****

# CS 305 Project Two

**Practices for Secure Software Report**

**Table of Contents**

[Document Revision History 3](#_1fob9te)

[Client 3](#_3znysh7)

[Instructions 3](#_2et92p0)

[Developer 4](#_tyjcwt)

[1. Algorithm Cipher 4](#_3dy6vkm)

[2. Certificate Generation 4](#_1t3h5sf)

[3. Deploy Cipher 4](#_4d34og8)

[4. Secure Communications 4](#_2s8eyo1)

[5. Secondary Testing 4](#_17dp8vu)

[6. Functional Testing 5](#_3rdcrjn)

[7. Summary 5](#_26in1rg)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **[Date]** | **Michelle Hitchcock** |  |

## 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

Michelle Hitchcock

## 1. Algorithm Cipher

Encryption algorithms are designed to encrypt, or hide information. This is done so that those who shouldn’t have access to the information cannot.

Hash functions are used to improve security by allowing for verification and authentication. Bit levels are another factor that impacts the cipher. The Oracle website notes that “in general, longer keys provide stronger encryption. Key length is measured in bits.” (Oracle Corporation, 2010). one example of a hashing function is SHA-256.

Because this is a financial institution, this cipher will be used to encrypt user’s personal information, financial information, passwords, and transaction information. This cipher will also need to sign information such as transactions and logins in order to validate them.

I would suggest the use of the AES cipher, with the highest bit rate supported.

I think this is a good choice because this cipher is secure when implemented properly, it is supported, and according to NIST “Today, AES protects everything from classified data and bank transactions to online shopping and social media apps.” (NIST, 2018) suggesting that this type of systematic encryption algorithm is well suited to this application.

There are many components to hash functions such as symmetrical keys, random numbers, and the concept of forward security.

Symmetric keys are private keys, where the key itself is needed to decrypt the data and must be kept secret. Non-symmetric systems are two key systems where there is a public and private key. Another important factor noted is how symmetrical keys require fewer bits to be secure, whereas non-symmetrical keys require more bits to be secure.

Random numbers are used through cryptography, to generate secure keys, passcodes, etc., and ensure that the system is more secure from human error. (Smid, 2012)

“Forward secrecy protects your past communications even if the server’s private keys are compromised” (Manico & Detlefsen, 2014). this is another important thing to consider. It relates both to random numbers and to keys, as both are needed to implement forward secrecy. In the case of a financial institute, this type of system is invaluable.

Cryptography has been used throughout history since before computers, but the history of cryptography and encryption really started in WWII, as a tool used for espionage. “In the early 1930s, the German military adopted a new encryption protocol based on an existing commercial device called Enigma” (Roberts, 2016). The enigma machine was used both for encryption and description.

Going forward encryption has been used by the military ARPANET (Magazine, 2013), and finally in every properly secured internet device.

During each of the previous steps, enigma, ARPANET, the encryption algorithm was fervently being attempted to be broken. That has not changed today. Tufts University Scientist published research stating that “Quantum computing poses serious risks to widely-used encryption methods” (Kirsch & Chow, 2015) So although this is not a risk present today, it is perhaps one to consider for the future.

Citations:

American Bankers Association. (2016). *Ransomware Attacks*. Https://Www.Aba.Com/. <https://www.aba.com/advocacy/community-programs/consumer-resources/protect-your-money/ransomware-tips>

FDIC. (2016). *FDIC Law, Regulations, Related Acts - Rules and Regulations*. Fdic.Gov. <https://www.fdic.gov/regulations/laws/rules/2000-8660.html>

Jansen, J., & Leukfeldt, R. (2016). *Phishing and Malware Attacks on Online Banking Customers in the Netherlands: A Qualitative Analysis of Factors Leading to Victimization*. Https://Cybercrimejournal.Com/. https://cybercrimejournal.com/Jansen%26Leukfeldtvol10issue1IJCC2016.pdf

McKeever, G. (2021, March 22). *Why Banks Are Still A Top Target For DDoS Attacks*. Security Boulevard. https://securityboulevard.com/2021/03/why-banks-are-still-a-top-target-for-ddos-attacks/

