****

# 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** | **April 20, 2025** | **Devin Wright** |  |

## 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

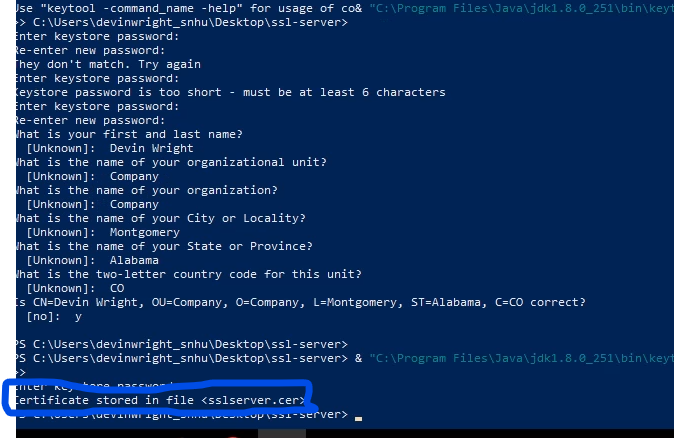
Devin Wright

## Algorithm Cipher

The program used SHA-256 hash function for creating a checksum from the static string "Hello World Check Sum!". The ChecksumController.java file created the checksum using the MessageDigest API in Java. The original string along with information about the hashing algorithm used and the resulting hash value is returned by the /checksum REST endpoint.

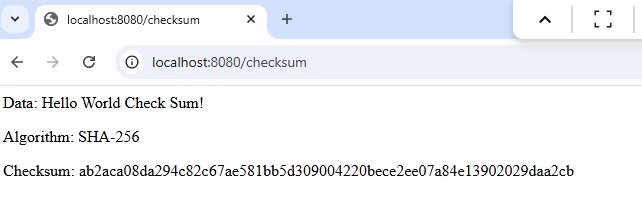
## Certificate Generation

Insert a screenshot below of the CER file.



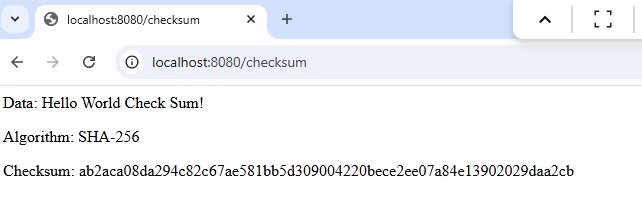
## Deploy Cipher

Insert a screenshot below of the checksum verification.



