****

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[Document Revision History 3](#_Toc33111302)

[Client 3](#_Toc33111303)

[Instructions 3](#_Toc33111304)

[Developer 4](#_Toc33111305)

[1. Algorithm Cipher 4](#_Toc33111306)

[2. Certificate Generation 4](#_Toc33111307)

[3. Deploy Cipher 4](#_Toc33111308)

[4. Secure Communications 4](#_Toc33111309)

[5. Secondary Testing 4](#_Toc33111310)

[6. Functional Testing 5](#_Toc33111311)

[7. Summary 5](#_Toc33111312)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **04/13/2022** | **Andrej Oljaca** |  |

## 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

Andrej Oljaca

## 1. Algorithm Cipher

Determine an appropriate encryption algorithm cipher to deploy given the security vulnerabilities, justifying your reasoning. Be sure to address the following:

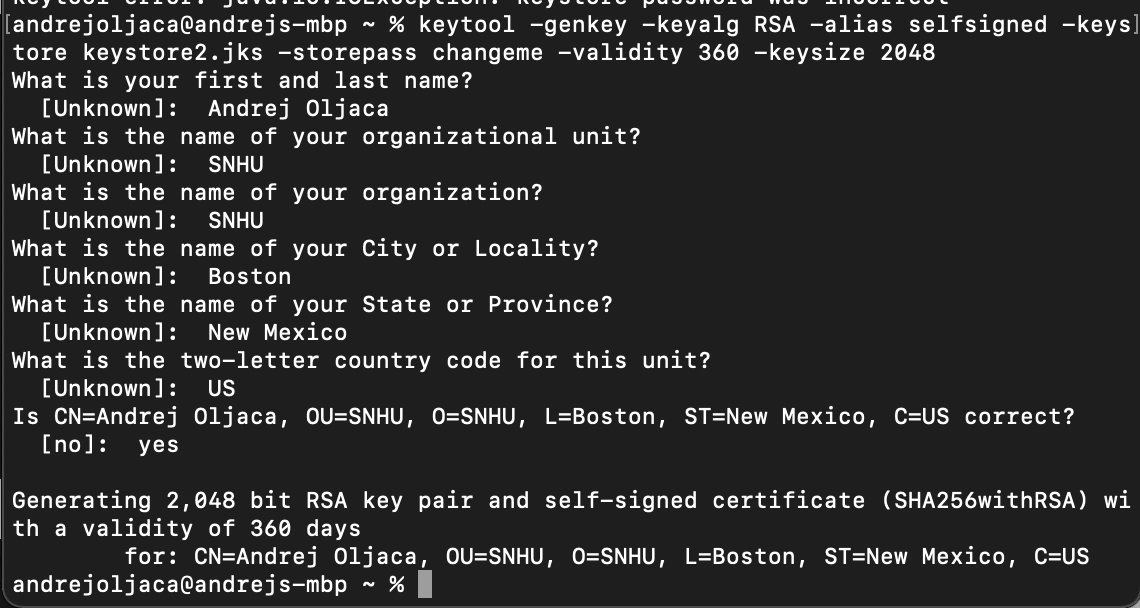
* Provide a brief, high-level overview of the encryption algorithm cipher.
  + I would recommend the SHA-256 encryptions algorithm cipher. SHA-256 is one of the popular hash algorithms. It can be used in digital signature verification, password hashing, SSL handshake, and integrity checks (Simplilearn, 2022).
* Discuss the hash functions and bit levels of the cipher.
  + SHA-256 has a digest size of 256 bits. The first step in the SHA-256 algorithm is padding bits. The second is padding length. The third is initializing the buffers. The fourth is compression functions. And the fifth is the output (Simplilearn, 2022).
* Explain the use of random numbers, symmetric vs non-symmetric keys, and so on.
  + Symmetric encryption is different from non-symmetric encryption because it uses the same key to encrypt and decrypt data. Asymmetric encryption requires two keys to work. A public key that is made public is used to encrypt the data. A private key is then used to decrypt the data. (Team, 2021).
* Describe the history and current state of encryption algorithms.
  + The first evidence of encryption dates back to 1900 BC in Egypt. In the 16th century, a person by the name of Vigenere created a cipher which was the first cipher that used an encryption key. Up to the second world war, cryptography was mostly used for military purposes (Sidhpurwala, 2019). In today's time, roughly 44% of organizations use encryption to secure their data and another 30% are using encryption in some way. The US is the country most using encryption with Australia right behind.

