****

# 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** | **10/23/2022** | **Earl Calkins** |  |

## 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

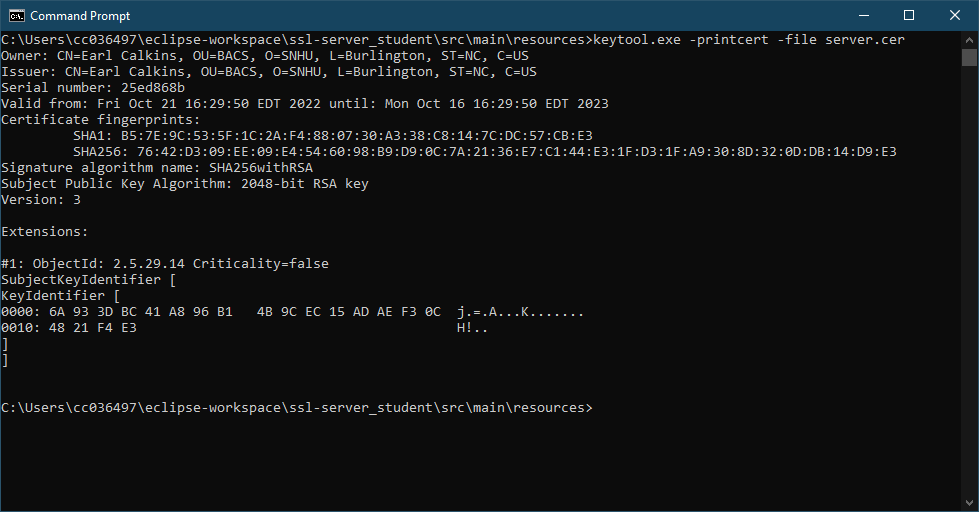
Earl Calkins

## Algorithm Cipher

Since the checksum verification will be performed using the MessageDigest class, I recommend using the more secure of the available algorithm ciphers, which would be SHA-256. SHA-256 has increased security due to its 256-bit hash value and output size. Since the has function is irreversible, potential attackers will be unable to decrypt any messages passed through it. The SHA-256 algorithm starts by appending padding bits to the initial message to make it 64 bits less than a multiple of 512. After that, additional bits of 64 bits in length are added to make the entire message a multiple of 512. 64 different key values are then initialized to be used in computing the hash value. Finally, the message is broken down into a number of chunks of 512 bits and each bit goes through 64 rounds of operation, with the output of each round being used as the input of the next. Just using this description of how the algorithm works, we can see that it is very thorough and secure.

## Certificate Generation

Insert a screenshot below of the CER file.



## Deploy Cipher

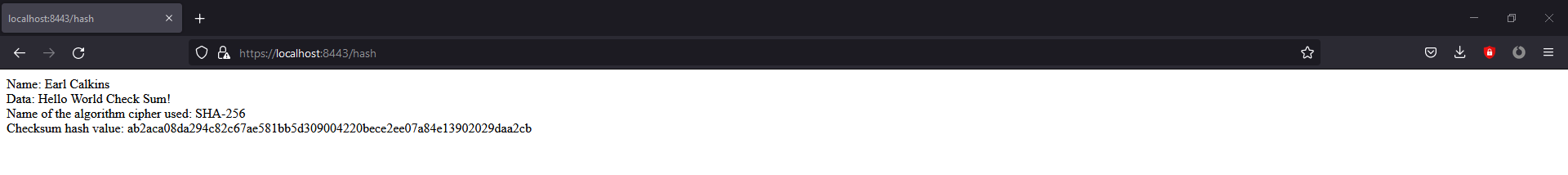
Insert a screenshot below of the checksum verification.

