****

# Practices for Secure Software Report

**Table of Contents**

Practices for Secure Software Report 1

Document Revision History 3

Client 3

Instructions 3

Developer 4

1. Algorithm Cipher 4
2. Certificate Generation 5
3. Deploy Cipher 5
4. Secure Communications 6
5. Secondary Testing 7
6. Functional Testing 9
7. Summary 9
8. Industry Standard Best Practices 9

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **06/20/2024** | **Kenneth Wilkerson** |  |

## 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

Kenneth Wilkerson

## Algorithm Cipher

I conducted a review of Artemis Financial’s need to modernize its operations to use the most current and effective software security. Artemis Financial wants incorporate a file verification step in their web application. Since the web application transfers financial data, Artemis Financial will need a data verification step which can be achieved by using a checksum. I have determined that the RSA or Rivest-Shamir-Adleman algorithm with a key size of 2048 bits paired with SHA-384 (Secure Hash Algorithm) will be suitable. This will be important for Artemis Financial’s web application because it will be used to develop financial plans for customers such as savings, retirement, investments, and insurance. Therefore, protection of user data and privacy will be crucial due to the highly sensitive nature of financial data.

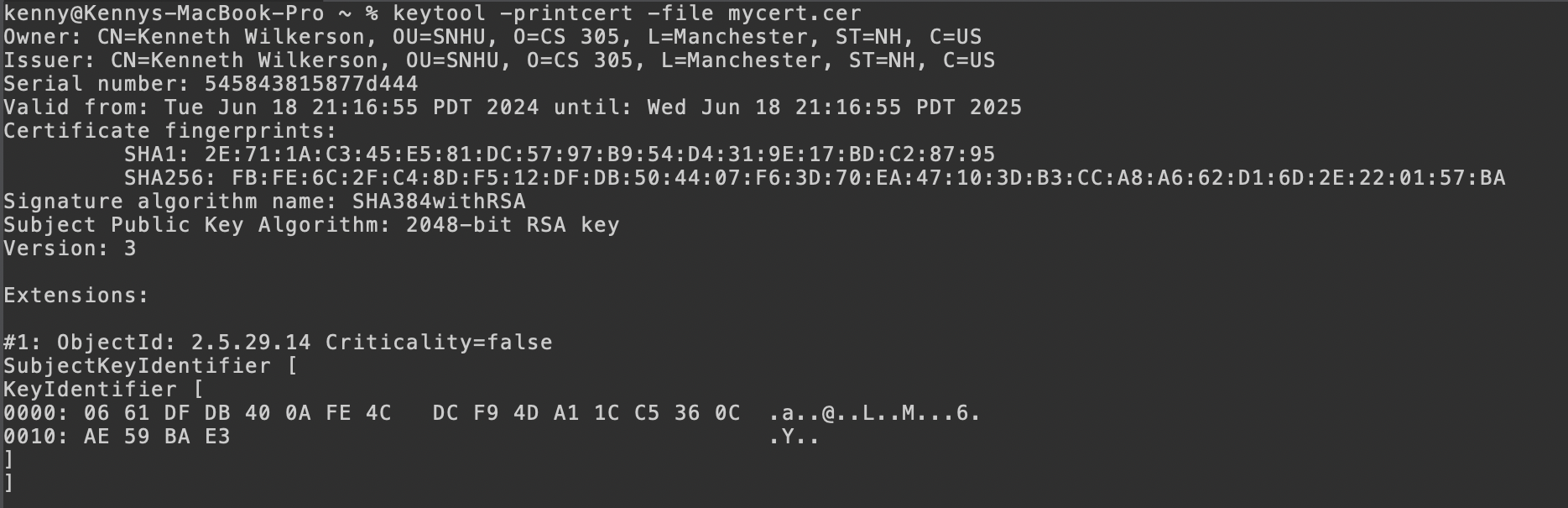
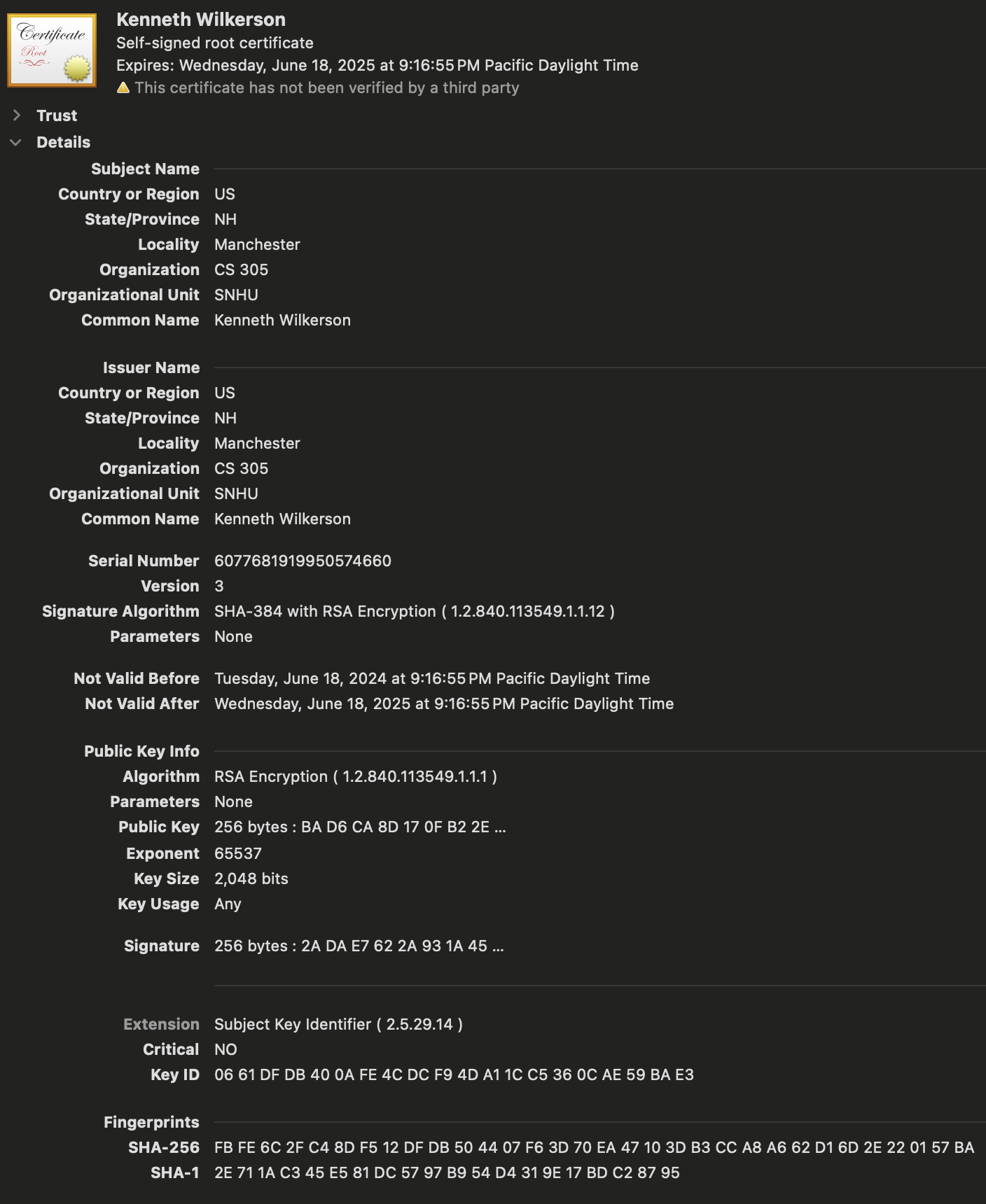
RSA was developed in 1977 and has become a commonly used standard algorithm. This algorithm is recommended by agencies such as the National Institute of Standards and Technology or NIST. The government uses RSA with a minimum 2048 bit key to protect classified information against potential attacks. In addition to the public sector, RSA is widely used in the private sector as well. RSA is useful in securing data as it moves between the client and the server (and vice versa). For example, RSA enables a HTTPS connection that utilizes encryption. It will protect a users account information ensuring the security of real-time updates and transactions. A 2048 bit key size is difficult to factor which is where it gets its strength from (larger number of bits results in larger keys). This complexity prevents bad actors from executing brute force attacks. This cipher reduces the risk of losses due to Artemis Financial’s web app becoming compromised. A 2048 bit key size balances security and performance.

