****

# 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** | **10/23/22** | **Kyle Blan** | **Initial Release.** |

## 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

Kyle Blan

## Algorithm Cipher

The transfer of data is the primary security concern for Artemis Financial, and with this in mind, a good choice for this selection would be any relevant high bit encryption algorithm. The provided code base incorporates the SSL algorithm, which is a good choice for protecting this application’s traffic.

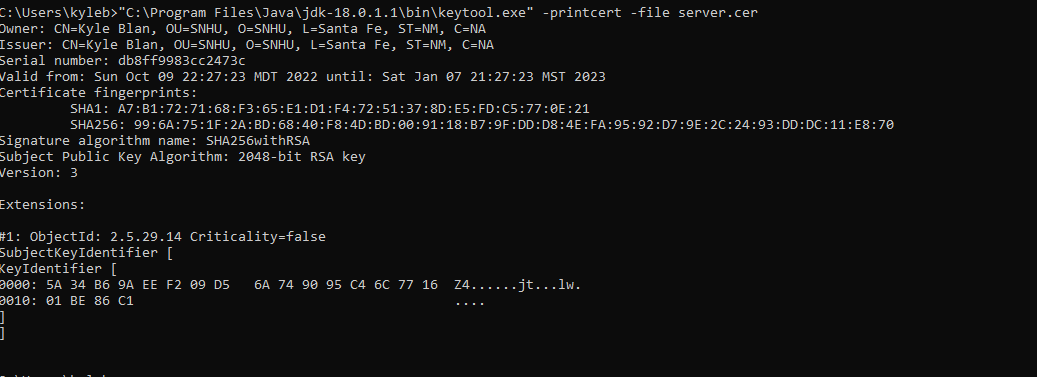
The SSL algorithm provides the necessary functionality for both data at-rest and in-transit, making it an ideal choice for an application that would incorporate several different databases. There is also ample functionality to implement a checksum to validate the successful transfer of data, and that the data in-transit remains unchanged. The file itself can then be hashed, while the key is provided.

I have chosen SHA-256 bit encryption specifically for this application, due to its middle of the road position per bitrate, and its ability to be both mobile and secure. This encryption has yet to be broken, and is likely more than sufficient security for the application in question, while remaining scalable.

A symmetric key would use the same key for both encrypting and decrypting, while a a non-symmetric key is the opposite, and the keys are different for both tasks. Ciphers such as SHA-256 have been used to send coded and secure communications since the invention of the computer, and have roots in the militarization of technology.

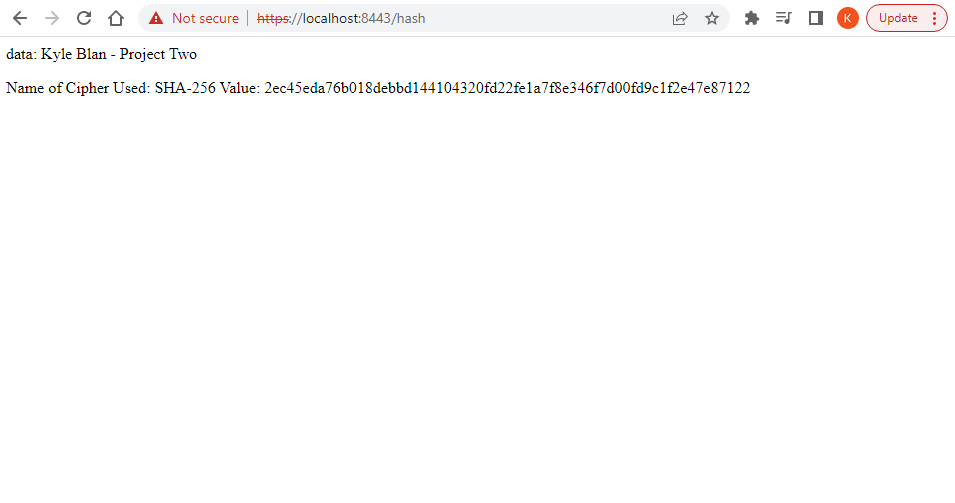
## Certificate Generation

Insert a screenshot below of the CER file.



## Deploy Cipher

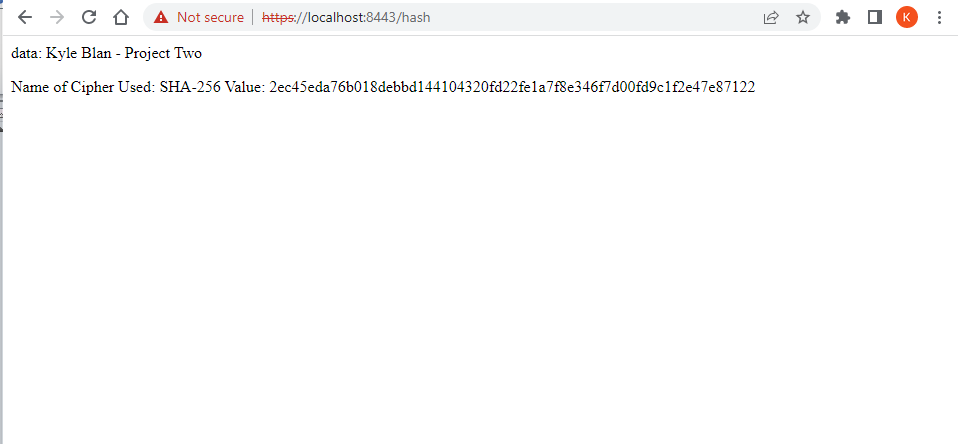
Insert a screenshot below of the checksum verification.



