****

# 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** | **10/20/2024** | **Zachary Fizet** |  |

## 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

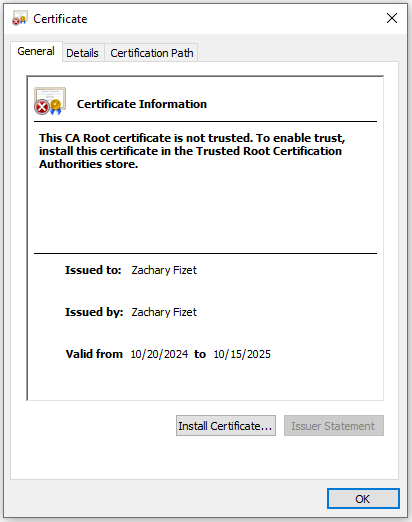
Zachary Fizet

## Algorithm Cipher

My personal recommendation for an algorithm cipher for Artemis Financial is AES-256. AES-256 is a symmetric block cipher established by NIST and is widely used in financial applications due to the high level of security and efficiency it provides. Additionally, AES-256 is widely accepted as it provides strong encryption that meets modern security requirements and integrates easily with other security protocols. AES-256 operates on 128-bit data blocks with a key size of 256 bits. AES itself is not a hash function, but is typically used in conjunction with cryptographic hash functions like SHA-256 or SHA-3. The SHA-256 hash function provides secure checksums that verify the integrity of the data. AES uses symmetric encryption, meaning that the same key is used for both encryption and decryption. This symmetry allows AES to operate quickly and efficiently. Asymmetric encryption would use two different keys, one public to encrypt the data, and one private key to decrypt the data. This form of encryption could be coupled with AES and used as a method for securing the keys used by AES, meaning that AES would be used for data encryption, and something like RSA would be used for ensuring that the keys for AES are securely stored and exchanged. Think of it like locking a car and storing the keys in a safety deposit box. The car is locked by the car keys, the car keys are then stored in a secure location that is accessible to anyone else with a safety deposit box, but only the parties with the keys to that specific deposit box will have access to the car keys. Random numbers are used in AES to generate keys. Random numbers are used to ensure complexity and to make it increasingly difficult for brute force attacks to work against AES encryption. In fact, AES-256 has never been broken via a brute force attack and is considered “virtually uncrackable”. It is estimated that a brute force attack would take millions of years to work against AES-256 encryption. This is due to the randomness of keys and their length. However, this isn’t to suggest that it is completely secure, as the main vulnerability to AES-256 is unsecure encryption key access. AES became the industry standard for encryption in 2001, replacing the original Data Encryption Standard and remains to this day.

