****

# 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** | **08/03/23** | **William Cocomise** | **Algorithm Cipher** |
| **1.1** | **08/05/23** | **William Cocomise** | **Certificate Generation** |
| **1.2** | **08/14/23** | **William Cocomise** | **Deploy Cipher, Secure Communications, Secondary Testing** |
| **1.3** | **08/15/23** | **William Cocomise** | **Functional Testing, Summary, Industry Standard Best Practices** |

## 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

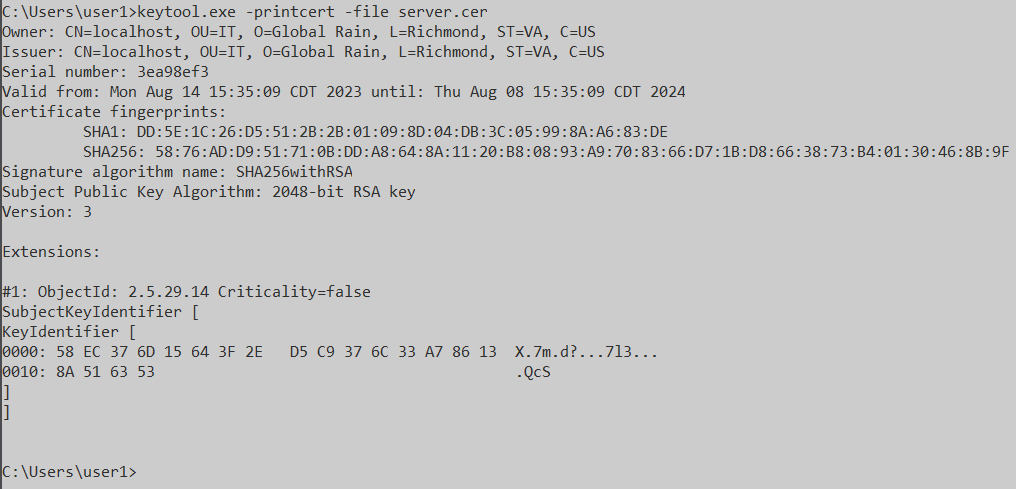
William Cocomise

## Algorithm Cipher

The recommendation we are making to Artemis is to encrypt their web traffic using a cipher suite. This suite includes the TLS protocol which is an industry standard. TLS uses a certificate signing architecture for developing trust relationships between client and server. Part of the certificate includes a private key which should be generated using RSA to have the most currently secure private key. AES is recommended for the symmetric encryption of web payloads. AES is also recommended by the National Institute of Standards and Technology (NIST). It is a block cipher which accounts for improved encryption times. AES also supports a variety of key sizes. The largest key supported is 256 bits. But AES-128 is considered secure and is less resource heavy. So it is recommended for web applications. AES also supports key hashing using SHA-256. At this time it is considered near impossible that a collision would occur using SHA-256. Because the web application is sending data between systems, an asymmetric cryptographic cipher is being recommended. Meaning both private and public keys will be needed for decryption. Using a secure key manager is imperative for this cipher. These recommendations are based on the most current cryptography tools and data available. Cryptography tools are liable to change and have been changing rapidly in recent decades. Because of this, it is important to get updated recommendations as often as possible.

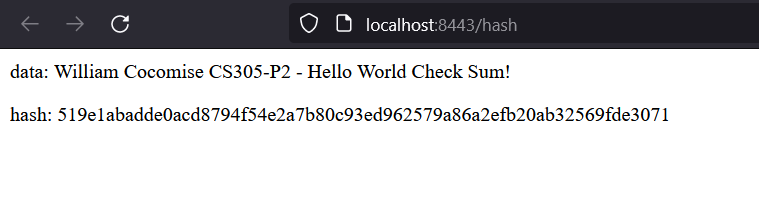
## Certificate Generation

Insert a screenshot below of the CER file.



