****

Brad Mills

# Practices for Secure Software Report

Table of Contents

[Document Revision History 3](#_Toc102040754)

[Client 3](#_Toc102040755)

[Instructions 3](#_Toc102040756)

[Developer 4](#_Toc102040757)

[1. Algorithm Cipher 4](#_Toc102040758)

[2. Certificate Generation 4](#_Toc102040759)

[3. Deploy Cipher 4](#_Toc102040760)

[4. Secure Communications 4](#_Toc102040761)

[5. Secondary Testing 4](#_Toc102040762)

[6. Functional Testing 4](#_Toc102040763)

[7. Summary 4](#_Toc102040764)

[8. Industry Standard Best Practices 4](#_Toc102040765)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **8/17/2023** | **Brad Mills** | **Final Version** |

## 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

[Insert your name here.]

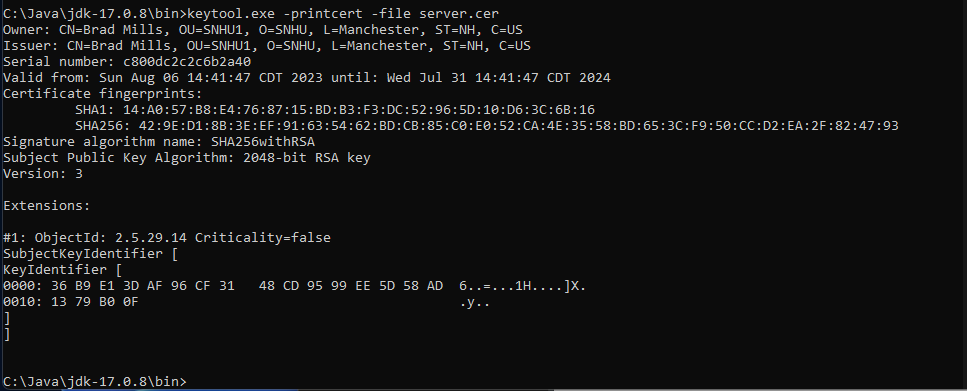
## Algorithm Cipher

I recommend the SHA256 cipher algorithm to avoid collisions. SHA256 belongs to the SHA-2 family of hash functions and uses a 256-bit key to transform data into a unique fixed-length string. This resultant string, known as a hash value, is consistently 256 bits long. The security of SHA-256 is attributed to three main features:

1. Deciphering the original data from the hash value is nearly impossible. It would necessitate a brute-force attack to make 2^256 attempts to retrieve the original data.
2. The probability of two distinct messages producing the same hash value, a situation known as a collision, is extremely low.
3. On another note, the Advanced Encryption Standard (AES) is recognized and trusted by the U.S. government and various entities. While the 128-bit version of AES is highly effective, it also comes in 192- and 256-bit key variants for more stringent encryption needs. It's worth noting that encryption techniques have been in use for many years.

## Certificate Generation

Insert a screenshot below of the CER file.



A screenshot of a certificate

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a computer

Description automatically generated

## Secure Communications

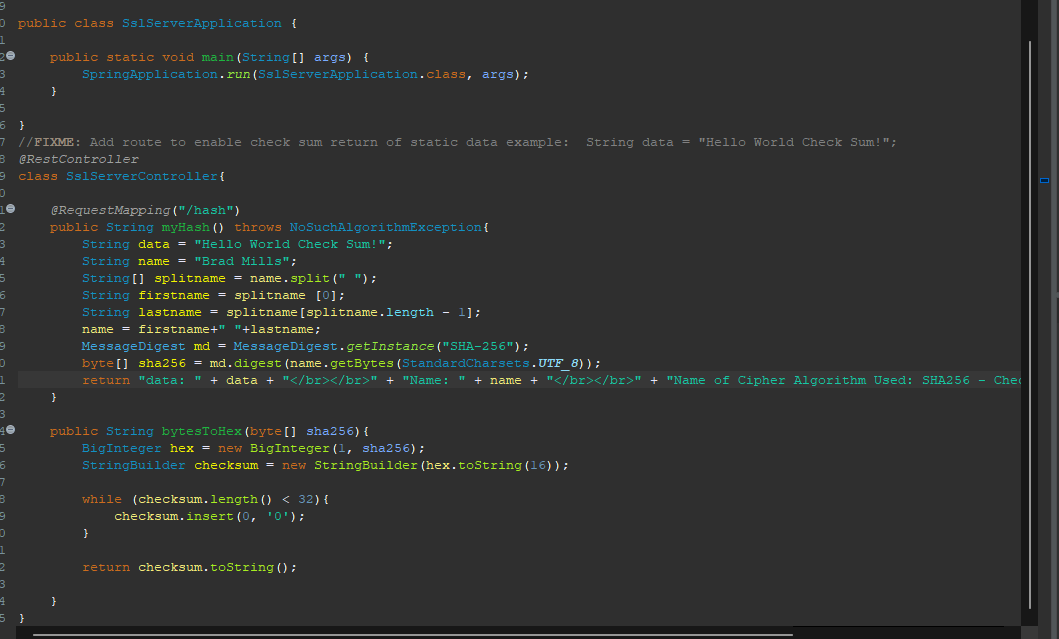
Insert a screenshot below of the web browser that shows a secure webpage.

