****

# 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** | **4/21/2024** | **Matthew Dunfee** |  |

## 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

Matthew Dunfee

## Algorithm Cipher

Ciphers are used to obfuscate data so that it becomes unintelligible to any person or persons whatsoever who do not have the means to decrypt it. There are two main types of encryption, Symmetric and Asymmetric. One way to imagine this is that there are two doors. Each door one door has one key the other has two. The door with one key’s key, can be used to lock and unlock it. The one with two keys has one key to lock and another to unlock.

Hashing is the process of changing data irreversibly into a fixed-size block of text. This is only used when the data doesn’t need to be retrieved but still needs to be verified. This process yields the same result given the same data is entered twice. In this way, a password can be stored without it being able to be retrieved later. To verify that password later the password must be reentered into the hashes and their results must be compared. This can also be used to check if data has been changed since its last hashing.

The people at Thalesgroup.com do a great job of explaining how encryption works. They also have a synopsis of the history of cryptography ranging from the B.C. to the modern era. That means the Caesar Cipher, the BOMBE in WWII, and even modern-day computer encryption.

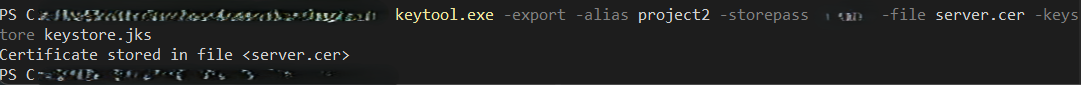
In terms of Artemis Financials’ needs, they should use Asymmetric encryption when communicating across the web. All users should! This form of communication is the standard for allowing communication to remain private between two parties on the web. Artemis Financial also needs to good hashing function since they are delivering sensitive data! For their needs, they should consider strong asymmetric encryption like RSA 4096-bit fs and a Hashing Function like SHA-256. Let's take a look at how something like that can be applied via self-signed certificates.

## Certificate Generation

Using Keytool.exe from PowerShell we can create a pair of keys.

A screenshot of a computer program

Description automatically generated



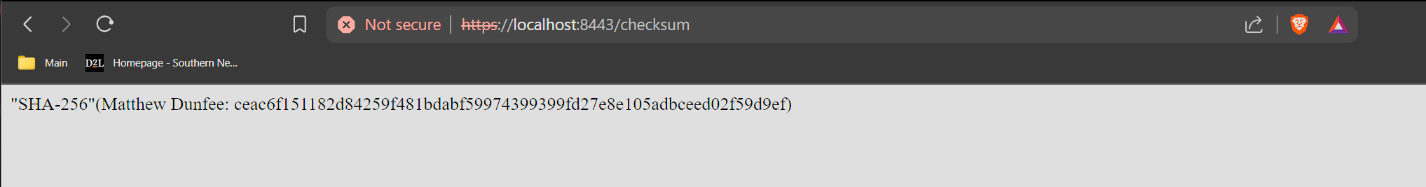
## Deploy Cipher

A screenshot of a computer

Description automatically generated

## Secure Communications

A screenshot of a computer

Description automatically generated

## Secondary Testing

A screenshot of a computer program

Description automatically generated

A screenshot of a computer screen

Description automatically generated

## Functional Testing

A computer screen with a black background

Description automatically generated

The two major issues I saw are related to the environment variables and the extra .cer file. I’m not sure if the .jks file should be left in the project itself either. I haven’t worked with Java and Github in the same project, but in the Node.js projects I’ve deployed, I always had the environment variable file path included in the .gitignore file. As for the extra .cer file, it's wasted space and just inviting trouble.

1. Summary

In summary, I hate Spring-Boot and Maven. Most of my time was spent struggling with the IDE and configuration just to spin up the project (wondering if I skipped a class that goes over those…). All jokes aside, in the context of Global Rain or Artemis Financial, this is an okay way to get around internal conflicts, but with the psychological impact of having to go deal with the warning page on the browser and even after that having the HTTPS have a line through it, this type of self-certifying is only acceptable for internal APIs. Its damage to the customer’s trust in the company would be incalculable.

## Industry Standard Best Practices

While this project covers a large swath of security, there is much work to be done. If this was a real-world application, there would need to be hours spent combing through dependency-check. There would need to be changes to the file structure to account for data leaks when pushing to Github. It would need a major reconstruction to avoid having a warning popping up on the browser. That is only considering the front end of the application. As former projects and assignments have shown, the company would also need to correct issues with security regarding access control (least privileges) and bolster defenses against direct attacks (sanitation and pen testing).

Citations:

Thales Group. (n.d.). A Brief History of Encryption. Retrieved from <https://www.thalesgroup.com/en/markets/digital-identity-and-security/magazine/brief-history-encryption>