IMB Corperation. (2015). *Hashing functions*. Imb.Com. https://www.ibm.com/docs/en/psfa/7.2.1?topic=toolkit-hashing-functions

Kirsch, Z., & Chow, M. (2015). *Quantum Computing: The Risk to Existing Encryption Methods*. Cs.Tufts.Edu. <http://www.cs.tufts.edu/comp/116/archive/fall2015/zkirsch.pdf>

Magazine, S. (2013, September 6). *How the NSA Stopped Trying to Prevent the Spread of Encryption And Decided to Just Break It Instead*. Smithsonian Magazine. https://www.smithsonianmag.com/smart-news/how-the-nsa-stopped-trying-to-prevent-the-spread-of-encryption-and-decided-to-just-break-it-instead-4569969/

Oracle. (2020). *Secure Coding Guidelines for Java SE*. Oracle.Com. https://www.oracle.com/java/technologies/javase/seccodeguide.html

Oracle Corporation. (2010). *Key Length and Encryption Strength (Sun Directory Server Enterprise Edition 7.0 Reference)*. Doc.Oracle.Com. https://docs.oracle.com/cd/E19424-01/820-4811/aakfw/index.html

Roberts, E. (2016). *Cryptography*. Stanford.Edu. <https://web.stanford.edu/class/cs106j/assignments/Assignment-05/Cryptography.pdf>

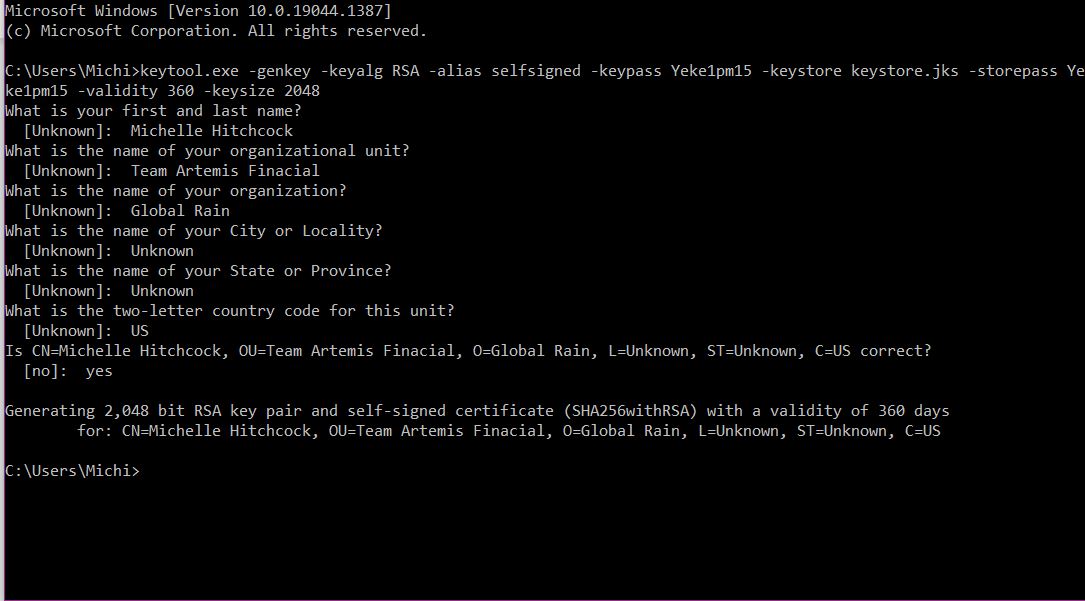
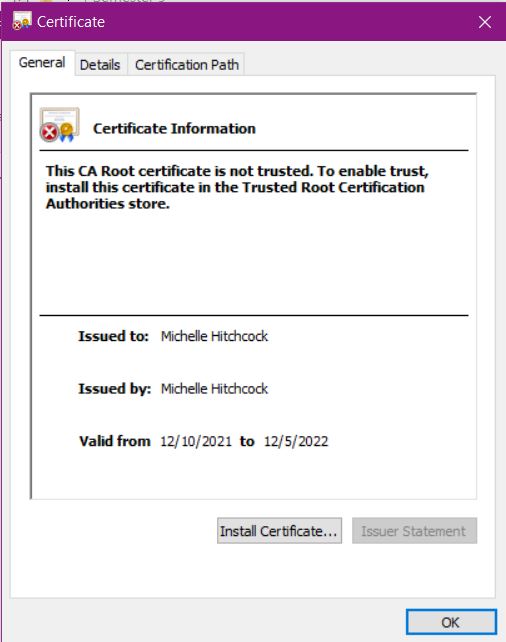
NIST. (2018, December 11). *NIST’s Encryption Standard Has Minimum $250 Billion Economic Benefit, According to New Study*. <https://www.nist.gov/news-events/news/2018/09/nists-encryption-standard-has-minimum-250-billion-economic-benefit>

