****

# 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** | **04/16/24** | **Brennon Fultz** |  |

## 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

Brennon Fultz

## Algorithm Cipher

Encryption algorithms have been in use for decades to protect data from being accessed by unauthorized attackers. Encrypting data prevents it from being interpreted by a third party if it is intercepted in transit, which is a common occurrence especially when dealing with financial information. Encryption algorithms use random numbers generated by the user’s computer in order to ensure that the encrypted information cannot be recreated. Some algorithms use symmetric keys, meaning the same key is used to decrypt the data that was used to encrypt it, and this key must be kept private. Other algorithms use asymmetric encryption, in which one key is public and one is private. The public key is used to encrypt the data and the private key is used to decrypt. When deciding on an encryption algorithm, it is important to use one that is well known and trusted because this means the cipher has been thoroughly tested in real applications. Different algorithms exist for different purposes as well, with some being better for large amounts of data, such as symmetric encryption algorithms. For the purposes of generating a checksum to ensure data integrity, we recommend the use of SHA-256 for Artemis Financial’s web application. SHA-256 is the 256-bit-level version of the Secure Hashing Algorithm created by NSA and NIST (Jena, 2023). Bit levels are a measure of the strength of an encryption algorithm. This algorithm cipher has been in use for over 20 years in many enterprise level applications and it is highly collision resistant. Collision resistance means the algorithm is not vulnerable to attacks which attempt to produce the correct output with an inauthentic input. Ensuring an authentic input is what enables the checksum to be used to verify the integrity of data being transferred.

## 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 screenshot of a computer

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 screenshot of a computer

Description automatically generated

A screenshot 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 screenshot of a computer

Description automatically generated

## Summary

The key areas of security from the Vulnerability Assessment Process Flow Diagram are Client/Server Interactions, Code Error, and Cryptography. The first part of the process was to generate a self-signed certificate to provide authentication. HTTPS protocol was also implemented to make communication between the client and server secure. Then the code was refactored to enable a SHA-256 checksum verification, thus utilizing a cryptographic hash function. The code was then static tested for vulnerabilities and one additional improvement was made, adding secure error handling.

## Industry Standard Best Practices

Using a checksum with SHA-256 is an industry standard best practice for secure coding, as is the implementation of HTTPS for communication. These prevent data from being intercepted in transit and also verify the authenticity of data that is received (Manico & Detlefsen, 2014). Secure error handling is also an industry standard best practice which ensures error messages do not display sensitive code vulnerabilities to an attacker. Static testing is also a practice used by the industry to review code and assess potential vulnerabilities, preventing attackers from exploiting dependencies. Using these best practices will help to keep the Artemis Financial application secure and reduce the possibility of security failures. Given the nature of the data being handled by this application, any possible security vulnerability should be addressed to prevent financial losses and legal liability. Applying secure coding best practices is also an ethical consideration, particularly due to the trust users place on applications like these to handle their private information. Securing customer data should be a top priority for Artemis Financial to build and maintain trust and brand loyalty.

References

Jeva, B.K. (August 29, 2023). *A definitive guide to learn the SHA-256.* SimpliLearn. https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm

Manico, Jim and Detlefsen, August. *Iron-Clad Java: Building Secure Web Applications*. New York: McGraw-Hill Education, 2014.