****

# 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** |
| --- | --- | --- | --- |
| **1.1** | **8/16/2025** | **Christian Wallace** |  |

## 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

Christain Wallace

## Algorithm Cipher

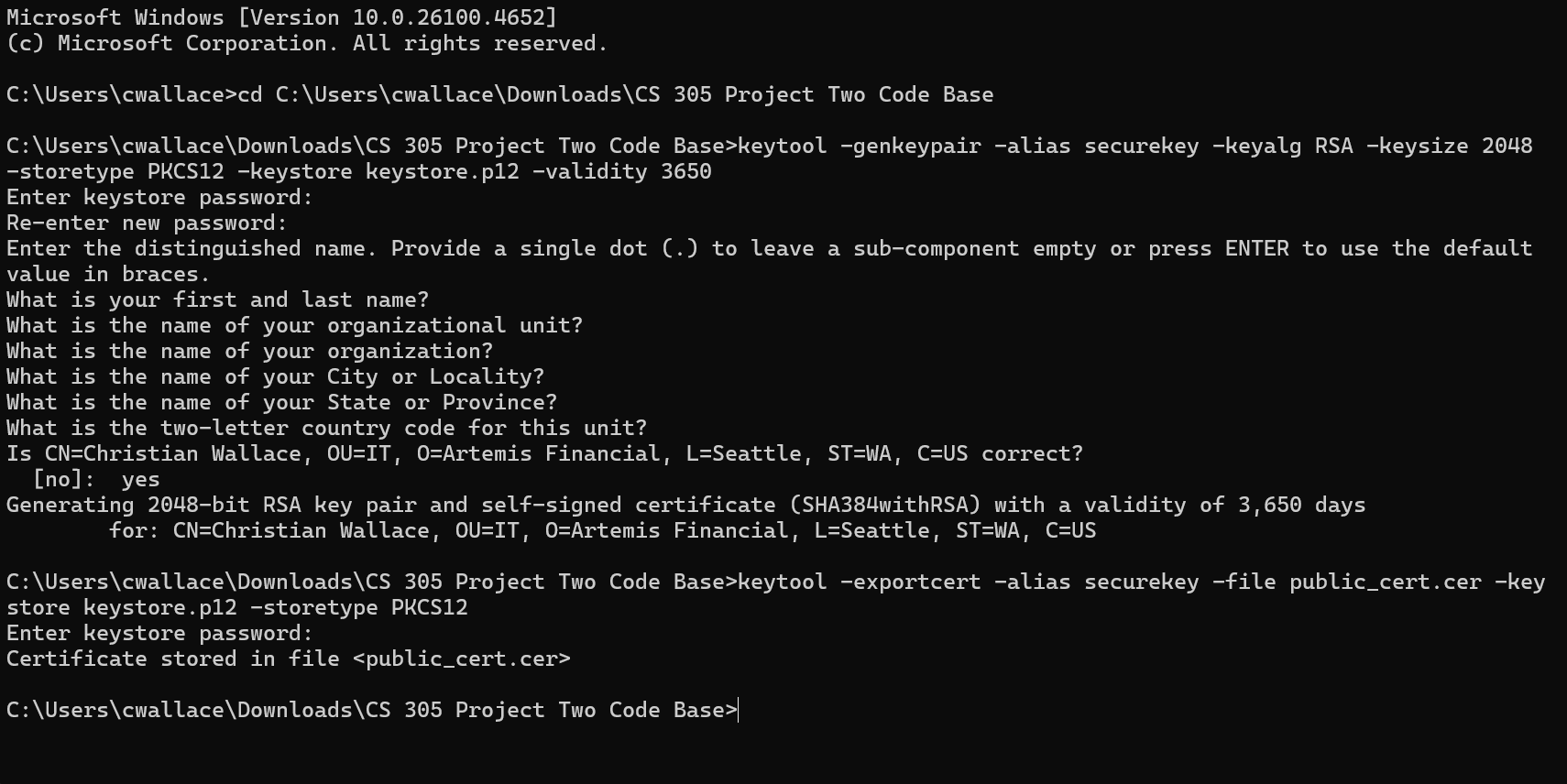
The Advanced Encryption Standard ois a symmetric block cipher that became the U.S. standard in 2001. It replaced DES which was no longer secure. AES works with 128 bit blocks and allows keys of 128, 192, or 256 bits. Its strength comes from running data through many rounds of substitutions and permutations which makes brute force attacks almost impossible.

