****

# Practices for Secure Software Report

Table of Contents

[Document Revision History 3](#_Toc102040754)

[Client 3](#_Toc102040755)

[Instructions 3](#_Toc102040756)

[Developer 4](#_Toc102040757)

[1. Algorithm Cipher 4](#_Toc102040758)

[2. Certificate Generation 4](#_Toc102040759)

[3. Deploy Cipher 4](#_Toc102040760)

[4. Secure Communications 4](#_Toc102040761)

[5. Secondary Testing 4](#_Toc102040762)

[6. Functional Testing 4](#_Toc102040763)

[7. Summary 4](#_Toc102040764)

[8. Industry Standard Best Practices 4](#_Toc102040765)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **10/11/2022** | **Kentrell Edwards** | **Initial Release** |

## Client



## Instructions

Submit this completed practices for secure software report. Replace the bracketed text with the relevant information. You must document your process for writing secure communications and refactoring code that complies with software security testing protocols.

* Respond to the steps outlined below and include your findings.
* Respond using your own words. You may also choose to include images or supporting materials. If you include them, make certain to insert them in all the relevant locations in the document.
* Refer to the Project Two Guidelines and Rubric for more detailed instructions about each section of the template.

## Developer

Kentrell Edwards

## Algorithm Cipher

AES cipher, which supports a range of key sizes and is one of the best standards currently in use, is the encryption technique. With 256-bit being incredibly challenging for attackers to crack, 128-bit or 256-bit encryption would likely be the preferred option for this application. The application's ability to generate symmetric keys will enable it to encrypt data as necessary and provide keys to its clients, who serve as the recipients of communications.

Depending on what the application needs to do, different symmetric and non-symmetric keys are used. In contrast to non-symmetric keys, which include public and private keys, with the former known to everyone and the latter known exclusively to the client, symmetric keys are essentially shared between the server and the client. These keys encrypt data and can only decrypt data when the proper key is used. A drawback of this is that if the key to any encrypted data is lost, the data would also likely be destroyed because it will be impossible to decrypt the data without the key. A unique identifier for each transaction can be created using random number generators, which may aid in determining whether particular events, such as data transfers or communications, occurred.

Because there are so many possibilities, the current version of AES can encrypt data using up to 256 bits, which is practically impossible to crack, for our application, we used 256-bit encryption, which ought to ensure secure communication between the server and client.

## Certificate Generation

Text

Description automatically generated

## Deploy Cipher

Graphical user interface, text, application

Description automatically generated

## Secure Communications

Graphical user interface, text, application

Description automatically generated

## Secondary Testing

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

## Functional Testing

Text

Description automatically generated

## Summary

The secure Restful Controller mapping for the hash endpoint that I included in my refactoring complies with the vulnerability assessment's criteria for secure code. Client-server applications now meet the evaluation criteria thanks to the SSL certificate installation, which creates a secure connection between the various clients and the server application. The cryptography is finished with the application of the SHA-256 hashing method.

Additionally, I upgraded the Spring Boot Framework to version and the Maven Dependency Check to its most recent iteration (7.2.1) (2.4.0). These steps ensure that the dependency checker can identify the most recent software issues and that we are using the most secure versions of our program. With the use of the SHA-256 cipher, which has one of the lowest collision probabilities, and the SSL layer, we can effectively encrypt our data and offer a safe connection to our clients, earning their trust as well as adhering to all legal requirements.

## Industry Standard Best Practices

In this application, it is crucial to maintain secure connections since insecure communications can allow data to be viewed by an attacker, who can then steal sensitive or private information. The repercussions of a security breach might include, but are not limited to, a loss of confidence between the owner of the program and the user, financial loss from government fines for not mandating secure communications, and financial loss from resources used to fix or resolve the security issue. By maintaining security in the application, the organization may safeguard its goods and resources while retaining the confidence and business of its customers.

The best practices for maintaining current security include checking the code periodically for vulnerabilities, particularly after adding new features or altering existing ones and before publishing or applying the changes in the live code. Before a function is live, while it is being developed or updated, the development team should fix any new vulnerabilities discovered. They should choose the appropriate course of action for any vulnerabilities that currently have no fixes and ascertain whether they affect any application components. The team can omit those weaknesses from their reports if they discover that they affect dependencies, not the code itself. The development team will need to watch out for upgrades or fixes for any security vulnerabilities in the code.

**References**

Lake, J. (2020, November 11). What is AES encryption (with examples) and how does it work? Comparitech. https://www.comparitech.com/blog/information-security/what-is-aesencryption/#:~:text=The%20Advanced%20Encryption%20Standard%20(AES,use%20all%20of %20the%20time.

Manico, J., & Detlefsen, A. (2015). *Iron-clad java: Building secure web applications*. McGraw-Hill Education.

*How to create a self signed certificate using Java Keytool*. SSL Certificate Comparison and Reviews. (2010, October 30). Retrieved October 16, 2022, from https://www.sslshopper.com/article-how-to-create-a-self-signed-certificate-using-java-keytool.html

Java security standard algorithm names. (n.d.). Retrieved October 16, 2022, from https://docs.oracle.com/javase/9/docs/specs/security/standard-names.html#cipher-algorithm-names