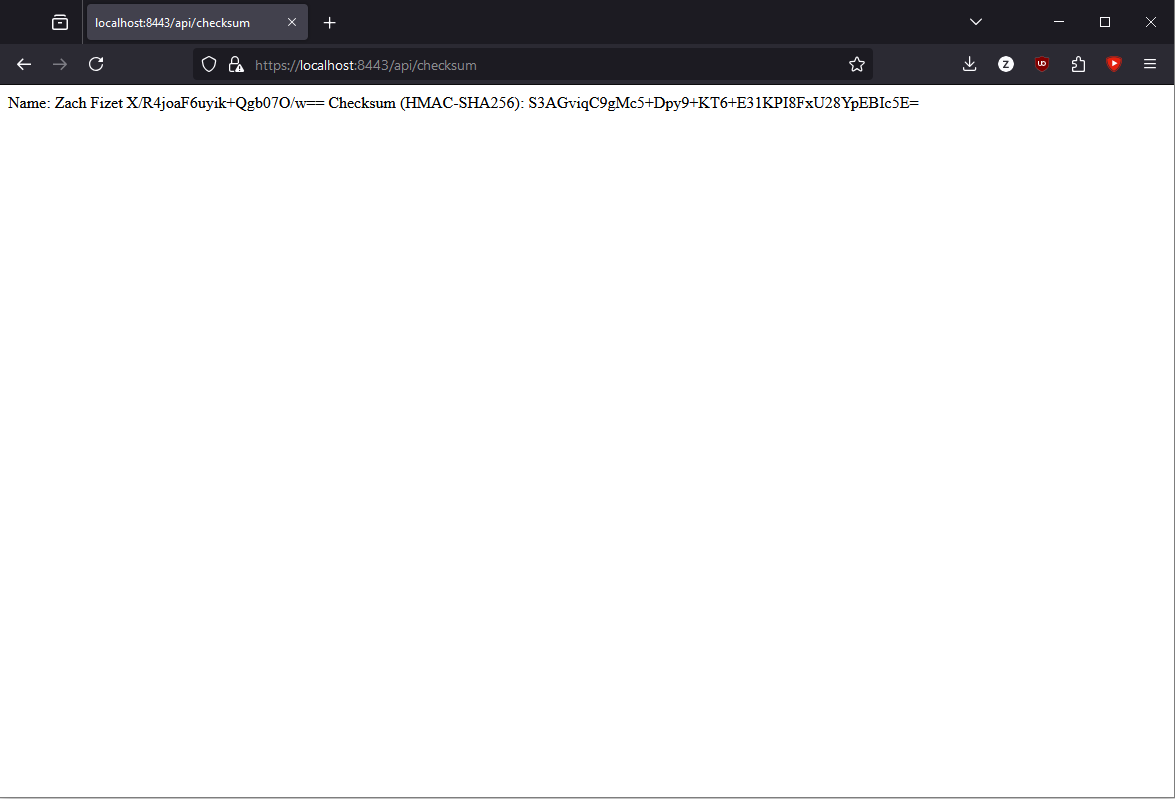
## Certificate Generation

Insert a screenshot below of the CER file.