A screenshot of a computer

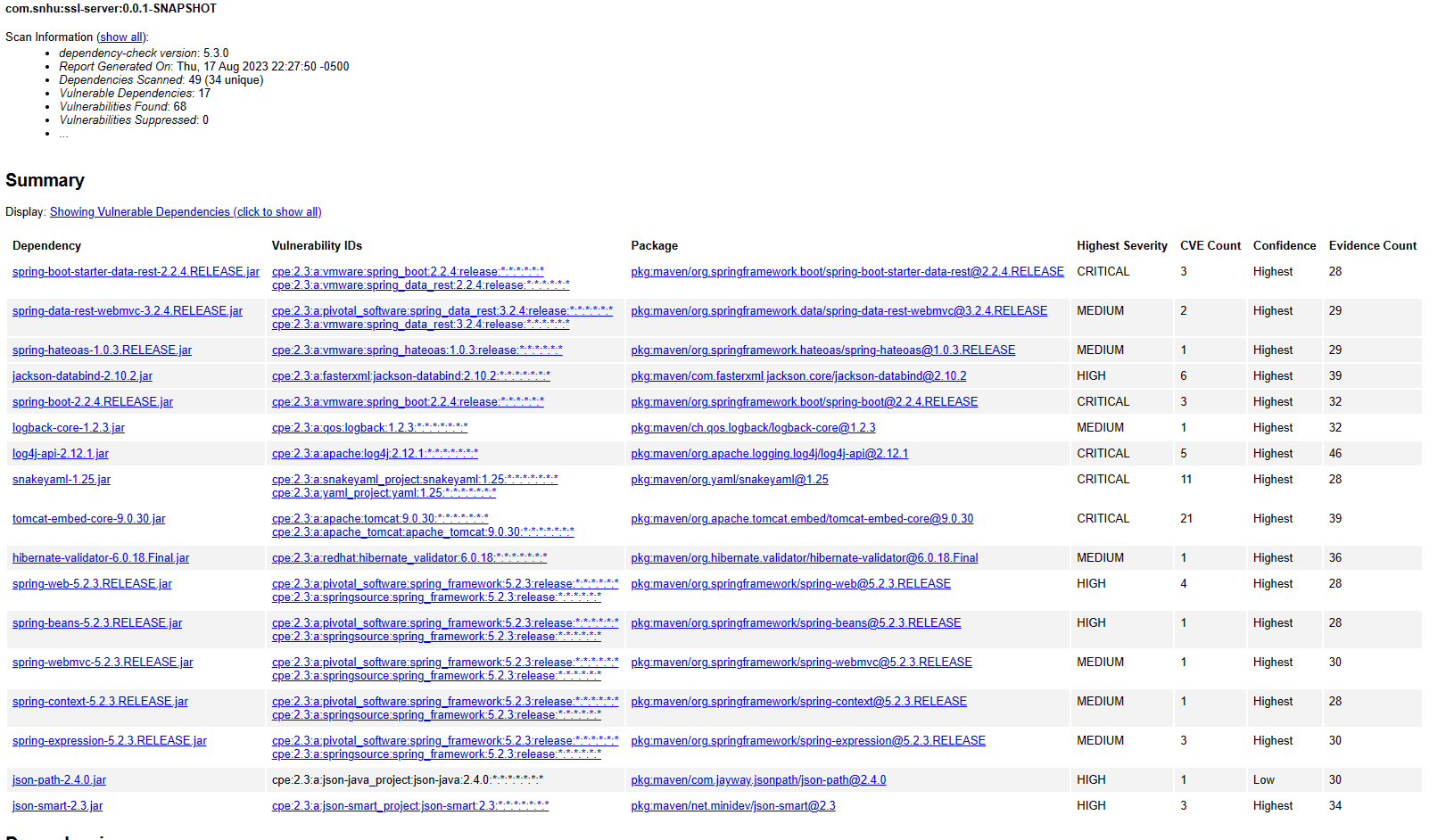
Description automatically generated

## Secondary Testing

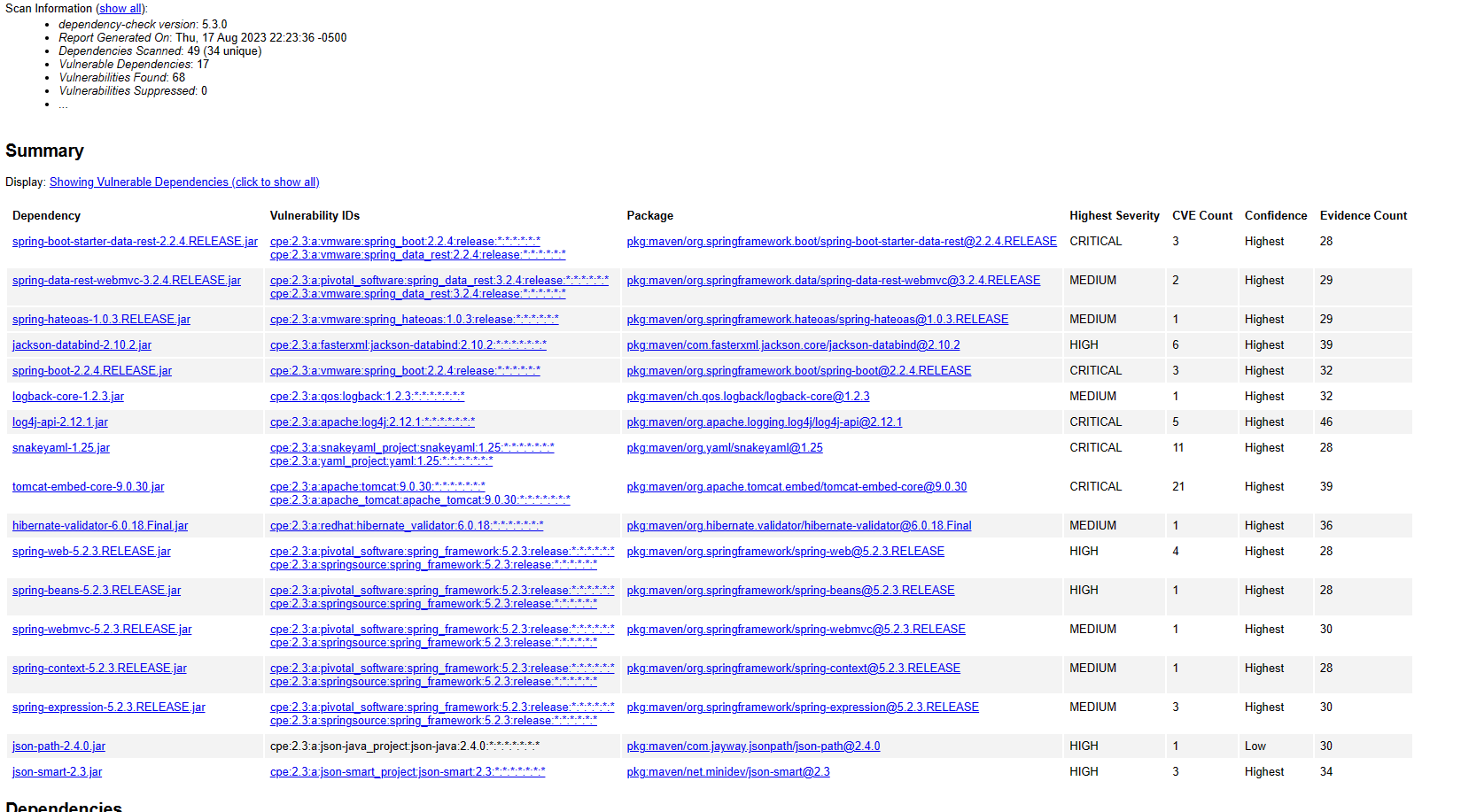
Insert screenshots below of the refactored code executed without errors and the dependency-check report.



Pre-code revision



Post-code revision



## Functional Testing

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

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Summary

I enhanced our app's security by incorporating self-signed certificates, allowing us to safely use HTTPS. We also spruced up the pom.xml file to address any vulnerabilities that popped up during our checks.

First on my list was getting those certificates set up correctly. This step is crucial because, with HTTPS in place, our users can confidently know they're interacting with our authentic platform and not some counterfeit site.

Then, I made sure our hashing function was working as it should. We used a checksum for validation. This ensures that the user data gets mixed up in a way that's secure and not easy for prying eyes to decipher.

Finally, we addressed and patched any security gaps we found. This effort assures us that our app is operating efficiently and securely.