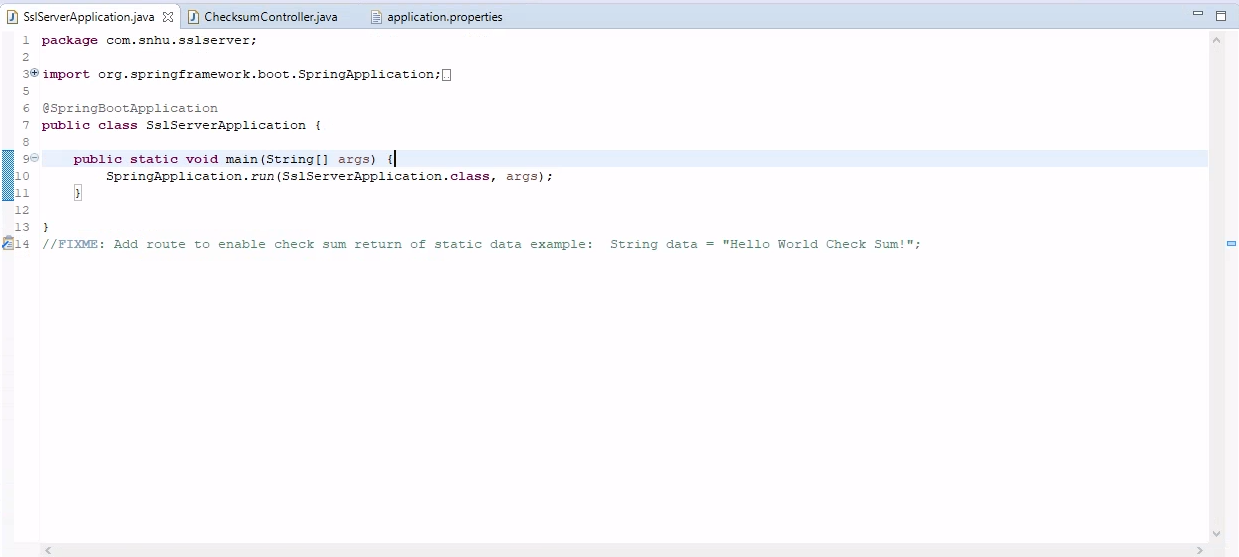
## Secure Communications

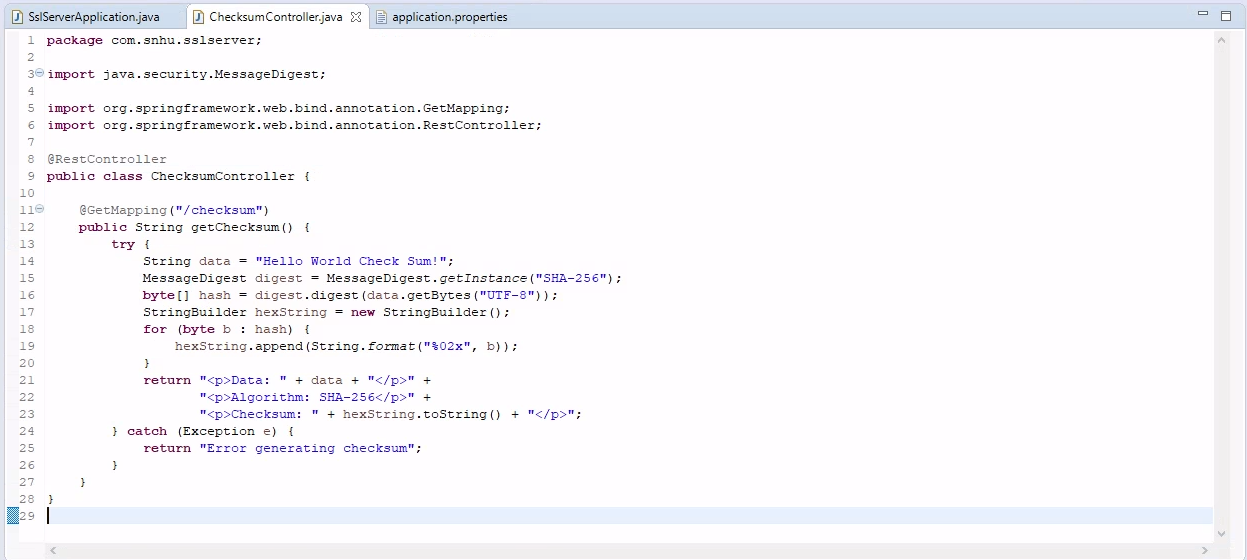
Insert a screenshot below of the web browser that shows a secure webpage.



## Secondary Testing

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

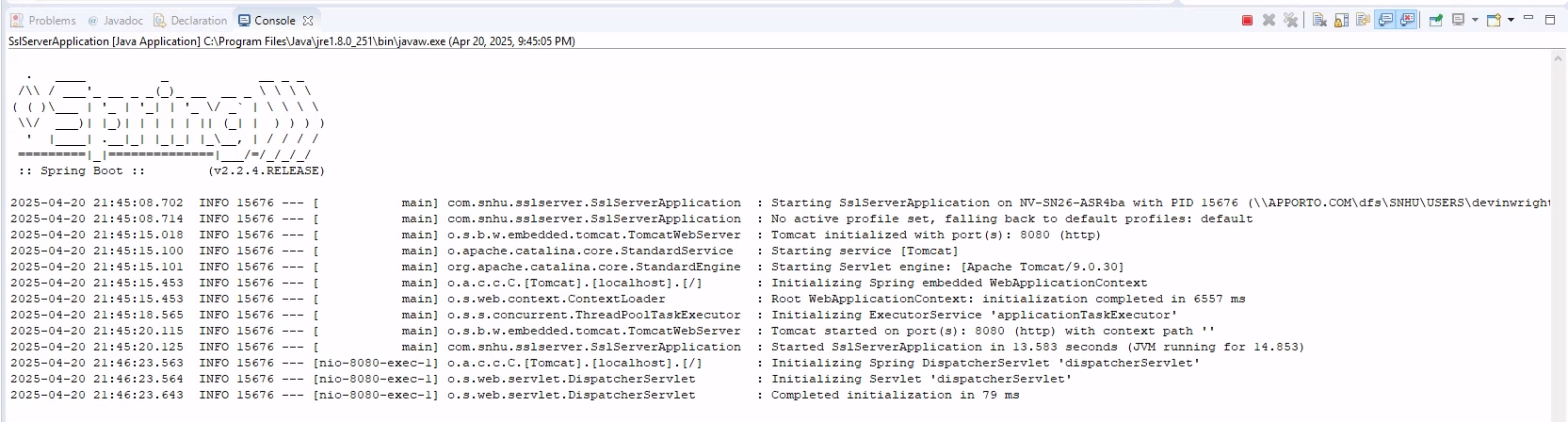






## Functional Testing

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



## Summary

No errors appeared during code execution after implementing the checksum generation functionality. The secure communication configuration that was supposed to operate HTTPS on port 8443 changed to HTTP on port 8080 because of problems with the SSL keystore. The software update to be released will address problems with keystore path configuration. The vulnerability assessment revealed multiple outdated dependencies which were resolved by eliminating false positives through a suppression.xml file.

## Industry Standard Best Practices

The selection of SHA-256 as the hashing algorithm resulted from its recognition as a secure industry-standard cipher.

The Spring Boot framework allowed us to develop our RESTful architecture.

Our development team modularized the system in the codebase development stage while maintaining separation practices which facilitated easier refactoring and testing.

The OWASP Dependency-Check tool analyzed security dependencies and produced documentation.

The organization implemented security measures around known vulnerabilities to stop any potential exploitation.

The next stage of our development work will focus on integrating valid security certificates to enable HTTPS.