In regards to hash functions, I recommend the usage of the SHA-384 which belongs to the SHA-2 family of functions developed in 2001 by the National Security Agency or NSA. SHA-384 creates a 384 bit hash value that serves as a signature of authenticity. The hash value is a 96 character hexadecimal string that can be used on a wide range of data. The chances of collisions with SHA-384 are extremely low. This is important because each piece of data retains a unique hash value. When the data is changed, a different hash value is generated. Therefore, you can trust with a great degree of certainty that no kind of data manipulation would lead to the the same hash value. Using RSA-2048 along with SHA-384 will provide a solid foundation for Artemis Financial’s web app.

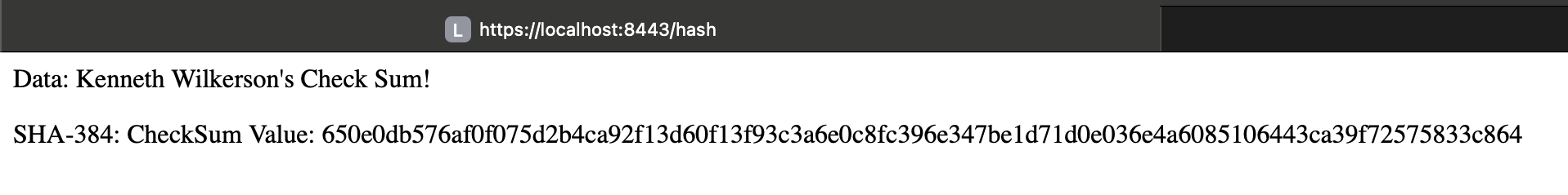
Random numbers are important in the creation of symmetric and asymmetric keys. The same key can be used for encryption and decryption with symmetric keys. An example of an algorithm that uses symmetric keys would be Advanced Encryption Standard or AES. On the other hand, asymmetric keys have two separate keys. A public key and a private key. Public keys are used encryption and private keys are used for decryption. The RSA cipher that I have recommended utilizes asymmetric keys. These keys will be useful in verifying the authenticity of a file. Currently, the algorithm cipher and hash function mentioned in this report have not been cracked and are strong options for secure software development. In the foreseeable future, they should remain effective.

## Certificate Generation

Insert a screenshot below of the CER file.

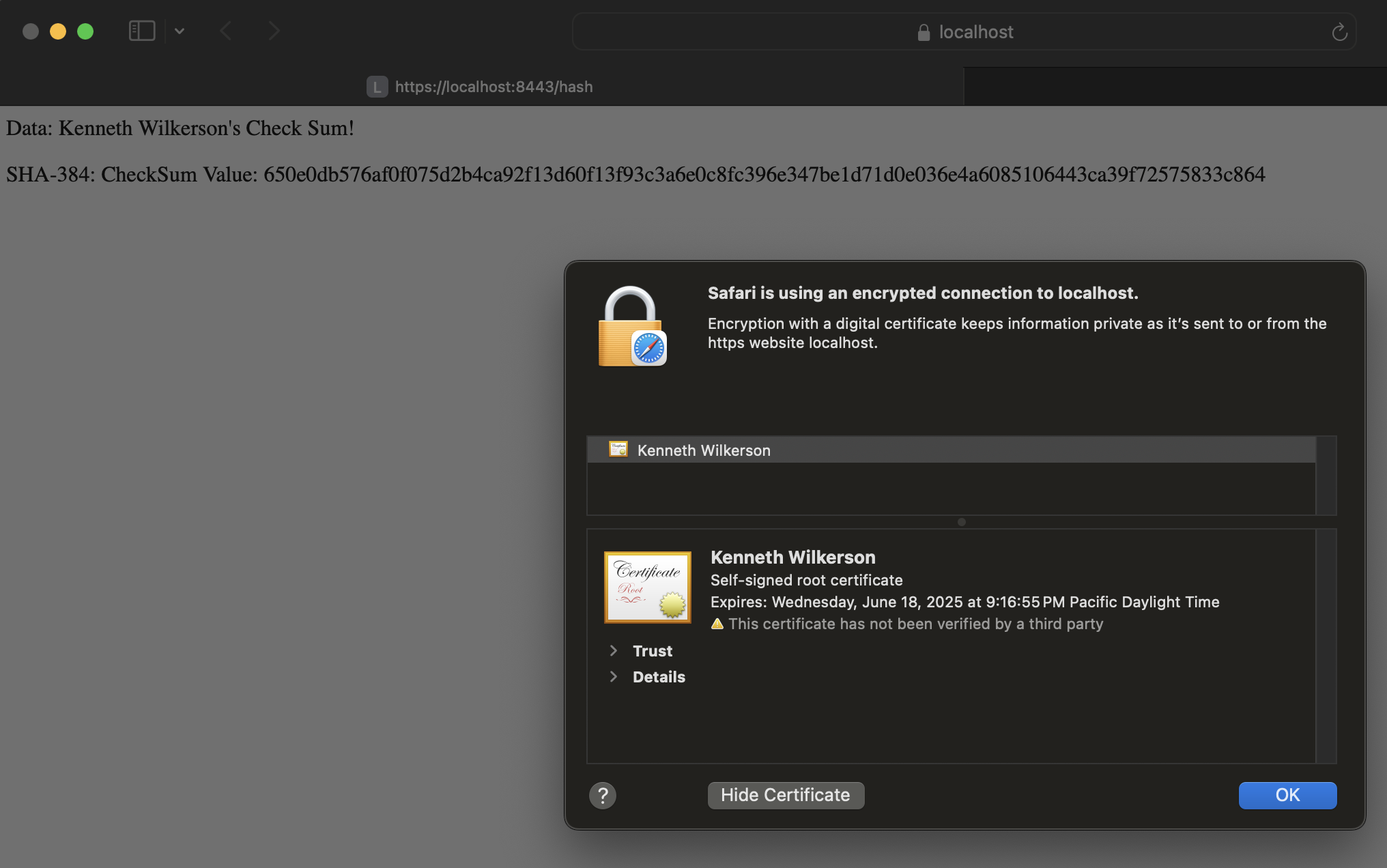
* Certificate in Eclipse Terminal
* Certificate from Keychain Access (Mac)

