****

# 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 14, 2023** | **Cody Faircloth** |  |

## 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

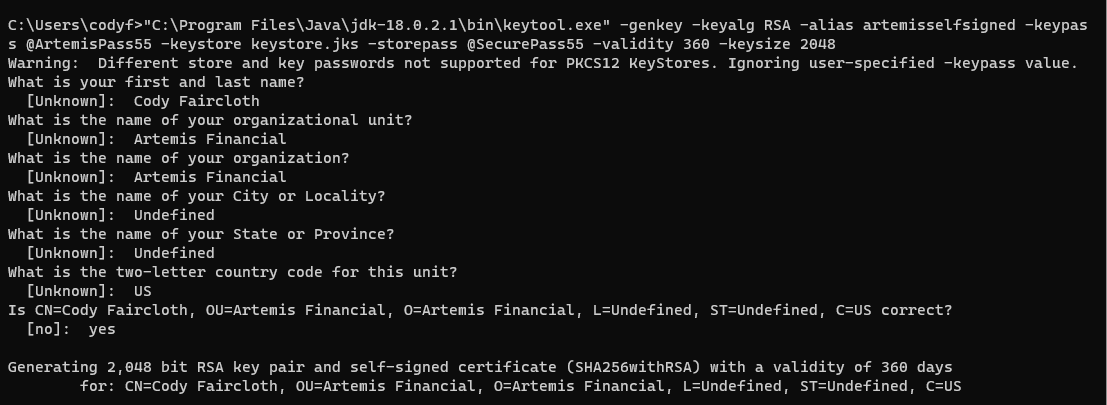
Cody Faircloth

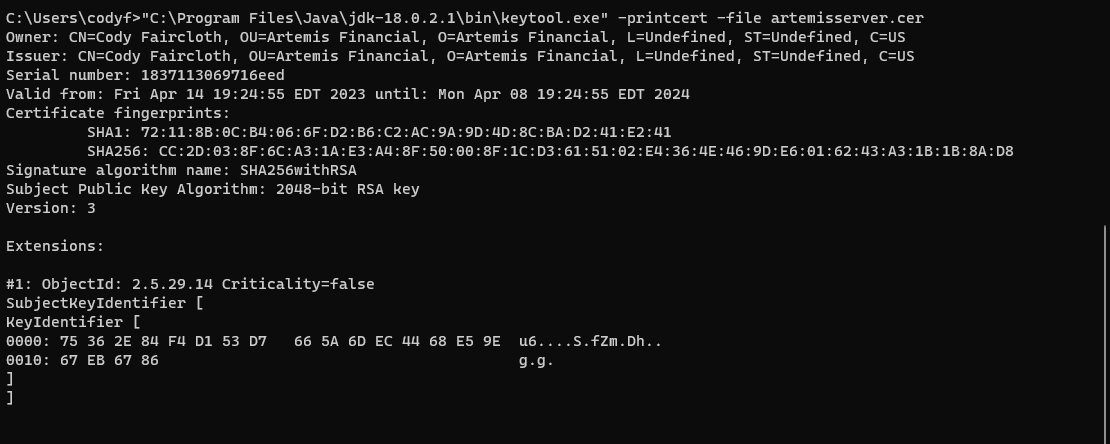
## Algorithm Cipher

Artemis Financial is requesting an encryption algorithm for the encryption and archive of long-term files. Since these files will not be in transit SHA-256 would be the recommended encryption cipher. SHA-256 is a symmetric key cipher meaning that a single key is created which will be kept by Artemis Financial. The key will be created through the use of Java’s random number generator using the system’s entropy as the seed for the random number generator. The data will then be hashed based on the random number generator’s output, encrypting the data. SHA-256 encrypts using a 256-bit level providing the highest level of security for brute force attacks. Since these long-term files will not be in transit asymmetric keys will not be implemented. The vulnerability of a symmetric key is that the key could be stolen but since the key will not be transferred outside of the local network of Artemis Financial, with the implementation of a secure network, the risk of key theft is minimal.

