****

# 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** | **04/23/2022** | **Krista Mosser** |  |

## 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

Krista Mosser

## 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.

.

SHA-256 is part of SHA-2 set of cryptographic hash functions, designed by the U.S. National Security Agency (NSA) and published in 2001 by the NIST as a U.S. Federal Information Processing Standard (FIPS). SHA-256 is a secure hash algorithm for computing a condensed representation of electronic data.

Hash functions are extremely useful and appear in almost all information security applications. A hash function is a mathematical function that converts a numerical input value into another compressed numerical value. The input to the hash function is of arbitrary length but output is always of fixed length. The bit level of the cipher determines the length of the output of the hash. The higher the bit level, the longer the output, and the more difficult it is to hack.

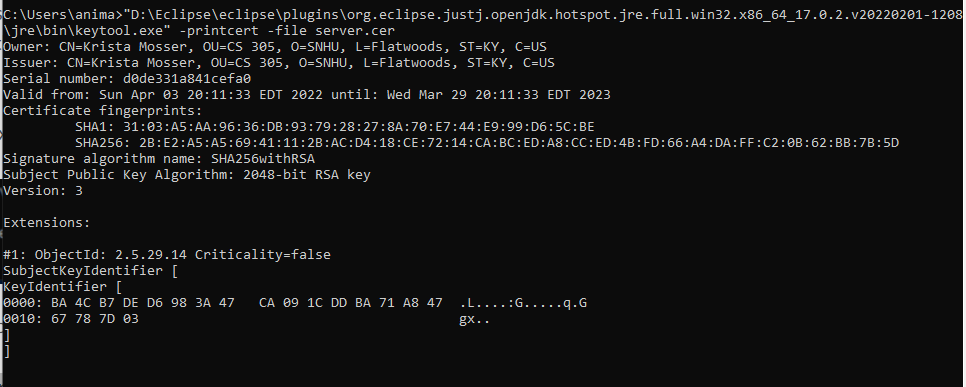
In cryptography randomness is important because it removes any reasoning and therefore any predictability. An attacker is usually trying to attain information on a system, when this information is randomly generated there are no clues as to what it maybe and therefore no open opportunities to attack the system. Symmetric encryption is a widely used data encryption technique whereby data is encrypted and decrypted using a single, secret cryptographic key. Asymmetric encryption uses mathematically linked public- and private-key pairs to encrypt and decrypt senders’ and recipients’ sensitive data. Both turn plain text into encrypted output. Symmetric encryption is very secure, taking billions of years to crack using brute force techniques, has shorter key lengths and is relatively simplistic compared to asymmetric, and it is almost universally accepted. In asymmetric encryption, key distribution is not necessary and people can use their private keys to digitally sign and verify that information is from them. Unfortunately, asymmetric encryption is much slower than symmetric and tends to bog down networks.

The history of cryptography goes all the way back to the ancient Egyptians when an Egyptian scribe used non-standard hieroglyphs in an inscription. Julius Caesar used a substitution cipher. Different substitution ciphers were used moving toward the modern era and also the use of symbols or entire words to represent characters. In 1466, the first "cipher disk" was made for polyalphabetic ciphers. Various versions of the disk or wheel were then made and used. In 1917 the father of US cryptanalysis worked for the US Government through a private corporation and then went on to start a school for military cryptanalysts. Also in 1917, a man working for AT&T invented a cipher machine capable of using a key that was totally random and never repeats. Various cipher machines based on rotors were then invented. Nazi Germany used the Enigma cipher machine, which was eventually broken through obtaining ciphertext and daily keys through espionage. The precursor of DES encryption was developed by IBM 1970. In 1976, the idea of public key cryptography was introduced. The RSA algorithm was invented in 1977. In 1997 AES (Advanced Encryption System) was adopted by NIST. As computers grow more powerful, a new system of encryption will probably need to be designed.

## 2. Certificate Generation

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

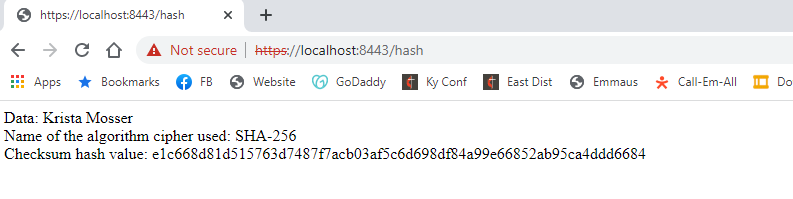
* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.



