****

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[Document Revision History 3](#_Toc33111302)

[Client 3](#_Toc33111303)

[Instructions 3](#_Toc33111304)

[Developer 4](#_Toc33111305)

[1. Algorithm Cipher 4](#_Toc33111306)

[2. Certificate Generation 4](#_Toc33111307)

[3. Deploy Cipher 4](#_Toc33111308)

[4. Secure Communications 4](#_Toc33111309)

[5. Secondary Testing 4](#_Toc33111310)

[6. Functional Testing 5](#_Toc33111311)

[7. Summary 5](#_Toc33111312)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **6/20/2021** | **Nicole Penner** | **First full report on secure communication** |

## 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:

**Nicole Penner**

## 1. Algorithm Cipher

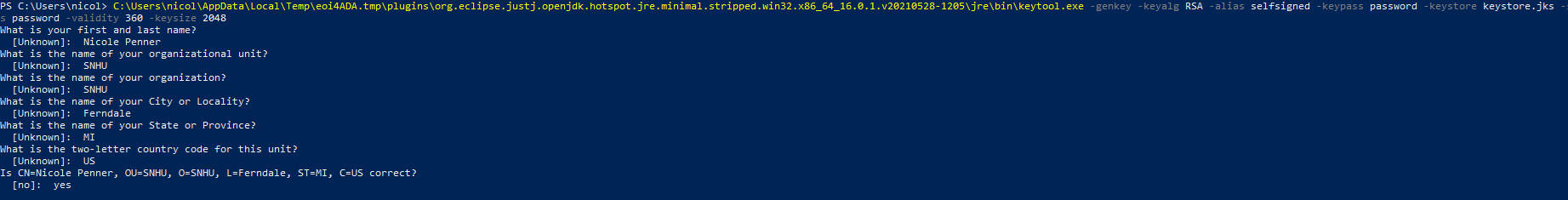
Determine an appropriate encryption algorithm cipher to deploy given the security vulnerabilities, justifying your reasoning. Be sure to address the following:

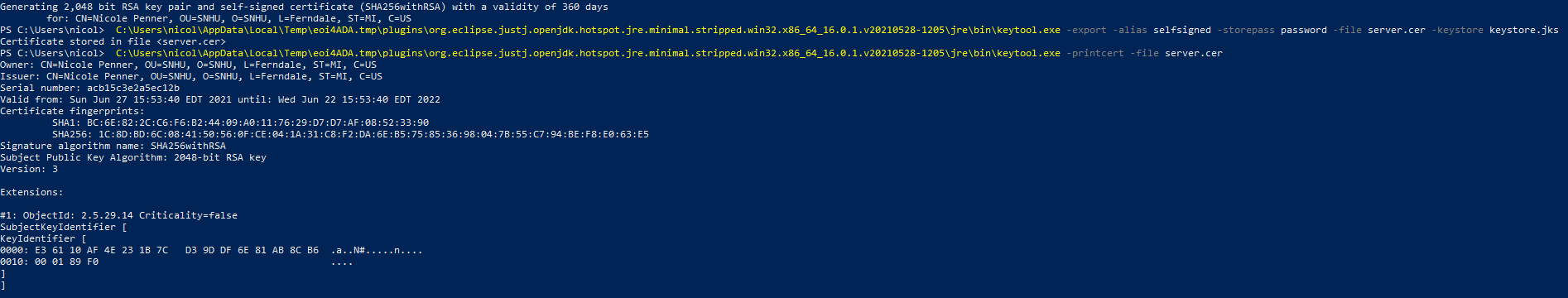
Secure communication within a company, as well as outside communication with users’ needs to be protected. This is extremely important to help ensure that both consumer and company data remains as secure and protected as possible. Since Artemis Financial specifically deals with sensitive information of clients and for the company as a whole, the highest degree of security is paramount. I am recommending and implementing the “SHA-256” cipher algorithm. It will take longer to encrypt files but since a 256-bit form is used, it has an extremely long key which will keep information and data as safe as possible. It is efficient and secure though the keys will take longer to generate. Triple DES (Data Encryption Standard) replaced the original DES as the industry standard, which is now largely replaced by AES. AES is also considered impervious to most attacks, minus attacks that male use of straight brute force. Hash functions use mathematics to convert data into fixed size bit strings, or hash values. Each hash value has a specific key to access it in the hash table that is created. These hash values and keys are created in pairs, and the creation of these pairs depends on which algorithm and its bit-form. Random numbers are made use of in cryptography to encrypt and decrypt data. Symmetric-key uses the same key for both encryption and decryption of data, which an asymmetric encryption uses a public key for encrypting data and a private key for decrypting. Encryption has been in used for almost 4,000 years and we can trace use of encryption back to 600 BS where it was used to ensure messages were secure until they reached their intended recipient ("A brief history of encryption", 2021).

