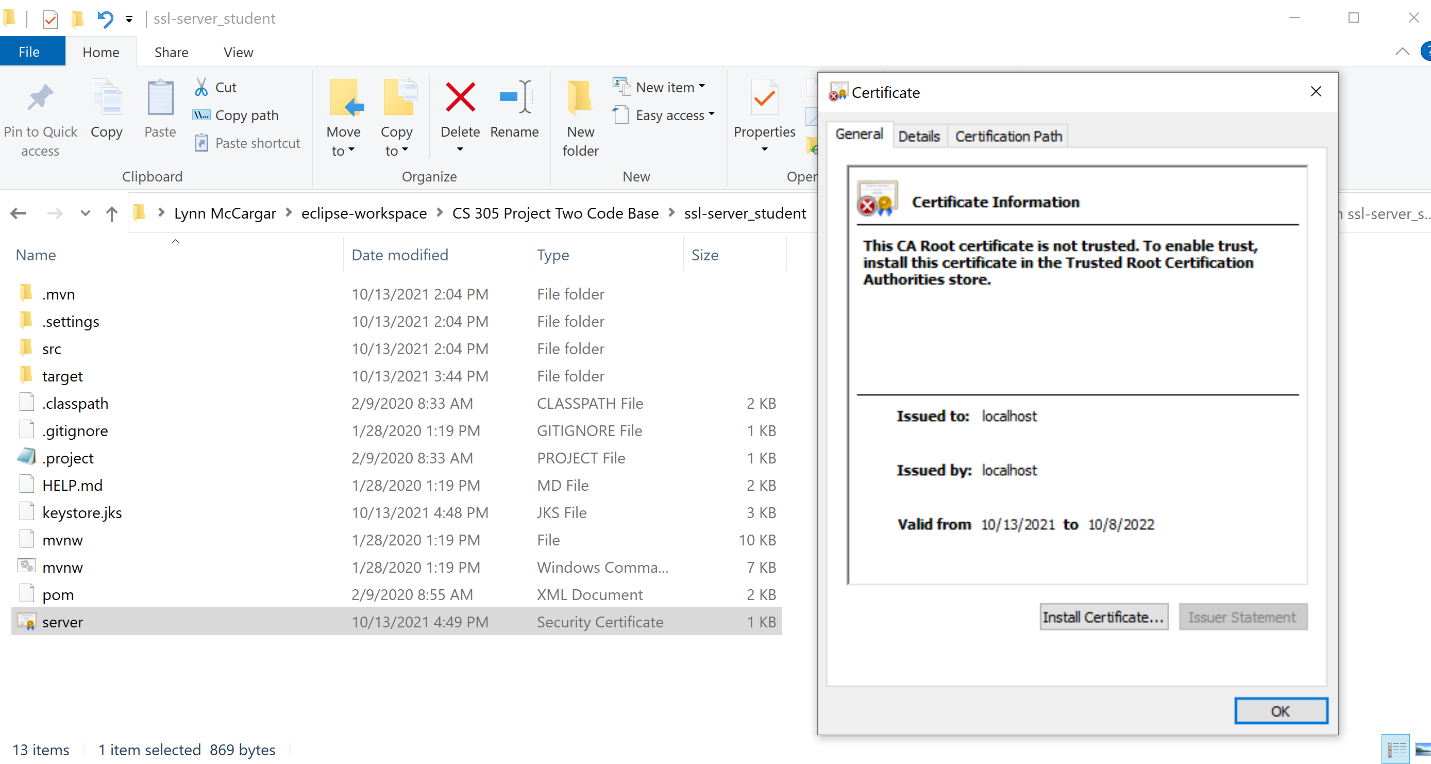
AES is a symmetric key cipher. Which means the same secret key is used for both encryption and decryption messages, and both the sender and receiver of the data need a copy of the key. Whereas, asymmetric key systems use a different key for each of the two processes. Asymmetric keys are best for external file transfers, so symmetric keys are better suited to internal encryption. The advantage of symmetric systems like AES is speed. Thus a symmetric key algorithm requires less computational process than an asymmetric one, it is faster and much more efficient.

