****

# 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 23, 2023** | **Marye Bierbaum** |  |

## 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

Marye Bierbaum

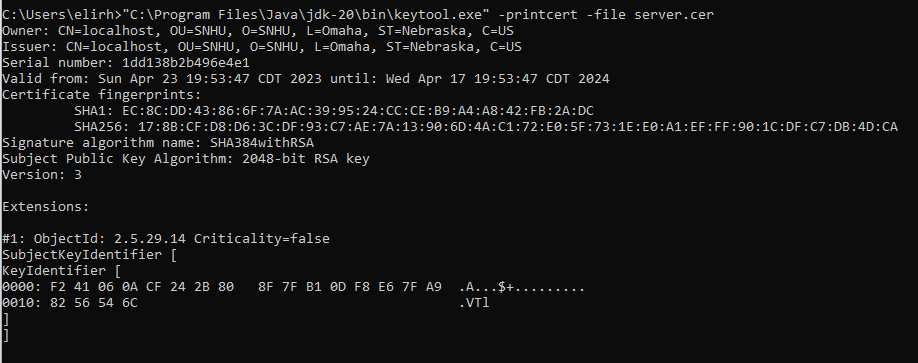
## Algorithm Cipher

Because Artemis Financial will be using this software to transfer important and confidential financial information to their customers, it is of the utmost importance that that data is secure and cannot be altered or stolen. With this in mind, I suggest implementing an SHA 256 algorithm cipher into the code.

The SHA 256 algorithm is currently the world's strongest cryptographic hash and has yet to be hacked, which is why it is recommended for use by the US Government. These types of algorithms work by scrambling any size of string in a specific way to produce a hash that