GLOBALRAIN

Practices for Secure Software Report

Table of Contents

DOCUMENT REVISION HISTORY..... 3
CLIENT..... 3
INSTRUCTIONS..... 3
DEVELOPER..... 4
1. ALGORITHM CIPHER..... 4
2. CERTIFICATE GENERATION..... 4
3. DEPLOY CIPHER..... 5
4. SECURE COMMUNICATIONS..... 5
5. SECONDARY TESTING..... 6
6. FUNCTIONAL TESTING..... 7
7. SUMMARY..... 7
8. INDUSTRY STANDARD BEST PRACTICES..... 8

Document Revision History

Version	Date	Author	Comments
1.0	20-OCT-2025	Joseph Ebersole	Initial Changes

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.

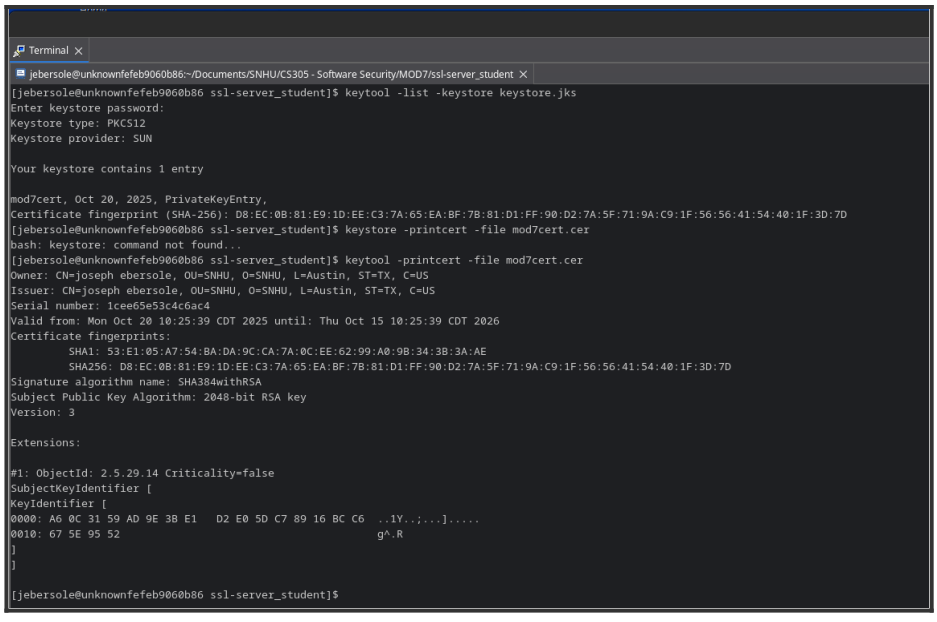
Developer
Joseph Ebersole

1. Algorithm Cipher

Artemis Financial's goal to secure data transmission and verification makes the SHA-256 algorithm an excellent choice. SHA-256 is widely recognized as one of the most secure and reliable hashing algorithms in use today, and is particularly fit for the needs required by Artemis. By implementing SHA-256, Artemis Financial can create a checksum to verify the integrity of files exchanged through its web application. If the calculated checksum matches the original, the data is confirmed to be authentic and unaltered.

SHA-256 is widely trusted due to its collision resistance, meaning it is nearly impossible for two different inputs to produce the same hash value. While SHA-256 does not use random numbers or encryption keys since it is not an encryption algorithm, random numbers and cryptographic keys still play an important role in overall data security. For instance, symmetric encryption methods like AES use the same secret key for both encryption and decryption, while asymmetric encryption methods like RSA use a public and private key pair for secure key exchange and authentication. Together, these technologies provide Artemis with a strong defense system. By using SHA-256 for data verification along with encryption methods for confidentiality, Artemis Financial can ensure its data remains protected from unauthorized modification.

