****

# 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** | **October 15, 2022** | **Dino Suljic** | **N/A** |

## 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

Dino Suljic

## Algorithm Cipher

Artemis Financial is a financial-based company that handles sensitive data. There are many secure hashing algorithms that could be used, but the most recommended choice for security would be SHA-256, which is the same hashing algorithm that the United States government uses. SHA-256 has become an industry standard, and to date, not a single vulnerability has been discovered in the SHA algorithm.

The SHA algorithm can take any data, including plaintext or bytes, and output a fixed 64-character checksum with minimum collisions. Collisions are extremely unlikely with SHA-256, including compared to alternative hashing algorithms such as MD-5. The reason the output is 64 characters long is because we are using the 256-bit version of the algorithm, which holds up to 64 characters. This means slower computing times but less collisions, and the algorithm is less likely to be brute forced as well as the string it outputs. If Artemis Financial needed performance and speed over security, then I would recommend the MD-5 algorithm, but it is not applicable in this case since sensitive data is being handled.

Moreover, with SHA-256, the integrity of the data is very well preserved. This is because even the smallest change in the input data will produce an entirely different and unique checksum. Given all of this, SHA-256 would be a solid choice for producing checksums of passwords and sensitive data.

On the topic of encryption, such as for Artemis Finacial's archive files, I would recommend AES-256. AES is an acronym that stands for Advanced Encryption Standard. AES is also a government standard, and it is a solid encryption algorithm that Artemis Financial can use to encrypt files. AES uses symmetric encryption, which means that there is a single secret key (as opposed to asymmetric encryption that has both a public key and a private key). Symmetric encryption has the benefit of being faster and more convenient, but at the cost of slightly less security. With AES, Artemis Financial will need to ensure that all communications are secure so that the secret key cannot be intercepted in transit.

**To break it down simply:**

**SHA-256 –** Recommended hashing algorithm to generate secure checksums of passwords and sensitive data with minimal file collisions.

**AES-256 –** Recommended encryption algorithm to encrypt and decrypt files and data with.

**Works Cited**

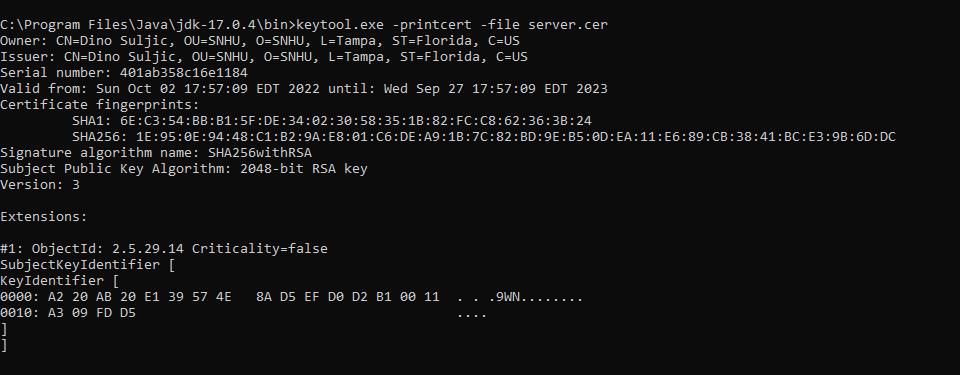
Donico (n.d.). Retrieved October 15, 2022, from <https://www.doncio.navy.mil/chips/ArticleDetails.aspx?ID=7375>

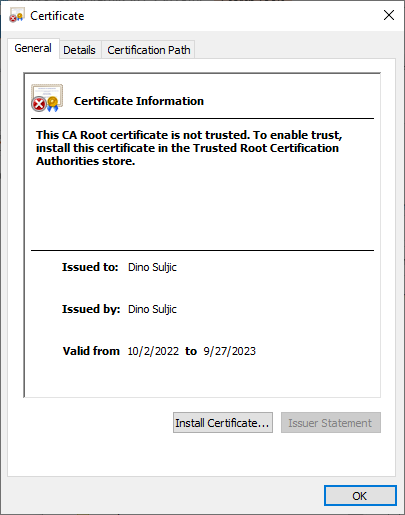
Callaghan, P. (n.d.). *Why you should use SHA-256 in evidence authentication*. Pagefreezer Blog. Retrieved October 15, 2022, from <https://blog.pagefreezer.com/sha-256-benefits-evidence-authentication>

## Certificate Generation

Insert a screenshot below of the CER file.

