****

# 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** | **2/26/2022** | **Nicholas Cleveland** |  |

## 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

Nicholas Cleveland

## 1. Algorithm Cipher

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

* Provide a brief, high-level overview of the encryption algorithm cipher.
* Discuss the hash functions and bit levels of the cipher.
* Explain the use of random numbers, symmetric vs non-symmetric keys, and so on.
* Describe the history and current state of encryption algorithms.

When looking over the security needs of the Artemis Financial software the need for an algorithm cipher is evident. Based on the needs, I would say that the software should user the SHA-256 and the AES-256 algorithm encryption. SHA-256 is something called a cryptographic hash which “generates an almost-unique 256-bit (32-byte) signature for a text.” (Veness, 2019) This encryption will use something called hash functions to help to secure the secure information. It is noted though that “A hash is not ‘encryption’ – it cannot be decrypted back to the original text (it is a ‘one-way’ cryptographic function and is a fixed size for any size of source text).” (Veness, 2019)

The other cipher that should be used for message digest is AES-256. Encryption ciphers use different types of way to encrypt data and two of those ways are symmetric and asymmetric. “AES is a symmetric key cipher. This means the same secret key is used for both encryption and decryption, and both the sender and receiver of the data need a copy of the key. “ (N-able, 2021) This shows how the was the cipher works in a symmetric way. “By contrast, asymmetric key systems use a different key for each of the two processes.” (N-able, 2021)

AES-256 is something called a block cipher and it divides the information into sections to be encrypted. “AES uses a 128-bit block size, in which data is divided into a four-by-four array containing 16 bytes.” (N-able, 2021)

## 

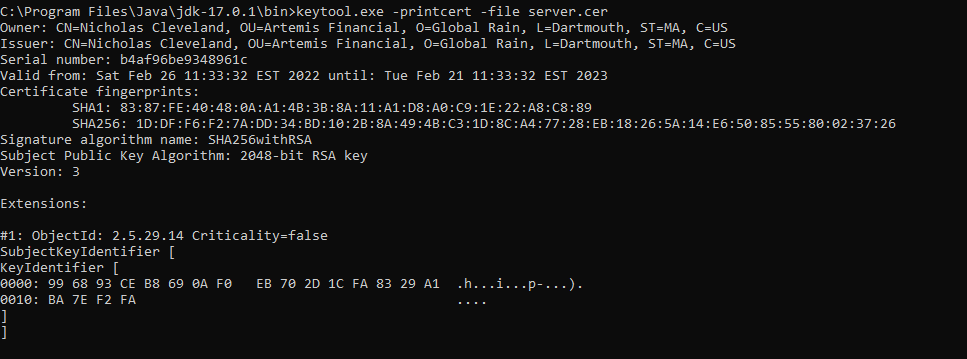
## 2. Certificate Generation

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

* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.

