

# 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** | **2/22/25** | **Shawn Millin** | **Creating secure communications for client** |

## 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

Shawn Millin

## Algorithm Cipher

The algorithm cipher used to refactor the code for Artemis Financial is the SHA-256 cipher (Secure Hash Algorithm 256-bit). This cipher is a cryptographic hash function and resistant to collisions.

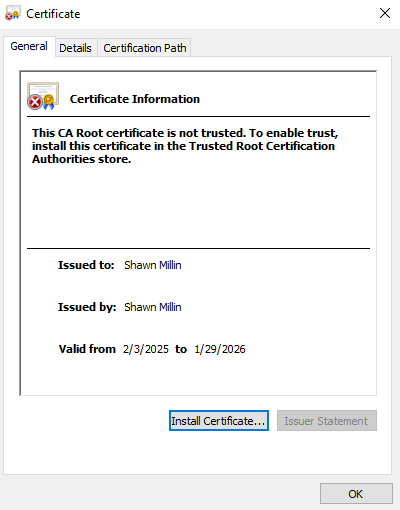
SHA-256 will take an input, no matter the length, and output a hexadecimal string of a fixed length of 256 bits with no way of reverse engineering the output.

Explain the use of random numbers, symmetric versus non-symmetric keys, and so on.

The family of SHA ciphers were designed by the US Government’s National Security Agency (NSA) and published by the National Institute of Standards and Technology (NIST) in 2001. Currently the SHA-256 cipher is used for Bitcoin and other cryptocurrencies, Transport Layer Security(TLS) and Secure Socket Layers(SSL), software distribution, digital signatures, data integrity and verification. In the Future, quantum computers will bring more of a challenge to this cipher but for now it is still a strong cipher to use.

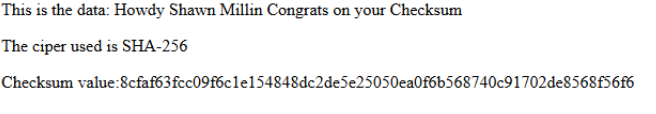
## Certificate Generation

Insert a screenshot below of the CER file.



## Deploy Cipher

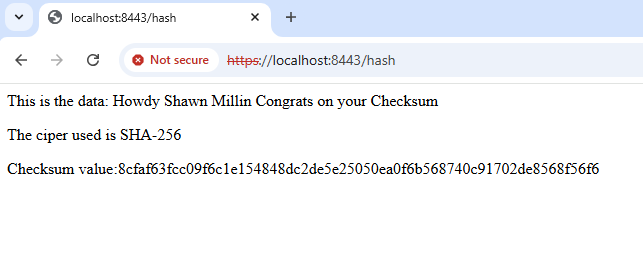
Insert a screenshot below of the checksum verification.



## Secure Communications

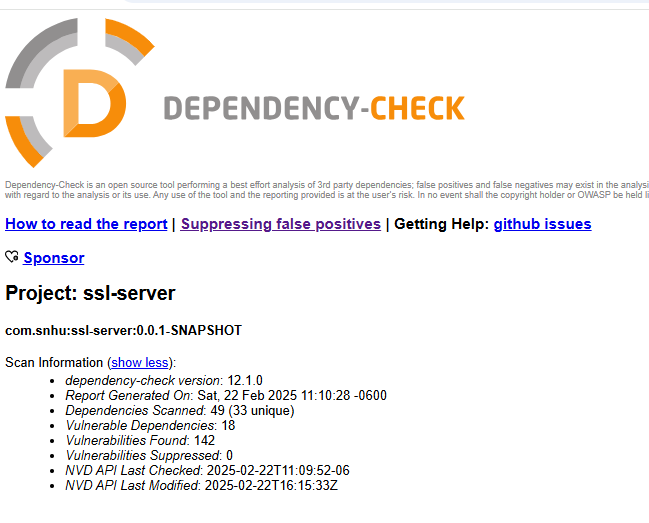
Insert a screenshot below of the web browser that shows a secure webpage.

The self signed certificate prevents a secure website.