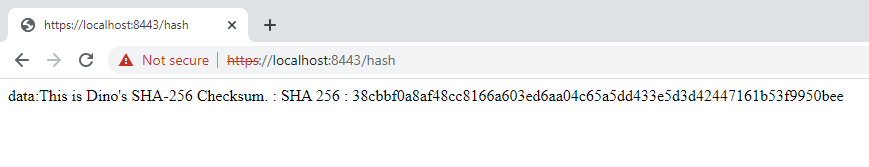
This certificate is valid until September 27, 2023.





## Deploy Cipher

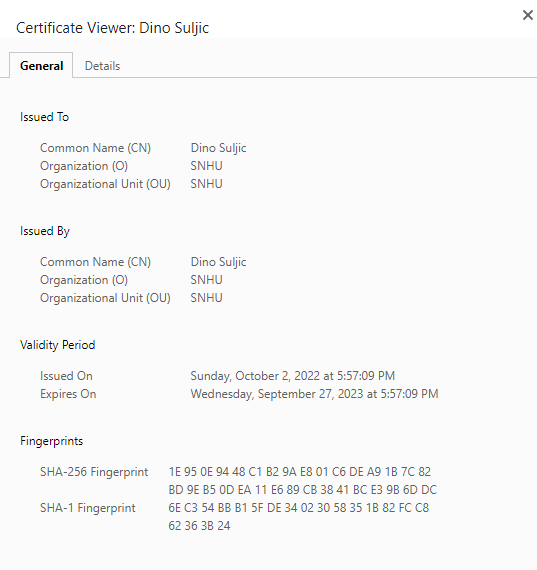
Insert a screenshot below of the checksum verification.

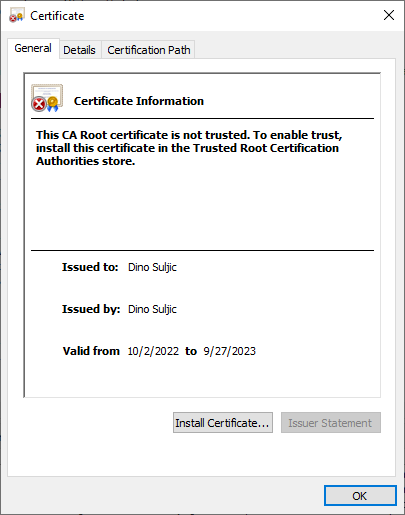


## Secure Communications

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

The certificate is working, and it is the certificate the Spring server is using. However, because the certificate is self-signed and not an official and trusted certificate authority, Chrome is showing it as insecure.