2. Certificate Generation



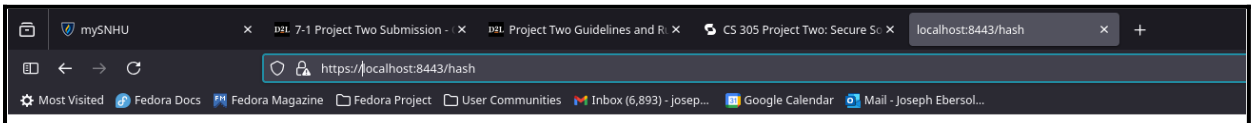
3. Deploy Cipher

data: Hello Joseph Ebersole, I love check sum!

Name of Cipher Algorithm Used: SHA-256

CheckSum Value: 8F2CC880DA1954F59B20D60F29586410D5C8A7FF40543894BB92450D033FA544

4. Secure Communications

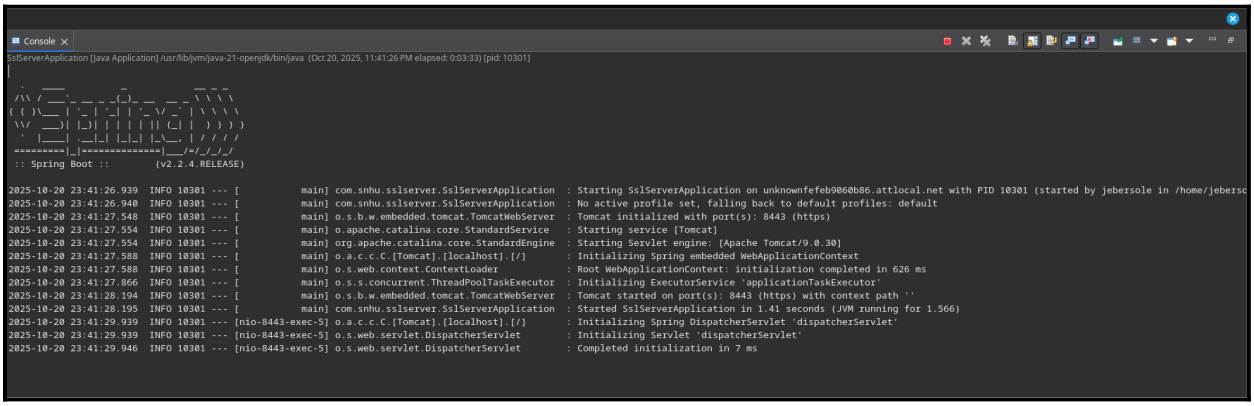


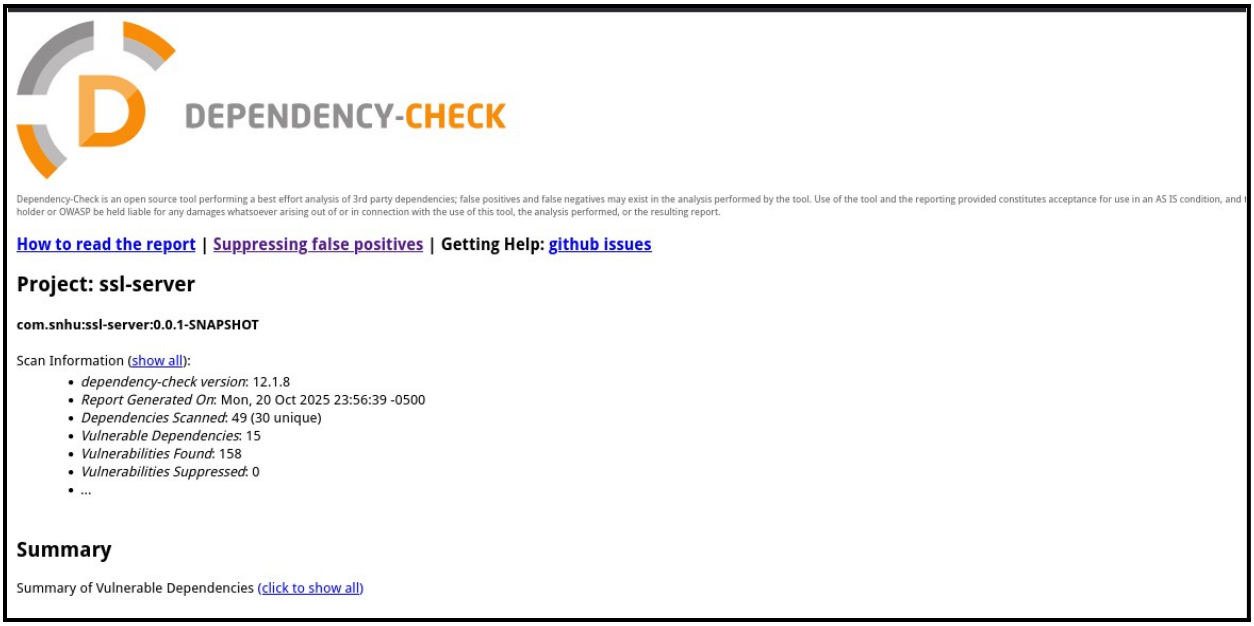
data: Hello Joseph Ebersole, I love check sum!

