****

# 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** | **08/15/2024** | **Benjamin Sturgeon** | **Created document** |

## 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

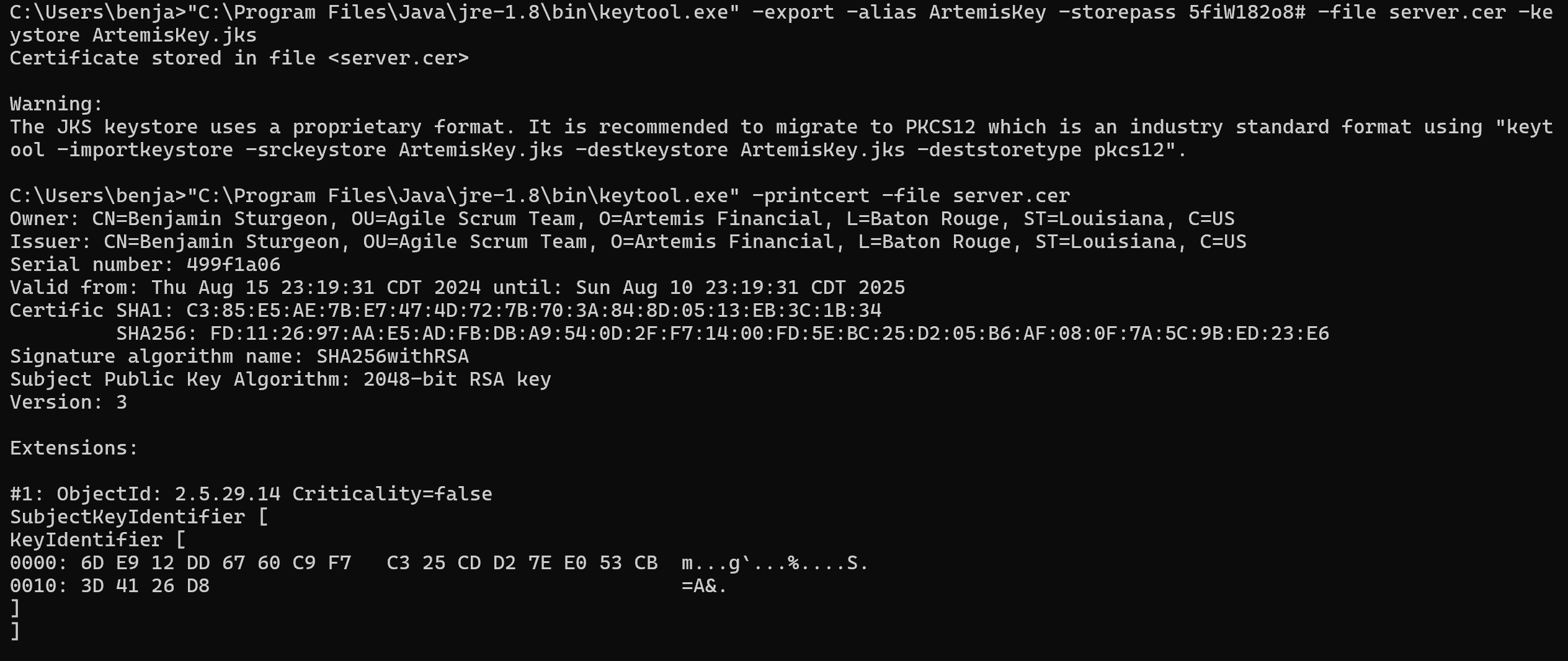
Benjamin Sturgeon

## Algorithm Cipher

There are two main ways of making data unreadable to a hacker: encryption and hashing. Encryption makes data unreadable but can be reversed to display the original plaintext (*Encryption vs hashing: Is hashing secure than encryption,* 2024). Not all encryption algorithms work the same. When data is encrypted a key is used to scramble and rearrange the data. A symmetric algorithm uses the same key for encryption as decryption. Some algorithms use a separate key for decryption, and these are known as asymmetric encryption algorithms (*Encryption vs hashing: Is hashing secure than encryption,* 2024).