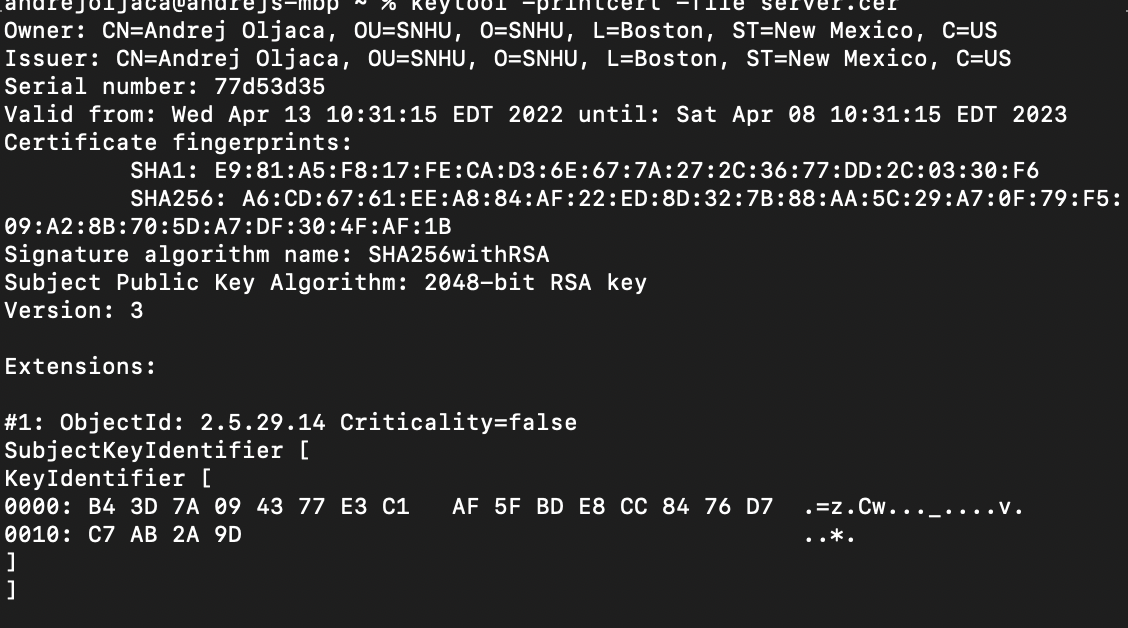
## 2. Certificate Generation

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

* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.

[Insert screenshot(s) here.]

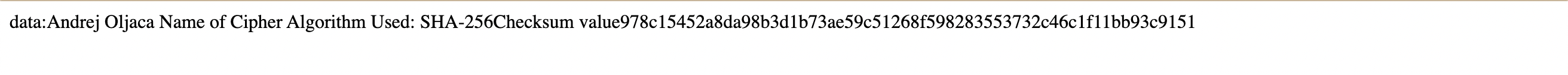




## 3. Deploy Cipher

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.

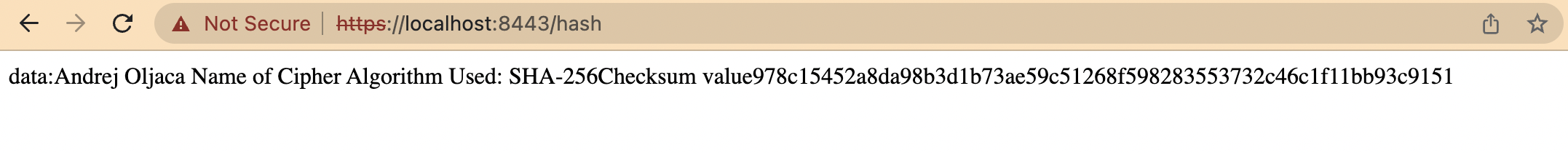
* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.