Text

Description automatically generated

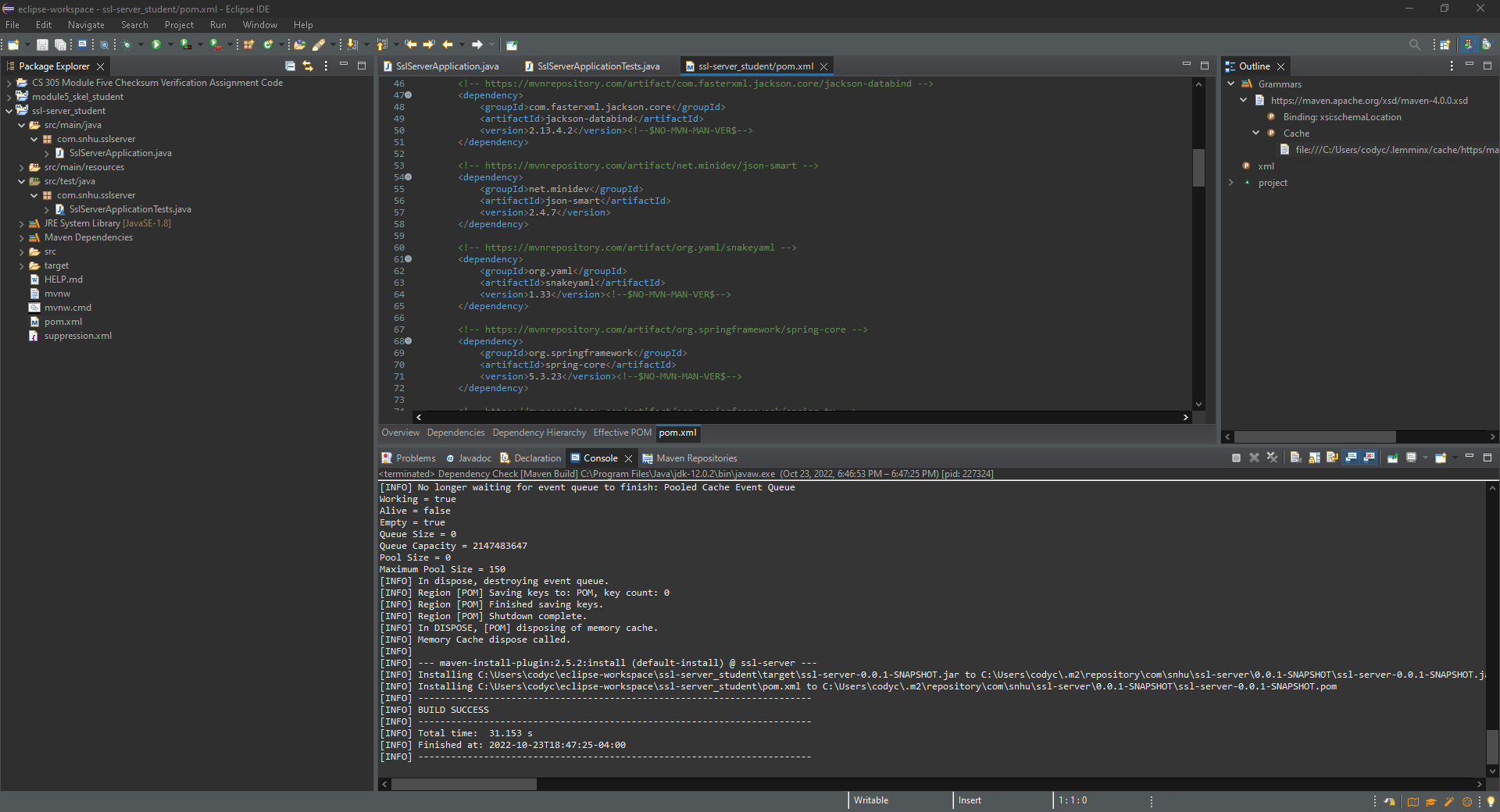
## Secure Communications

