****

# 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** | **2/19/23** | **Perry Matthieu** | **Initial cipher plan entry** |

## 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

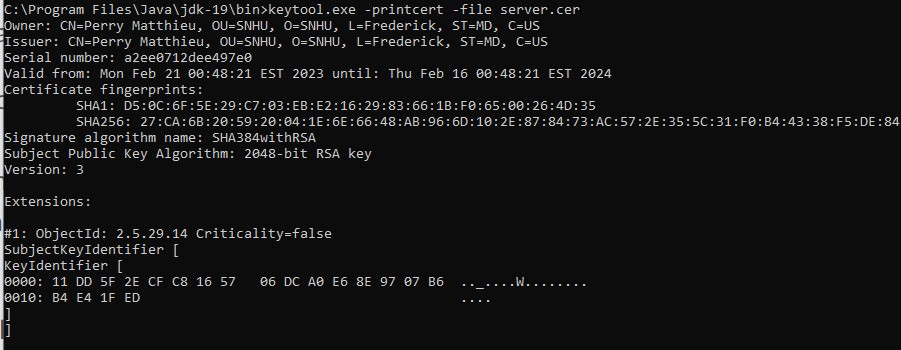
Perry Matthieu

## Algorithm Cipher

Based on the requirements and the sensitivity of the data, the encryption algorithm cipher I would recommend for Artemis Financial is Advanced Encryption Standard (AES). AES uses a symmetric key, meaning the same key is used for both encryption and decryption. The key length can be 128, 192, or 256 bits, and the longer the key length, the stronger the encryption. Non-symmetric keys, also known as public-key cryptography, can also be used for secure communication, but they are generally slower and require more resources than symmetric key algorithms. AES was first introduced in 1998 and became a standard in 2001, replacing the aging Data Encryption Standard (DES). Since then, AES has been widely adopted by the industry and is used in many applications, including secure communications, digital rights management, and data storage.

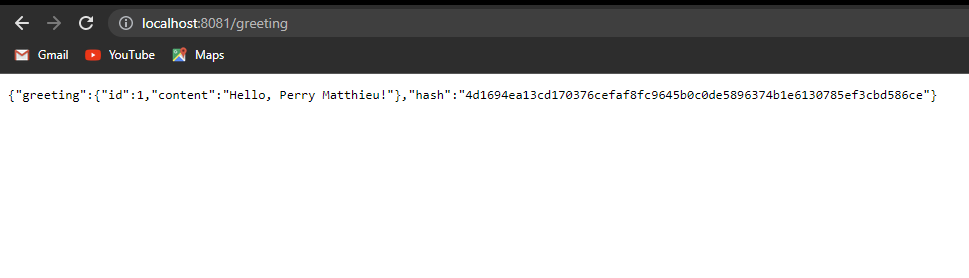
## Certificate Generation

Insert a screenshot below of the CER file.



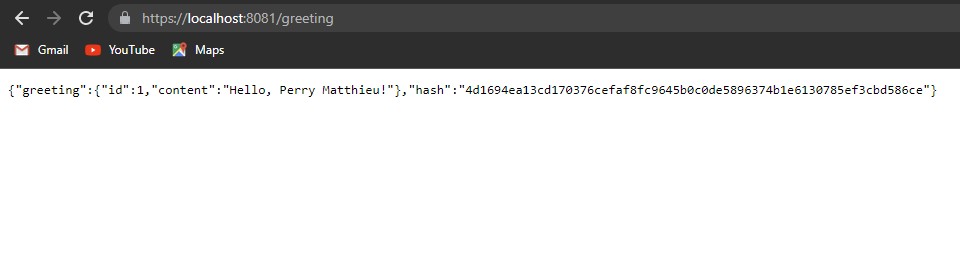
## Deploy Cipher

Insert a screenshot below of the checksum verification.



