****

# 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** | **2/5/2023** | **Brian Bentley** | **Initial document creation.** |

## 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

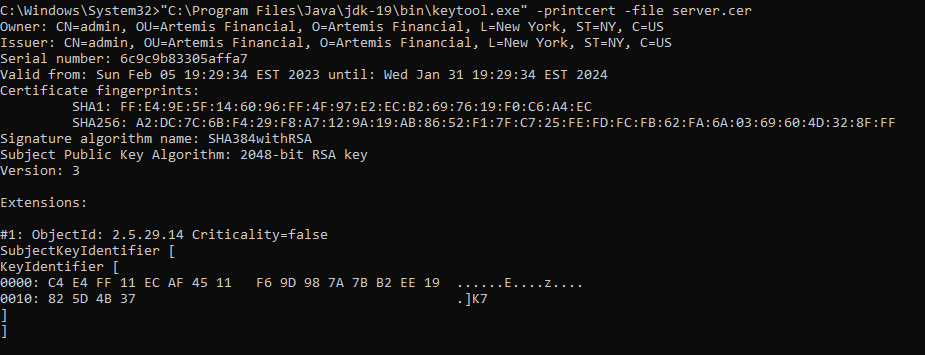
* Brian Bentley

## Algorithm Cipher

The cipher that we will be using for the algorithm will be the SHA-256 cipher. We chose this cipher since it will have the least chance of “collision” attacks occurring. Although this cipher is not collision proof, it is as collision resistant as they come. SHA-256 is a cryptographic hash function. It is built with the Merkle-Damgard construction from a compression function built by a specific specialized block cipher. 256 is a hash value that The Secure Hash Algorithm digests, these are computed with eight, 32-bit and 64-bit words. Although they use different shift amounts and initial values their structures are identical besides the amount of rounds the cipher performs. For informational purposes symmetric encryption uses the same secret key to encrypt and decrypt data. Asymmetric encryption uses a public key and a mathematically linked public and private key pairs to encrypt and decrypt data. SHA is a symmetric encryption algorithm.

## Certificate Generation

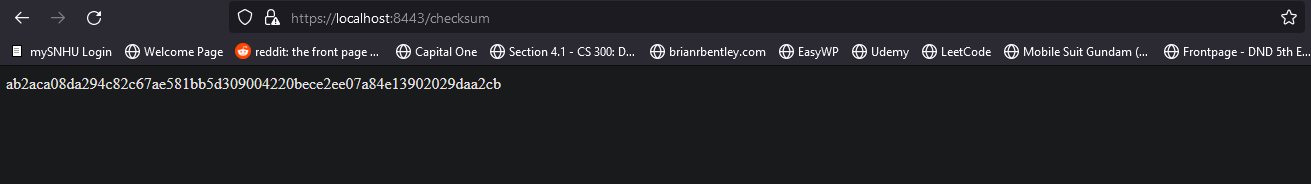
Insert a screenshot below of the CER file.



The above image is of our certificate generation.

## Deploy Cipher

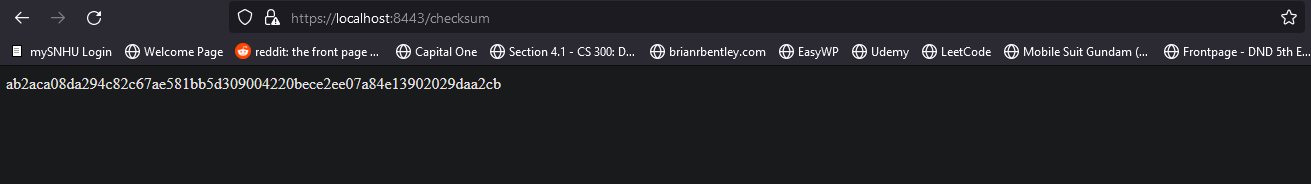
Insert a screenshot below of the checksum verification.



