****

# 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** | **14 October 2021** | **Sergio H. Passos** |  |

## 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

Sergio H. Passos

## 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.
* Discuss the hash functions and bit levels of the cipher.
* Explain the use of random numbers, symmetric vs non-symmetric keys, and so on.
* Describe the history and current state of encryption algorithms.

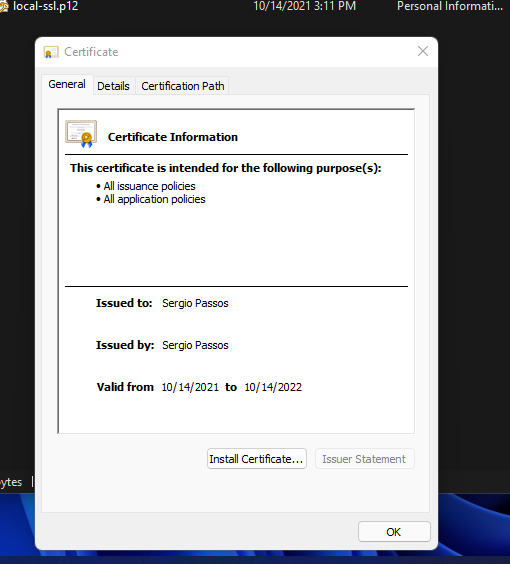
256-bit Advanced Encryption Standard (AES) cipher is the recommended encryption algorithm. AES is the only approved cipher by the United States National Security Agency (NSA) for protecting top secret information. AES offers many different levels of encryptions, such as 128-bit and 192-bit ciphers. Utilizing the 256-bit ciphers makes it almost impossible to attempt a brute force attack with today’s processing power. This makes it the most secure cipher to use today. The 256-bit cipher goes through 14 rounds—steps which include substitution, transposition, and mixing—to encrypt data. Random Number Generation (RNG) is essential for encrypting any data. Each round must have a unique key for that round. This key is generated using RNG and is applied to the block of data. AES process is a symmetric since it uses one key to encrypt and decrypt information. Keeping the key safe is critical for maintaining security. A non-symmetric process has two keys, one which is public and available to the public; and another key that is kept secret. These two keys have some mathematical connection to each other since they are used together for encryption and decrypting of information.

Cryptography has been used for thousands of years. They usually used simple substitution for sending letters, nothing to complicated. As you progress forward in time, World War 1 (WW1) and World War 2 (WW2) is when more advanced cryptography is starting to get utilized. However, the public never needed to know or even understand cryptography back then. The introduction of the home computer would change all that. In today’s computer driven world cryptography is more important than ever. Companies such as Microsoft would like to do away with passwords all together. Microsoft offers an authentication app which would be the tool used to enter your Microsoft accounts. Encrypting data has become more complicated, and more valuable to unauthorized actors. Some have even suggested that the government step in and create a national login system. So that a citizen wouldn’t need to create a secure password when trying to utilize anything on the web.

