****

# Practices for Secure Software Report

**Table of Contents**

[**Document Revision History 3**](#_i5mjlsfn9wp9)

[**Client 3**](#_jdr9sxkta7pw)

[**Instructions 3**](#_vhlm95zer1m0)

[**Developer 4**](#_12m0o4mmivqu)

[**1. Algorithm Cipher 4**](#_xqg518en9hr5)

[**2. Certificate Generation 4**](#_ymc4v4pejtjp)

[**3. Deploy Cipher 4**](#_o7e1hsr84esn)

[**4. Secure Communications 4**](#_o0tju5z4mryx)

[**5. Secondary Testing 4**](#_s1i87as2trl9)

[**6. Functional Testing 4**](#_hed19554klc6)

[**7. Summary 4**](#_p8jvll1jgckq)

[**8. Industry Standard Best Practices 4**](#_31xfe9gchwyc)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **10/19/25** | **Abdullah Jafri** |  |

## 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

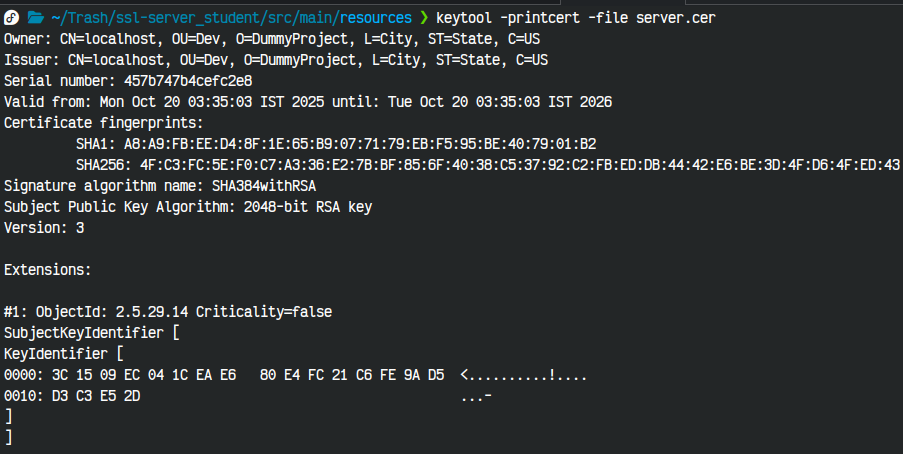
Abdullah Jafri

## Algorithm Cipher

SHA-256 is a cryptographic hash function from the SHA-2 family that turns any input data into a fixed 256-bit output, creating a unique “fingerprint” for verifying data integrity. Unlike traditional encryption, it does not use keys, though it can be combined with a secret key in methods like HMAC. The algorithm is deterministic, collision-resistant, and produces drastically different results even for small changes in the input. SHA-256 replaced older, weaker algorithms like SHA-1 and is widely used today in SSL certificates, blockchain, and file integrity checks. It does not rely on random numbers by itself, but can use them in techniques like salting for password hashing, making it secure and practical for modern applications.

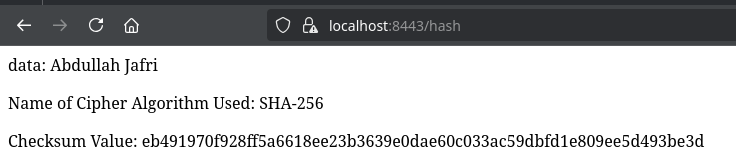
## Certificate Generation

Insert a screenshot below of the CER file.



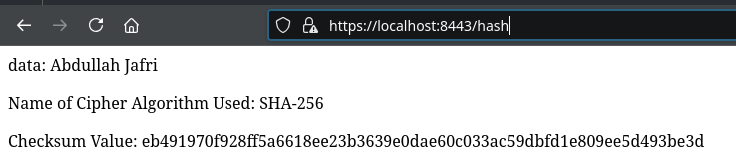
## Deploy Cipher

Insert a screenshot below of the checksum verification.