## Deploy Cipher

Insert a screenshot below of the checksum verification.

## Secure Communications

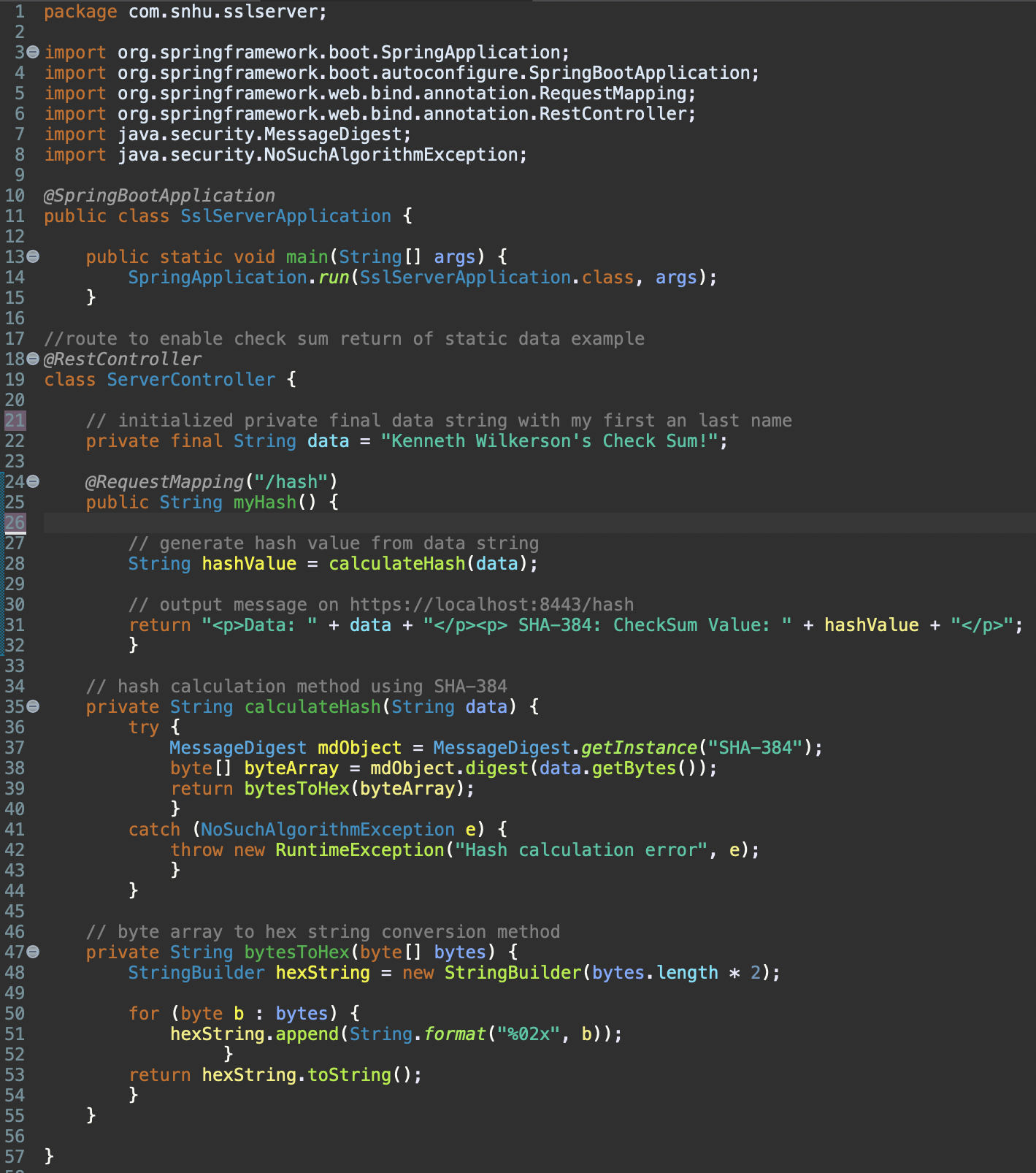
Insert a screenshot below of the web browser that shows a secure webpage.