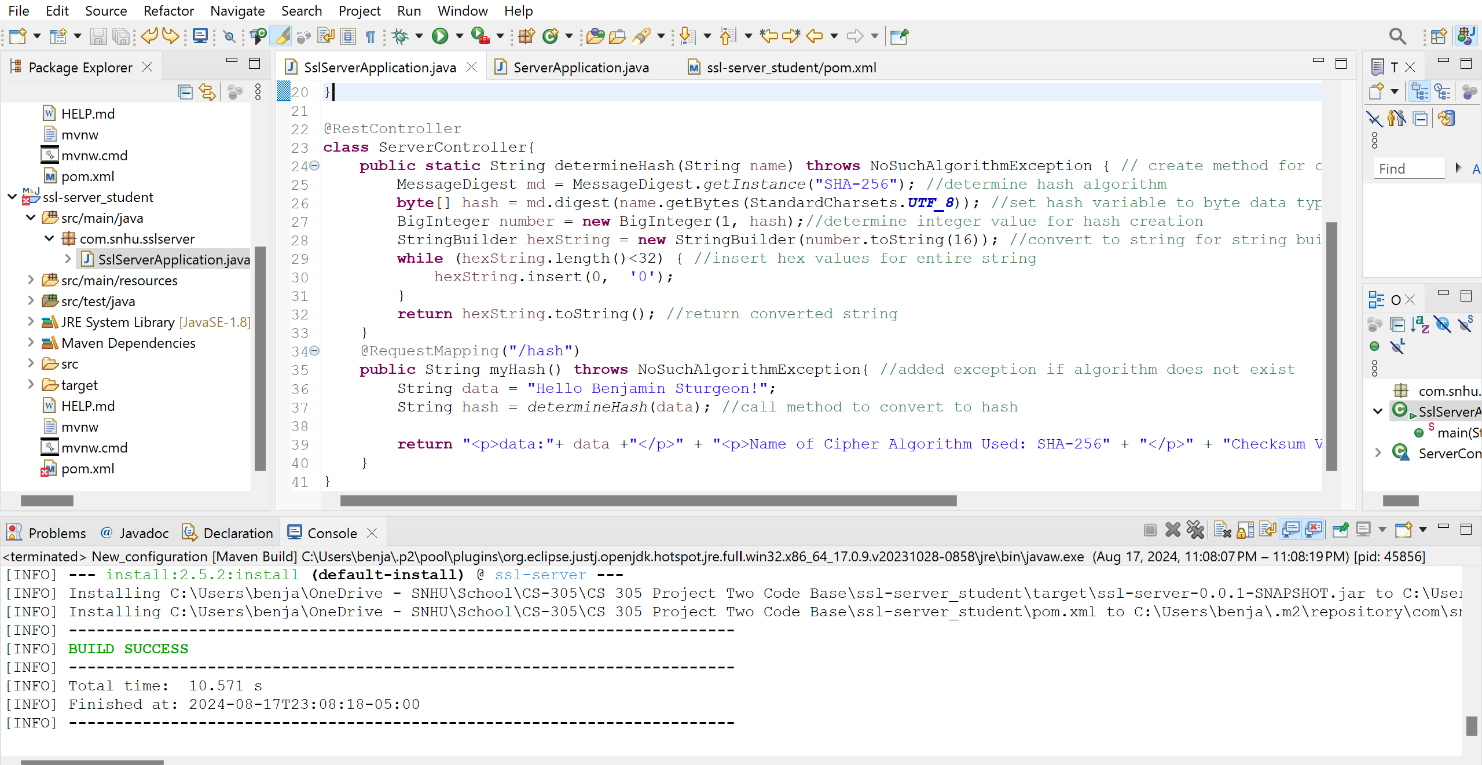
For this assignment a checksum validation is being implemented. This is a way to verify that the data is what is expected, and that no changes have been made. A value will be returned based on input, and if the input changes, the value should change as well. For this reason, I believe a hashing algorithm will be more suitable. SHA-256 uses mathematical functions to generate a hash based on user input. Any unexpected changes to the data will result in a different hash, allowing programmers to easily see evidence of malicious changes (Ballejos, 2024). There are two main benefits to SHA-256 that will benefit Artemis Financial: it’s speed and it’s security. SHA-256’s hashes are almost impossible to reverse (). This means that a hacker can’t derive the original data based solely on the hash value. SHA-256 is also incredibly fast meaning that computation won’t be stalled due to its implementation (Ballejos, 2024).

