****

# 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** | **10/18/2024** | **Tyler Doupe** |  |

## 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

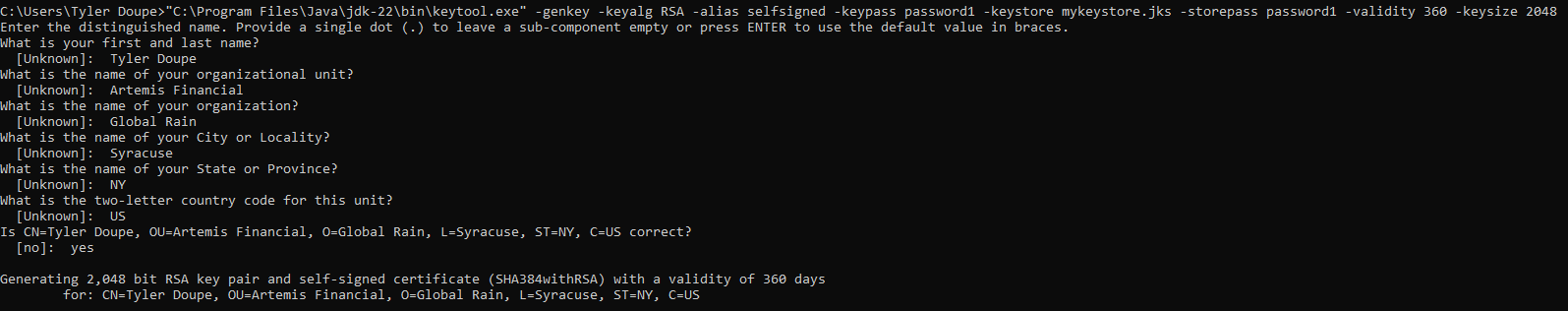
Tyler Doupe

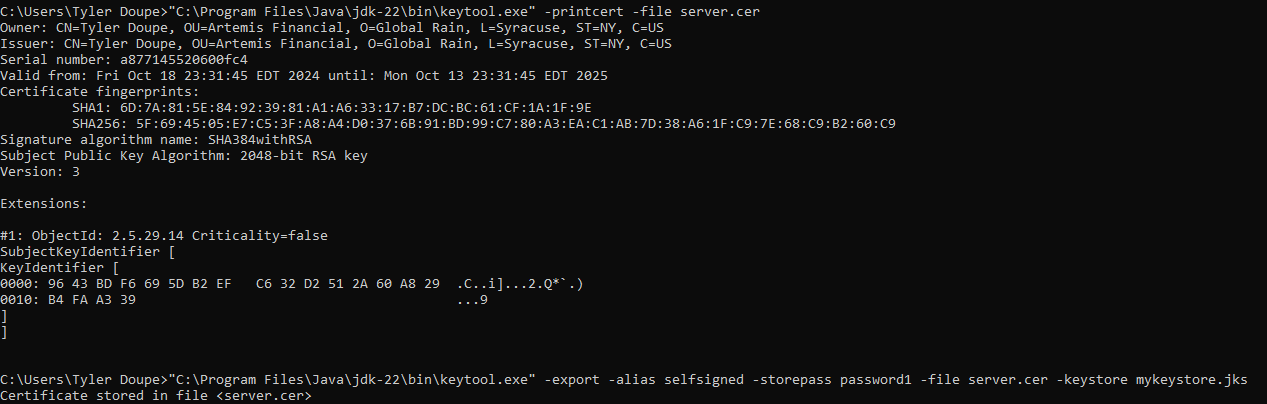
## Algorithm Cipher

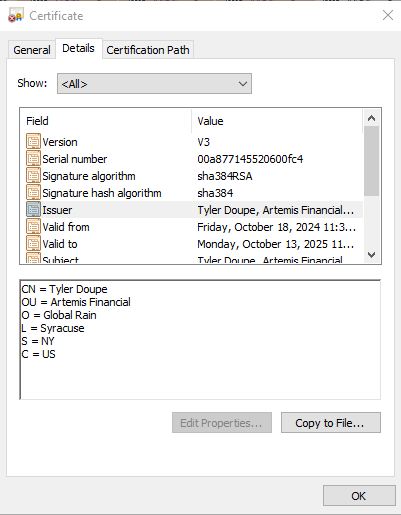
From my personal experience and knowledge gained through this course, I would recommend deploying Advanced Encryption Standard, AES, as an appropriate encryption algorithm cipher. AES was developed by the national Institute of Standards and Technology, NIST, to provide strong security protection through various key lengths of 128, 192, or 256 bits. AES is efficient and widely implemented in securing sensitive data, internet communication and encrypting files. As mentioned, AES uses 128, 192, or 256-bit key lengths with a greater key length increasing the complexity and security of the encryption. AES alone does not use a hash function but it can be combined with a secure hash algorithm such as SHA-256 to provide file integrity by providing a digest of the data that ensures the file is authentic. In AES, random numbers are crucial for generating secure encryption keys. To avoid predictable patterns, a cryptographically secure random number generator, CSPRNG, should be used, ensuring that the keys are random and resistant to attacks. Symmetric keys are keys that can be used for both encryption and decryption which allows them to be more efficient when encrypting large amounts of data. Asymmetric keys consist of two keys, a public and private key which allows them to have an added layer of security. AES makes use of random numbers for key generation and uses symmetric keys. AES supersedes the Data Encryption Standard that was published in 1977 with its offer of larger key lengths and a more secure encryption algorithm. Today, AES is a widely used algorithm cipher and is resistant to most cryptographic attacks.

## Certificate Generation

Insert a screenshot below of the CER file.

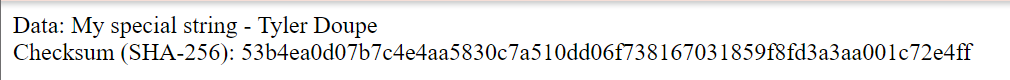






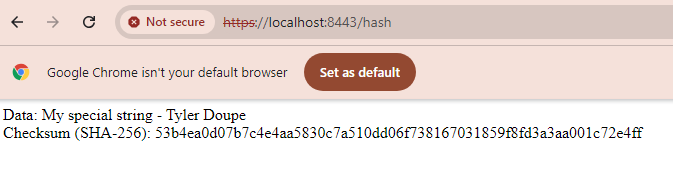
## Deploy Cipher

Insert a screenshot below of the checksum verification.



