****

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[\_Toc33111301](#_Toc33111301)

[Document Revision History 3](#_Toc33111302)

[Client 3](#_Toc33111303)

[Instructions 3](#_Toc33111304)

[Developer 4](#_Toc33111305)

[1. Algorithm Cipher 4](#_Toc33111306)

[2. Certificate Generation 5](#_Toc33111307)

[3. Deploy Cipher 5](#_Toc33111308)

[4. Secure Communications 6](#_Toc33111309)

[5. Secondary Testing 6](#_Toc33111310)

[6. Functional Testing 7](#_Toc33111311)

[7. Summary 9](#_Toc33111312)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **8/14/2022** | **Dakota McDonough** |  |

## 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

Dakota McDonough

## 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.

Encryption algorithm ciphers are used to “convert the original message… into ciphertext using a key to determine how it is done” (HYPR, 2022). Meaning we take and input and convert it to another form to prevent hackers from accessing the information either during transit or storage. For example, if we were to store data in a document using a cipher would make it harder for anyone accessing it without authorization to actually understand the information stored within.

A hash function is a type of encryption algorithm. It is “the process of scrambling raw information to the extent that it cannot reproduce back to its original form” (Simplilearn, 2022). There are various types of hash functions. A specific type of these algorithms is called a secure hash algorithm. This then has further subdivisions based on the bit level of the cipher. This is normally appended to the end of the hash style name. For example, SHA-512 is a 512 bit secure hash algorithm (Simplilearn, 2022). This means that the output value is made up of a string of 512 bits. The bigger the number of bits the more a space a system needs an, normally, the slower is runs to be able to do all the calculations needed (Simplilearn, 2022).

In encryption it is often hard to be truly random. Any number generator is never perfectly random as there are always tells for what the next value could be. For example, you can look at some of the indie videogames present in the market. I’m specifically considering a single developer game called Stardew valley. The creator accesses a random number generator, RNG, for aspects in most of the game. There are now people who play the game that have been able to determine that if a list of these following conditions happen in the game you are able to predict the entire end game for the current save file. This has resulted in full breakdown spreadsheets used by all of the top speed runners in the industry. Now, if this is the length people will go to for a video game, it’s easy to see that hackers would gladly go through the same or greater effort if it meant potential monetary gain. This is why for cryptography it is recommended that any RNG used need to “meet K3 and k4 standards” (Easttom, 2017) as outlined by the German Federal Office for Information Security. The US NIST also documents how to test a RNG being used to make sure it can be considered random enough (Easttom, 2017).

In encryption there is also the consideration of who should have access to the key used to encrypt the data. Symmetric keys refer to the idea that the same key is used to encrypt and decrypt the information, meaning that the owner and the client would have the same key (GeeksforGeeks, 2022). Asymmetric keys refer to the idea that the information is encrypted using one key and then decrypted using a different key (GeeksforGeeks, 2022). There are benefits and draw backs for both methods. Symmetric keys tend to be faster to use, but asymmetric keys are more secure. If using a symmetric key it is also important to make sure that there is secure transfer methods available for delivering the key between parties (GeeksforGeeks, 2022).

Artemis Financial has asked for additional security regarding their web based financial system. For this application I have decided to implement a secure hash function, specifically SHA-256, as the encryption algorithm cipher (Simplilearn, 2022). This allows for the security required without compromising on speed for the client. This is also the most current encryption algorithm and deemed the most secure option because it was created by the NSA and NIST to replace the original SHA-1 as brute force attacks over time had lessened its effectiveness (Simplilearn, 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.

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.

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.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

