****

# 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)

[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** | **[Date]** | **[Your Name]** |  |

## Client



## Instructions

Submit these 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

Frederick Brehm

## Algorithm Cipher

Artemis Financial is looking to modernize its operations by adding a file verification step to its web application to ensure secure communications. VAPF will primarily focus on the cryptography portion of the flow diagram to successfully integrate secure communication on their web application. This will be generated by utilizing a checksum. An algorithm cipher will need to be added to the existing software application to complete this task for Artemis Financial. The best algorithm cipher for this will be an AES or Advanced Encryption Standard. This method utilizes bits of different sizes to create a secure key.

One of these methods is known as a hash function. We will be utilizing SHA-256 which is a hash function that will utilize a 256-bit key. SHA which stands for “secure hash algorithm” is the standard of the NIST for use with most businesses and governments nowadays. Having a 256-bit key means that the hash value that is generated will remain 256 bits, no matter the size of the plain text. Random numbers are extremely important in cryptography as they provide what is known as an RNG or random number generator. RNG’s are used to provide a key that is generated randomly. This is crucial in security as nobody knows the random number except for the person generating it, they provide unpredictability. Symmetric cryptography is often what we hear and use most. It is used for encryption rather than signing. It provides a secret key to scramble information making it impossible to unscramble without the key. In symmetric cryptography, the same key is used for both encrypting and decrypting information. Asymmetric cryptography, also known as public key cryptography, is essential for signing. This process uses two different keys, a secret key that is only known to the sender used to sign the data, and the public key which can be known to anyone which is used to validate the signature. Because asymmetric keys utilize 2 keys that are often more secure than symmetric keys when dealing with internet communications. Encryption dates as far back as 1900 BC when one of the first encryption methods was utilized. In 600 BC ancient Spartans utilized a scytale device to send secret messages during battle, which became one of the first true uses of ancient cryptography. In Circa 60 BC, Julius Ceasar expanded on cryptography by creating a cypher that shifted characters by three places in the alphabet, becoming a simple yet effective coding method at the time. Fast forward to modern day cryptography, a group at IBM created block cyphers to protect their customers data. In 1973 the US adopted that national standard, knows as Data Encryption Standard. DES remained in use until it was cracked in 1997. In 2000 AES or Advanced Encryption Standard surfaced to replace DES. Today, AES rules the world in Encryption standard and impossible to crack, making it the preferred encryption standard for companies and governments worldwide.

## Certificate Generation

Insert a screenshot below of the CER file.

A computer screen with white text

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

Unfortunately, I have had several issues when trying to deploy the actual cipher for this project through for the checksum verification. I have utilized resources such as google, stackoverflow, geeks on geeks, YouTube videos, oracle, the Brightspace resources from the previous and current modules. I have tried troubleshooting as much as possible, from disabling the firewall, to emptying the DNS, to checking the proxy, to adding port 8443 and unfortunately none of these have helped my <https://localhost:8443/hash> work successfully. I am at a loss for what the issue is that I am clearly not understanding or seeing it.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A computer screen with white text

Description automatically generated

## Secure Communications

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

Refer to previous answer as this caused the issue of not being able to provide a screenshot for this section.

## 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

A screenshot of a computer

Description automatically generated

## Functional Testing

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

A screen shot of a computer

Description automatically generated

## Summary

The code has been refactored to provide a better sense of security by implementing specific parts of the vulnerability assessment flow diagram through security, API’s, cryptography, client/server, code error and code quality. We were supposed to create our own self signed certificate which I did however, I was unfortunately unable to successfully connect to the restFul API server which in terms shows that my attempt to security, API, and client/server have failed. However, even though I came across several issues and troubleshooted for days I was still able to learn from this project in creating software that @requestmapped and created a restController to provide security. I was able to utilize organization and readability to create good code quality and the least amount of code error in my eyes. Even though I did not successfully connect to the server I was able to learn a lot about how API’s work when interacting between reader and sender and the encryption/decryption process behind it. Many layers of protection were achieved throughout this project. First refactoring the code made it possible to establish a secure connection to an HTTPS. Further we created code that would connect to the restFul API which would require a signed certificate to have access. The dependency check shows 16 vulnerabilities which could all be updates or false positives. More detail and research would have to go into those dependencies, but it is something that could be done to improve overall security.

## Industry Standard Best Practices

Secure communications are extremely important when it comes to any interaction or communication over the internet. Companies, the public, and the government all have extremely sensitive information that is sent through the web daily. I used industry best practices to maintain the existing companies’ software. This was done by refactoring the code and building off the previous code written. Employing practices such as intermittent testing, code quality, encryption, and signed certificates helped create a more stable security for the company. Encryption/decryption are most important when creating a secure connection between reader and sender. Security breaches can cripple a business and government as the public will no longer see that company or entity as fit to handle their information. The “software engineers code of ethics” principles mostly revolve around keeping the safety of the public in best interest. If a company or government entity fails to do so due to poor security or lack of encryption/decryption it could potentially be life threatening to consumers. Companies will take huge financial hits, trust will dissipate, their name will be stained, and they will have to use several resources to try and recover from such an attack. This also goes for the government, as they are in control of our economy, judicial system, infrastructure, and military, a breach to one of these entities could be potentially deadly on a global scale. The constant pursuit to utilize best practices by using agile methods, employing the best encryption and decryption standards like AES and symmetric or asymmetric, utilizing self-signed certificates and following standards and guidelines will help keep the internet as secure as possible while maintaining the trust and health of the public.