****

# Practices for Secure Software Report

**Table of Contents**

[**Document Revision History 3**](#_heading=h.30j0zll)

[**Client 3**](#_heading=h.1fob9te)

[**Instructions 3**](#_heading=h.3znysh7)

[**Developer 4**](#_heading=h.2et92p0)

[**1. Algorithm Cipher 4**](#_heading=h.tyjcwt)

[**2. Certificate Generation 4**](#_heading=h.3dy6vkm)

[**3. Deploy Cipher 4**](#_heading=h.1t3h5sf)

[**4. Secure Communications 4**](#_heading=h.4d34og8)

[**5. Secondary Testing 4**](#_heading=h.2s8eyo1)

[**6. Functional Testing 4**](#_heading=h.17dp8vu)

[**7. Summary 4**](#_heading=h.3rdcrjn)

[**8. Industry Standard Best Practices 4**](#_heading=h.26in1rg)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **20250218** | **Alexander Peck** | **Initial** |

## 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.

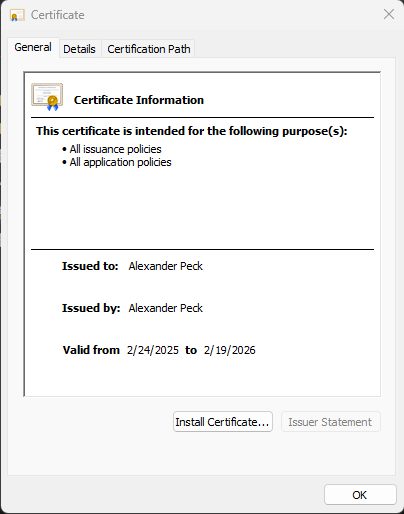
## Developer

Alexander Peck

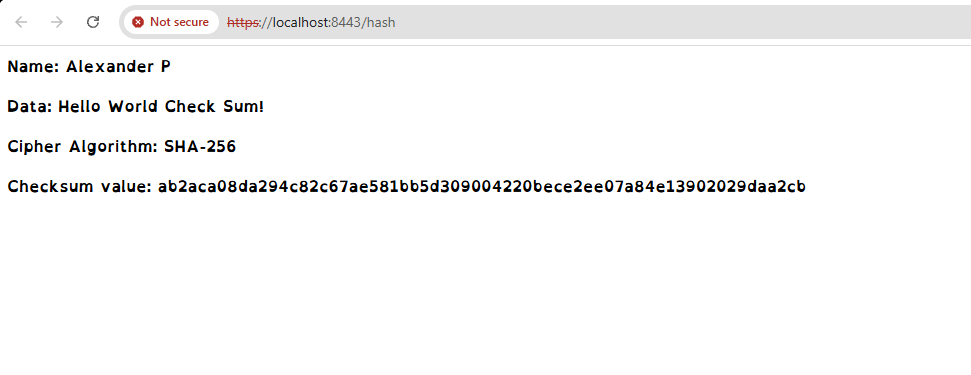
## Algorithm Cipher

Our client, Artemis Financial, requests our assistance ensuring secure communication between their web API and back end. Specifically, they asked us to help implement a secure hashing algorithm to verify the transferred data. For this task, I recommend implementing **SHA-256 with RSA**. SHA-256 is part of a family of cryptographic hash functions. It takes input messages and produces a fixed-length hash. Hashes are deterministic, meaning that the same input into a specific SHA will always make the same output. RSA is an asymmetric encryption and digital signature algorithm. Encryption algorithms typically fall into two categories: symmetric and asymmetric encryption. Symmetric encryptions use a single key for both encryption and decryption. This differs from asymmetric algorithms that use two separate keys to encrypt and decrypt data. Using both SHA and RSA together, we can implement a form of a checksum and ensure that the data makes it to its destination without any tempering.

