

# Practices for Secure Software Report

Table of Contents

[Document Revision History 3](#_Toc102040754)

[Client 3](#_Toc102040755)

[Developer 4](#_Toc102040757)

[1. Algorithm Cipher 4](#_Toc102040758)

[2. Certificate Generation](#_Toc102040759) 5

[3. Deploy Cipher](#_Toc102040760) 6

[4. Secure Communications](#_Toc102040761) 6

[5. Secondary Testing](#_Toc102040762) 7

[6. Functional Testing](#_Toc102040763) 9

[7. Summary](#_Toc102040764) 10

[8. Industry Standard Best Practices](#_Toc102040765) 11

**References…………………………………………………………………………………….....12**

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **August 12, 2024** | **Damean Murphy-Short** | **Origin of the Document** |

## Client



## Developer

Damean Murphy-Short

## Algorithm Cipher

For the purposes of a checksum, a fixed length output algorithm is often desireable, as this makes the output more easily compared by the human eye and unlikely to reveal the length of the original input. These algorithms are referred to as message digest algorithms, the most popular being MD5 and different versions of Secure Hash Algorithm(SHA). Importantly, they provide collision resistance so that it is unlikely that any modified version of the input will produce the same output. I recommend SHA-256, an industry standard, for the use in verifying proper, untampered distribution of Artemis Financial’s data.

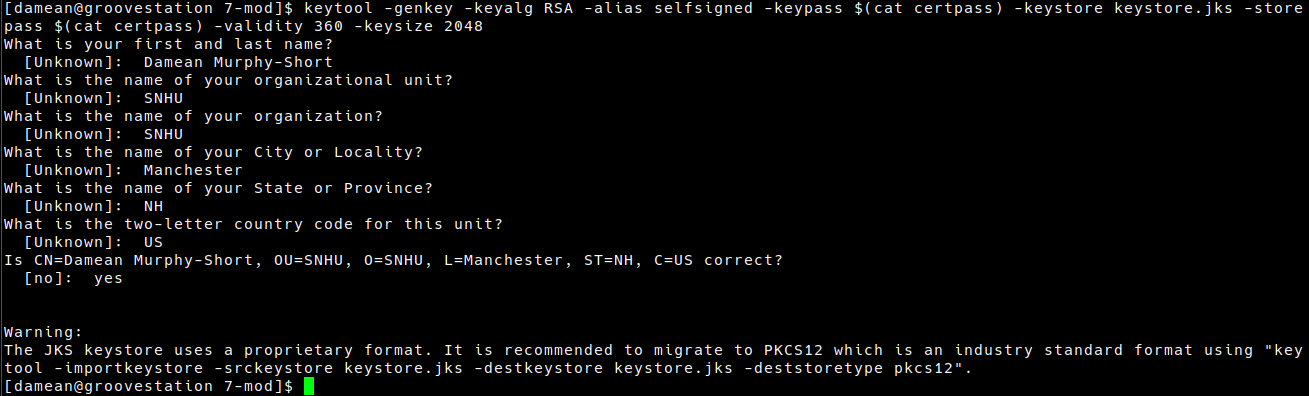
As stated above, SHA-256 takes an input of any length and provides a fixed length resultant ‘hash.’ This hash is irreversible, so it would not be possible for an attacker to know the original input or create a malicious input without implausible time and effort that would result in the same hash. This is the other beneftit of resisting collisions besides revealing small changes in the original message: the output is not easily predicted until the hash is generated, making attacking a particular hash almost impossible.

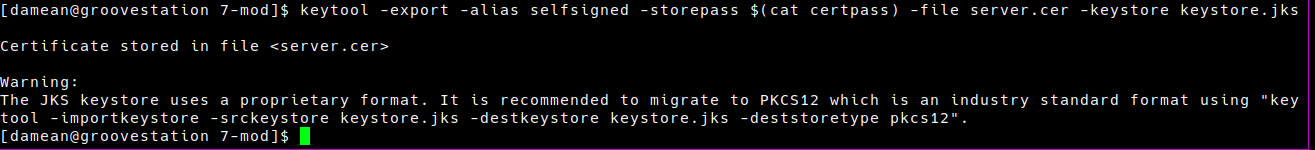
The reasoning for using the 256-bit algorithm over the 512-bit producing algorithm is this: it is not currently feasible to produce an artificial collsion with the 256-bit output, while it also takes less memory and computing power to calculare the 256-bit hash. Likewise, MD5 was not chosen, because it is less resistant to collision, and with current technology it is feasible to produce a collision.