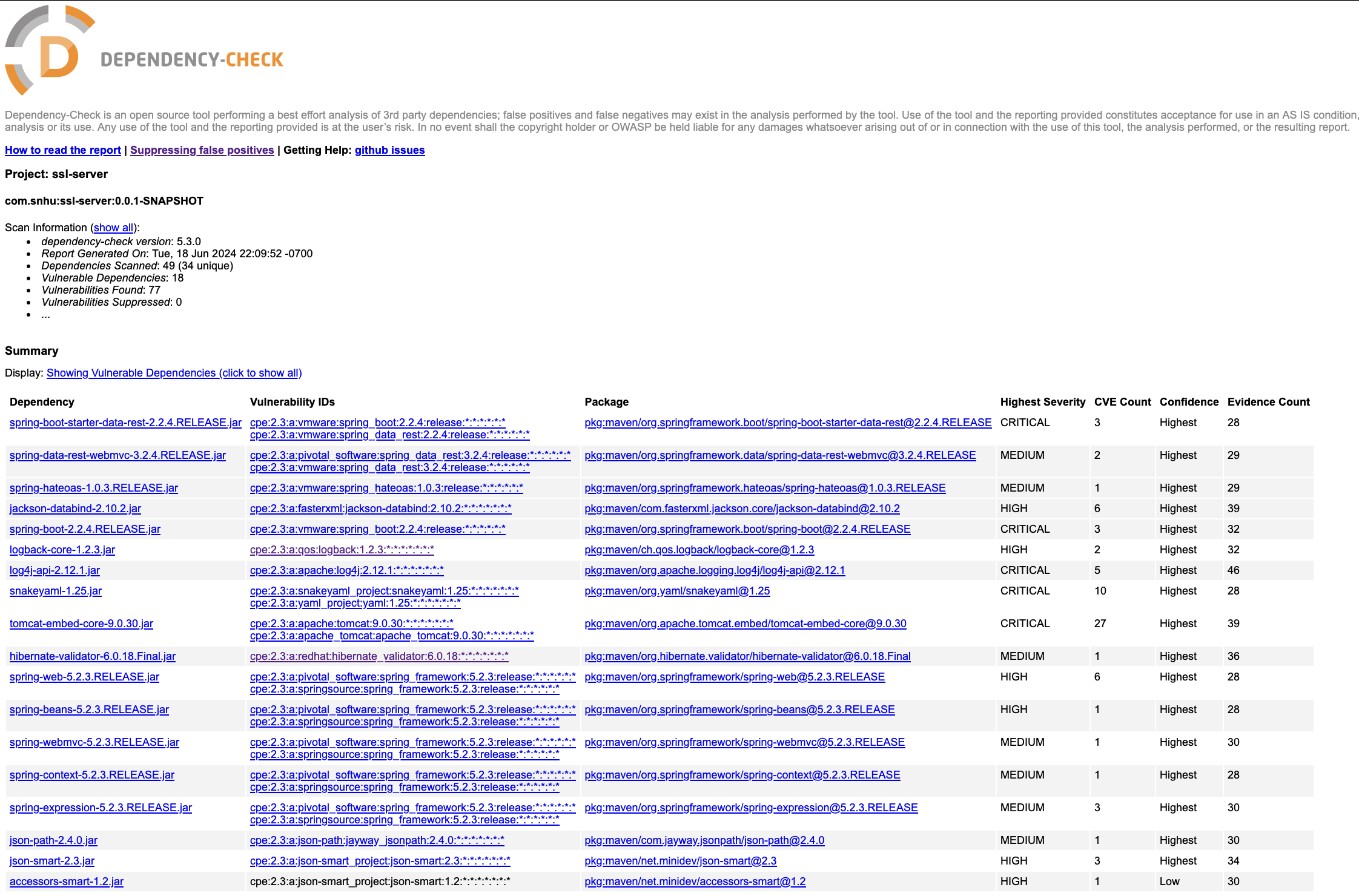


## Secondary Testing

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

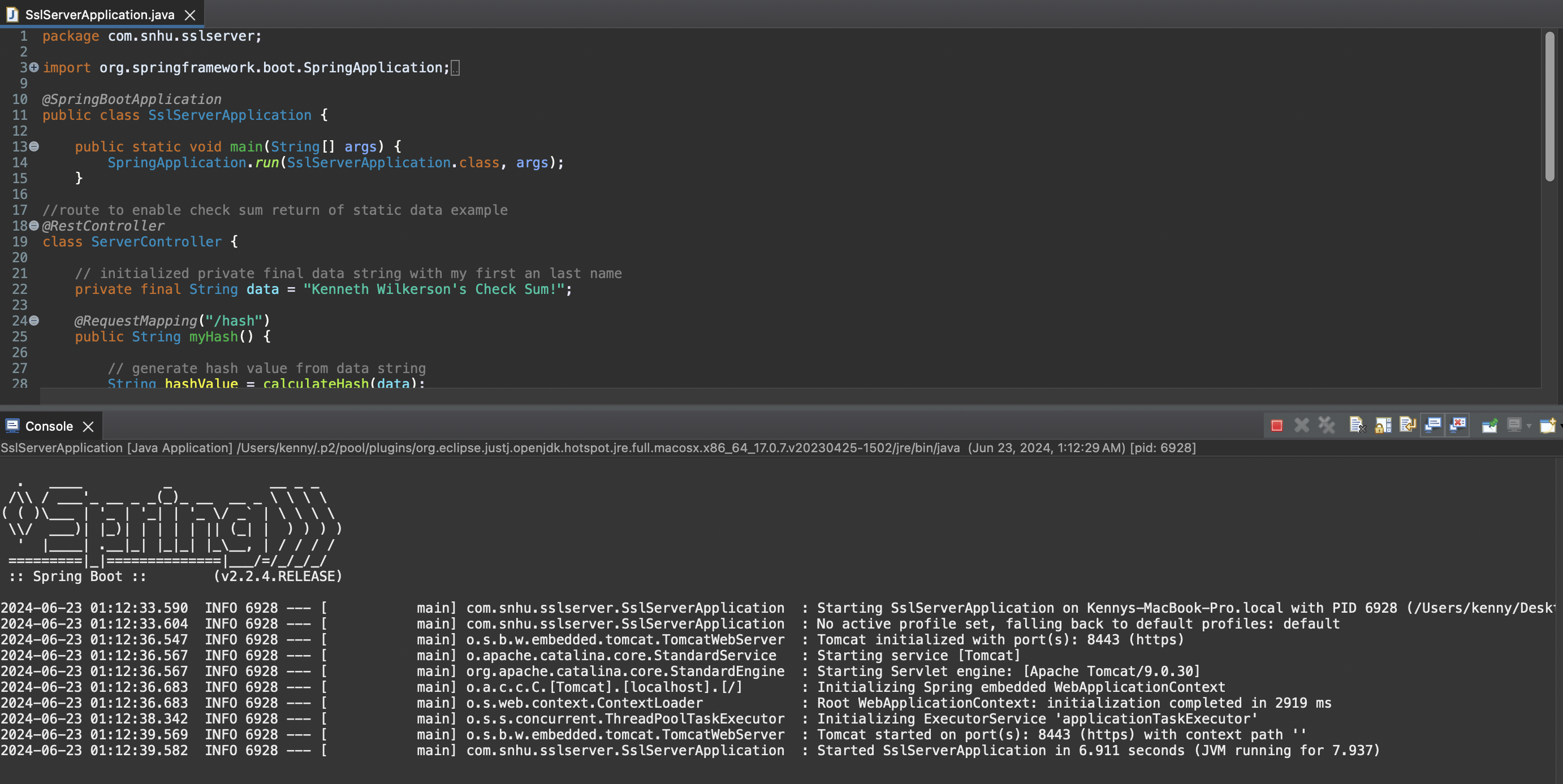
* Refactored Code



* Dependency Check (No additional vulnerabilities after refactoring code)

## Functional Testing

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

* Console Output (Refactored code executed without errors)

## Summary

The areas of security that were improved by refactoring the code is cryptography, code error and encapsulation. For cryptography, I helped secure data in transit and at rest. The RSA algorithm with 2048 bit keys helps to secure data in transit through a HTTPS connection. This helps prevents bad actors from stealing and/or tampering with data while it is being transferred between the client and server. The SHA-384 algorithm secures data at rest through generating hash values which helps to protect data integrity because there is a very low chance of collisions. I added a MessageDigest object on line 37 to add SHA-384 into the source code. In order to use MessageDigest, I had to import the MessageDigest library. Artemis Financial will benefit from these features because they can conduct file verification with any data as long as it is passed through the myHash method on lines 24-32.

