****

**CS 305 Project Two**

# 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** | **February 18, 2023** | **Blake Kemp** |  |

## 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

Blake Kemp

## Algorithm Cipher

Artemis Financial has expressed a need for enhanced security measures for their web application to ensure secure communications. Given that a financial institution is highly susceptible to cyber-attacks seeking financial gain by accessing sensitive information, it is recommended to implement encryption to prevent unauthorized access. Asymmetric encryption is suggested for securing communication, whereby the public key is used for encryption and the private key is used for decryption. To ensure optimal security, it is recommended to utilize the SHA-256 cipher algorithm with 256-bit keys for encryption. This algorithm offers a high level of bit encryption and an extensive range of possible key combinations with a key length of 256 bits. Furthermore, the SHA-256 algorithm is highly secure due to its utilization of Java’s random number generator, which creates a non-reversible checksum, thus ensuring the validity of the file. As a part of the encryption process, the hash function will apply the SHA-256 cipher to generate a checksum of the provided message.

## Certificate Generation

Insert a screenshot below of the CER file.

Text

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

Graphical user interface, text, application, email

Description automatically generated

## Secure Communications.

My HTTPS site works however, because this certificate is self-assigned it is not trusted therefore displaying the “Not-Secure”

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application, table

Description automatically generated

## Secondary Testing

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

Text

Description automatically generated

Graphical user interface

Description automatically generated

## Functional Testing

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

Text

Description automatically generated

Text

Description automatically generated

## Summary

The code provided is a Spring Boot application that includes an endpoint /hash that returns a SHA-256 hash of a fixed string Hello Blake Kemp!. The code has been refactored to be more secure by adding HTTPS support using a self-assigned certificate. The certificate is specified in the application.properties file with the necessary configurations to enable SSL. The use of HTTPS improves the confidentiality and integrity of the data exchanged between the client and the server.

The code also addresses the security testing protocols by implementing secure coding practices. The use of the MessageDigest class for hashing data is a secure way of creating a checksum of the data. The code also ensures that the hash is returned over a secure HTTPS connection, reducing the risk of data interception and modification.

To add layers of security to the software application, the following practices were followed:

1. Enabling HTTPS support with a self-assigned certificate, improving confidentiality and integrity.
2. Using the secure MessageDigest class for creating a checksum of data.
3. Ensuring that the hash is returned over a secure HTTPS connection.
4. Limiting the amount of sensitive data exposed in the application code.
5. Keeping the application and dependencies up to date to prevent security vulnerabilities.

## Industry Standard Best Practices

In refactoring the provided code, several industry-standard best practices for secure coding were applied to mitigate known security vulnerabilities. These include:

1. **Using HTTPS encryption:** By configuring the application.properties file to use HTTPS encryption, we ensure that all communication between the server and clients is secure, and that sensitive data is not transmitted in plaintext.
2. **Hashing sensitive data:** By hashing sensitive data, we ensure that any data that may be intercepted is unreadable and can't be reverse engineered to reveal the original data.
3. **Limiting exposure to attack:** By refactoring the code to limit the routes and data exposed, we reduce the potential attack surface of the application, making it less vulnerable to attacks.
4. **Input validation:** By validating user input, we ensure that only valid data is accepted by the server, and that any potential attacks that use invalid or malicious data are blocked.
5. **Regular updates and patching:** By keeping the software up to date with the latest security patches and updates, we ensure that known vulnerabilities are addressed and patched before they can be exploited by attackers.

By using these best practices, we can maintain the security of the software application and reduce the risk of potential security vulnerabilities. In turn, this helps to protect the company's overall wellbeing by reducing the risk of data breaches, loss of intellectual property, reputational damage, and other potential harms. Additionally, following industry-standard best practices can help the company to stay compliant with relevant laws, regulations, and industry standards, which can help to build trust with customers and stakeholders.