## 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.

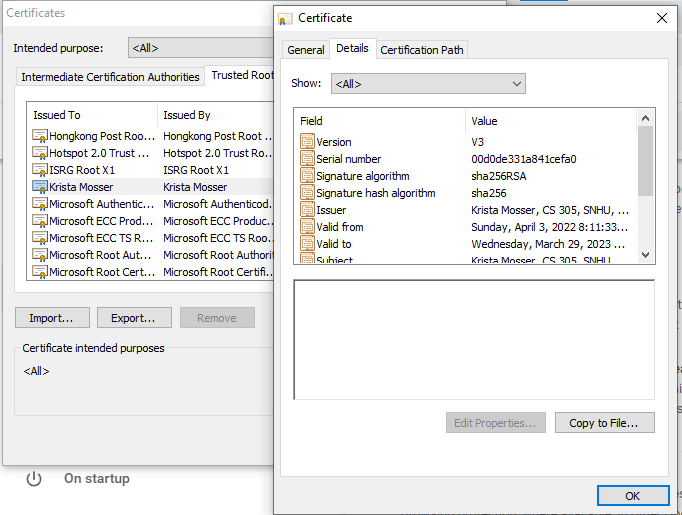


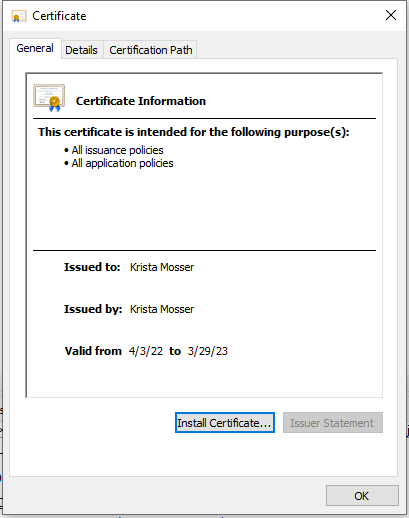
## 4. Secure Communications

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.

I cannot figure out how to get the code to use my certificate to make the site secured. As you can see in the screenshots below, I have created the certificate, and I have imported it to the Trusted Root Certificate Authority, but my browser is still using the "Tad Kellogg" certificate, which is still not trusted, even when I import it into the Trusted Root Certificate Authority.

