****

# 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** | **[06/25/2023]** | **Maximilian Cozone** |  |

## Client



## Instructions

Submit these 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

Maximilian Cozzone

## Algorithm Cipher

I chose AES 256 (Advanced Encryption Standard).

Advanced Encryption Standard (AES) is among the best and most versatile encryption ciphers available as of my last update in September 2021. AES has several advantages that make it a strong choice: Security, efficiency, and Scalability.

Symmetric encryption, also known as secret-key encryption, uses the same key for both encryption and decryption. The sender and the receiver share the same secret key, which is used to encrypt the data by the sender and decrypt the data by the receiver.

Asymmetric encryption, also known as public-key encryption, uses a pair of mathematically related keys: a public key and a private key. The public key is used for encryption, while the private key is used for decryption. Asymmetric encryption allows for secure communication, key exchange, and digital signatures.

During my research I came across an article calling it the “MOST SECURE “,

As described by N-able: “Originally adopted by the federal government, AES encryption has become the industry standard for data security. AES comes in 128-bit, 192-bit, and 256-bit implementations, with AES 256 being the most secure. “(July 29th, N-able.com)

Other than using AES 256 it is best to use security standard protection measures such as Firewalls to prevent intrusion, Updates, and patches to avoid zero-day attacks and Least-Privilege authentication for integrity and authenticity measures to avoid intrusions from the inside. Antivirus to detect any phishing attacks or unwanted programs/files downloaded or unknown. And lastly, the networking principle of using a secure protocol such as HTTPS.

Today AES is a trusted system with widespread adoption. AES libraries have been developed for programming languages including C, C++, Java, JavaScript, and Python. AES is used by file compression programs including 7 Zip, WinZip, and RAR; disk encryption systems like BitLocker and File Vault; and file systems like NTFS. It’s an important tool in database encryption as well as in VPN systems like IPsec and SSL/TLS. Password managers like LastPass, KeePass, and 1Password use AES, as do messaging programs like WhatsApp and Facebook Messenger. An AES instruction set is integrated into all Intel and AMD processors. Even video games like Grand Theft Auto IV use AES to guard against hackers.

During the Roman era, cryptography played a significant role in securing military communications. The Caesar cipher was a commonly used encryption method at that time.

## Certificate Generation

Insert a screenshot below of the CER file.

A picture containing text, screenshot, font

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a computer

Description automatically generated with medium confidence

## Secure Communications

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

A screenshot of a computer

Description automatically generated with medium confidence

## Secondary Testing

Insert screenshots below of the refactored code executed without errors and the dependency-check report.

A picture containing text, screenshot

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated with medium confidence

Before Suppression:

A screenshot of a computer

Description automatically generated

After Suppression

A screenshot of a computer

Description automatically generated with medium confidence

## Functional Testing

Insert a screenshot below of the refactored code executed without errors.

A screenshot of a computer program

Description automatically generated with medium confidence

I had to update the Dependency-check-Maven version from 5.3.0 to 8.3.1.

Also had to suppress over hundreds of vulnerabilities found during the dependency check. Resulting in no visible vulnerabilities detected.

## Summary

A yellow and blue square with black text

Description automatically generated with low confidence

RESTful API services were utilized to implement the functionality.

We utilized localhost to host an open port on a server and implemented the necessary security measures, including the appropriate HTTPS protocol.

The use of HTTPS protocol and SSL/TLS certificates ensured encrypted communication between the client and the server.

A picture containing text, font, screenshot, electric blue

Description automatically generatedAll user inputs were carefully validated and sanitized to prevent common vulnerabilities such as SQL injection, cross-site scripting (XSS), and command injection attacks.

A picture containing text, font, screenshot, electric blue

Description automatically generated

Strong cipher suites and protocols were configured to prevent vulnerabilities like protocol downgrade attacks and weak encryption.

## A picture containing text, screenshot, font, electric blue Description automatically generatedDuring the code review process, we identified errors and promptly resolved them by implementing error handling mechanisms, such as catch blocks and exceptions.

Specifically, in the cryptography area, we implemented a hash function for checksum verification to ensure data integrity. Additionally, we utilized APIs, leveraging the Spring Framework, to develop a RESTful API for seamless communication. Lastly, in the client/server area, we focused on generating certifications and effectively managing keystores to establish secure connections between clients and servers. These practices align with industry-standard best practices and enhance the overall security posture of the application.

## Industry Standard Best Practices

Sensitive data was properly protected, and encryption techniques were applied where necessary.

To add layers of security to the software application, the following process was followed:

Implemented security best practices: input validation, secure communication, access control, error handling, and data protection, code maintenance, error handling and patching/updating regularly.

Secure coding guidelines were followed to prevent security vulnerabilities.

Regular security maintenance: Ongoing monitoring, vulnerability scanning, and timely patching of security vulnerabilities were incorporated into the software development lifecycle. This ensures that the application remains secure over time and can adapt to emerging threats.

Such as SQL injections, DDOS and Brute forcing DNS spoofing.

By following this process and addressing the identified areas of security, the code was refactored to comply with security testing protocols and enhance the overall security posture of the software application.

*Citation:*

N-able. (2023, January 26). *Advanced encryption standard: Understanding AES 256 - N-able*. N. https://www.n-able.com/blog/aes-256-encryption-algorithm

*Creating unpredictability: Random Number Generator Use Cases - Unitychain.io: Blog*. Unitychain.io | Blog. (2021, May 3). https://www.unitychain.io/blog/random-number-generation-and-their-use-cases/

*ISO/IEC 19790:2012*. ISO. (2015, November 1). https://www.iso.org/standard/52906.html

ClickSSL. (n.d.). *256-bit encryption – IS AES 256 bit encryption safe in modern times?* ClickSSL. https://www.clickssl.net/blog/256-bit-encryption

Callaghan, P. (n.d.). *Why you should use SHA-256 in evidence authentication*. Pagefreezer Blog. <https://blog.pagefreezer.com/sha-256-benefits-evidence-authentication#:~:text=Collisions%20are%20incredibly%20unlikely%3A%20There,the%20exact%20same%20hash%20value>.