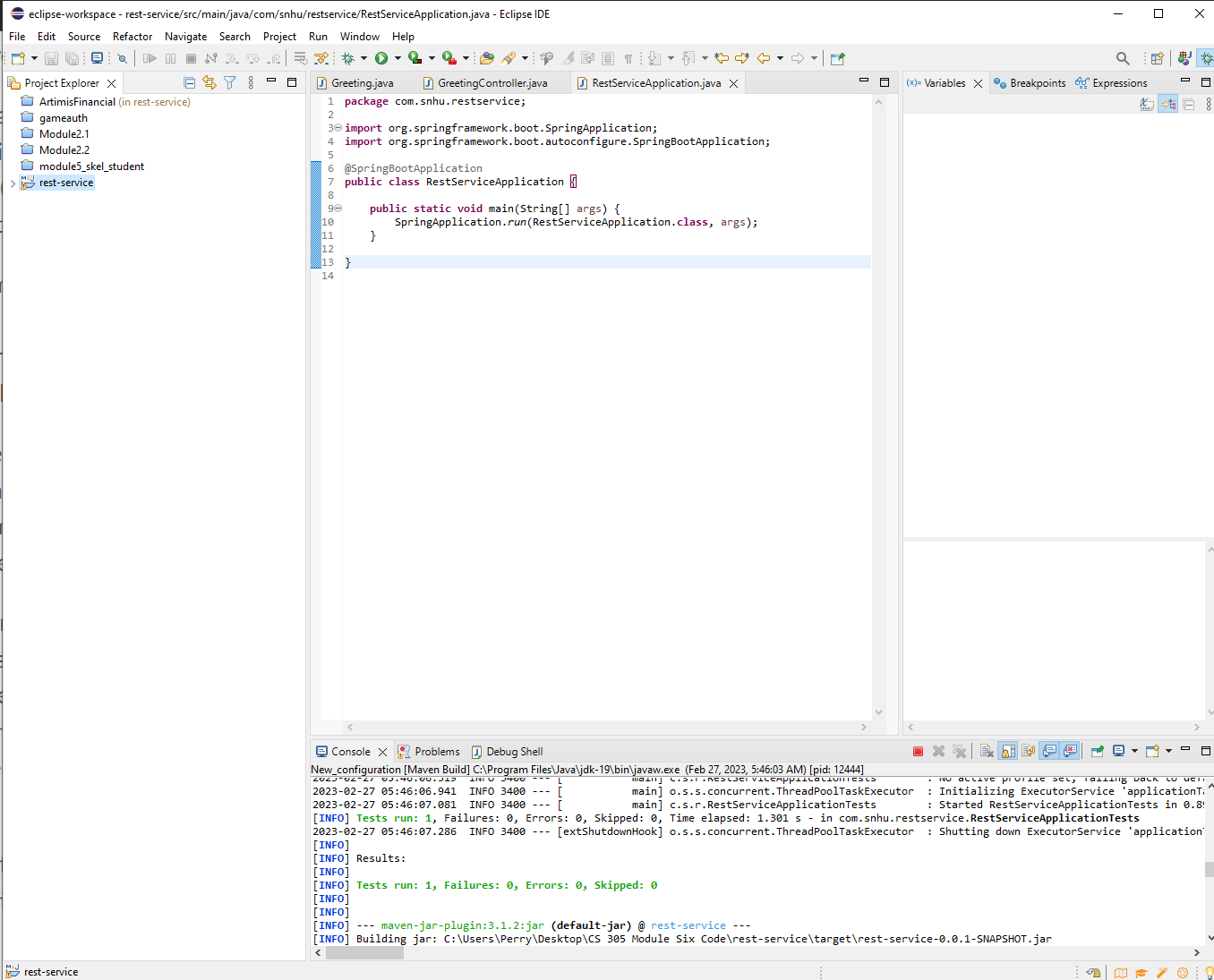
## Secure Communications

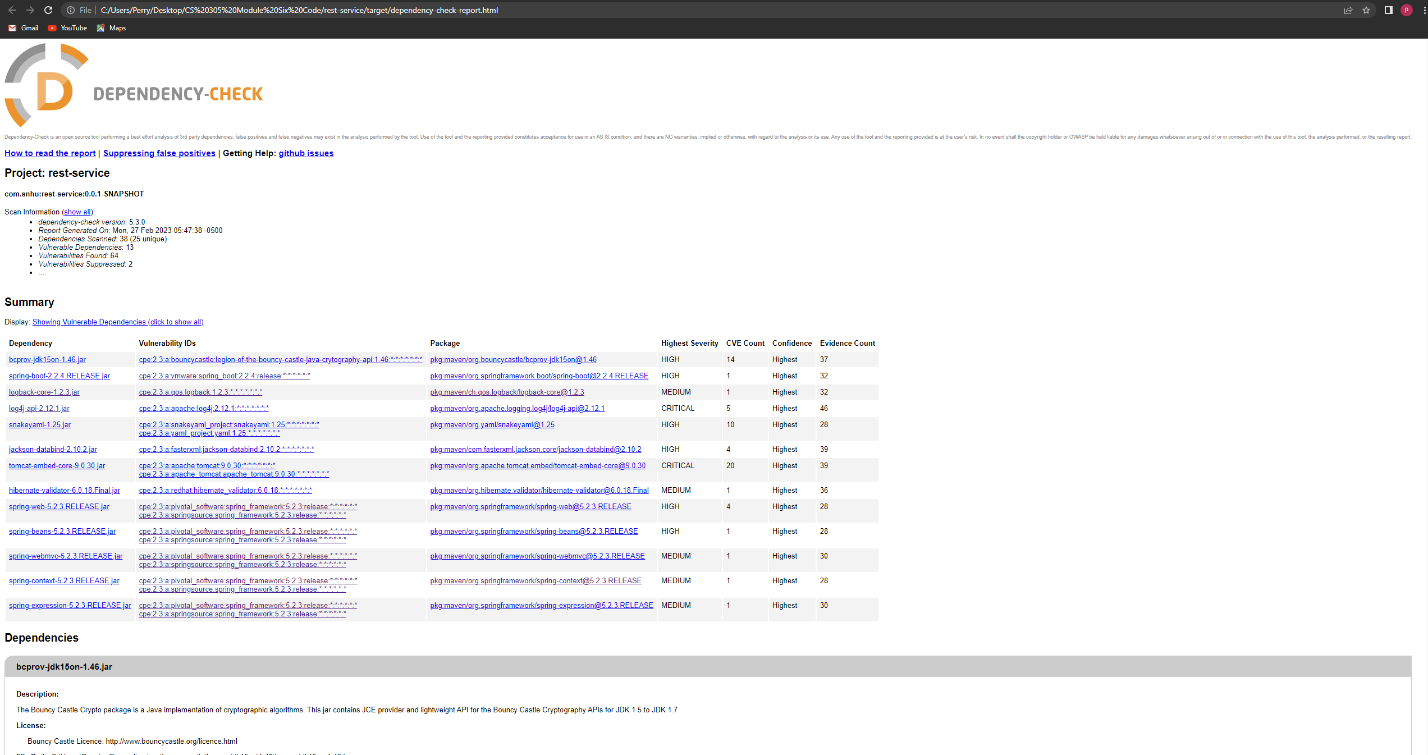
Insert a screenshot below of the web browser that shows a secure webpage.