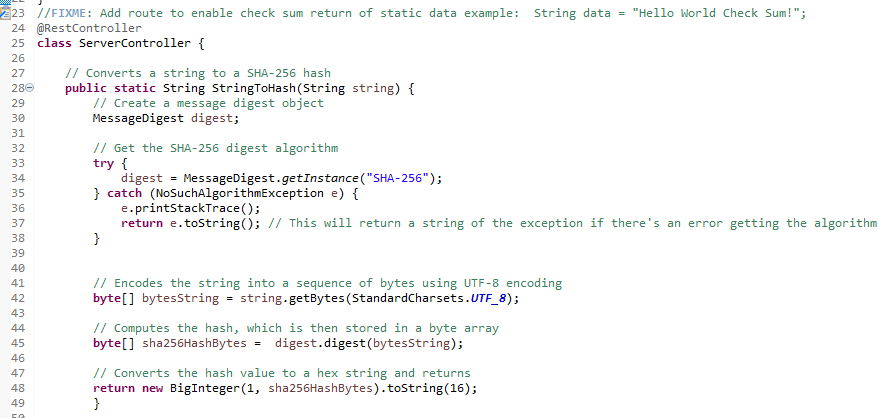


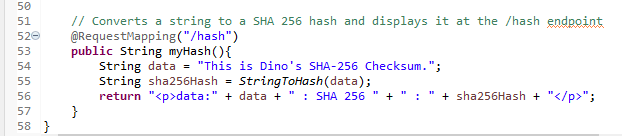


## Secondary Testing

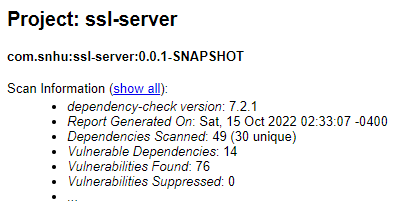
Insert screenshots below of the refactored code executed without errors and the dependency-check report.

