

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[Document Revision HistoryPAGEREF \_Toc33111302 \h3](#_Toc33111302)3

[ClientPAGEREF \_Toc33111303 \h3](#_Toc33111303)3

[InstructionsPAGEREF \_Toc33111304 \h3](#_Toc33111304)3

[DeveloperPAGEREF \_Toc33111305 \h4](#_Toc33111305)4

[1. Algorithm CipherPAGEREF \_Toc33111306 \h4](#_Toc33111306)4

[2. Certificate GenerationPAGEREF \_Toc33111307 \h4](#_Toc33111307)4

[3. Deploy CipherPAGEREF \_Toc33111308 \h4](#_Toc33111308)4

[4. Secure CommunicationsPAGEREF \_Toc33111309 \h5](#_Toc33111309)5

[5. Secondary TestingPAGEREF \_Toc33111310 \h6](#_Toc33111310)6

[6. Functional TestingPAGEREF \_Toc33111311 \h7](#_Toc33111311)6

[7. SummaryPAGEREF \_Toc33111312 \h7](#_Toc33111312)7

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **October 13** | **Benito Patino** | **Research algorithm cypher** |

## Client



## Instructions

Deliver this completed Practices for Secure Software Report documenting your process for writing secure communications and refactoring code that complies with software security testing protocols.

Respond to the steps outlined below and replace the bracketed text with your findings in your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

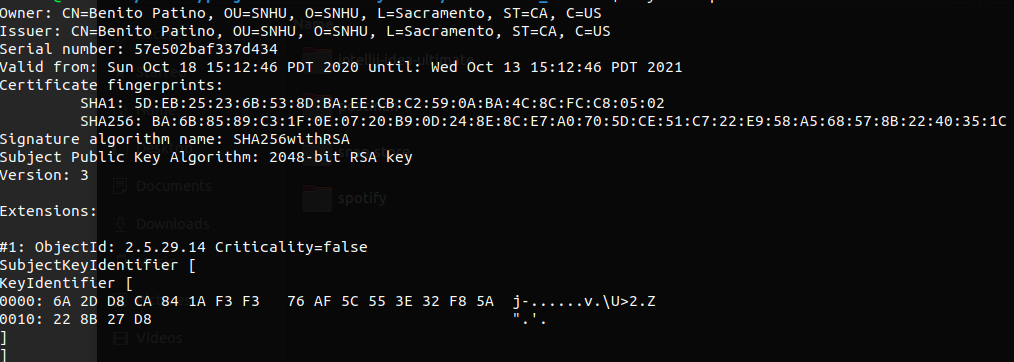
Benito Patino

## 1. Algorithm Cipher

Using the SHA-256 algorithm cipher, is the appropriate option for implementing data verification in the form of a checksum. SHA-256 or SHA-2 belongs to the Secure Hash Algorithm family of cryptographic functions. Originally designed by the United States National Security Agency, SHA-2 provides six different hash functions with hash values such as: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256 (Wikipedia, 2020). Numbers such as “256” or “512” represent the size of the hash value output in bits. Regarding security, SHA-256 are prone to *length extension attacks.* Yet, collisions are difficult to create. SHA-256 low collision probability makes it the ideal algorithm.

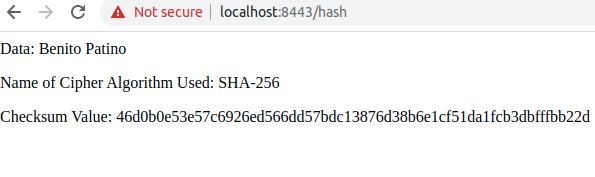
## 2. Certificate Generation

Generate appropriate self-signed certificates using the Java Keytool, which is used through the command line.