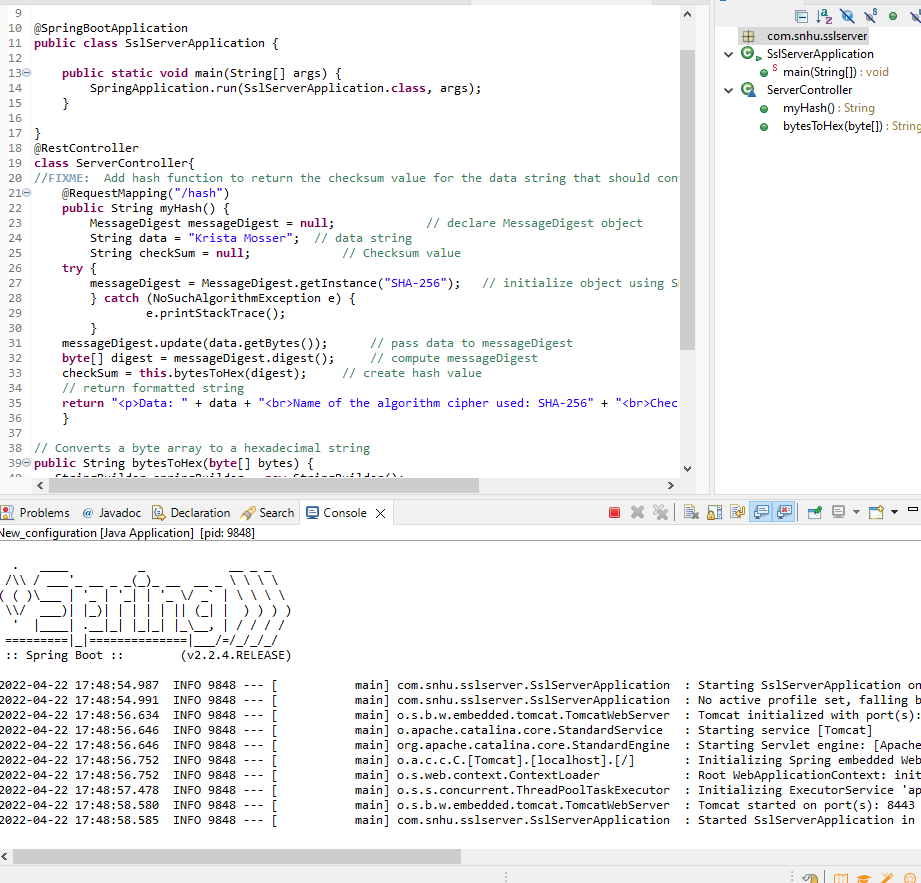




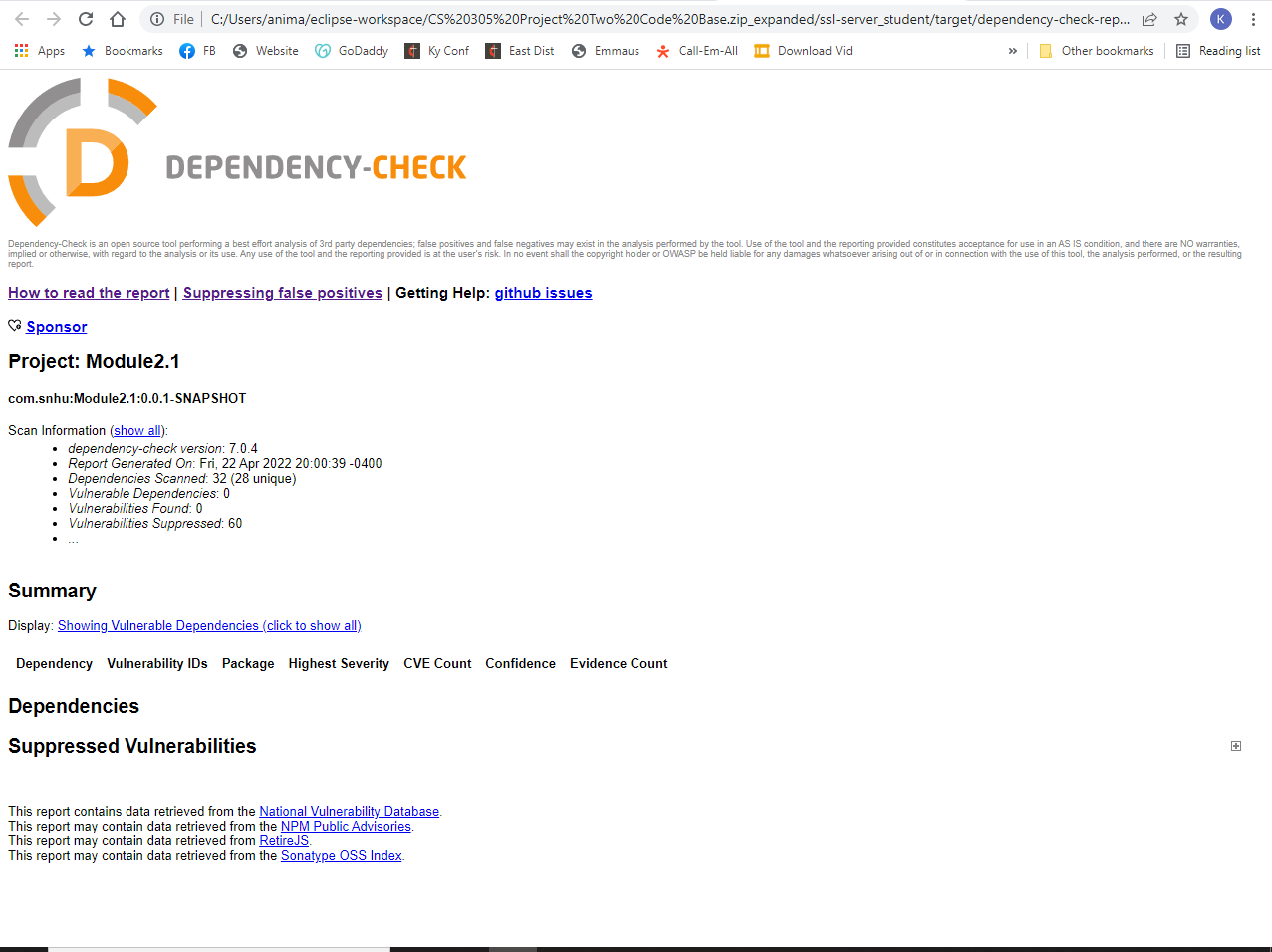
## 5. Secondary Testing

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

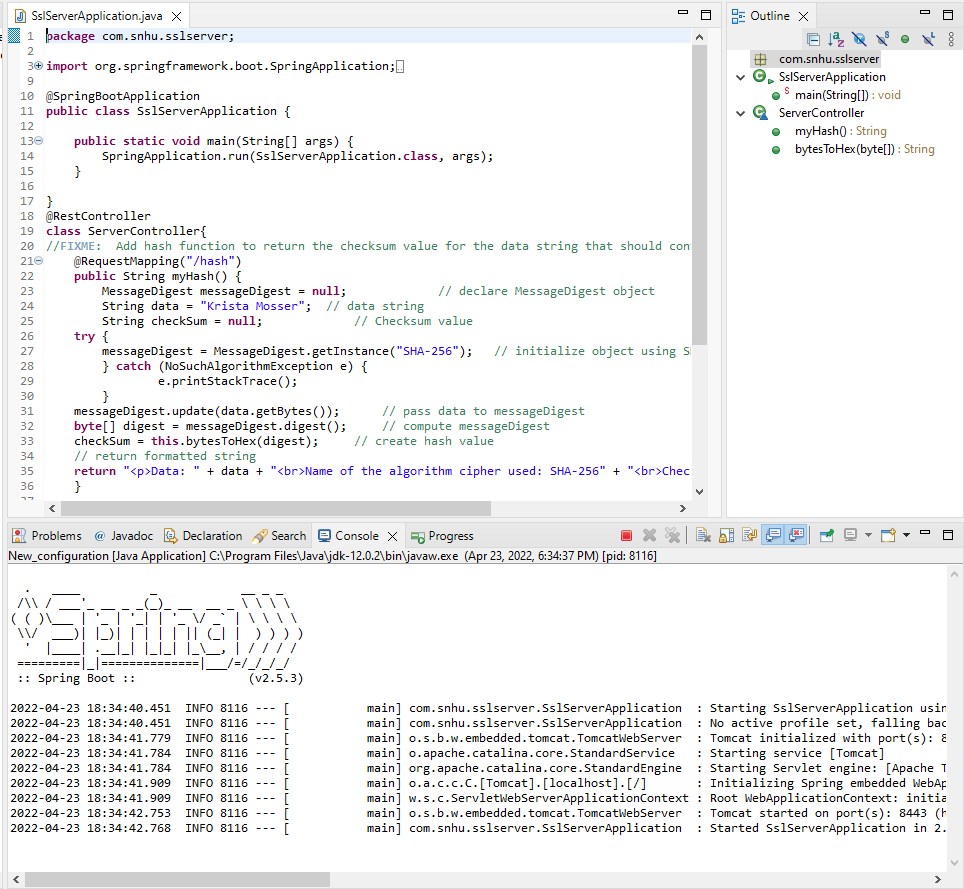


## 6. Functional Testing

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.

The refactored code executed without errors for the functional testing. There are still a few minor areas that are of risk and that the client should be aware of. Those are the use of plaintext values for the input string to the hash function and the use of a cleartext password in the application.properties file. Neither of these provide a significant vulnerability unless someone had underlying access to the uncompiled code base. Below is a screenshot of the refactored code and its execution 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 components of the vulnerability assessment being addressed are:

* APIs: The API section is addressed because of the HTTPS enablement that allows for secure interaction with the interface.
* Input Validation: The input validation while not systematically addressed is framed using the hash function and checksum value.
* Code Quality: The code quality is addressed through the functional review and utilization of best practices in code.
* Cryptography: The cryptography is addressed using RSA (through the software certificate) and SHA- 256 hash function for creating the checksum.

