****

# 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** | **4-19-2025** | **Daniel Jones** |  |

## Client



## Developer

Daniel Jones

## Algorithm Cipher

## 

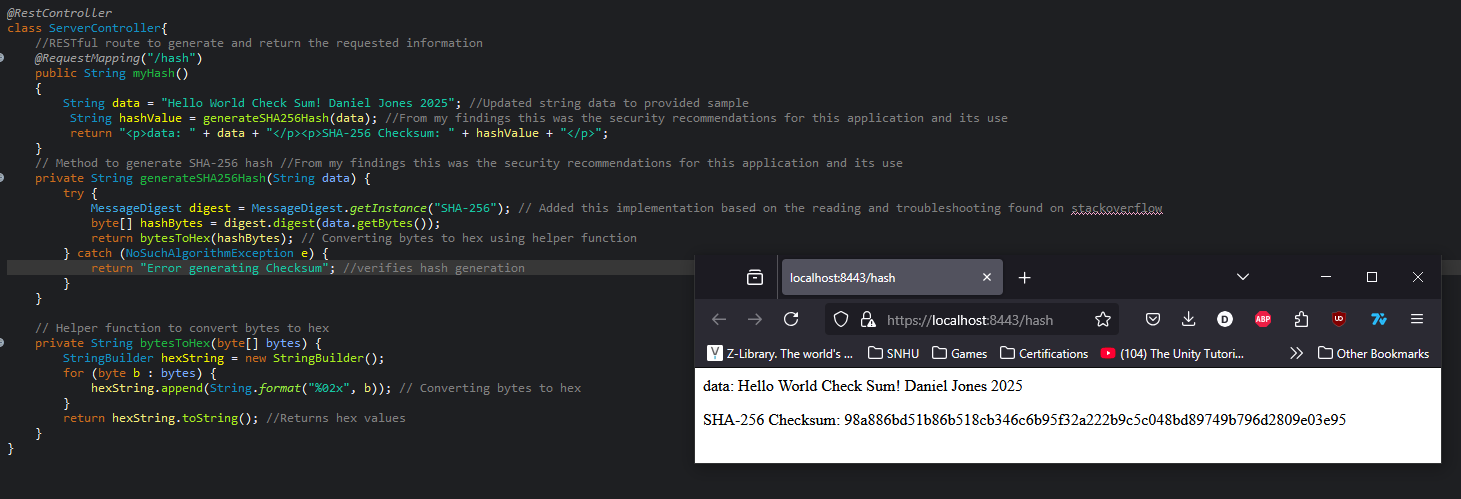
## For securing Artemis Financials’ software application, I recommend using AES (Advanced Encryption Standard). AES is a type of symmetric encryption, which means it uses the same key to both encrypt and decrypt data. This makes it fast and efficient while still being very secure (IBM, n.d.). AES works with 128-bit blocks and supports keys that are 128, 192, or 256 bits long, which helps protect data against brute-force attacks (NIST, 2001). It also uses random numbers to create strong, unpredictable keys, making it harder for hackers to guess them (Cloudflare, n.d.). AES is different from asymmetric encryption (like RSA), which uses a public and private key pair. While asymmetric encryption is great for secure key exchange, AES is better for encrypting copious amounts of data quickly (OWASP, n.d.). AES became the standard in 2001 when it replaced the older DES algorithm, which had become too easy to break. Today, AES is trusted by governments, banks, and tech companies around the world for keeping information safe (NIST, 2001).

## Certificate Generation

## Insert a screenshot below of the CER file.

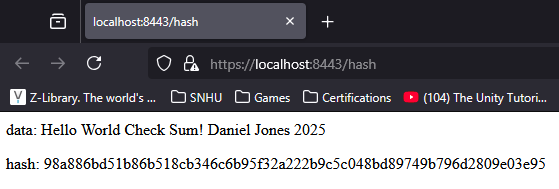
## 

## Deploy Cipher

Insert a screenshot below of the checksum verification.

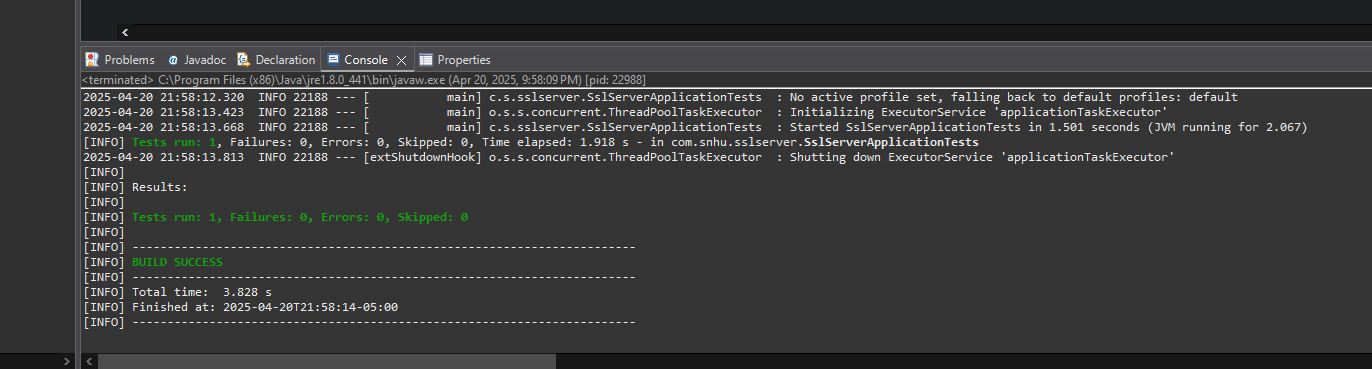
## Secure Communications

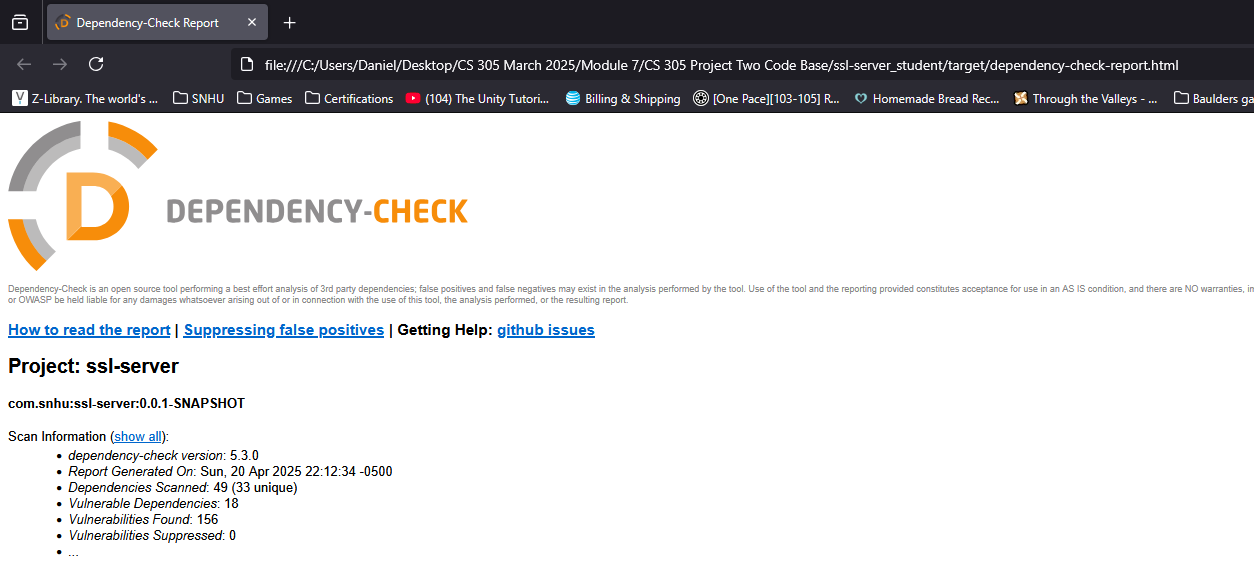
Insert a screenshot below of the web browser that shows a secure webpage.



## Secondary Testing

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

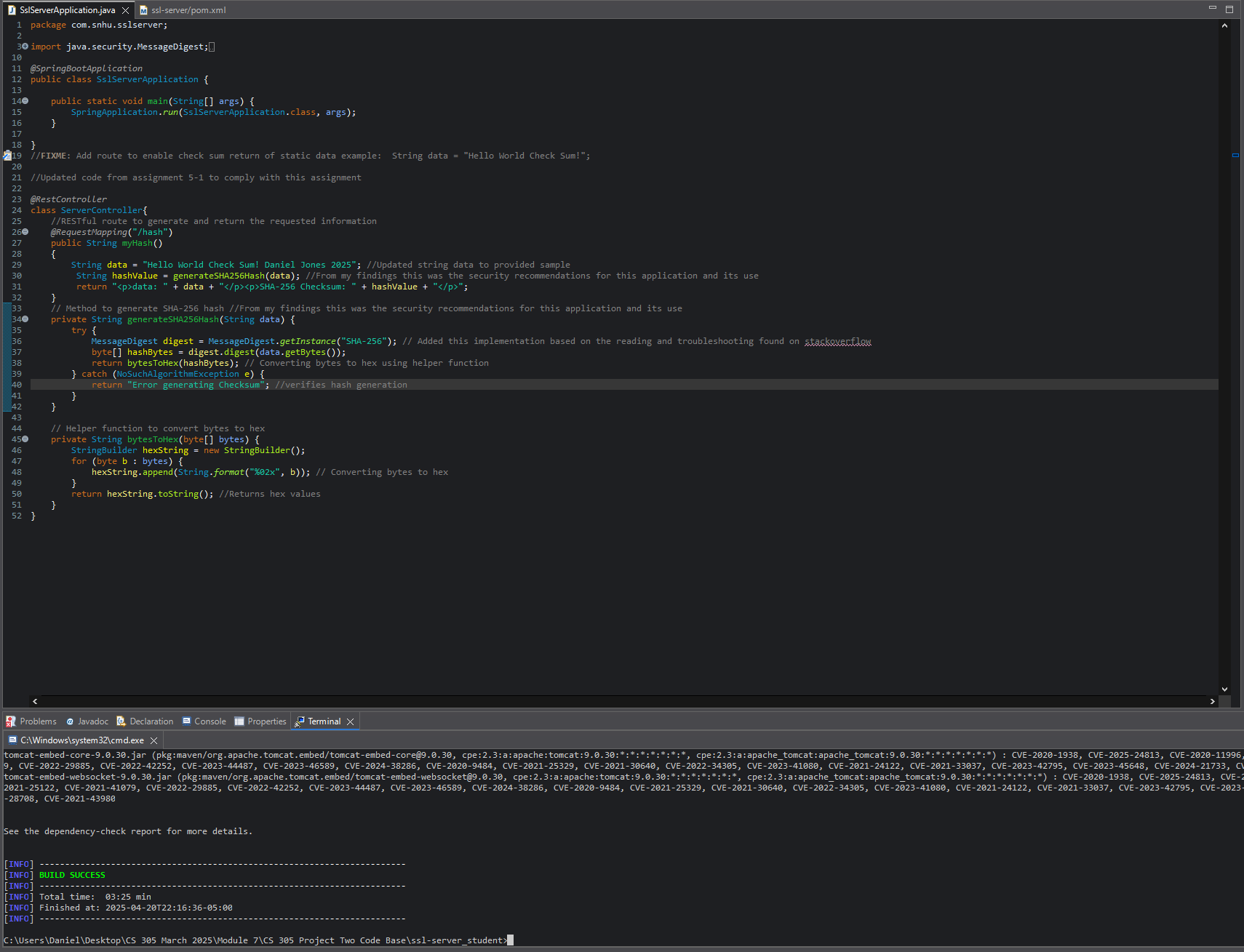




## 

## Functional Testing

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



## Summary:

To improve the security of the software, the code was refactored by adding a REST controller (Server Controller) that introduces a new route (/hash) to return a SHA-256 checksum for a static string. This checksum ensures data integrity by generating a unique hash using Java’s built-in Message Digest class. The refactored code transforms the application from a simple shell to a secure and functional system that includes cryptographic operations. By using SHA-256, a widely recognized cryptographic hash function, the application can now detect unauthorized changes to data. Additionally, using Java’s native cryptographic tools reduces the risk of vulnerabilities from third-party libraries (Menezes et al., 1996).

To further enhance security, the application follows the principle of least privilege, limiting its use to only necessary libraries and functionalities. The SHA-256 algorithm ensures the integrity of the data by producing a fixed-size, secure hash. By avoiding external libraries and relying solely on Java’s built-in Message Digest, the application minimizes the risk of introducing security flaws from untrusted third-party dependencies. These changes implement essential layers of security, ensuring the application is robust against common attacks and compliant with modern security standards (Kaufman et al., 2002).

## Industry Standard Best Practices: .

To keep the software secure, I followed the best industry-standard practices. One of the key practices was using SHA-256, a strong and trusted cryptographic algorithm, instead of older, less secure algorithms like MD5 or SHA-1 (National Institute of Standards and Technology [NIST], 2021). This is in line with recommendations from experts on how to keep data safe. I also reduced the risk of attacks by limiting the application's entry points, exposing only the /hash route for access. By doing this, fewer parts of the system are open to potential attackers, making it harder for them to find vulnerabilities. Additionally, I made sure to manage errors properly (like cryptographic errors), which prevents the system from crashing or accidentally revealing sensitive information (Menezes, van Oorschot, & Vanstone, 1996). Lastly, I kept the code well-organized by separating the different tasks into clear methods, making it easier to manage, evaluate, and fix any problems.

