****

# 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** | **10/15/2022** | **Muhammad Atif** |  |

## Client



## Instructions

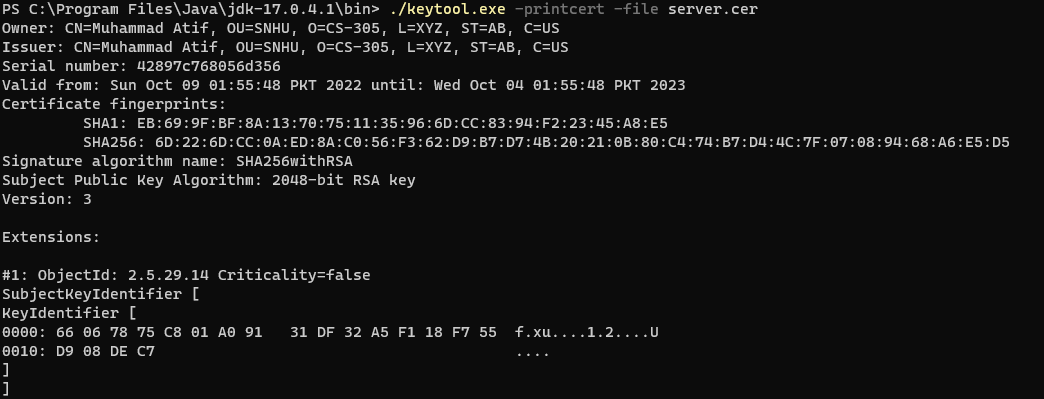
## Developer

Muhammad Atif

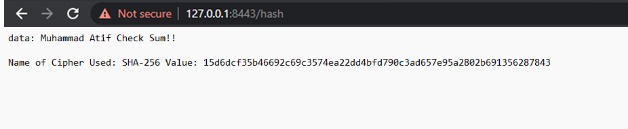
## 1. Algorithm Cipher

To maintain safe connections, Artemis Financial has requested increased protection for their online application. Assuming that the most likely attack vector against a financial institution will be a bad actor aiming to achieve financial benefit by accessing the information maintained, encryption is the best advice. This renders the files unusable to any would-be attacker in the absence of a key. I would propose Asymmetric communication to the company in order to safeguard communication. That is, the key to encrypt is public, but the key to decode is private. To provide the best level of security, as this information may be communicated outside, I recommend encrypting it with the SHA-256 cypher algorithm and 256-bit keys. SHA-256 encryption provides good high-level bit encryption with a large number of distinct key combinations with a key length of 256 bits. Furthermore, the SHA-256 technique employs Java's random number generator, guaranteeing that the encryption is extremely secure by generating a non-reversible checksum that certifies the file's validity. The hash function will generate a checksum of the specified message using the SHA-256 encryption.

## 2. Certificate Generation

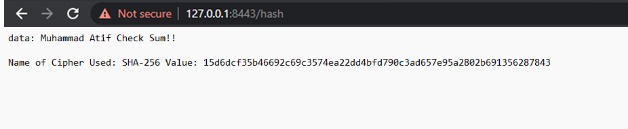


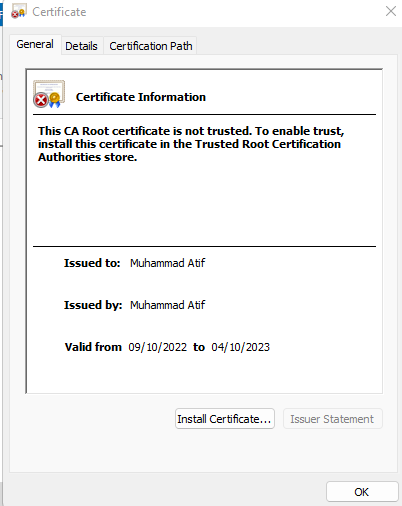
## 3. Deploy Cipher



## 4. Secure Communications

* Showing the HTTPS is working but that my Cert isn’t official because it’s self signed.





## 5. Secondary Testing

Text, application

Description automatically generated

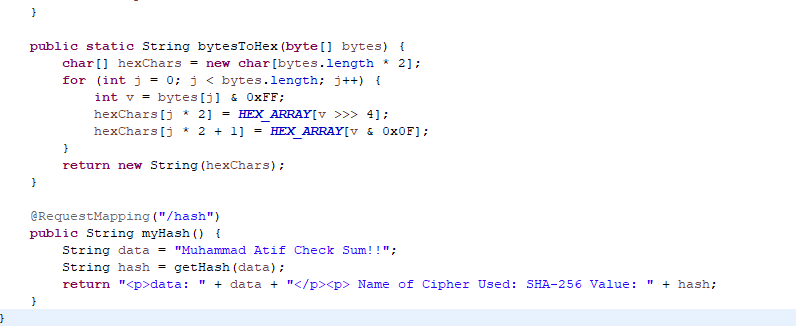
Graphical user interface, text, application

Description automatically generated

## 6. Functional Testing

Graphical user interface, text, application

Description automatically generated



## 7. Summary

I'm refactoring my code. I've created a protected RestController to act as the secure controller for my program's RESTful stop. The ServerController class is responsible for matching the problems depicted in the vulnerability assessment diagram. I also selected to use the SHA-256 hashing cypher since it is incredibly safe and has a very low possibility of collisions. To best maintain the application's existing security, I would recommend once or twice weekly dependency checks to stay up to speed on any vulnerabilities. This will assist to secure the firm and its sensitive data. Keeping the plugins within the pom.xml would also ensure that the most recent revisions of the plugins are running, guaranteeing the maximum level of security.