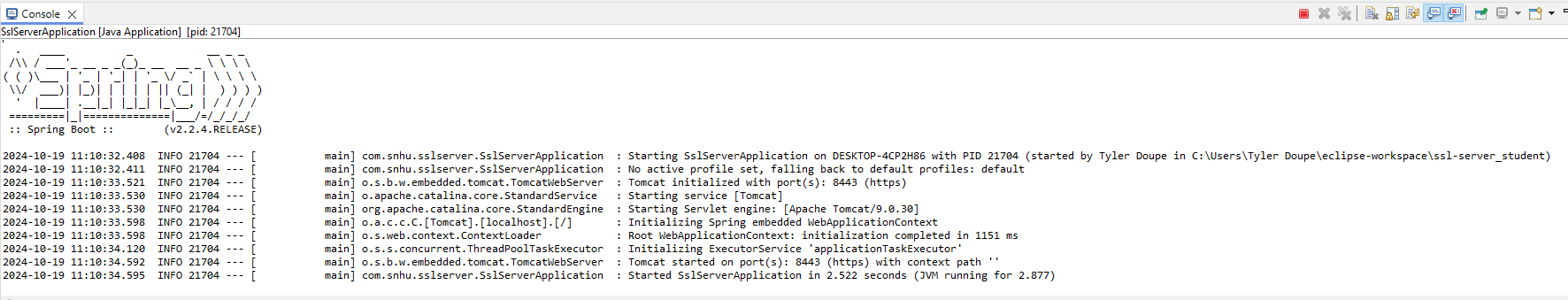
## Secure Communications

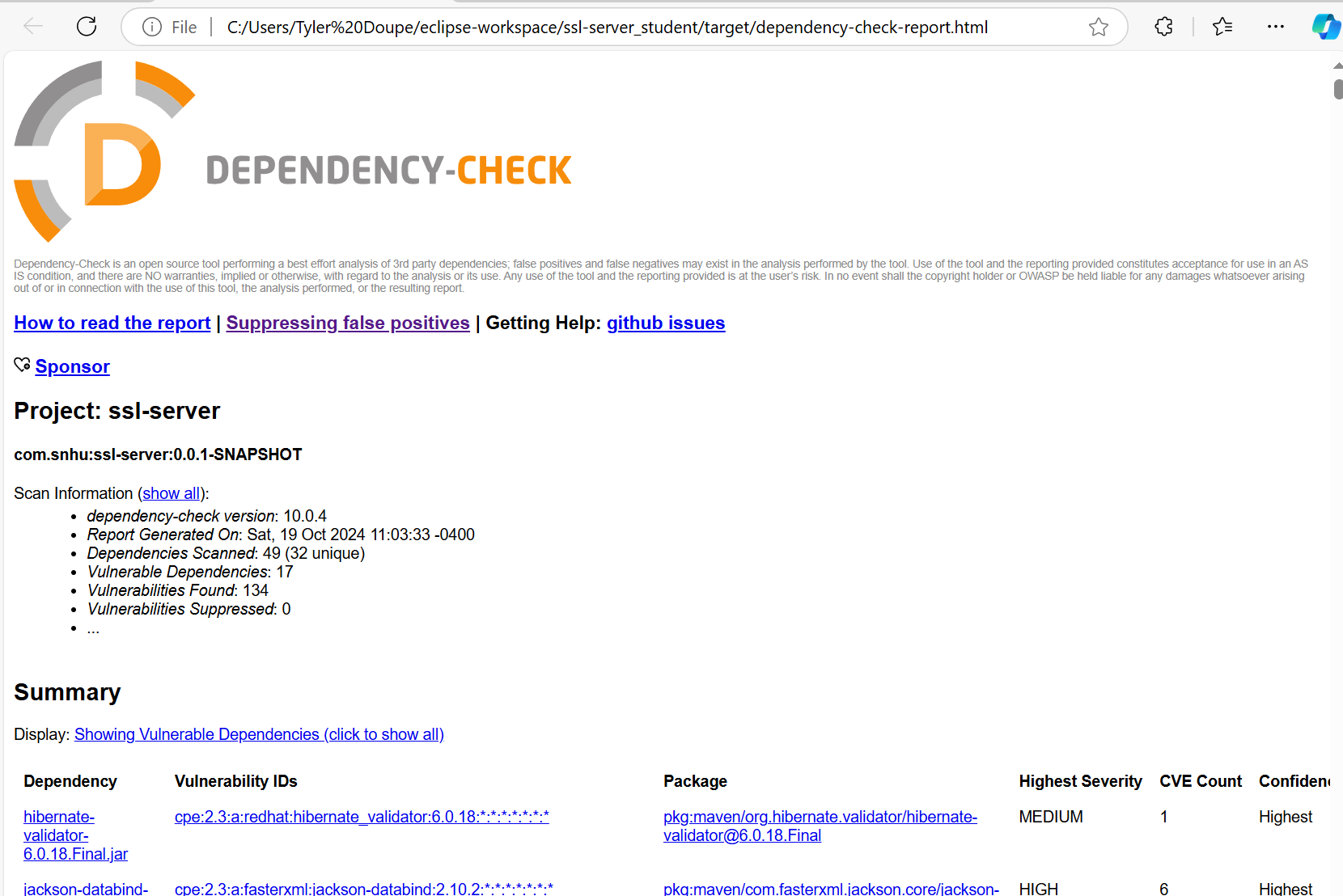
Insert a screenshot below of the web browser that shows a secure webpage.