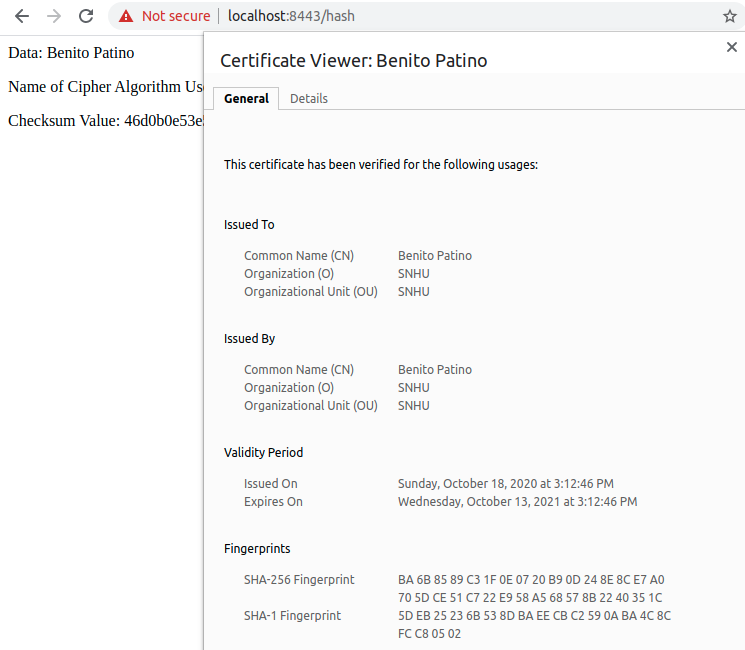
## 3. Deploy Cipher

Refactor the code and use security libraries to deploy and implement the encryption algorithm cipher to the software application. Verify this additional functionality with a checksum.

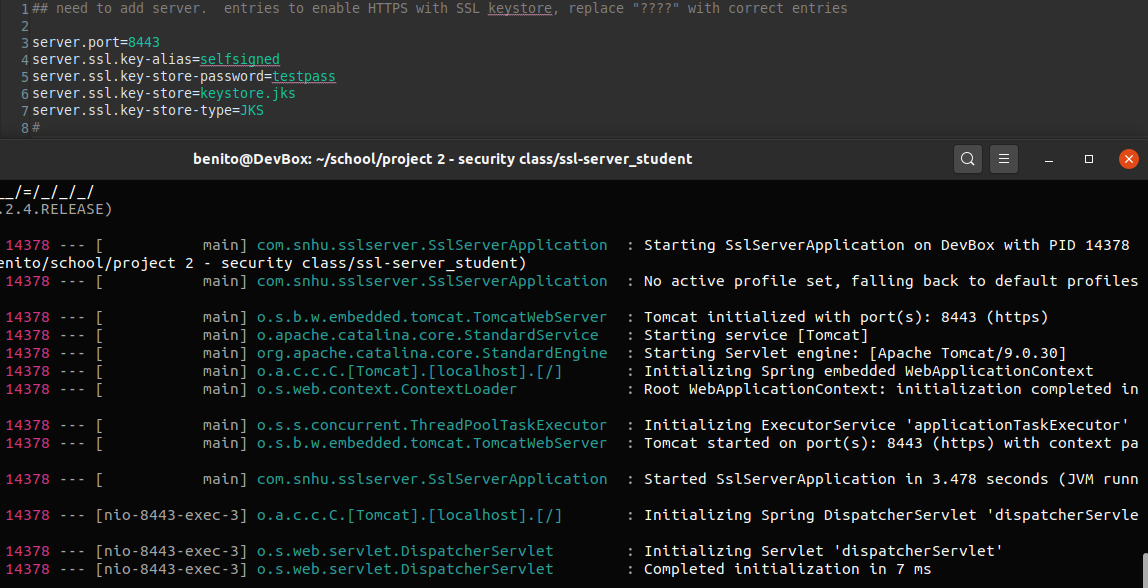


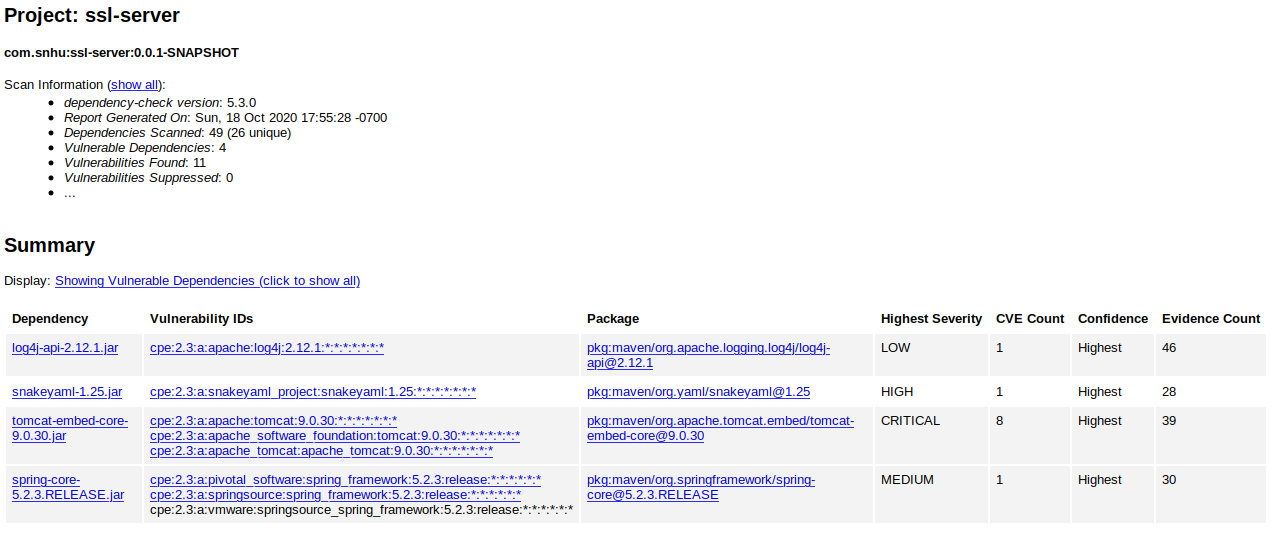
## 4. Secure Communications

Refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code to verify secure communication by typing **https://localhost:8443/hash** in a new browser window to demonstrate that the secure communication works successfully.