Smid, M. (2012). *Why Random Numbers for Cryptography?* NIST. <https://csrc.nist.gov/csrc/media/events/random-number-generation-workshop-2004/documents/developmenthistory.pdf>

Manico, J., & Detlefsen, A. (2014). *Iron-Clad Java* [E-book]. Oracle Press.

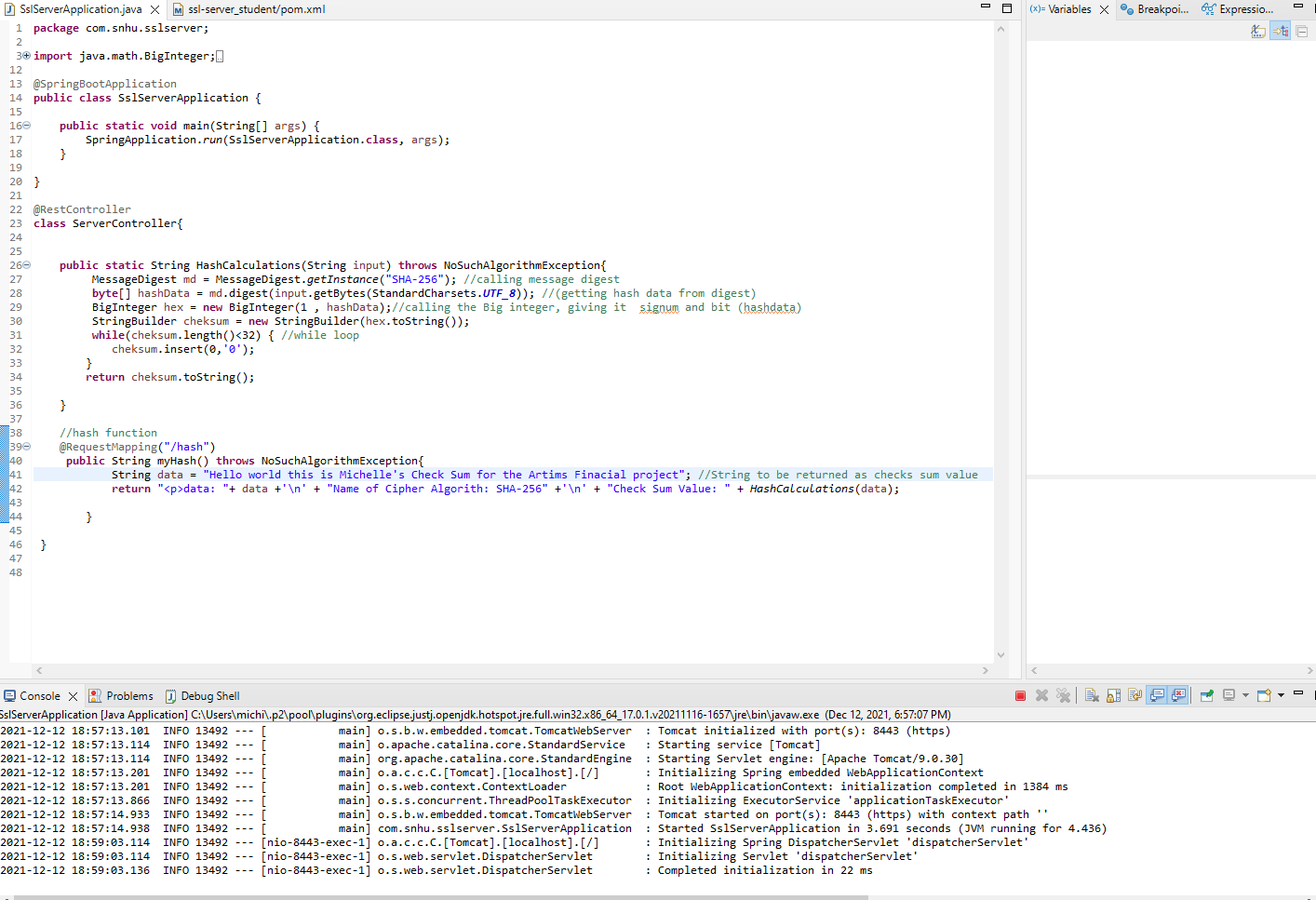
## 2. Certificate Generation

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.



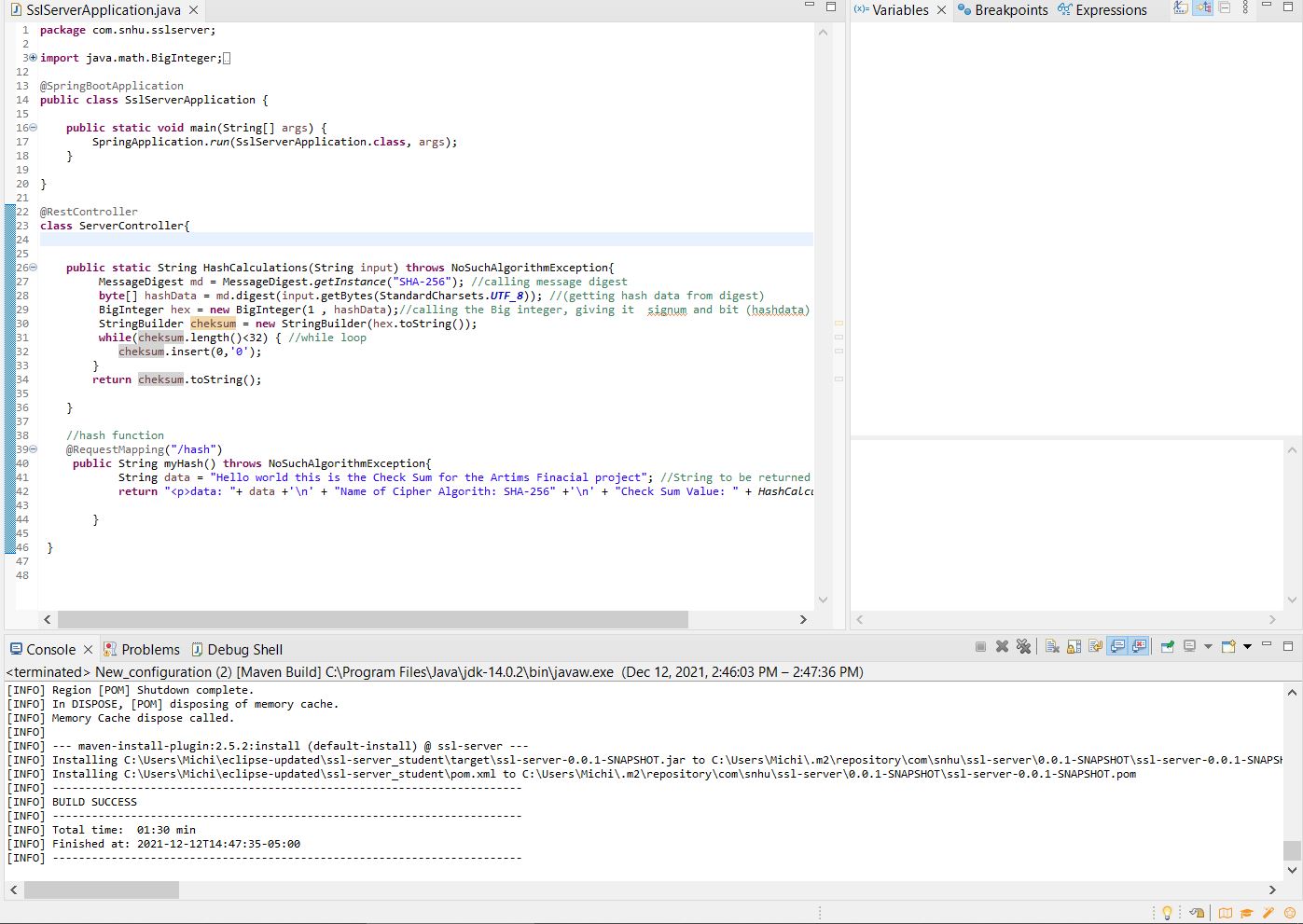
## 4. Secure Communications

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.



