

# 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** | **04/15/2025** | **Mike R. Montminy** | **Added Photos** |
| **1.1** | **04/18/2025** | **Mike R. Montminy** | **Finished Document** |

## 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

Mike R. Montminy

## Algorithm Cipher

Recommendation:

I would recommend the implementation of SHA256 for Artemis Financials security systems as it offers very strong security, low collision probability, and is considered by many to be an ideal way to generate a digital signature or checksums for any application.

Overview:  
SHA256 was developed by the NSA and later published by NIST and is a one-way hash transforming input to a 256-bit hash value. In all cases the same input will have the same hash. Another upside to SHA256 is that it is near mathematically impossible to find two separate inputs that have the same hash value. In terms of digital signatures and file verifying, this is why many people consider SHA256 to be a must.

Functions & Bit Levels:

SHA256 produces a 256-bit hashing value which is presented as a 64-character hexadecimal string. The higher the bit level, the higher resistance to any potential brute-force hacking attempts that may occur. SHA256 is mainly used for ensuring data integrity. This means that nearly any alteration to the input hash, even it being a single character swap from ‘A to B’ or ‘6 to 1’ would throw out a vastly different output which makes it incredibly easy to detect any tampering with inputs.

Random Numbers & Symmetric vs. Asymmetric:

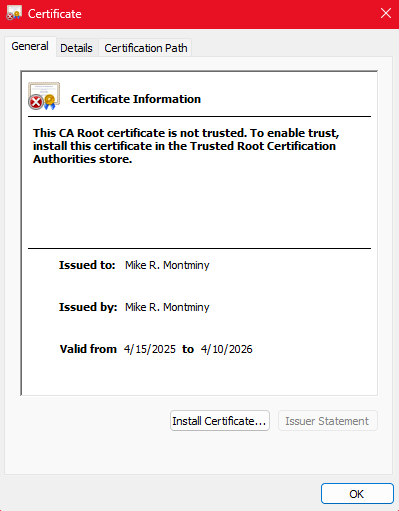
In terms of secure communications, having a cryptographically secure random number setup is essential. These help to prevent any forms of possible pattern recognition for key generation which helps to guard against brute force attacks. Symmetric keys are when the same key gets used for both encryption and decryption, while asymmetric uses private and public key pairs. Symmetric is usually much faster and efficient, but it needs to have a hyper-secure distribution method associated with it. Asymmetric tends to be ideal for key exchanging and digital signing, but due to being asymmetric they take a lot more resources to work properly.

History & Current:

The SHA-2 family of algorithms (which SHA256 is included in) was introduced back in 2001 after the original SHA-1 family was due for replacement after recent developments of cracking that algorithm. To this day SHA256 is one of, if not the, most used hashing algorithms due to the balance of speed, resistance, and security against all types of cryptographic attacks that can occur. The SHA-3 family was standardized back in 2015, but is not as widely used due to how much raw performance and compatibility the SHA-2 family currently holds.

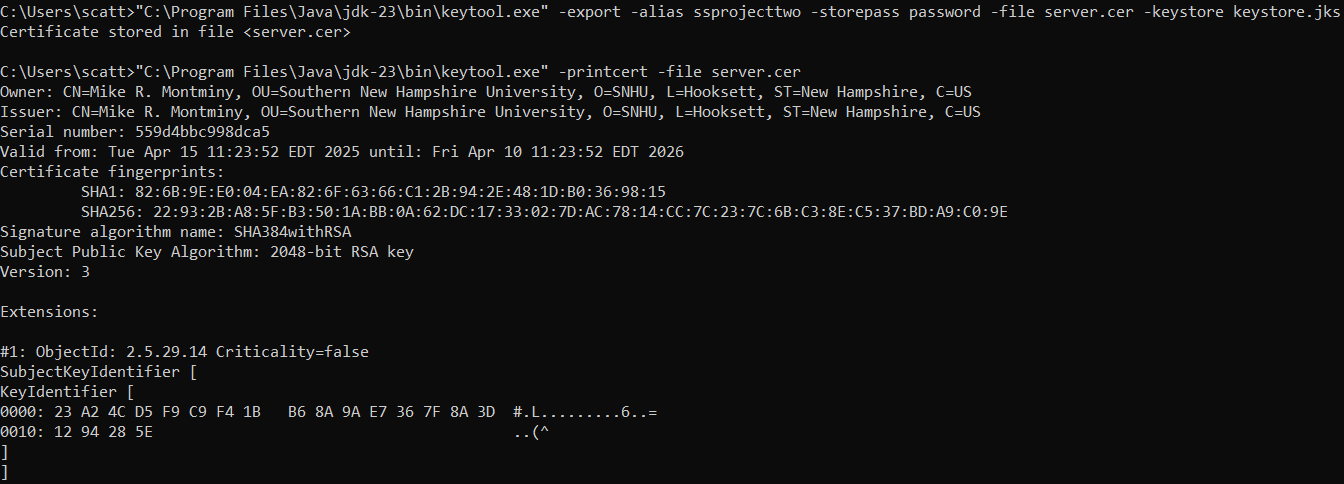
## Certificate Generation

Insert a screenshot below of the CER file.



