****

# 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** | **Keri Ludemann** |  |

## 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

Keri Ludemann

## Algorithm Cipher

* I recommend the AES cipher due to it being one of the safest ciphers around and it supports numerous key sizes. AES 128 or 256 would work best for this application with AES 256 being the hardest to crack due to the amount of key values. This also extremely reduces the likely hood of two documents having the same key.
* AES hash takes a string input and returns a fixed length string as output. For example, if you use AES 256, then the string will be returned as a 256 bit string.
* AES is a symmetric cipher meaning it encrypts and decrypts using the same key while an asymmetric cipher uses a pair of keys, a public key and a private key. The public is used for encrypting the data while the private is used to decrypt it.

## Certificate Generation

Insert a screenshot below of the CER file.

A computer screen with white text

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

Sadly, this is the only screenshot I can provide that shows the code is working but I was unable to figure out how to get it to output anything.

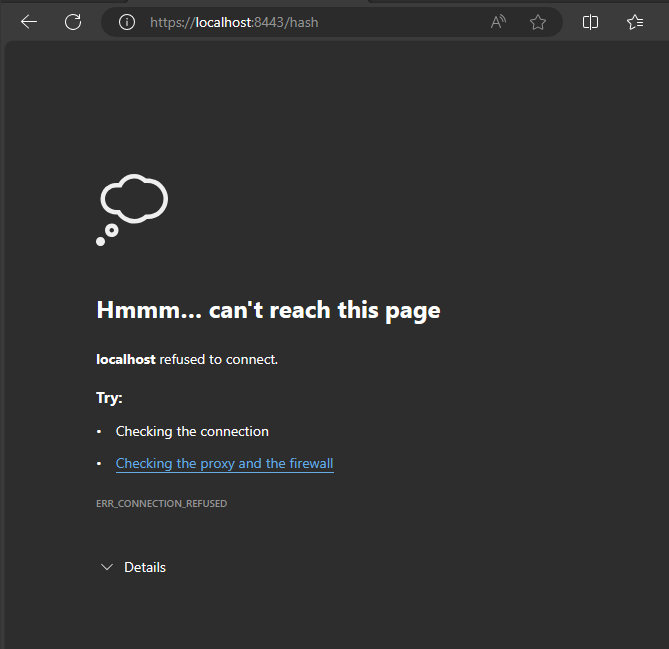
A screenshot of a computer program

Description automatically generated

## Secure Communications

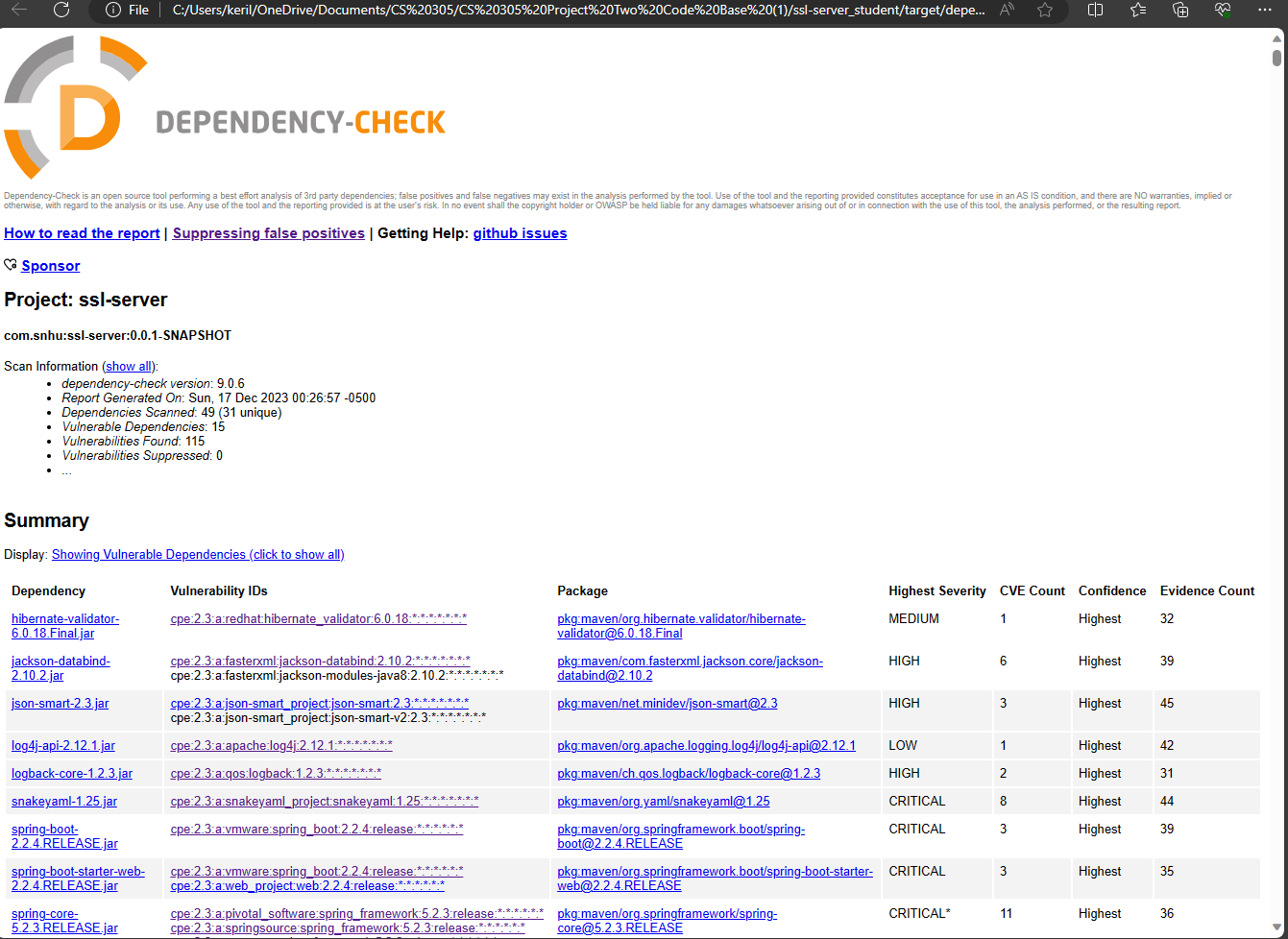
Insert a screenshot below of the web browser that shows a secure webpage.

I’ve tried different browsers and deleted all the cookies and cashe for them but it still wouldn’t connect.