A banker during the Civil War named Frank Miller in 1882 designed an encryption method that lasted for nearly 140 years. This method used random numbers to encrypt and decrypt messages. Currently the National Institute of Standards and Technology (NIST) has standardized this encryption cipher to secure sensitive data since 2001.

## 2. Certificate Generation

Generate appropriate self-signed certificates using the Java Keytool, which is used through the command line.

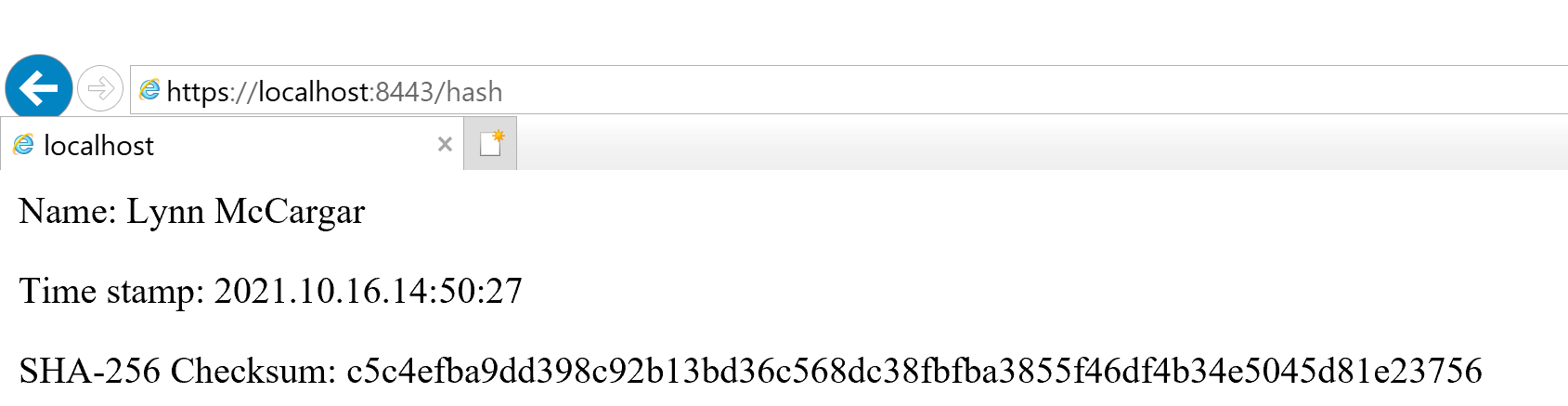
* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.



## 3. Deploy Cipher

Refactor the code and use security libraries to deploy and implement the encryption algorithm cipher to the software application. Verify this additional functionality with a checksum.