To maintain our app’s security in the long run, it's vital to keep our software updated. This minimizes the risk of hackers taking advantage of outdated systems. Also, even though we're still working on it, implementing a system where users only get access based on what they actually need is a smart move for internal security.

## Industry Standard Best Practices

Secure coding standards are like guidelines we follow to dodge security pitfalls. By sticking to these standards, we can catch and fix issues before our software even goes live. The big deal about these standards? They're all about making sure our code is safe, reliable, and hard to hack into. For this particular software, I went with AES. Just to clarify, AES stands for Advanced Encryption Standard. It's a top-notch method to keep data encrypted and, honestly, it's a must-have in our toolkit.

References

Manico, J. (n.d.). *Iron-Clad java*. O’Reilly Online Learning. https://learning.oreilly.com/library/view/iron-clad-java/9780071835886/?sso\_link=yes&sso\_link\_from=SNHU

*Securing Spring Boot applications with SSL*. (2023, June 7). Securing Spring Boot Applications With SSL. https://spring.io/blog/2023/06/07/securing-spring-boot-applications-with-ssl

*The keytool Command*. (2022, January 6). https://docs.oracle.com/en/java/javase/16/docs/specs/man/keytool.html

*Transport Layer Security (TLS) Protocol Overview*. (n.d.). https://docs.oracle.com/javase/8/docs/technotes/guides/security/jsse/tls.html