## Certificate Generation

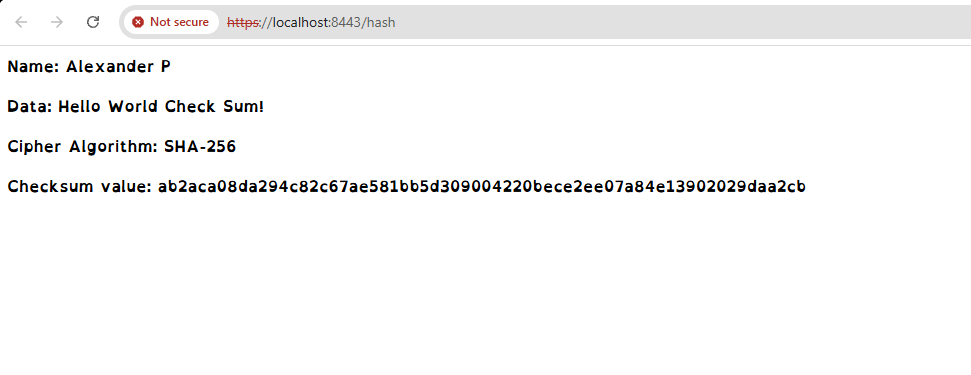


## Deploy Cipher



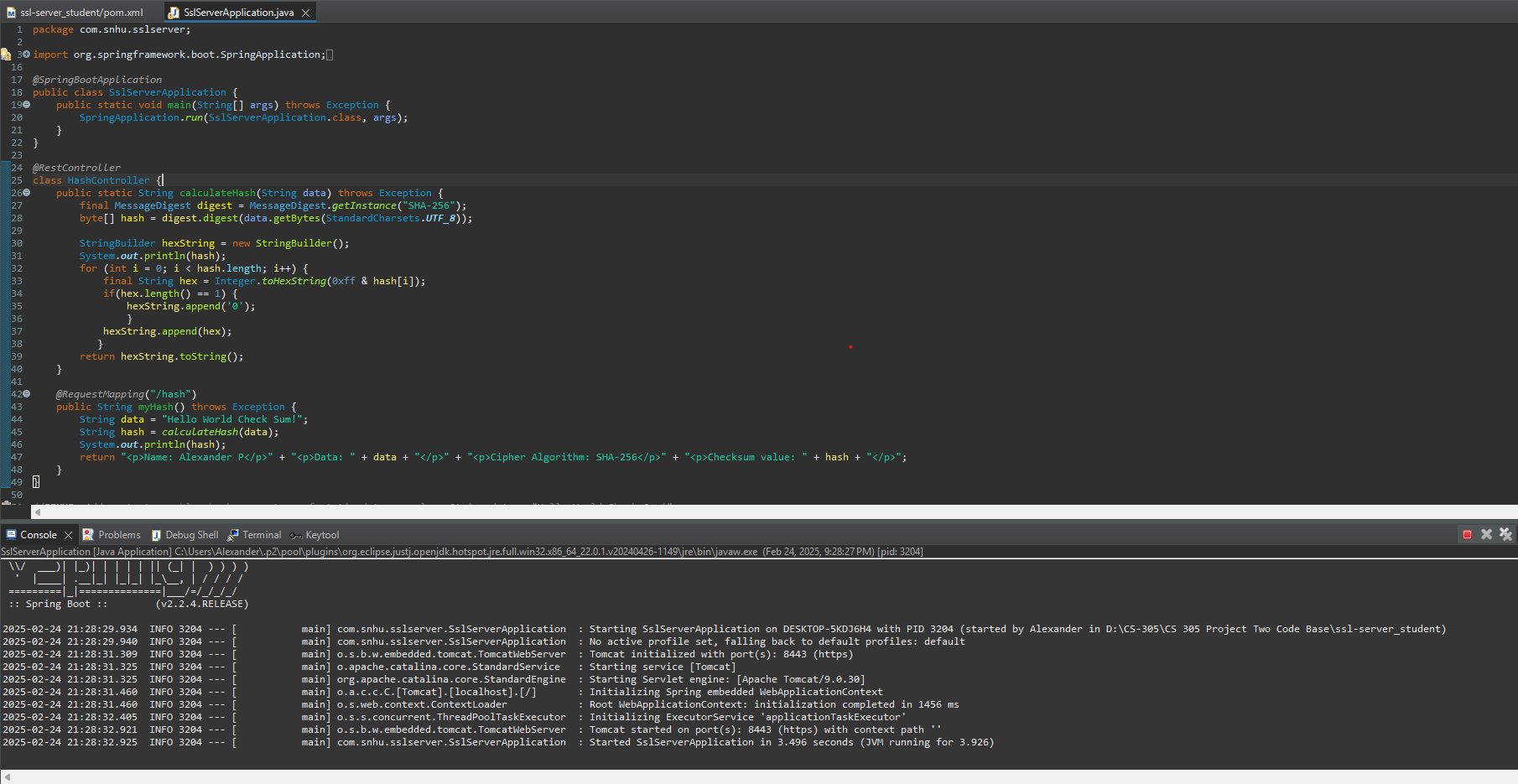
## Secure Communications

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

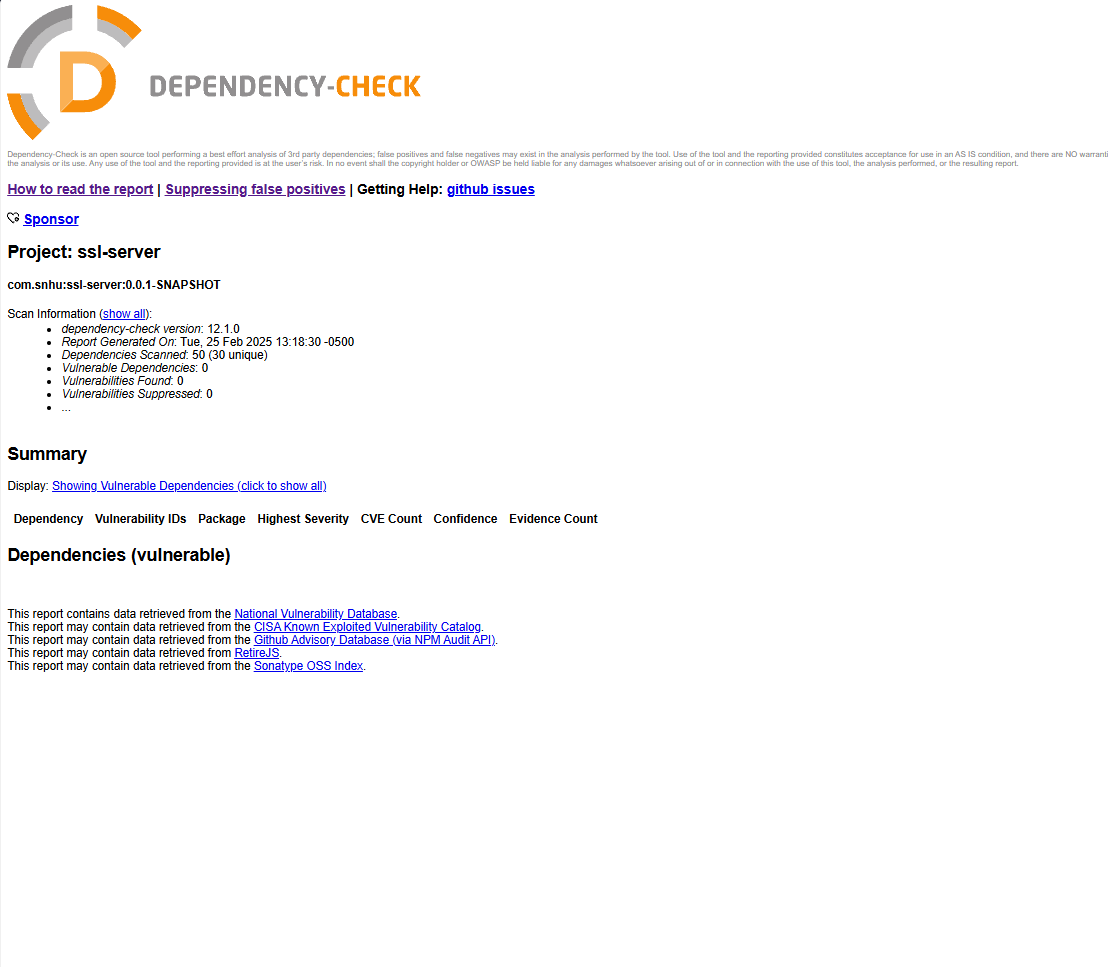
Insert a screenshot below of the refactored code executed without errors.



**Before Update**

## 

**After Update**



## Summary

I aimed to implement a secure cipher algorithm to ensure the data transferred is valid. To do this, a SHA-256 cipher was implemented to hash our data for verification upon being received by the expected party. The SHA cipher is implemented using the method calculateHash. I also generated a certificate using the RSA algorithm. This cert verifies a secure connection between the browser and the backend. During this project, I could not provide a secure connection through SSL. I verified the **application.properties** file to ensure everything on the cert was correct; however, I was still unable to secure the connection.

## Industry Standard Best Practices

I manually reviewed the code before performing any test or implementing any code. I found that there wasn’t a cipher or encryption algorithm implemented. After implementing both, I reviewed the pom file to implement a dependency check. The dependency check was updated to its latest version,**12.1.0.** This dependency report pulled multiple dependencies that were identified with known vulnerabilities. These dependencies were then updated to their latest version, and the check was rerun. Rerunning the check showed that there are no vulnerabilities present in the program.