## Secure Communications

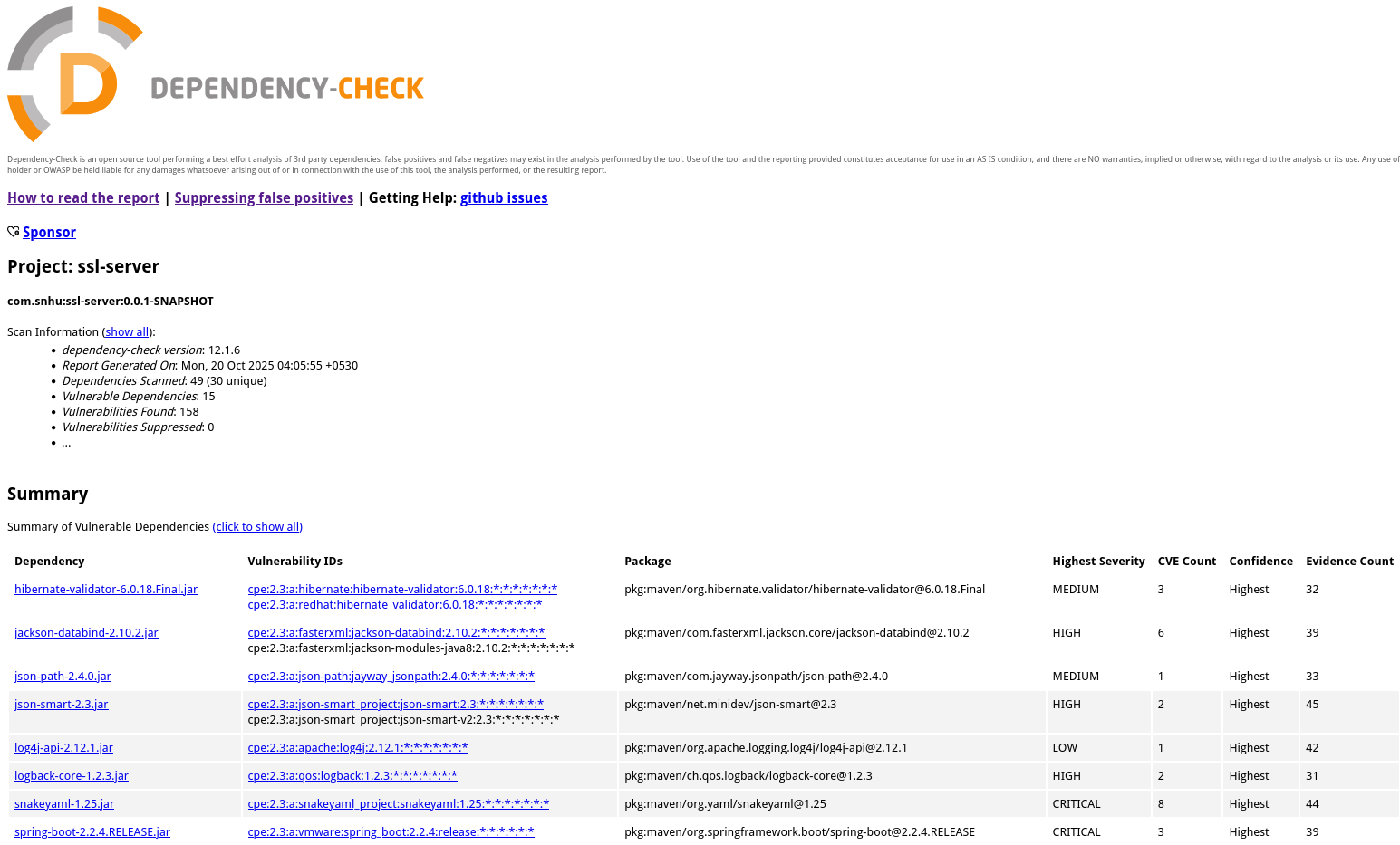
Insert a screenshot below of the web browser that shows a secure webpage.



## Secondary Testing

Insert screenshots below of the refactored code executed without errors and the dependency-check report.







## Functional Testing

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



## Summary

I refactored the code by adding a secure RestController to the SSLServerApplication.java file to handle the hash endpoint safely. This ServerController helps address the security issues shown in the Vulnerability Assessment Diagram by keeping the code simple and reducing possible attack points. I used SHA-256 for hashing because it is strong and reliable, and the code is kept minimal to avoid extra risks. I also updated the Maven Dependency Check plugin from version 5.3.0 to 12.1.6 so the software can check for vulnerabilities in its libraries with the latest version. These steps add extra layers of security and follow good practices for keeping the application safe.

## Industry Standard Best Practices

I applied industry standard best practices by keeping the code simple, secure, and up-to-date. For example, I used a secure RestController for the hash endpoint, implemented SHA-256 for strong hashing, and minimized unnecessary code to reduce potential attack points. I also updated the Maven Dependency Check plugin to the latest version to identify vulnerabilities in external libraries. Following these best practices helps maintain the application’s security over time and ensures that known risks are addressed before they can be exploited. Applying these practices benefits the company by protecting sensitive data, maintaining customer trust, and reducing the chance of costly security breaches.