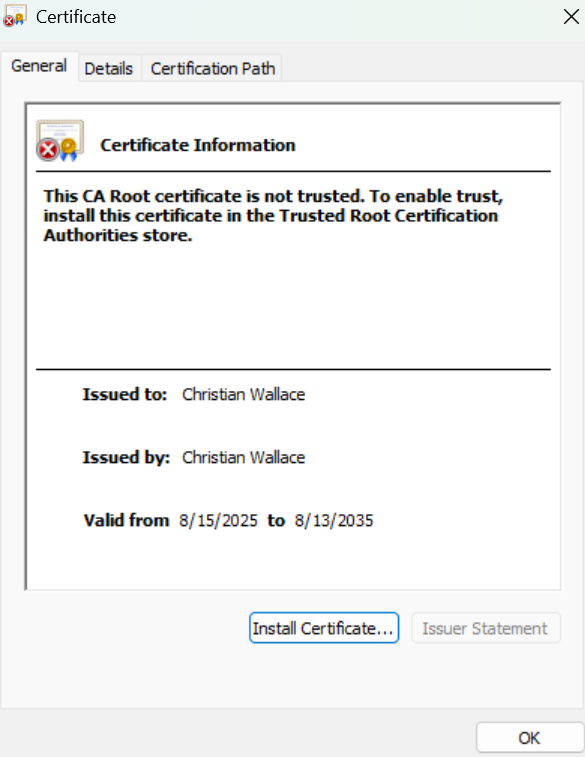
Applications also use cryptographic hash functions to make sure data is not changed. SHA 256 is a common one. It creates a fixed 256 bit value from any input and is designed to avoid collisions. AES is often used with SHA 256 so that AES keeps the data private while SHA 256 makes sure it has not been tampered with. Random numbers are another important part of encryption. They are used to create things like initialization vectors, salts, and session keys. This makes sure that even if you encrypt the same data twice you will not get the same result.

AES is symmetric so the same key is used to encrypt and decrypt. Asymmetric methods like RSA or ECC use a public key and a private key. They are slower but are often paired with AES to share session keys securely. Encryption has changed a lot over time. Early ciphers like Caesar shifts were simple. DES was once common but its short key length made it weak. AES replaced it and is now used everywhere including banks, governments, and commercial software. Today research continues on encryption that can handle new threats like quantum computing.

## Certificate Generation

Insert a screenshot below of the CER file.





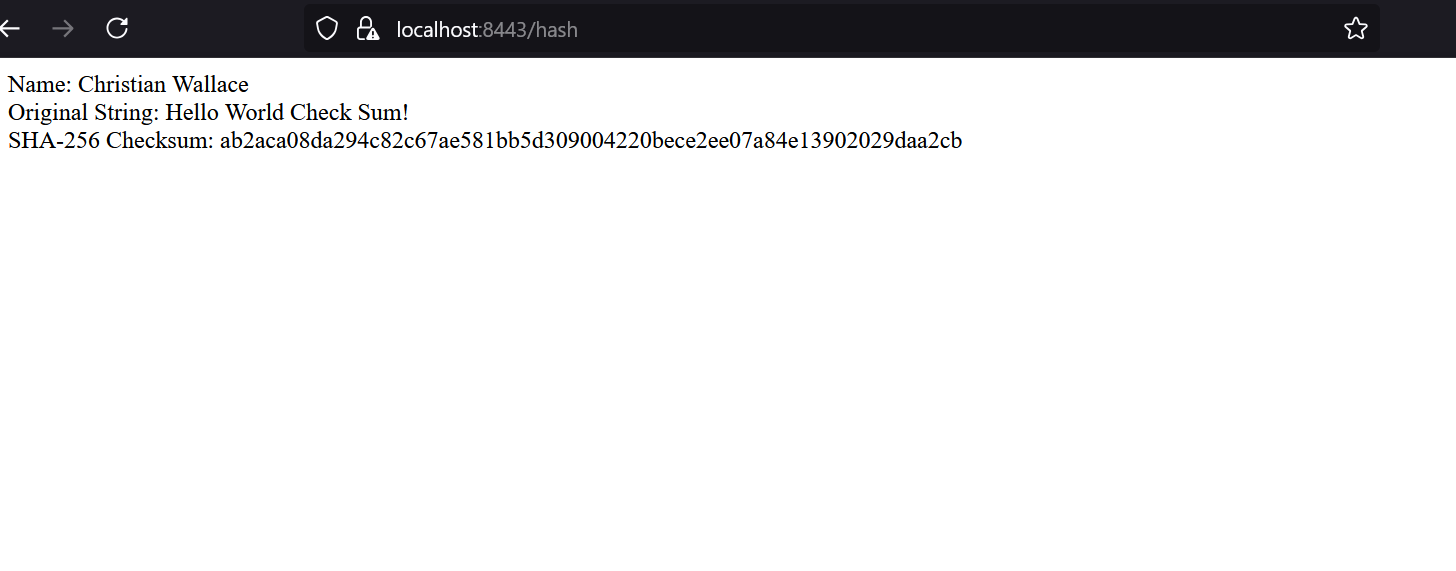
## Deploy Cipher

Insert a screenshot below of the checksum verification.