Modern cryptography can be traced back to IBM in the 1970’s, where they formed a cryptography group ("A brief history of encryption", 2021). Today elliptic-curve cryptography is in use in most cases. This helps to ensure security because the keys that are created are actually shorter, but more difficult to crack ("A brief history of encryption", 2021).

## 2. Certificate Generation

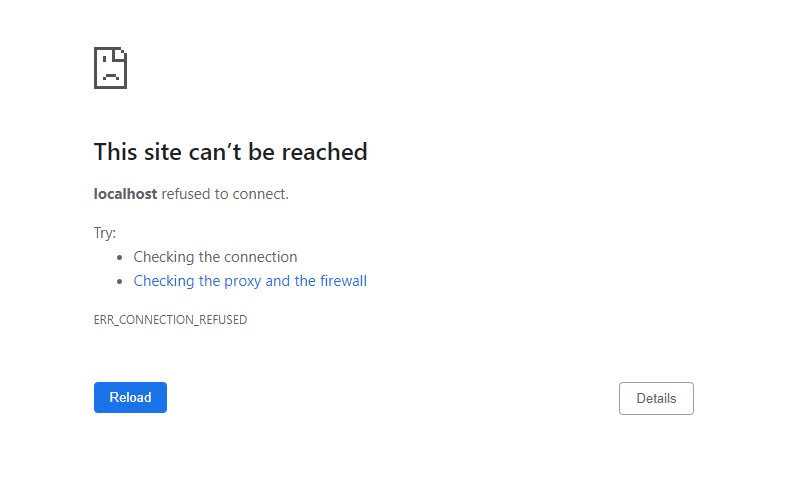
Generate appropriate self-signed certificates using the Java Keytool, which is used through the command line.





## 3. Deploy Cipher

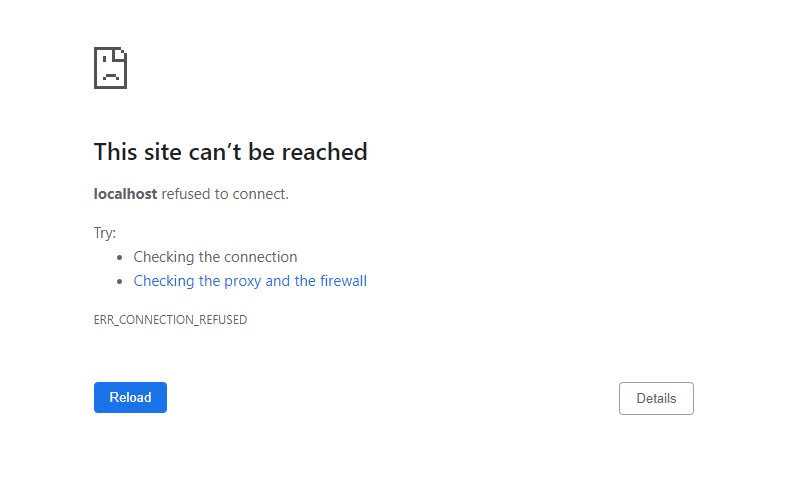
Refactor the code and use security libraries to deploy and implement the encryption algorithm cipher to the software application. Verify this additional functionality with a checksum.



I had issues getting this to connect correctly to my web-browser.