## 2. Certificate Generation

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

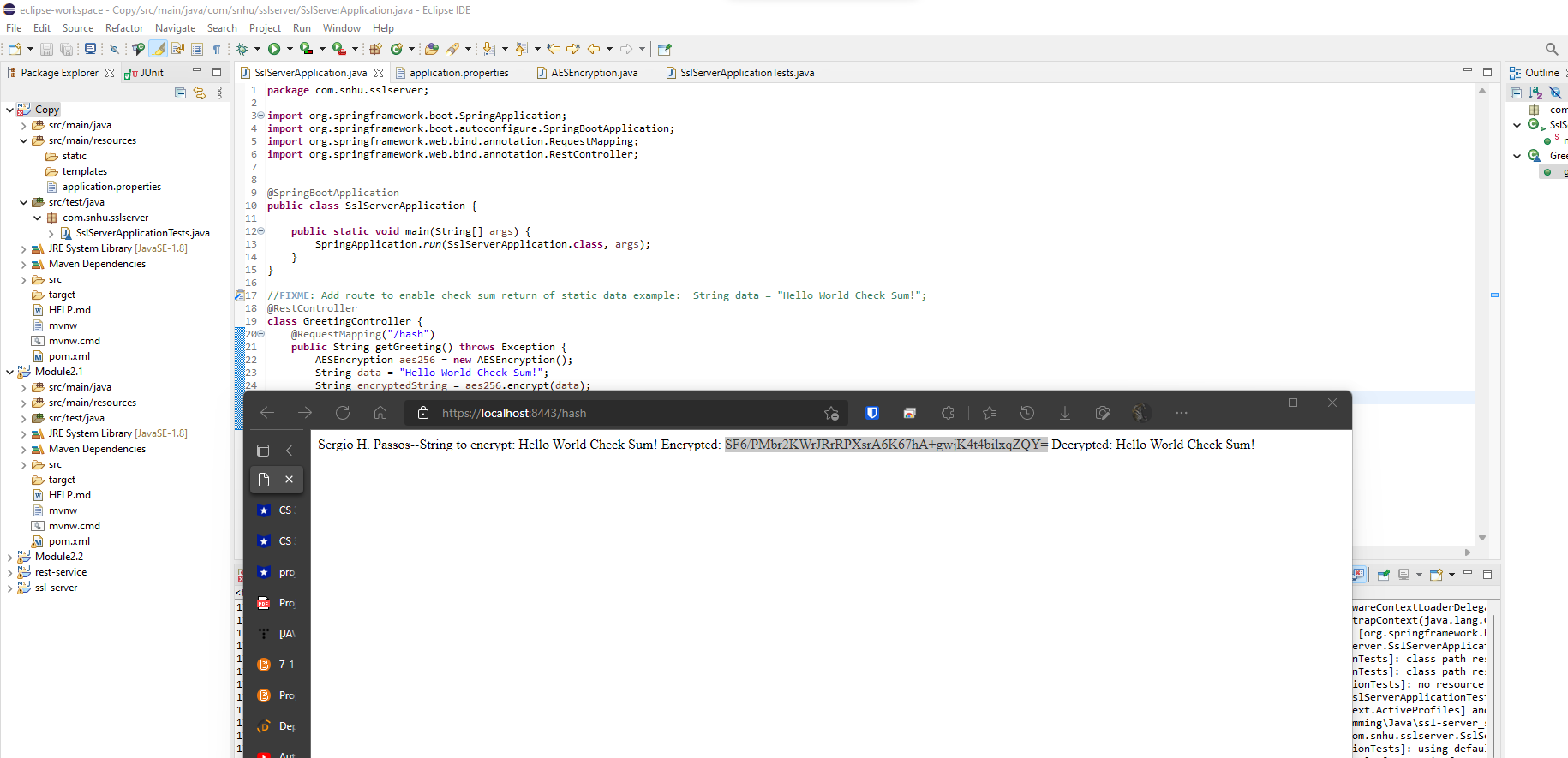
* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.



## 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.

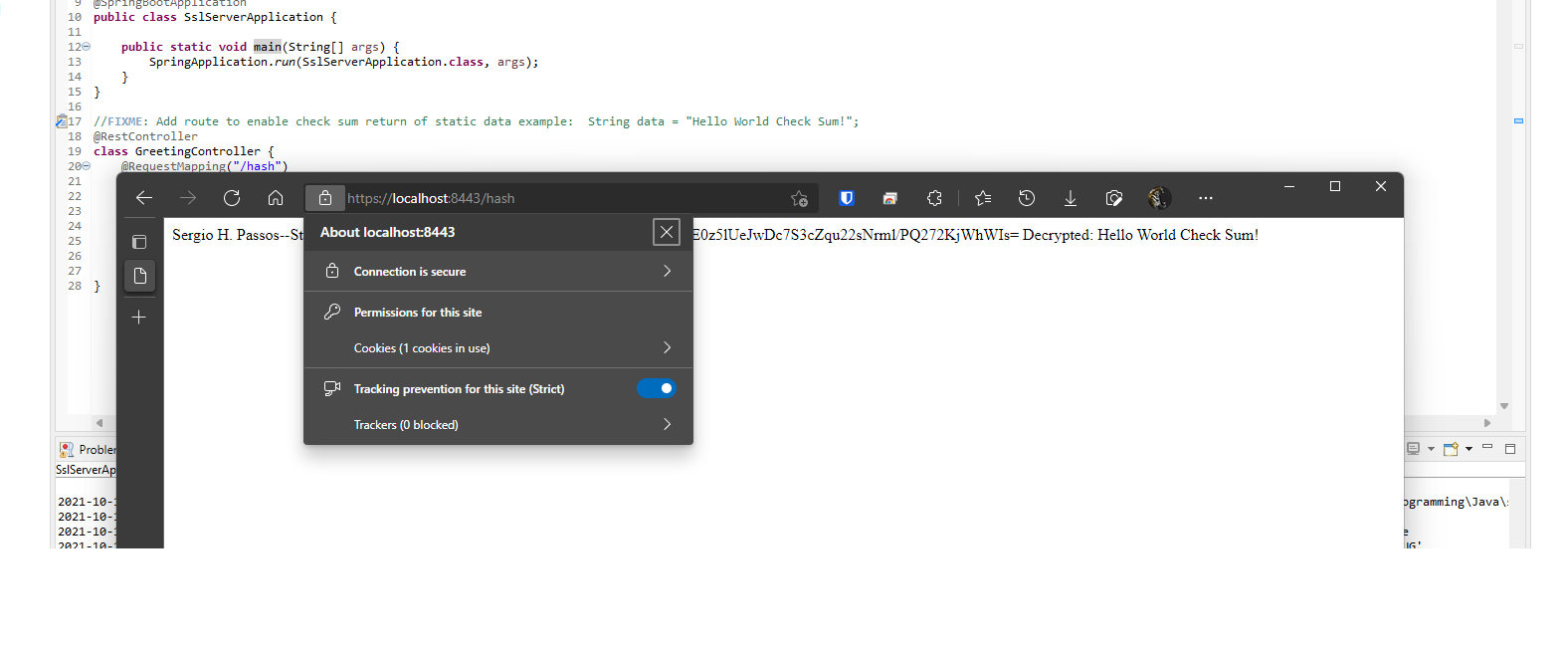
* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.