## Certificate Generation



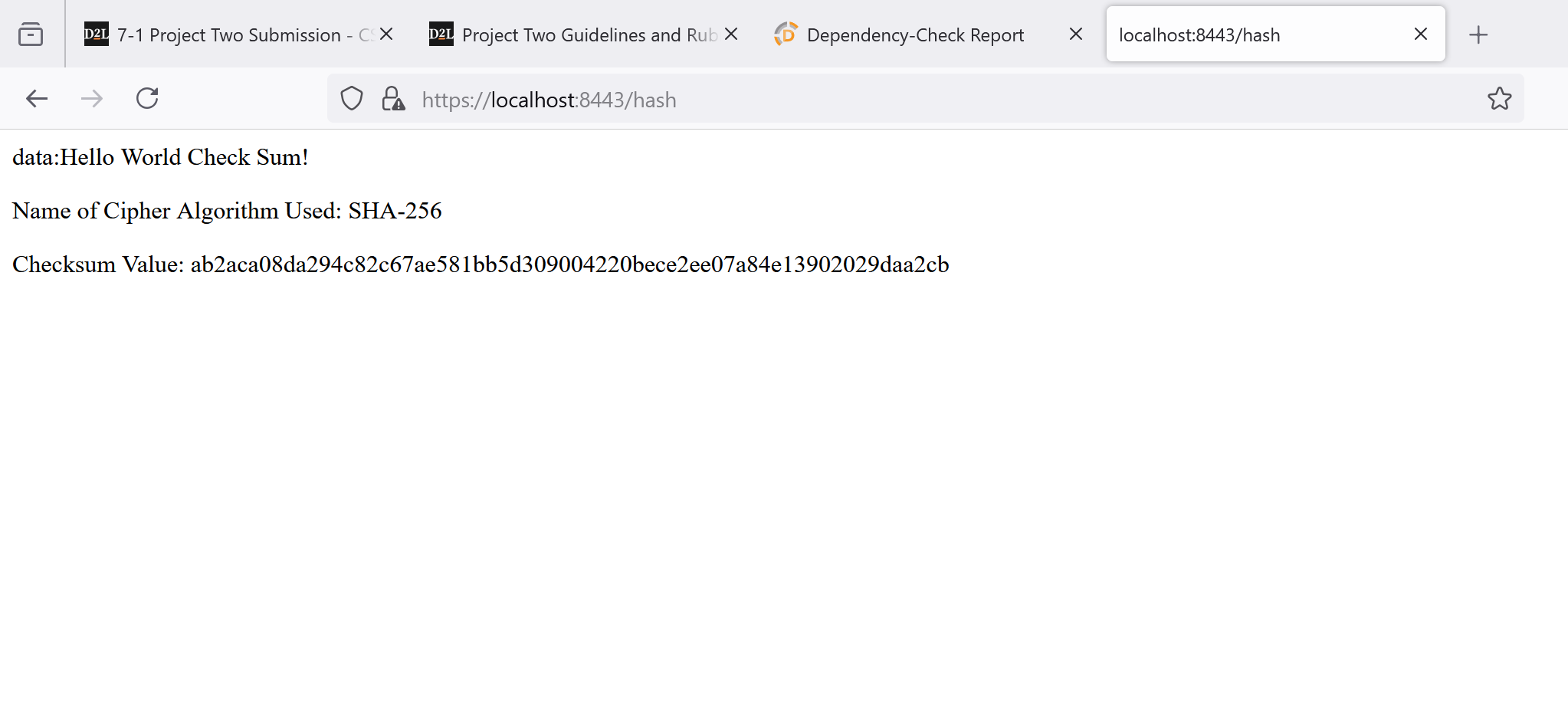
## Deploy Cipher

Insert a screenshot below of the checksum verification.



