****

# 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 4](#_heading=h.4d34og8)

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

[5. Secondary Testing 4](#_heading=h.17dp8vu)

[6. Functional Testing 5](#_heading=h.3rdcrjn)

[7. Summary 5](#_heading=h.26in1rg)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **10/17/21** | **Patrick Valencia** |  |

## 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

Patrick Valencia

## 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.

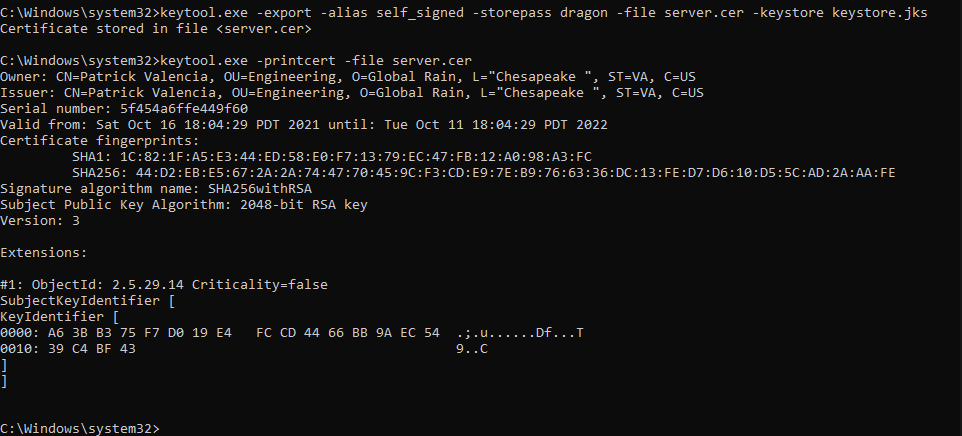
The Algorithm Cipher that we will use will be the AES-256 (Advanced Encryption Standard). I believe that this is the best choice of cipher due to the security it provides especially with such sensitive materials such as financial data. The most important thing about this is that AES has been the go to encryption for several governments, financial institutions, and security-conscious businesses around the globe. (Franklin, 2021) It has been the go to when it comes to algorithm ciphers because of how its block cipher is a symmetric encryption algorithm in which it uses one key to encrypt and decrypt information. This type of bit encryption is much better due to the fact that if someone is trying to decrypt and makes a mistake in the decryption process, the entire key would be reset and they would have to start from the very beginning (“Secure your data”, n.d)

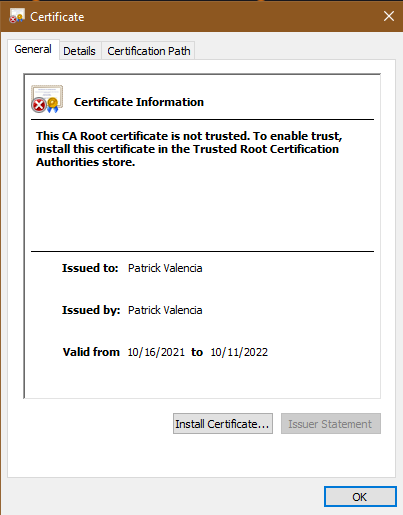
When it comes to hash functions, SHA or Secure Hashing Algorithms are common hash functions. The most important thing with this is that SHA’s can be used alongside AES to ensure the datas security. The main process of a hash function is that it takes plaintext data as input and then creates a cipher text that is highly unlikely to be deciphered. Since this type of cipher is bit level security this means that a person trying to decipher would need to correctly do 2^n operations. And since the AES we’ll be utilizing is AES-256, they would have to make 2^256 correct operations, or 115 quattuorvigintillion, thus making AES-256 the most secure.

## 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.

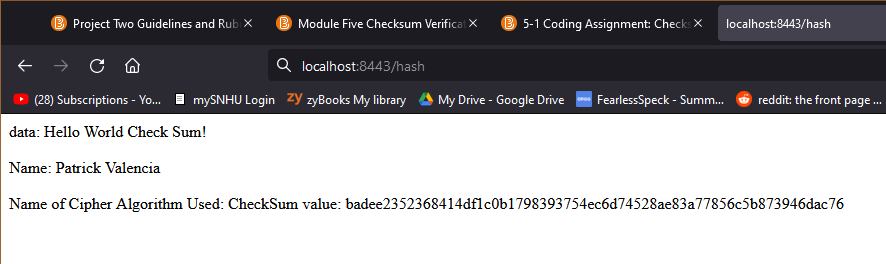




## 3. Deploy Cipher

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.

