****

# 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** | **[Date]** | **[Your Name]** |  |

## 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.

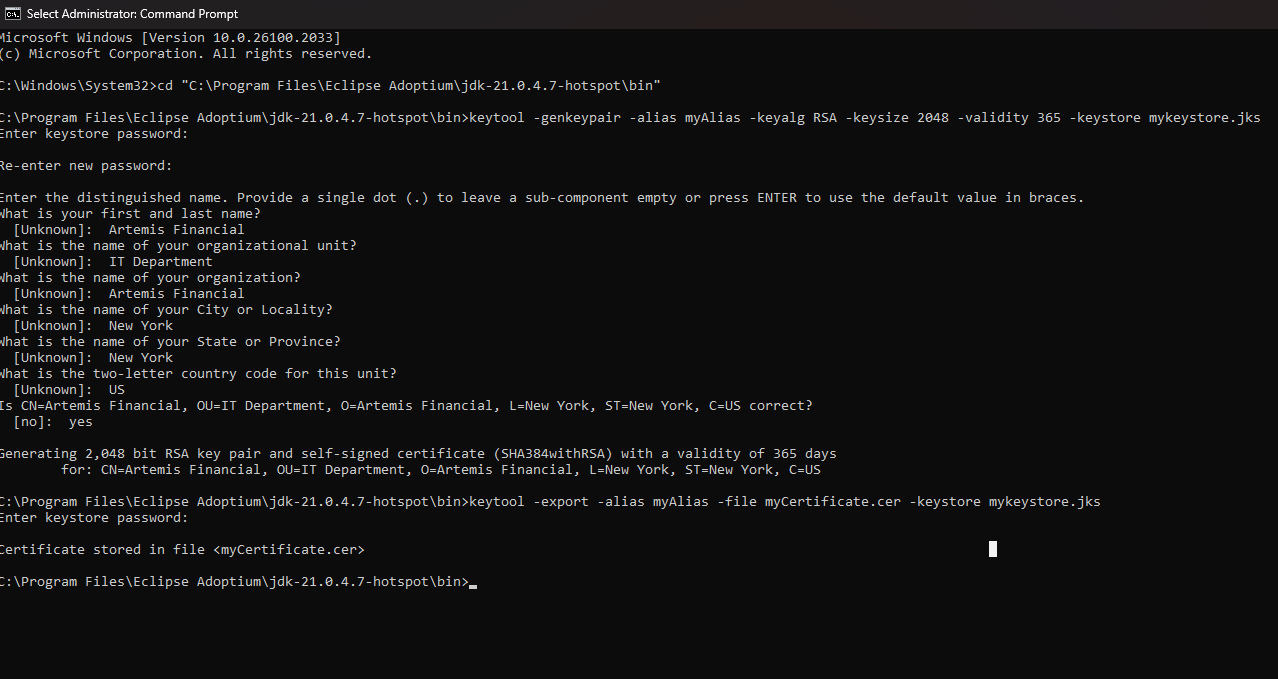
## Developer

Andrew Corrigan

## Algorithm Cipher

I recommend using the SHA-256 encryption algorithm for its balance of security and performance. I have used it in previous projects, designed by the NSA, the algorithm provides a strong hashing function that is widely accepted for secure data handling. SHA-256 operates by processing data in 512-bit blocks and produces a fixed 256-bit hash value. This function takes an input of arbitrary length and produces a unique 256-bit output. Random numbers are another critical element in the encryption processes. A unique key is generated, which helps to prevent attacks that exploit predictable values. SHA-256 was released in 2001 as a part of the SHA-2 family. As previously stated, it is widely used in various applications, and well trusted. The adoption of SHA-256 has been largely positive due to its resilience against collision attacks.