## 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.

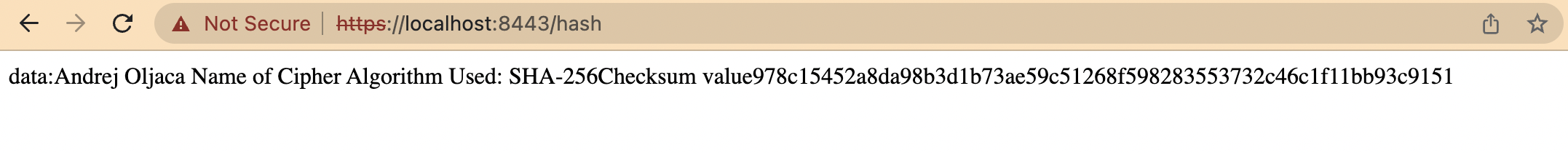
* Insert a screenshot below of the web browser that shows a secure webpage.

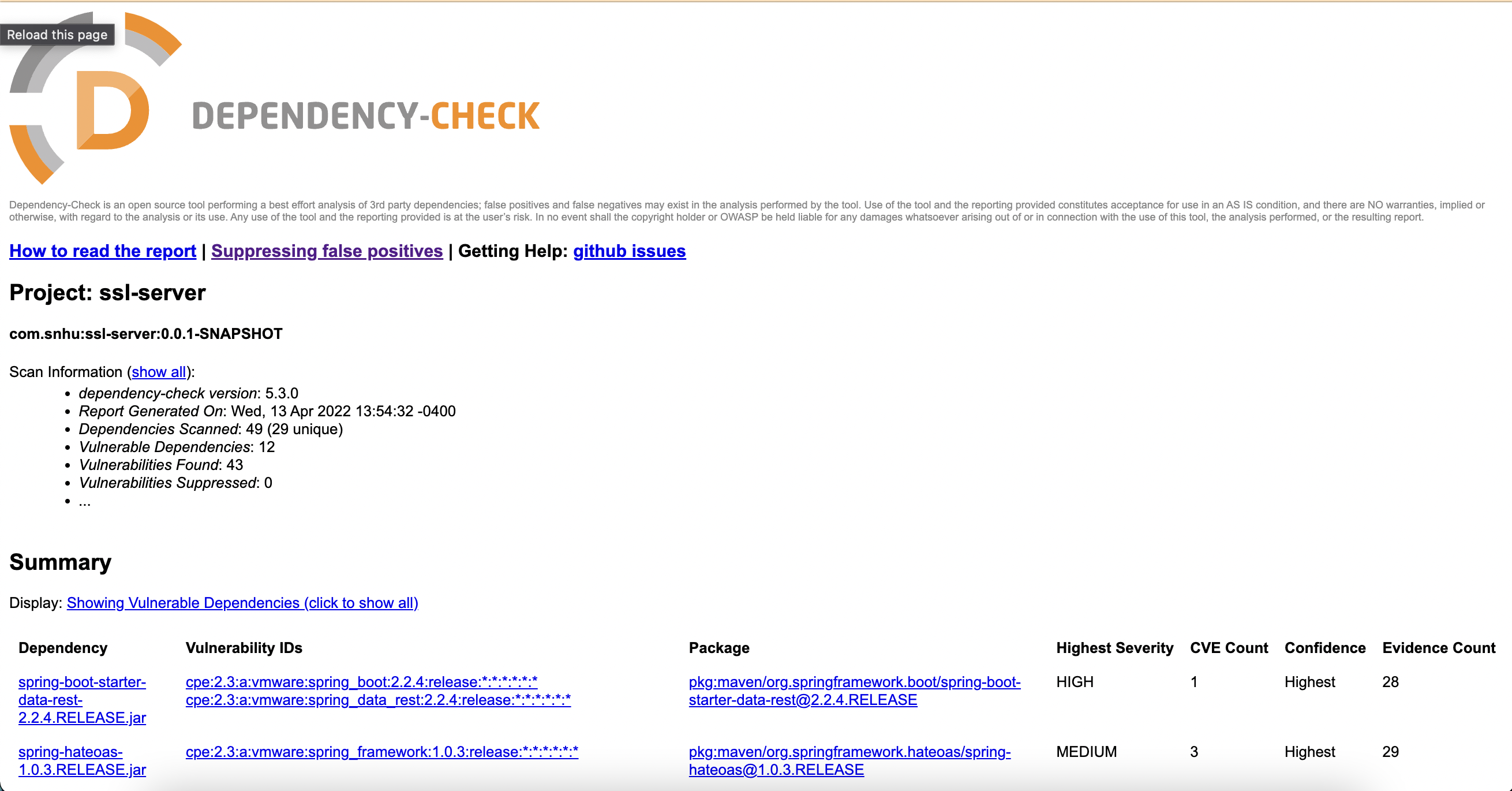


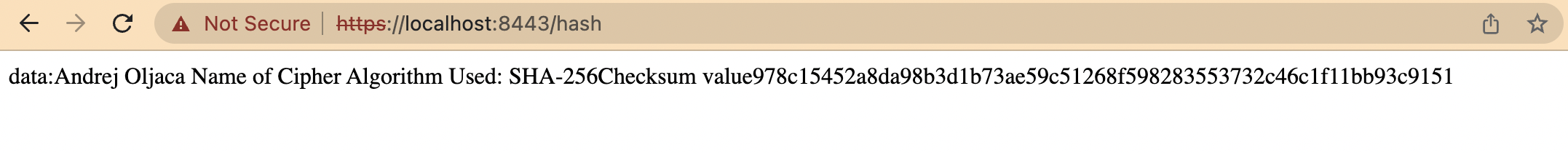
## 5. Secondary Testing

Complete a secondary static testing of the refactored code using the dependency check tool to ensure code complies with software security enhancements. You only need to focus on the code you have added as part of the refactoring. Complete the dependency check and review the output to ensure you did not introduce additional security vulnerabilities.

* Include the following below:
  + A screenshot of the refactored code executed without errors
  + A screenshot of the dependency check report



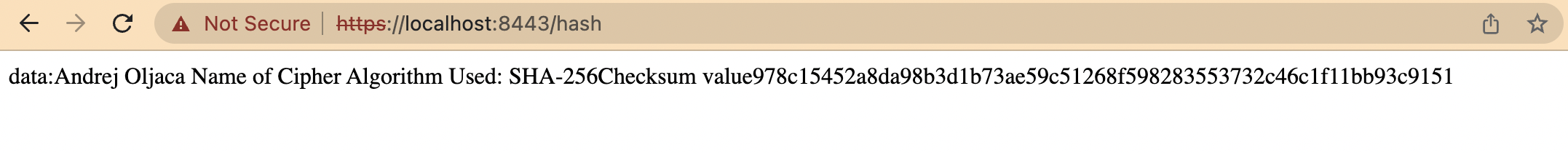




## 6. Functional Testing

Identify syntactical, logical, and security vulnerabilities for the software application by manually reviewing code.

* Complete this functional testing and include a screenshot below of the refactored code executed without errors.



## 7. Summary

Discuss how the code has been refactored and how it complies with security testing protocols. Be sure to address the following:

* Refer to the Vulnerability Assessment Process Flow Diagram and highlight the areas of security that you addressed by refactoring the code.
  + One thing I changed was working over https vs http. This makes for a more secure API because the hypertext transport protocol secure ensures the integrity, authenticity, and confidentiality of a website. I also used cryptography to ensure that my data was encrypted and could only be decrypted by me. I used the SHA-256 encryption hash algorithm. I made sure to focus on code quality and encapsulate my code.
* Discuss your process for adding layers of security to the software application and the value that security adds to the company’s overall wellbeing.
  + My process to adding layers of security to the software was to first do research, then look through the code, and then apply. Security helps the company keep a good reputation and prevents loss of profits that can come from attacks and hackers succeeding.
* Point out best practices for maintaining the current security of the software application to your customer.
  + The best practices for maintaining the current software application are to always add comments when adding more code, to follow code best practices, to encapsulate code, and to follow object-oriented programming basics.

References

Simplilearn, S. (2022, February 17). *What is SHA-256 algorithm: How it works and applications [2022 edition]: Simplilearn*. Simplilearn.com. Retrieved April 13, 2022, from <https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm>

Team, M., & Nadeem, M. S. (2021, March 17). *Symmetric vs asymmetric encryption: What's the difference?* Mailfence Blog. Retrieved April 13, 2022, from <https://blog.mailfence.com/symmetric-vs-asymmetric-encryption/#:~:text=Symmetric%20encryption%20uses%20the%20same,private%20key%20to%20decrypt%20information>.

Sidhpurwala, H. (2019). *A brief history of cryptography*. Red Hat - We make open source technologies for the enterprise. Retrieved April 13, 2022, from <https://www.redhat.com/en/blog/brief-history-cryptography>