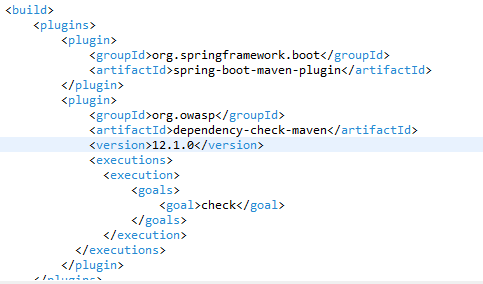


## Secondary Testing

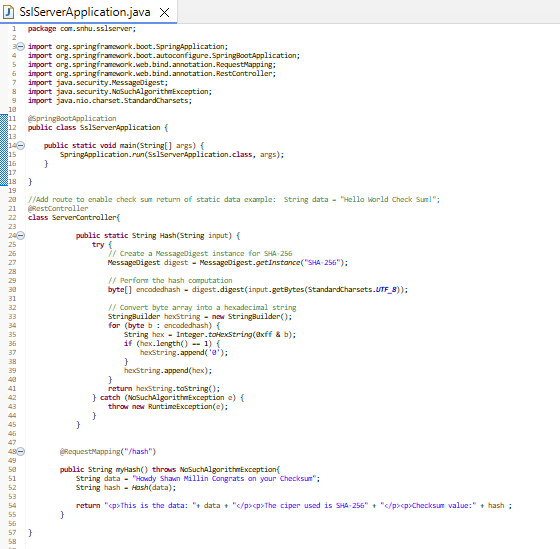
Insert screenshots below of the refactored code executed without errors and the dependency-check report.



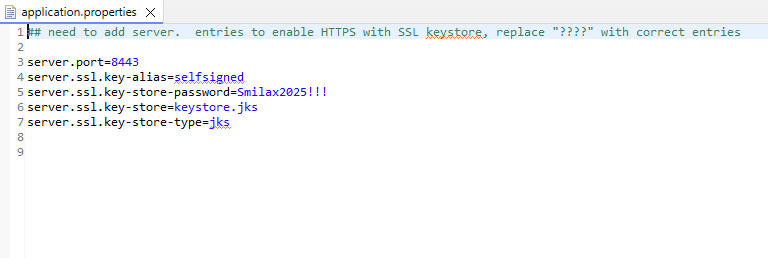
Dependency Check updated to current version of 12.1.0



Refactored code with no errors



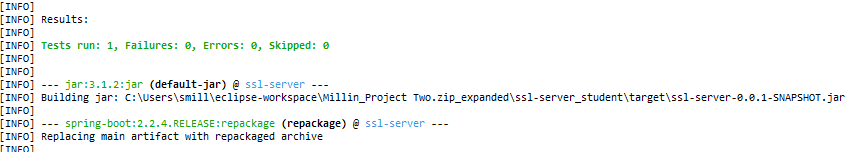
Certificate information updated



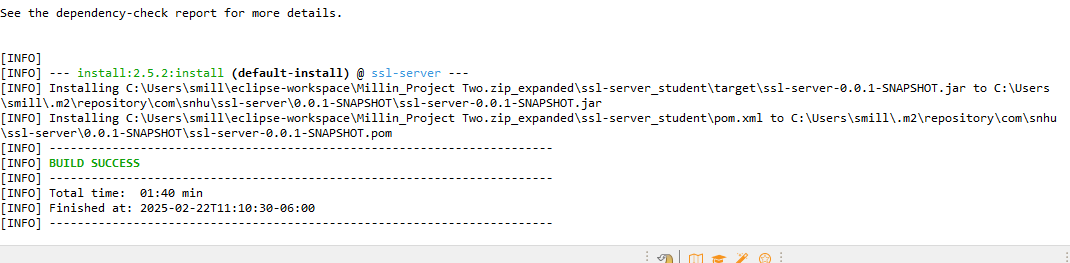
## Functional Testing

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

After running Maven build, no errors occurred



The Maven build was successful



## Summary

From the vulnerability assessment process flow chart, the code has been refactored to reflect the categories of Cryptography, Client/Server, Code Error, and Secure API. In the given code, a REST controller was added (@RestController) as part of the Spring MVC controller to create a RESTful web service. This annotation was placed above the ServerController class. A certificate was generated and added to the project via a cer file and added to the code via the application.properties file for verification and to add HTTPS. In the ServerController class the SHA-256 cipher was added to encrypt input data into a hexadecimal output by using the MessageDigest java security class. The input in myHash() method was mapped to using @RequestMapping annotation to use the input in the SHA-256 cipher in ServerController class.

Code Error handling was implemented with a try catch block and using the NoSuchAlgorithmException java security class to throw an exception if a specific crypotgraphic algorithm is not found.

## Industry Standard Best Practices

Industry Sandards Best Practices were followed by utilizing the OWASP dependency check. The pom.xml file was updated with latest version of the dependency check, version 12.1.0 to catch any known vulnerabilities to date. Practices were also followed by using a Certificate Authority for verification in the secure webpage. Encrypting the data and handling the code errors are also part of following Industry Standard Best Practices.