## 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.

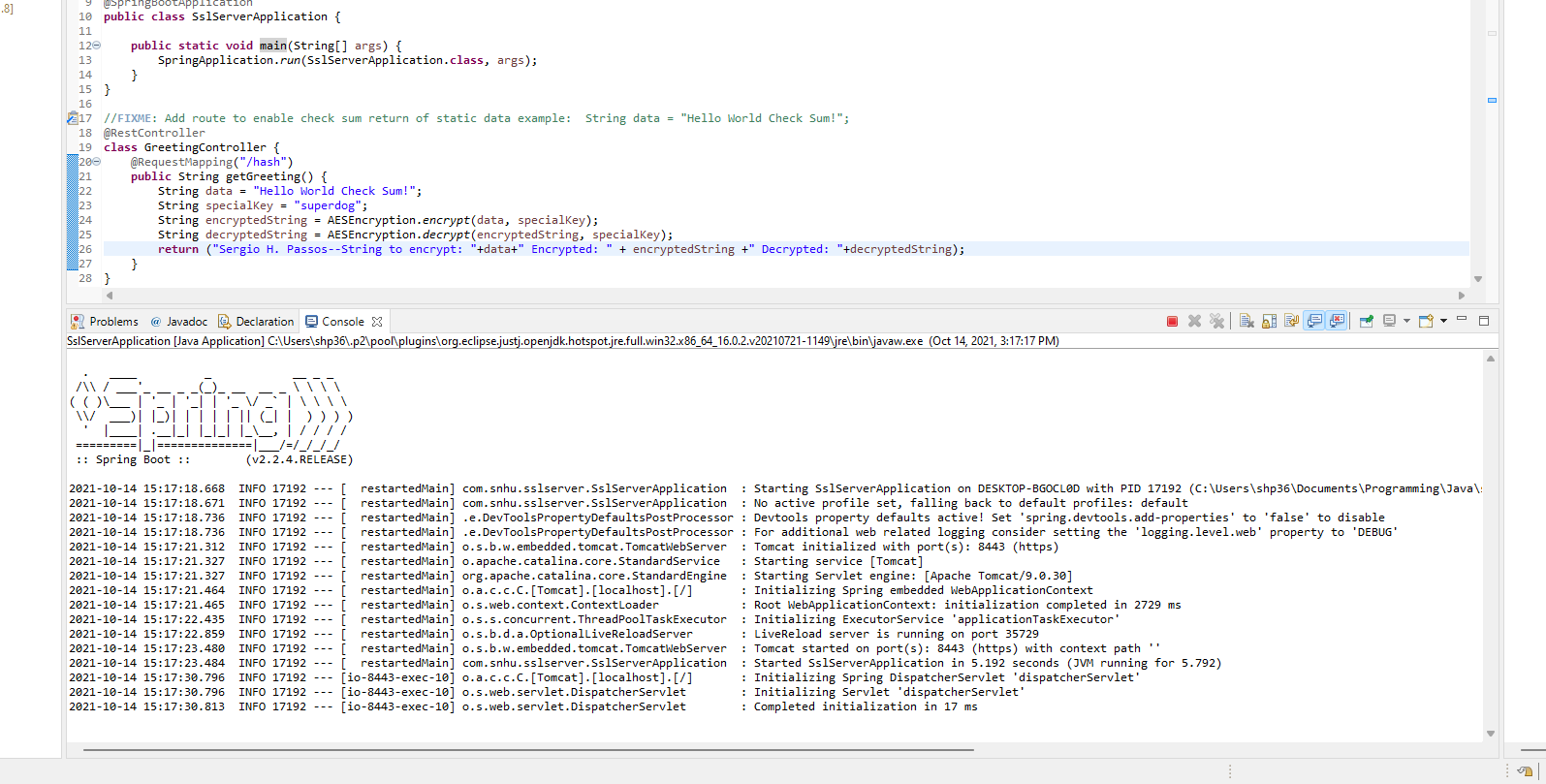
* Insert a screenshot below of the web browser that shows a secure webpage.

