****

# 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** | **6/20/2025** | **Justin Perez** | **Added Information** |

## 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

Justin Perez

## Algorithm Cipher

I implemented SHA-256.  
  
The reason SHA-256 is used is because it’s one of the most common security measures that’s widely used and trusted. SHA-256 is also approved by the National Institute of Standards and Technology (NIST) This means that Artemis financial could avoid legal issues if in a rare event of data breach as we are doing everything in our power to protect customer data.

## Certificate Generation

Insert a screenshot below the CER file.  
  
Note: Using the name John Doe for testing purposes.

A screenshot of a computer

AI-generated content may be incorrect.

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a computer

AI-generated content may be incorrect.

## Secure Communications

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

A screenshot of a computer

AI-generated content may be incorrect.

## Secondary Testing

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

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a checklist

AI-generated content may be incorrect.

## Functional Testing

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

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Summary

In this project I mainly only added to what was already there instead of a standard refactor. I created a new controller that is for SHA-256 to return a checksum for a static string. This can be used to check that data isn’t changed or lost during transmission. Next, I created a self-signed certificate using Javas’s key tool that is stored inside a. p12 keystore. I then updated the application properties to enable HTTPS. After I’ve done that, I then did a dependency check to ensure the code will be ready for years to come. This will allow the company to update the software in the future as well as adapt to new alerts and potential vulnerabilities in the future. This also keeps a log of every vulnerability proving the company is doing its best to protect customer data.

## Industry Standard Best Practices

For this project I used the best industry standard practices like:

* HTTPS
  + HTTPS ensures data is private unlike HTTP
* Self-Signed Certificate
  + Though we are only using a self-signed certification. A self-signed certification is a common way of stopping MITM attack. In the future we can consider purchasing a certification instead.
* SHA-256
  + SHA-256 is a commonly used government level hashing algorithm. It allows for data to be private and verified.
* OWASP Dependency Check
  + This will allow the company to stay up to date on updates as well as security issues that could arise in the future.
* Stores keys in a. p12 format
  + .p12 format ensures that unauthorized users can’t have access to encryption keys that the company uses.
* Absolutely no hardcoded sensitive data in source files.
  + Nothing was hardcoded in case the source files get leaked ensuring safety.