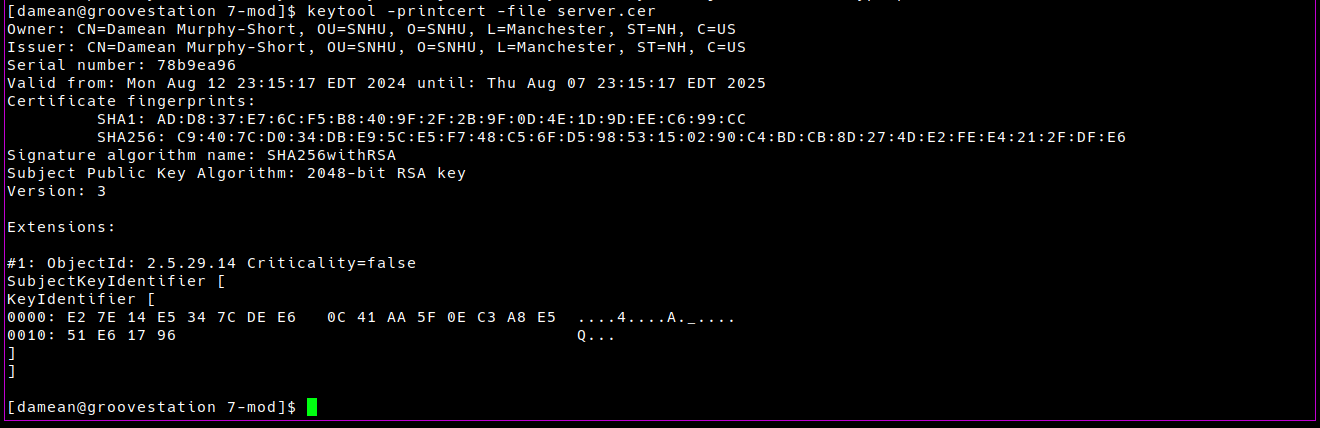
For the secure connection between an SSL server and a client, a reversible encryption is needed, such as that provided by public-private key pairs. The RSA encryption algorithm is an industry standard algorithm that is suitable for this purpose, being unfeasible to ‘crack’ (Cobb, M., 2021), and providing a way for the server and client to each verify the identity of the other and communicate without sending plain text messages, which could be read by any connection that is between them. This type of encryption uses the ‘private’ key to write messages, which can only be decrypted with the ‘public’ key, which is shared with the intended recipient.

## Certificate Generation

Below the creation of and result of the RSA self-signed certificate is shown.

