****

# 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** | **10/13/2023** | **Jessie Smith** |  |

## 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

Jessie Smith

## Algorithm Cipher

Artemis Financial wishes to modernize their operations, as well as to protect the data concerning clients and financial information. Encrypting the data stored in Artemis Financials servers, and the data communicated to clients, will help protect the data. If an attacker were to intercept any data, it would be useless without the key to the encryption. I recommend using SHA 256, as it is one of the most secure cryptographic hashing algorithms. SHA 256 has a hash output size of 256 bits, making it nearly impossible to reverse engineer to the original value. Since SHA 256 is a hash encryption method, it does not use asymmetric or symmetric keys to decrypt. The hash values output from SHA 256 varies even if a small difference is made, for example “bye” would output a different has value than “bye!”, so using hash value for file integrity can show if a file has been tampered with, even with the smallest changes.

Encryption methods date back to the ancient Spartans, who used cryptography to secure secret messages during battle, and while this wasn’t “computer cryptography”, it still played a major role in the way sensitive data and messages are handled today.

## Certificate Generation

Insert a screenshot below of the CER file.

A computer screen with white text

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.



This is a screenshot when using HTTP.

## Secure Communications

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



Here is an image on HTTPS, it is showing as “not secure” because my certificate is self signed:

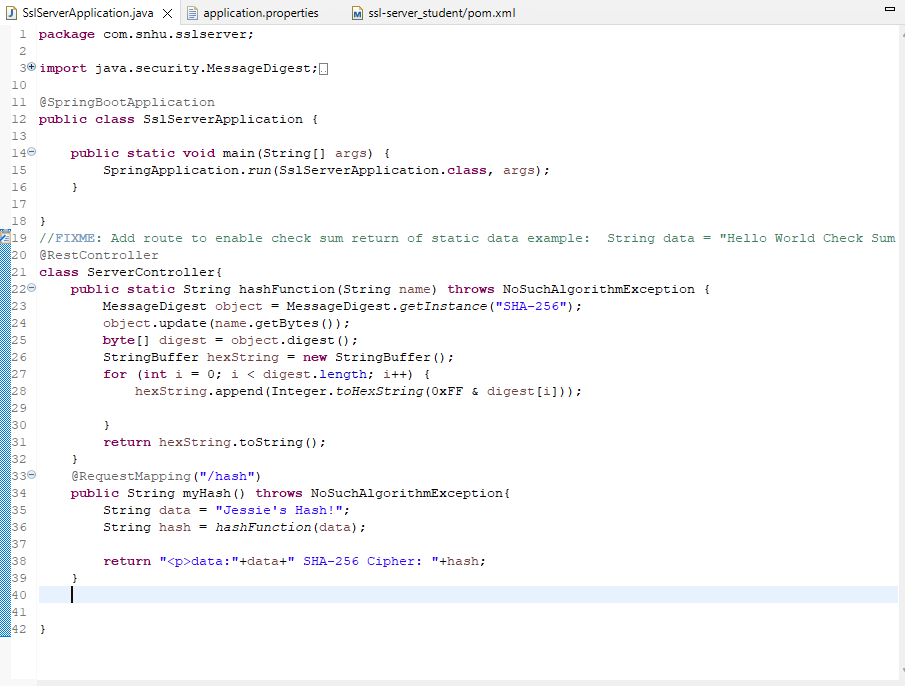
A screenshot of a computer error

Description automatically generated

## Secondary Testing

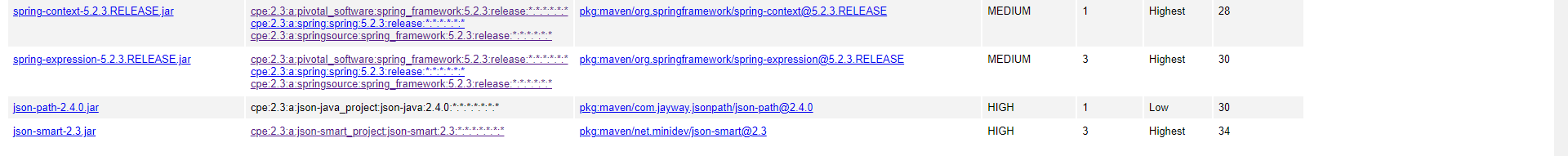
Insert screenshots below of the refactored code executed without errors and the dependency-check report.