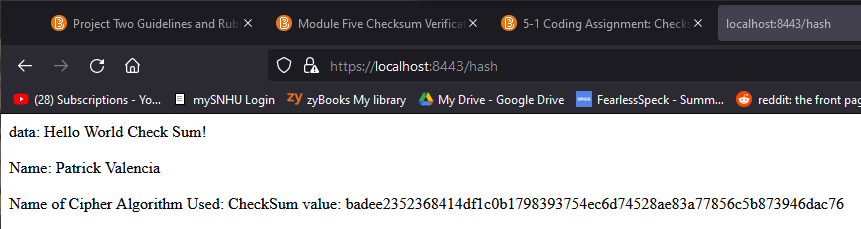
* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.



## 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.

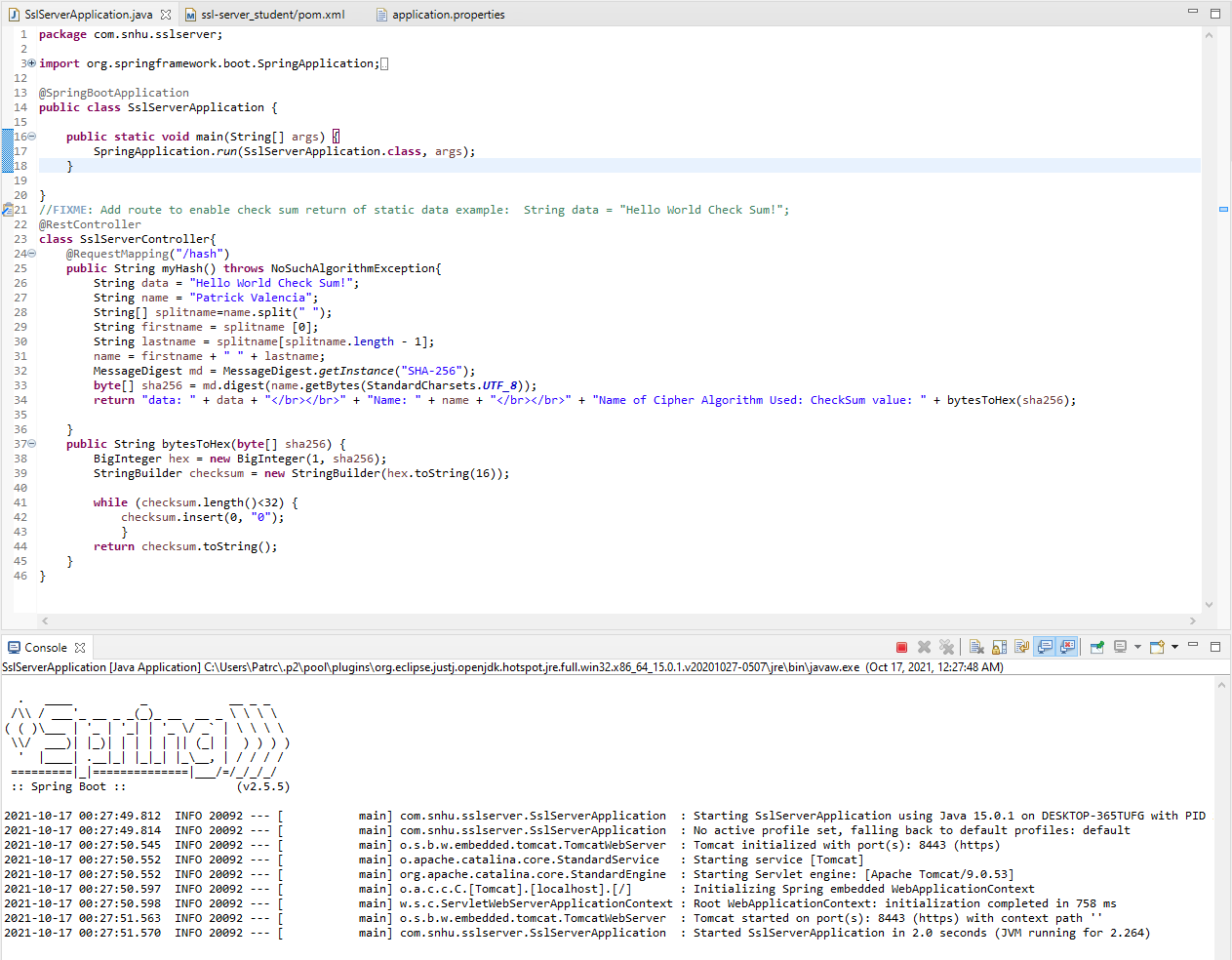
* Insert a screenshot below of the web browser that shows a secure webpage.