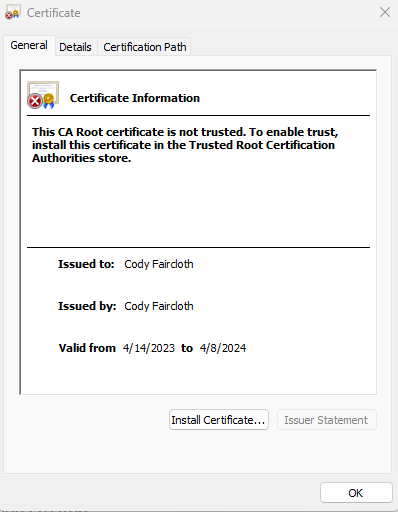
## Certificate Generation

Insert a screenshot below of the CER file.



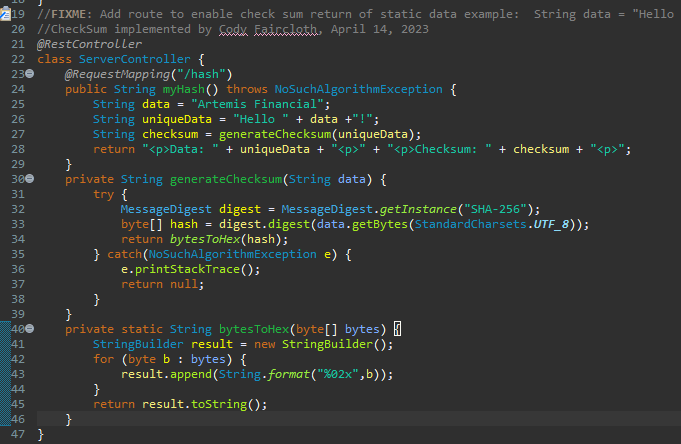


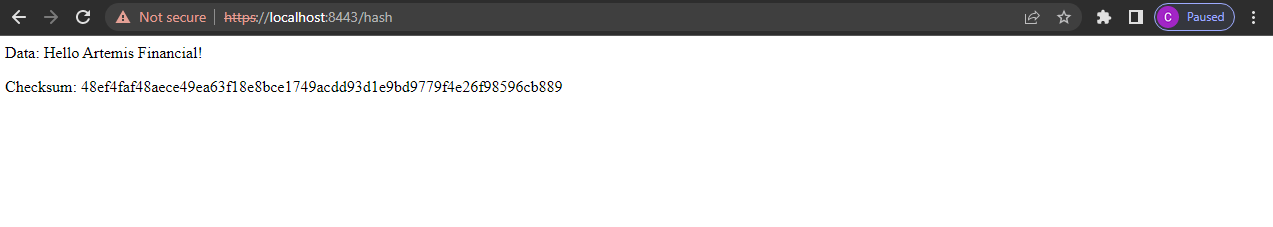




## Deploy Cipher

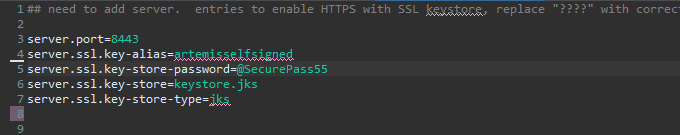
Insert a screenshot below of the checksum verification.

