****

# 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** | **2/16/23** | **Zac McBride** |  |

## Client



## Developer

Zac McBride

## Algorithm Cipher

## For Artemis Financial it is recommended to use an AES cipher as the application being developed needs to have high priority for encryption. This cipher is certified for use by the US government so there is no need to worry about regulations for handling financial information. AES uses a symmetric key which is less secure but more efficient. In this case, the tradeoff for efficiency is justifiable as it is unlikely the key will be discovered if properly guarded. A key of 256 bits would be best however if speed is of any concern a 128-bit key may be used (Cowper et al., 2022). For the purposes of this application a SHA-256 cipher algorithm will be used. It is also important to note that this algorithm is a well known and commonly used algorithm today.

## Certificate Generation

Text

Description automatically generated

## Deploy Cipher

Graphical user interface, text, application, email

Description automatically generated

## Secure Communications

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

## Secondary Testing

## Table Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

## Functional Testing

Graphical user interface, text

Description automatically generated

## Summary

As this was a skeletal structure for the application focus was made on Client/Server and Cryptography as well as secure APIs. Other parts of the Vulnerability Assessment Process will be addressed when developing code for the internals of the application. APIs need to be looked at as they can be vulnerable to attacks if they are not up to date. At times there are no fixes for a newly found vulnerability and those are suppressed in the dependency report within the application being developed. They can be reviewed and changed once fixes are found to the vulnerabilities. The application attempts to address Client/Server vulnerabilities by using a certificate to verify that the information is from the right sources. Though a self-signed CA is used, in practice it would be far better to purchase a registered CA so that they can be verified on any terminal that is accessing the cite. As this application will be sending protected information a hash algorithm is used to encrypt the data being handled. A very basic set of code is used for demonstrating how it may work but the idea can be expanded so that all data sent through the application can be encrypted.

## Industry Standard Best Practices

The most important thing that done within the skeletal application for this report is the use of a CA. Making sure that a CA is used so that a secure connection can be made to the server is very important for REST applications (though this is missing several components to make it a Rest application). As stated, before it is important to make sure to use a CA that can be verified. Registered Cas can be found for free use however it may be more beneficial for an organization to purchase a CA that will most likely be a more secure option. This is especially the case for Artemis Financial as they handle highly protected personal and financial data. Though this application does not use Input Validation outside of the spring boot framework, this is another important thing to consider when coding the rest of the structure as this can create vulnerabilities. The other important thing to think about for the rest of the application is Encapsulation. The rest of the code needs to be structured in a way that does not leave holes in the application. Encapsulation also reduces redundant code and therefore reducing the size of the application. Sometimes having less code means that there are less places for a vulnerability to be found.

**Citations**

Cowper, B., Vilaysom, S., Sharkey, K., Lanfear, T., & Lama, S. (2022, June 6). *Microsoft SDL Cryptographic Recommendations - Security Documentation*. Security Documentation | Microsoft Learn. Retrieved January 26, 2023, from https://learn.microsoft.com/en-us/security/sdl/cryptographic-recommendations