****

# 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** | **4/14/2021** | **Michael Bauer** | **Added information in all areas** |

## 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

Michael Bauer

## 1. Algorithm Cipher

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

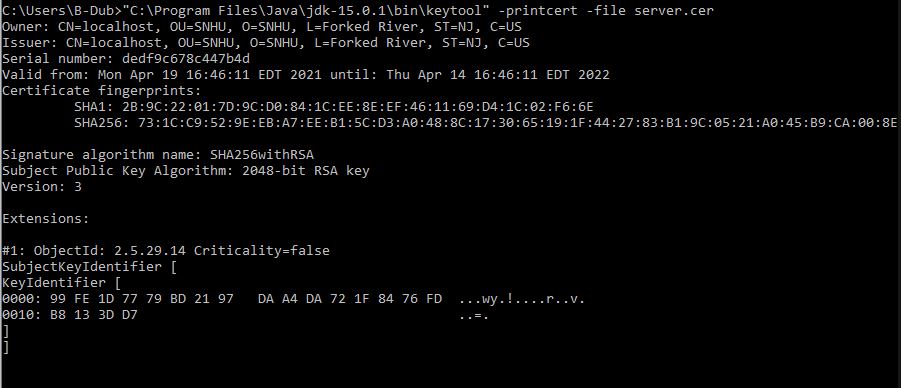
* Provide a brief, high-level overview of the encryption algorithm cipher.
* Discuss the hash functions and bit levels of the cipher.
* Explain the use of random numbers, symmetric vs non-symmetric keys, and so on.
* Describe the history and current state of encryption algorithms.

RSA algorithm when implemented correctly will be a secure algorithm. This is an asymmetric algorithm that deals with public keys. An asymmetric algorithm uses a public and private key. This differs from a symmetric algorithm that only uses a private key. These keys are used to encrypt and decrypt the data. To put it simple, two random numbers are used, one is used to change each bit which encrypts and the other is used to decrypt changing the bit back to what it was originally. The hash function for the checksum used is SHA-256, 256 stands for 256-bit encryption. This is very secure and difficult to crack. A hash function is not able to be decrypted. This is used to compare to another checksum and make sure they match.

## 2. Certificate Generation

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

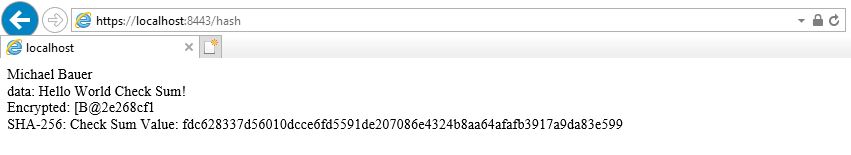
* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.



## 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.

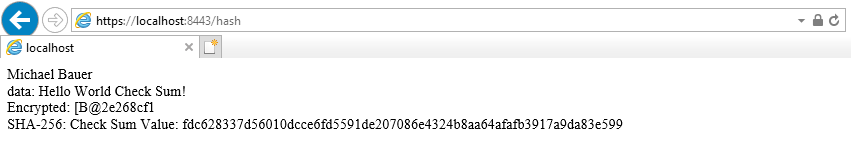
* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.



## 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.

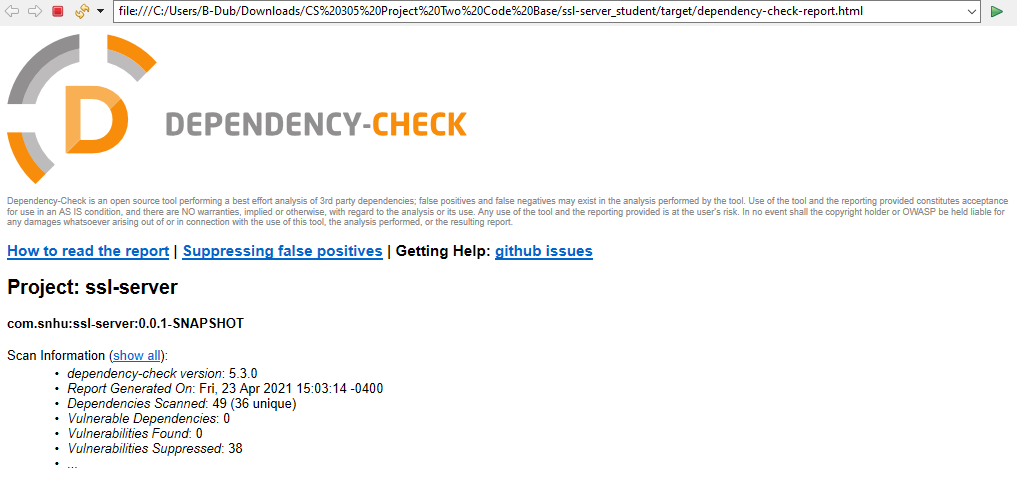


## 5. Secondary Testing

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.

The areas addressed are for cryptography and secure communications. The data being transferred is encrypted using a public key and decrypted using a private key. The connection uses a certificate to transfer the public key and this is used to encrypt the data being transferred. A checksum is sent along with the data and another checksum is created on the data that arrives. These two checksums are compared and are used to determine that the data that was transferred has not been tampered with before arriving.

The certificate allows the users to trust that their connection is actually to your web application. This certificate allows the public key to be used to encrypt the data being transferred. This makes it so the data that is seen by someone else, as an example a man in the middle attack, only sees the encrypted data. They cannot do anything with the data since they do not have the private key that will be needed to decrypt the data. The checksum is sent as another safety measure to prove that the data was not tampered with and then resent. If data is intercepted and changed before sending it to its destination the checksum will be different then the checksum created at the destination.

It is recommended that a vulnerability check is done periodically to make sure there has not been any new vulnerabilities found.