## Secondary Testing

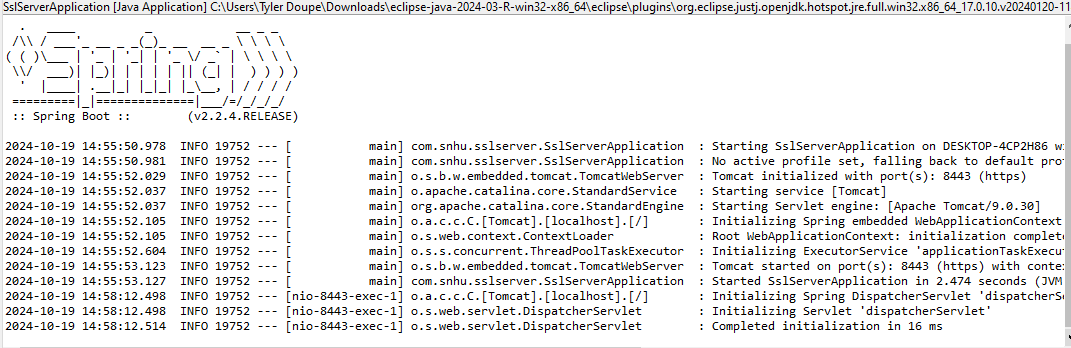
Insert screenshots below of the refactored code executed without errors and the dependency-check report.