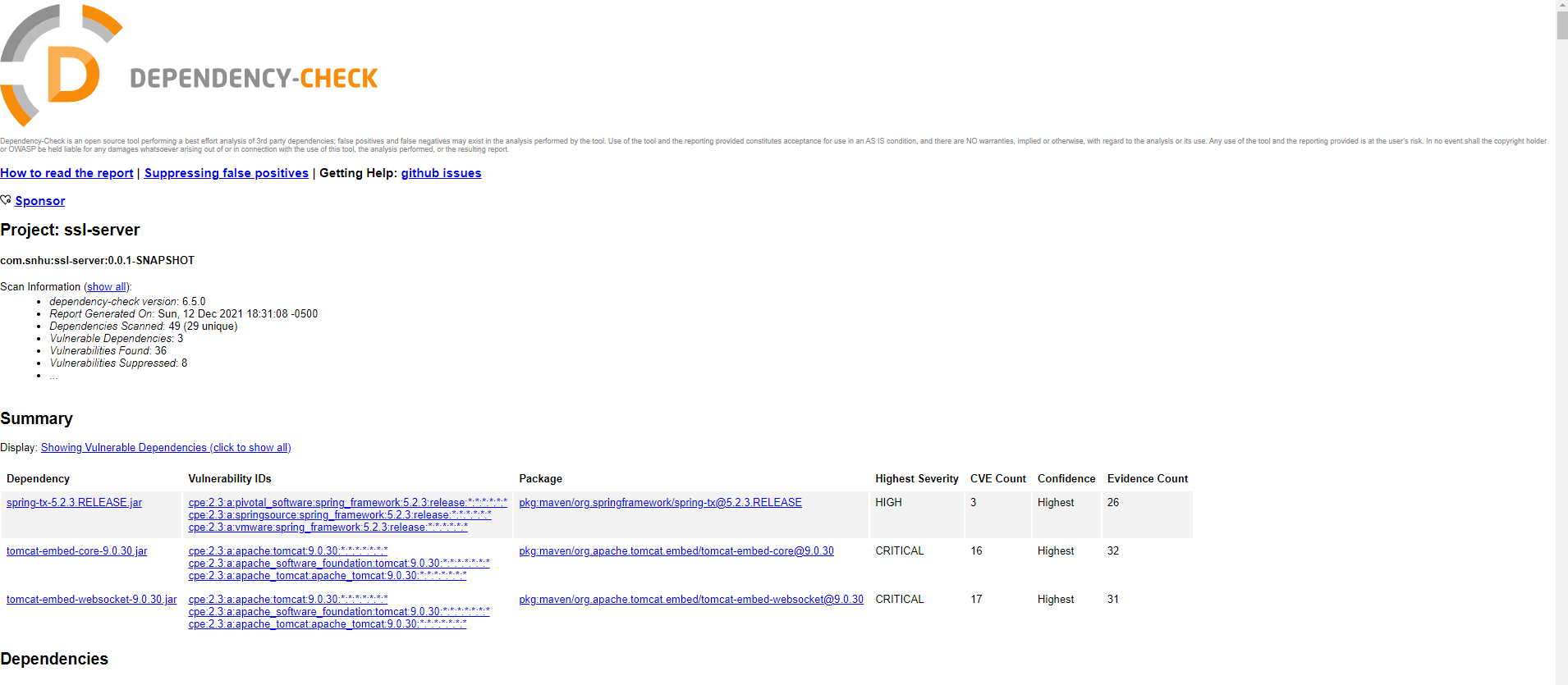
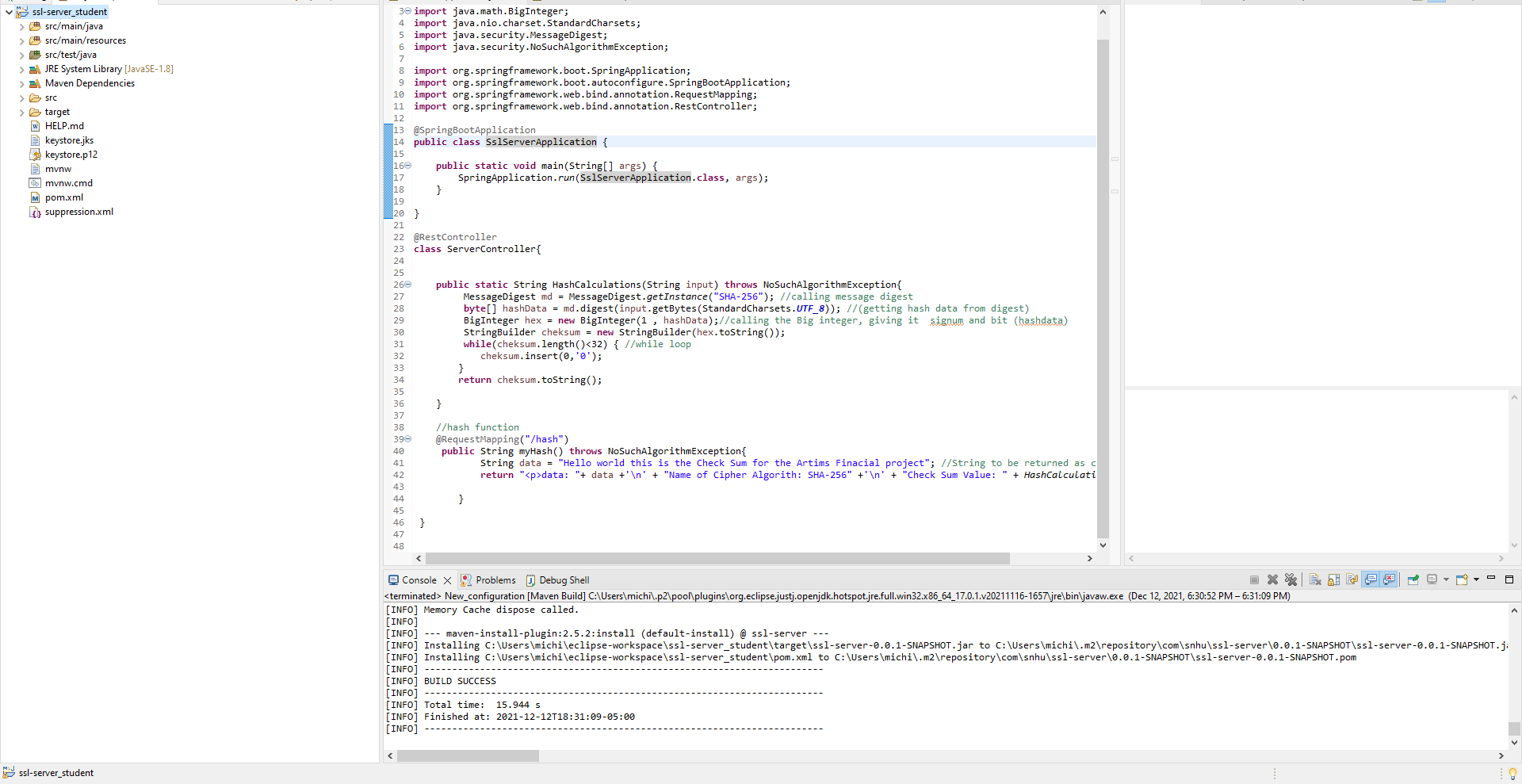
## 5. Secondary Testing

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



## 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.

In order to make this code more secure I focused on the cryptography and secure API interactions. Although I was ultimately unable to fully update the APIs, which was my intent, I did implement an update for some of the dependencies and suppression and for false positive vulnerability alerts.

The cryptography I implemented used SHA-256 in order to provide better security.

In order to maintain the code I suggest proper API updates, comments, and continuing to follow proper secure coding procedures.