## 5. Secondary Testing

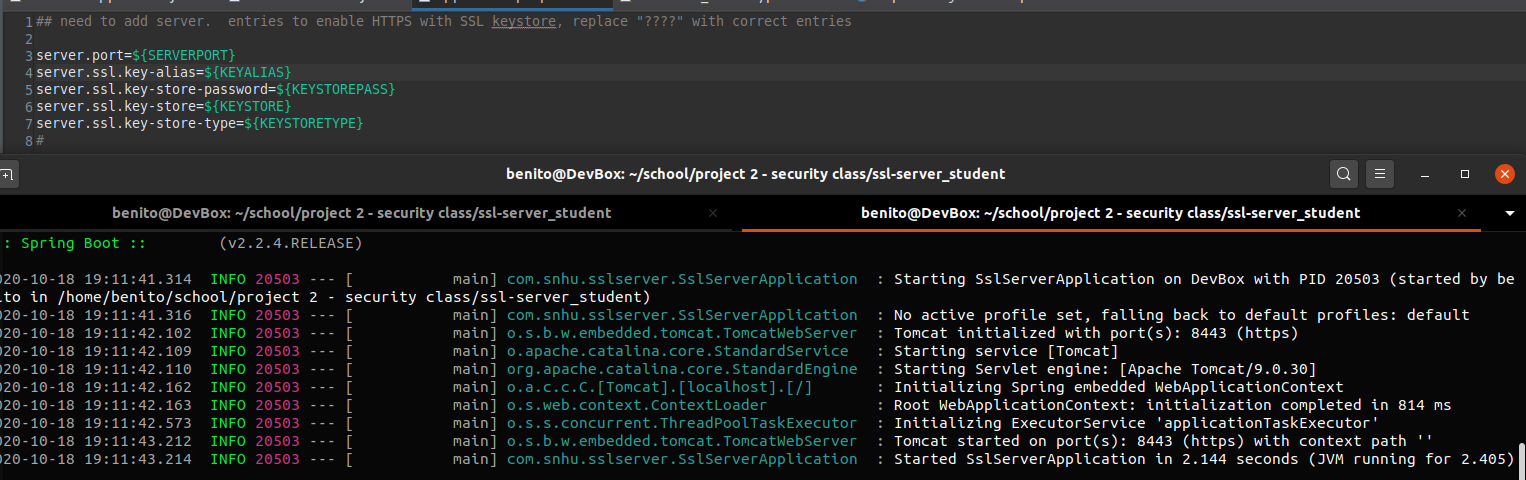




## 6. Functional Testing

Critical configuration information, like the key store password

was stored in an environment variable. Other configuration information like the key store name and server port were also stored in environment variables.



## 7. Summary

Refactoring the code, we addressed different areas of security. First, we enabled secure API interactions by implementing HTTPS communication. Second, in order to implement our algorithm cipher, we needed to make sure secure error handling was in place handling exceptions. We also make use if secure coding practices by storing important configuration information in environment variables, rather than in plain text in the code base.

We added different layers of security by following a simple process. First, we ran the application and called the end API endpoints. After calling the API endpoints, we compared the expected and actual data returned by the API. Then, we performed a code review of the code base. We focused on sensitive and important portions of the code base like the application properties. Within the application properties, we found that the key store password was stored in plain text within the code base. To mitigate this vulnerability, we stored the password and other configuration data in environment variables.

For incorporating the file verification function, we first had to research an appropriate algorithm cipher. After much research, we found the Secure Hash Algorithm 256 bit cipher is the best choice for our purpose. We created an endpoint within the *ServerController,* which returns a checksum value. The endpoint will take in data and will return a checksum value – generated from the SHA-256 hash function.

It is important to abide by best practices for ensuring the the current and future security of the application. First, there must be a focus on protecting important sensitive information. Protecting sensitive information will maintain consumer confidence will help avoid any catastrophic financial losses. It is also important to abide by government regulations and industry standards. For example, handing credit card information requires an organization to follow the Payment Card Industry Data Security Standard. It was developed to protect sensitive customer data. Finally, it is important to educate other users of the system on secure practices and understanding the the negative impacts of security breaches.