## 4. Secure Communications

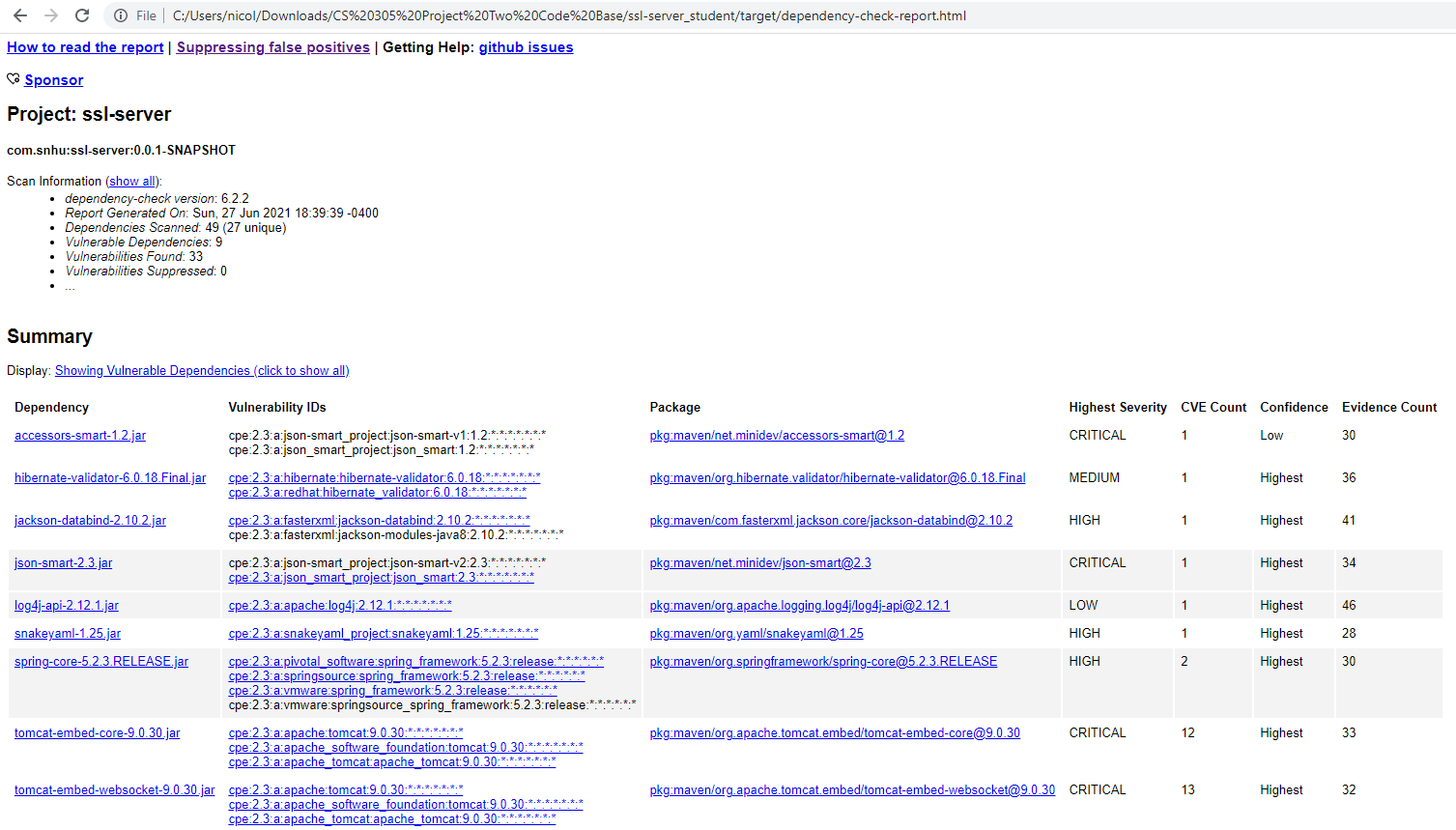
Refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code to verify secure communication by typing **https://localhost:8443/hash** in a new browser window to demonstrate that the secure communication works successfully.