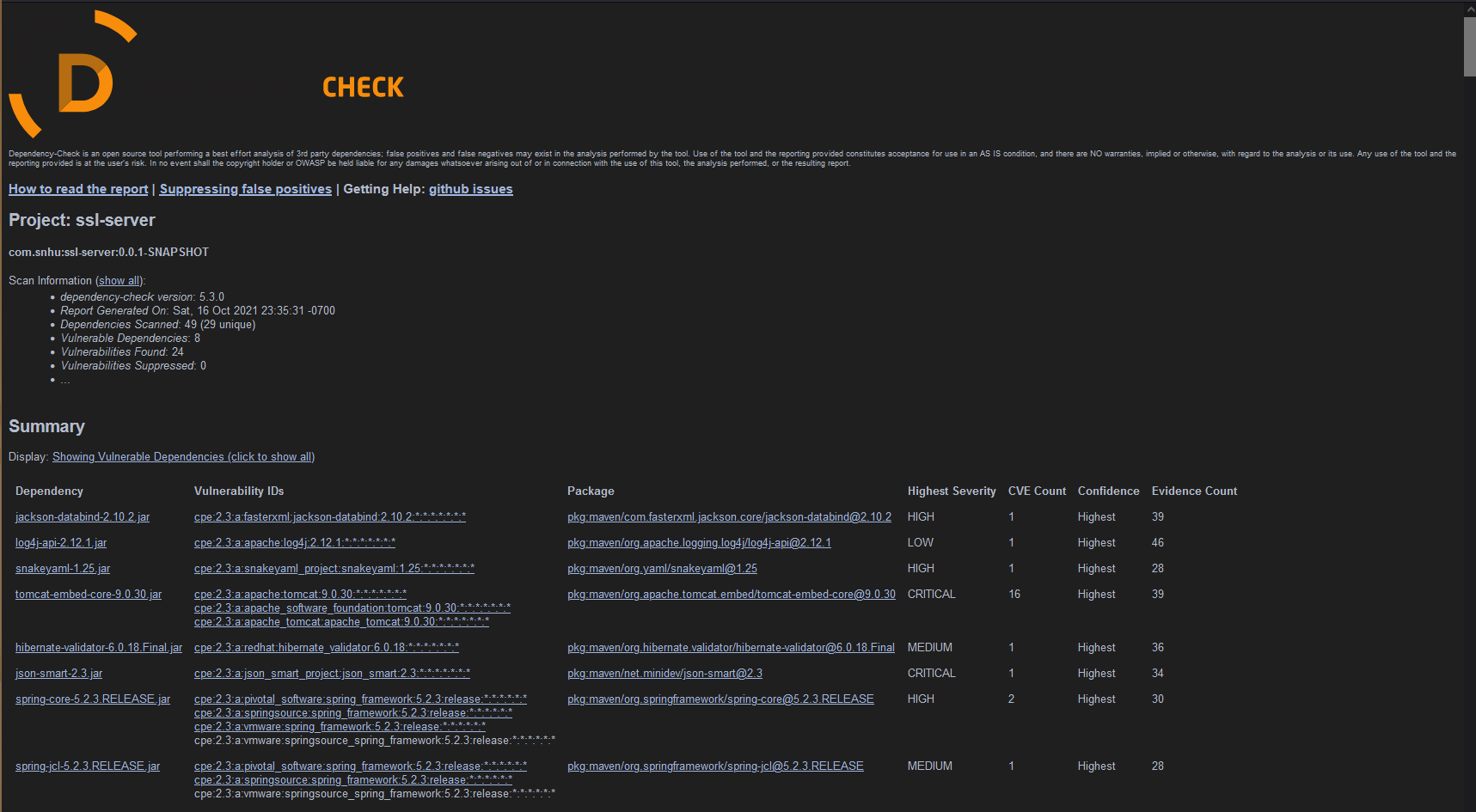


## 5. Secondary Testing

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