## Deploy Cipher

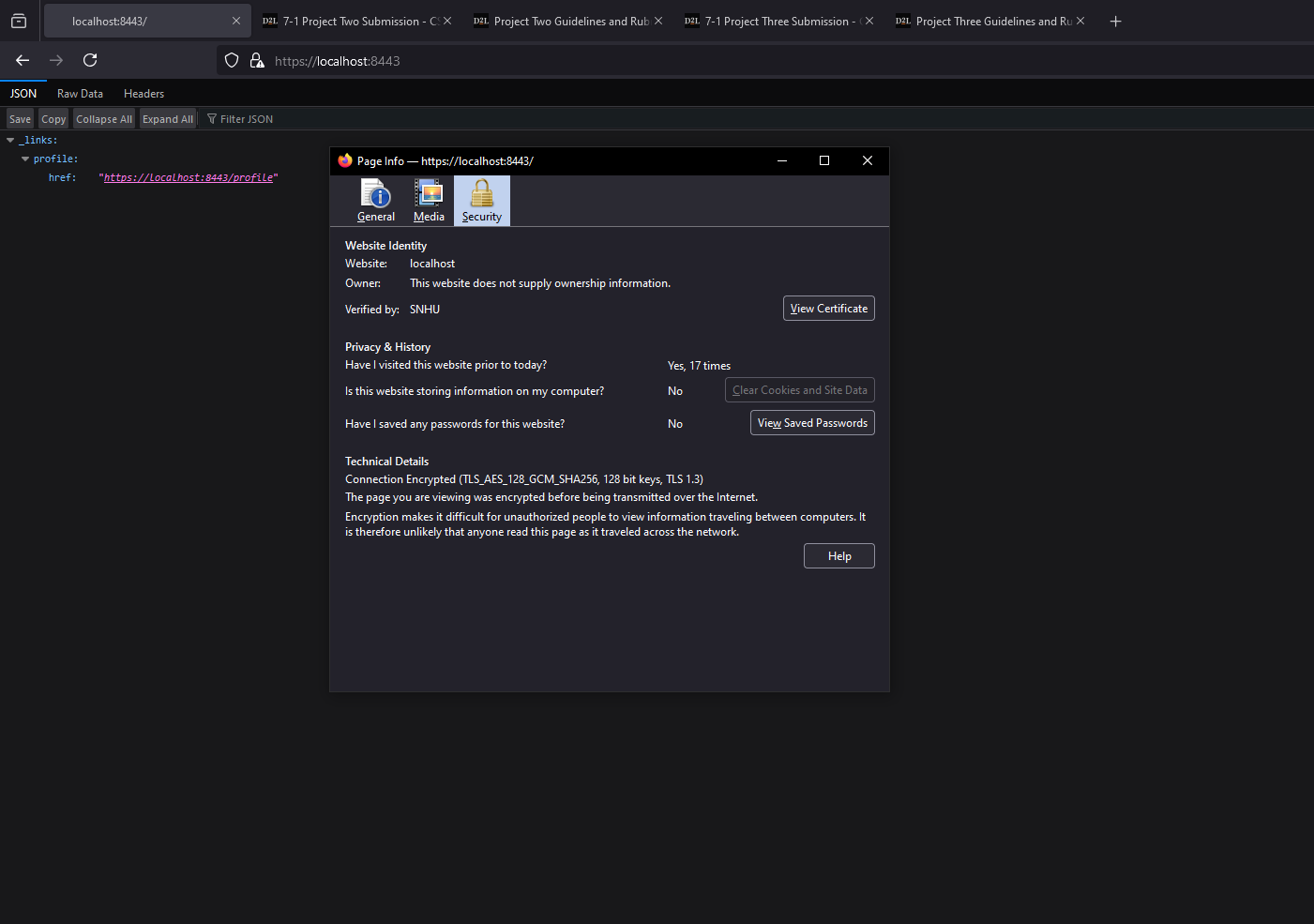
Insert a screenshot below of the checksum verification.



