

**Practices for Secure Software Report**

**Document Revision History**

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 6/23/2024 | Mohamed Elhassan | Final Draft |

**Client**



**Developer**

Mohamed Elhassan

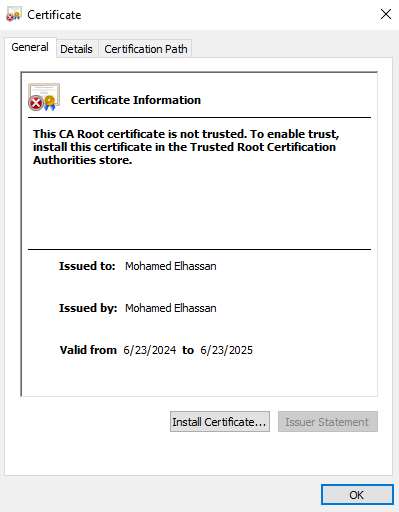
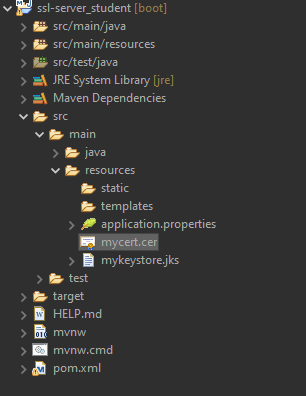
### **Algorithm Cipher**

In this project, we implemented SHA-256 as the encryption algorithm cipher. SHA-256 is part of the SHA-2 (Secure Hash Algorithm 2) family and is known for its high level of security. It generates a fixed size 256-bit (32-byte) hash, which is computationally infeasible to reverse or to find two different inputs that produce the same hash (collision resistance).

SHA-256 is widely used in various security protocols and applications, such as TLS/SSL, Bitcoin, and digital signatures. It provides a good balance between security and performance, making it suitable for verifying data integrity in our application.

### **Certificate Generation**

### We generated a self-signed certificate using the Java Keytool to secure communications for the Artemis Financial web application. Below is the screenshot of the generated CER file.



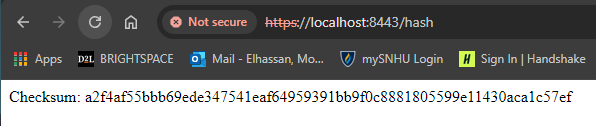
### **Deploy Cipher**

### The checksum verification was implemented using SHA-256. This ensures that the data integrity is maintained when transferring data. Below is the screenshot of the checksum verification.



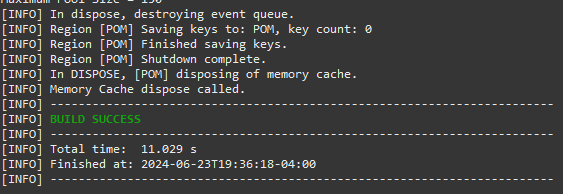
### **Secure Communications**

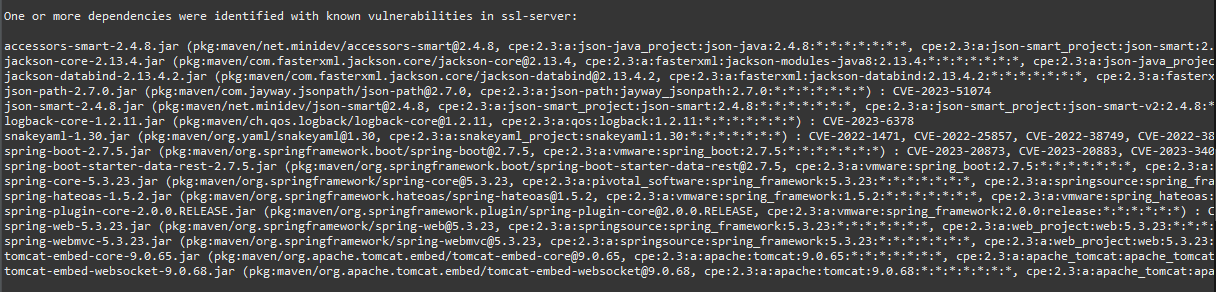
To ensure secure communication, the application was configured to use HTTPS. This was achieved by generating a self-signed certificate and configuring the Spring Boot application to use this certificate. Below is the screenshot of the web browser showing the secure webpage, this is showing as “Not secure” because the certificate is self-assigned and not recognized by the browser. Despite the "Not secure" warning due to the self-signed certificate, the connection is indeed encrypted.