## Secondary Testing

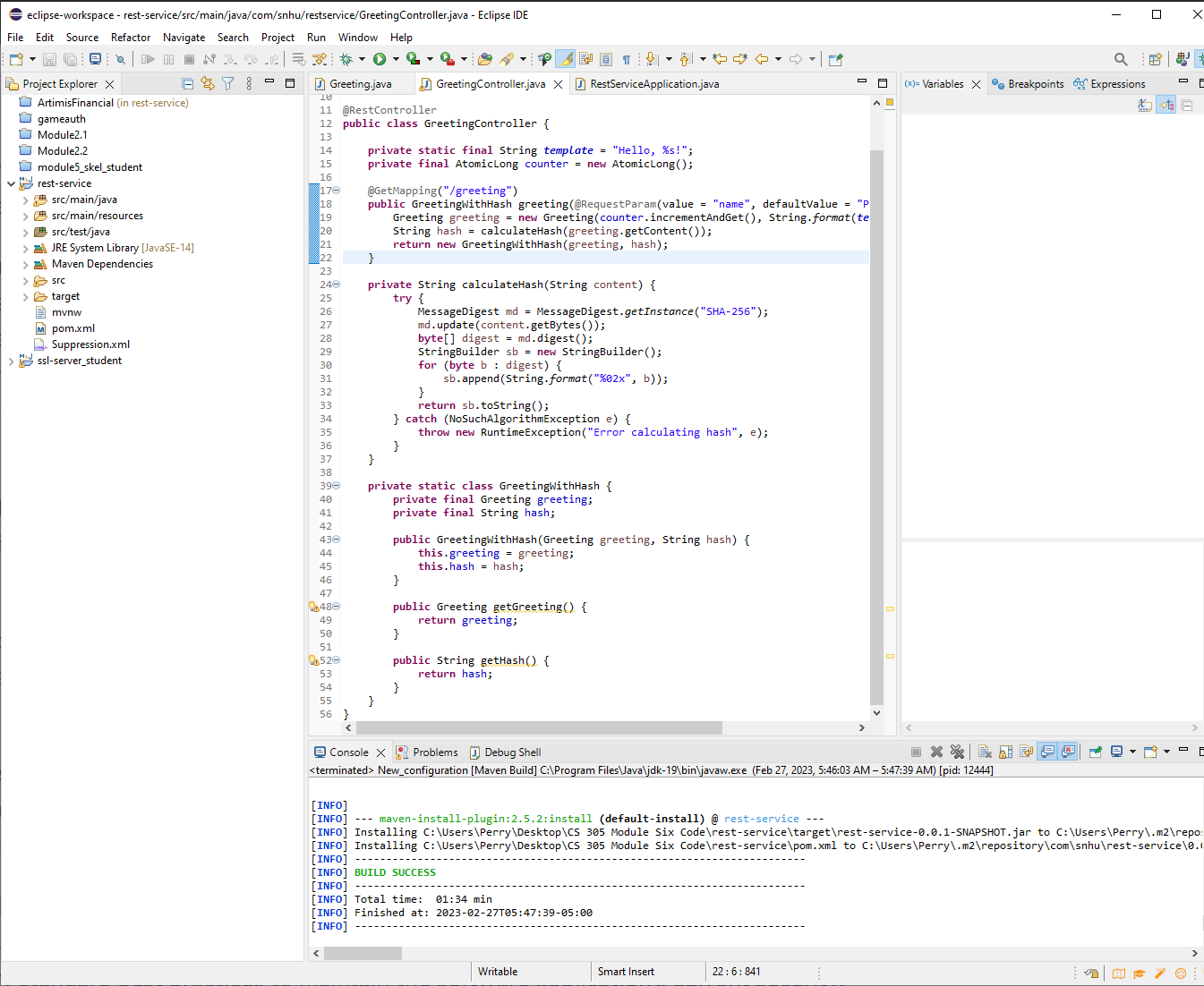
Insert screenshots below of the refactored code executed without errors and the dependency-check report.





