****

# Practices for Secure Software Report

**Table of Contents**

[**Document Revision History 3**](#_sr7fldx2ss7i)

[**Client 3**](#_68jzje5j6jjq)

[**Instructions 3**](#_dhn6p9pfjvz7)

[**Developer 4**](#_k8edfwf6dffn)

[**1. Algorithm Cipher 4**](#_tfeuuakhykfs)

[**2. Certificate Generation**](#_5grfb5n1e177) **5**

[**3. Deploy Cipher**](#_aegnh92azl3b) **6**

[**4. Secure Communications**](#_f7r2vy9hp03a) **6**

[**5. Secondary Testing**](#_kx5bbreyhkjq) **7**

[**6. Functional Testing**](#_tz6zbl4k97pv) **8**

[**7. Summary**](#_pjqilhm3ja86) **8**

[**8. Industry Standard Best Practices**](#_o2244cmbtbuc) **8**

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **6/20/2025** | **Cameron Sharp** |  |

## Client



## Instructions

Submit these completed practices for a 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

Cameron Sharp

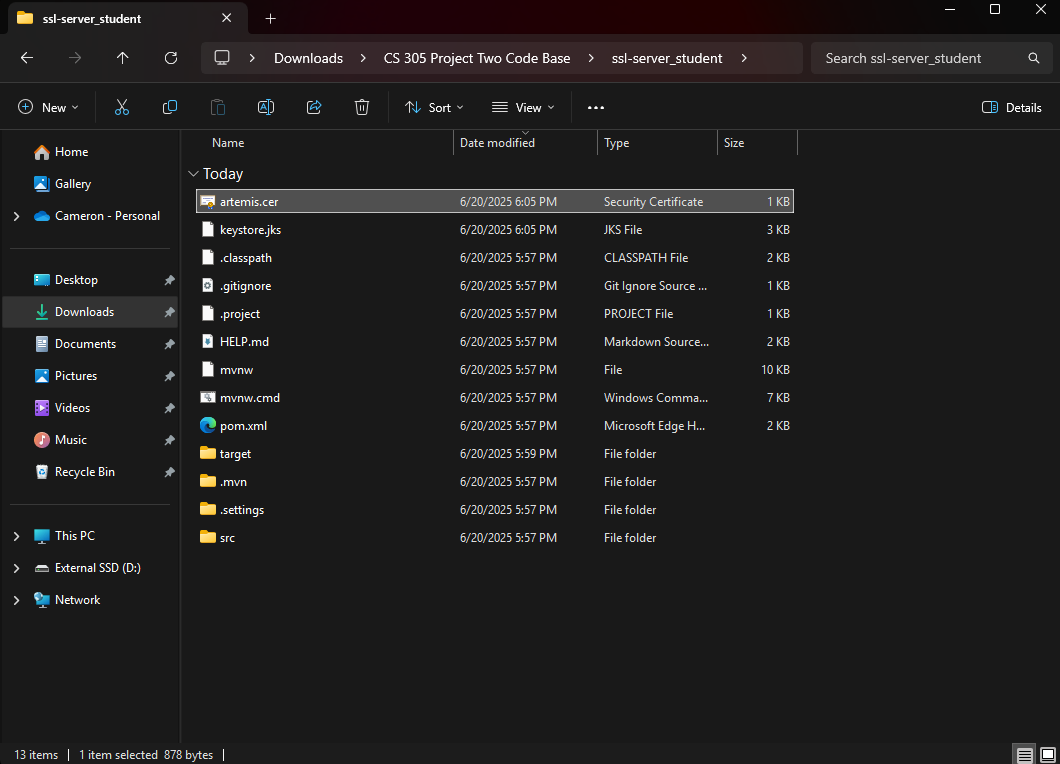
## Algorithm Cipher

The Advanced Encryption Standard (AES) encryption cipher and the SHA-256 cryptographic hash function are recommended for the Artemis Financial application. AES is a symmetric block cipher that supports 128, 192, and 256-bit keys and encrypts data in fixed 128-bit blocks. SHA-256 was used for the project's checksum verification, and any input data produced a fixed 256-bit output. Checksum verification for the project was done using SHA-256, with a fixed 256-bit output generated for any input data. Unlike encryption, however, SHA-256 is not reversible and ideal for checksum verification to provide data integrity.

