****

# 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** | **[Date]** | **[Your Name]** |  |

## 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

Dylan Harmon

## Algorithm Cipher

Artemis Financial wants to modernize its operations and use the most effective software security, add file verifications, and to ensure secure communications. The company’s mission is “Security is everyone’s responsibility”. With this in mind I would recommend using SHA256 Secure Hash Algorithm. The hash algorithm is not a cipher, as in it does not require keys to decipher. What SHA256 does instead is takes an input and produces a fixed sized (in this case 256bit) hash value. SHA256 Aims to provide data integrity and authenticity while generating a unique “fingerprint” for each unique input.

The SHA256 hash function offers great security against brute force attacks because the output is 256 bits, or 2256possible values. Hackers trying to use the collision risk (two inputs with the same hash) of hash functions to “crack the code” .

Why SHA256 is considered collision resistant:

* Every *bit* of output depends on every *bit* of input in a nonlinear way
* The birthday bound in this case states that you need roughly 2128 attempts to find a collision.

The hash value can’t be reversed to get the original message, and guarantees the messages integrity and authenticity. To secure communications SHA-256 can be combined with Hash-based Message Authentication Code (HMAC). HMAC combines the hash algorithm with keys allowing for verification of the messages integrity and origin.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer program

AI-generated content may be incorrect.

## Deploy Cipher

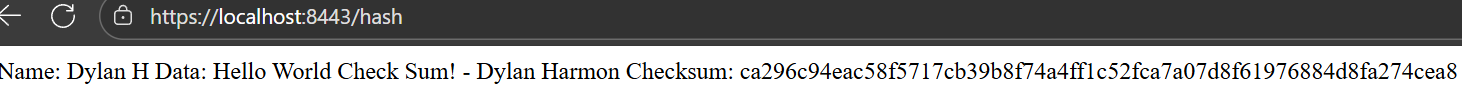
Insert a screenshot below of the checksum verification.

A close up of a screen

AI-generated content may be incorrect.

## Secure Communications

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



## Secondary Testing

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

A close up of a screen

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.c

## Functional Testing

A computer screen shot of a program code

AI-generated content may be incorrect.

## Summary

I refactored the code in order to create a secure encrypted communication and verified the changes where static with security testing. After the initial checksum I focused on replacing unsecured HTTP access with secure HTTPS and displaying data integrity with the hash algorithm.

A Restcontroller was incorporated to add an endpoint (/hash) that returns a SHA-256 checksum. A connector bean (Tomcat) was introduced in my configuration class to redirect all HTTP requests on port 8080 to 8443, this is intended to force access to a secure TLS session. The application used a self signed certificate that includes a Subject Alternative Name (SAN) that is bound to localhost. This allows for Chrome to recognize the connection is secure.

I ran a OWASP Dependency-Check against the refactored code and verified that no *new* vulnerabilities were introduced. I also updated the maven dependency check version the news available version.

Security Layers Added:

* TLS: HTTPS Encryption protects against man-in-the-middle attacks.
* Forced Redirect: Utilize a connector to redirect all HTTP traffic to secure HTTPS (localhost)
* Server authentication with Self signed certificate

## Industry Standard Best Practices

By following the OWASP Secure Coding Practices, I enforced TLS for all communications, generated a SHA-256 checksum on the trusted servers side, and managed secret keys via a keystore. The SHA-256 checksum within the REST controller prevents tampering and ensures data integrity. The keystore safely stores private key, and fails the application if the certificate is invalid, offering protection against unauthorized access. Enforcing *HTTPS only*  and using a Connector to redirect HTTP the codebase aligns with OWASP communication security guidelines, ensuring confidentiality and integrity in transit. The REST controller returns only the checksum (and for project purposes name and data), no sensitive metadata is shown ensuring clients sensitive details are not exposed. These all help to follow OWASP Data Protection secure coding practices.

These cryptographic practices offer communication security, error handling, and data protection to ensure Customer trust and user confidence. Encrypted communications and secure key management are foundations in regulatory standards that will apply to Artemis Financial. The codebase offers scalable security, maintainable and modular code that can evolve as the requirements change, giving the application a sense of adaptability.