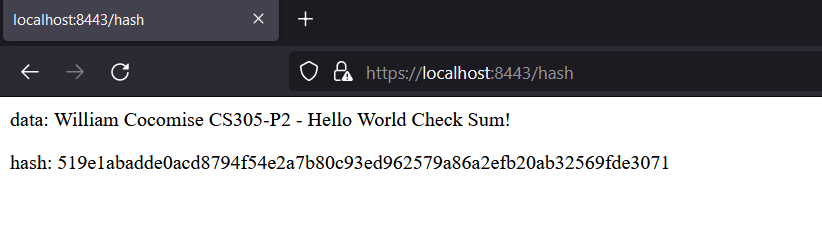
## Deploy Cipher

Insert a screenshot below of the checksum verification.



## Secure Communications

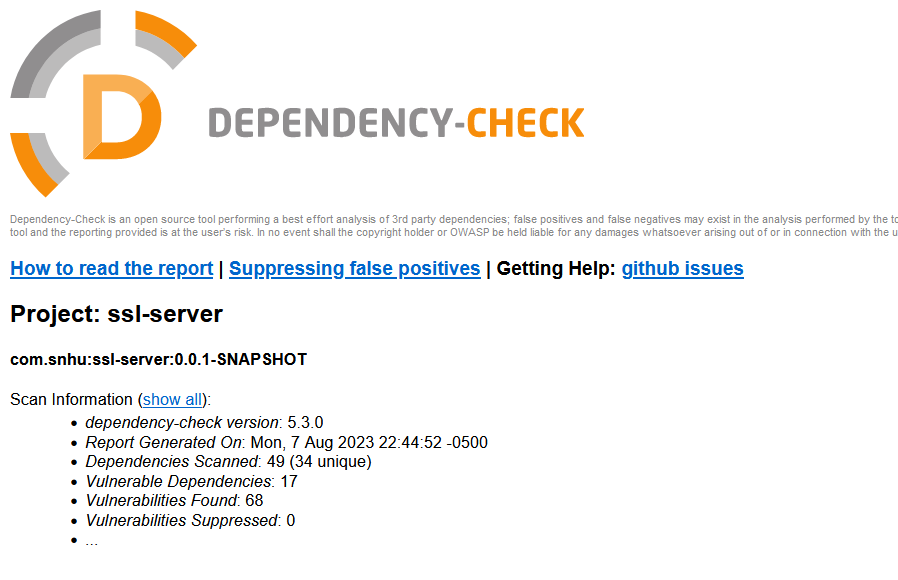
Insert a screenshot below of the web browser that shows a secure webpage.



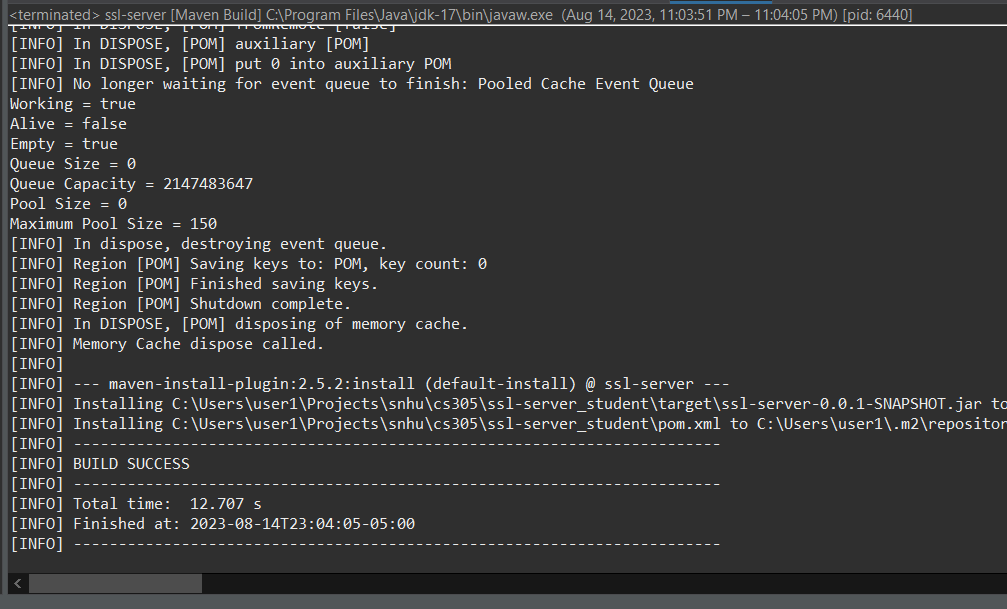
## Secondary Testing

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

Initial Dependency Check Report:



Code executes without errors:

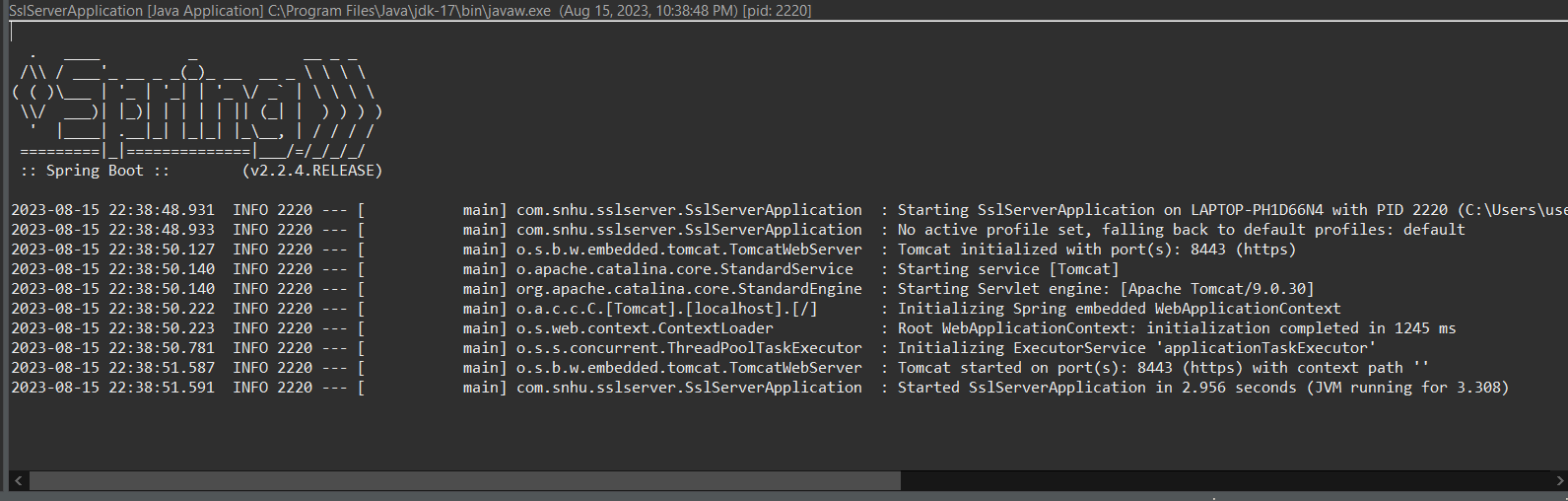


Secondary Dependency Check Report:



## Functional Testing

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



## Summary

Here I will lay out the code changes I made according to the vulnerability assessment process. Views: There were no views to refactor. Models: There were no models to refactor. Controllers: The rest controller was updated to specify the http request type. Data Access: The application cannot compile locally without hardcoded key-store credentials. This can be updated in production environment. Services: Thee were no services to refactor. Plug-Ins: the dependencies were checked for vulnerabilities. APIs: There were no APIs to assess.

## Industry Standard Best Practices

Our application of industry best practices included using the latest encryption technologies, refactoring code to be more secure, and using a vulnerability detection tool to scan dependencies. By applying these strategies of secure coding at the outset, it will save Artemis Financial many resources over time. Using a security first approach means less refactoring of code and more importantly, less incident response in the future.