****

# 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** | **10/14/2022** | **Ryan Betts** |  |

## 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

Ryan Betts

## Algorithm Cipher

## The client, Artemis Financial, is a finance consulting company that creates financial plans for its customers. Artemis Financial would like to verify that the data being communicated to and from its customers through its web app is secure and has not been altered by attackers. A hash function takes data as an input and outputs a unique checksum value for that data. If any single bit of data is modified, the hash function returns a completely different checksum. By comparing checksums between two files, Artemis Financial and its customers can verify that no modifications have been made to the data being communicated.

## Due to the risky consequences of financial data such as transaction quantities or bank account numbers being modified it is important that the algorithm cipher used is resistant to key collisions where two different input return the same checksum. Therefore, I will recommend deploying the SHA3-256 to generate checksums. Since 2015, SHA3 has been used by the United States National Institute of Standard and Technology for hashing and encryption making it a valid choice for securing sensitive information.

## Certificate Generation

Calendar

Description automatically generated with medium confidence

## Deploy Cipher

Graphical user interface, text, application

Description automatically generated

## Secure Communications

Graphical user interface, text, application, email

Description automatically generated

## Secondary Testing

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

Text

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

## Functional Testing

Text, letter

Description automatically generated

Graphical user interface, text, application

Description automatically generated

The added code implements functionality to hash a static string using the built-in Java SHA3-256 algorithm. The result of the hashing algorithm called a checksum is inserted into an HTML string. The HTML string containing the checksum is displayed to the user in their web browser. HTTPS has also been incorporated into the web application. This allows encrypted secure communication between the client and the server.

## Summary

The area of security associated with the refactored code is Cryptography. The refactored code uses cryptography to secure data in transit and data at rest. The generation of TLS/SSL certificate allows the web application to use HTTPS for secure communication between clients and the server. HTTPS ensures that the data transmitted between the web application and the users cannot modified or read by malicious actors during transit. This is especially important for a company like Artemis Financial because the communication between the company and its customers may include sensitive financial or personal data.

The refactored code also uses cryptography to secure data at rest. The data, “Ryan Betts CS-305-T1166 Software Security 22EW1” is statically defined in the web application. The application applies the SHA3-256 algorithm to encrypt this data into a unique checksum value. If any tiny bit of the original data is modified, the algorithm would result in a completely different checksum value. This allows either the client or the server to verify if two pieces of data are exactly identical and have not been modified by malicious actors.

Making a software application more secure starts with assessing the potential threats against the application. As a financial company, I assume that Artemis Financial stores sensitive financial and personal information on its servers. The company also transfers and receives data to and from its customers through the Internet. This data would be valuable to cybercriminals and thus has an elevated risk of threats. I also used a static testing tool to analyze the codebase for the software application and its dependencies for known vulnerabilities. After analyzing the potential threats, the next step I performed was determining and implementing the strategies and tools that would be must effective at reducing the identified threats. I decided that using encryption to create unique checksums and implementing the HTTPS protocol would secure data at rest and data during transit, respectively. I also updated the dependencies that the static testing tool determined to be vulnerable which drastically reduced the number of vulnerabilities found after re-analyzing the codebase. The last step for securing a software application is maintenance. This includes continuously testing old and new code for security flaws and keeping the codebase’s dependencies up to date to reduce the possibility of known vulnerabilities in dependencies from being exploited against the application.

## Industry Standard Best Practices

## The Open Web Application Security Project (OWASP) states in its secure coding practices guide that encryption for should be used to secure transmissions containing sensitive information (2010). By implementing HTTPS, I am ensuring that requests and responses made to and from the web application are secure and cannot be read or modified by untrusted sources. It also states that the web application should use trusted built-in functions rather than writing new custom code to perform common tasks. I use Java’s built-in MessageDigest object to encrypt the data, rather than code an encryption algorithm myself. I encrypt the data using the SHA3 algorithm which is what the United National Institute of Standards and Technology recommends using for the encryption of sensitive information (2001). Finally OWASP recommends all third-party libraries and dependencies should be regularly reviewed in case there are new vulnerabilities discovered and the dependencies need to be updated. I used the Maven dependency check static testing tool to review the codebase for known vulnerabilities and updated dependencies to their latest versions to reduce the number of potential exploits.

## Using industry standard best practices for secure coding ensures that a company’s sensitive data and the privacy of its customers are protected. The company has a responsibility to protect the data of its customer in order to build trust and credibility. Data breaches could also cause legal issues for the company or lead to financial burdens. It is important to follow the industry standard best practices because they have been developed with the best interests of the company and the consumer in mind. Because they are now considered the standard, following these best practices aligns their security policies with other companies and keeps them competitive because consumers know that they can trust the company with their data.

# References

Announcing the advanced encryption standard. (2001, November 26). *National Institute of Standards and Technology*, *197*. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

Manico, J., & Detlefsen, A. (2014, September). *Iron-Clad Java: Building secure web applications*. McGraw-Hill.

Mouha, N. (2021, July 23). Review of the advanced encryption standard. *National Institute of Standards and Technology*. https://doi.org/10.6028/nist.ir.8319

# References

SHA-3 standard: Permutations-based hash and extendable-output functions. (2015, August) *National Institute of Standards and Technology*, *202*. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf