****

# Practices for Secure Software Report

Table of Contents

[Document Revision History 3](#_Toc102040754)

[Client 3](#_Toc102040755)

[Instructions 3](#_Toc102040756)

[Developer 4](#_Toc102040757)

[1. Algorithm Cipher 4](#_Toc102040758)

[2. Certificate Generation 4](#_Toc102040759)

[3. Deploy Cipher 4](#_Toc102040760)

[4. Secure Communications 4](#_Toc102040761)

[5. Secondary Testing 4](#_Toc102040762)

[6. Functional Testing 4](#_Toc102040763)

[7. Summary 4](#_Toc102040764)

[8. Industry Standard Best Practices 4](#_Toc102040765)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **14OCT22** | **Dustin Haugh** |  |

## Client



## Instructions

Submit this completed practices for secure software report. Replace the bracketed text with the relevant information. You must document your process for writing secure communications and refactoring code that complies with software security testing protocols.

* Respond to the steps outlined below and include your findings.
* Respond using your own words. You may also choose to include images or supporting materials. If you include them, make certain to insert them in all the relevant locations in the document.
* Refer to the Project Two Guidelines and Rubric for more detailed instructions about each section of the template.

## Developer

Dustin Haugh

## Algorithm Cipher

Recommend an appropriate encryption algorithmcipher to deploy, given the security vulnerabilities, and justify your reasoning. Review the scenario and the supporting materials to support your recommendation. In your practices for secure software report, be sure to address the following:

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

Artemis Financial

There are two encryption methods that I would recommend for use in Artemis Financial data. This would include:

* AES (Advanced Encryption Standard which is the Rjindael cipher)
* RSA (Rivest-Sharmir-Adleman) if the key contains a minimum key length of 2048 bits

AES 256 is a symmetric encryption method that would be useful in local database encryption, or on larger files that need to be transferred quickly but have less security. It’s still very secure, but only as secure as the private key it uses.

When transferring smaller—more sensitive data, the asymmetric encryption method RSA would be more appropriate. This is more secure online and uses both a private and public key to decrypt.

To add security to certain types of data, like passwords, I would not encrypt but instead use a hashing algorithm like SHA2 – 256. I wouldn’t need to store the password itself, but instead just the hash value.

## Certificate Generation

Generate appropriate self-signed certificates using the Java Keytool in Eclipse.

1. To demonstrate that the certificate was correctly generated:
   1. Export your certificates (CER file).
   2. Submit a screenshot of the CER file in your practices for secure software report.

Insert a screenshot below of the CER file.

Text

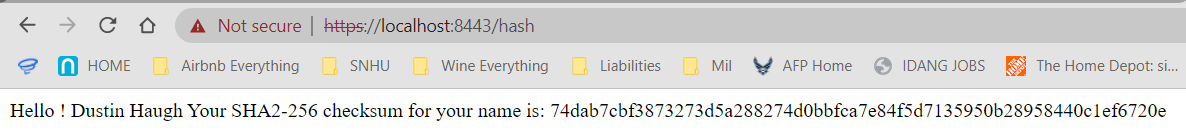
Description automatically generated

## Deploy Cipher

Deploy and implement the cryptographic hash algorithm by refactoring code. Demonstrate functionality with a checksum verification.

1. Submit a screenshot of the checksum verification in your practices for secure software report. The screenshot must show your name and a unique data string that has been created.

Insert a screenshot below of the checksum verification.



This is the checksum my application created based on the input of my name (Dustin Haugh). I then verified the has value with an independent source. It works!

<https://xorbin.com/tools/sha256-hash-calculator>

Graphical user interface, text, application, email

Description automatically generated

## Secure Communications

Verify secure communication. In the application.properties file, refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code. Then once the server is running, type https://localhost:8443/hash in a new browser to demonstrate that the secure communication works successfully.

1. Create a screenshot of the web browser that shows a secure webpage and include it in your practices for secure software report.

## 

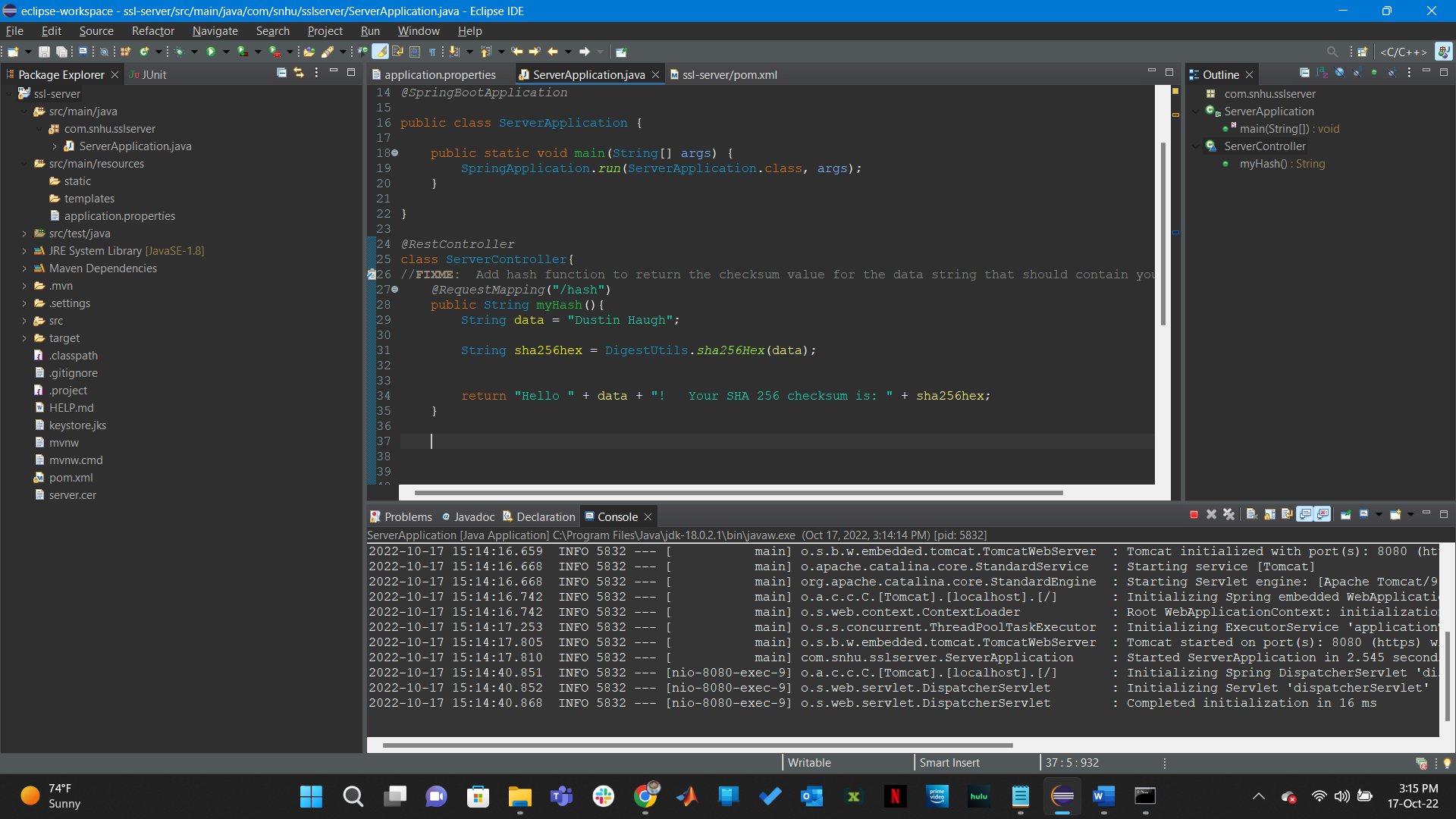
I notice that with the self-signed certificate that it is showing as invalid, but I am at least able to access using https protocol and see my certificate information.

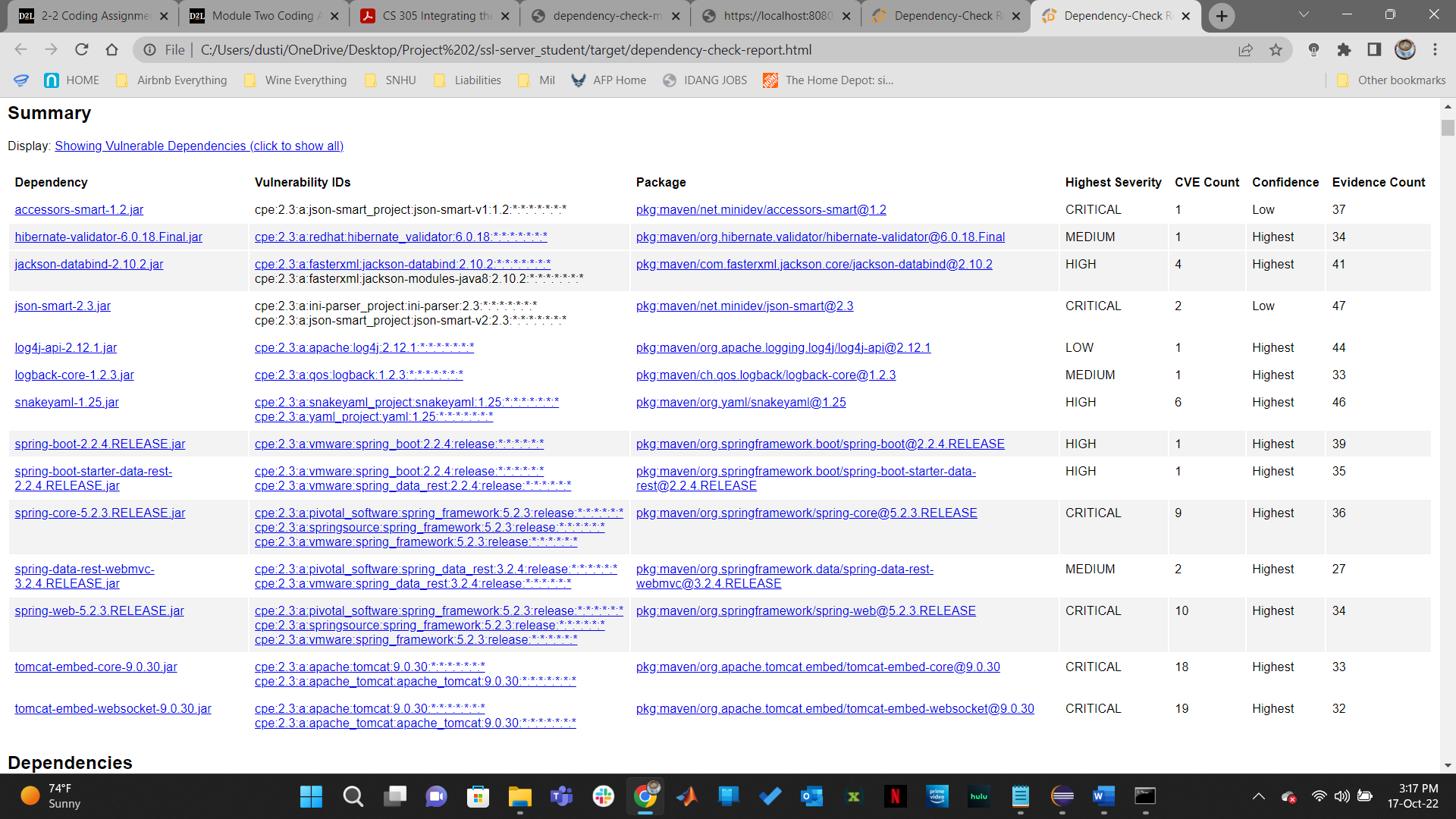
## Secondary Testing

Run a secondary static testing of the refactored codeusing the OWASP Dependency-Check Maven (see Supporting Materials) to ensure code complies with software security enhancements. You need to focus on only the code you have added as part of the refactoring. Complete the dependency check and review the outputto ensure you did not introduce additional security vulnerabilities. Include the following in your practices for secure software report:

1. A screenshot of the refactored code executed without errors
2. A screenshot of the report of the output from the dependency-check static tester

Insert screenshots below of the refactored code executed without errors and the dependency-check report.