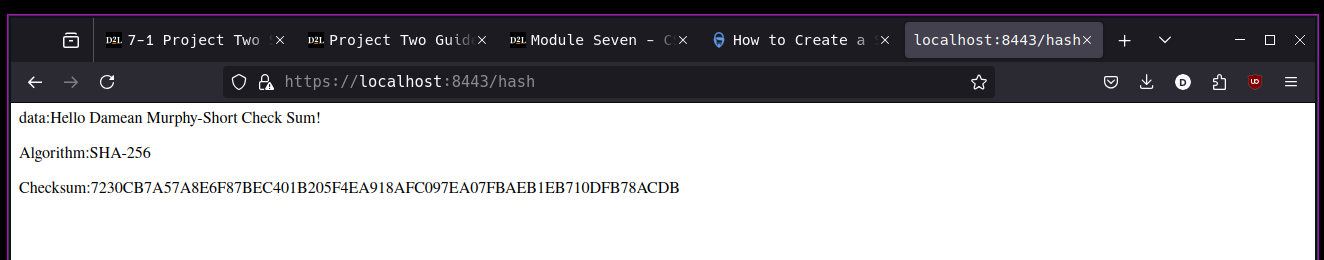


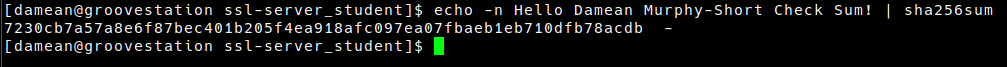




## Deploy Cipher

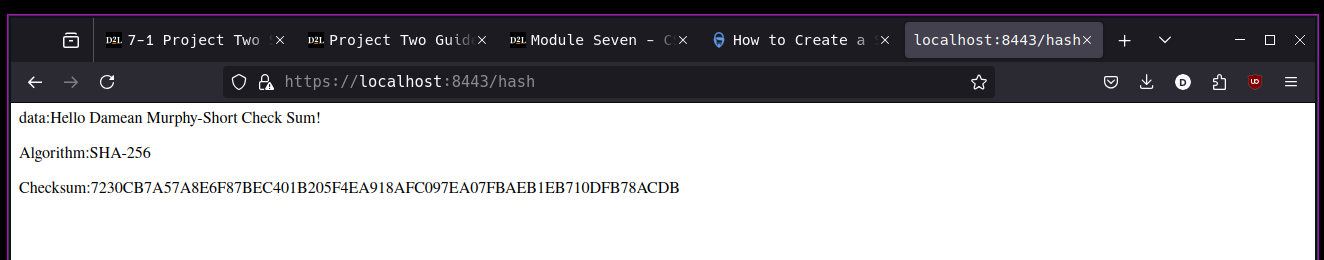
Pictured below is the web server displaying the checksum of the example data with a verification from the command line using the same algorithm.





## Secure Communications

Keen readers will recognize this image of the web server. Note that it also displays an HTTPS connection established with the client browser.



## Secondary Testing

In the image shown, the refactored code is presented alongside the command line that is running the web server, executed without error.