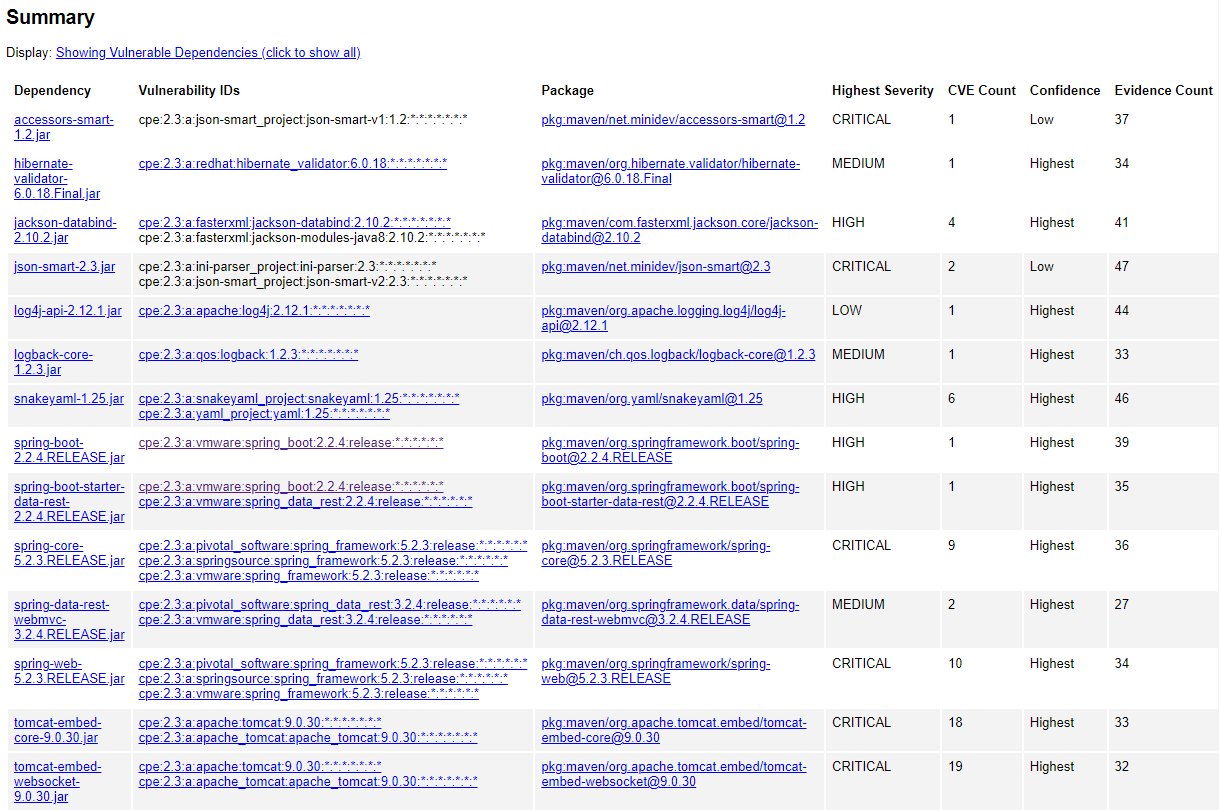
**The Refactored Code without errors**



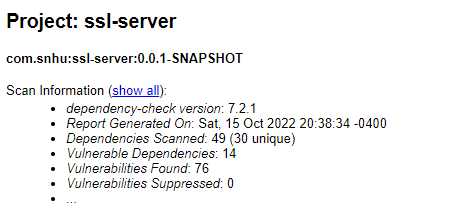


**Dependency Check Screenshot Before Code Refactoring**





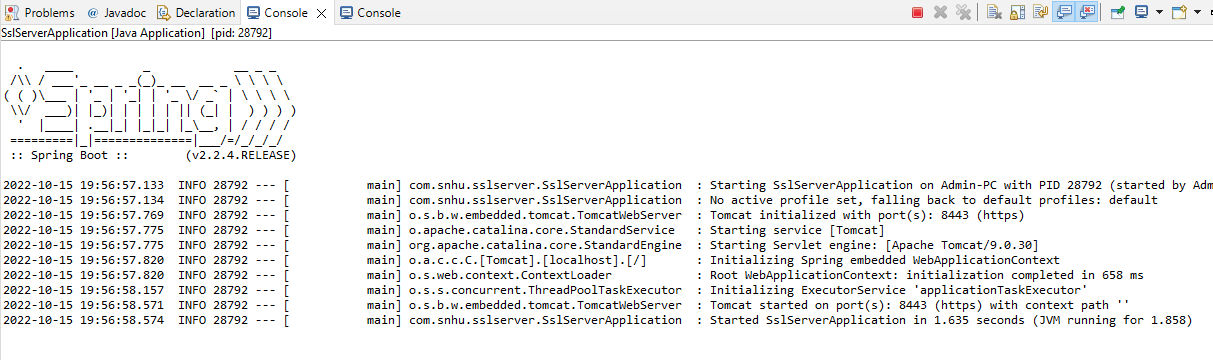
**Dependency Check Screenshot After Code Refactoring – No additional vulnerabilities have been introduced!**



## Functional Testing

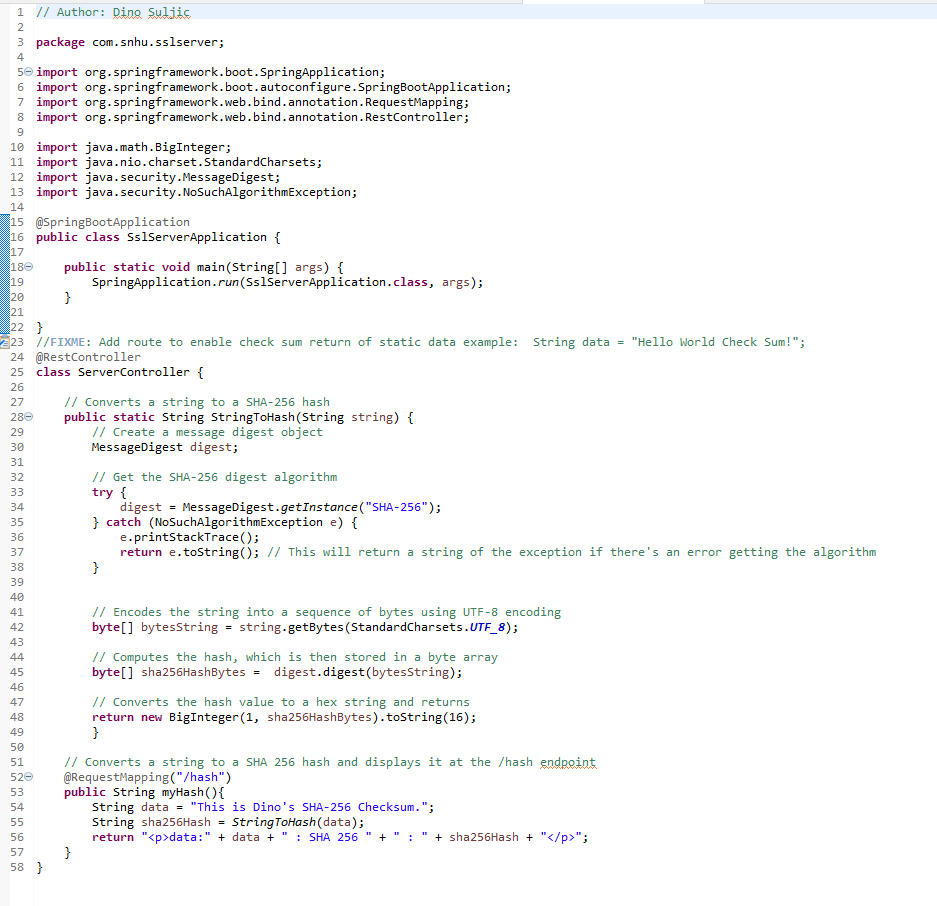
Insert a screenshot below of the refactored code executed without errors.