SHA-256 usage does not apply to encryption keys but does offer the capability of secure verification when transmitting data between systems. AES, however, uses symmetric keys, where the same key which is used for encryption is also utilized for decryption. Symmetrical ciphers are more efficient and quicker to process than asymmetrical ciphers and are used to encrypt bulk data most of the time. Even though this project was working with hashing and not encryption, secure key generation and numbers would be included in the production use of AES. The history of encryption has indeed progressed away from outdated DES encryption and towards more secure AES, and AES is one of the most efficient and strongest ciphers developed to this point. Having added SHA-256 to this solution, we’ve provided the capability to securely and reliably verify data being transmitted without affecting the underlying logic of the application.

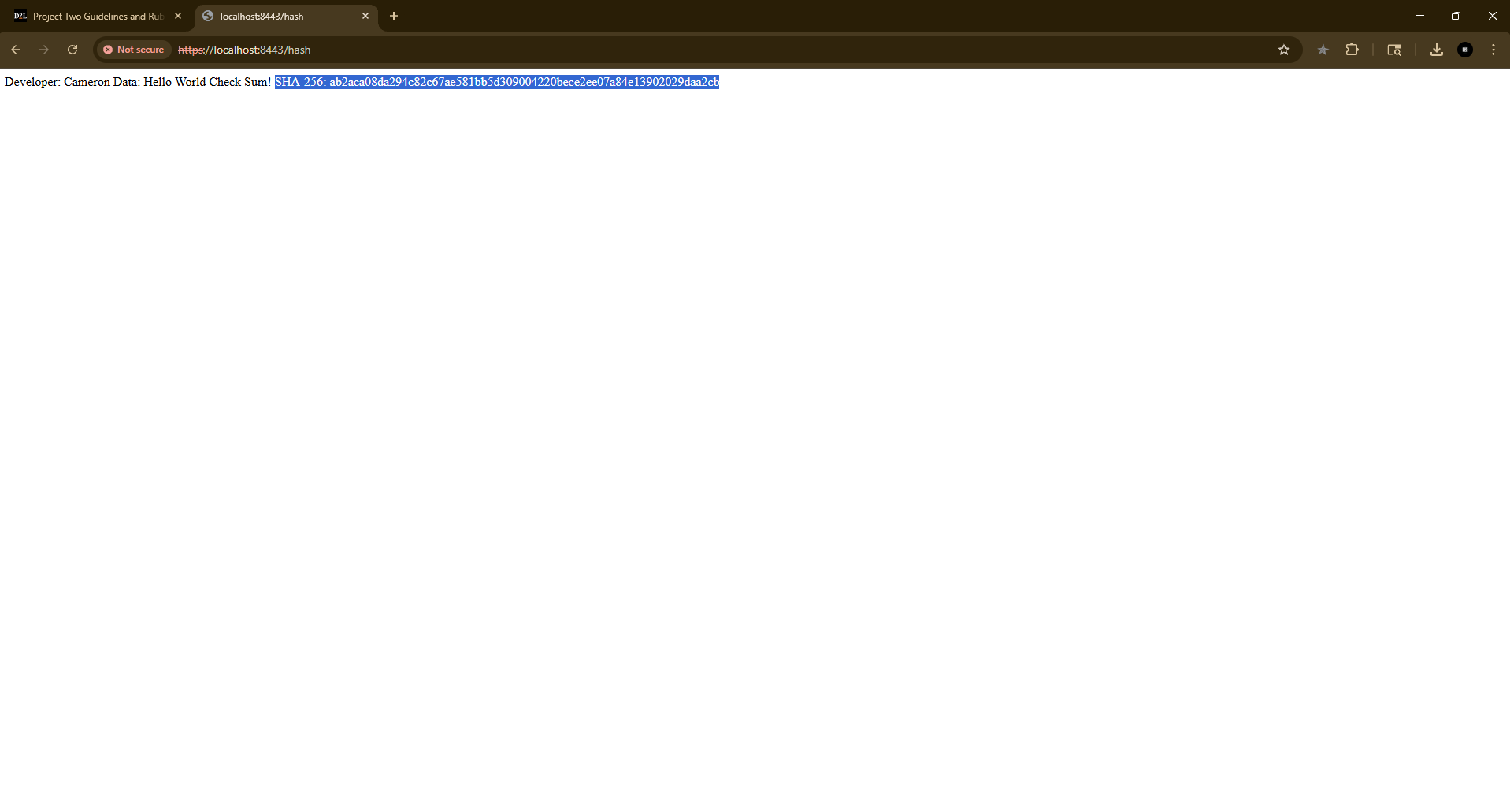
## Certificate Generation

Insert a screenshot below of the CER file.