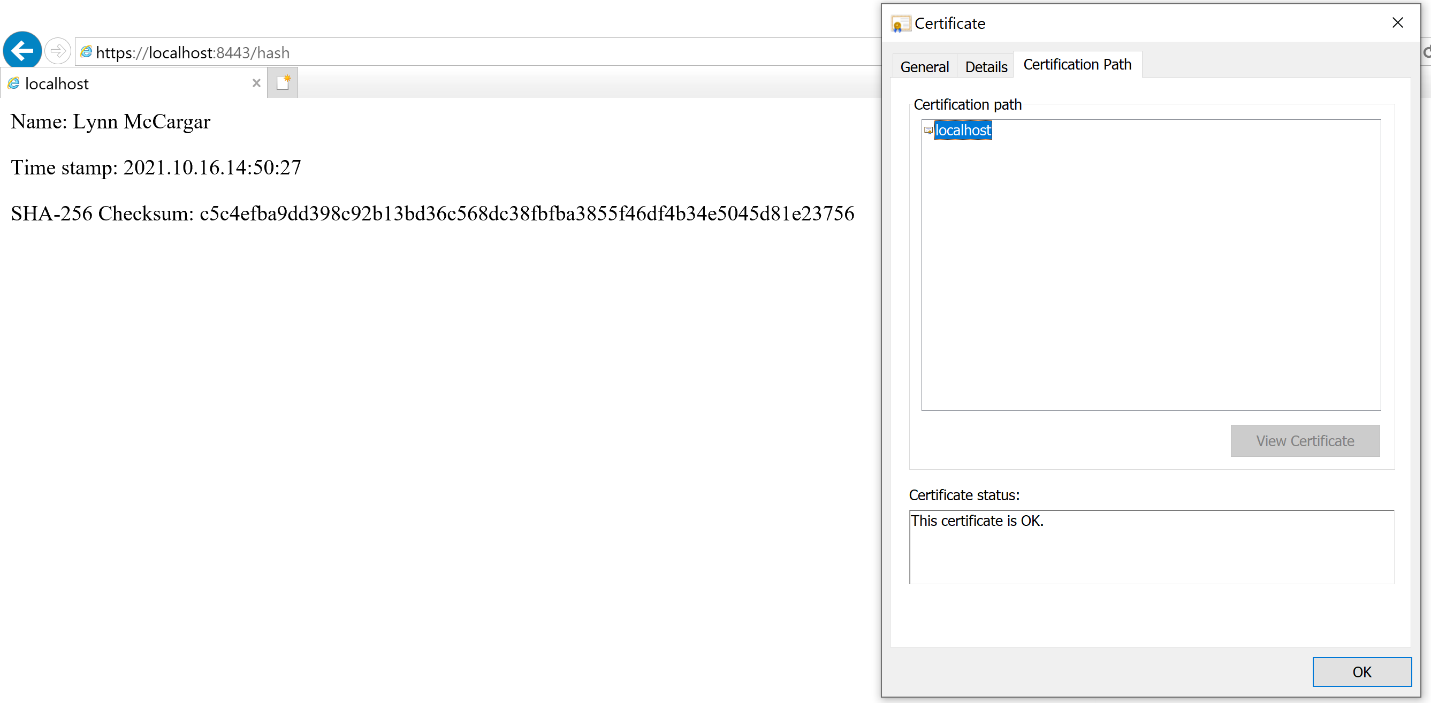
* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.



## 4. Secure Communications

Refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code to verify secure communication by typing **https://localhost:8443/hash** in a new browser window to demonstrate that the secure communication works successfully.

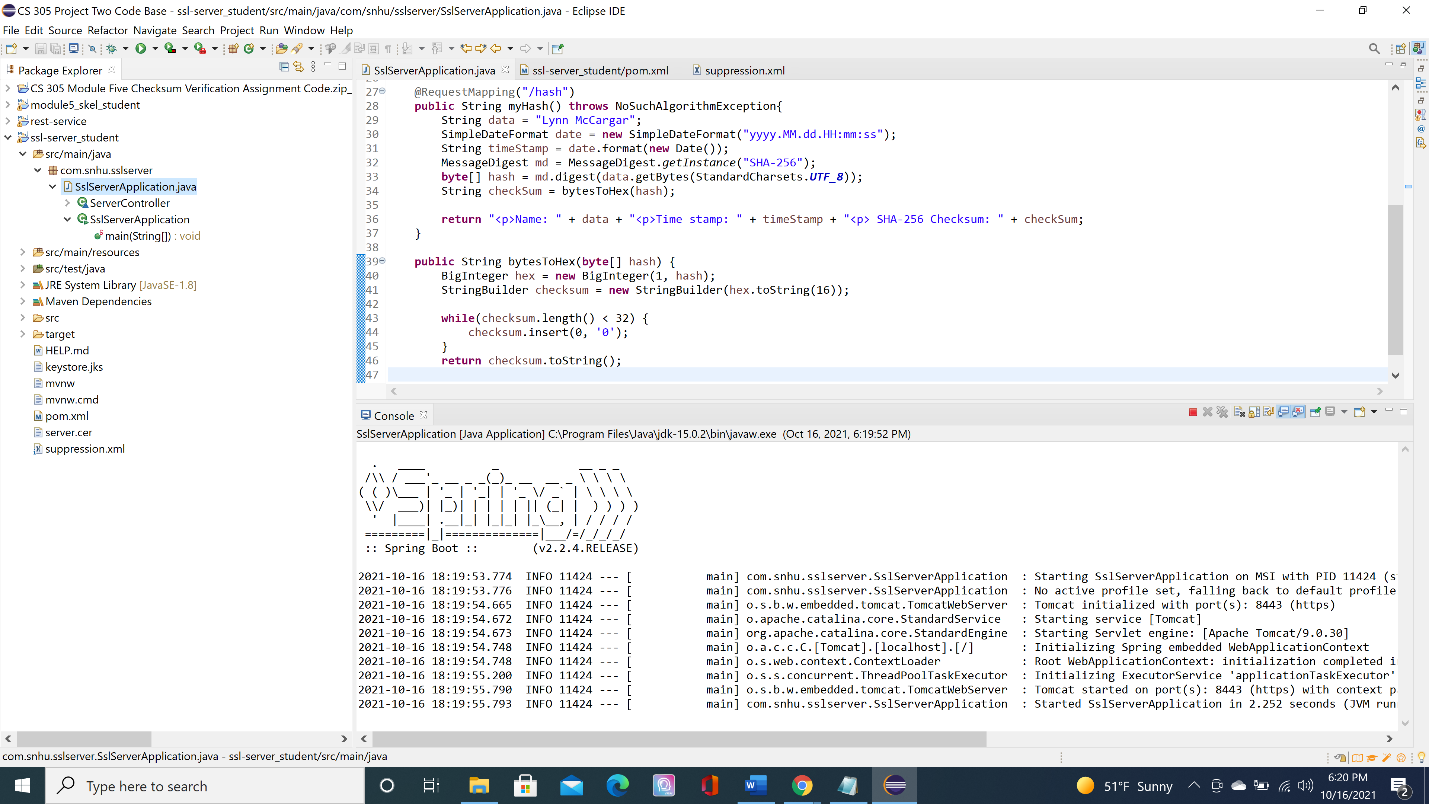
* Insert a screenshot below of the web browser that shows a secure webpage.