## Secure Communications

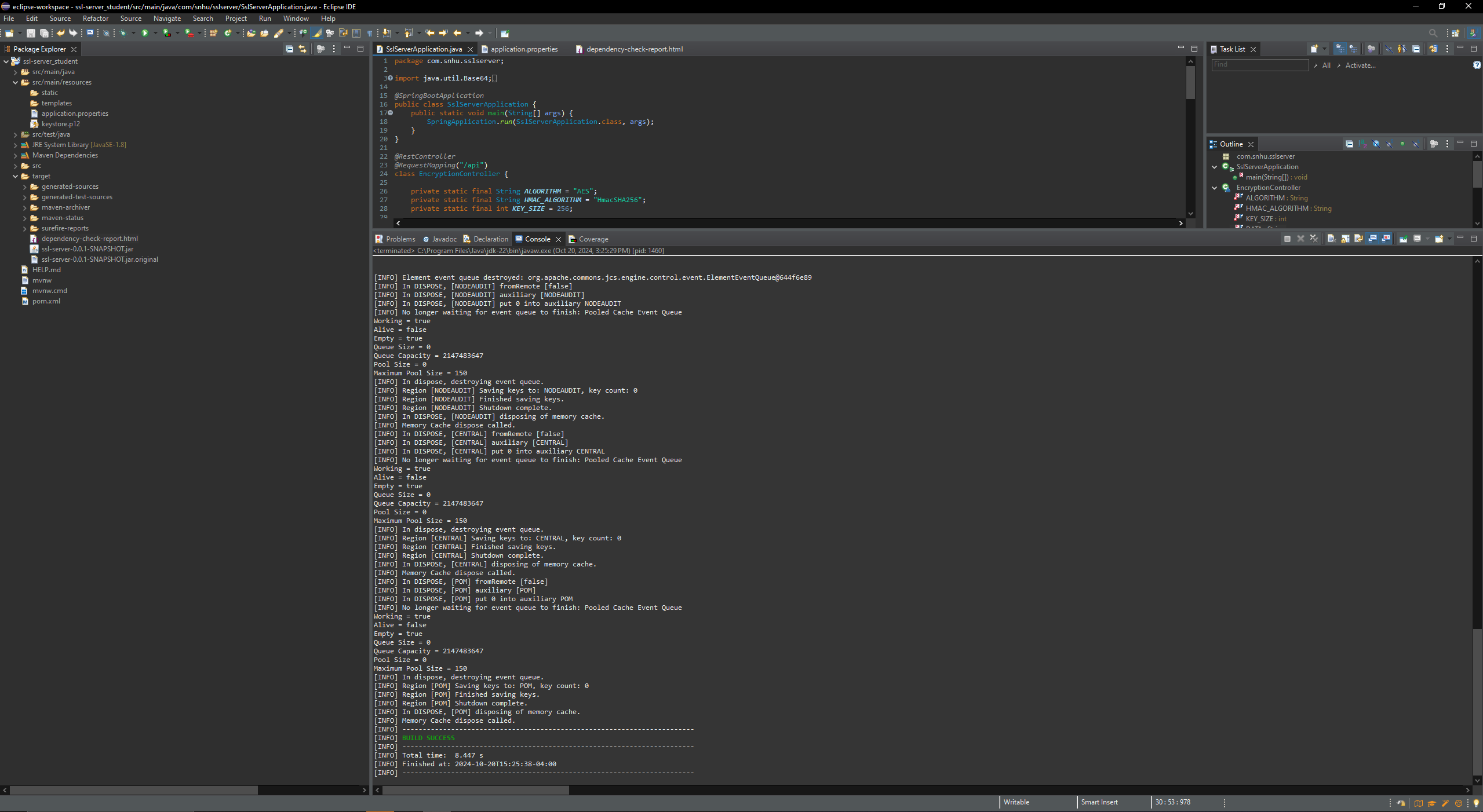
Insert a screenshot below of the web browser that shows a secure webpage.

I am not sure why I cannot get my browser to cooperate as I have enabled SSL in the application, regenerated and replaced the certificate a dozen times, but when I connect to <https://localhost:8443/> it is still appearing as “not secure” despite me installing the certificate, including uploading it as “trusted” into my browser and such. However, the site is showing as encrypted when I inspect the details.