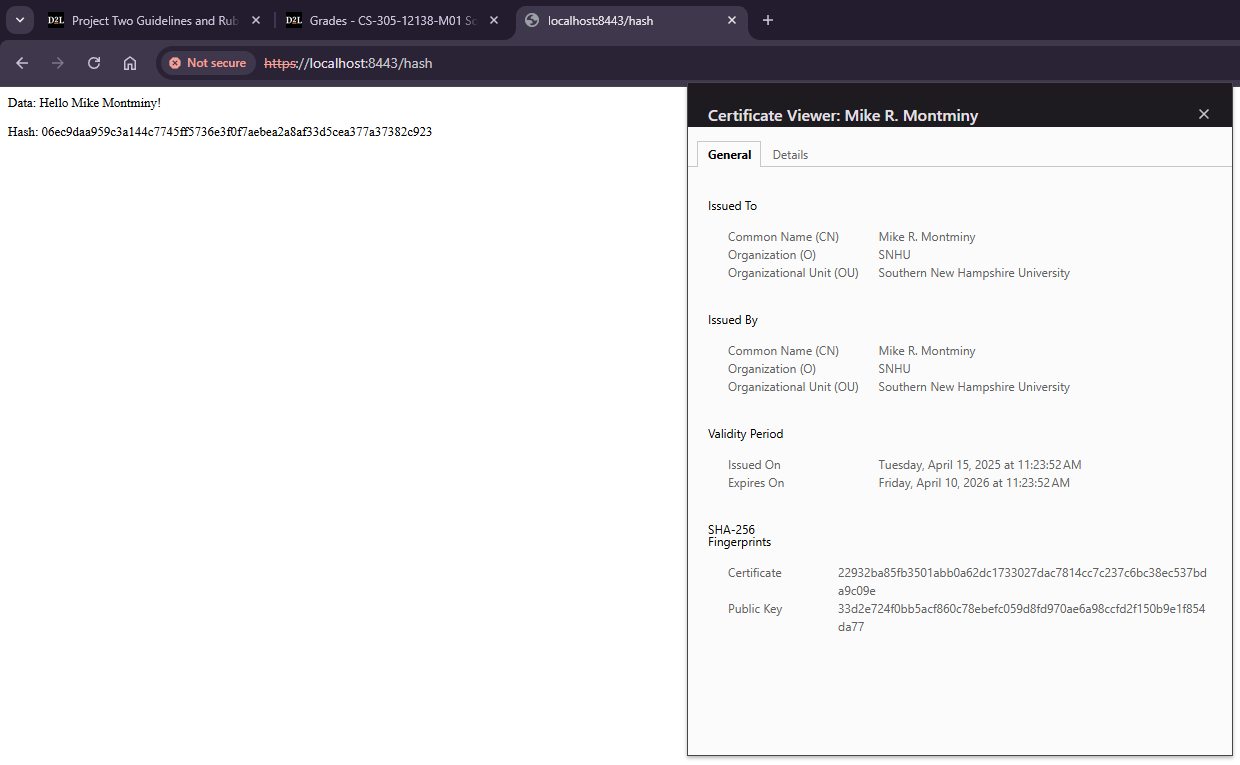
## Deploy Cipher

Insert a screenshot below of the checksum verification.



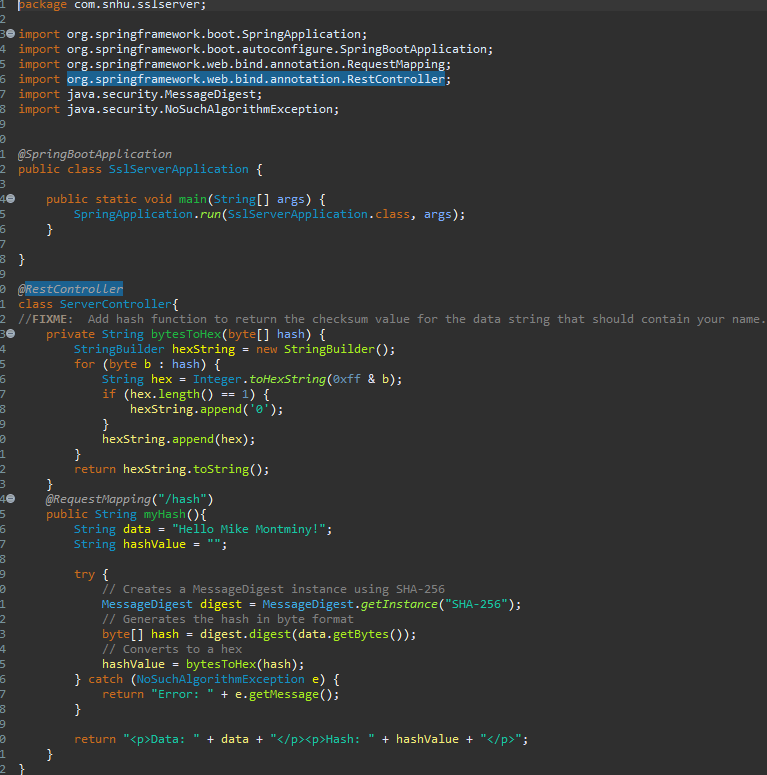
## Secure Communications

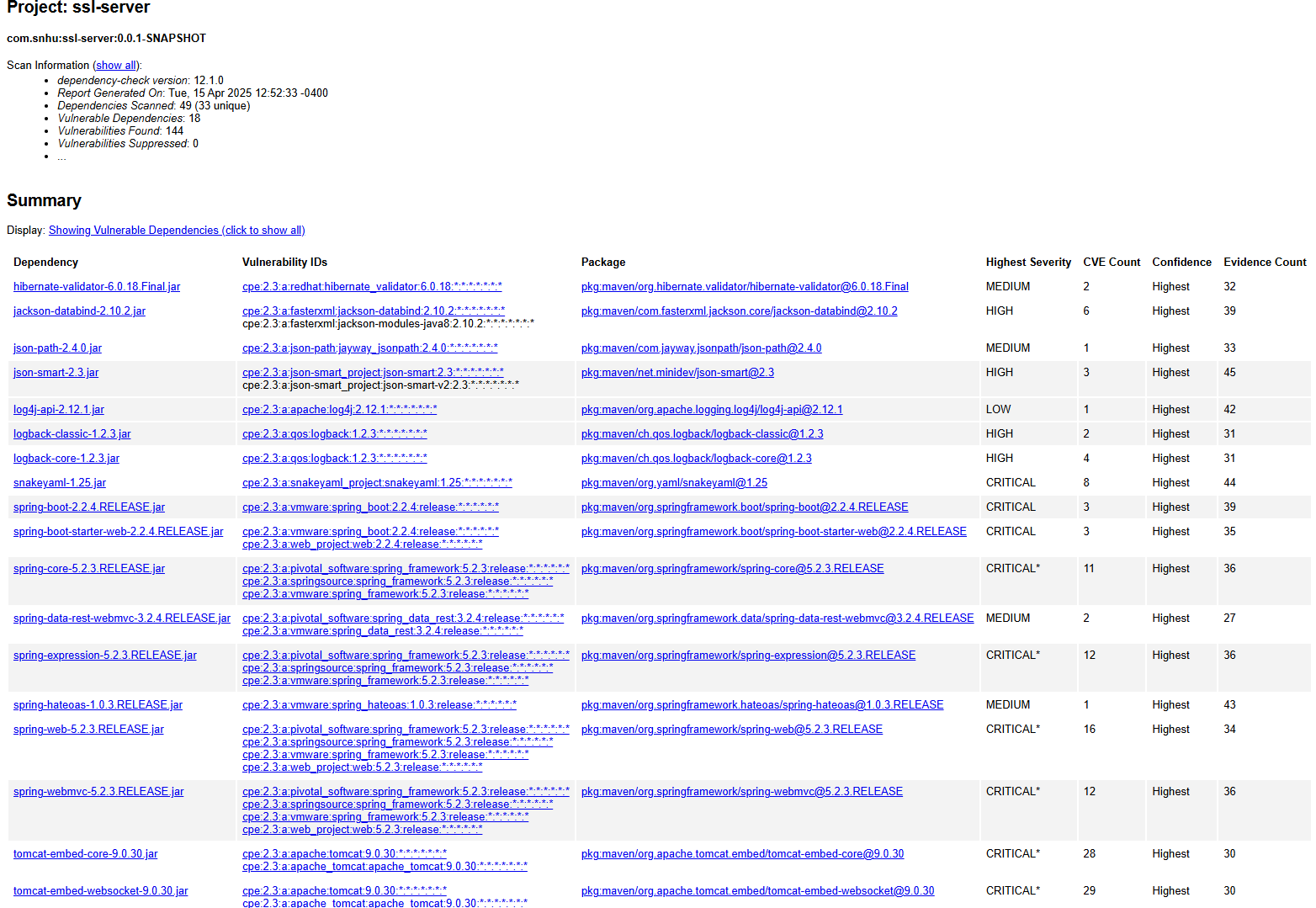
Insert a screenshot below of the web browser that shows a secure webpage.



## Secondary Testing

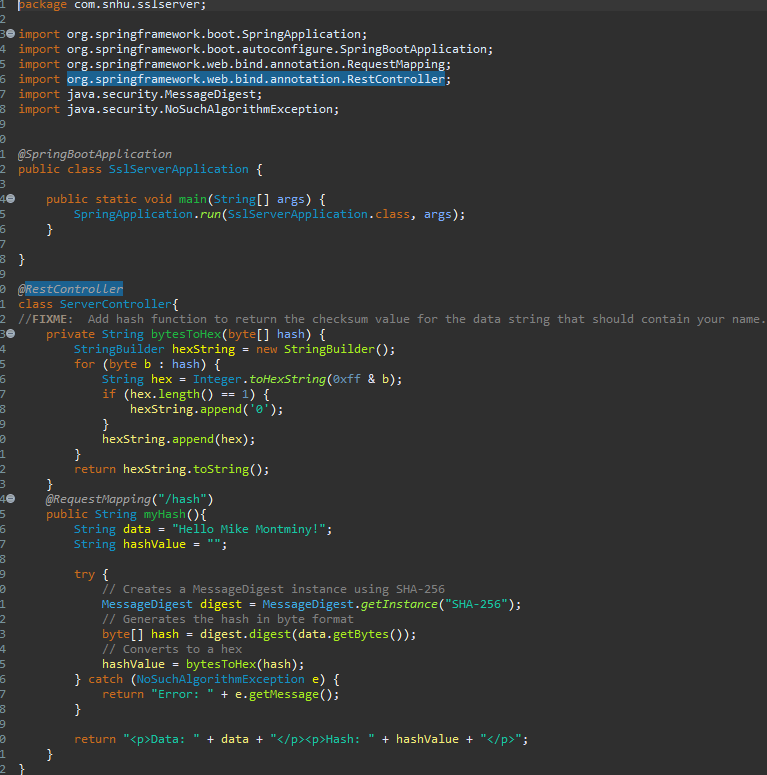
Insert screenshots below of the refactored code executed without errors and the dependency-check report.





## Functional Testing

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



Implemented a RestController providing endpoint generation for the hashing checksum. ServerController applies several secure code principles while properly handling the entire hashing process. SHA256 was chosen due to the vast number of combinations for a key, how established it is, and the extremely low risk for any key collisions. To keep security vulnerabilities down the dependencies should be checked at a minimum of once a month, preferably at the beginning of the month.

## Summary

The focus on this project was to refactor the system for Artemis Financial with the goal of implementing secure data handling practices. The original files did not have any form of hashing and had little to no protection for secure communications. In relation to the flow diagram, a hashing function (SHA-256) was implemented as a form of cryptography. Clean, readable, and easily adjustable code was added to the document for future use if any pieces have to be adjusted down the line. The RESTful checksum generation helps ensure safe and secure communications between a client and a banking server. Many of the methods are kept private to ensure that no outside access or modification occurs from external classes.

To add security layers, the code was refactored with the RESTful endpoint using SHA256 to create the checksum. Proper exception handling is provided to prevent exposing any critical system details to a malicious outside source. Secure coding standards such as making sure specific methods are kept private will help to minimize external modifications by classes. I would also recommend at minimum running monthly dependency checks and updating any plugins as soon as able to the newest and safest version.

## Industry Standard Best Practices

Practices:

* Use of proven algorithm such as SHA256 due to wide adoption and approval by regulators
* Internal logic has minimal outside exposure as the endpoint is exposed through the request map
* Structure for inputs is designed to be able to expand easily and securely to handle dynamic inputs using proper validation and sanitation techniques down the line

Values:

* Integrity and trust gets increased and helps to build up a long-term relationship with our current and future clients
* Secure coding helps to ensure optimal compliance with any regulation standards that must be considered such as PCIDSS and GDPR
* Secure coding assists with minimizing any potential for a security breach and reducing the cost of damage from a cyberattack of any form
* Secure and modular code is much easier to audit, test, refactor, and expand which will cut down on development time, developer stress, and overall project cost