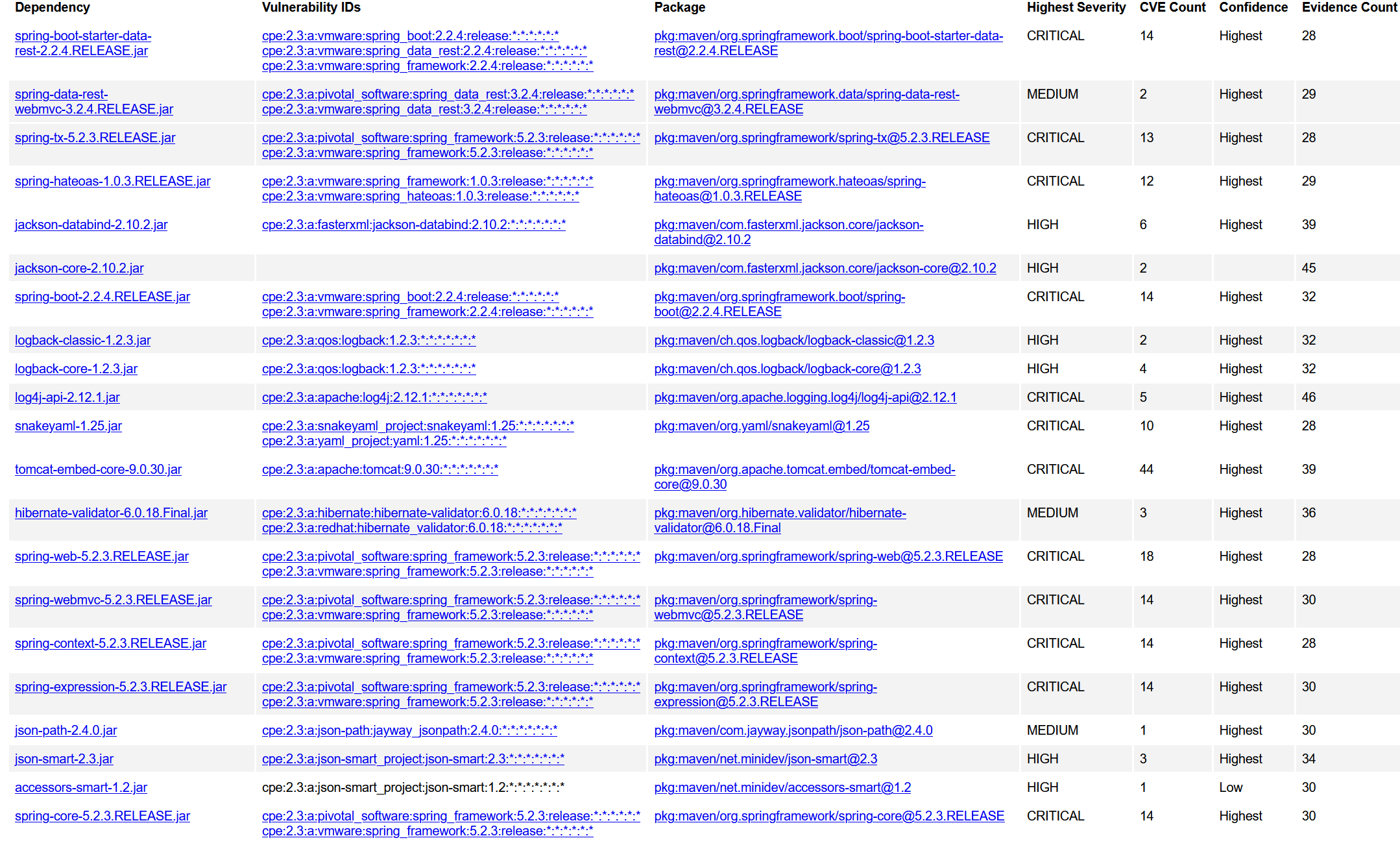
