****

# 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** | **12/10/2023** | **Tacia Mitchell** |  |

## 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

Tacia Mitchell

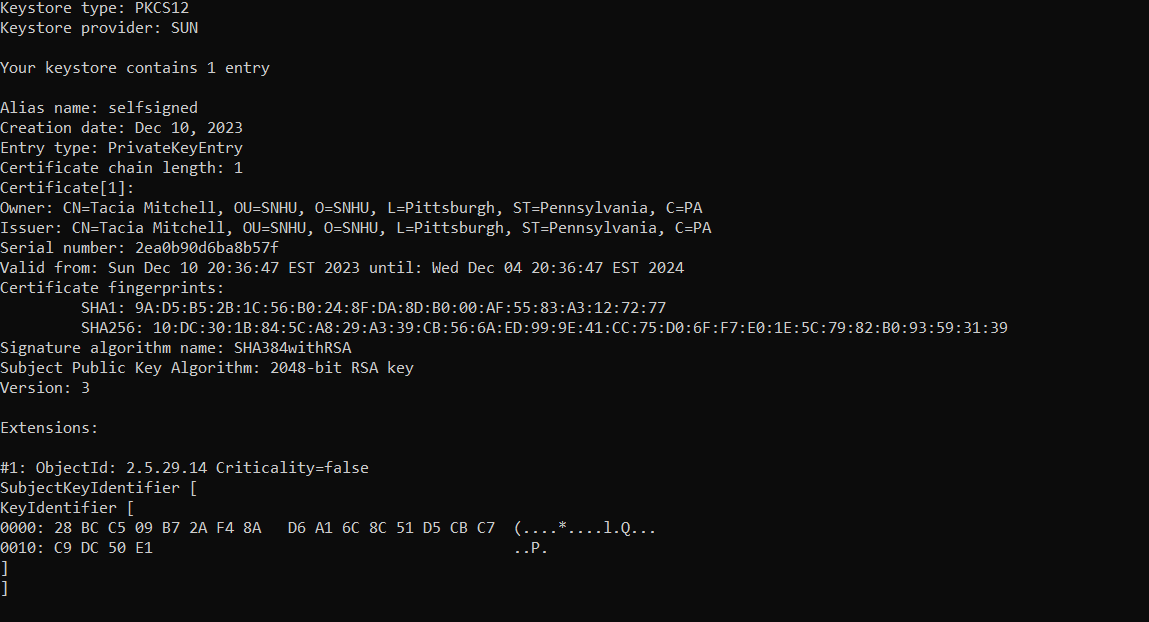
## Algorithm Cipher

The best algorithm cipher to suit Artemis' needs is the SHA-256 of the message digest algorithm. This algorithm produces a fixed-sized (256-bit) hash value, which is usually represented as a hexadecimal number. It changes the input data in a way that is impossible to reverse. It was published in 2001 and is part of the SHA-2 family, which was created to address vulnerabilities and weaknesses in the older SHA-1 version. The 256-bit output of SHA-256 is a hash value of string binary digits, which provides a large and secure address space. Due to the constant file transmissions, Artemis will likely ask us to provide a hash value to verify the integrity of the documents being sent to the clients. The SHA-256 hash is useful in preventing collisions. Since it has a larger output size, it is more resistant to collision attacks, making it difficult to have two different inputs that display the same hash value. [Plate\_number\_1] is widely accepted and supported in many cryptographic applications, making it a reliable choice when creating hash values that can be shared and verified across multiple systems and platforms. SHA-256 is used for password storage that stores a hash password instead of the plain-text password. It also uses blockchain technology which creates a secure and tamper-resistant block.

SHA-256 does not involve keys since it is not used for encryption or decryption and does not use symmetric or asymmetric keys. It primarily operates on a fixed algorithm that does not require the usage of keys. In addition, the SHA-256 cipher does not use random numbers, which means that the same input will always produce the same hash value. SHA-256 continues to remain widely utilized while providing the best security. Brute force attacks are difficult to use against this algorithm due to continuous advancements in computing.

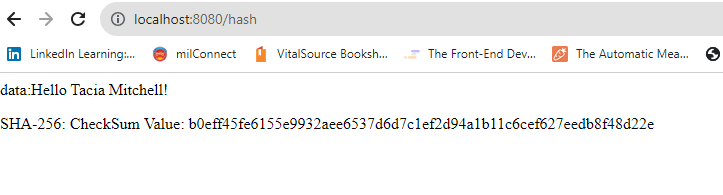
## Certificate Generation

Insert a screenshot below of the CER file.