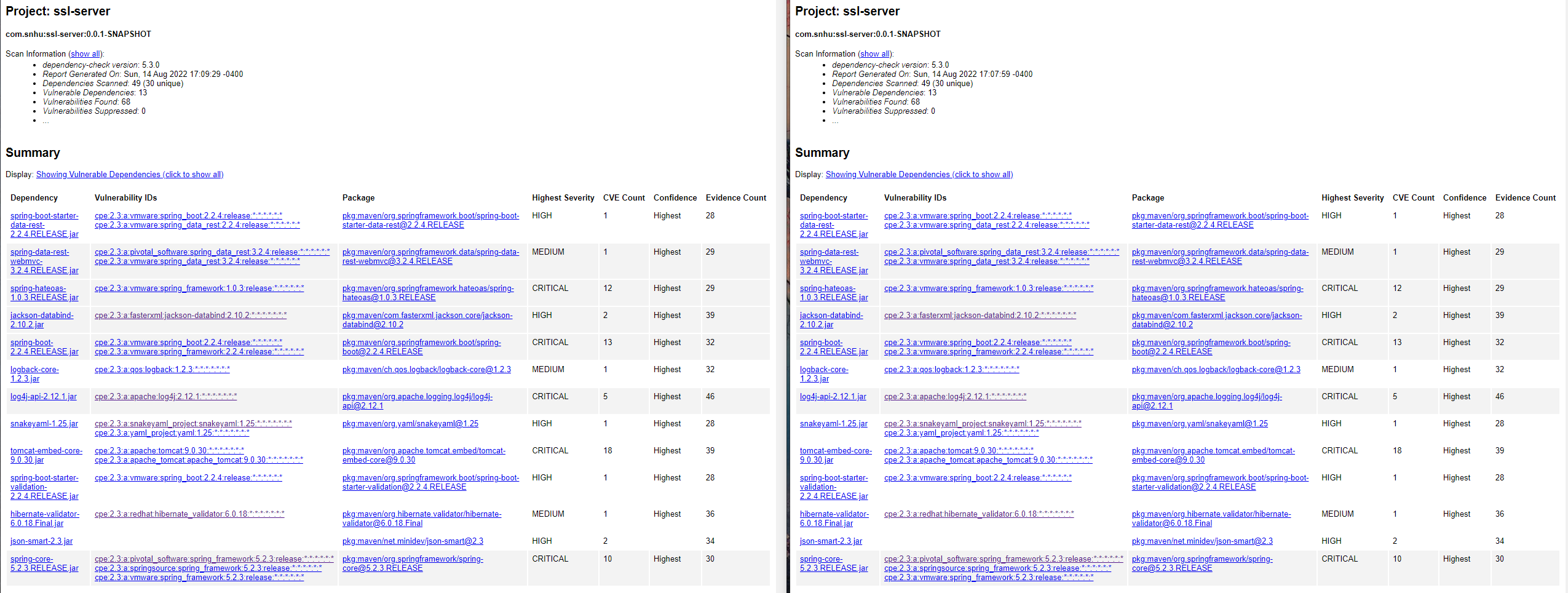
## 5. Secondary Testing

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, email

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.

Graphical user interface, text, application

Description automatically generated

Looking through my code, I have noted the following are possible vulnerabilities that need to be addressed.

I believe we should go through with updating the noted vulnerabilities in the dependency tests to make sure that they are not allowing people in through a back door.

I also believe that the inputted values for the has function should be taken from a separate file that is stored in a has value itself. This double hashing process would be a lot more secure as it would keep people from being able to access the information if the server where the code is stored is accessed (OWASP, 2022). This really depends on how critical this information is, but if the info being hashed is potentially a password of any kind I would recommend keeping it stored as a hashed value that is called when needed.

I would also note that the password used to access the key for this project is being kept in an unsecured section of the code under the application.properties. This should be kept hashed in a separate file and referenced to in the same manor that I described above.

## 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.

For this assignment my refactored mainly focused on APIs and cryptography. Secure API interaction was practiced using the RESTful API and the security key created with the java keytool. The cryptography was handled using the has function within the code to randomize the data input given within the code. Code quality was also addressed as this relates to just making sure that your code is done well and securely. Any vulnerabilities of the code itself was touched on during the vulnerability section of this report. The other areas of the vulnerability assessment process were not really touched on with the project.

The hash function used in this project is helpful in maintaining the security of the information present in the code. Utilizing the RESTful API also allows us to run the code without directly showing the areas where the sensitive information is being held. Both of these contribute to making it harder for a hacker to gain access to the critical information without extended steps.

To maintain the security of the system the customer must refrain from sharing the key created with anyone who does not have proper clearance. It is also recommended to save the code and the key separately, updating the new folder location of the key in the application.properites file. This ensures that if someone gains access to the code, they do not automatically have access to the key for deciphering. It is also recommended that the customer have a dependency check done annual at a minimum to make sure that none of the referenced libraries have created a security issue. Any dependencies that are flagged would need to be reviewed to determine if this is a false positive or not.

# References

Easttom, C. (2017, February 22). *Generating Cryptographic Keys: Will Your Random Number Generators (PRNGs) Do The Job?* Retrieved from Cyrptomathic: https://www.cryptomathic.com/news-events/blog/generating-cryptographic-keys-with-random-number-generators-prng

GeeksforGeeks. (2022, June 24). *Difference Between Symmetric and Asymmetric Key Encryption*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/difference-between-symmetric-and-asymmetric-key-encryption/

HYPR. (2022, August 14). *Cipher |Security Encyclopedia*. Retrieved from HYPR: https://www.hypr.com/cipher/#:%7E:text=Ciphers%2C%20also%20called%20encryption%20algorithms,determine%20how%20it%20is%20done.

OWASP. (2022, August 14). *Password Storage - OWASP Cheat Sheet Series*. Retrieved from OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/cheatsheets/Password\_Storage\_Cheat\_Sheet.html

Simplilearn. (2022, July 11). *What Is SHA-256 Algorithm: How it Works and Applications [2022 Edition]: Simplilearn*. Retrieved from Simplilearn.com: https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm