****

# 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** | **10/15/2024** | **Darrell Walker** | **Initial Creation** |

## 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

Darrell Walker II

## Algorithm Cipher

Based on Artemis Financials requirements for data verification and secure communications, I recommend AES.

 **Overview**: AES is a symmetric encryption algorithm known for its speed and security. It's widely used in modern encryption tasks.

 **Hash Functions and Bit Levels**: AES typically supports 128-bit, 192-bit, and 256-bit key sizes. The larger the key size, the more secure the encryption.

 **Symmetric vs. Non-Symmetric**: AES is symmetric, meaning the same key is used for encryption and decryption. This is ideal for efficient encryption of large amounts of data, like financial transactions.

 **Use of Random Numbers**: Secure random numbers are used in generating keys and initializing encryption processes, crucial for the strength of AES encryption.

* **History**: AES was introduced in 2001 to replace DES, offering stronger security due to its longer key sizes and more complex encryption mechanisms.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer program

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 error

Description automatically generated

## Functional Testing

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

A screenshot of a computer program

Description automatically generated

## Summary

1. **Using HTTPS for Secure Communication:**

* The code was changed to use HTTPS instead of HTTP, which means that all data sent between the client and server is now encrypted. This helps protect sensitive information, such as customer data, from being stolen during transmission. I configured the SSL certificate to run the application securely on port 8443, which makes the communication safe from attacks.

1. Adding SHA-256 Checksum for Data Integrity:

* I added a feature that uses the SHA-256 algorithm to create a checksum. This helps verify that the data hasn’t been tampered with. The checksum was set up in the Checksum Controller class and can be used to check if the data stays the same when it's being transferred or stored. This adds an extra layer of security for important data.

1. **Running a Vulnerability Check with OWASP:**

* I used the latest version of the OWASP Dependency-Check tool to scan the project’s third-party libraries for security problems. This makes sure that no known vulnerabilities were introduced through the libraries I used. The scan confirmed that the libraries are safe to use.

Conclusion:

The code updates focused on securing the application by using industry best practices. I now have encrypted communication with HTTPS, data integrity checking with SHA-256, and a tool to find vulnerabilities in our libraries. These changes help protect sensitive information and ensure the application meets modern security standards.

## Industry Standard Best Practices

1. **Using Best Practices to Keep the Application Secure**:
   * I followed well-known security practices to make sure the application stays safe. One of the key things I did was switch to HTTPS using Transport Layer Security (TLS). This encrypts all the data sent between the client and the server, protecting it from attacks like man-in-the-middle attacks, where someone might try to steal data as it's being sent.
   * I also used SHA-256 for hashing data, which creates a secure “fingerprint” to check if data has been changed. SHA-256 is strong and secure, making it very hard for hackers to create fake data with the same hash.
   * Input validation was another important step to block common attacks like SQL injection and cross-site scripting (XSS). By checking and cleaning input data, I made sure that no harmful data could enter the system.
   * I used OWASP Dependency-Check to scan the third-party libraries used in the project. This helps catch any known security problems in the libraries, which could otherwise leave the system vulnerable.
2. **Why Best Practices Are Important for the Company**:
   * **Protecting Sensitive Data**: Using secure coding practices helps keep private information, like customer financial data, safe. By using HTTPS and strong encryption, I lowered the chances of data being stolen. This is important for protecting the company’s clients and keeping their trust.
   * **Preventing Security Issues**: Following best practices, like validating input and scanning for vulnerabilities, helps prevent many common attacks before they can happen. This makes the application much more secure and reduces the need for emergency fixes later.
   * **Following the Law**: Many companies, especially in finance, must follow strict laws that require them to protect customer data, like GDPR or PCI DSS. By following security best practices, Artemis Financial can avoid fines or legal trouble and stay in line with these rules.
   * **Building a Good Reputation**: Companies that focus on security are seen as more trustworthy. When customers know their data is being handled securely, they’re more likely to do business with Artemis Financial. This can also make the company stand out from competitors.
   * **Saving Time and Money**: By building security into the application from the start, the company avoids spending time and money fixing security problems later. This helps the company grow without having to constantly worry about security breaches.

In conclusion, by following industry-standard security practices, I’ve made the application stronger and safer. This not only protects Artemis Financial’s clients but also helps the company avoid risks and grow its reputation as a trustworthy business.