## Certificate Generation



## Deploy Cipher

A screen shot of a computer

Description automatically generated

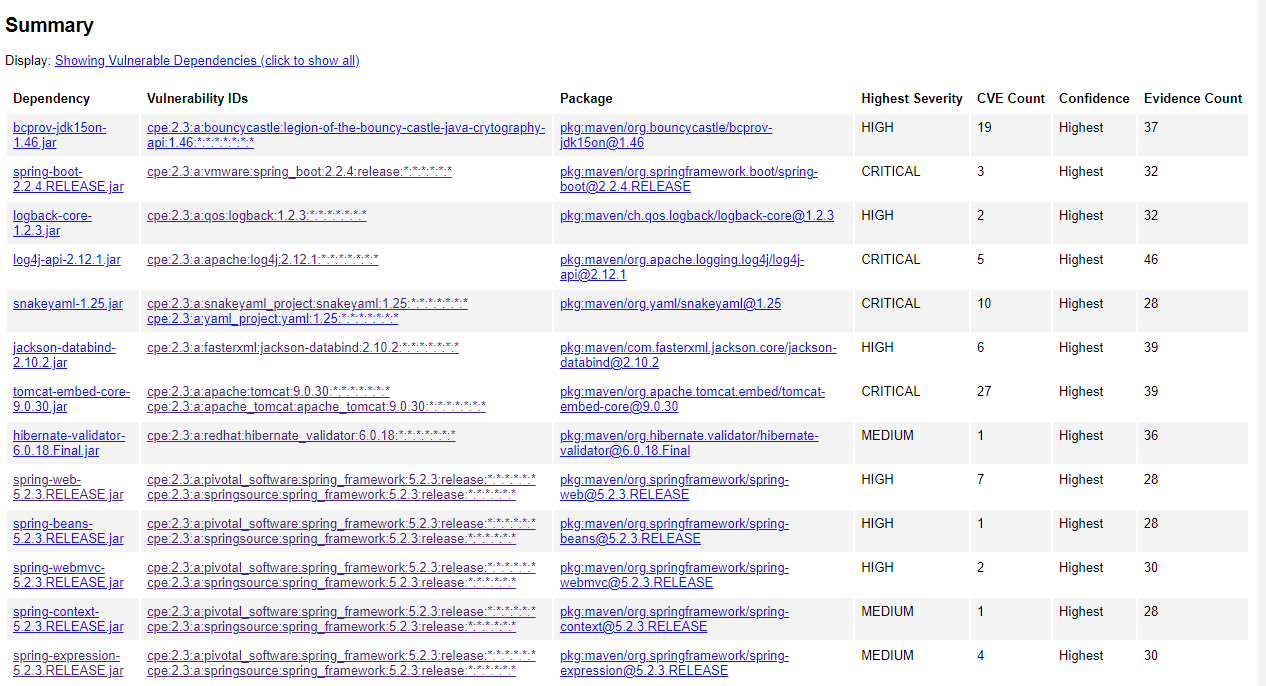
## Secure Communications



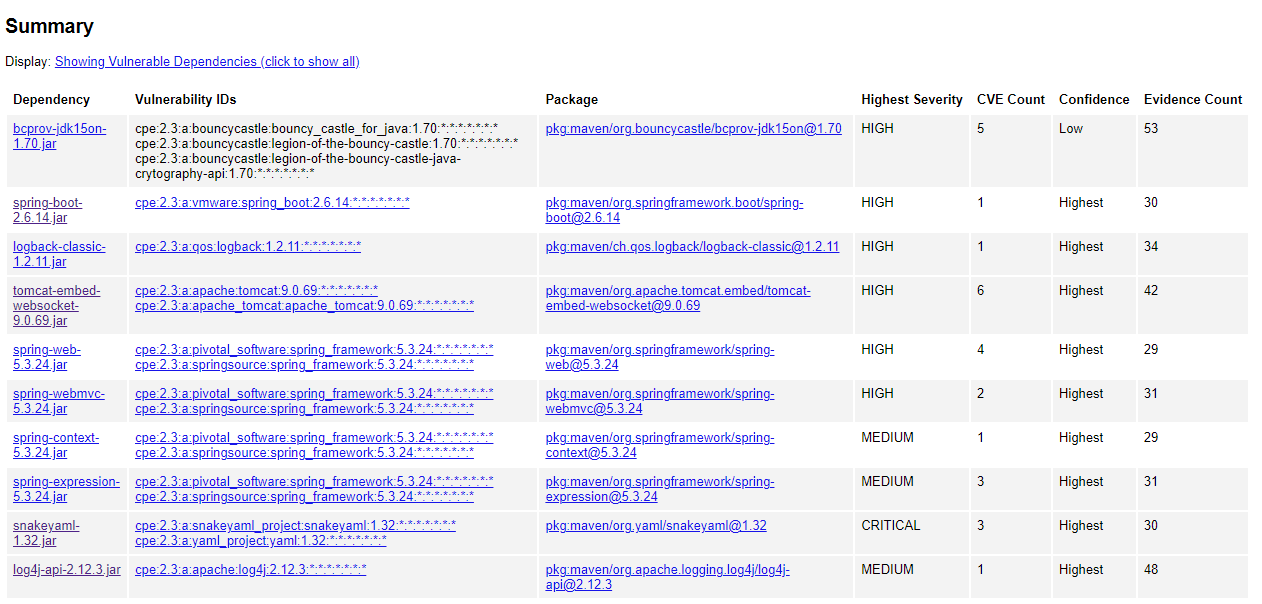
## Secondary Testing

A computer screen shot of a program

Description automatically generated



## Functional Testing A computer screen shot of a program Description automatically generated



## Summary

The code has been refactored and tested, ensuring it complies with security protocols. The changes made address previously identified vulnerabilities, although I will admit wasn’t able to solve all the issues. Still, the application now employs secure algorithms, utilizes strong encryption practices, and has been thoroughly tested for vulnerabilities.

## Industry Standard Best Practices

To maintain the application’s security, I implemented best practices such as input validation and sanitization to prevent SQL injection and XSS attacks. I have ensured secure data transmission by using HTTPS and have monitored third-party dependencies with OWASP Dependency-Check to address vulnerabilities. Error handling was managed carefully to avoid revealing sensitive information in error messages, and I advocated for secure configurations and regular security training for the team. These practices will significantly reduce risks associated with data breaches and enhance client trust. Compliance with industry regulations mitigates legal risks and fosters a culture of security awareness that can ultimately leading to cost savings and improved software quality.