The above image is of our websites checksum

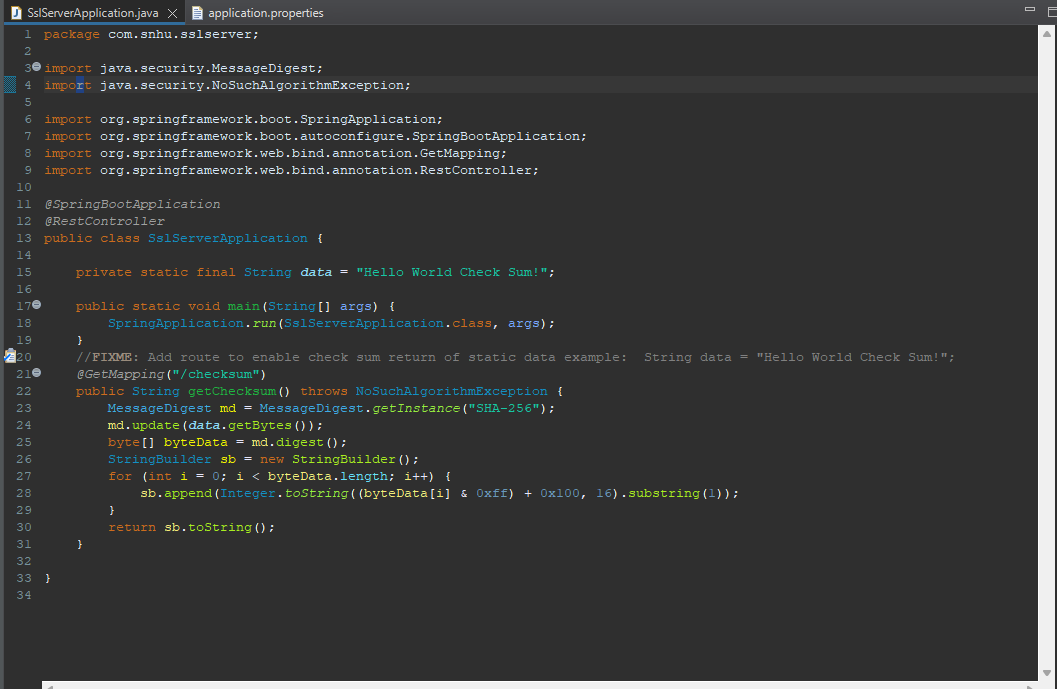
## Secure Communications

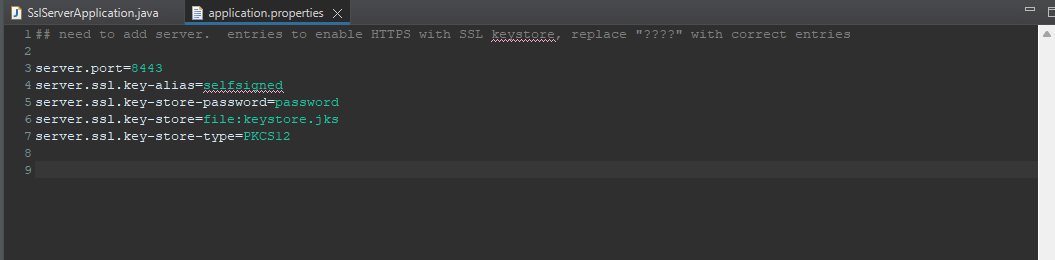
Insert a screenshot below of the web browser that shows a secure webpage.