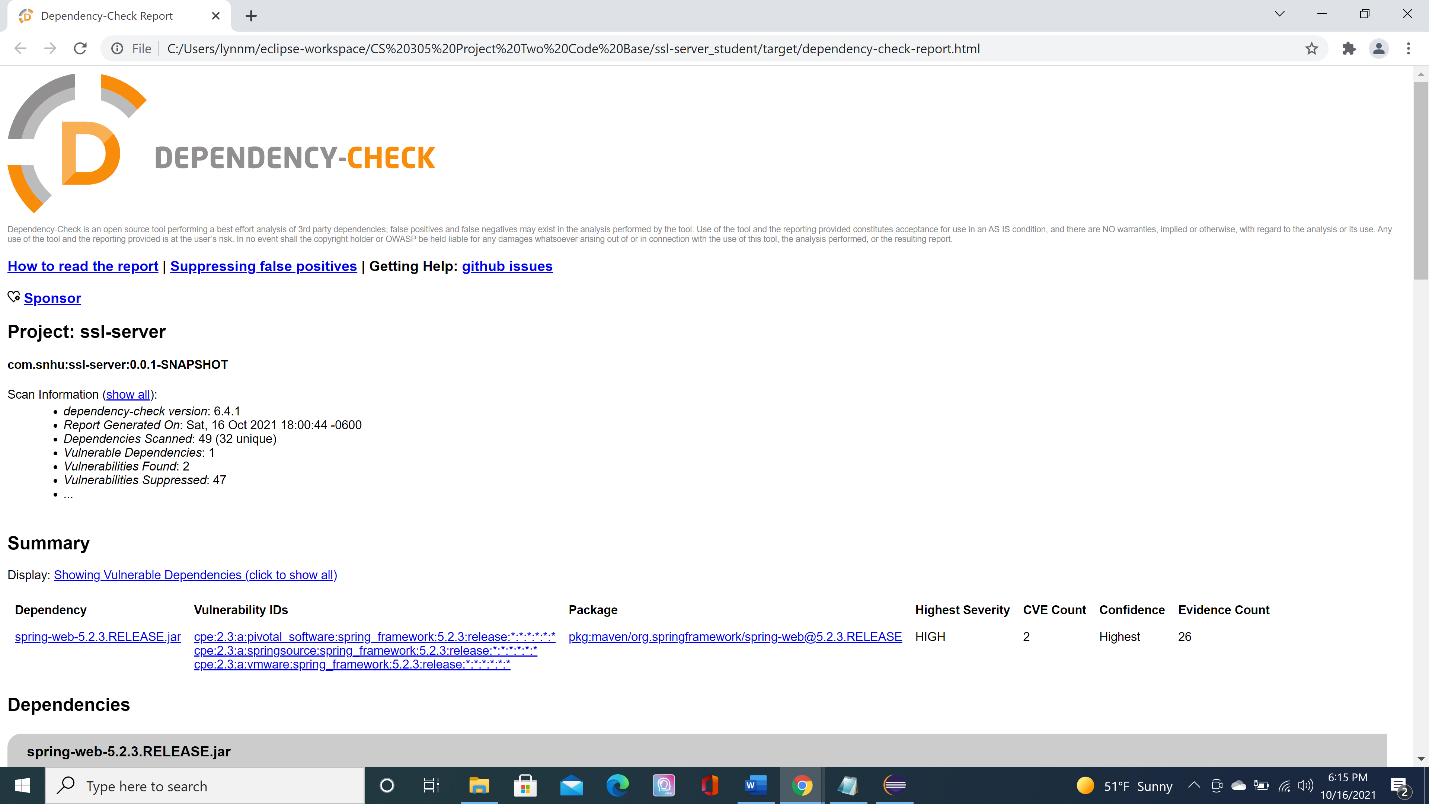


## 5. Secondary Testing

Complete a secondary static testing of the refactored code using the dependency check tool to ensure code complies with software security enhancements. You only need to focus on the code you have added as part of the refactoring. Complete the dependency check and review the output to ensure you did not introduce additional security vulnerabilities.

* Include the following below:
  + A screenshot of the refactored code executed without errors
  + A screenshot of the dependency check report