## Secure Communications

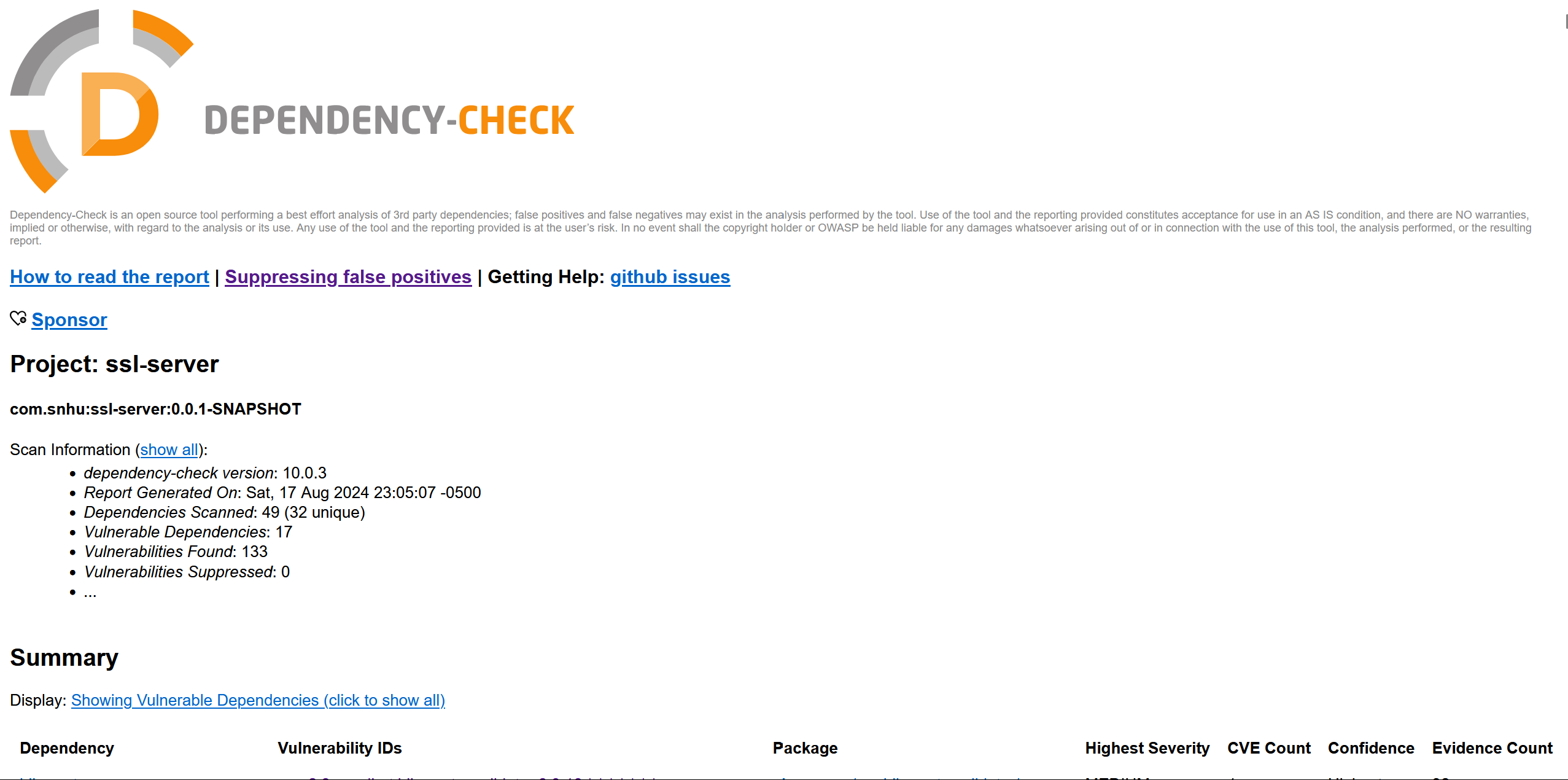
Insert a screenshot below of the web browser that shows a secure webpage.