Here is an image of the refactored code:



Here is a screenshot of the dependency check:

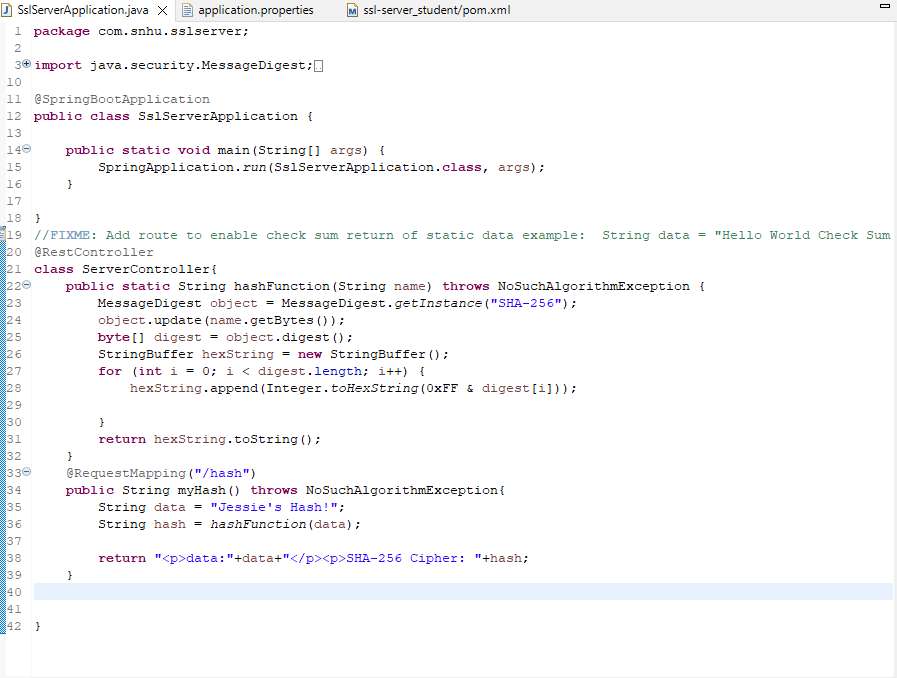
A screenshot of a computer

Description automatically generated

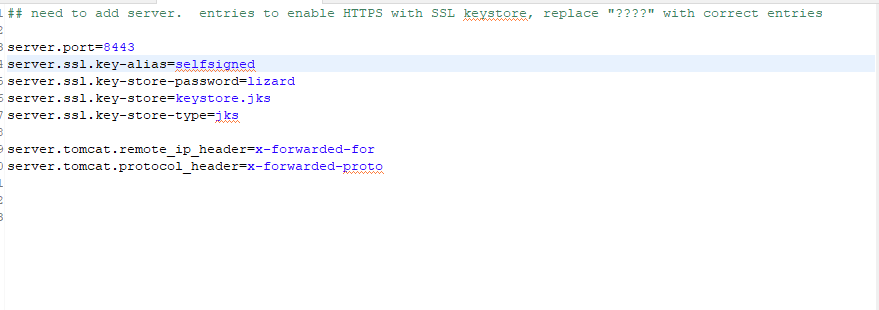
## Functional Testing

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

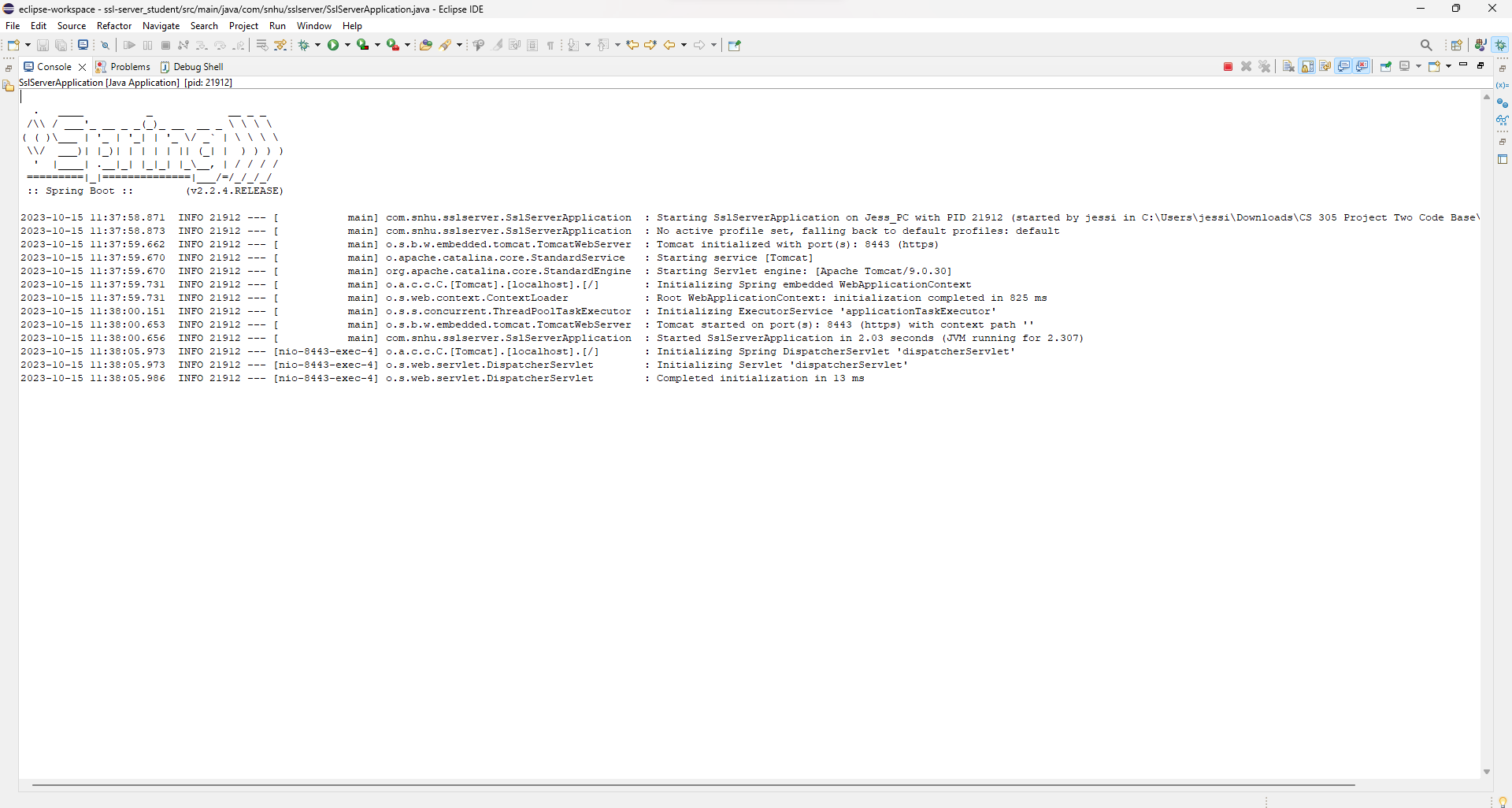
An image of the code (corrected to show the data and hash on separate lines):



An image of the application properties:



An image showing that the code runs without error on HTTPS:



And a final image of what the local server looks like with a newline between the data and hash:

A red and white rectangle

Description automatically generated

## Summary

In the refactored code, I added a RestController to help secure the application which follows the vulnerability assessment diagram. The controller helps to secure the application and handles the requests made through the application. SHA-256 is also used to follow the diagram as it is one of the most secure hash functions and reduces the chances of collision. The code is also able to run because of the “throws NoSuchAlgorithmException” which helps avoid exceptions or the application crashing.

## Industry Standard Best Practices

I used a RestController to help me build this application as a RESTful web service. The HTTP/HTTPS requests are handled through the controller. I also redirected the code to provide on an HTTPS local host instead of HTTP, although the web browser still says that the web service is “not secure”, that is because my certificate is selfsigned and therefore not valid. Refactoring the code to follow RESTful guidelines will help the application become a client/server application, providing more security for the website. As the code is further advanced, authentication can also be included to further protect data transmitted through the application.