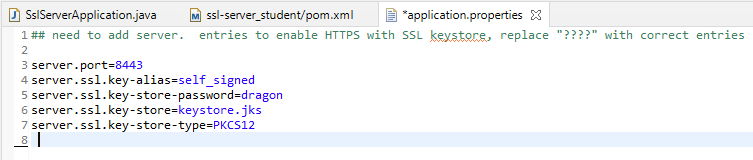
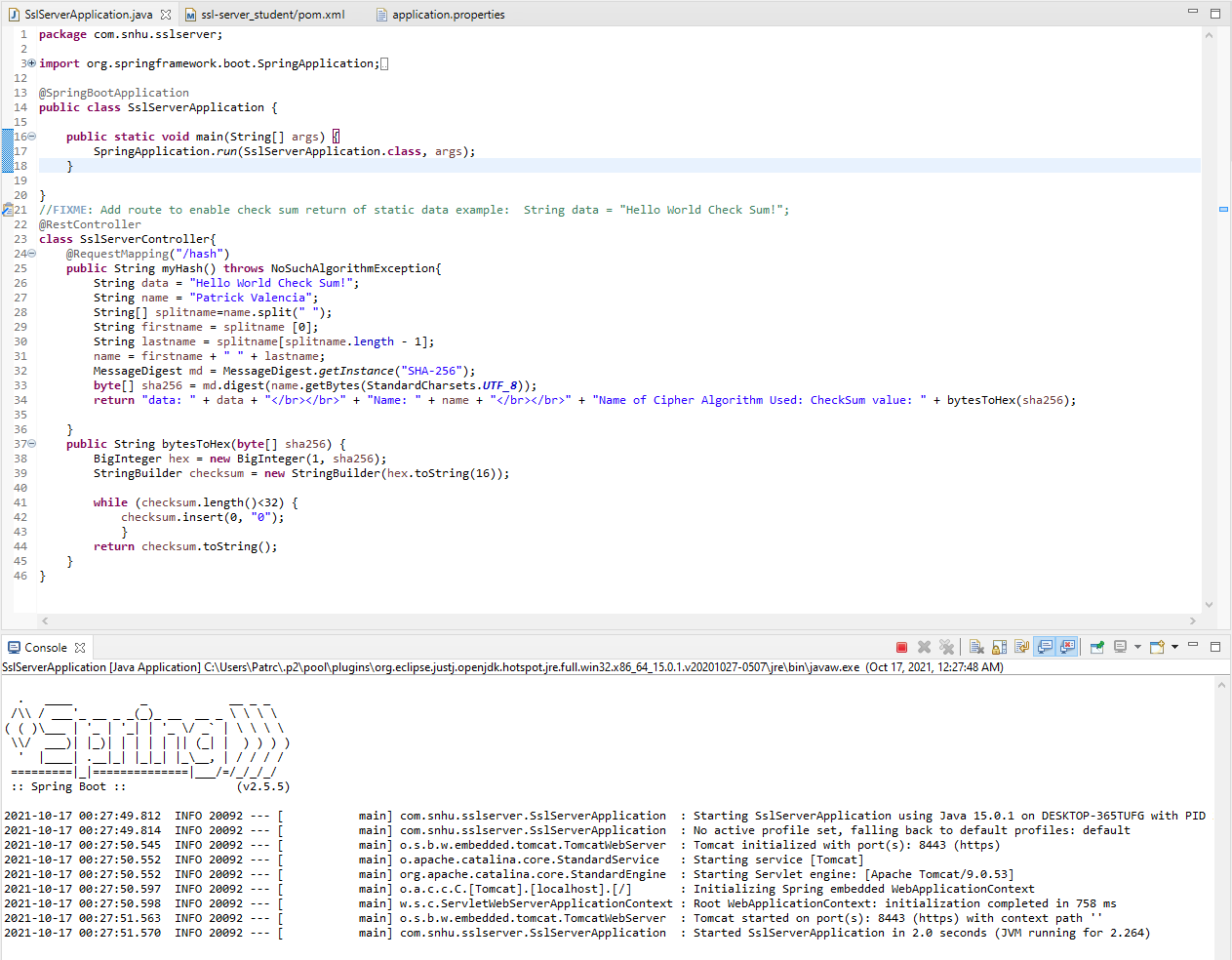