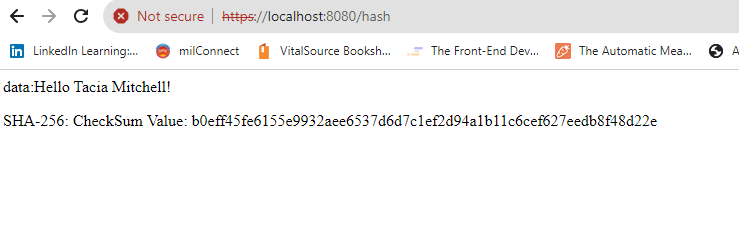
## Deploy Cipher

Insert a screenshot below of the checksum verification.



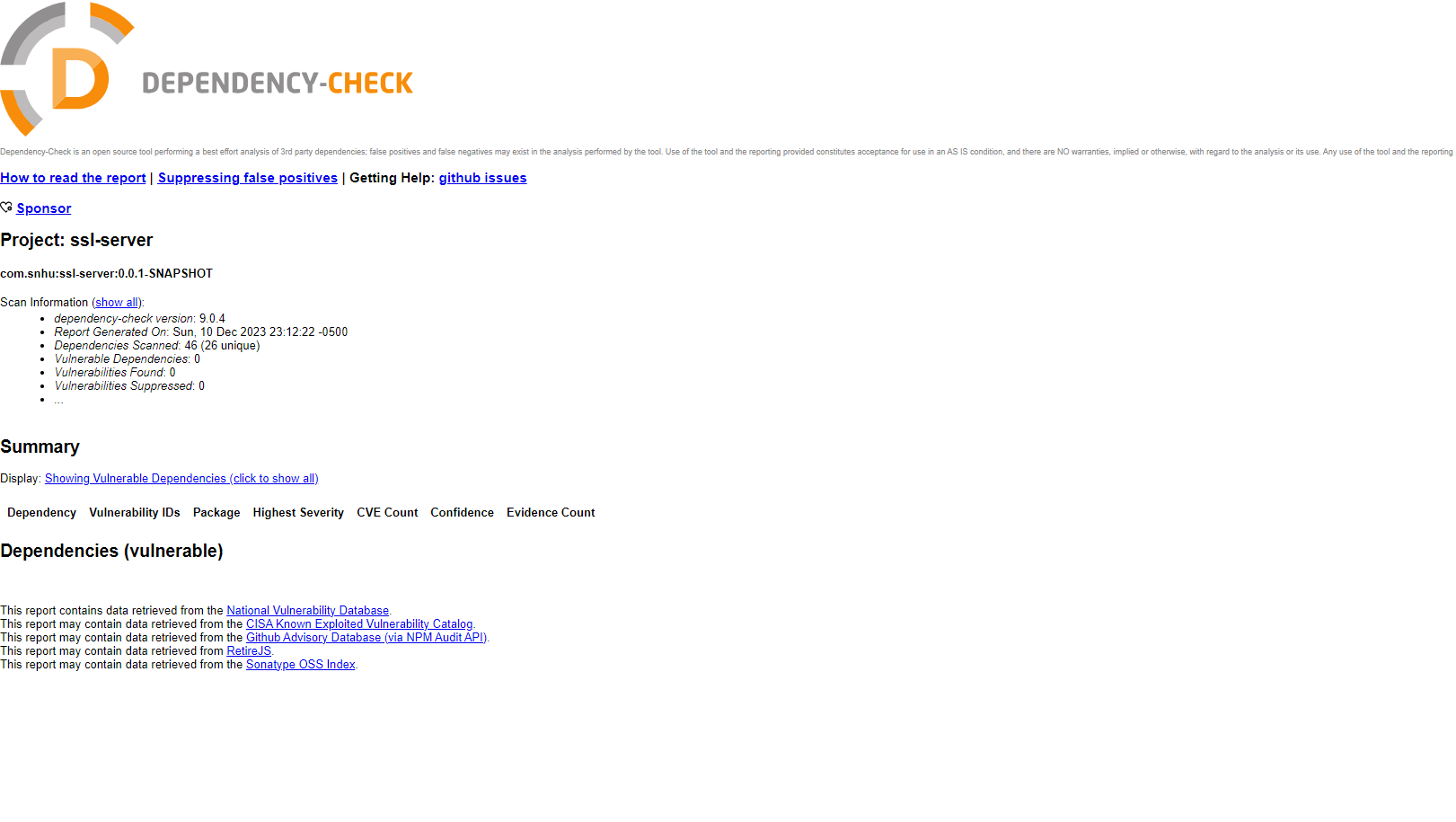
## Secure Communications

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



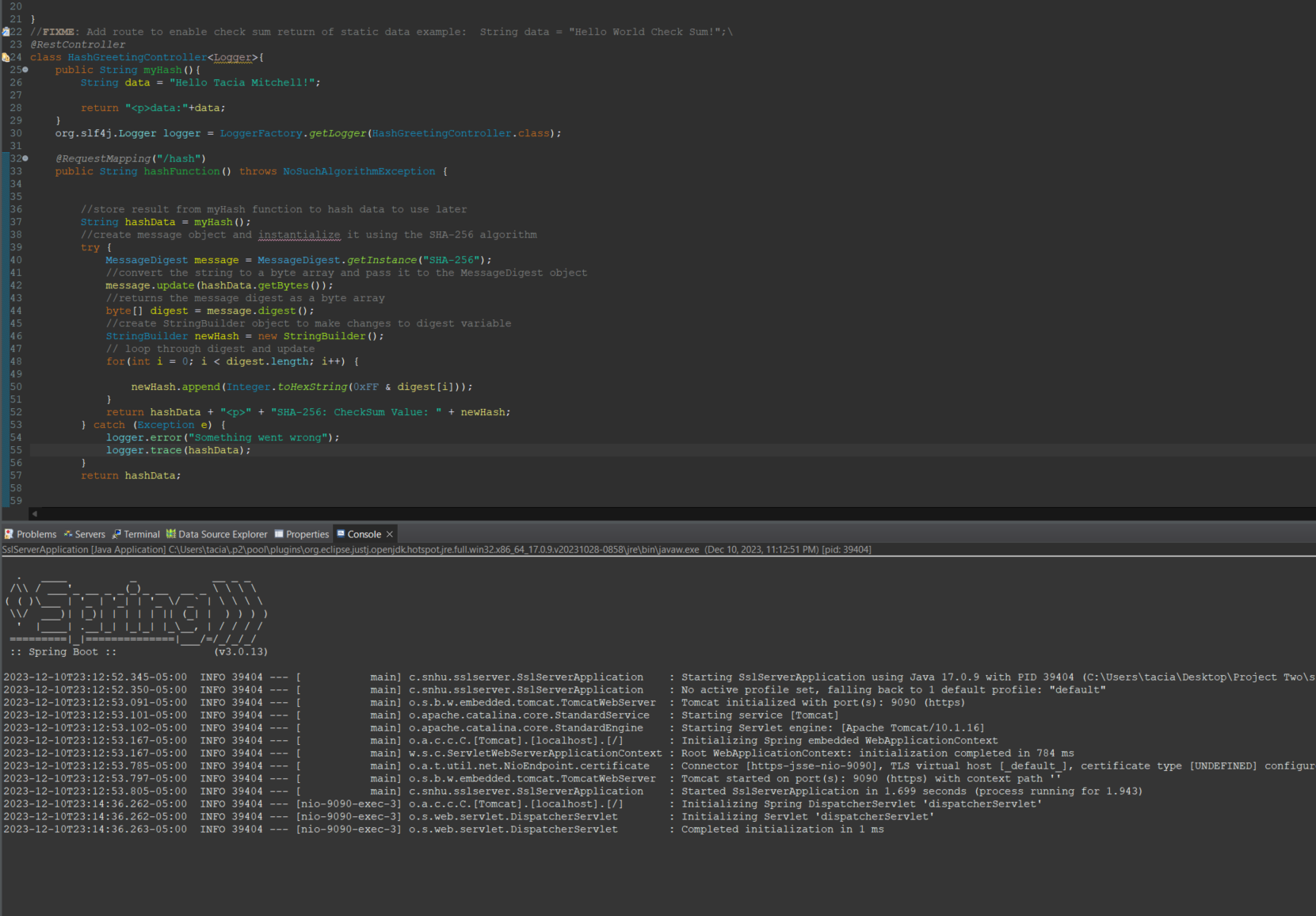
## Secondary Testing

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

To ensure the security of SHA-256, I utilized the message digest algorithm to create a hash code. Once I confirmed that my project was functioning smoothly, I proceeded to configure the application.properties file and generate a self-signed certificate. With the successful integration of the certificate, I was able to enable secure web browsing using the SSL.enabled command. Throughout the vulnerability assessment process, I prioritized cryptography and code review to identify and address potential vulnerabilities. My main goal was to improve security by upgrading spring boot dependencies. After upgrading to a stable version, I was pleased to see a reduction in my vulnerability count. To further enhance security, I added several dependencies to my pom, successfully remediating all known vulnerabilities.

## Industry Standard Best Practices

To ensure the utmost security of our software applications, I have implemented HTTPS into the application properties file and obtained the necessary certificates. Additionally, I have updated coding libraries by analyzing the dependency report and identifying and resolving any issues through the pom hierarchy. To further improve security, I have integrated a try-catch statement within the controller to detect any potential problems. In the event of an issue, detailed information will be logged to assist with troubleshooting.

In the technology industry, it is critical to follow best practices for secure coding. These practices help mitigate risks by addressing known vulnerabilities and implementing appropriate security measures. By adhering to regulatory compliance, companies can reduce the likelihood of security breaches and protect sensitive information. Furthermore, implementing best practices can enhance a company's reputation and build trust with customers.