Text

Description automatically generated

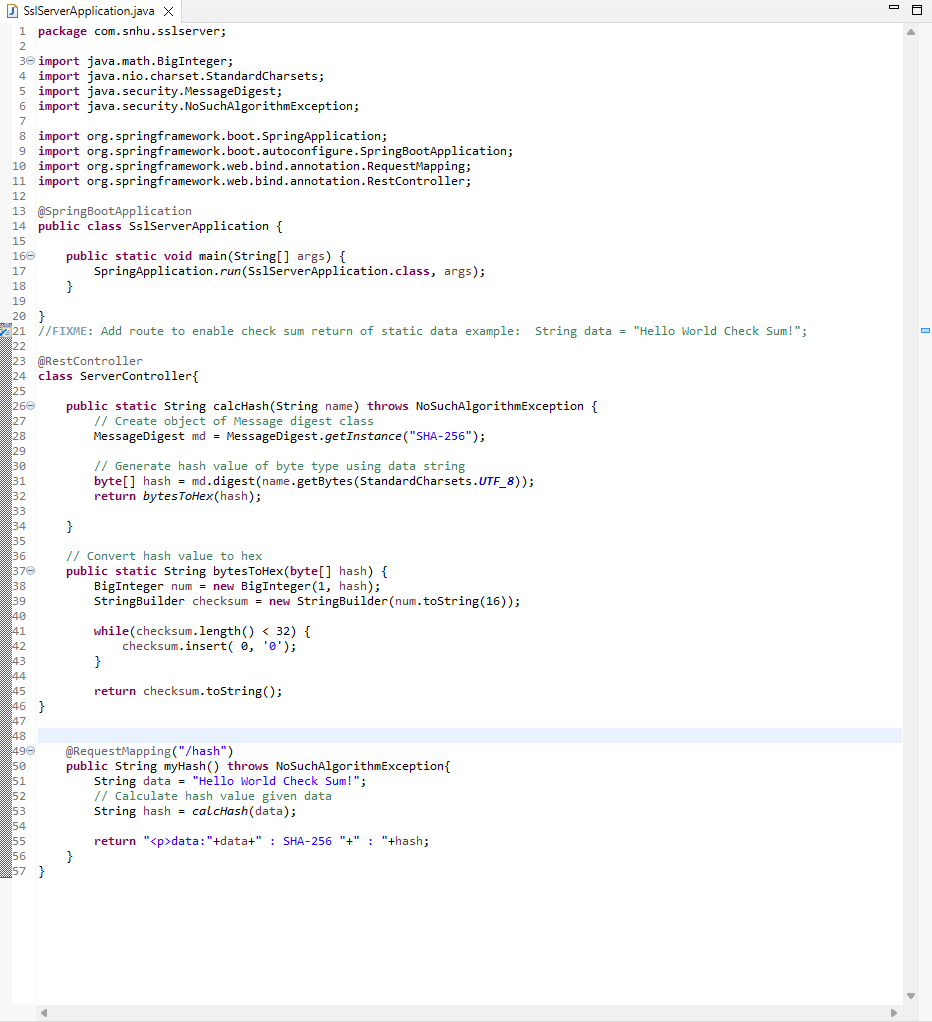


## 

## 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.

* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.



## 

## 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.

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

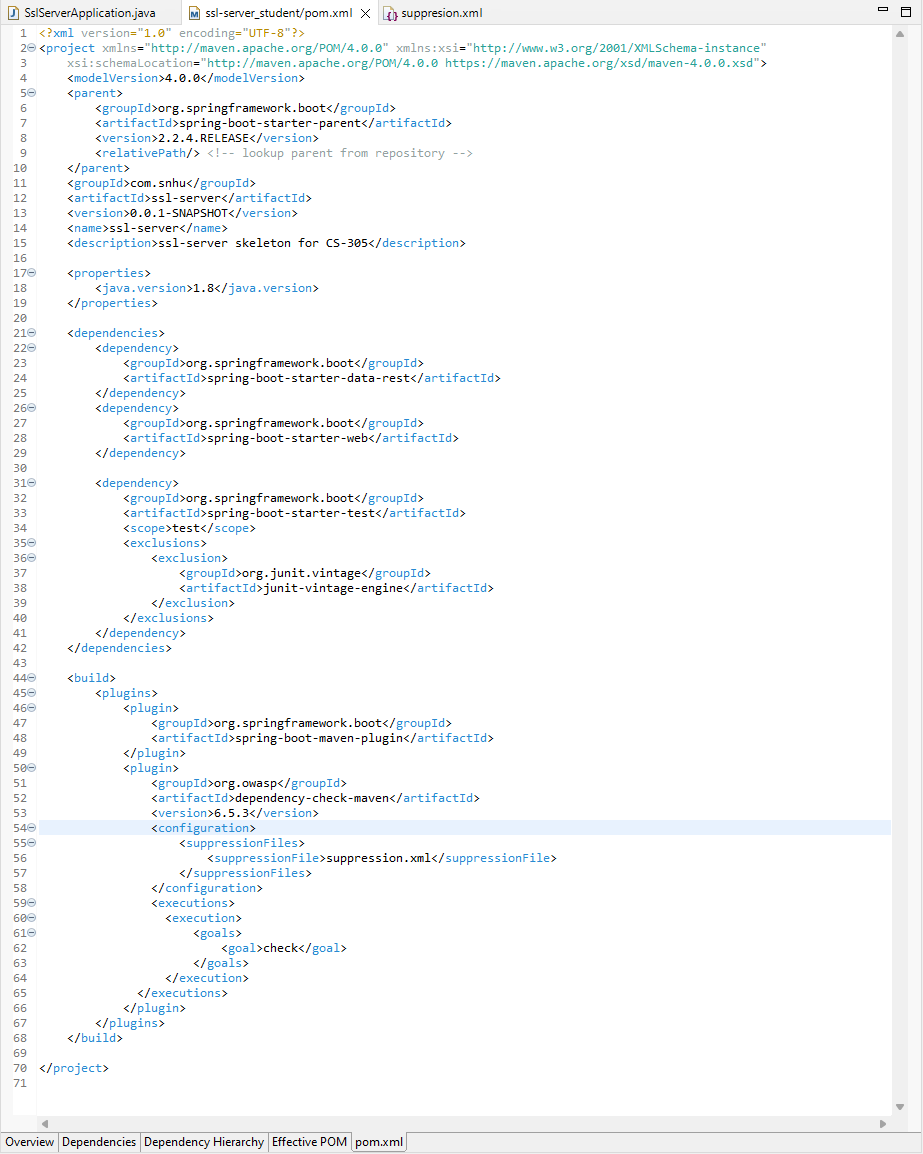
## 5. Secondary Testing

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.