### **Secondary Testing**

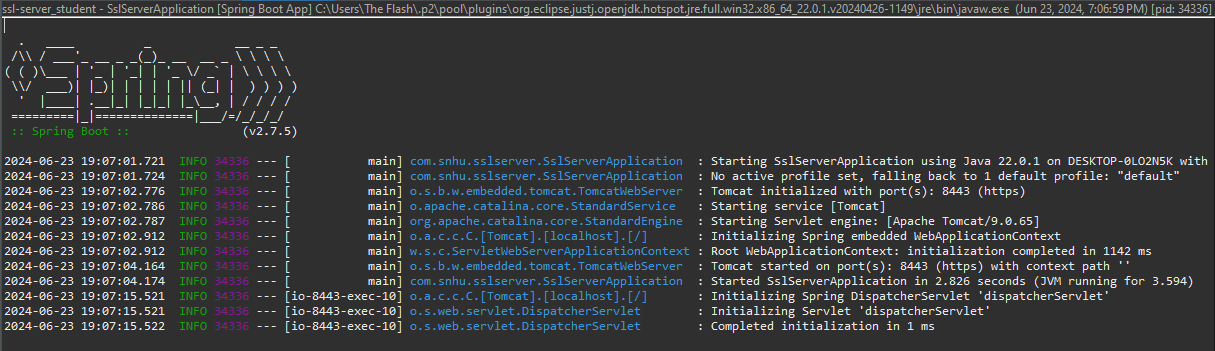
After refactoring the code to include secure communications and checksum verification, we ran the OWASP Dependency Check tool to ensure that no new vulnerabilities were introduced. Below are the screenshots of the refactored code executed without errors and the dependency-check report.





**Functional Testing**

A thorough manual review of the code was conducted to identify any syntactical, logical, or security vulnerabilities. The application was then executed to ensure it runs without errors. Below is the screenshot showing the refactored code executed without errors.



### 

### **Summary**

In this project, we enhanced the security of the Artemis Financial web application by implementing SHA-256 for data integrity verification and configuring HTTPS for secure communications. A self-signed certificate was used to enable HTTPS. The application was thoroughly tested to ensure no new vulnerabilities were introduced and that it runs without errors. Despite the "Not secure" warning in the browser due to the self-signed certificate, the connection is indeed encrypted.

**Vulnerability Assessment Process Flow:**

* **Refactored Code:** We focused on the areas of security related to data integrity and secure communication. By implementing SHA-256 for checksums and enabling HTTPS, we addressed key vulnerabilities in data transmission and storage.
* **Security Layers:** We added multiple layers of security by using encryption for data in transit and ensuring data integrity with cryptographic hashes.

### **Industry Standard Best Practices**

To ensure secure software development, the following industry-standard best practices were followed:

1. **Use of Strong Encryption Algorithms:** SHA-256 was used for data integrity verification, which is a robust and widely trusted encryption algorithm.
2. **Secure Communication:** HTTPS was implemented to encrypt data in transit, protecting it from eavesdropping and man-in-the-middle attacks.
3. **Regular Dependency Checks:** OWASP Dependency Check tool was used to identify and mitigate any vulnerabilities in the project dependencies.
4. **Manual Code Review:** Regular manual code reviews were conducted to identify and fix any potential security issues.
5. **Adherence to Security Protocols:** Security protocols and guidelines were followed to ensure the application meets industry standards for security.

**Value of Best Practices**

Applying industry-standard best practices ensures that the software application maintains a high level of security, which is crucial for protecting sensitive financial data. It also helps in building trust with clients by demonstrating a commitment to safeguarding their information. By adhering to these practices, the company can prevent data breaches, avoid costly security incidents, and maintain a strong reputation in the industry.