****

# CS 305 Project Two

**Practices for Secure Software Report**

**Table of Contents**

[Document Revision History 3](#_heading=h.1fob9te)

[Client 3](#_heading=h.3znysh7)

[Instructions 3](#_heading=h.2et92p0)

[Developer 4](#_heading=h.tyjcwt)

[1. Algorithm Cipher 4](#_heading=h.3dy6vkm)

[2. Certificate Generation 4](#_heading=h.1t3h5sf)

[3. Deploy Cipher](#_heading=h.4d34og8) 5

[4. Secure Communications](#_heading=h.2s8eyo1) 6

[5. Secondary Testing](#_heading=h.17dp8vu) 6

[6. Functional Testing](#_heading=h.3rdcrjn) 9

[7. Summary](#_heading=h.26in1rg) 10

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **10/5/2021** | **Nathan Chuluda** | **Initial Document** |

## Client



## Instructions

Deliver this completed Practices for Secure Software Report documenting your process for writing secure communications and refactoring code that complies with software security testing protocols.

Respond to the steps outlined below and replace the bracketed text with your findings in your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

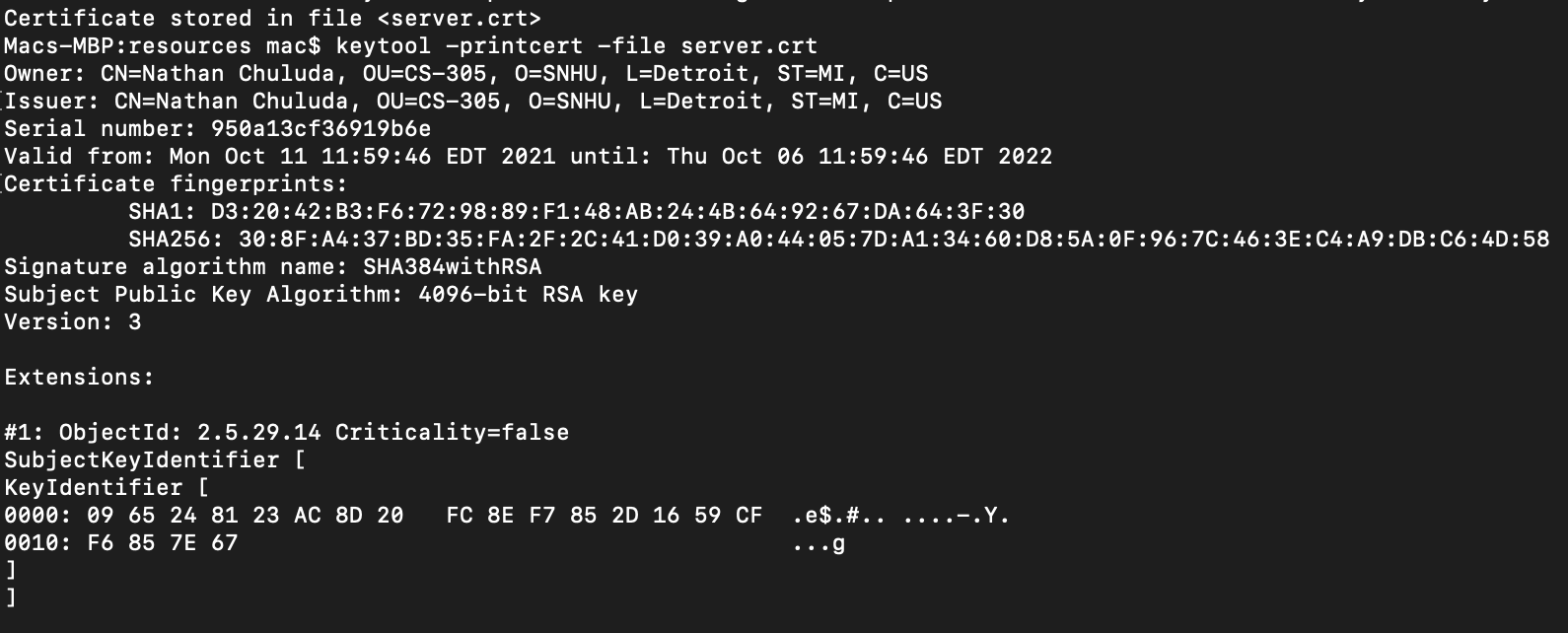
## Developer

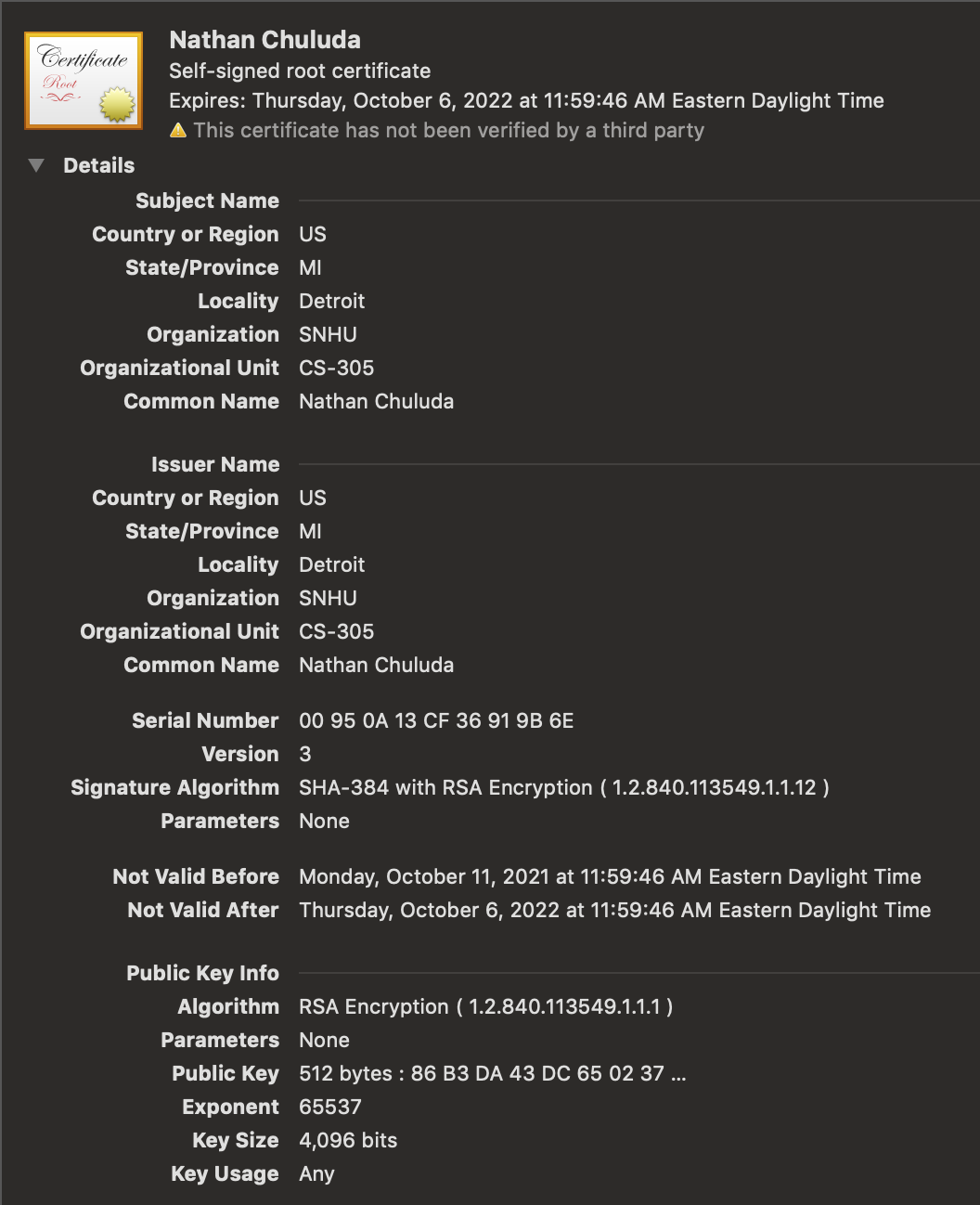
Nathan Chuluda

## 1. Algorithm Cipher

In order to verify the integrity of data transmitted using Artemis Financial’s system, the SHA-256 hash algorithm should be used. After downloading a file, the hash can be compared to that of the originating file that exists on the server. If this hash matches that of the original file, one can be assured that the data has not been corrupted or altered during transfer. This hash algorithm is only used in one direction, and no matter the size of the inputted data, will always return 256-bits or 64 characters. This means that given only the hash, it is not possible to revert back to the original file. The value of this algorithm is that it is virtually impossible to alter the input file while maintaining the same hash output. At this time SHA-256 is resistant to collision attacks.