has a fixed length and cannot be worked backwards to determine the original string. The ‘256’ in the name of the cipher refers to the number of bits the resulting hash will have and also to the sheer number of unique hashes that can be generated (2256). Though that number is finite, it is so large that the likely hood that two different inputs would generate the same hash is extremely small, almost impossible. It also means that a hacker attempting to brute force their way into the data by just guessing the correct hash would have to spend multiple lifetimes doing so. Implementing the SHA 256 algorithm into the Artemis Financial project should be our first step in securing the application.

## Certificate Generation

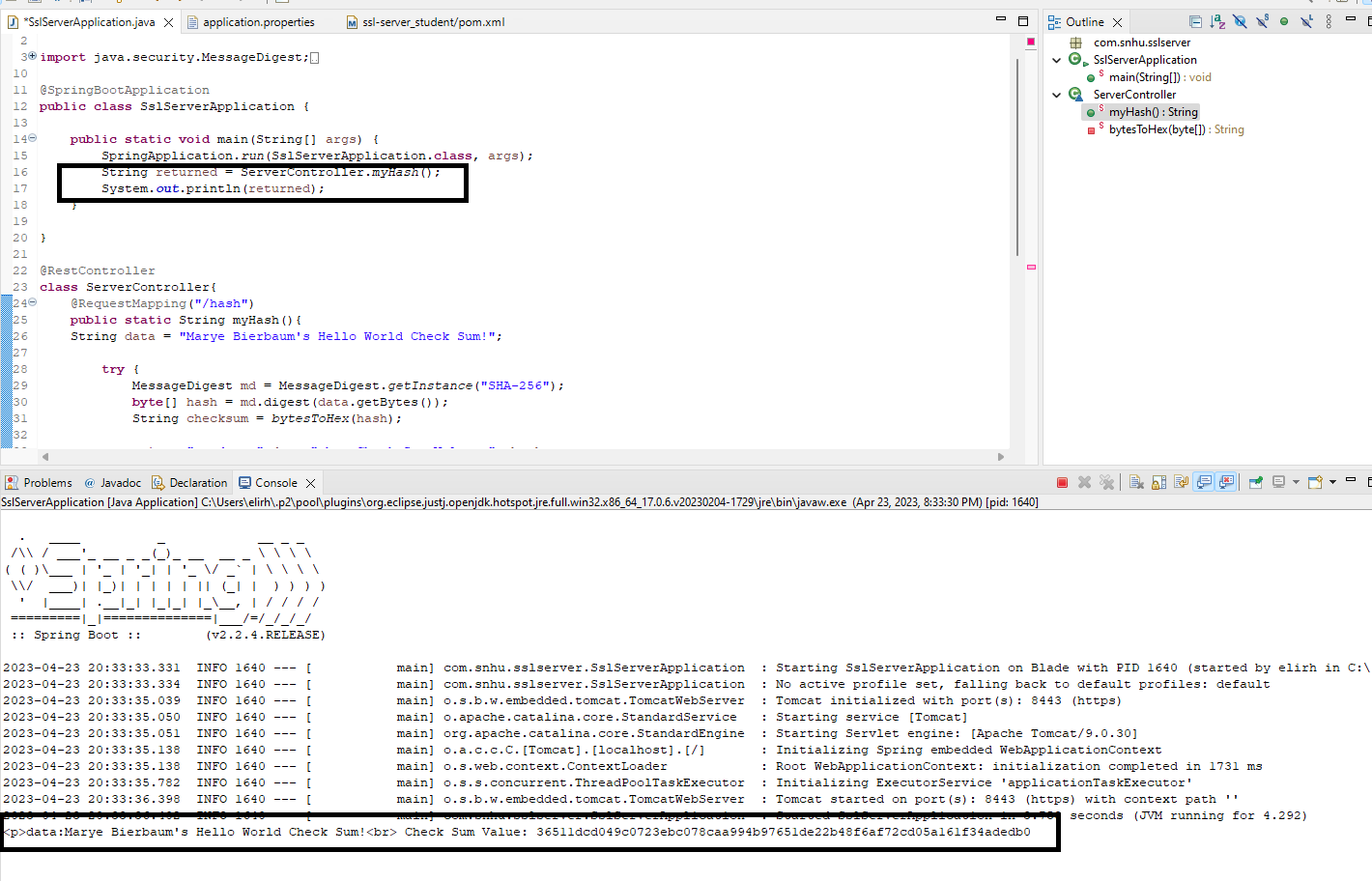
Insert a screenshot below of the CER file.