Name of Cipher Algorithm Used: SHA-256

CheckSum Value: 8F2CC880DA1954F59B20D60F29586410D5C8A7FF40543894BB92450D033FA544

5. Secondary Testing





6. Functional Testing

SslServerApplication.java | MessageController.java | application.properties

```java
package com.snhu.sslserver;

import org.springframework.web.bind.annotation.GetMapping;

//FIXME: Add route to enable check sum return of static data example:  String data = "Hello World Check Sum!";

@RestController
public class MessageController {

    @GetMapping("/hash")
    public String getHash() {
        String data = "Hello Joseph Ebersole, I love check sum!";
        String hashUsed = "SHA-256";

        String htmlResponse = "";
        htmlResponse = htmlResponse + "<h3>data: " + data + "</h3>";
        htmlResponse = htmlResponse + "<h3>Name of Cipher Algorithm Used: <u>" + hashUsed + "</u></h3>";

        try {
            MessageDigest md = MessageDigest.getInstance(hashUsed);
            md.update(data.getBytes());
            byte[] digest = md.digest();

            String hex = "";
            for (int i = 0; i < digest.length; i++) {
                hex += String.format("%02X", digest[i]);
            }

            htmlResponse += "<h2> CheckSum Value: " + hex + "</h2>";

        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (Exception e) {
            System.out.println("something unexpected occured");
            e.printStackTrace();
        }



        return htmlResponse;

    }
}
```

7. Summary

For the Artemis Financial project, I refactored the web application to include multiple layers of security that align with modern software security practices. I began by using the Java Keytool utility to generate selfsigned SSL certificates for HTTPS communication, ensuring that all data transmitted between client and server is encrypted. Implementing HTTPS protects sensitive financial and personal information from interception or man-in-the-middle attacks, which is a critical first layer of defense. In addition, I developed an API endpoint, /hash, that uses the SHA-256 hashing algorithm to generate a checksum for message verification. This implementation demonstrates the use of a secure, one way hash function to validate data integrity, ensuring that transmitted data has not been altered or tampered with.

8. Industry Standard Best Practices

When refactoring the code, I followed industry standard best practices for secure coding to mitigate common vulnerabilities. For example, I handled exceptions safely using structured try/catch blocks to prevent stack trace exposure in production environments, which could reveal sensitive system information. I also avoided using outdated or weak algorithms, opting for SHA-256, which is part of the SHA-2 family and widely recommended by NIST for secure message digests. Furthermore, I ensured that user input and system data were properly handled and not exposed directly in the response, maintaining confidentiality and preventing potential injection attacks. Overall, these measures align Artemis's system with recognized cybersecurity standards and demonstrate a proactive approach to mitigating known threats.