## Secure Communications

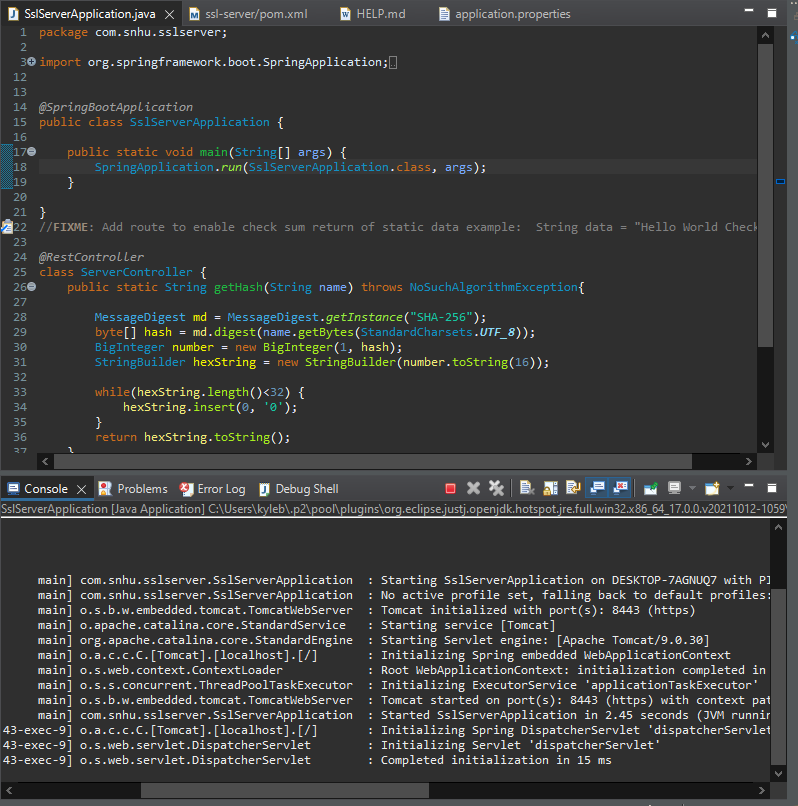
Insert a screenshot below of the web browser that shows a secure webpage.

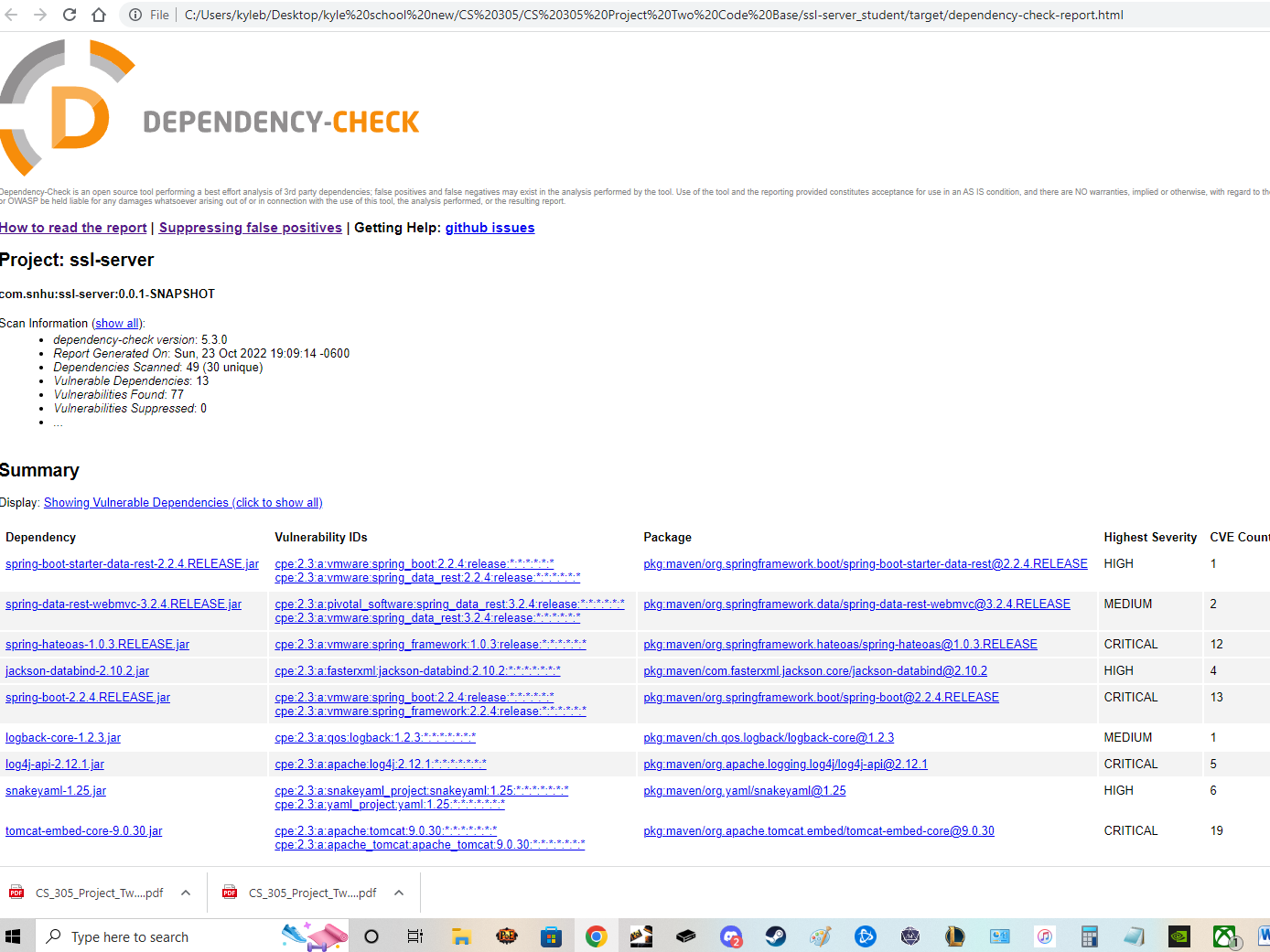
The screenshot below shows that this application does use the https protocol; however an error occurred in verifying the supplied certificate.