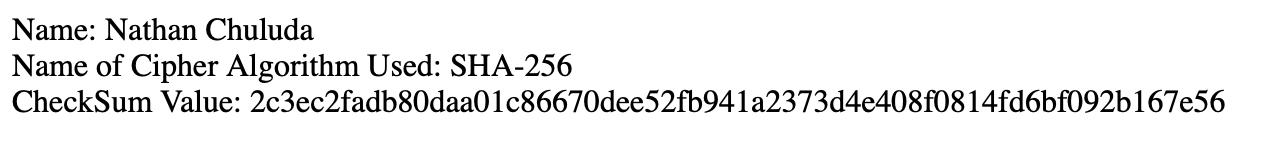
## 2. Certificate Generation





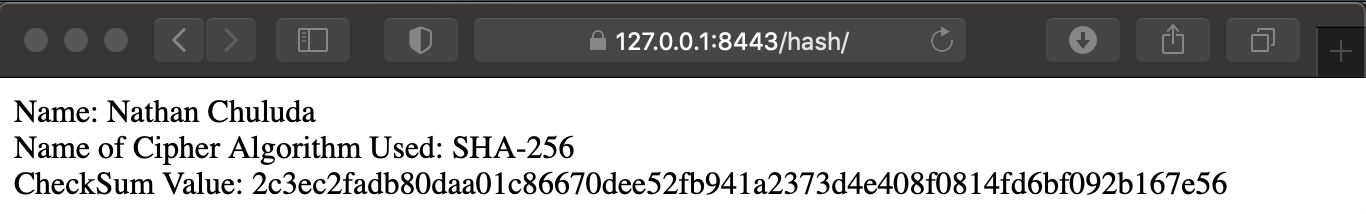
## 3. Deploy Cipher

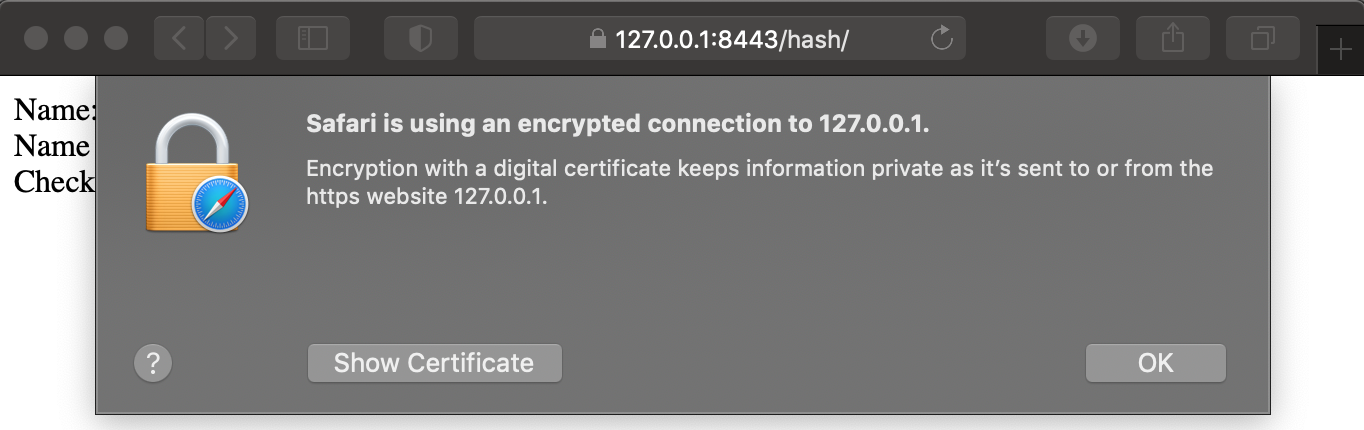
Refactor the code and use security libraries to deploy and implement the encryption algorithm cipher to the software application. Verify this additional functionality with a checksum.



## 4. Secure Communications

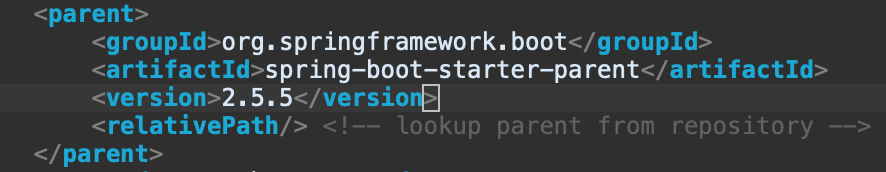
Refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code to verify secure communication by typing **https://localhost:8443/hash** in a new browser window to demonstrate that the secure communication works successfully.





