****

# 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 5](#_Toc102040760)

[4. Secure Communications 5](#_Toc102040761)

[5. Secondary Testing 5](#_Toc102040762)

[6. Functional Testing 7](#_Toc102040763)

[7. Summary 8](#_Toc102040764)

[8. Industry Standard Best Practices 9](#_Toc102040765)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **23 Oct 2022** | **Jesse Carter** |  |

## 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

Jesse Carter

## Algorithm Cipher

The SHA-512 Algorithm Cipher is recommended for the purpose of the checksum verification. This is used to encrypt or hash data that requires protection.

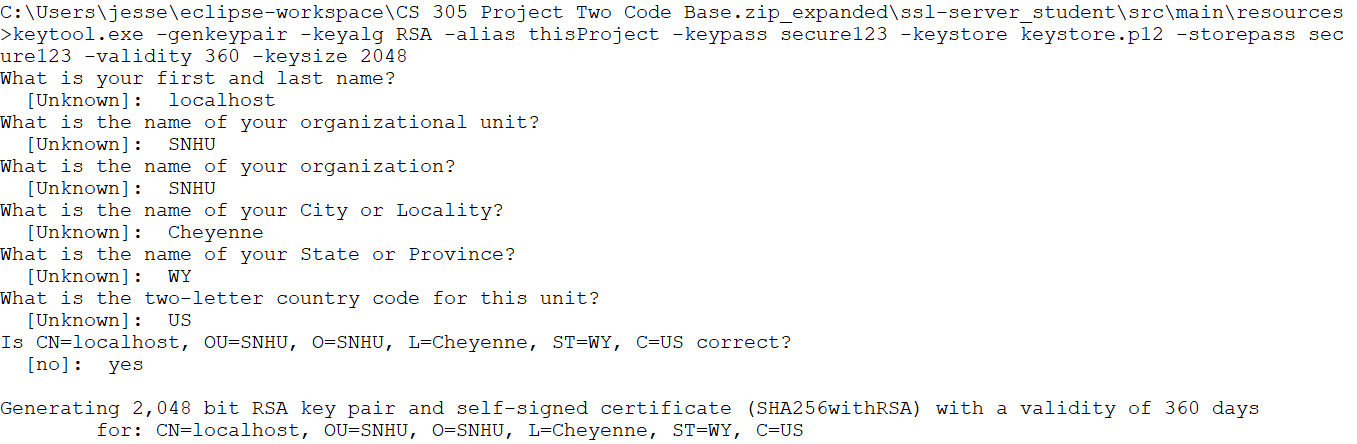
In this case, we are creating hashes from data which are analogous to a data fingerprint. If the data is manipulated in the slightest way, the corresponding hash produced will differ dramatically. The higher the bit level, the stronger the algorithm is in encrypting/hashing data. It is important when producing hashes that there is collision avoidance. Collision avoidance ensures that the corresponding hash is unique. If a piece of data produced an identical hash as a different piece of data, the algorithm would not be useful.

In cryptology, the use of random numbers is required to seed the generation of keys. Symmetric keys are used for encrypting and decrypting data. The key must remain protected to ensure the security of the encrypted data. Asymmetric keys are useful in verifying the sender. It requires two keys, one is private and remains with the sender, and the other is public so receivers can authenticate the sender.

Standards for algorithms have changed over decades, as computer hardware becomes increasingly more powerful rendering some algorithms obsolete. The NIST defines the current standards with very robust algorithms. However, it is only a matter of time and technology before new algorithms will be necessary.

## Certificate Generation

Insert a screenshot below of the CER file.