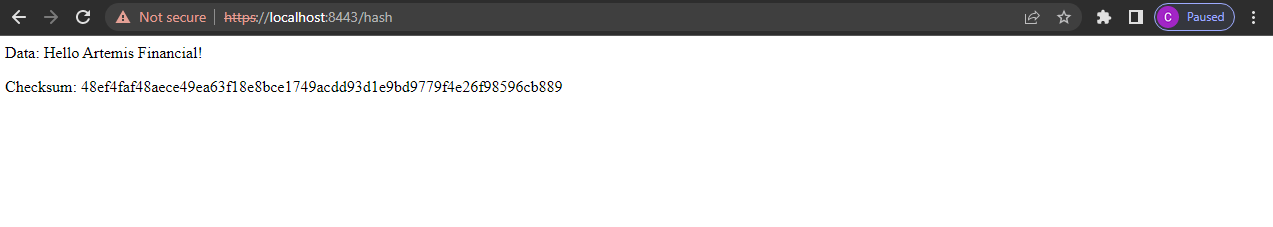




## Secure Communications

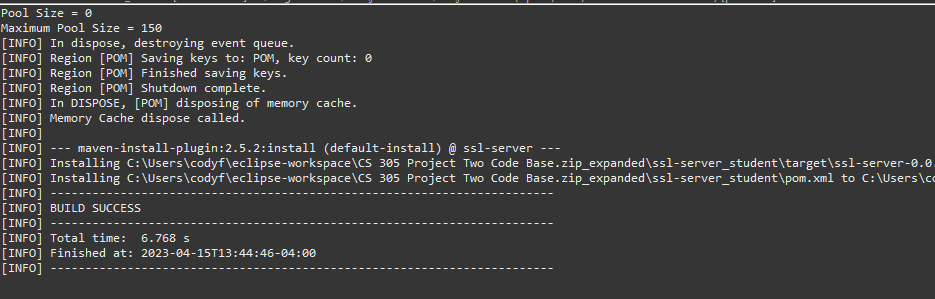
Insert a screenshot below of the web browser that shows a secure webpage.