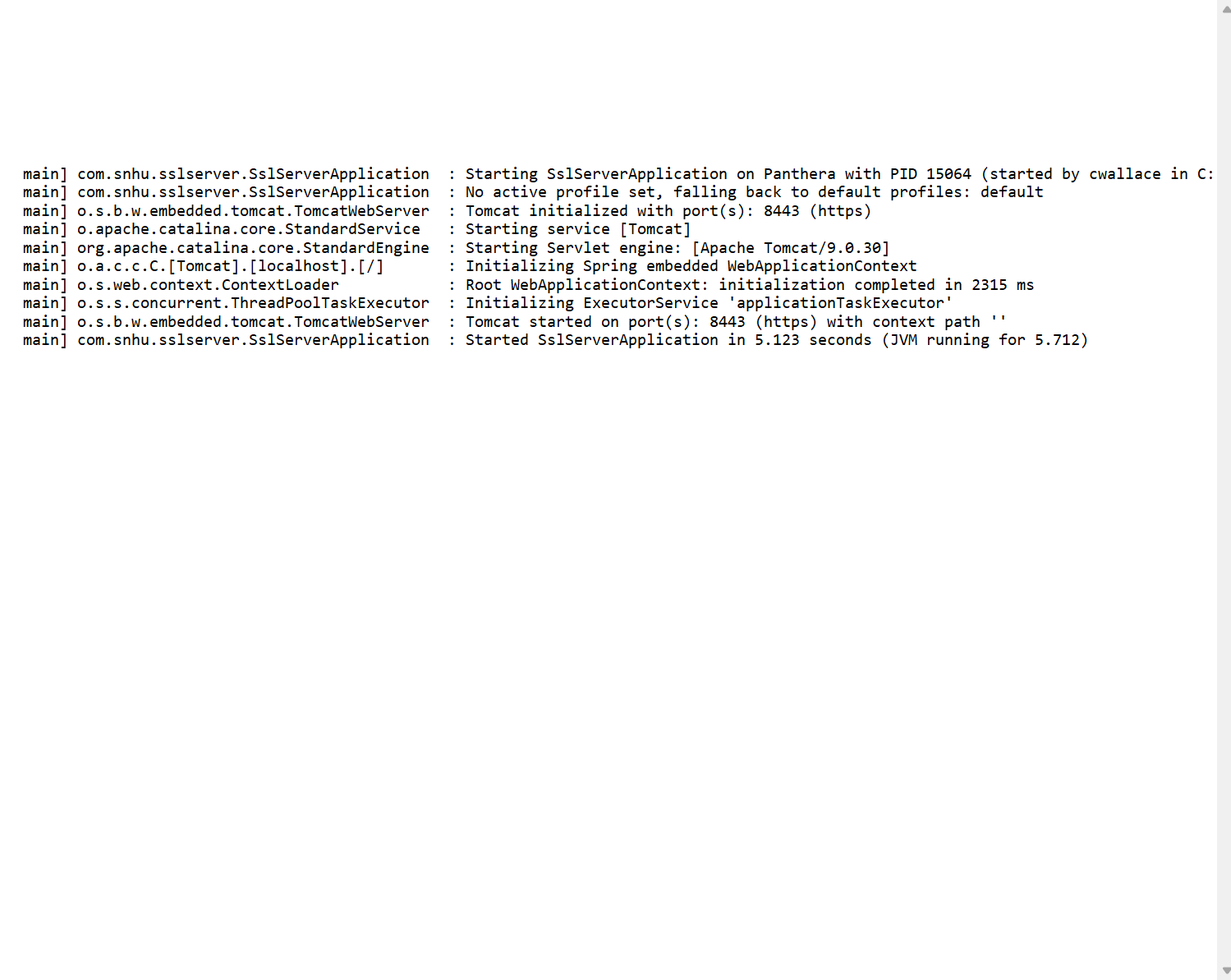
## Secure Communications