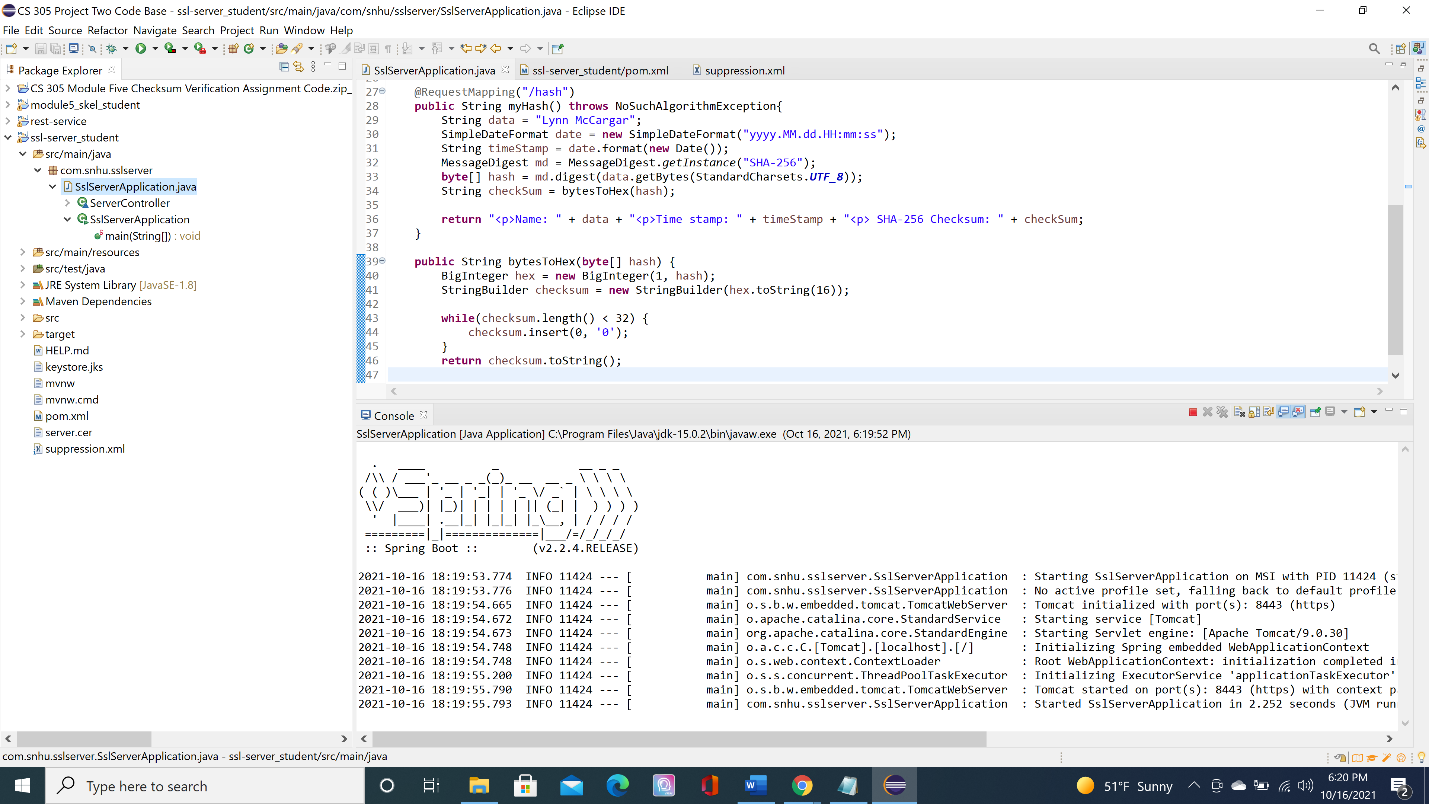


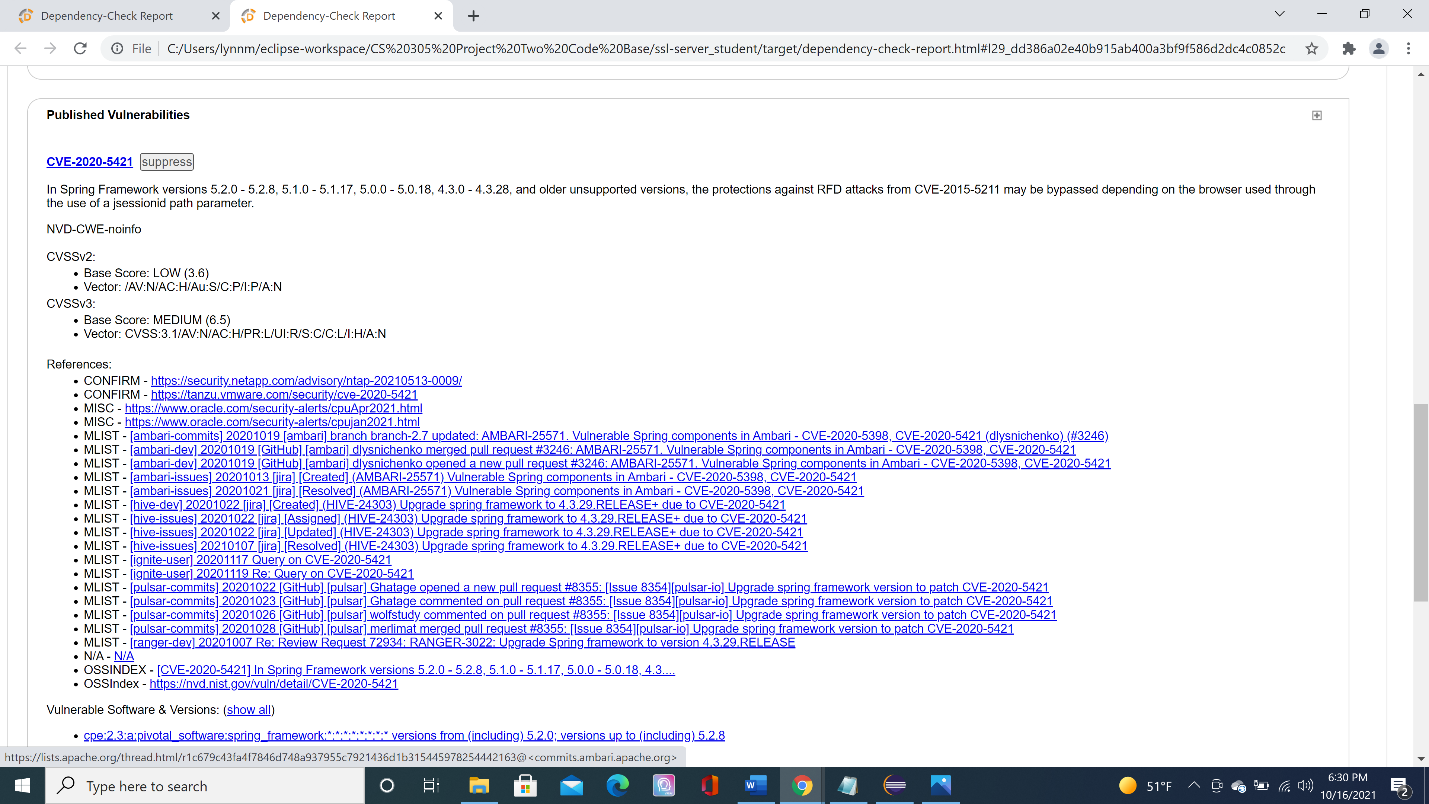


## 6. Functional Testing

Identify syntactical, logical, and security vulnerabilities for the software application by manually reviewing code.

* Complete this functional testing and include a screenshot below of the refactored code executed without errors.



After suppressing all the false positives, I was unable to suppress one vulnerability dependency and two vulnerability not related to my code. Below are the screenshots of the unsuppressed false positives.  




## 7. Summary

Discuss how the code has been refactored and how it complies with security testing protocols. Be sure to address the following:

* Refer to the Vulnerability Assessment Process Flow Diagram and highlight the areas of security that you addressed by refactoring the code.
* Discuss your process for adding layers of security to the software application and the value that security adds to the company’s overall wellbeing.
* Point out best practices for maintaining the current security of the software application to your customer.

The self-signed certificate was a security measure to enable a secure socket layer for the web access. I refactored the code in the POM.xml to incorporate the latest dependency check (version 6.4.1). Adding hash mapping and SHA 256 in the SslServerApplication.java. The layer of security implemented was enabling a secure socket layer for the web site. Creating a hash mapping to encrypt the data. Running a static dependency check to find the false positives. And using a functional test to manually inspect the dependencies. All this ensures the wellbeing of the company’s sensitive data.

Best practices for maintaining the current secure application is a mindset that needs to be followed. Meaning, a DevSecOps should be used in the company. Security training for company developers as a best practice in development. Ensuring security is part of the software development life cycle. Limit user access to the application. Ensuring there are no data leaks outside of the secure application. Encrypting company data and applying input validation.