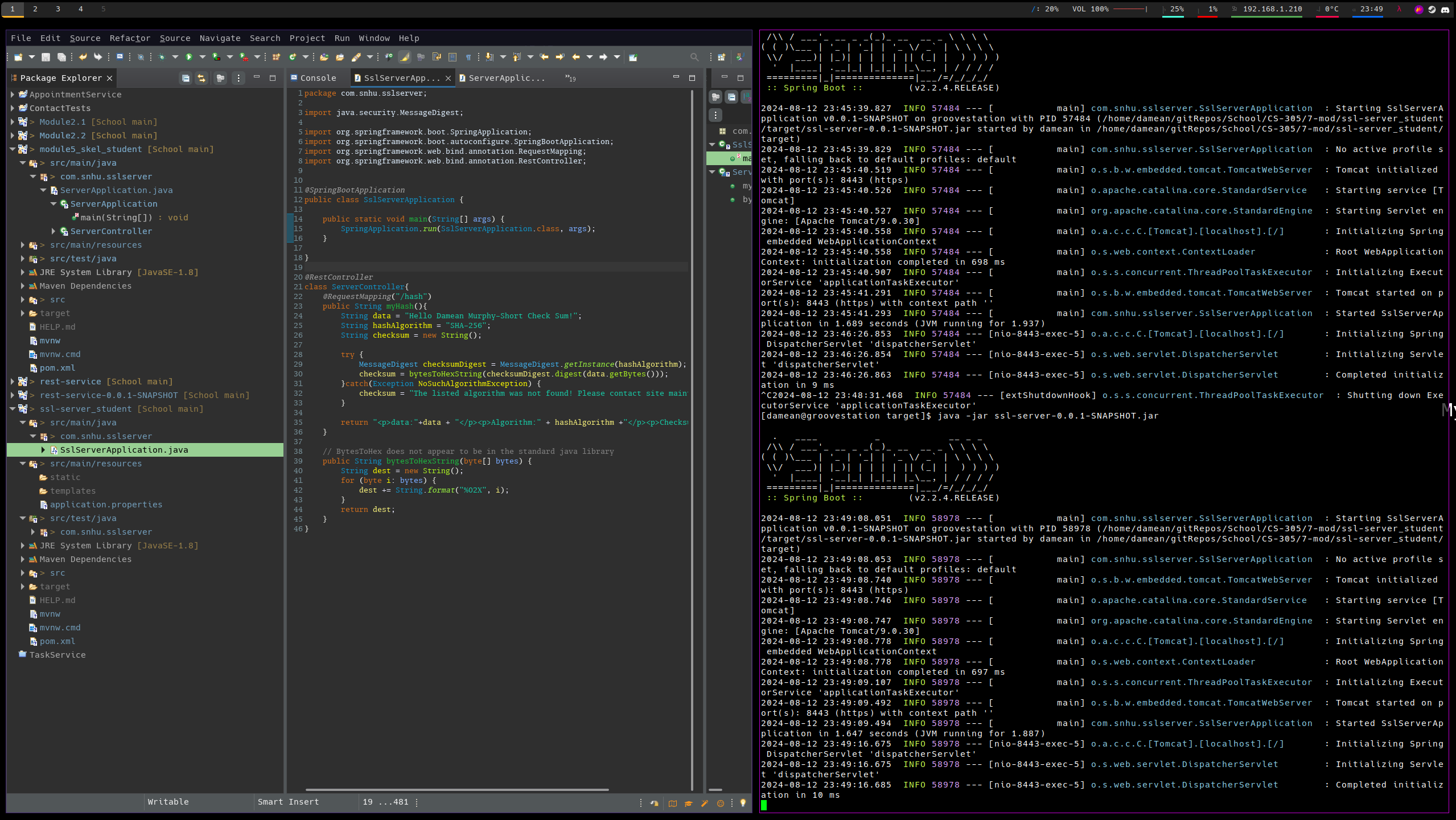