* Include the following below:
  + A screenshot of the refactored code executed without errors
  + A screenshot of the dependency check report

Graphical user interface, text, application

Description automatically generated

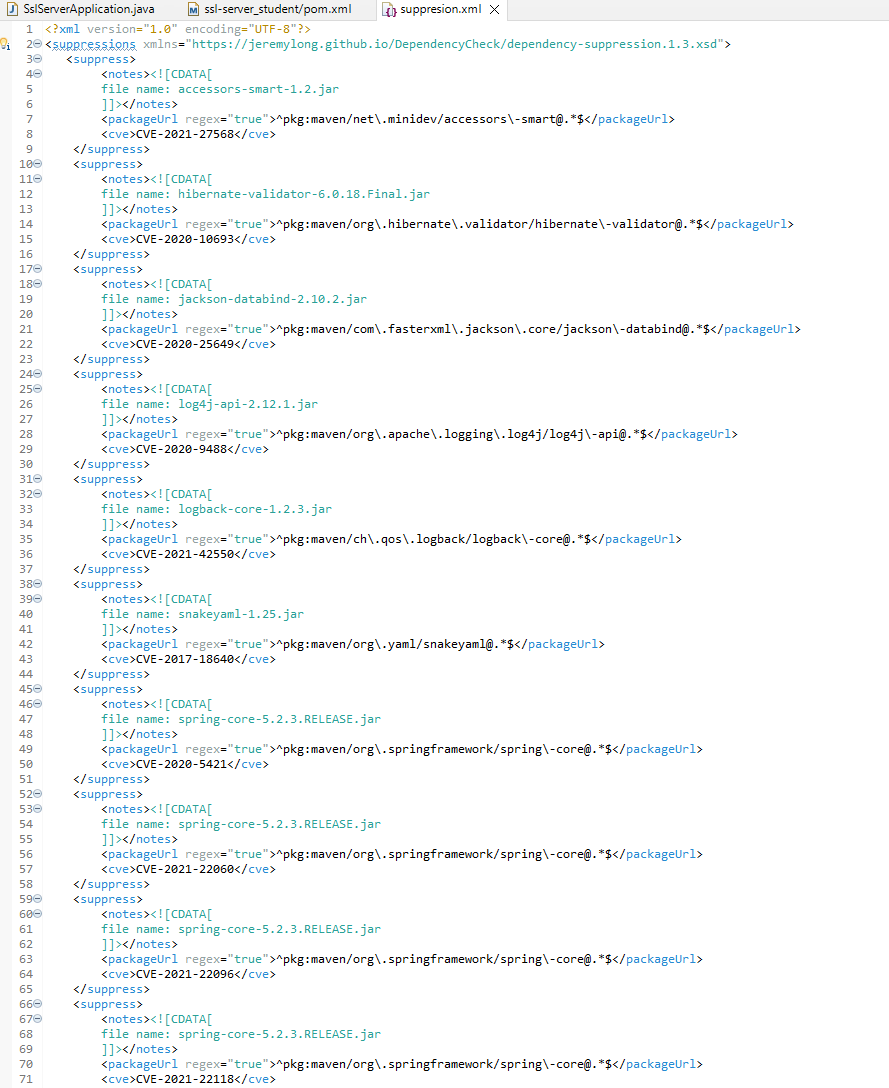


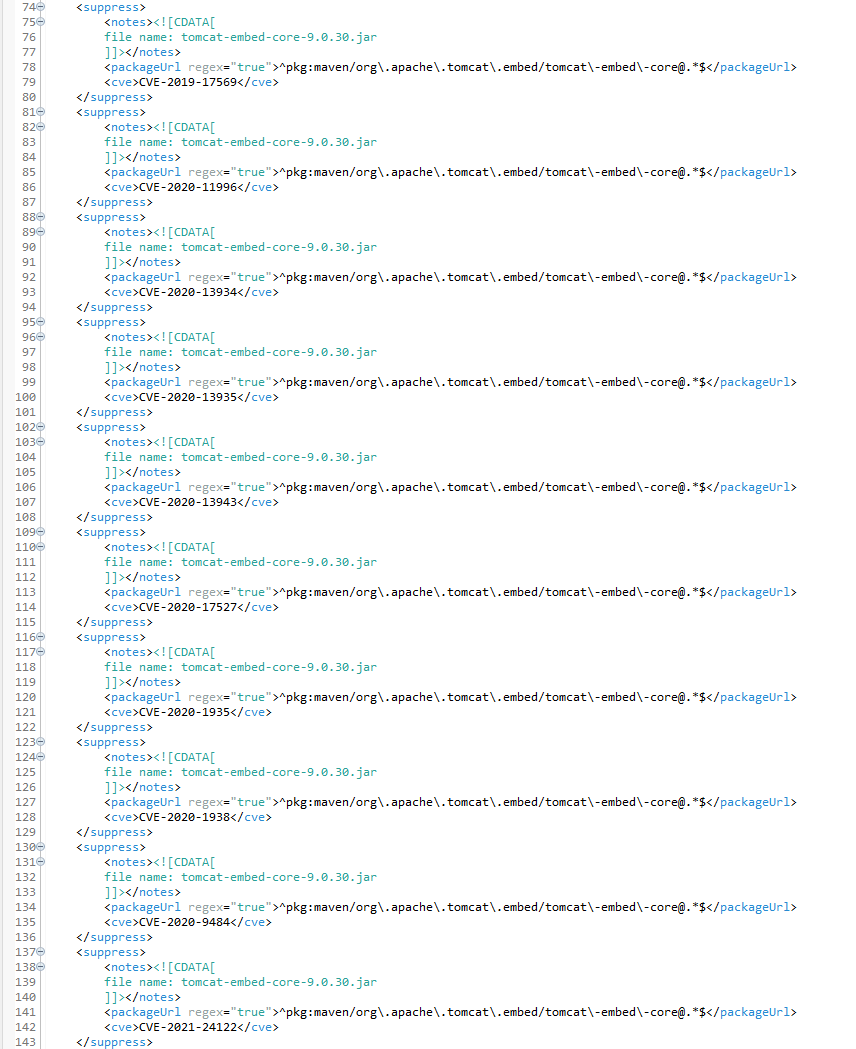
## 

## 6. Functional Testing

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

* Complete this functional testing and include a screenshot below of the refactored code executed without errors.



Table

Description automatically generated with medium confidenceGraphical user interface, text, application

Description automatically generated

## 

## 7. Summary

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

* Refer to the Vulnerability Assessment Process Flow Diagram and highlight the areas of security that you addressed by refactoring the code.
* Discuss your process for adding layers of security to the software application and the value that security adds to the company’s overall wellbeing.
* Point out best practices for maintaining the current security of the software application to your customer.

Many areas were addressed by refactoring the code. Secure Cryptography was addressed when we selected and algorithm cipher and implemented it into the code. This helped us to be able to encrypt data. Client and Server was addresses when we generated a certificate with the using the Java Keytool. This helped to make the server secure when it is being accessed. Code Error and Code quality was also something that was addressed when refactoring the code. We had errors come up when running the dependency report that helped us find false positives and be able to suppress them. This helped to make the code more secure and help with adding layers of security. \

Adding layers of security to the software helps to strengthen its defense again anyone who tries to gain unauthorized access. We first determined the algorithm cipher that would best fit the needs of the software. The next thing that was added a certificate that helped to give us access to the software with a private key that we created. We then deployed the algorithm cipher into our software to help to