****

# 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** | **6/15/2022** | **Patrick Chu** |  |

## 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 Chu

## 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.

“Based on the information provided by Artemis Financial, and the needs of their system, I would recommend that they utilize the Advanced Encryption Standard or AES as their algorithm cipher. As the name would imply, this particular cipher is commonly utilized in both the government and private sector to protect information such as financial transactions or classified data. Without the designated key, a brute force attack method of decrypting the data would be practically impossible, as it would take far too long, billions, trillions, or more years.

A hash function can loosely be defined as a function that takes data and converts it into another form. It takes data and translates it into a form that is unreadable or unrecognizable from it’s original form, and it many cases, it cannot be converted back to its original form. A hash function is generally described in levels of bit-encryption, which represent the number of characters in the key. AES has several levels of encryption, offering 128-bit, 192-bit, and 256-bit encryption levels which can be used to protect various data within the organization for Artemis Financial.

Encryption keys are random in order ensure security, as easy to remember keys are much easier to crack than a randomized key. Symmetric keys imply that a piece of data is encrypted with the same key that would be used to decrypt the data, while asymmetric keys would use two different keys, one for encryption and another for decryption. This generally means that asymmetric keys and encryption are more secure, as two keys would need to be compromised for the data to be entirely compromised instead of one. However, while AES utilizes symmetric encryption, it does use long enough key lengths to ensure that it is more secure than other cipher types.

Encryption as it is currently known can be traced back to the 1970’s, and the first standard to be adopted was the Data Encryption Standard or DES. However, when that was cracked and no longer considered secure, the AES was developed and adopted, and is still in widespread use. “(Chu, 2022)

## 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.

Graphical user interface, text, application, email

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.

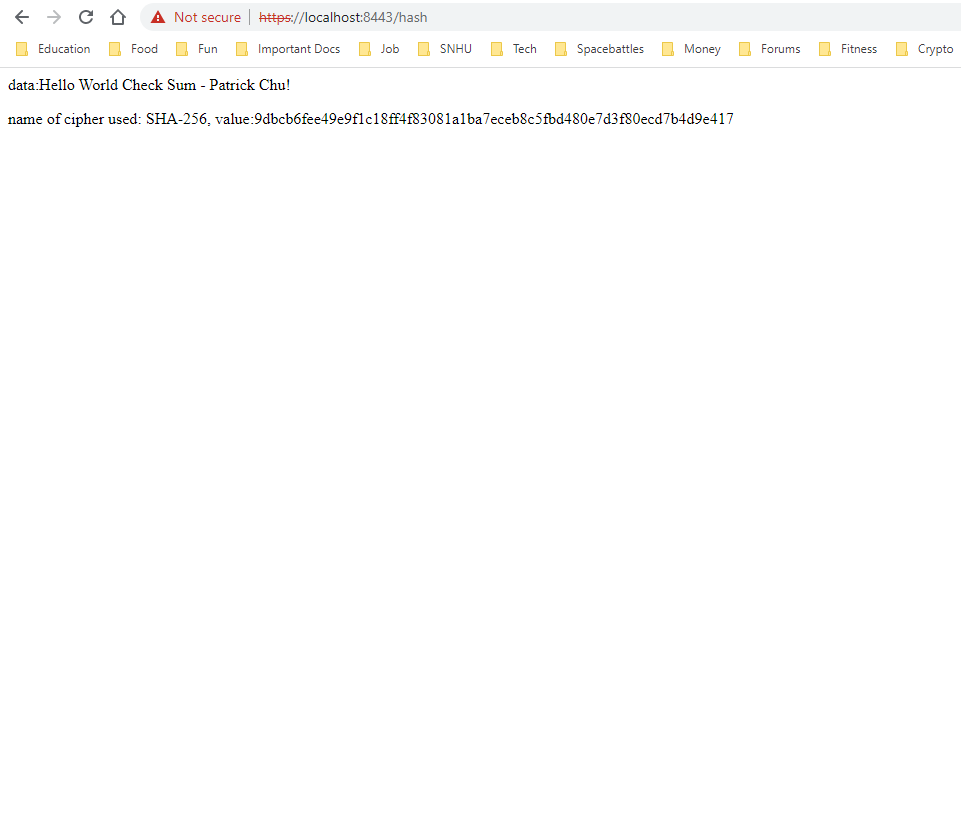
Graphical user interface, text, application, email