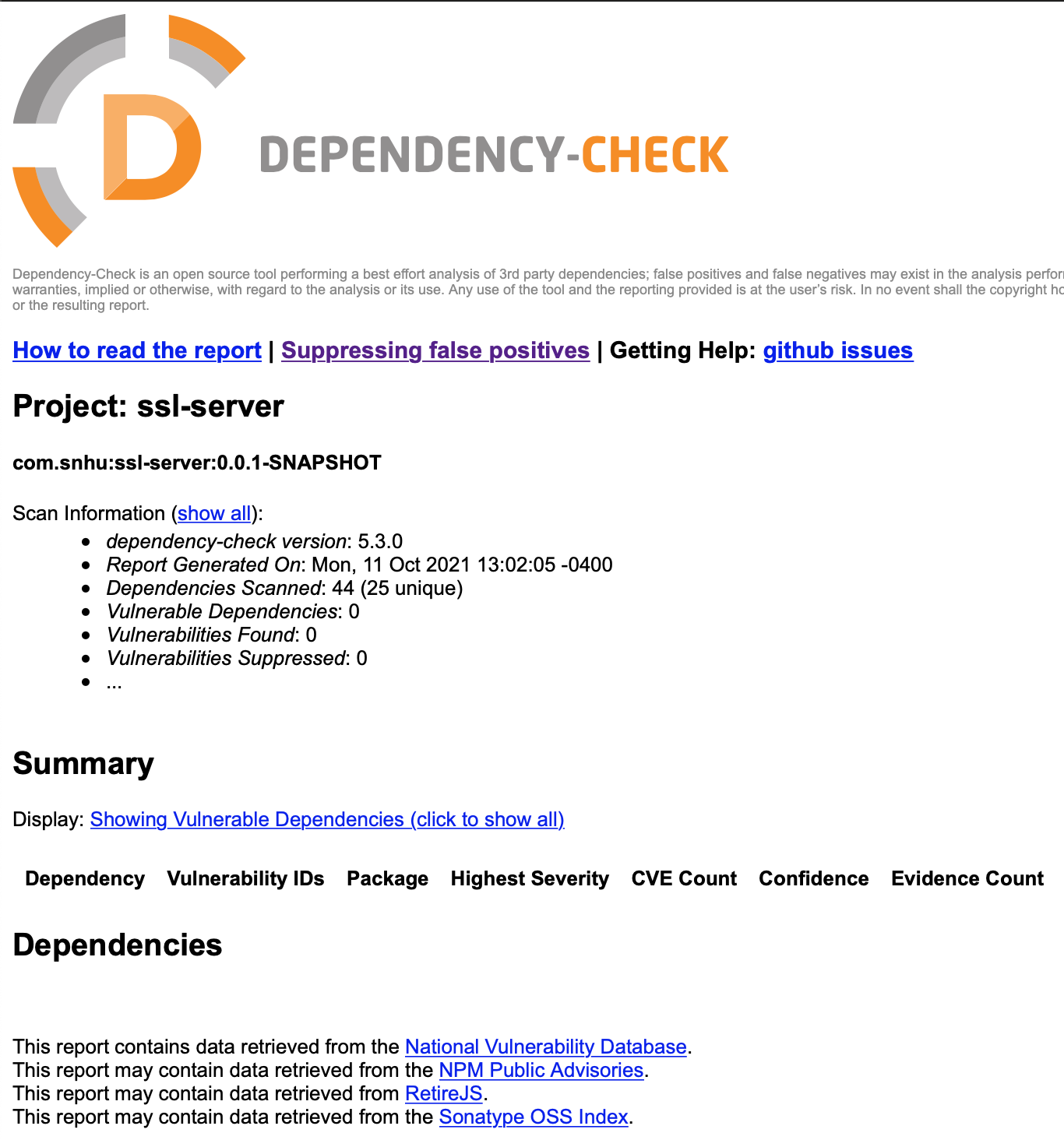
## 5. Secondary Testing

I updated the spring-boot-starter-parent to the newest version 2.5.5 from September 2021 to resolve the vulnerabilities listed in the dependency check:

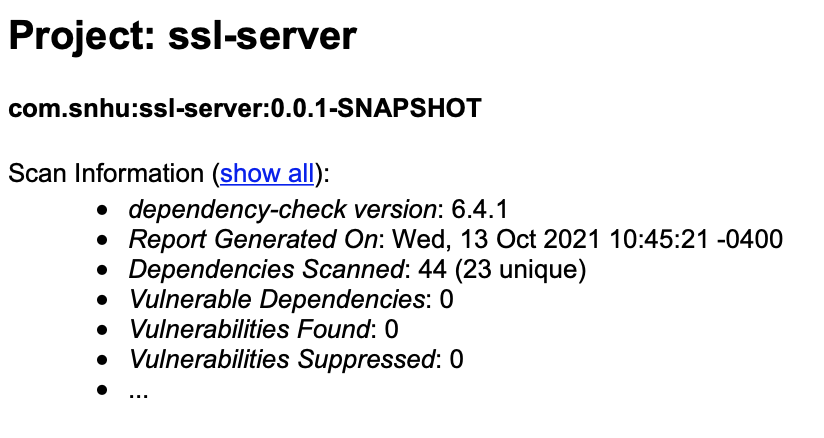


The refactored code in SslServerApplication.java does not introduce any further vulnerabilities and executes without error:



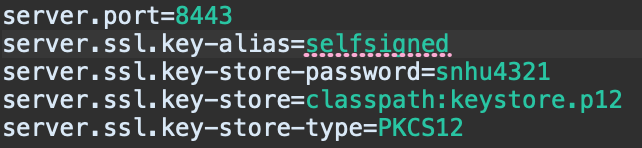


Although it appears that this has resolved the vulnerabilities discovered in our initial dependency check, this dependency check was run using an outdated version. The original code specified to use version 5.3.0 of dependency-check-maven, but the most current version is 6.4.1 released October 2021. For good measure, I’ve updated this to the newest version and run an additional dependency check report.



## 6. Functional Testing

With this refactor, I updated the application.properties file so that the https protocol would be used. This is an important step towards securing the project. Before deploying the final version of this application, it is important that the self-signed certificate is replaced with one issued from a recognized certificate authority. The self-signed certificate is appropriate for development purposes, but is not a final solution and will require an update at a later time.



There is currently only a single endpoint for this API defined for “/hash/”. This is a static endpoint that does not accept user input. Even so, there is currently no error handling in place should a user try to reach an undefined endpoint. A custom error message should be implemented and all requests leading to the error should be logged.





## 7. Summary

In this refactor, the code has been updated in several ways to improve security for Artemis Financial. First, the SHA-256 hash algorithm has been implemented. Currently, this code only provides the hash of a static string, but could be expanded later to provide the hash of any files transferred by the system. This will allow users to ensure that files have not been corrupted or altered in the process of being transferred.

Next, the code has been updated so that the https protocol is used in place of the regular http protocol. A self-signed certificate is being used as a placeholder during development, but should be replaced with a certificate issued from a certificate authority before release. This certificate can help users ensure that they are interacting with the authentic Artemis Financial website. In addition, the https protocol can be used to encrypt data transferred between client and server. This is a crucial security measure to take for a financial institution that is transferring sensitive private and financial information. Without using this secure protocol, data such as login credentials could be intercepted by attackers.

Lastly, attention was taken to the dependencies in use by the code. In this case, it was sufficient to update the spring framework and maven dependency check to the newest version so that previous vulnerabilities are resolved.

Moving forward, these and other security factors should be considered as changes are made to the project.

Input validation: There is currently no user input to validate. When this is added later, it is important that the input be validated before being accepted and processed by the system.

APIs: This API currently only has a single endpoint. When more functionality is added, care should be taken to validate input and ensure that the amount and type of data returned is appropriate.

Cryptography: This code uses the SHA-256 hashing algorithm as a way to verify the integrity of data. Later, once Artemis Financial stores personal information, an encryption method for storing data at rest should be used. This would utilize a different cipher algorithm. AES-128 would be an appropriate choice to implement to protect this sensitive information.

Client/server: This code is an example of client server architecture, and should be developed with appropriate security in mind.

Code error: Further error handling should be implemented in this code. Currently the default error message is displayed to users if they try to access anything other than the “/hash/” endpoint.

Code quality: All code should be developed following best patterns and practices to ensure it stays secure.

Encapsulation: When the code is updated to include classes, they should be designed in a way that uses encapsulation to limit who can access this data.