## Functional Testing

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



## Summary

The code for Artemis Financial has been refactored to comply with several security testing protocols outlined in the vulnerability assessment process flow diagram. For starters, input validation was implemented in the refactored code with the implementation of error handling. Any exceptions such as NoSuchAlgorithmException are handled securely which provides secure handling of sensitive information. This is essential for Artemis Financial’s application as it will be handling and communicating a large amount of sensitive information between the clients and server. Cryptography was addressed in the refactored code with the implementation of SHA-256 as a cryptographic hash algorithm by generating a checksum for data integrity verification. In addition to implementing SHA-256, the application was refactored to execute via HTTPS with the use of a self-signed SSL certificate. Although using a self-signed SSL certificate is not practical for public use, for the purposes of this project, it allows us to test creating an application that would operate via HTTPS and thus represent creating an application that can communicate securely between a client and server. Code quality was addressed in several manners when refactoring the application. It was first addressed through the use of secondary testing when the code was refactored to run a secondary static test using the dependency-check tool. This tool identified and assessed vulnerabilities resulting from the third-party dependencies associated with the application. This allowed me to then manually review these vulnerabilities and determine if they are in fact vulnerabilities or if they are false positives. The refactored code was also manually reviewed for syntactical, logical and security vulnerabilities as part of addressing the code quality vulnerability. When discussing layers of security for the software application, several layers of security were implemented through the refactored code. With the use of HTTPS and SSL, the application ensures secure communication between the clients and server through encryption which ensures the data being transmitted is not intercepted. The use of SHA-256 cryptography ensures that the data is authentic and maintains its integrity. The refactored code also implements the dependency-check tool which can be used to monitor for third-party dependencies vulnerabilities. This allows the application to assess vulnerabilities in an efficient manner and allow for known exploits to be caught earlier so they can be updated.

## Industry Standard Best Practices

When refactoring the code for Artemis Financial, I applied industry best practices for secure coding to mitigate known security vulnerabilities in several different ways. First, through the use of SHA-256 cryptography, I implemented a cryptographic hash function that is recognized across the industry as a strong cryptographic algorithm that provides security and data integrity. The use of SHA-256 is a stronger algorithm than some older algorithms such as MD5, is less susceptible to collision, and is efficient, making it an optimal cryptographic hash function for Artemis Financial. I also implemented HTTPS communication between the clients and server which ensures encrypted communication through SSL communication and the use of a self-signed certificate. Although the use of a self-signed SSL certificate is not practical for an organization’s use, it allowed for me to demonstrate the use of this type of communication with the application for development purposes. In addition, I implemented the dependency-check tool which allows the application to assess third-party dependencies vulnerabilities in an efficient manner and ultimately allow us to identify dependencies to potentially be updated sooner. This provides continuous monitoring of the third-party dependencies and thus allows the application to stay more up to date from vulnerabilities. From a company’s overall well-being, applying industry standard best practices is vital to their success! Artemis Financial will be handling sensitive information to clients worldwide and breaches of this information could be catastrophic to their success as a company. Malicious actors are always on the prowl for means to exploit a vulnerability within an application and gather sensitive information. Following a breach of sensitive information, questions begin to spiral about the overall security of the information a company possess on their clients and it can lead to many clients losing trust and faith with the company. By applying industry standard best practices, not only can we prevent data breaches from occurring, but we can build and/or maintain public trust and even save money in the long term by avoiding law suits and by future proofing the application for what is the most up to date standard.

**References:**

GeeksforGeeks. Advanced Encryption Standard (AES). GeeksforGeeks. https://www.geeksforgeeks.org/advanced-encryption-standard-aes/

National Institute of Standards and Technology. (2001). Advanced Encryption Standard (AES) (FIPS Publication No. 197). U.S. Department of Commerce. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf

Oracle. (n.d.). Cipher algorithm names. Oracle. https://docs.oracle.com/javase/9/docs/specs/security/standard-names.html#cipher-algorithm-names

Rouse, M. (n.d.). Advanced Encryption Standard (AES). TechTarget. https://www.techtarget.com/searchsecurity/definition/Advanced-Encryption-Standard

Wikipedia contributors. (2024, October 17). Advanced Encryption Standard. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Advanced\_Encryption\_Standard