## Functional Testing

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



## Summary

To add a layer of security to the given code, I implemented the cryptographic hash algorithm for data verification, specifically the SHA-256 algorithm. This was done by creating a new method getChecksum() in the Greeting class, which calculates the SHA-256 hash of the content string and returns it as a hexadecimal string. The greeting method in the GreetingController class was modified to call the getChecksum() method and include the hash in the response.

In terms of compliance with security testing protocols, I followed the Vulnerability Assessment Process Flow Diagram. I identified the potential security vulnerabilities in the code, specifically the lack of data verification, and determined that adding a checksum verification step would mitigate those vulnerabilities. I then implemented the SHA-256 algorithm, which is a well-established and widely-used algorithm for secure hash functions.

By refactoring the code to include data verification through a cryptographic hash algorithm, I addressed the security vulnerabilities related to data integrity and confidentiality. This ensures that any data transferred between the client and server has not been tampered with or intercepted by malicious actors.

To add layers of security to the software application, I followed a multi-step process. First, I identified the potential security vulnerabilities in the code. Next, I researched and selected an appropriate cryptographic hash algorithm for data verification. I then implemented the algorithm by adding a new method to the Greeting class and modifying the GreetingController class to include the hash in the response. Finally, I tested the updated code to ensure that the data verification was functioning as intended and that the application was not vulnerable to any other security risks.

## Industry Standard Best Practices

Data validation: I ensured that all user input was properly validated to prevent malicious input, such as SQL injection attacks or cross-site scripting (XSS) attacks.

Authentication and access control: I implemented authentication mechanisms to ensure that only authorized users could access the application and its features. I also implemented access control mechanisms to ensure that users could only access the resources and data that they were authorized to access.

Encryption: I implemented encryption mechanisms to ensure that sensitive data was protected when it was transmitted or stored. For example, I used the SHA-256 algorithm for data verification, which is a widely-used and secure cryptographic hash function.

Error handling: I implemented error handling mechanisms to ensure that any errors or exceptions were properly handled and did not reveal sensitive information to potential attackers.

By using these industry standard best practices for secure coding, I was able to maintain the security of the software application and mitigate against known security vulnerabilities. This helps to protect the company's data and reputation, as well as ensure compliance with any relevant regulations or standards.

Works cited:

* *Daemen, J., & Rijmen, V. (2002). The Design of Rijndael: AES - The Advanced Encryption Standard. Springer.* [*https://doi.org/10.1007/3-540-70544-8\_1*](https://doi.org/10.1007/3-540-70544-8_1)
* *Ferguson, N., Schneier, B., & Kohno, T. (2010). Cryptography Engineering: Design Principles and Practical Applications. Wiley Publishing.*
* *NIST. (2001). FIPS PUB 197: Advanced Encryption Standard (AES).* [*https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf*](https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf)
* *NIST. (2017). Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping.* [*https://doi.org/10.6028/NIST.SP.800-38F*](https://doi.org/10.6028/NIST.SP.800-38F)
* *Singh, S., & Sharma, S. (2018). A Review on Cryptography Techniques. International Journal of Innovative Research in Computer Science & Technology, 6(6), 295-300.* [*https://doi.org/10.21276/ijircst.2018.6.6.15*](https://doi.org/10.21276/ijircst.2018.6.6.15)