encrypt the data to help add security to the code. We verified this and made sure that the connections were secure between the software and the server using a checksum verification. Then we ran a dependency chert to see if any vulnerabilities in the code were found. After viewing this report, we were able to suppress false negatives and fix the errors that were found. Doing all of this helped to add security to the software and make it even more difficult for hackers to gain access to the information.

There are a lot of best practices that should be followed for maintaining the current security of the software. One of the main things is to make sure that your software is up to date with any fixes to possible vulnerabilities that may arise. “Many attackers exploit known vulnerabilities associated with old or out-of-date software.” (Software, 2021) This will help to make sure your software stays protected. You should also train users with all the information that we accomplished to make sure that they can make any necessary changes to the existing code. “Having a well-organized and well-maintained security training curriculum for your employees will go a long way in protecting your data and assets.” (Software, 2021) One other thing you should do is to keep a record of all the changes made and a description of the policies in place for security of the software. “Security policies allow your employees, including network administrators, security staff, and so on, to understand what activities you’re performing and why.” (Software, 2021)

References

N-able *Advanced encryption standard: Understanding AES 256:*. N. (2021, April 9). Retrieved February 26, 2022, from https://www.n-able.com/blog/aes-256-encryption-algorithm

Software Integrity Blog *Top 10 software security best practices: Synopsys*.. (2021, July 1). Retrieved February 26, 2022, from https://www.synopsys.com/blogs/software-security/top-10-software-security-best-practices/

Veness, Chris www.movable-type.co.uk. (n.d.). *Movable type scripts*. SHA-256 Cryptographic Hash Algorithm implemented in JavaScript | Movable Type Scripts. Retrieved February 26, 2022, from https://www.movable-type.co.uk/scripts/sha256.html