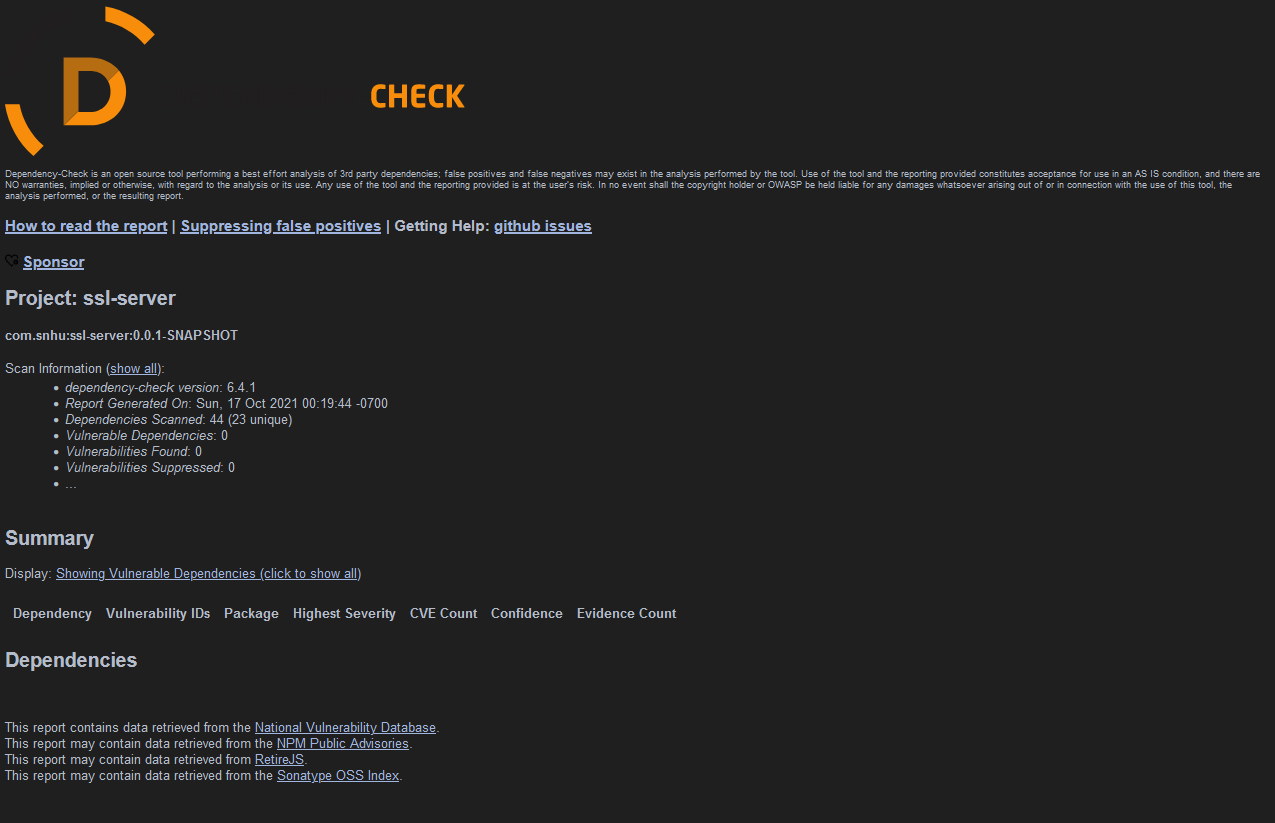
## 6. Functional Testing

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.







## 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.

The areas of security that I addressed by refactoring the code involve API's, Cryptography, Client/Server, and Code Quality. With API’s, we secured the interactions by creating a self signed certificate and generated a key for the application when using the AES-256 encryption. This encryption is used to verify the proper users are the ones viewing such sensitive financial materials that would otherwise be used for malicious intent. With the self signed certificate, it helps build both security and trust with the company and their users with the addition of a secured means of communication without the worry of any form of leak. The security certificate as well as the keystore.jks are located within the ssl-server\_student folder.

As for Cryptography, this was addressed within the refactored code by having a checksum for both the string data of “Hello World Check Sum!” and my name “Patrick Valencia”, to show that the cipher algorithm worked. With this we can see that it properly ciphers data correctly and encrypts it so that no one without the proper key can decipher it. This is important for security for the company because with proper cryptography, if someone does get the data and tries to use it for malicious intent, they would still need to decipher it to be of use to them, and with AES-256 along with SHA-256, the method of brute force is near impossible.

With Client/Server, the main part that affects it with the refactored code is with the application properties. The application cannot properly launch with the incorrect properties as well as the incorrect keystore. Additionally when launched correctly, it would take them through a secured HTTPS secured network. This benefits both client and server due to not having to fear any type of data leaks as well as man in the middle attacks due to an unsecured connection.

Finally, when it comes to code quality, this was one of the most important things that had to be done correctly. With the refactored code, there were no errors when running it as well as with the dependency report, there were no found vulnerabilities. Thanks to the updated dependencies in the pom.xml, the dependencies located were not vulnerable as well as didn’t put the integrity of security at risk. The best practices to keep this code well and running would be to keep an up to date dependency list as well as having a proper structured code to keep everything organized.

References

Franklin, R. (2021, May 13). *Aes vs. rsa encryption: What are the differences?* Precisely. Retrieved October 15, 2021, from <https://www.precisely.com/blog/data-security/aes-vs-rsa-encryption-differences>.

*Secure your data with aes-256 encryption*. How does AES-256 encryption work to protect your data. (n.d.). Retrieved October 15, 2021, from <https://www.atpinc.com/blog/what-is-aes-256-encryption>.