## Secondary Testing

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

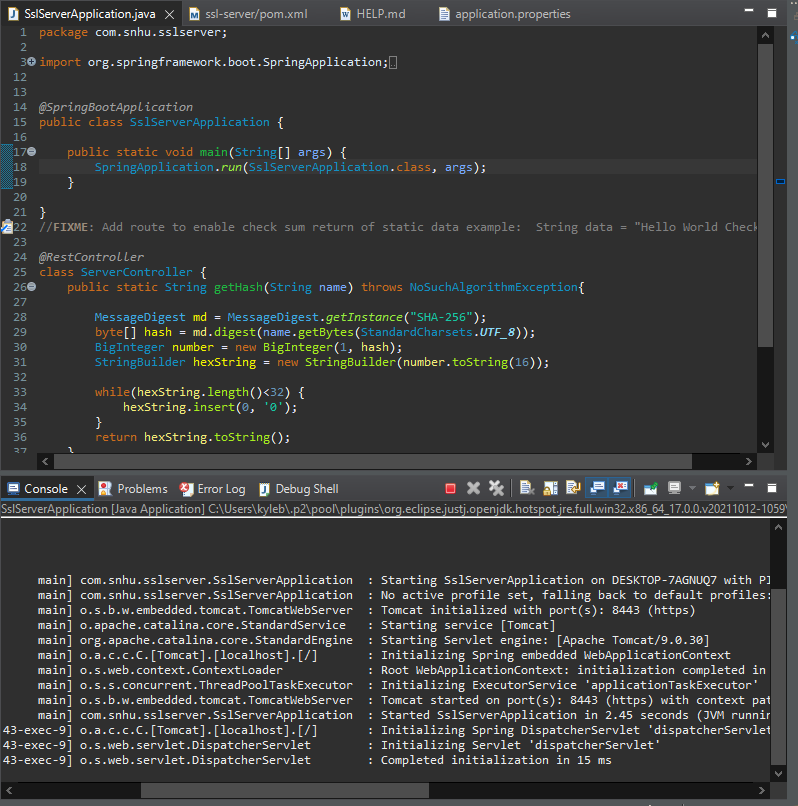




This screencap shows that no new vulnerabilities were introduced by the use of HTTPS protocol. A simple update to the springboot version would alleviate most of the listed vulnerabilities.

## Functional Testing

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



The above screenshot shows the code executed without error. Upon manual review, I found that there are still a few minor areas of risk for this type of communication. The plaintext values found in the input strings could cause an unintentional release of secured info, depending on who has access to the source code.

I performed this review with the following areas of focus, in testing for vulnerability:

HTTPS application, input validation, code quality, and the type of cryptography used

## Summary

This refactoring incorporated a checksum for input validation, and a self-signed certificate for use in HTTPS protocol. This method of data transfer as a whole seems to be secure, due to the nature of the encryption used, and the overall use case of the application. The use of financial data will be secure in its transfer from customer to database, and protected at rest by the same level of encryption.

The areas of security manually reviewed are as follows: HTTPS application, input validation, code quality, and the type of cryptography used

## Industry Standard Best Practices

The update of known dependencies, and the implementation of standard framework updates is widely considered an industry best practice, and will need to continue to be monitored in a deployed application. This will protect the customer from possible data breaches, and unintentional disclosure of their customer’s financial data.