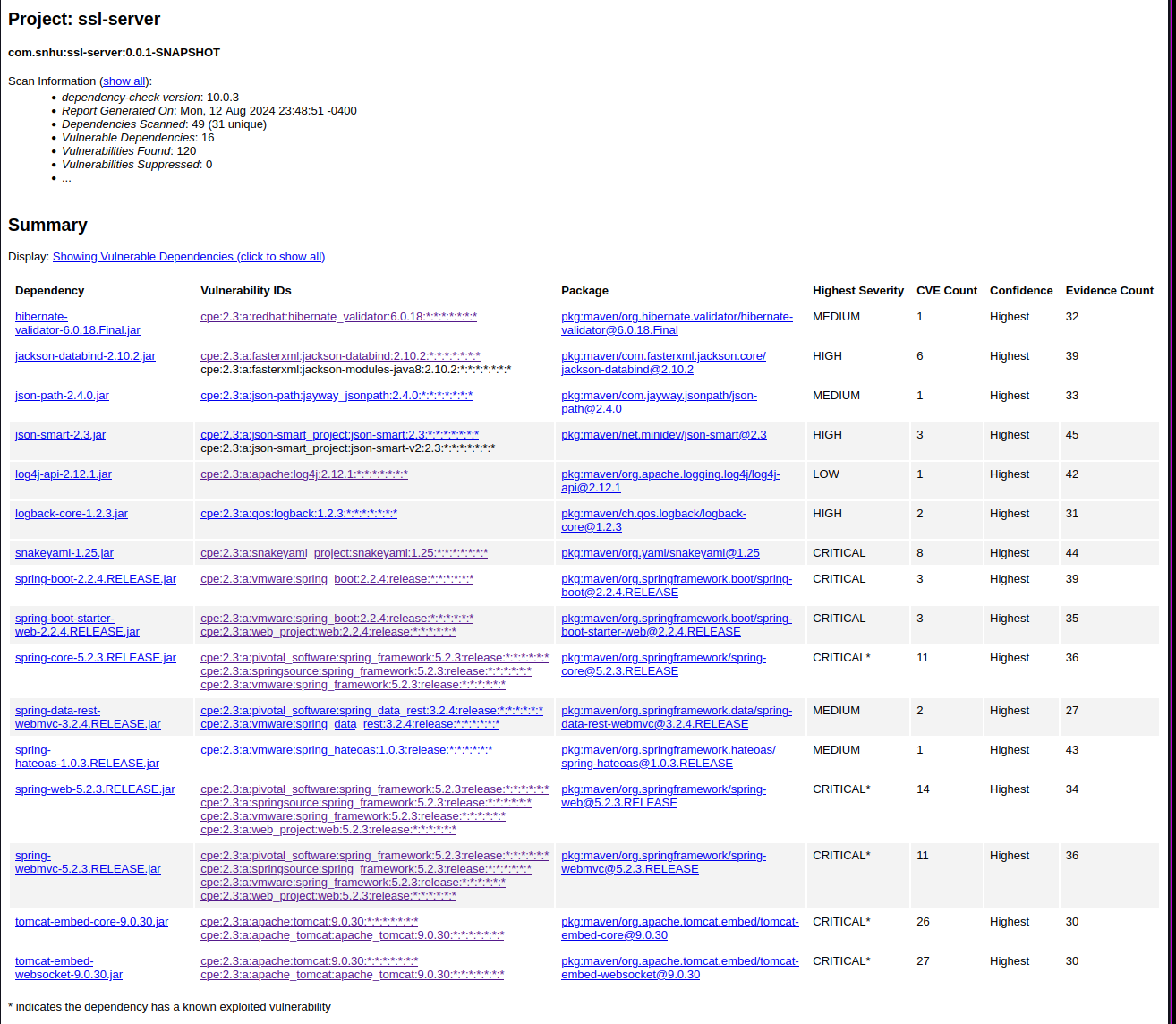