## Deploy Cipher

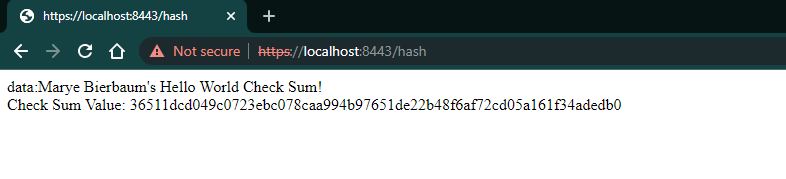
Insert a screenshot below of the checksum verification.

To verify the checksum is working, I added a print line to the main function of the application so that it would print in the console on execution. See the boxed areas below:



## Secure Communications

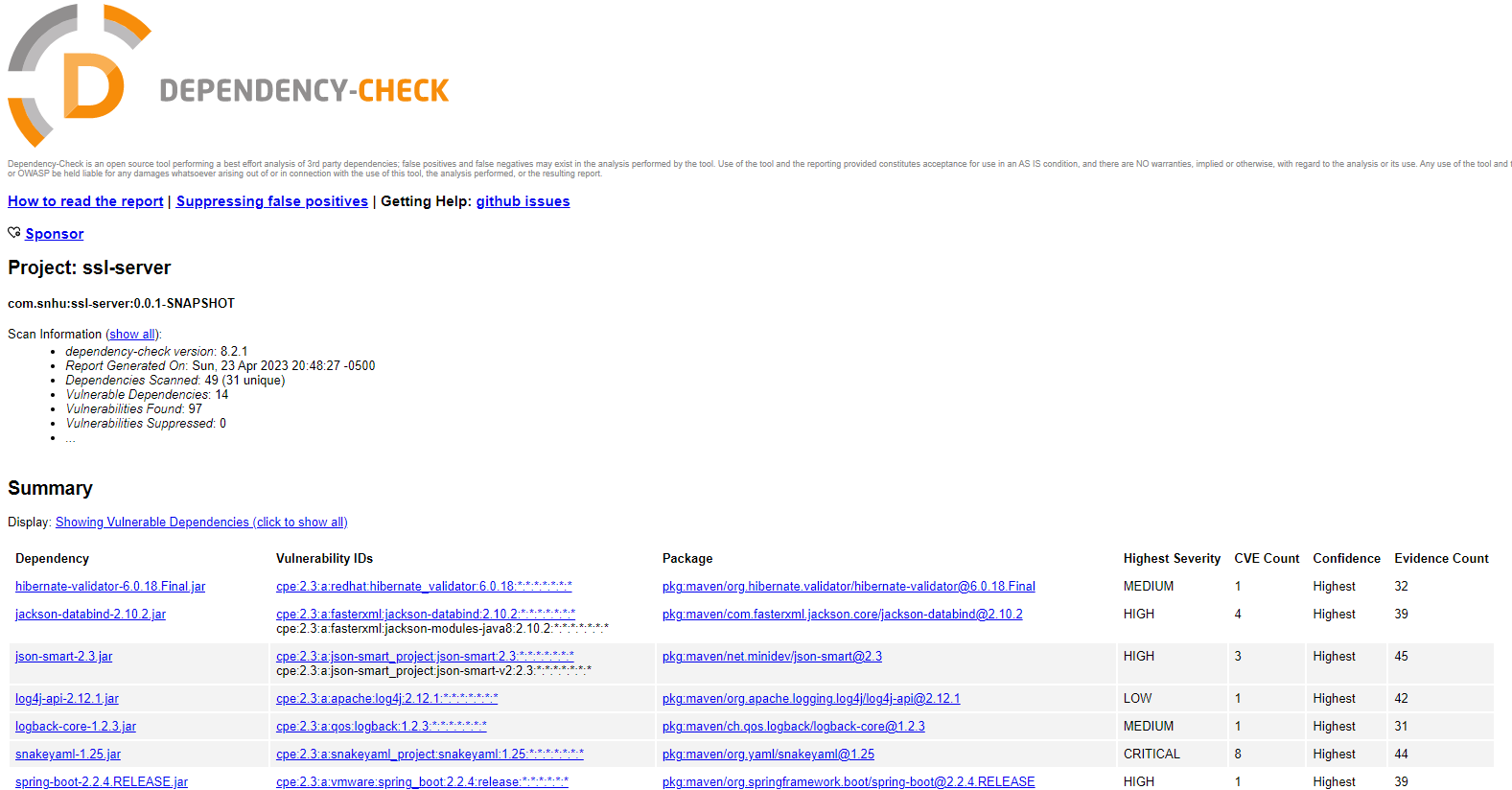
Insert a screenshot below of the web browser that shows a secure webpage.



