****

# 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** | **10/20/2024** | **Emily Schupbach** |  |

## 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

Emily Schupbach

## Algorithm Cipher

In reviewing the requirements for Artemis Financial related to personal information and financial security needs, I believe the SHA-256 algorithm is the best secure hash algorithm for their project. SHA 256 is one of the most widely used hash algorithms due to its high level of security, combined with a strong processing speed that is not too clunky.

With a bit count of 256, there are too great a number of possible iterations that it would be next to impossible to crack the security through brute force, making this a coveted secure hash algorithm. Additionally, by its design, the hash functions (including SHA 256) are irreversible, further improving the security. This non-symmetric key means that the hashing can not be reversed with the same algorithm, making the process more secure.

## Certificate Generation

Insert a screenshot below of the CER file.

A computer screen with white text

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a computer program

Description automatically generated

A close-up of a computer screen

Description automatically generated

## Secure Communications

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

A screenshot of a computer

Description automatically generated

## Secondary Testing

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

A close-up of a computer screen

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 computer code with many black and white text

Description automatically generated

## Summary

In summary, I have integrated the SHA-256 hash algorithm within the program due to its standard of security currently within the field. With a 256 size bit count, this reduces the risk of a brute force attack due to the sheer size of possible combinations, which would be nearly impossible to iterate through in any scalable manner. Additionally, by ensuring the edits have not generated any new security concerns through the static testing, we can feel confident that for the time being, the program is secure. As a general best practice however, we as a team should regularly review our code and re-run the static testing to ensure no newly-found security risks have arisen.

In this program in particular, we are using various methods of cryptography and secure client/server connection points through the https connection; further vulnerability assessment can provide a more robust input validation that can throw errors based off of unexpected inputs (e.g. too long, certain phrases, etc).

## Industry Standard Best Practices

As a general best practice, incorporating software security measures from the start of development is of the utmost importance. It is far easier as a team to develop secure software from the beginning, instead of trying to retroactively fit security measures after development. Additionally, this formal review of code through both a static testing and dynamic testing will help catch any potential vulnerabilities. Doing research on best practices, such as the use of the ‘gold-standard’ SHA-256 hashing encryption can also help ensure our software is as secure as possible.

Additionally, adopting a mentality that security is a 24/7, 365 day priority as a team will ensure that the level of security we establish now will be maintained and expanded continuously throughout the project. Continued education for the team, as well as internal trainings will ensure all employees working on the project prioritize security, especially for such an at-risk sector such as financial information. The security of the end clients maintains a level of trust that they have with Artemis Financial, which is pivotal for the company’s overall success. If consumers don’t trust Artemis to maintain security of their personal information, as well as their finances, the core business is compromised.