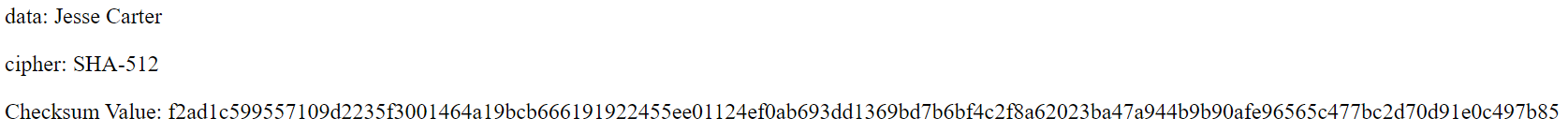


Text

Description automatically generated

## 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.

A picture containing graphical user interface

Description automatically generated

## Secondary Testing

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

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

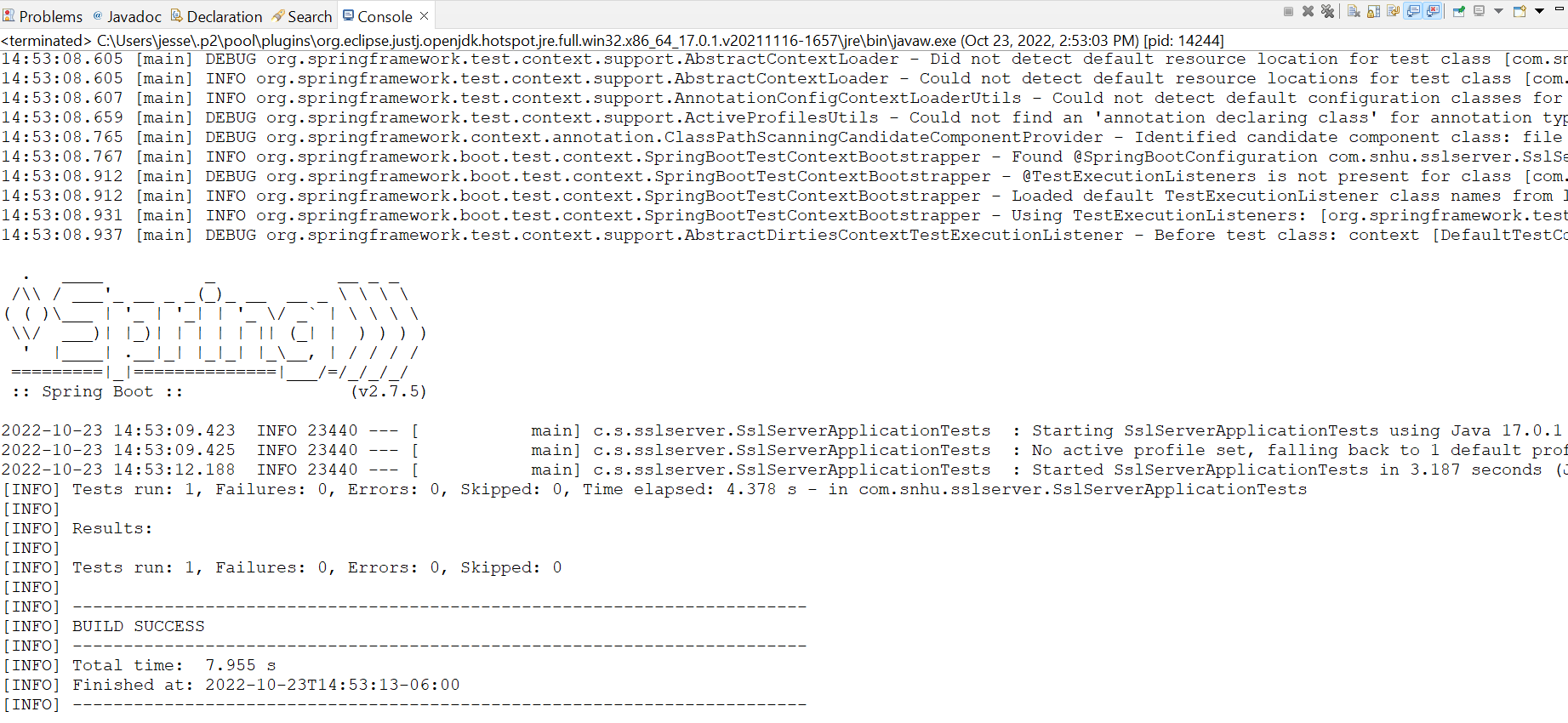


Graphical user interface, text, application

Description automatically generated

Graphical user interface, website

Description automatically generated



Graphical user interface, text, application, email

Description automatically generated

## Functional Testing

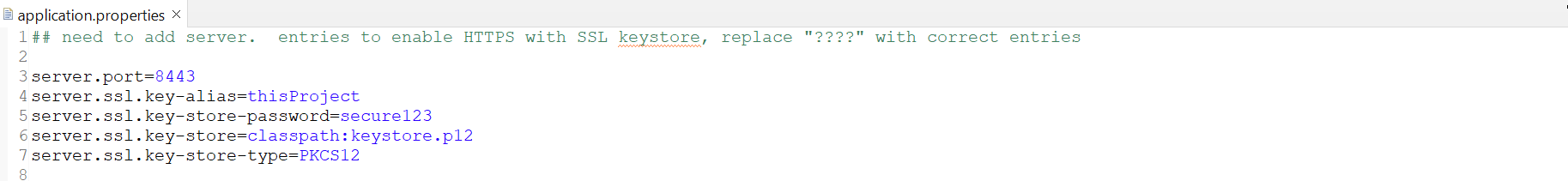
Insert a screenshot below of the refactored code executed without errors.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated



A picture containing scatter chart

Description automatically generated

## Summary

This build establishes an HTTPS protocol to facilitate secure data transmission. After running the static tests and reviewing the vulnerable dependencies, I updated Spring Boot from version 2.2.4.RELEASE to 2.7.5. This reduced the recorded vulnerabilities greatly. After analyzing the remaining vulnerabilities and determining the security concerns were not relevant to this build, I created a suppression file. The Assessment Flow Diagram drew my attention to Data Access and the API. The vulnerabilities were related to passing unparsed input, which could result in DoS attacks. As this build does not currently receive input outside of the system, each vulnerability was suppressed. As noted, the Spring Boot version was updated.

The layers of security added began with the establishment of the code to provide a secure HTTPS connection. Once the program ran, and no security issues were presenting, a static test was performed to identify vulnerable dependencies. Analysis led to fixing or suppressing the highlights as applicable. A final inspection of all details and reports was conducted before determining the build secure.

## Industry Standard Best Practices

NIST’s security guidance (FIPS 140-2) was reviewed for utilizing a strong encryption standard algorithm. Further, OWASP dependency checks were used throughout the development of the program. This was used in conjunction with reviewing the Vulnerability Assessment Process Flow to determine the areas of potential vulnerability. Applying these best practices, integrates robust SecDevOps processes across all development environments. Incorporating this level of perspective encourages security-minded environments that permit developers to create robustly secure and reliable systems.