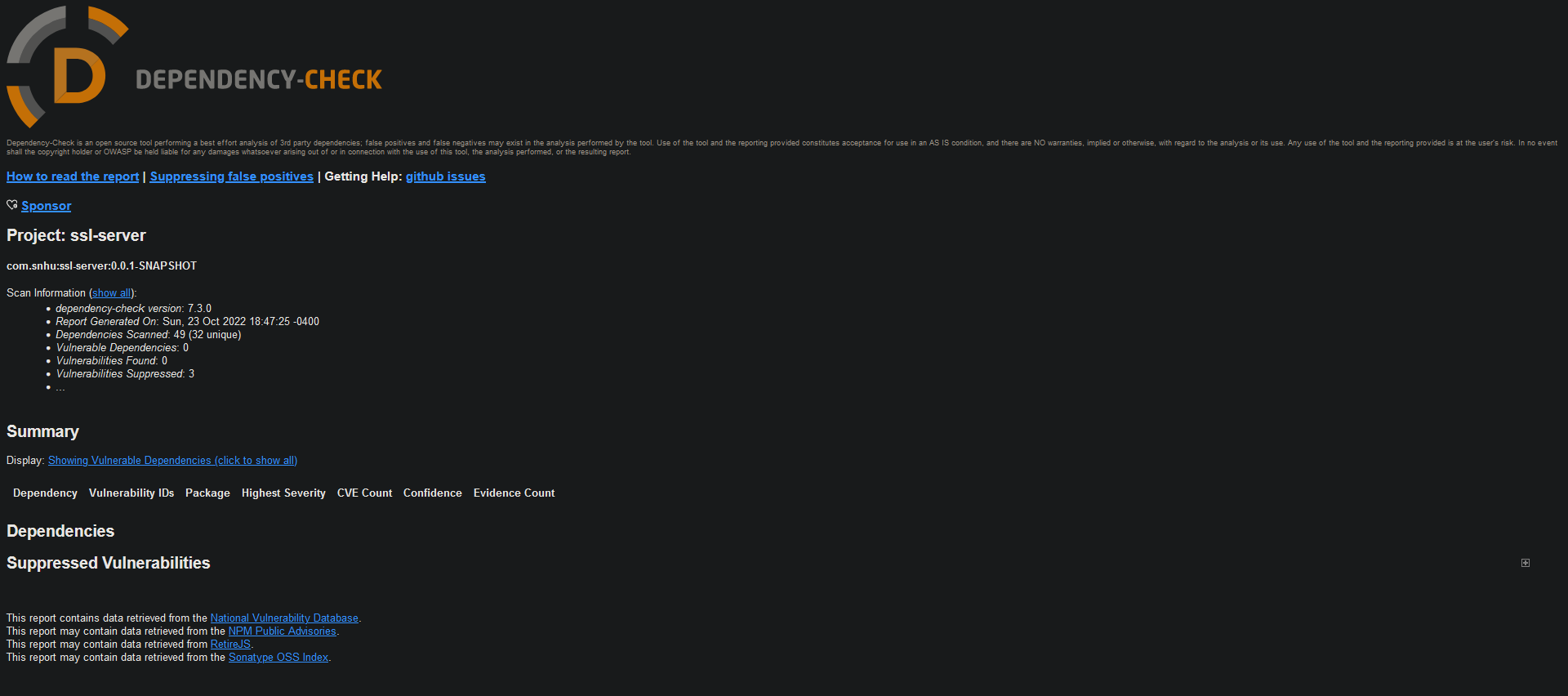
Insert a screenshot below of the web browser that shows a secure webpage.



