****

# Practices for Secure Software Report

Table of Contents

[Document Revision History 3](#_Toc102040754)

[Client 3](#_Toc102040755)

[Instructions 3](#_Toc102040756)

[Developer 4](#_Toc102040757)

[1. Algorithm Cipher 4](#_Toc102040758)

[2. Certificate Generation 4](#_Toc102040759)

[3. Deploy Cipher 4](#_Toc102040760)

[4. Secure Communications 4](#_Toc102040761)

[5. Secondary Testing 4](#_Toc102040762)

[6. Functional Testing 4](#_Toc102040763)

[7. Summary 4](#_Toc102040764)

[8. Industry Standard Best Practices 4](#_Toc102040765)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **10/11/2022** | **Maya Neely** |  |

## Client



## Instructions

Submit this completed practices for secure software report. Replace the bracketed text with the relevant information. You must document your process for writing secure communications and refactoring code that complies with software security testing protocols.

* Respond to the steps outlined below and include your findings.
* Respond using your own words. You may also choose to include images or supporting materials. If you include them, make certain to insert them in all the relevant locations in the document.
* Refer to the Project Two Guidelines and Rubric for more detailed instructions about each section of the template.

## Developer

Maya Neely

## Algorithm Cipher

Artemis Financial is a consulting company requesting our service at Global Rain to add a file verification step to their web application to ensure secure communications. My recommended encryption algorithm cipher to deploy given their security vulnerabilities is SHA-256. SHA stands for Secure Hashing Algorithm and the 256 stands for the final hash digest value which will always be 256 bits. Even though SHA-256 is the slowest of most algorithms, it is the most reliable in terms of security and is favored by most of the secure networks in the world, such as the U.S government.

According to (Nduati, 2021), hash functions "transforms data of arbitrary size into a fixed size output with its main objective being to verify data authenticity”. Hash functions make SHA practically uncrackable. Bit levels refer to the encryption size of the cipher, with higher bit size creating a higher security level. Higher bit levels are more secure since they create longer encrypted messages, another aspect that makes a cipher uncrackable. The use of random characters in the ciphertext removes any chance of predictability since there are no patterns. Without patterns, the message cannot be cracked, and no amount of knowledge or equipment will allow someone to guess a ciphertext of such length (Manico et al, 2014). This algorithm is also asymmetric being as though it uses a public key to encrypt and a private key to decrypt. An asymmetric key is preferred over a symmetric key in this instance since it uses a single key to both encrypt and decrypt.

Encryption “can be traced back to ancient Egyptian in tombs where unusual hieroglyphs were used to obscure the meaning of text (Team, 2022)”. One of the first well known methods of encryption was Caesar’s cipher named after Julius Caesar created around 60 BC and the Playfair cipher invented by Charles Wheatstone in 1854. The earliest known encryption ciphers use symmetric keys to incorporate the utilization of non-standard symbolic representations as a substitute for the concealed data. This later evolved into machines that could encrypt data; The Hebern rotor machine, Enigma machine and Bombe machine were all used before and during WW2, leading to the invention of modern computers. “In 1973, DES was created to protect customers’ data over the internet and remained the US national standard until it was cracked in 1997 (Thales, 2021)”. In 2000, the AES became the successor of DES and is currently the trusted standard algorithm used by the U.S. government due to its invulnerability to attacks and high bit levels. With cloud computing appearing to be the way of the future, “the future of encryption and decryption will have to address how data is handled by cloud services and their customers (Team, 2022).”

## Certificate Generation

Insert a screenshot below of the CER file.

Graphical user interface, table

Description automatically generated with medium confidence

## Deploy Cipher

Insert a screenshot below of the checksum verification.

Graphical user interface, text, application, email

Description automatically generated

## Secure Communications

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

Since my certificate is self signed and I did not purchase an SSL certificate, the window will not successfully show a padlock indicating that it is secure. Instead, the browser warns that the connection is not private and the site should not be trusted.

Text

Description automatically generated

## Secondary Testing

Insert screenshots below of the refactored code executed without errors and the dependency-check report.

**Refactored code**

**Graphical user interface, text, application, email

Description automatically generated**

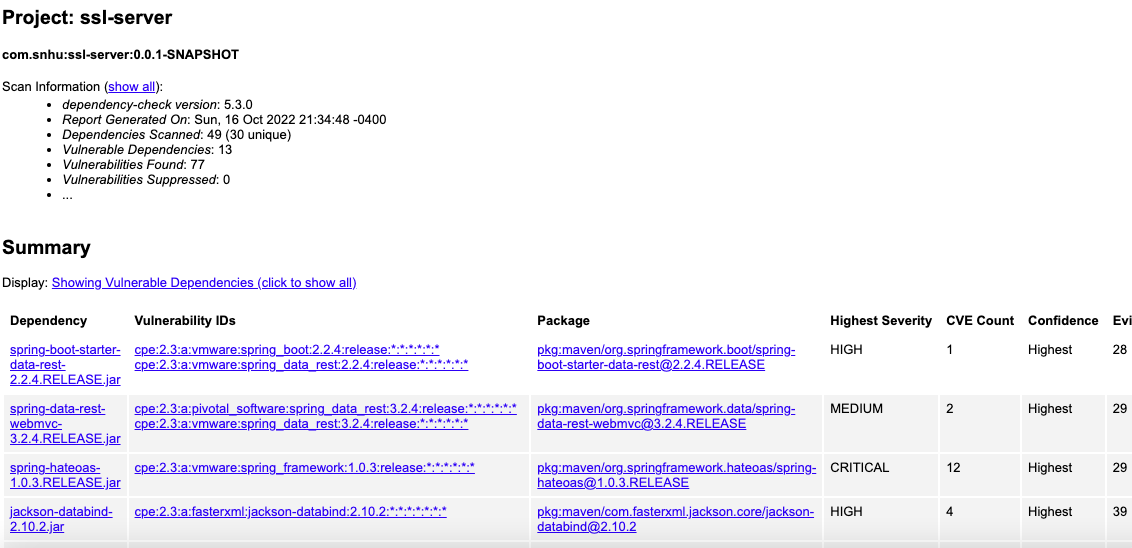
**Dependency Check**

**Pre**

**Graphical user interface, text, application, email

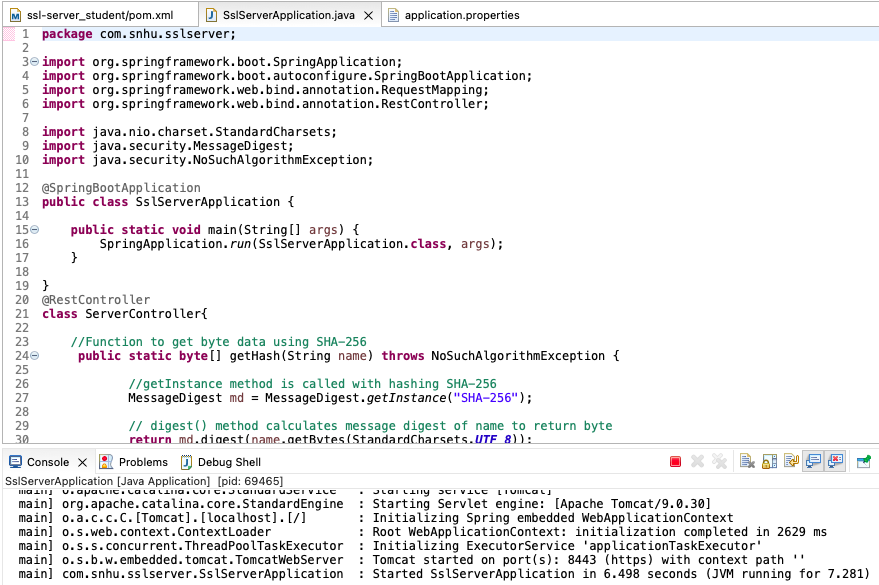
Description automatically generated**

**post**

****

## Functional Testing

Insert a screenshot below of the refactored code executed without errors.



Graphical user interface, text, application

Description automatically generated

Syntactical vulnerabilities are minimized within the code being as though majority of the code is templated. An assumption that users will use the software for its intended purposes is one-way logical vulnerabilities arise. There are parts within the code that are hardcoded, making it more difficult to clean enabling attackers to manipulate the functionality of the code. The security vulnerabilities were detected by the dependency checks reports and in most cases can easily be resolved by updating the dependencies to their current versions. Another solution can be to follow industry standard coding practices and not to mention suppressing vulnerabilities.

## Summary

The purpose of the refactored Project Two Code Base is to add a file verification step to Artemis Financial’s web application to ensure secure communications. Secure communications are a crucial aspect to consider when protecting the client data and financial information of the company. A security breach could lead to financial loss as a result of fines, lawsuits and funds needed to address the breach. Due to this, I chose to implement SHA-256 as the most secure algorithm cipher for its reliability and high bit count of 256. I added the @RestController annotation above the ServerController class to make the web service restful. Within the class, the security vulnerabilities addressed were API’s, cryptography, client/server and code quality. API’s were secure by use of a one-way hashing password, cryptography was enforced by use of the SHA algorithm which had to be securely distributed. RequestMapping is then used with the hash method to map requests onto the controller class.

For an added layer of security, dependency checks of the application should occur frequently to check for vulnerabilities. This will be especially helpful if new functions are implemented, updated or patched because this is when new vulnerabilities are most likely to arise. It is also essential that the software application is safe against the most dangerous web vulnerability, SQL Injection. The best defense against this type of attack is by use of parameterized queries and use of properly constructed stored procedures (OWASP, 2021).

## Industry Standard Best Practices

**References**

Manico, J., & Detlefsen, A. (2014). *Iron-clad Java.* Retrieved October 15, 2022, from https://learning.oreilly.com/library/view/iron-clad-java/9780071835886/ch06.html#ch06lev2sec6

Nduati, J. (2021). *Understanding hashing in cryptography*. Retrieved October 15, 2022, from <https://www.section.io/engineering-education/understand-hashing-in-cryptography/#cryptographic-hash-functions>

OWASP. (2021). *Query parameterization cheat sheet¶*. Retrieved October 16, 2022, from https://cheatsheetseries.owasp.org/cheatsheets/Query\_Parameterization\_Cheat\_Sheet.html#using-cold-fusion-built-in-feature

Team, T. (2022). *The history of encryption: The roots of modern-day cyber-security*. Tresorit Blog. Retrieved September 25, 2022, from <https://tresorit.com/blog/the-history-of-encryption-the-roots-of-modern-day-cyber-security/>

Thales. (2021, October 01). *A brief history of encryption (and cryptography).* Retrieved October 15, 2022, from https://www.thalesgroup.com/en/markets/digital-identity-and-security/magazine/brief-history-encryption