****

# 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** | **2/18/2024** | **Jake** |  |

## 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

Jake Thompson

## Algorithm Cipher

## Our client Artemis Financial is looking for extra security for their web application to promote and maintain secure communication. Since most of the attacks done on a bank would be attempting to gain information about its clients to use to gain wealth or any other nefarious activities, encryption is necessary, this will help protect files from attackers by requiring a key to access. When it comes to communication type, I would recommend Asymmetric communication, this makes it to where the public is used for encryption and a private key is used for decryption. The SHA-256 algorithm would be perfect for the needs of Artemis financial, when signing a message with a digital signature, SHA-256 hash function creates a hash of the message, and the contents of the message is revealed using a public or private key. SHA-256 also has a 0.1% chance of collisions, making it safe to not really worry about collisions. The keys for sha-256 are normally 256 bits in length and you can do multiple different key combinations making it a great choice for encryptions. SHA-256 uses java’s random number generator, this creates a checksum that is irreversible, and this is used to verify validity of files, that hash function uses the SHA-256 cipher to create a checksum of the message.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screen shot of a computer

Description automatically generated

For some odd reason it would not let me run a http server on my computer also, I had to refactor the application.properties before it was able to run my checksum verification properly from my local host.

## Secure Communications

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

Same Screenshot as above, which demonstrates a https link, the link is claiming “not secure” because I used self sign, which obviously to other people I would not be considered a trusted site.

A screen shot of a computer

Description automatically generated

## Secondary Testing

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

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Functional Testing

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

A screen shot of a computer

Description automatically generated

I did not see any logical/syntactical errors in the code I ran first, so I am doubling the screenshot from “5. Secondary Testing” and “6. Functional Testing”.

## Summary

Using Spring 4.0’s @RestController helped to simplify the creation of the RESTful webservices, which is what is used to perform our checksum verification using a localhost and hashing. @RestController combines two previously used annotations, @Controller and @ResponseBody, this helped us eliminate the need to annotate every request handling method of the controller class. Our controller class that we created is called ServerController, which uses the ideas brought forth by the vulnerability assessment diagram by using the following parts, Input Validation, API’s, Cryptography, Code Error, Code Quality, and Encapsulation. Our Server Controller class contains our SHA-256 Algorithm. SHA-256 was my top choice because they were looking for secure communication, because it is asymmetrical and uses a public and private key for any message sent. SHA-256 also has one of the lowest potentials for collision, with some stating, its not even worth worrying about a collision when using SHA-256. We added different layers to our security, first we generated an authentication certificate to secure the Checksum Verification, making our local host HTTPS. Next, we used Maven Dependency Checker to see the potential dependencies the code is exhibiting, and to ensure no new vulnerabilities appear. A good way to keep up with this is by frequently running Dependency Checks to ensure no new vulnerabilities appear as time goes on. Ensuring that our POM.xml file is consistent and is constantly updated to the latest version of the software used, EX: changing our maven dependency checker plugin from version 5.3.0 to 9.0.9.

## Industry Standard Best Practices

Using industry Standard Best Practices ensures that the code you create is not only secure but efficient, logical, and syntactically correct. Using our Vulnerabilities flow chart as a guide for what is expected helped me stay within the best coding practices possible. Also using @RestController instead of Older Spring Annotations to reduce handling method request. The Value of applying these different standards is creating efficient and secure code. Within this project we used multiple different tools to ensure our codes’ security and to reduce logical fallacies within the code. Without using these practices, you are opening your project to multiple different attacks or breaches, which depending on what project you are working on, in this case Artemis Financial, can be extremely detrimental to the company and the customers the company supports.

Citations (not necessarily quotes, but pieces of information pulled from website.)

baeldung, W. by: (2024, February 28). *The spring @Controller and @RestController annotations*. Baeldung. https://www.baeldung.com/spring-controller-vs-restcontroller#:~:text=Spring%204.0%20introduced%20the%20%40RestController%20annotation%20in%20order,of%20the%20controller%20class%20with%20the%20%40ResponseBody%20annotation.

Thurmond, T. (2023, December 27). *8 best secure coding practices*. KirkpatrickPrice. https://kirkpatrickprice.com/blog/secure-coding-best-practices/