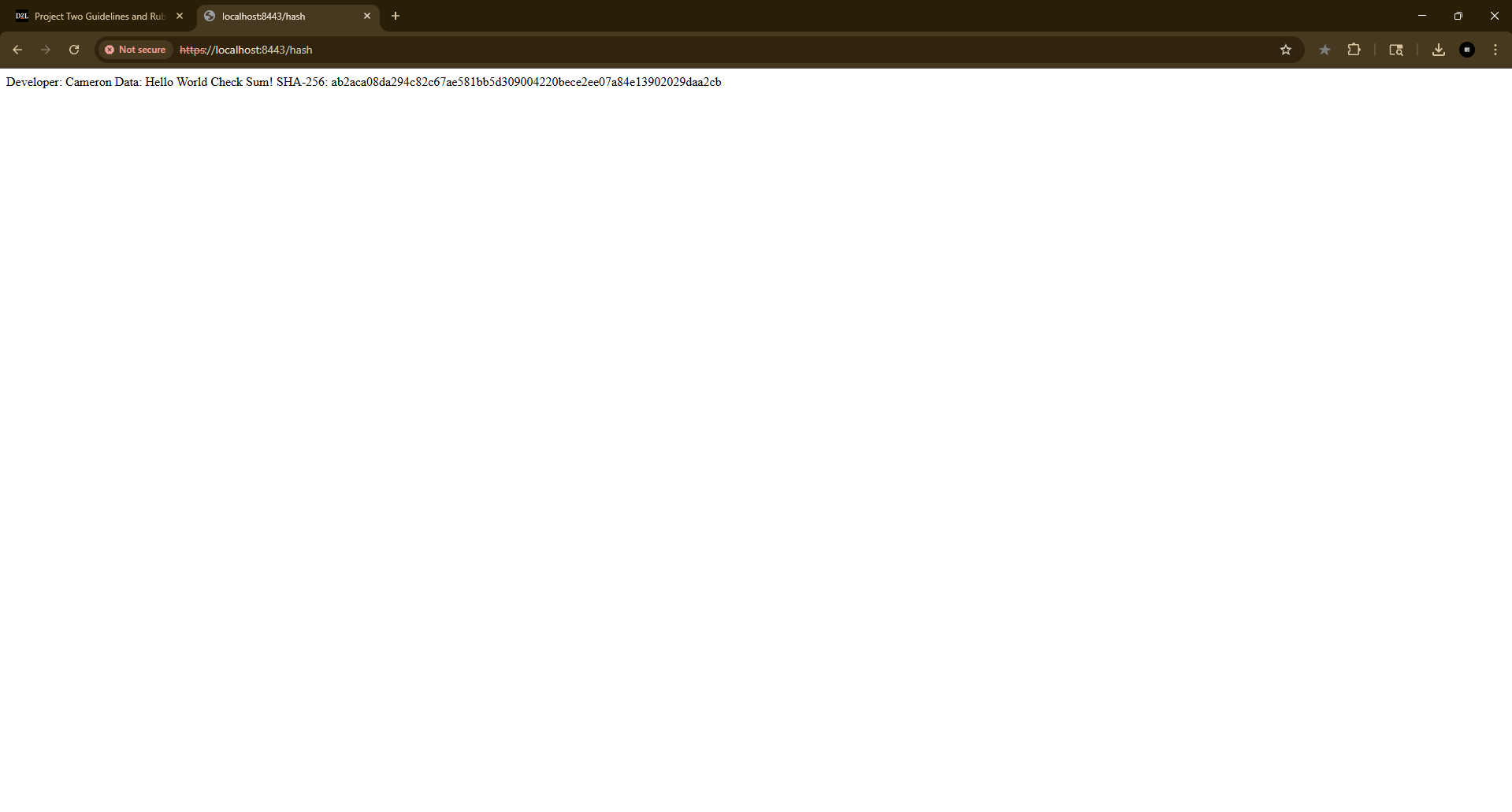
## Deploy Cipher

Insert a screenshot below of the checksum verification.