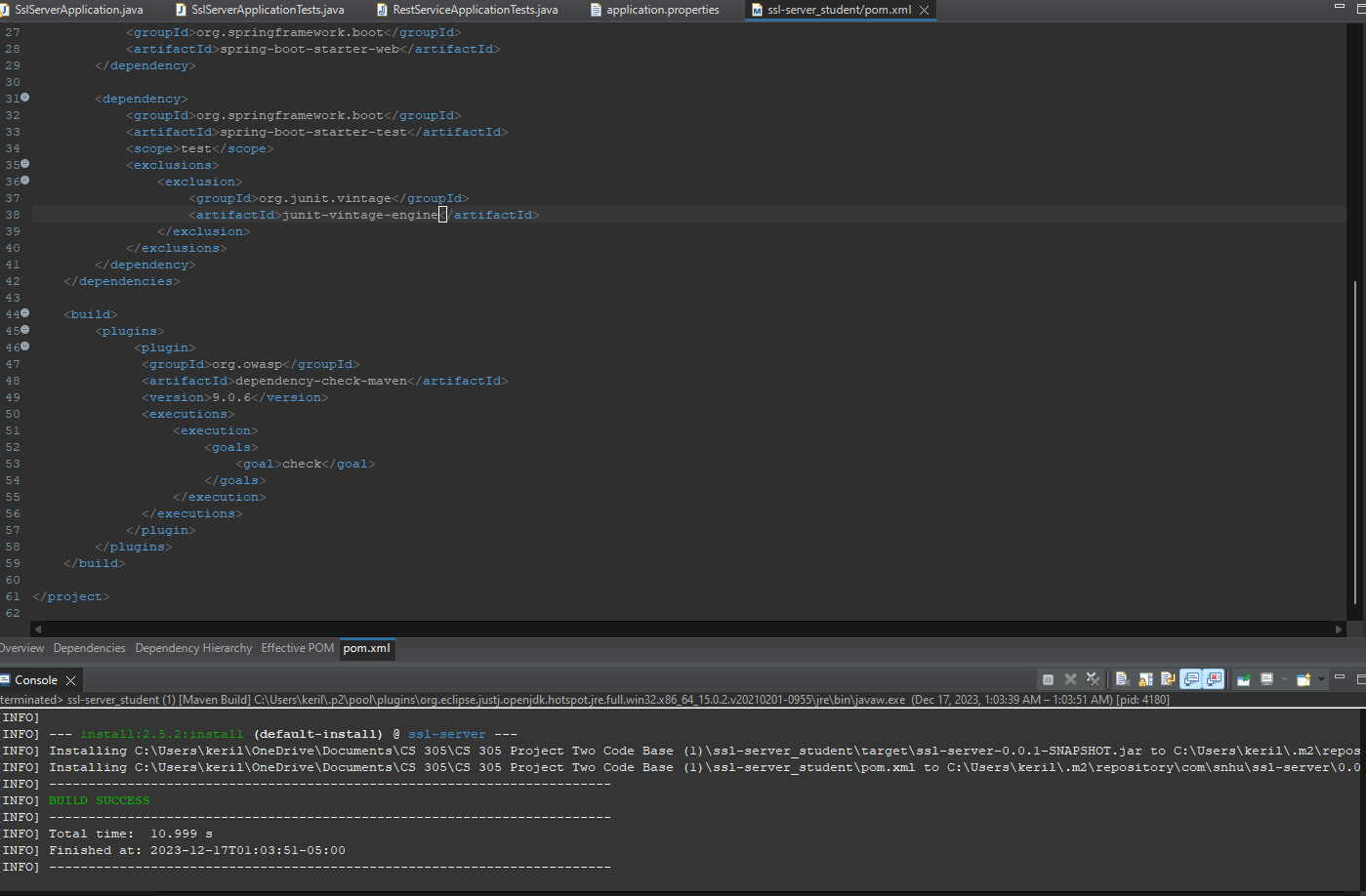
## Secondary Testing

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

A SHA-256 bit checksum was added, the maven dependency check was updated to the newest version and a self-signed certificate was generated and added. The updates to the code focused on API, cryptography, and the client/server areas of the Vulnerability Assessment Flow Diagram. The checksum allowed us to encrypt our string data meaning only the intended recipient will be able to decrypt it. Updating the dependency check was important to ensure that we are being the correct vulnerabilities within the code, although there was no blocking of false positives.

## Industry Standard Best Practices

## Code Reviews

## This was implemented through the thorough review of the code. This standard is meant to ensure quality of the code and ensure eases of understanding.

## Refactoring

## Code had to be refactored to allow for a checksum, updated vulnerability assessment and so that the certificate could be implemented. This standard is used to improve code’s flexibility and maintainability.

## Security

## All steps we did with refactoring the code were meant to ensure the security of the program. Utilizing security best practices allows for safer programs that better protects its users.

## Resources

## https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/aes-hash/aeshash.pdf

<https://www.progress.com/blogs/use-aes-256-encryption-secure-data>

<https://nordvpn.com/blog/sha-256/#:~:text=Many%20consider%20SHA%2D256%20to,cannot%20produce%20an%20identical%20hash>.

https://www.thinkful.com/blog/software-engineering-best-practices/