****

# 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** | **[2/24/2024]** | **Elliot Putnam** |  |

## 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

Elliot Putnam

## Algorithm Cipher

The client Artemis Financial is requesting our services to secure their web application further to prevent the loss of company and customer data. Being a web application, many vulnerabilities may exist, and an analysis of the current system is required to develop a mitigation plan. I recommend using a form of encryption for all communications across the web servers, along with encryption between functions within their software. This will create a secure format in which data can be transmitted with less risk of the intercepted data being interpreted. A SHA-256 cipher algorithm is recommended as it provides excellent encryption. Using a 256-bit key pair ensures the complexity of generating a matching pair and provides a non-reversable checksum to check if it’s valid.

## Certificate Generation

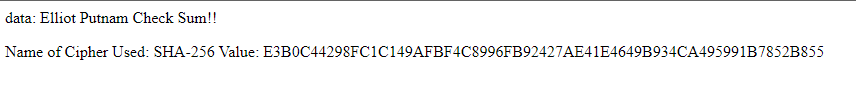
Insert a screenshot below of the CER file.

A computer screen shot of a black screen

Description automatically generated

## 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.

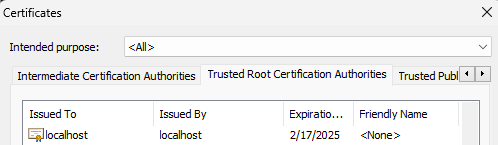


Figure 1 Trusted Root Certificate Authorities list containing the localhost certificate.

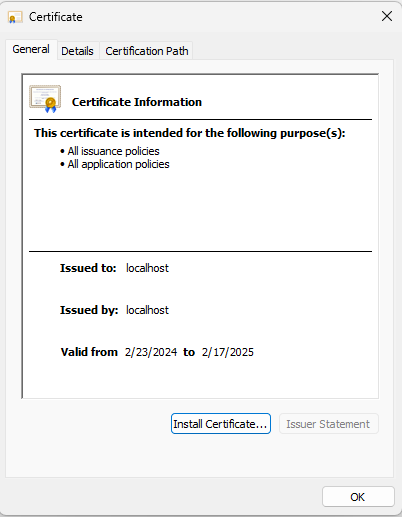


Figure 2 Certificate details for localhost certificate

## Secondary Testing

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

A screenshot of a computer program

Description automatically generated

Figure 3 Code for assignment

A screenshot of a computer

Description automatically generated

Figure 4 Dependencies report

## Functional Testing

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

A close-up of a text

Description automatically generated

Figure 5 Console output for application building and running successfully

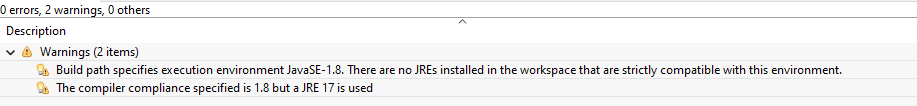


Figure 6 Error log depicting zero errors in build, some minor warnings

## Summary

As seen in this report, I have implemented a rest controller class which manages the servers hash for its checksum. This added class works along the guidelines of the vulnerability diagram and is to increase security of the server. I chose to work with the SHA-256 cipher due to the high security it provides and its low chance of collision. The dependency check results must be monitored for new vulnerabilities moving forward.

## Industry Standard Best Practices

Some standards and best practices I implemented during this analysis and design would be the implementation of a modern and secure cipher algorithm. Using SHA-256 cipher algorithm increases the security of the clients web service. I tried to write clean code and reduce any redundancies within to ensure readability. I created a selfsigned certificate that could later be submitted to a certificate authority to further secure the https connections.