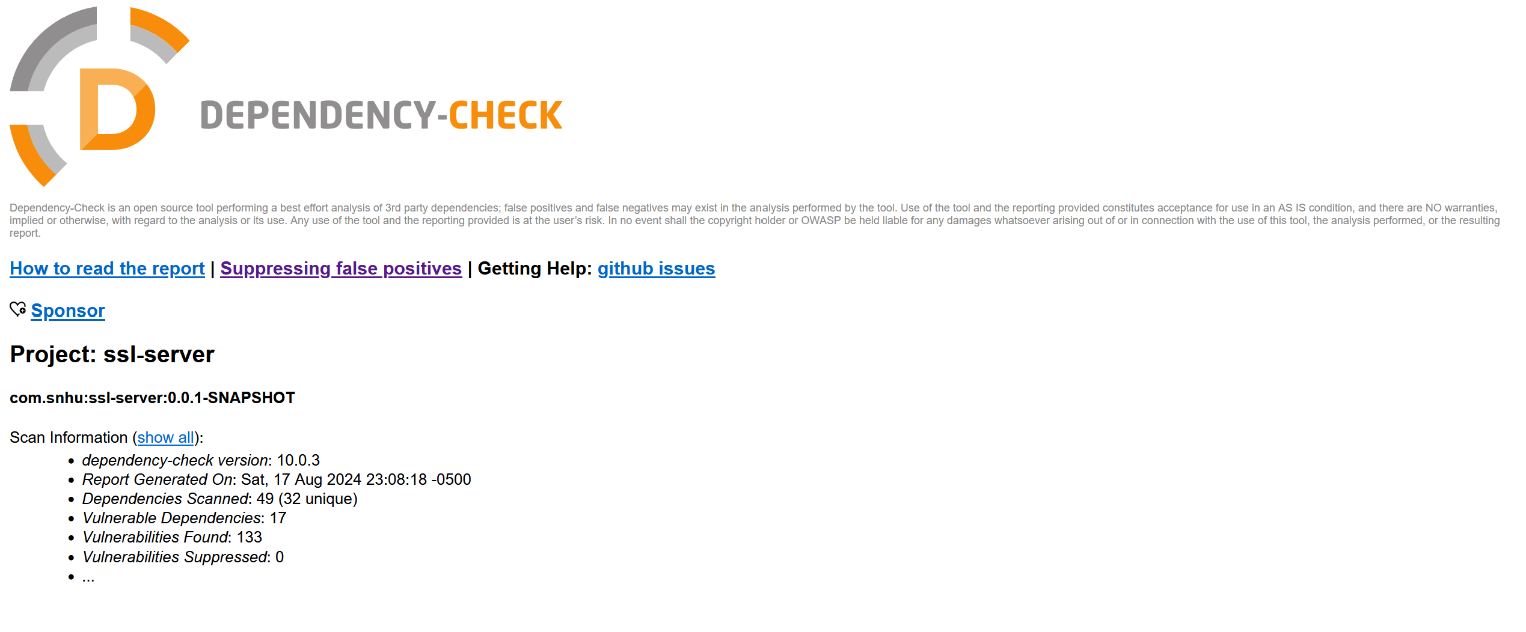
## Secondary Testing

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

Initial Screenshot Before Added Code:

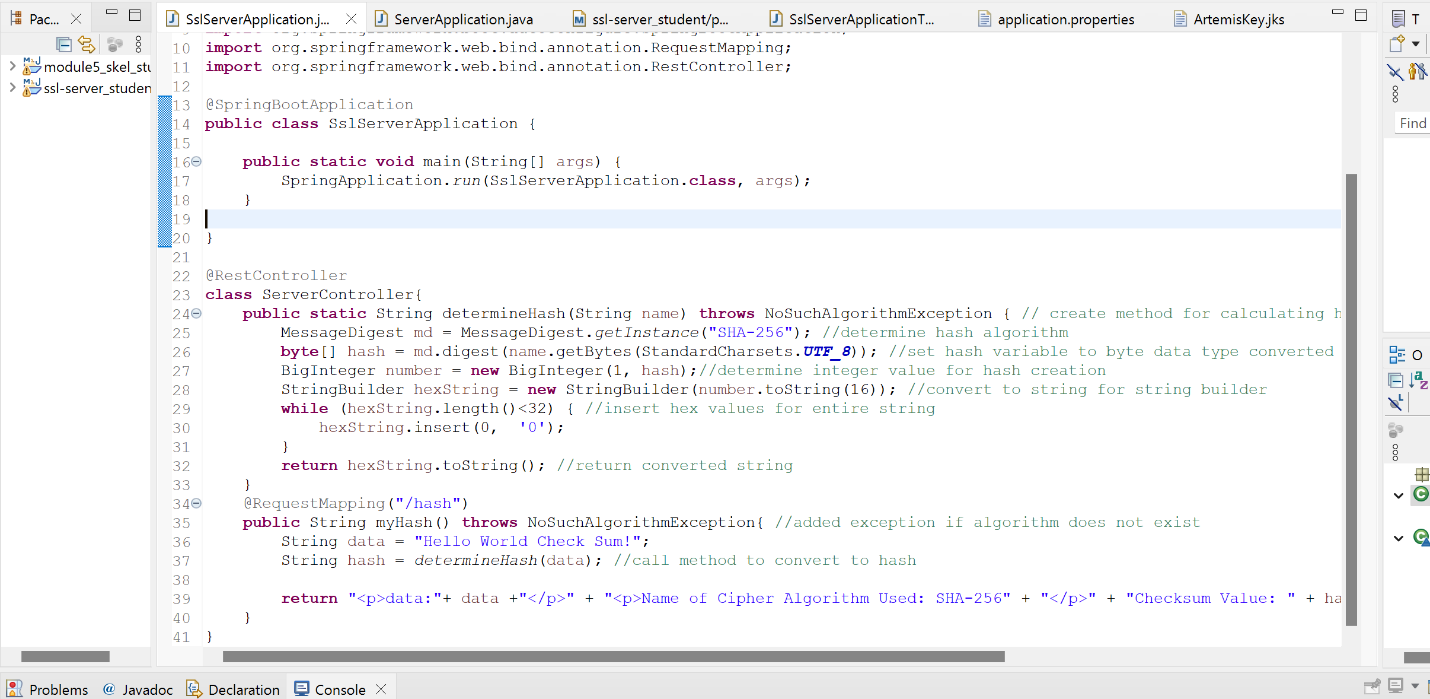


After Added Code:



## Functional Testing

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



## Summary

To begin I first looked at the information provided by the client. Knowing that this was a financial institution I tried to be mindful of how secure this system would need to be. A checksum validation would be needed to make sure no that no tampering to the application occurred. I used this information to decide on the best algorithm for the company’s needs. An encryption algorithm relies on the keeping the key secure, and given that the checksum data would never need to be decrypted this seemed like an unnecessary risk.

Next I wanted to make sure that the HTTPS protocol was implemented to make connection to the server more secure. To make sure that all data in transit is secure I created a certificate and added the information into the application.properties file. I checked to make sure that the web browser correctly displayed the connection as HTTPS. These two additions helped bring the principal of encryption into the program.

I needed to make sure that no errors were added, and so I compared my vulnerability assessment against the original assessment. No new errors were added. I performed a manual check last to verify that the code was working correctly, and that all best practices had been observed.

## Industry Standard Best Practices

When working on this project I tried to observe the best practices wherever possible. When implementing the checksum validation, I made sure to account for common errors, and to build throw statements into my new code. This makes it easier for future developers to troubleshoot errors that might arise as the program continues to be developed. I also added comments where able so that other developers could easily see the intended functionality behind each method.

When reviewing the code for errors I performed both a manual and static check for potential vulnerabilities. Using both methods can help make sure that no errors slip through the cracks and remain. HTTPs is a very standard protocol, and implementing it brings the server’s security more in line with what’s expected.

Citations:

Encryption vs hashing: Is hashing secure than encryption. Encryption Consulting. (2024, May 2). <https://www.encryptionconsulting.com/education-center/encryption-vs-hashing/>

Ballejos, L. (2024, July 1). *What is SHA-256?: Definition & overview*. NinjaOne. https://www.ninjaone.com/it-hub/endpoint-security/what-is-sha-256/