## Secondary Testing

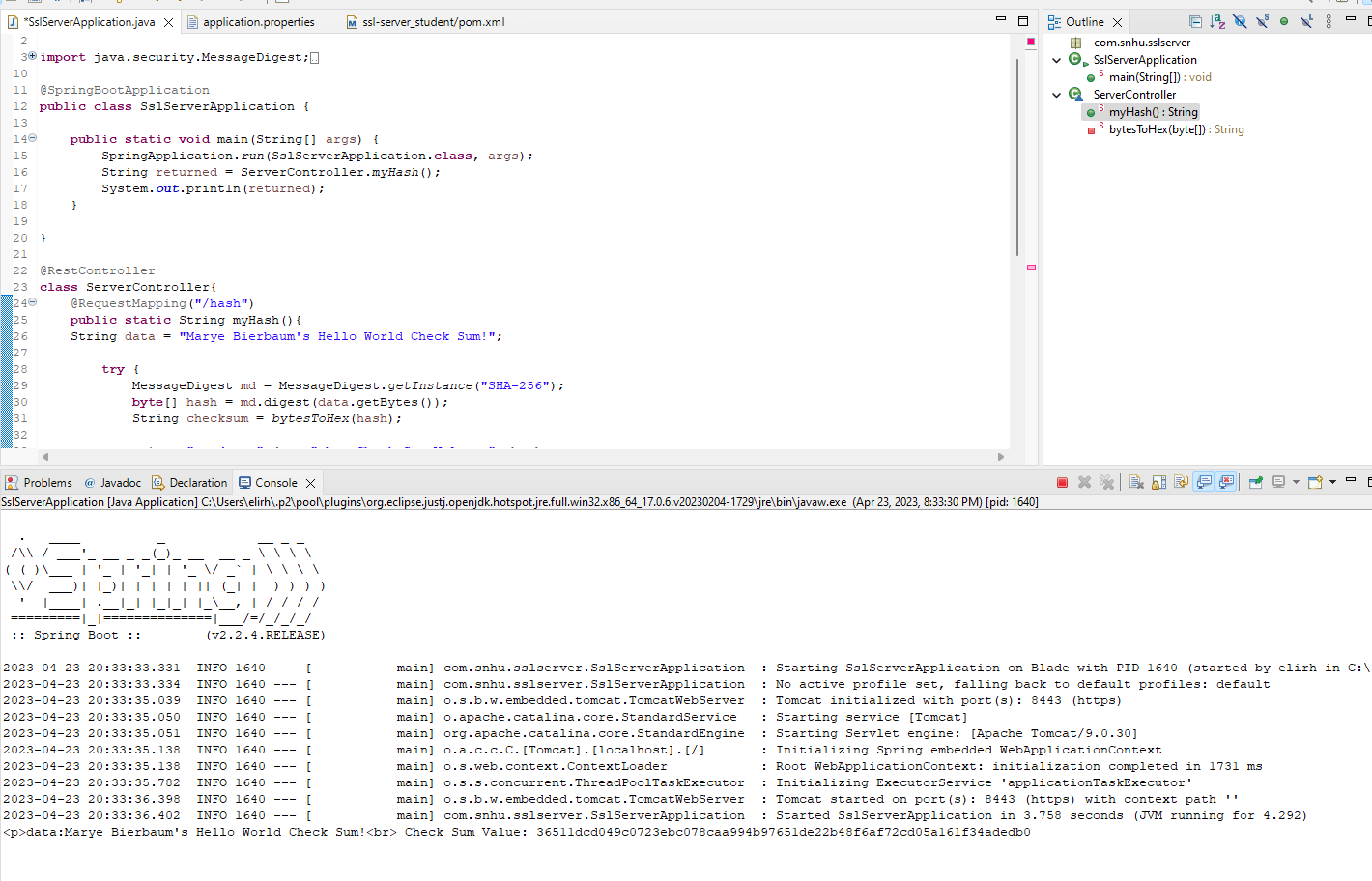
Insert screenshots below of the refactored code executed without errors and the dependency-check report.

I compared the before and after reports and verified no new dependencies were added after my additions.



## Functional Testing

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



## Summary

My review and refactoring of this code base included adding a hash function, enabling the application to run on HTTPS, and verifying that my changes created no new vulnerabilities or concerns.

The first change I implemented was to add a hash function. Hash functions take secure data (like the reports compiled by Artemis Financial and passed to their clients) and translate it to an unrecognizable string. I built the hash function to utilize the SHA-256 cipher algorithm, the most secure cipher currently available. See the first section of this report for additional information on hash functions and the ciphers behind them.

Hash functions are an important part of software security, but the most important change I made was in the application.properties file and it enabled the website to run on HTTPS, which encrypts the data that is sent and received by the application. When using regular HTTP, information sent is stored in plain text and can easily be found. HTTPS protects against this by making the information difficult to decipher.

In order to use HTTPS I had to add a self-signed certificate to the project. Certificates are used to verify that websites are being managed and run by who they say they are. Most major websites have certificates that are verified by third-party Certificate Authorities (CA). Our devices look for these certificates to ensure we aren’t tricked into using websites that are being run maliciously, like if a hacker tried to recreate a bank website in order to steal login credentials. Because we do not yet have a certificate for this project for a CA, I created a self-signed certificate and added it. This tells HTTPS that our website is safe and can be accessed by users.

Finally, I made sure that my changes had not generated any additional vulnerabilities in the dependency report.

## Industry Standard Best Practices

The changes that I have made align with industry standards. For instance, the SHA-256 cipher algorithm is the most secure to-date and is recommended by use by the US Government and a number of security organizations. Utilizing HTTPS is another industry standard. In fact, most modern web browsers will warn users if they are trying to access a site that is not HTTPS.

The final step we need to take to get this project ready to ship is to generate certificates with our chosen CA. Self-signed certificates work great while developing and testing in-house, but we really need to get an official certificate so that our clients can be assured we are operating at the highest level of security.