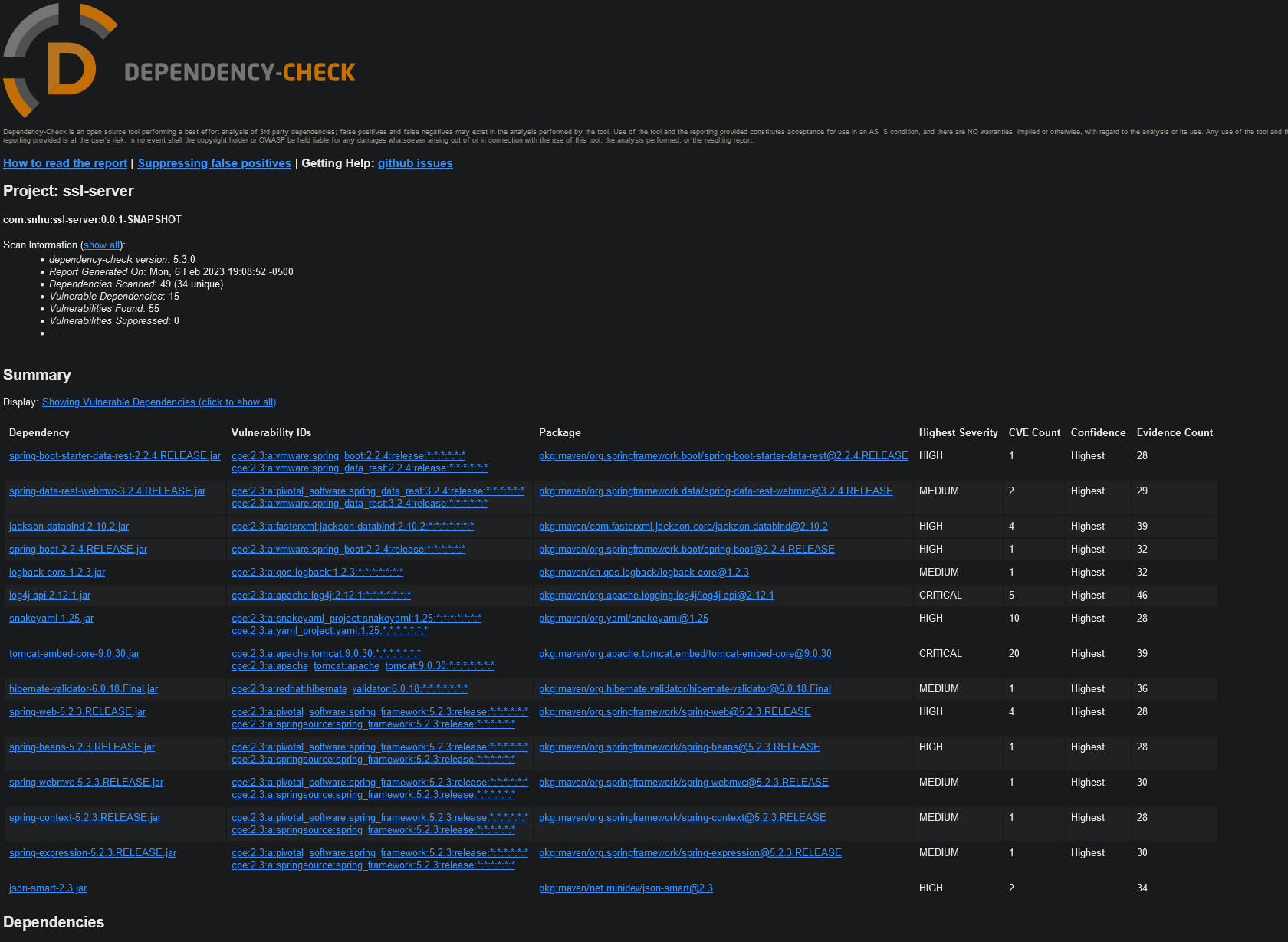
## Secondary Testing

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





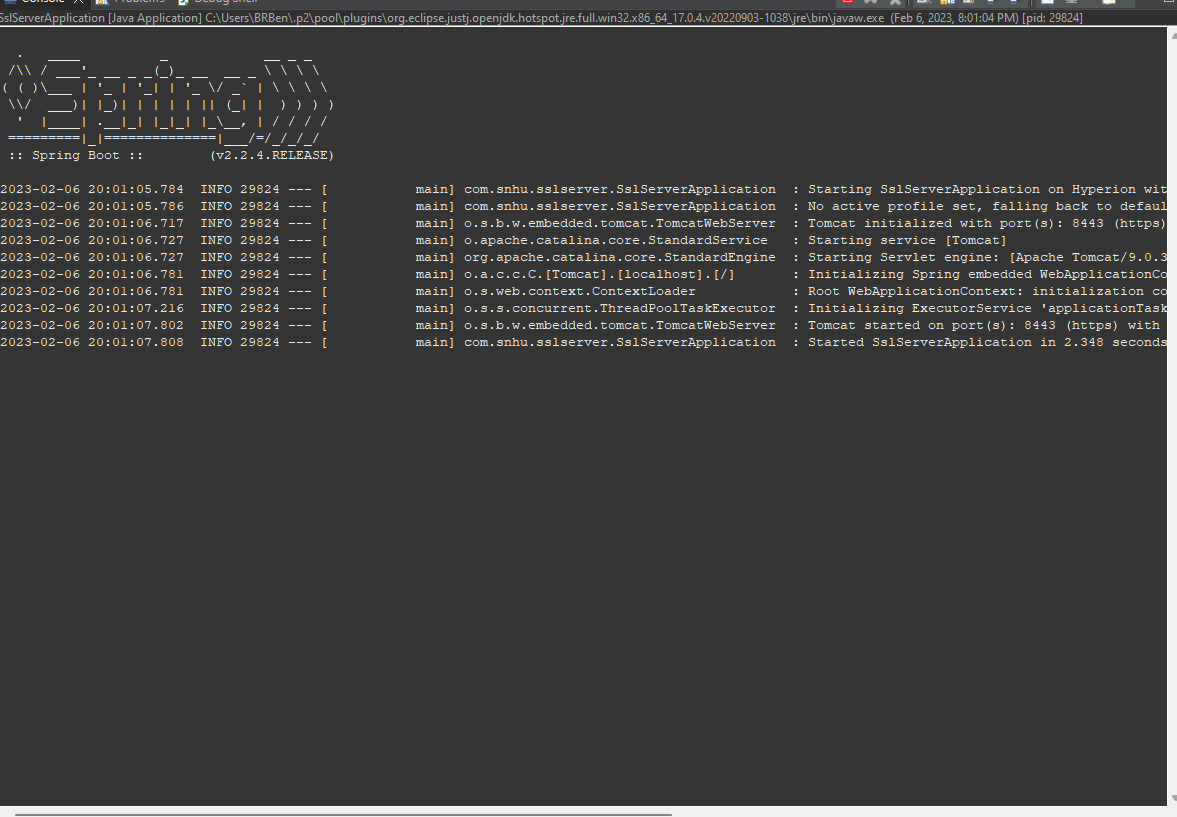
The above image is of our refactored code. Both the checksum method as well as the edits made to the application.properties to point to the correct key store and key alias.



The above is a dependency check showing vulnerabilities. Here you can see that we have many valid vulnerabilities and should update the range of libraries in our application.

## Functional Testing

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



The above is an image of the server application starting correctly.

## Summary

By following the vulnerability assessment process flow diagram, we have hit a few points here to ensure that our software is more secure. We are using secure API interactions and code quality by refactoring our code in the springboot framework. We also utilized cryptographic encryption with our inclusion of SHA-256 to our program. We also ensured that our client/server interactions were secure by building and deploying java server certificates with the java keygen tool. Then another example of crypthography would be ensuring that our website properly displayed a checksum. We also tested our application with maven to run a dependency check which satisfies code quality and input validation in some cases. However, our test does not really show any false positives and we recommend updating the projects code base.

## Industry Standard Best Practices

We maintained industry standard best practices with the applications current security in many ways. One important way is be publishing server certificates for our application, another is properly deploying a checksum check into our applications website. We also utilized secure code in the way that we refactored our code. We also highly recommend updating the dependencies throughout the project as the many vulnerabilities found in this code base are from outdated dependencies.

Following industry standard best practices will ensure that our software is secure and our customers and developers can trust us with their information and business. We also ensure that there are not mishaps from our software being breached and having disastrous unintended consequences.