****

# 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** | **8/15/2024** | **Rylan Champion** |  |

## 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

Rylan Champion

## Algorithm Cipher

Artemis Financial deals with highly sensitive client information, including savings, retirement, investments, and insurance data. Given the company’s need to modernize its operations with a strong focus on security, I believe SHA-512 is the right choice for the job. SHA-512 is part of the SHA-2 family and produces a 512-bit hash value, which means it generates a long and secure output that is difficult to break.

One of the big advantages of SHA-512 is its collision resistance, meaning it’s designed to ensure that no two pieces of data will produce the same hash. This is key for Artemis Financial, as it helps maintain the integrity of client data during transfers. Additionally, SHA-512 is a trusted algorithm that is widely used in the industry, so you know the financial data will be protected by a well-established and reliable method.

When it comes to hash functions and bit levels, SHA-512 generates a 128-character hexadecimal hash. This might sound complex, but the main point is that the longer the hash, the harder it is to reverse or break. Unlike encryption algorithms, SHA-512 doesn’t use keys this is because it’s not meant to encrypt data but to verify it. If encryption is needed, other algorithms like AES or RSA would come into play, which use symmetric or asymmetric keys depending on the situation.

As for the history, SHA-512 is part of a family of algorithms introduced by NIST in 2001, and it’s still considered secure today. Using SHA-512 in Artemis Financials’ application aligns with modern cryptographic standards, ensuring the company stays ahead of potential security threats.

By integrating SHA-512, Global Rain helps Artemis Financial ensure its data verification process is secure, building trust with clients and demonstrating a commitment to using up-to-date security measures.

## 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

The above screenshot shows the website was accessed using the https secure protocol. It

also indicates an error in the trust of the underlying certificate. While I believe regenerating the keystore on the name localhost:8443 would correct the issue. I tried that and it didn’t work. I went down the rabbit hole and got lost and decided to move on.

## Secondary Testing

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

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA 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

A screenshot of a computer

Description automatically generated

## Summary & 8. Best practices

In my recent project, I've taken significant steps to enhance security by refactoring the code and integrating various security protocols. Here’s a summary of the key changes and practices applied:

Refactoring and Security Enhancements

Refactoring Code to Address Vulnerabilities

I began by closely examining the Vulnerability Assessment Process Flow Diagram to identify critical areas needing improvement. One major focus was on refining how we handle checksum hash tables. This refactoring improved data integrity and tamper detection by ensuring that any unauthorized changes to files are promptly identified. Enhancements in the checksum hash process have added a robust layer of verification, thus preventing potential data tampering and maintaining file integrity.

Adding Layers of Security:

For server/client security, I implemented strong encryption practices, specifically utilizing 512-bit encryption methods. This upgrade significantly bolstered the protection of sensitive data by ensuring that only authorized personnel with the correct key can access secure files. Additionally, I integrated an SSL certificate to secure the site and complement our encryption strategy, ensuring all sensitive information is protected during transmission.

Enforcement of HTTPS was another critical update. By mandating HTTPS, I ensured that all data transmitted between users and the server is conducted over a secure channel, thus preventing any unprotected HTTP transmissions. This measure is vital for maintaining the confidentiality and integrity of user data.

Best Practices for Secure Coding:

Following industry-standard best practices has been central to maintaining and improving the software application's security. Regular updates and patches for dependencies were performed to address known vulnerabilities. Thorough security testing was conducted to identify and mitigate potential risks. Implementing strong encryption algorithms, like SHA-512, has been crucial in safeguarding data against attacks.

Industry Standard Best Practices

Maintaining Existing Security:

By adhering to industry-standard best practices, such as applying the latest security patches and updates, I ensured that the software application remains secure. Implementing robust encryption algorithms and enforcing secure communication channels (HTTPS) have been key practices in maintaining the application’s security.

Value to the Company:

Applying these best practices for secure coding is vital for the company’s overall well-being. It ensures that we proactively address potential vulnerabilities and reduce the risk of data breaches. By integrating security measures throughout the development cycle and maintaining a vigilant approach to security, we not only protect our systems and data but also build trust with clients and stakeholders. This ongoing commitment to security fosters a more resilient and reliable software environment, ultimately contributing to the company's success and reputation.

Challenges and Future Improvements

Certificate Trust Issues

One area I encountered challenges with was getting the SSL certificate to be trusted by the server. This issue needs further investigation and resolution to ensure proper certificate validation and trust.

Suppressing False Positives

Another aspect requiring attention is learning how to effectively suppress false positives in the dependency check. Fine-tuning this process will help in accurately identifying true vulnerabilities and avoiding unnecessary alerts.

In conclusion, by refactoring the code and implementing these security measures, I’ve significantly enhanced the software's security posture. This effort is aligned with industry standards and best practices, contributing to a more secure and reliable application. Continuous attention and improvement are essential to maintain and advance security throughout the development lifecycle.