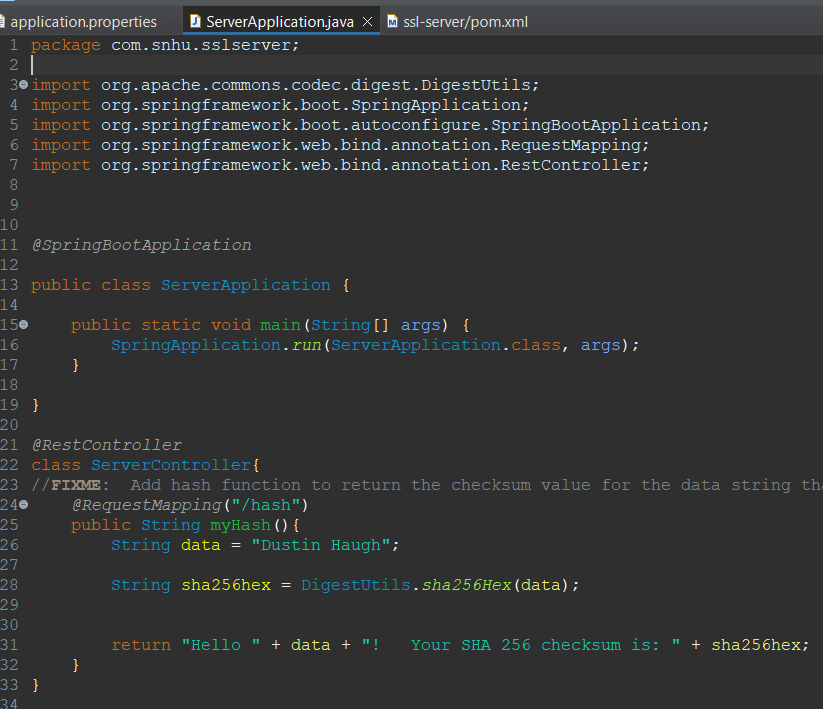
My code works, and I did not introduce any new threats. However, I did not ignore the past threats, and I’m not quite sure if we needed to do that.

## Functional Testing

Identify the software application's syntactical, logical, and security vulnerabilitiesby manually reviewing code.

1. Complete this functional testing and include a screenshot of the refactored code, executed without errors, in your practices for secure software report.

Insert a screenshot below of the refactored code executed without errors.



[Insert screenshots here.]

## Summary

Discuss how the code has been refactored and how it complies with security testing protocols. In the summary of your practices for secure software report, be sure to address the following:

1. Refer to the Vulnerability Assessment Process Flow Diagram. Highlight the areas of security that you addressed by refactoring the code.
2. Discuss your process for adding layers of security to the software application.

I integrated OWASP static testing, Apache Commons Codec for SHA 256 and other functions, and added a secured REST controller to return a hashed value of my name. I also ensured that I did not have any errors in the code, and checked to make sure that I did not add any vulnerabilities in static testing.

## Industry Standard Best Practices

Explain how you applied industry standard best practices for secure codingto mitigate against known security vulnerabilities.Be sure to address the following:

1. Explain how you used industry standard best practices to maintain the software application’s current security.
2. Explain the value of applying industry standard best practices for secure coding to the company’s overall wellbeing.

I utilized static testing to check for threats, suggested and utilized SHA 256 methods for encrypting data, and regularly tested the program.

Testing code often for security threats is a smart idea. If the threat is a false positive, I would suppress it so it was easier to keep track of new threats.