**Console of the Code Executed and Running without errors:**





**The Full Code:**



## Summary

The code has been refactored and now has a secured Rest Controller that uses our self-generated and self-signed certificate. The website uses HTTPS instead of HTTP now, which will help against man-in-the-middle attacks. The code also now contains a function that converts a string to a SHA-256 hash. Future developers of Artemis Financial can use this hashing function to generate checksums for files or for sensitive data if required, such as user passwords before they are stored in the database.

Unused dependencies and libraries have been removed, and the **pom.xml** file has been updated to use the latest dependency check version, which is **7.2.1.** Artemis Financial should always make sure to update the software to use the latest version of dependency check, as there could be vulnerabilities that aren’t discovered in the codebase because it’s using an old CVE database. It also would be a solid idea for Artemis Finacial to do at least a few scans a month by running Maven install with dependency check in the plugins list.

Moreover, Artemis Financial should make sure to always be using the latest version of the Spring framework. Artemis Finacial uses the 2.2.4.RELEASE version of the Spring framework. There are newer versions that contain less vulnerabilities and improvement fixes.

Other than that, the code has been carefully audited and refactored to make sure that it is clean and performing well without errors.

**Identified and improved security areas:**

**Secure Communications** and **Client/Server –** The application has secure communications now by requiring HTTPS with a self-signed certificate.

**Code Quality –** The code is robust and executes without errors or bugs. Unused dependencies have been removed.

**Code Error –** The application catches and throws exceptions. Errors will not break the application.

**Cryptography –** The application uses HTTPs now with a self-signed certificate and uses cryptographic protocols to generate a SHA-256 checksum.

**Encapsulation –** Data is encapsulated from both the user and developer using classes and other means of abstraction, such as the function that generates a SHA-256 hash.

## Industry Standard Best Practices

The code follows industry standard best practices and Java naming conventions. First, I am not importing or using any libraries that the program doesn’t use. It contains only the essentials. This helps increase the overall security of the program by reducing the attack vector. The less dependencies that are imported, the less likely it is that the application gets hacked.

Next, I made sure to use decent and legible variable names. For instance, instead of using vague function names such as **Hash()**, I named the function **StringToHash()** in order to indicate that the function takes a string and returns a hash. Similarly, I named the message digest variable “digest” instead of something more vague like “d”. This also helps improve the security of the overall application so that other developers don’t actually misunderstand what a variable or function does. Most security vulnerabilities occur due to accidental human error, and the best we can do is minimize it.

Next, I also make sure to catch exceptions and print an error message, rather than letting the error “break” the application. All in all, I made sure to follow best industry practices and Java naming conventions. The application is secure.

As for the value of applying industry standard best practices, it is essential if we want to keep our software and users safe. We have to make sure that all consumer data is safeguarded with precautions and security controls. Otherwise, we may be violating government regulations such as the GDPR that require data to be kept secure. Morally and ethically, we also wouldn’t want anyone be careless with our password and financial information, would we?