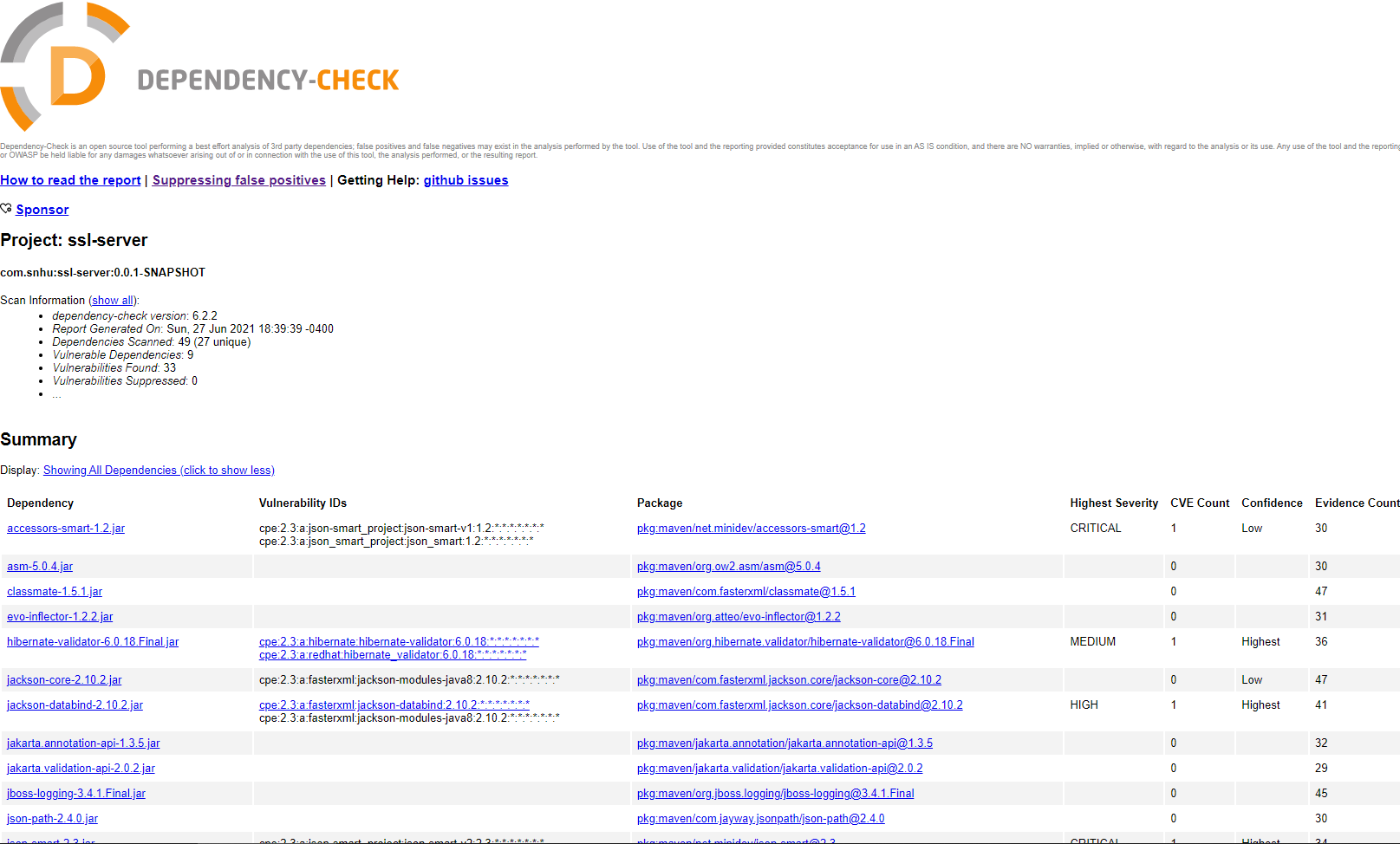


I had issues getting this to compile correctly.

## 5. Secondary Testing

Complete a secondary static testing of the refactored code using the dependency check tool to ensure code complies with software security enhancements. You only need to focus on the code you have added as part of the refactoring. Complete the dependency check and review the output to ensure you did not introduce additional security vulnerabilities.