Applying these security practices has clear benefits for the company. First, using SHA-256 ensures the integrity of data, which builds trust with users and stakeholders because they can rely on the system's output. Second, following secure coding practices helps prevent common attacks, such as tampering or spoofing, which could lead to expensive downtimes or damage to the company's reputation. By addressing security issues early in development, the company also avoids costly fixes later; studies show that fixing security flaws after release is much more expensive (Kaufman, Perlman, & Spafford, 2002). Moreover, using these industry standards ensures the software follows important regulations like HIPAA or PCI-DSS, helping avoid legal problems. Lastly, keeping a clean and secure code makes it easier for future developers to understand and update the system without introducing new risks.

## References and Citations:

## National Institute of Standards and Technology. (2001). *Announcing the Advanced Encryption Standard (AES)* (FIPS PUB 197). <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>

## National Institute of Standards and Technology (NIST). (2021). *Secure Hash Standard (SHS)*. Retrieved from <https://csrc.nist.gov/publications/detail/sp/800-107/rev-1/final>

## “Cryptographic Storage Cheat Sheet¶.” *Cryptographic Storage - OWASP Cheat Sheet Series*, cheatsheetseries.owasp.org/cheatsheets/Cryptographic\_Storage\_Cheat\_Sheet.html. Accessed 20 Apr. 2025.

## IBM. (n.d.). *What is AES encryption?* IBM Knowledge Center. <https://www.ibm.com/topics/aes-encryption>

## Cloudflare. (n.d.). *How does encryption work?* <https://www.cloudflare.com/learning/ssl/how-does-encryption-work/>

## Kaufman, C., Perlman, R., & Spafford, E. H. (2002). (“Network Security: Private Communication in A Public World 2Nd Ed.”) Prentice Hall.

## Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (1996). *Handbook of Applied Cryptography*. CRC Press.