Description automatically generated

## 4. Secure Communications

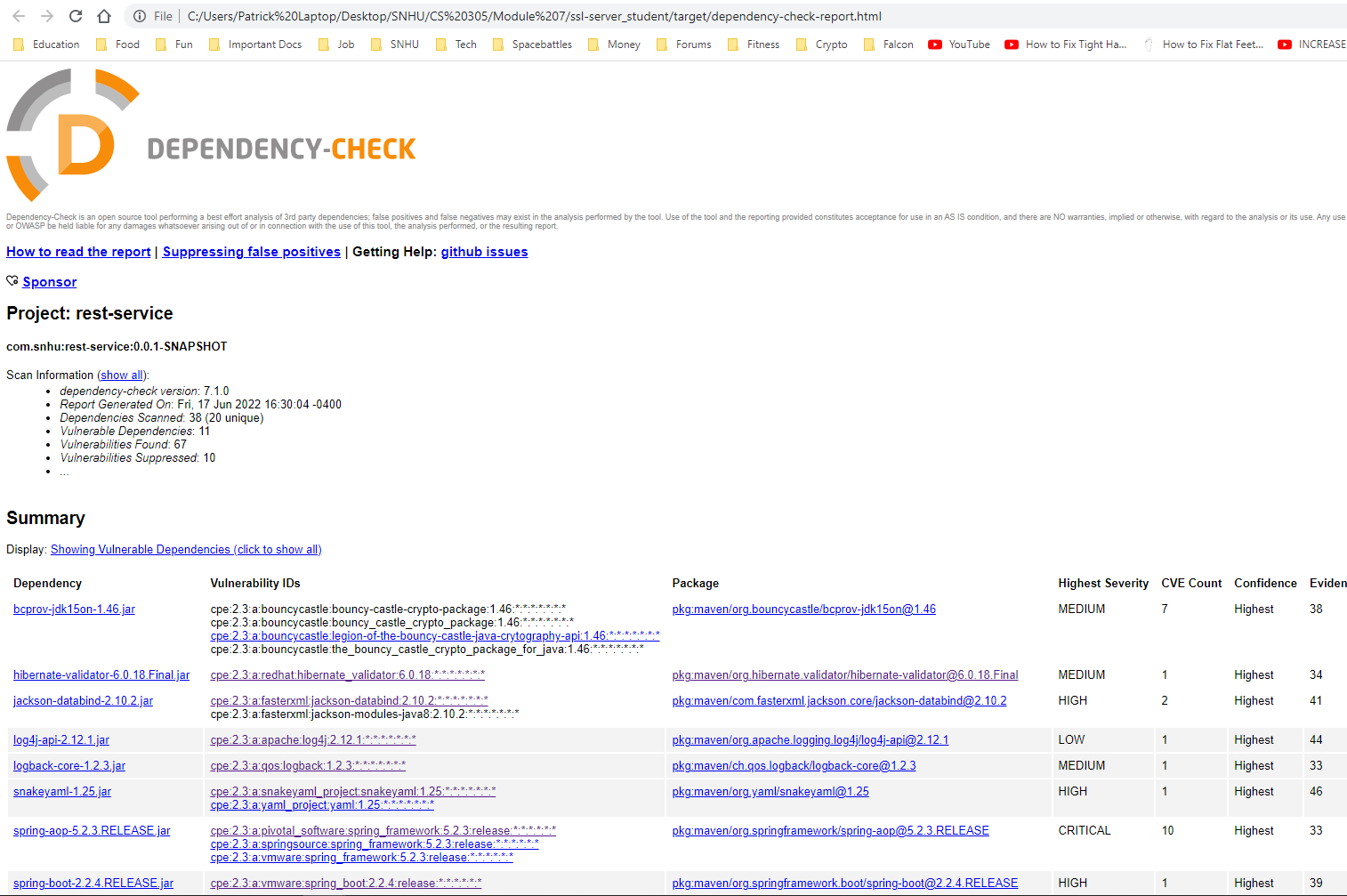
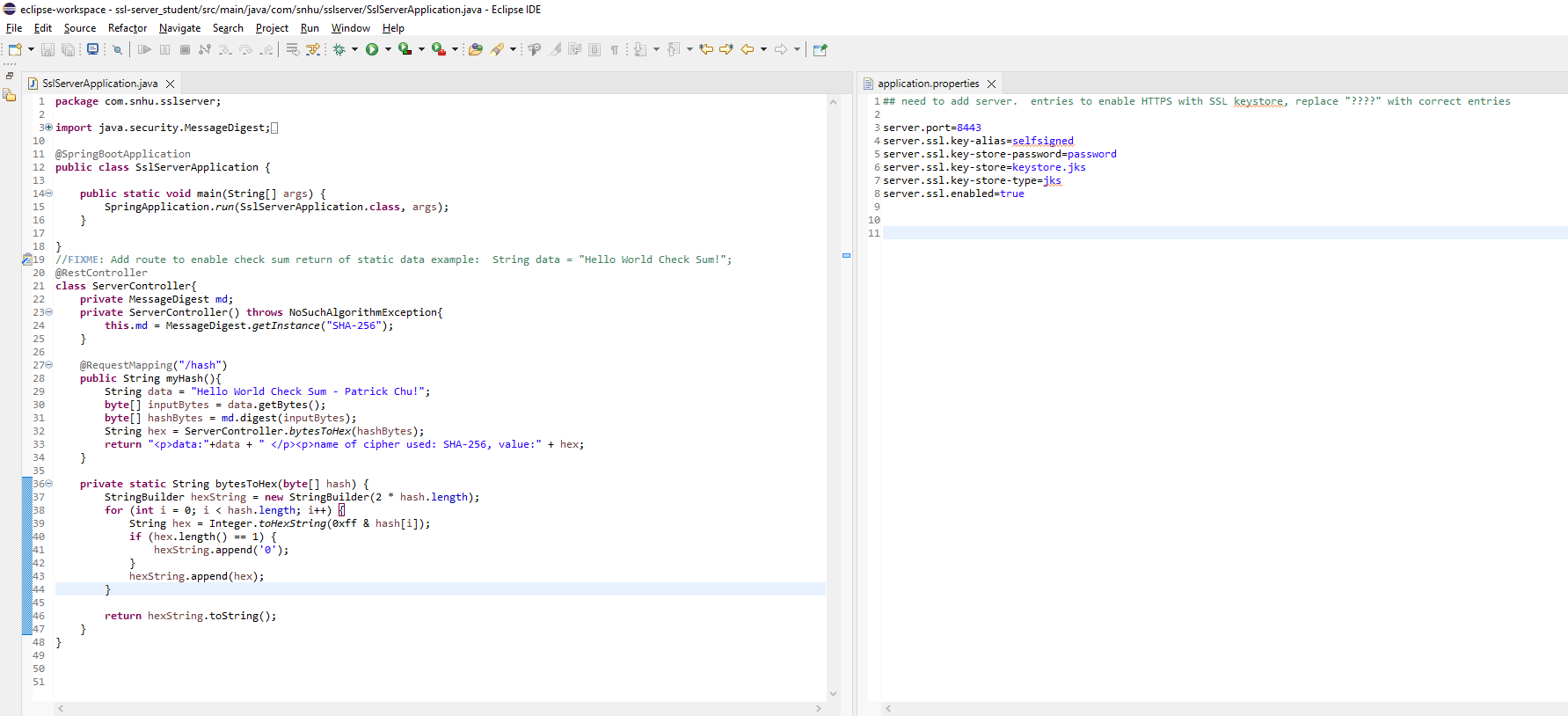
Refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code to verify secure communication by typing **https://localhost:8443/hash** in a new browser window to demonstrate that the secure communication works successfully.