Image 1 shows the dependency check before modification.

  
  
Image 2 depicts the dependency check results after modification.

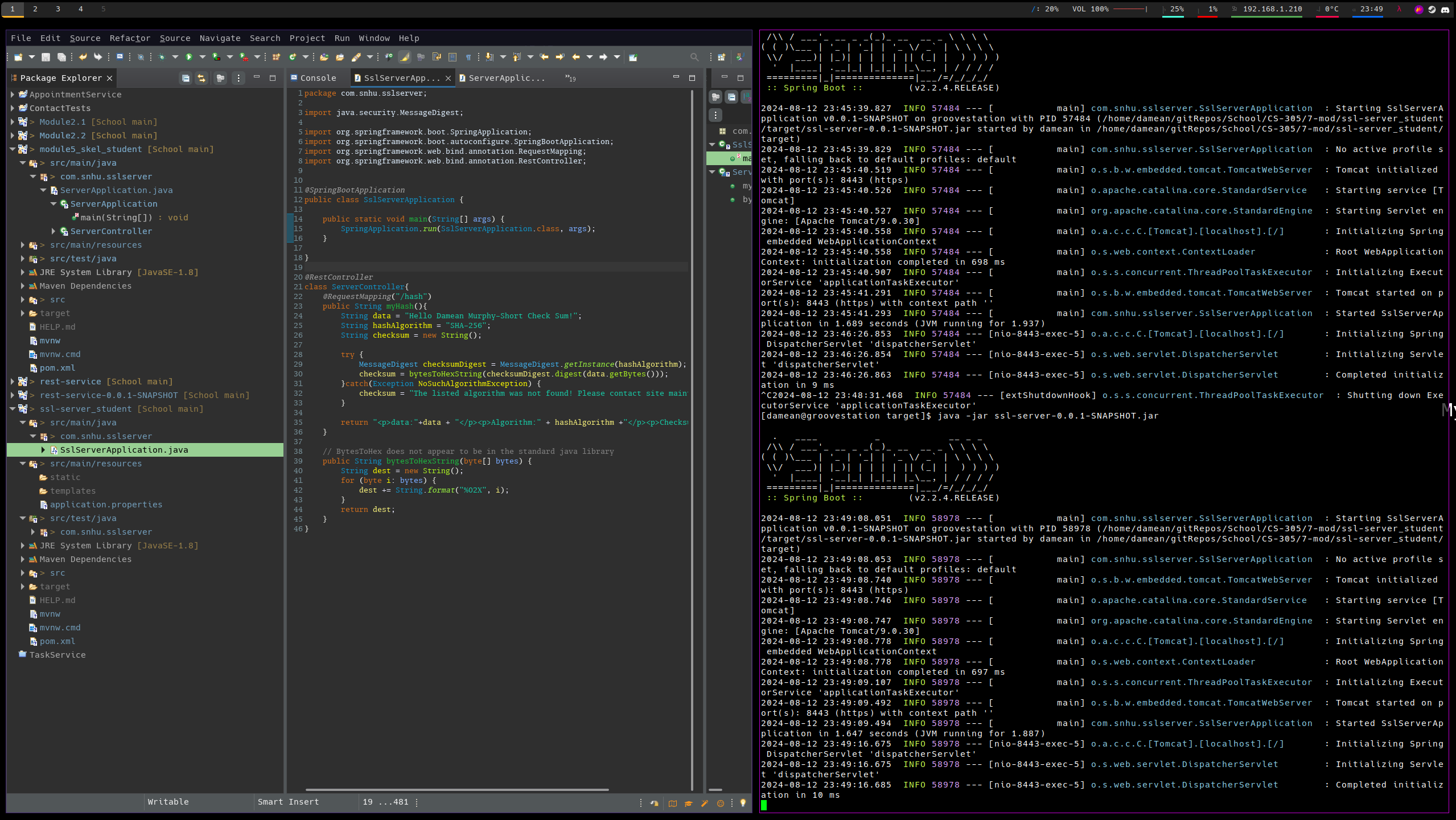


As can be seen, there are no discrepencies between the vulnerabilities found.

## Functional Testing

The refactored code is quite simple and compact, leaving only a small surface for attack. There is no user input accepted, and little is revealed about the site’s internal structure. The most vulnerable component of the system is the certificate, whose relative directory is revealed in the application.properties folder, along with the password, which is not strong for industry standards (“123456”). During production, I recommend using a stronger password for the certificate and placing the certificate and keystore in a specified directory that is in the same root folder as the server executable.

Again, a screenshot of the refactored code executed without errors.



## Summary

The changes made to the code are as follows: the application.properties file was modified to contain the relevant SSL certificate information, and the SSLServerApplication class was modified to contain the hash mapping for the server, which returns the requested checksum along with the original data. These changes address how secure the connection between the client and server is by utilizing cryptography. Important to note is that these modifications address different issues and may operate separately from one another. For instance, we could implement an insecure connection that provides a checksum of the data to ensure that it is not changed when in transit, however by adding the SSL layer of security to the connection, we ensure that the data is also not read during transit in plain text.

## Industry Standard Best Practices

Web connections by HTTPS rather than HTTP are standard due to the increased security afforded by encrypted communications. Establishing this type of connection ensures that messages between the client and server are only readable at each endpoint (“What is HTTPS?”, n.d).

Checksums also provide assurance to the client that the requested data has not been tampered with since leaving the server. This is because there is a near-impossible chance that data could be altered and still purposely produce the same hash (VanData Consulting, 2023).

Failure to adhere to these practices would be damaging to Artemis Financial’s reputation and business interests. Many browsers will warn against connections to websites that are not using encryption. Artemis Financial also handles sensitive client data, where securely encrypted data is necessary for transit in order to prevent any connections in between each endpoint from reading the data in plain text. This is not only a moral imperative for the company, but a legal one (“FTC safeguards rule…”, 2023). On top of this, checksums provide a peace of mind to the client by ensuring that all data being transmitted is received successfully without modification.

References.

Cobb, M. (2021, November 4). *RSA algorithm (Rivest-Shamir-Adleman)*. Security. https://www.techtarget.com/searchsecurity/definition/RSA

*FTC safeguards rule: What your business needs to know*. (2023, October 6). Federal Trade Commission. <https://www.ftc.gov/business-guidance/resources/ftc-safeguards-rule-what-your-business-needs-know>

VanData Consulting. (2023, October 24). *Ensuring data integrity: the vital role of Checksums*. https://www.linkedin.com/pulse/ensuring-data-integrity-vital-role-checksums-vandata-njs1c/

*What is HTTPS?* (n.d.). CloudFlare. Retrieved August 13, 2024, from https://www.cloudflare.com/learning/ssl/what-is-https/