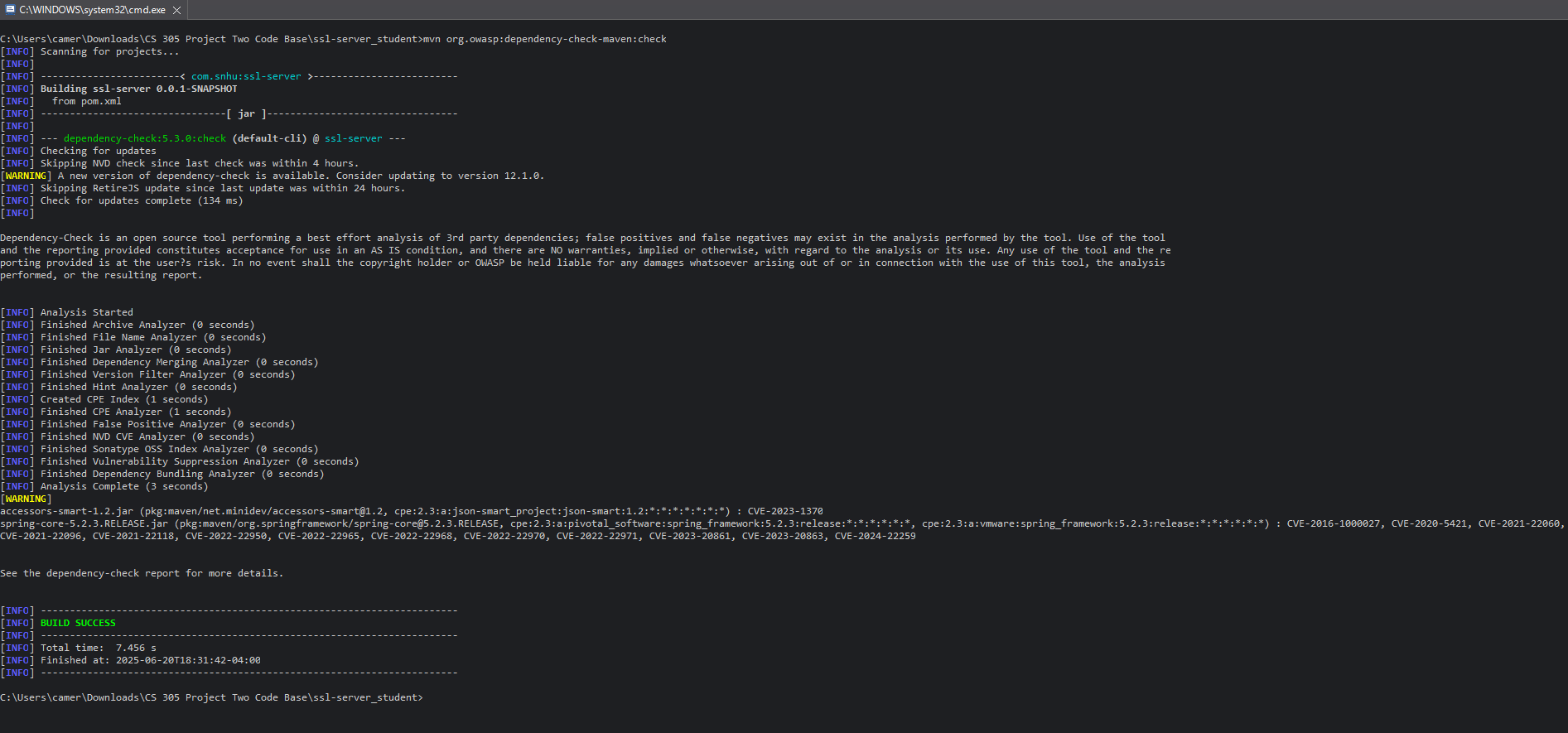
## Secure Communications

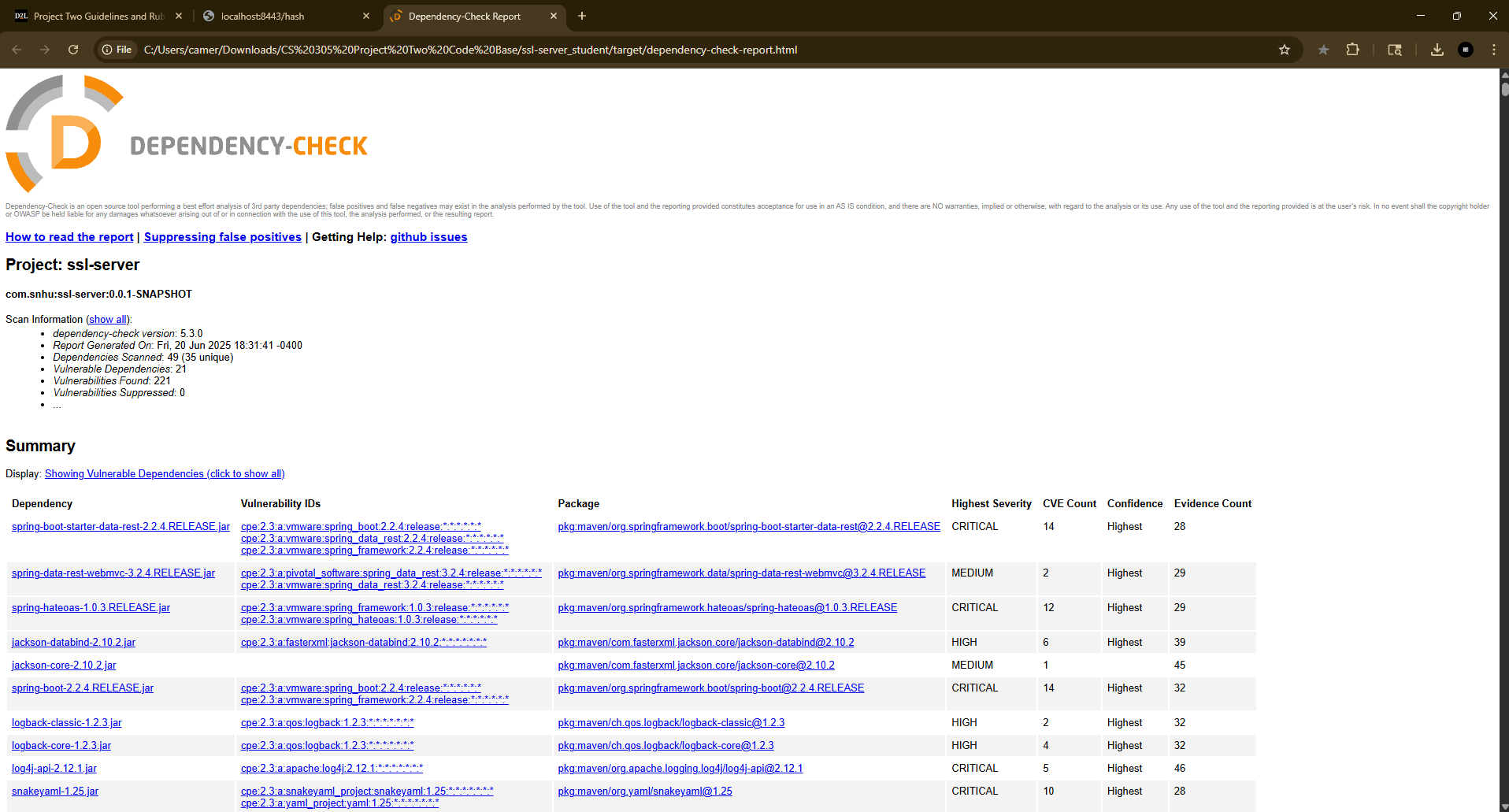
Insert a screenshot below of the web browser that shows a secure webpage.



## Secondary Testing

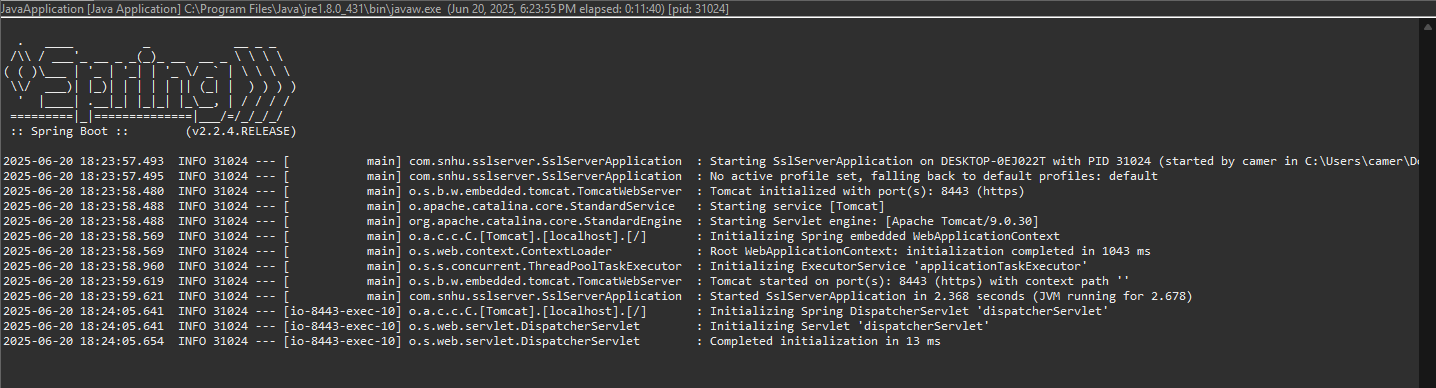
Insert screenshots below of the refactored code executed without errors and the dependency-check report.





## Functional Testing

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



## Summary

I added a new section that enables the app to return a hash value using SHA-256 in order to increase its security. When someone hits the /hash route, they see the hash, my name, and the original string because it is based on a defined data string. This way the hash would also change if anything about the data changed, and alert you to the issue. I also created a self-signed certificate to set up HTTPS, which encrypts the connection between the browser and the server by using port 8443 when the application starts.

The security domains depicted in the vulnerability flowchart are partially covered by these modifications. Data integrity is enhanced by the checksum component, and information is protected during transmission by using HTTPS. In order to avoid interfering with existing functionality, I also kept my new code distinct and straightforward. Everything checks okay since I used the OWASP dependency-check tool to perform a static scan once I was done and found nothing that would be regarded as a new risk.

## Industry Standard Best Practices

While updating the app, I tried to follow industry best practices that are common in the software sector. To keep things organized, I developed a separate controller just for the hash function rather than cramming it into the main file. I utilized SHA-256, which is regarded as reliable and secure, and a self-signed certificate to establish HTTPS in order to stop data from flying around unprotected. When working with user data, such procedures are frequently expected.

While updating the app, I was careful to add to the original code without changing the application's primary functionality because I did not want to destroy anything that was already working. With this, I ran the OWASP dependency check to make sure I did not inadvertently include any risky packages or dependencies. For a company like Artemis Financial, where clients expect their private information to be kept protected, these protections make the app safer, overall leading to a more trusted and transparent connection between Artemis and the clients.