****

# 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** | **06/13/2023** | **Ryan St George** |  |

## 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

Ryan St George

## Algorithm Cipher

Based on the current application Artemis Financial has I would recommend an Advanced Encryption Standard algorithm. This is a widely recommended and secure encryption algorithm which is commonly used to protect sensitive data. With AES being a symmetric encryption algorithm that uses a block cipher to encrypt and or decrypt data. With fixed sized blocks of data and can have a range of key sizes. Including 128, 192, and 256 bits. Based on the key size mixing, substitution, and other operations take place time and time again.

With AES not actually being a Hash function but it’s a block cipher. Hash Functions commonly are used with encryption algorithms to make things more secure especially in dealing with financial information. For instance, a popular hash function is SHA-256 or SHA-3 which can produce a fixed size output of 256 bits or even more.

Now in these algorithms random numbers play a big part. As they are used for generating encryption keys, initialization vectors, or any other parameter. Having randomness time and time again ensures the encryption is secure to prevent attackers from predicting the key or even the pattern in the encrypted data. While AES uses symmetric encryption where the same key is used for both encryption and decryption. The biggest concern with this is communicating the key to both parties in a secure way.

Overall, the history of encryption algorithms have evolved like any other technology as computers get stronger, security requirements change. For instance, DES prior to AES. To this day AES is still widely used and considered secure. Oftentimes being recommend by agencies and regulators depending on the information needing encryption. Noting as vulnerabilities are found, computers gain more power, things change and evolve.

## Certificate Generation

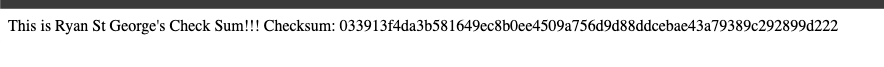
Insert a screenshot below of the CER file.

A screenshot of a computer

Description automatically generated with medium confidence

## 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.

A screenshot of a computer

Description automatically generated

## Secondary Testing

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

A screen shot of a computer

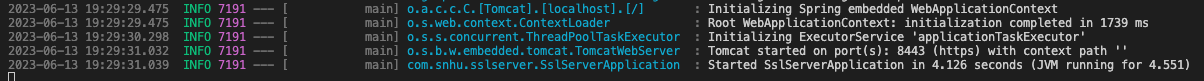
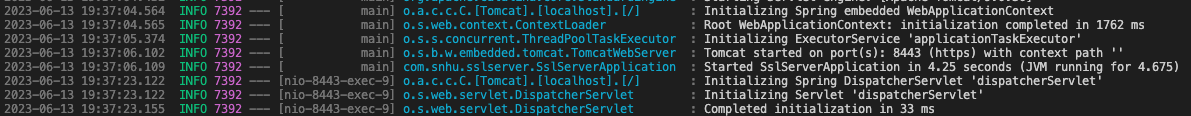
Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with low confidence

## Functional Testing

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



Code refactored to ensure things were up to date and we can see here spring boot loaded up and started with no errors. My application.properties file would not save the current configuration after functional testing I noticed I had two of these which one was located outside the resources file. I removed that file and then each time I ran springboot I no longer needed to adjust the configuration.

## Summary

Throughout this refactor of the code base I was able to analyze and detect current security vulnerabilities. Through checking the code with maven dependency checker, creating a certificate and moving to HTTPS instead of HTTP, and doing functional testing we were able to make the code more secure. With cryptography integrating our checksum verification. Alongside this we were able to use an API as restfulAPI was built using spring framework. Alongside this I also used the client/server with the certificate generation and keystore use as shown in above images and in the codebase itself.

I was able to integrate the hash function to have data integrity for our client and their financial and customer information. And ensuring dependencies were being updated to remove current vulnerabilities thanks to the maven dependency checker. If any further vulnerabilities still exist or arise in the future we can take care of these to limit the risk to Artemis Financial and their customers.

## Industry Standard Best Practices

Overall providing industry best practices and implementing everything above we were able to make the code base more secure and ensure the business and customer are safe from any security vulnerability. From overall code readability for anyone in the future that may need to update or run checks. To moving the web interface to HTTPS instead of HTTP. Ensuring all dependencies were updated and unused code was removed. With this implemented we hope to keep out any bad actors. This will help Artemis Financial stay out of any legal trouble as we are trying to maintain current regulations for the company and their data.

**See refactored code that is attached “below” for any references.**