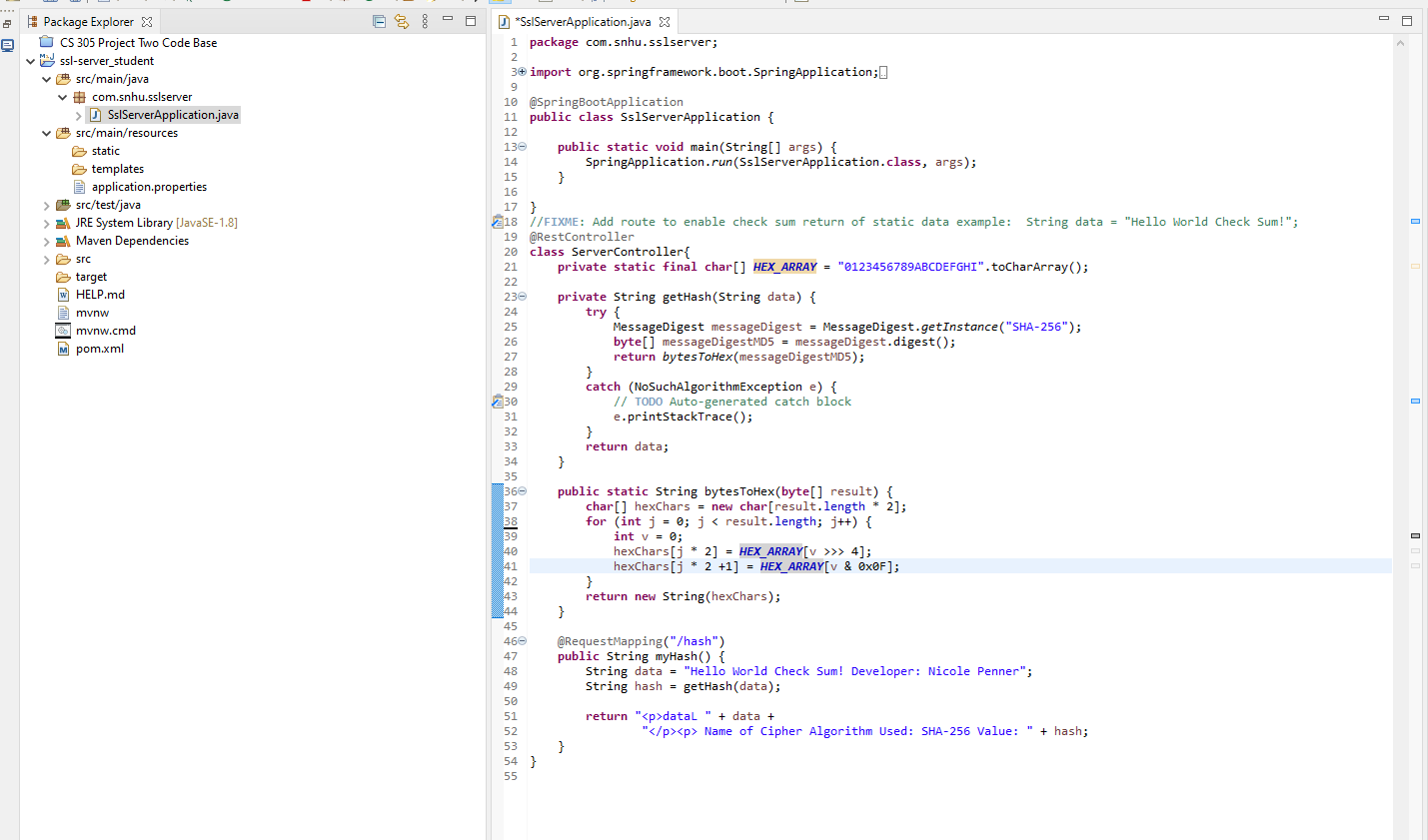




## 6. Functional Testing

Identify syntactical, logical, and security vulnerabilities for the software application by manually reviewing code.

*There are no errors in the code when manually reviewing the code.*



## 7. Summary

Discuss how the code has been refactored and how it complies with security testing protocols. Be sure to address the following:

While rewriting this code, I made use of a RestController, this serves as a secure way to use a RESTful API. This will address the concern of security as brought up in the Vulnerability Assessment Diagram in many ways. Another way to address security is that I have made use of the SHA-256 algorithm cipher, it has longer hash keys when encrypting and decrypting data. Since there is only a few files that the code uses to run, and it has very few dependencies (as checked through Maven), this also increases the security of the application. Maven needed to be updated to the most recent version which is 6.2.2, which was done in the pom.xml file.

Ways to continue to keep this application as secure as possible require a few different things. Dependency checks should be run often to ensure no new security issues have arose with different extensions the application uses. Versions of Maven and other extensions in the pom.xml file should be updated as often as new versions are released. Finally, if a more secure algorithm cipher exists in the future – this should be updated as well. It is important to keep up with the most recent security standards to ensure the most secure application possible.

A brief history of encryption. (2021). Retrieved 19 June 2021, from https://www.thalesgroup.com/en/markets/digital-identity-and-security/magazine/brief-history-encryption