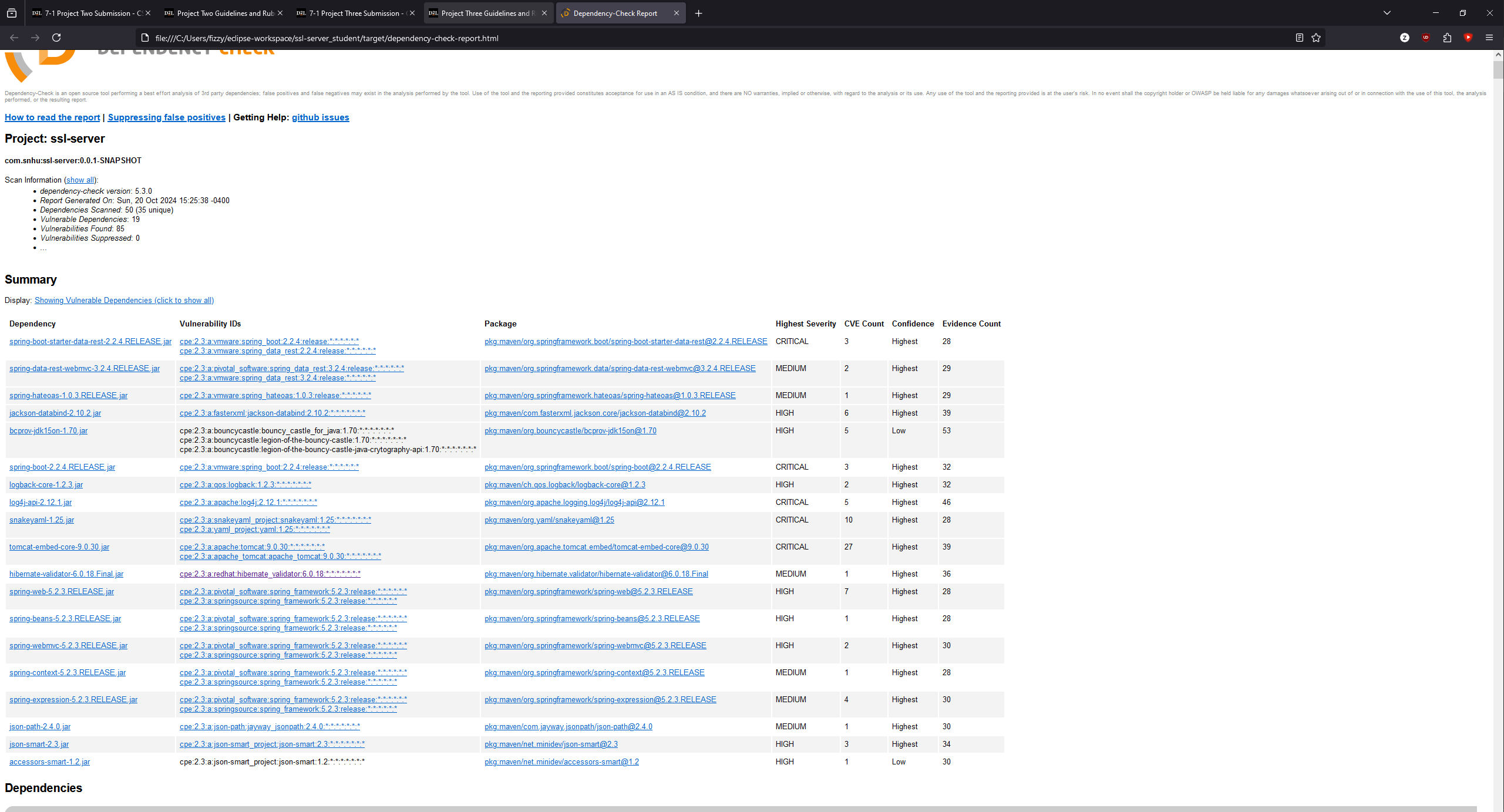


## Secondary Testing

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

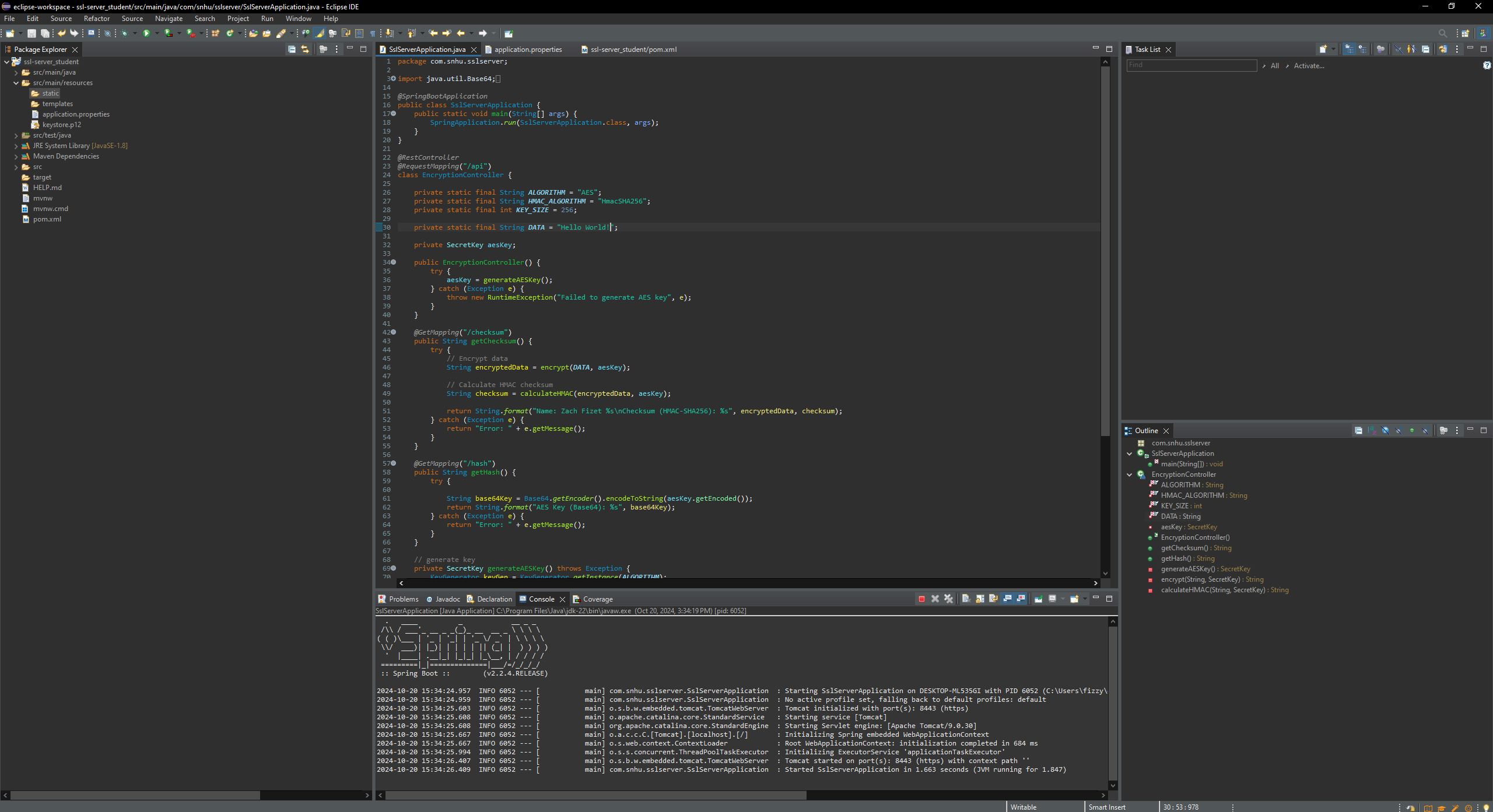


Dependency check report:



## Functional Testing

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



## Summary

The refactored code for this application uses several different types of security by implementing SSL encryption, secure key management, and proper error handling. These changes/additions to the code adhere to typical industry-standard best practices for secure coding and ensures compliance with certain requirements that Artemis Financial must follow as a banking institution. The code also underwent static testing, and to a lesser extent, a brief code review, which allow for identification of potential security threats and vulnerabilities within the code.

## Industry Standard Best Practices

The application uses multiple layers of security to ensure that it is secure in its function including network security with the implementation of HTTPS, secure coding practices, and certain runtime protections like input validation and error handling. By configuring HTTPS, the application has a reduced risk of main in the middle attacks and other attacks that could occur during data transmission. The application also follows industry standard best practices by using static testing via a dependency check to test for any potential vulnerabilities via a review of the dependency check report. Typically to address these vulnerabilities that appear within a dependency check, one would need to update the given dependency, if possible. In some cases, it is not possible to update a dependency, as it is possible that said dependency is no longer supported or updated. In this case, it would be best to potentially seek out other alternatives, if possible. Additionally, there is always the possibility of false positives appearing in your report, which is why it is a good practice to manually review each dependency that is flagged. The application also follows industry standard best practices by ensuring it is operating in compliance with certain requirements placed upon financial applications, as well as implementing certain risk mitigations to ensure data is protected and secure.