****

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[Document Revision History 3](#_Toc33111302)

[Client 3](#_Toc33111303)

[Instructions 3](#_Toc33111304)

[Developer 4](#_Toc33111305)

[1. Algorithm Cipher 4](#_Toc33111306)

[2. Certificate Generation 4](#_Toc33111307)

[3. Deploy Cipher 4](#_Toc33111308)

[4. Secure Communications 4](#_Toc33111309)

[5. Secondary Testing 4](#_Toc33111310)

[6. Functional Testing 5](#_Toc33111311)

[7. Summary 5](#_Toc33111312)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **8/10/2022** | **Daniel Behmer** |  |

## Client



## Instructions

Deliver this completed Practices for Secure Software Report documenting your process for writing secure communications and refactoring code that complies with software security testing protocols.

Respond to the steps outlined below and replace the bracketed text with your findings in your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

Daniel Behmer

## 1. Algorithm Cipher

There are many algorithm ciphers that could work for our project; however, I believe that SHA-256 encryption would be the best for this case. It was created by the National Security Agency in 2001 and is used as the standard for most encryption practices. SHA-256 also utilizes hashing. Hashing is when data is transformed into a secure format that is unreadable without a hash key. Through SHA-256 hashing, a 512-bit string of data can be converted to a 256-bit string. After this process, it is virtually impossible to convert the 256-bit string back to its original 512-bit form. The use of symmetric and non-symmetric keys and random numbers in encryption is because it allows for complex ciphers to be harder to crack. SHA-256 has the capacity to supply 2^256 possible hash values, keeping the possibilities of collisions to virtually nothing. Even changing one of the numbers or characters within the hash would alter the hash so much that it wouldn’t even remotely represent the original hash (N-able, 2019).

## 

## 2. Certificate Generation

Generate appropriate self-signed certificates using the Java Keytool, which is used through the command line.

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

## 3. Deploy Cipher

Refactor the code and use security libraries to deploy and implement the encryption algorithm cipher to the software application. Verify this additional functionality with a checksum.

Graphical user interface, text

Description automatically generated

## 4. Secure Communications

Refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code to verify secure communication by typing **https://localhost:8443/hash** in a new browser window to demonstrate that the secure communication works successfully.

Graphical user interface, text

Description automatically generated

## 5. Secondary Testing

Complete a secondary static testing of the refactored code using the dependency check tool to ensure code complies with software security enhancements. You only need to focus on the code you have added as part of the refactoring. Complete the dependency check and review the output to ensure you did not introduce additional security vulnerabilities.

A picture containing text

Description automatically generated

Graphical user interface

Description automatically generated

## 6. Functional Testing

Identify syntactical, logical, and security vulnerabilities for the software application by manually reviewing code.

Graphical user interface, text, application

Description automatically generated

A picture containing text

Description automatically generated

## 7. Summary

After viewing the Vulnerability Assessment Process Flow Diagram, there are a few areas of security that we addressed when refactoring the code. The first area is Input Validation. Because we are creating a new string that will carry unique data (our first and last name), it is important that the input is free of error. Another area addressed is the use of APIs. We want the API to be secure because they are the head of connecting services and the transferring of data. If the API is not secure, it will ultimately lead to data breaches (MicroFocus, 2022). The third area of focus is Cryptography. Cryptography is important because it ensures that data can be kept confidential. There are many ways this is achieved. One way that was explored throughout this project was the use of encryption. “Encryption is a way of scrambling data so that only authorized parties can understand the information.” (Cloudflare, Inc., 2022). The encryption cipher that I used in this particular project is SHA-256. This cipher is a hash function that produces a value that is 256 bits long, and once it is transformed into that value, it is nearly impossible to convert it back to its original value. There is also 2^256 possible outcomes for that value, this makes the potential of collisions (two messages having the same hash value) nearly impossible. So small that there are more possible outcomes for the hash values than there are atoms in the known universe (N-able, 2019). The last area that was taken into consideration was Code Quality. We want the ability to maintain our program, with this said, we need to ensure that the code is both readable and scalable.

With Code Quality in mind, it is important that we have the ability to add layers of security. According to IBM, there are multiple layers of security that are important to consider: system level security, network level security, application-level security, and transmission level security. System level security (IBM, 2021). System level security represents the last line of defense in an Internet-based security platform. System level security should be considered the first step in developing security. Network level security measures control of the operating system to other network systems. Typically, firewalls are the most common means of network level security. Application-level security is security that controls how users interact with specific system applications. This needs to be considered as it deals with both server-side and client-side security. Lastly, transmission level security protects data within and across networks (IBM, 2019).

To maintain security of the software application, it is important to consider all of the potential attacks, and what would be at stake. Since we are dealing with sensitive banking information, we will need to prepare for data breaches that will leak user’s information like account/routing number, account balance, or more personal information like address, social security number, phone number, username and password, etc. Frequent testing of our application and patching could help reduce the possibility of these breaches (Synopsys Editorial Team, 2020). It would also be beneficial to automate routine tasks. By automating day-to-day security tasks, it allows for security staff to focus on more strategic security initiatives (Synopsys Editorial Team, 2020).

Sources Cited

N-able. (September 12, 2019). *SHA-256 Algorithm Overview.* <https://www.n-able.com/blog/sha-256-encryption#:~:text=SHA%2D256%20is%20a%20patented,as%20long%20as%20when%20unencrypted>.

MicroFocus. (2022). *What is API Security?* <https://www.microfocus.com/en-us/what-is/api-security#:~:text=Why%20is%20API%20security%20important,Report%20by%20Micro%20Focus%20Fortify>.

Cloudflare, Inc. (2022). *What is encryption?* [*https://www.cloudflare.com/learning/ssl/what-is-encryption/*](https://www.cloudflare.com/learning/ssl/what-is-encryption/)

IBM. (April 14, 2021). *The layered defense approach to security.* [*https://www.ibm.com/docs/en/i/7.3?topic=security-layered-defense-approach*](https://www.ibm.com/docs/en/i/7.3?topic=security-layered-defense-approach)

Synopsys Editorial Team. (June 29, 2020). *Are you following the top 10 software security best practices?*

<https://www.synopsys.com/blogs/software-security/top-10-software-security-best-practices/>