****

# 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** | **8/15/2021** | **Joshua Hampton** |  |

## 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

Joshua Hampton

## 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.

The encryption algorithm used for creating the checksum is SHA3-256. SHA3 is the encryption method which is the newest iteration of the SHA3 algorithm and is the newest NIST standard for hashing algorithms as of 2012 which was updated to prevent collisions. The “-256” portion of the encryption algorithm the is the key size. In general, the longer the key size is the harder it is to break the encryption. With SHA3 reversing a message from a hash can take up to 2256 work.

To further protect information as it passes through networks and websites there are a variety of different techniques that are used. Random numbers are used to create cryptographically random session tokens to and are submitted in a hidden field on form submissions to ensure the device that started a session is the same device that is communicating within that session. Certificates are generated to help systems determine the authenticity of servers. There are two types of certificate key generation, symmetric and non-symmetric. A symmetric key uses the same key for client and server, the problem with using a symmetric key us that you need control over both the client and server in order to import the key in both locations. The non-symmetric key is more commonly seen where there is a private key installed on a server and a public key issued to all clients, such as created by Certificate authorities. Certificate authorities help to identify servers as authentic for the address you are trying to contact. In the case of a self-signed certificate, a certificate is created on a server, exported, then imported into a client certificate store so that the client knows the certificate can be trusted. In the case of a third-party certificate store, a third-party certificate authority creates a certificate that is then imported into a server and when the client attempts to contact the server it receives the certificate and can verify the certificates authenticity with the third-party certificate authority.

Encryption algorithms are constantly changing and being updated as exploits are discovered, and their security is diminished. As computer systems become more powerful it becomes easier and faster for systems to be broken and brute force attacks to take place. As an example, securing data in transit is commonly done using Transport Layer Security, or TLS, currently. “The original SSL specification was developed and released by Netscape in 1995. The original protocol standard has been revised several times to adjust for weaknesses in the specification and new security techniques. In 1999, the standard became known as TLS, and the current version of TLS is 1.2. Despite the name change, most people continue to use SSL to refer to both SSL and TLS.” (Manico, J., p 139) Since SSL was first developed in 1995 it has gone through three revisions before TLS took its place in 1999, since then TLS has been through four revisions from TLS 1.0 to the current version TLS 1.3 in 2018.

## 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.

Text

Description automatically generated

Text

Description automatically generated

## 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.

Graphical user interface, text, application

Description automatically generated

## 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.

Graphical user interface, text, application

Description automatically generated

## 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

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

## 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.

Text

Description automatically generated with low confidence

## 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.

When refactoring my code I took into account Client Server infrastructure, Code Error Handling, and Code Quality. Using a Try/Catch block ensures there are no crashes during runtime. Security was incorporated into the webserver using a certificate for server identity verification and the https protocol for traffic encryption. Conciseness and commenting improves code quality and reusability. To maintain software security, it is important to keep software packages up to date and to regularly scan for vulnerabilities. Vulnerabilities are found in software packages and dependencies regularly and must be kept in the forefront of secure design.

**References**

Manico, J., Detlefsen, A., Kenan, K., Smith, M. (2015). Iron-clad java: Building secure web applications. McGraw Hill Education.

Kelsey, J. (2013). The New SHA3 Hash Functions. NIST. https://csrc.nist.gov/CSRC/media/Events/ISPAB-DECEMBER-2013-MEETING/documents/new\_sha3\_functions.pdf.