Furthermore, code error is addressed with code refactoring. In order to address the NoSuchAlgorithmException that could happen when using the creating MessageDigest objects, I added a try-catch block on lines 36-43 to prevent a runtime error. I had to import the NoSuchAlgorithmException library in order to use it. This will prevent potential security vulnerabilities due to the unexpected behavior of the web app crashing. Lastly, encapsulation is addressed with the code refactoring. I added private fields and methods in order to prevent changes being made outside of the ServerController class on lines 18-55 which also improves security. For example the private final String field on line 22, the calculateHash method on lines lines 35-44 and the bytesToHex method on lines 47-54 all use the access type. Overall, the code refactoring reduces potential vulnerabilities and enhance security measures for Artemis Financial.

## Industry Standard Best Practices

While incorporating cryptography in the refactored code I made sure to incorporate important best practices. First, I made sure to use strong algorithms by using standard and recommended algorithms when choosing RSA 2048 and SHA 384. I also incorporated secure libraries when adding MessageDigest and NoSuchAlgorithmException because they are Java libraries which means they are a trustworthy source. In regards to code error, I incorporated exception handling in order to avoid runtime errors through using the NoSuchAlgorithmException in a try-catch block. Crashes could not only affect the performance of the app but also the security of the app. Lastly, encapsulation incorporates data protection and access control by preventing unauthorized access to fields and methods. It Is also important to make sure all libraries and dependencies are up to date.

For Artemis Financial, minimizing potential vulnerabilities reduces the damage that could be caused by bad actors. It also ensures that they are in compliance with different national and international laws. Important laws include the Gramm-Leach-Bliley Act, PCI-DSS and strong international privacy laws like in the EU. Data privacy is also good for the overall user trust in the web application which helps Artemis Financial maintain and/or improve their user base.

**Resources**

Cobb, M. (2021, November 4). *What is the RSA algorithm? definition from searchsecurity*. Security. https://www.techtarget.com/searchsecurity/definition/RSA#:~:text=The%20RSA%20algorithm%20(Rivest%2DShamir%2DAdleman)%20is%20the,an%20insecure%20network%20such%20as

Lake, J. (2022, February 17). *What is SHA-2 and how does it work?*. Comparitech. https://www.comparitech.com/blog/information-security/what-is-sha-2-how-does-it-work/#The\_history\_of\_SHA-2

Manico, J., & Detlefsen, A. (2015). *Iron-Clad Java: Building Secure Web Applications*. Chapter 6 Protecting Sensitive Data. Mc Graw Hill Education.

Oracle Help Center. (n.d.). Java security standard algorithm names. https://docs.oracle.com/javase/9/docs/specs/security/standard-names.html#cipher-algorithm-names