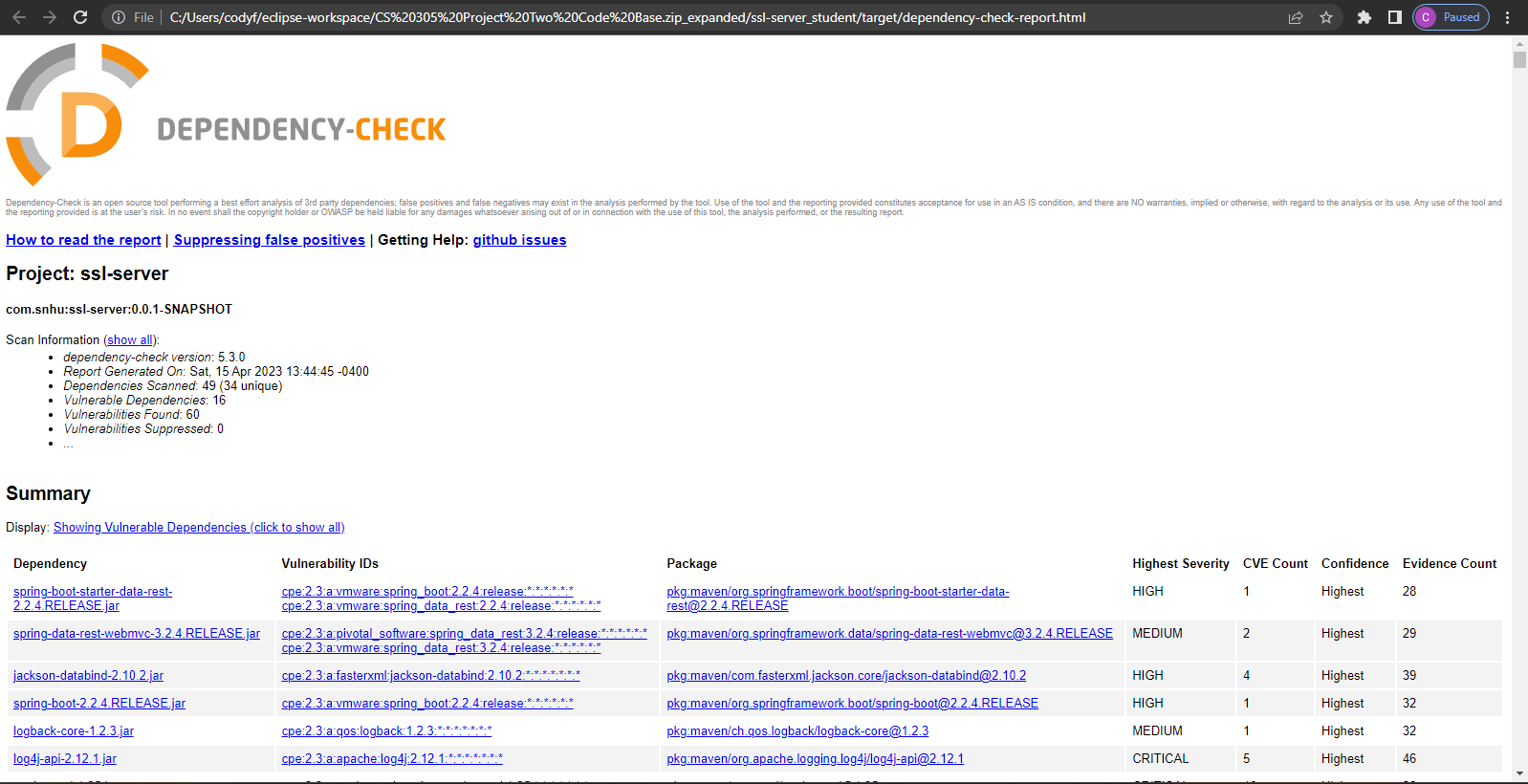




## Secondary Testing

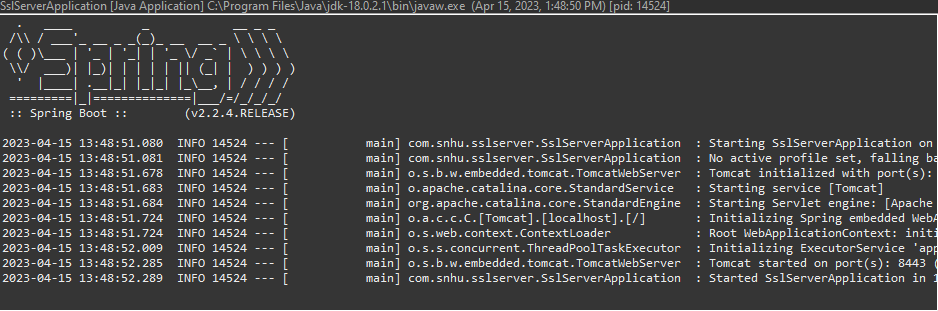
Insert screenshots below of the refactored code executed without errors and the dependency-check report.





## Functional Testing

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



## Summary

I implemented a checksum encryption function as shown above to encrypt data into the local server. This adds a layer of security through encryption as requested by Artemis Financial. Additionally, I implemented a self-signed certificate to convert the HTTP link to an HTTPS link adding an additional layer of security to the application. I checked vulnerabilities through dependencies using the OWASP dependency test and determined no additional vulnerabilities were created through my code modification. The areas of security addressed in this project are APIs, cryptography, client/server, code error, and code quality. APIs were secured through dependency testing. Cryptography was implemented through the inclusion of a checksum function. The client/server was secured through the implementation of HTTPS via a self-signed certificate. Code error was addressed by handling the NoSuchAlgorithmException in the checksum function. Code quality was addressed by using industry best practices when creating the program. My process for adding these layers of security was to implement the requirements in the order they were presented and then test to ensure each requirement was implemented properly. If a requirement was not implemented properly, I would research ways to solve the issue. For example, the server would not host on localhost:8443 at first due to a dependency not being located in the classpath. I added the keystore file to the classpath and the server hosted correctly.

## Industry Standard Best Practices

I used industry standard best practices in several ways. First, I coded using proper syntax. Using proper syntax allows your code to be maintainable and readable to other programmers who may modify the code in the future. Additionally, I implemented industry standard practices when creating certificates. I created my certificate using the data given to me and created a certificate that provided an additional layer of security. Using industry standard best practices is valuable in software development because it keeps all developers on the same standard in regards to security.