* 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.



## 6. Functional Testing

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

Graphical user interface, application

Description automatically generated

The only vulnerabilities I identified were the plaintext password being stored in the application.properties file and the hard-coded text that was hashed in the hash function.

## 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.

1. APIs: api security was enhanced by utilizing https protocol
2. Cryptography: added encryption utilizing algorithm cipher/hash function and a checksum verification
3. Code Quality: Proper formatting and standard practices were used while writing code, and code was reviewed to ensure function and readability.

The process taken for adding security to the application was by creating a self-signed certificate that allows for HTTPS to be utilized in the application. I also ensured that the hashing function was written correctly, and in future builds that hashing function will be applied to ensure that data can be send securely. These steps add to the company’s security by helping to ensure that our data and our client’s data can be securely stored and is not easily retrieved or deciphered.

Best practices for our application’s security are to ensure that all of our dependencies are up to date. This allows for a greater degree of security on the basis that known vulnerabilities are addressed and attackers cannot use commonly known exploits to attack our system. Validating and enforcing privileges within the system is also very important, as it ensures that only authorized personnel and users are able to have access to sensitive information or processes, and everyone else is restricted or unable to access things within the application.

**Citations**

Chu, P, (2022). *“Chu\_Algorithm\_Ciphers”* [Unpublished manuscript]. Southern New Hampshire University.