****

# Practices for Secure Software Report

Table of Contents

[Practices for Secure Software Report 1](#_Toc190992586)

[Document Revision History 3](#_Toc190992587)

[Client 3](#_Toc190992588)

[Instructions 3](#_Toc190992589)

[Developer 4](#_Toc190992590)

[1. Algorithm Cipher 4](#_Toc190992591)

[2. Certificate Generation 4](#_Toc190992592)

[3. Deploy Cipher 4](#_Toc190992593)

[4. Secure Communications 5](#_Toc190992594)

[5. Secondary Testing 6](#_Toc190992595)

[6. Functional Testing 7](#_Toc190992596)

[7. Summary 9](#_Toc190992597)

[8. Industry Standard Best Practices 10](#_Toc190992598)

9. **Reference** …………………………………………………………………………………………………………………………. **11**

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **2/17/25** | **Carlos Rodriguez** | **Project Two Answers** |

## 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

Hello my name is Carlos Rodriguez

## Algorithm Cipher

The algorithm cipher that I chose for Artemis Financials’ was the SHA – 384 for the system. This cipher contains 384 bits level to help keep the security, to generate the checksum process and to help authenticate users. This hash function helps to verify the data integrity, helps to detect unusual activity and helps to authenticate the documents and certificates. The way it does this is by taking information and using mathematical operations to message digest that data. For small changes to the information, it will continue to generate a very different digest. This helps to keep the information secure and prevent any attacks onto the system. Once the cipher is generated, it is almost impossible to undo the changes and go back to its original form without its valid verification measures (GeeksforGeeks, 2024). Since I am using it with RSA encryption, this will be an asymmetric key. This means that it has a public key for encryption and a private key for decryption, which provides another form of security and authentication.

One of the early stages of encryption came from Egypt dating back to 1900 BC and has changed throughout many forms of history. One of the notable changes was made by IBM when they created the Data Encryption Standard. This standard was soon followed by the creating of asymmetric algorithms and the RSA public key system. Further development introduced the Advanced Encryption Standard, which is an upgrade from DES and makes this encryption key uncrackable by modern hardware. SHA – 1 family was an early encryption that had very significant vulnerabilities and was made better when the introduction of SHA – 2 occurred (Schneider, 2024). These measures will help to keep the information of the customers secure and help prevent attacks to Artemis Financials’.

## Certificate Generation

Insert a screenshot below of the CER file.

Below is my CER file that was generated using the Java Keytool. The encryption shows the SHA – 384 with RSA algorithm being used:

A screenshot of a computer

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

Provided below is my checksum value generated from the site with my name and a unique data string. Also shown is my refactored code with the checksum implementation:

A computer screen shot of a program

Description automatically generated

A black and white background

Description automatically generated with medium confidence

## Secure Communications

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

Below is the website of <https://localhost:8443/hash>, that shows that the webpage is secure and that the communication is secure:

A black and white screen

Description automatically generated with medium confidence

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.

The screenshots below is my refactored code executed without errors and right below it is the Maven dependency-check report: A computer screen shot of a program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer screen

Description automatically generated

## Functional Testing

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

After manually reviewing the code, we can tell that there were no syntactical, logical and security vulnerabilities. The codes were able to be refactored and were executed without errors. The algorithm ciphers were applied using the industry standards and correctly implementing the annotations like @RequestMapping, @RestController, and MessageDigest. There was a proper set up of the Test file and the application properties were properly set up to be able to access the <https://localhost:8443/hash> website. One thing that must be fixed is on the pom.xml and that is to keep up to date with the newest software versions of the maven dependency check and java. This will help to reduce any possible false positives and outdated security measures.

A computer screen shot of a program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen with text and images

Description automatically generated

## Summary

In reference to the Vulnerability Assessment Process Flow, I looked at the APIs, Controllers, Models, and Plug-Ins. These code reviews help us to understand if there are any flaws or vulnerabilities that our system might generate. With the Maven-Dependency Check, this helps provide us with a starting point to reference back into our system. The vulnerabilities identified will tell us what the issue is and how we can fix it. Most of the time it can be that we need to update to the latest version of the tool we are trying to use, or it can be a generated false positive. The plug-ins were important because we had to make sure that we built a secure form of communication with the website. Originally it did not generate into a secure site, but I was able to have a HTTPS site that helps create this layer of secure communication connection. The /hash function was part of the Controller that I refactored to make sure that it can implement the algorithm. This layer of security is the basis of the checksum verification and how we can help protect Artemis Financials’ information. The Maven-Check has new updated versions that come out and these updates help with finding any new or removed vulnerabilities.

## Industry Standard Best Practices

With the code structure, I was able to use Industry Standard Best Practices to be able to maintain the software’s existing security. One of the ways was by implementing the SHA – 384 with RSA encryption onto the software. This is one of the most used and well renowned security measures by the National Institute of Standards and Technology (NIST) that helps with the verification processes. The next step was utilizing the HTTPS to make sure the communication between the server and website was secure. This makes sure that the information would not be breached or stolen from the access on the website. The Dependency Check Tool helps us continuously generate reports that have documented issues and a very reliable Standard Practice. Providing in-text citations/ documentation within the code helps with what the functions are doing and leaving it open for scalability or modifications.

The value of applying these practices for secure codes are highly valued and should be taken very seriously for the company’s overall well-being. Making sure our data is secure is important to retain and bring in new customers to the company. Without delivering a secure form of communication/ data, would drive customers away and would give an overall negative image for the company. Customers and individual’s finances are one of the most important things in their lives and we must be able to create the highest form of security to make sure that their money/ information is not stolen or tampered with. Keeping up with the standards helps with minimizing risks of legal issues and data breaches that could possibly arise. It will show that we are staying up to date with security measures and we must constantly monitor and update software’s with the latest versions. Overall, the scalability for this system design will provide growth and continued improvements to Artemis Financials’ well-being and market presence.

References

**GeeksforGeeks. (2024, June 13). *How Does a Secure Hash Algorithm work in***

***Cryptography?* GeeksforGeeks.**

[**https://www.geeksforgeeks.org/how-does-a-secure-hash-algorithm-work-in-cryptography/**](https://www.geeksforgeeks.org/how-does-a-secure-hash-algorithm-work-in-cryptography/)

**Schneider, J. (2024, July 17). *The History of Cryptography | IBM*. Ibm.com.**

[**https://www.ibm.com/think/topics/cryptography-history**](https://www.ibm.com/think/topics/cryptography-history)