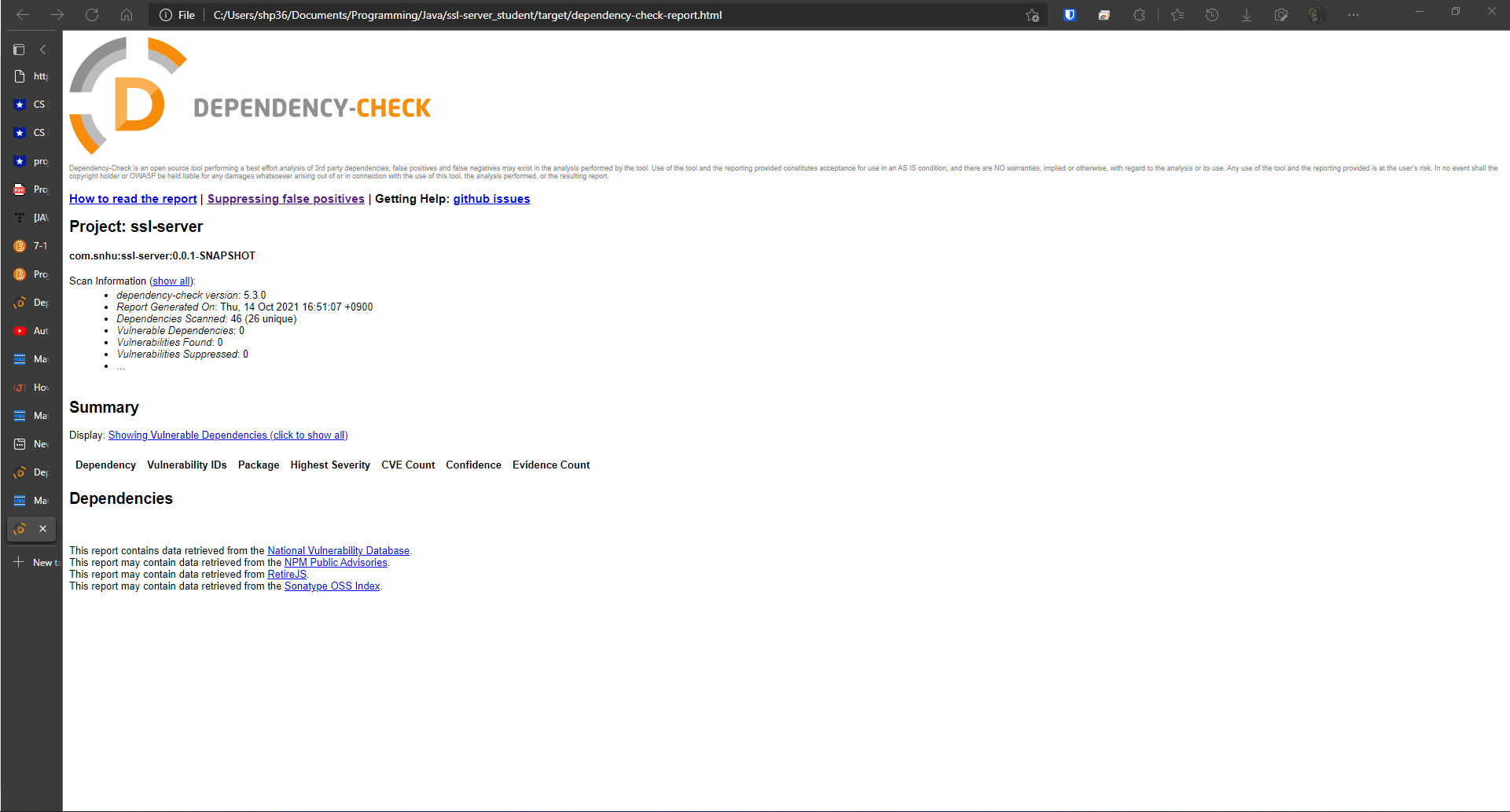


## 5. Secondary Testing

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

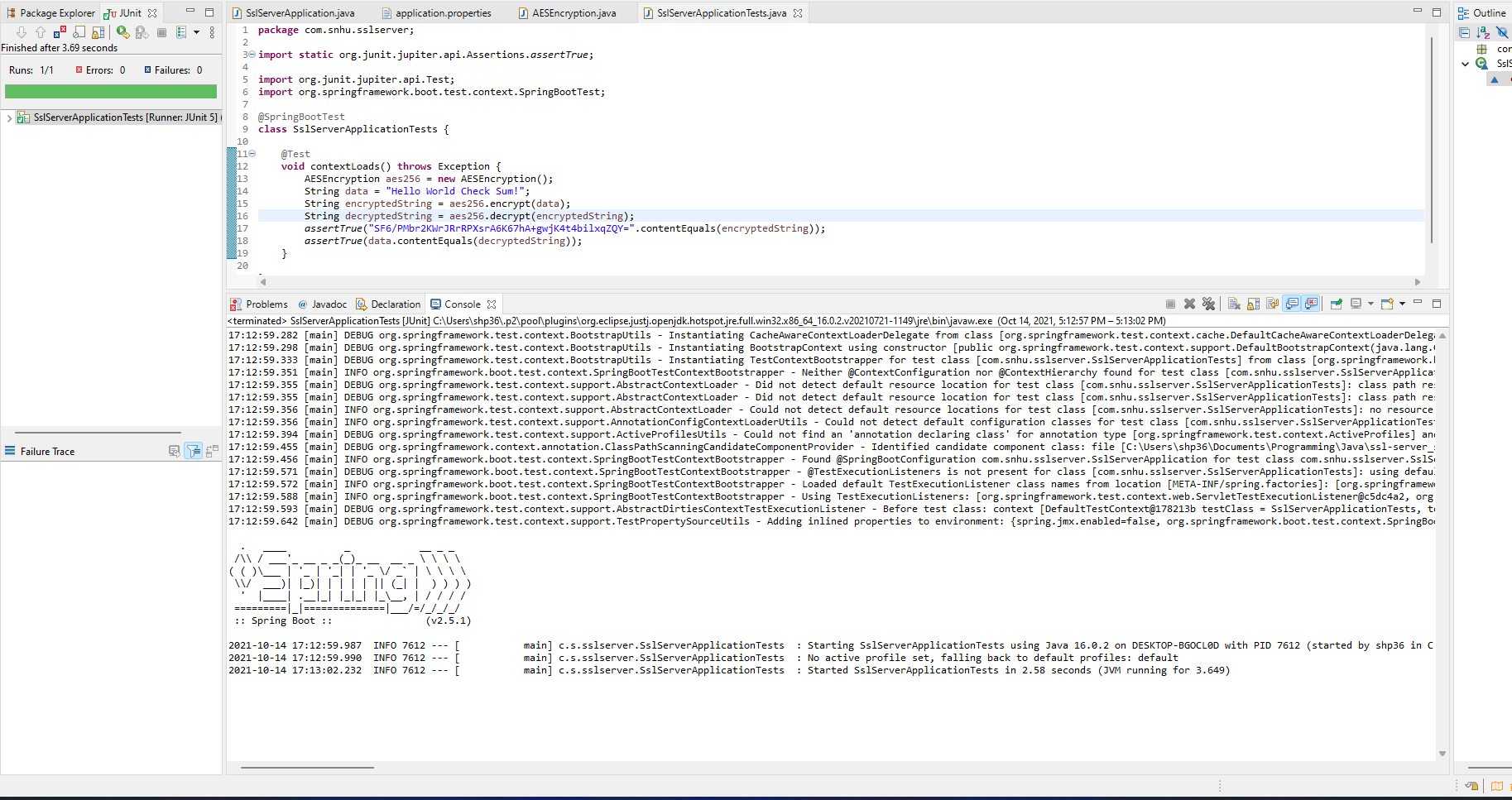




## 6. Functional Testing

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.
* Discuss your process for adding layers of security to the software application and the value that security adds to the company’s overall wellbeing.
* Point out best practices for maintaining the current security of the software application to your customer.

Cryptography, Client/Server, code error, and code quality have all been addressed during this refactoring. The addition of encryption into the application has already made it significantly more secure than before. Another addition to the security of the application was converting it from http to https. This has allowed for end-to-end encryption with each user who uses the application. Which makes it again considerably more secure than before. This means that the data that is being transmitted between the user and the server is encrypted. Making is substantially more difficult for dangerous actors to steal sensitive information. The best practice would be for the company to continue to check API vulnerabilities, and not update unless it helps improve the security. Additionally, the company should add extra functionality slowly, as to ensure the security of the application of above all else.