## Secondary Testing

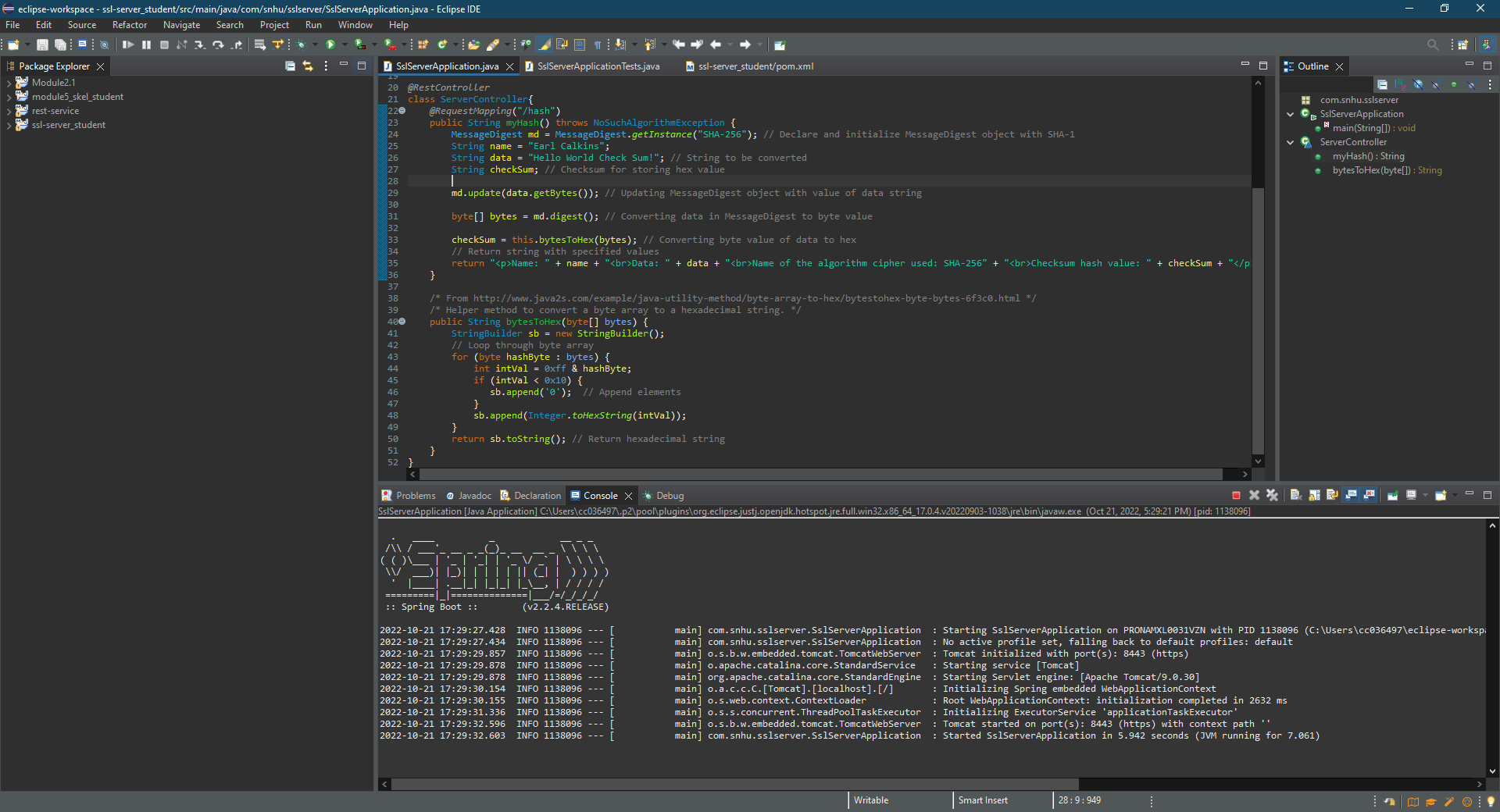
Insert screenshots below of the refactored code executed without errors and the dependency-check report.





## Functional Testing

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



## Summary

The original code bas has been refactored to include a secure checksum verification using the MessageDigest class and SHA-256 algorithm cipher. Additionally, any vulnerable dependencies were added to the POM file with their most up-to-date uncompromised versions. The specific areas of security that the code refactoring has addressed are APIs, Cryptography, Code Quality, and Encapsulation. Since there are no user interactions included in the program currently, attacks of that nature are not a security issue.

## Industry Standard Best Practices

I used industry standard best practices in maintaining the application’s current security by ensuring there were no issues in the refactored code. I made sure there was no room for attackers to interact with the system and that there was a secure algorithm cipher to support it. Applying industry standard best practices is very valuable when it comes to the nature of the client’s business. Since the client is a financial institution, it is very important that there is no way an attacker can go after their customers’ data.

**References**

Anand, A. (2019, November 27). *Breaking down : Sha-256 algorithm*. Medium. Retrieved October 23, 2022, from https://infosecwriteups.com/breaking-down-sha-256-algorithm-2ce61d86f7a3