To add the layers of security to the software, I began by identifying the client's security needs, including any laws or government controls that needed to be addressed. A vulnerability assessment was created based on the code provided by the client by both manually examining the code and also running a dependency check. The dependency check revealed several dependencies based on outdated versions of publicly available software. This was mitigated by updating the Springboot software to a more current version. The manual examination revealed plaintext passwords and user names within the code itself and no way of checking that output customers received was indeed the output of the code. This was mitigated by the creation of a hash function to encrypt the passwords and the creation and inclusion of a security certificate. These measures make the data that Artemis stores and sends much less open to hacks, and therefore lessens the amount needed to pay out for data breaches and/or for insurance against such. The best practice for maintaining security on the client's software is to keep software versions up to date in order to address any vulnerabilities that may occur with older software. Another practice is that the code base needs to be kept separate and inaccessible to attempts at hacking.

**References**

Crane, C. (2021, January 25). *What Is a Hash Function in Cryptography? A Beginner's Guide*. Hashed Out by The SSL Store™. Retrieved March 27, 2022, from https://www.thesslstore.com/blog/what-is-a-hash-function-in-cryptography-a-beginners-guide/

Oracle. (n.d.). Java Security Standard Algorithm Names. Retrieved March 27, 2022, from https://docs.oracle.com/javase/9/docs/specs/security/standard-names.html#cipher-algorithm-names