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



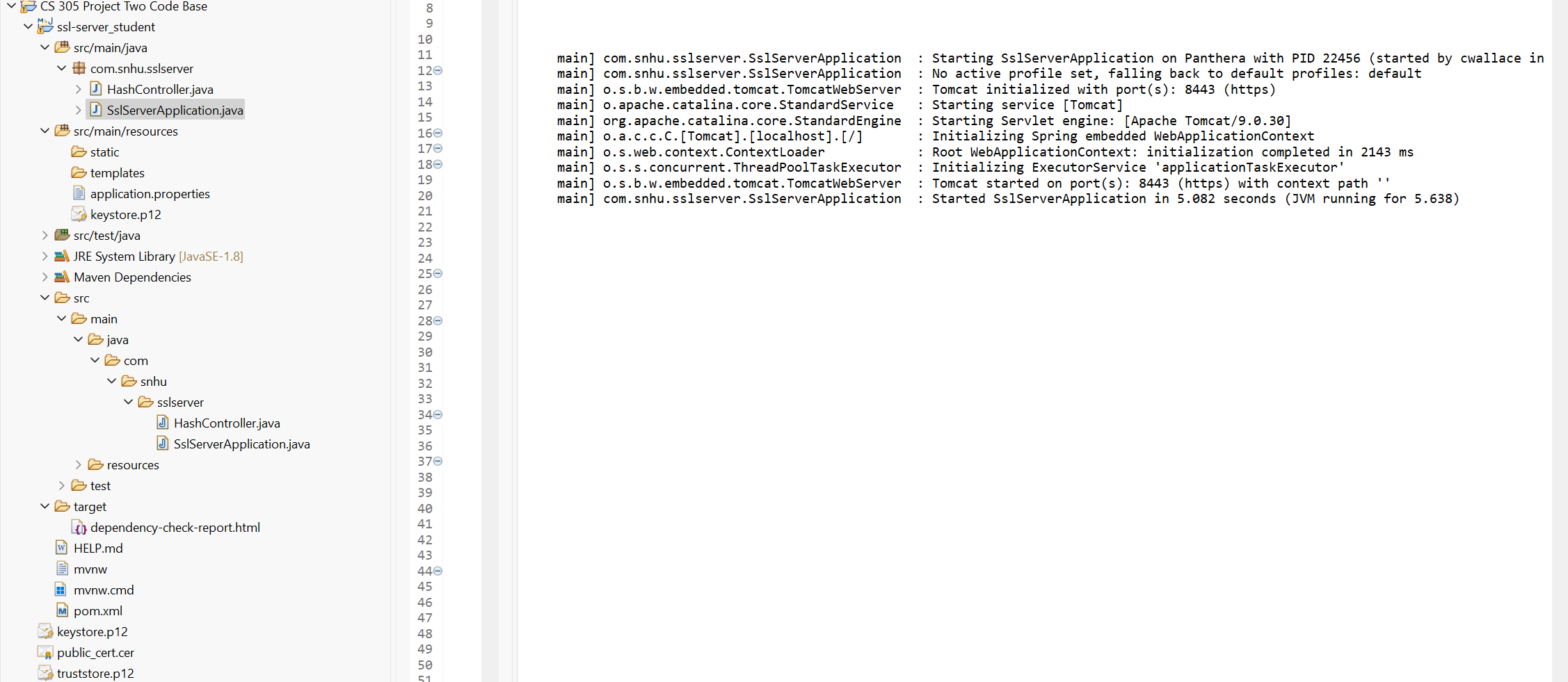
## Secondary Testing

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

The secondary static testing was completed using the OWASP Dependency-Check tool. The scan report identified several existing vulnerabilities in third-party dependencies none of these were introduced by the refactored code. This confirms that the changes made to implement encryption, hashing, and secure communication did not create additional security risks.

## Functional Testing

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



## Summary

In this project, the Artemis Financial web application was refactored to incorporate secure communication using HTTPS. A self-signed certificate was generated and configured in the application to enable SSL/TLS encryption. The application was also updated to use a checksum generation method that leverages a cryptographic hash algorithm to ensure file integrity. Static and functional testing were performed, including a dependency check scan to identify vulnerabilities and a successful execution of the refactored code.

## Industry Standard Best Practices

When developing secure software applications, it is important to follow industry standard best practices that focus on both reliability and protection. All communication should occur over HTTPS using SSL/TLS certificates issued by trusted authorities, ensuring strong encryption for data in transit. Applications should also implement cryptographic hashing algorithms, such as SHA-256, to verify file integrity and prevent tampering. To manage risks in third-party libraries, developers should regularly perform vulnerability scans with tools like OWASP Dependency-Check. Secure coding practices are also critical, including validating inputs, and avoiding hardcoding sensitive information such as passwords or keys. Access controls must be enforced to ensure that authentication and authorization policies restrict entry to sensitive resources. Frameworks, libraries, and servers should be kept up to date with the latest patches to minimize exposure to known vulnerabilities. Effective logging and monitoring should be in place to detect suspicions activity, provide insight into potential threats, and report incidents.