****

# 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** | **04/15/2024** | **Matt Knutson** | **Alias: selfsigned**  **Password: Knutson999** |

## Client



## 

## Developer

Matt Knutson

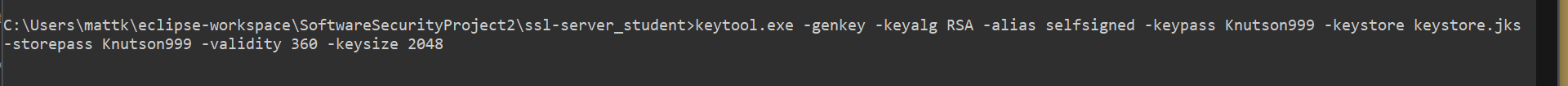
## Algorithm Cipher

For the Artemis Financial project, we will be using a checksum to verify data integrity. A checksum is a hashing algorithm that uses random numbers to generate a unique string of data. Random numbers are used to ensure that unique hash values are created and that collisions do not occur within the program. A Salt, or randomly added number/numbers, can be used to add an extra layer of protection and randomness when generating the checksum. Symmetric and Asymmetric key values will also need to be considered when deploying cryptographic algorithms. A symmetric key uses one key to encrypt data and decrypt data, while asymmetric cryptography uses a public and private key pair for encryption.

For this project in particular, we will be using SHA-256-bit for the hashing algorithm when dealing with data at rest and in transit. The checksum will use a symmetric key and be collision resistant. Sha-256 is part of the SHA-2 family and is the current standard of the N.I.S.T community. SHA-1 and its predecessors are currently not acceptable for use in the industry and are looking the be completely obsolete by 2031. It won’t be long before the SHA-2 family is following in SHA-1’s footsteps due to the advancement of quantum computing.

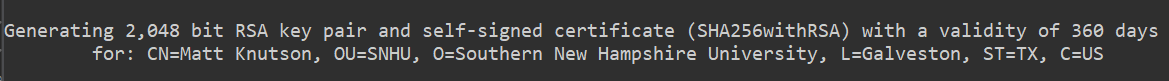
## Certificate Generation

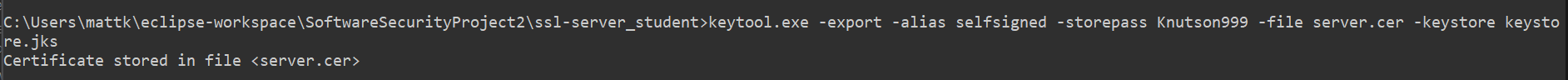
Generating a 2,048-bit RSA key pair and a self-signed certificate using keystore.exe



A black screen with white text

Description automatically generated





A computer screen shot of a black screen

Description automatically generated

A screenshot of a certificate

Description automatically generated

## Deploy Cipher

Unfortunately, after many, many failed attempts at deploying the cipher and rewriting the program, I was unable to get it to run and connect to host 8443:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

## Secure Communications

This was Artemis Financials’ original server connection:

A screen shot of a computer

Description automatically generated

This is the secure server with SSL enabled:

A screenshot of a computer

Description automatically generated

## Secondary Testing

This was the original program, before any security measures were taken:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated

For security, two new classes are created to assist in routing data through SSL and Error Control, a Dependency-Check is run, a keystore folder is created to hold the security certificates, and a suppression.xml file is created to suppress false positives:

A screenshot of a computer

Description automatically generated

This is the original Dependency-Check using OWASP version 5.3:

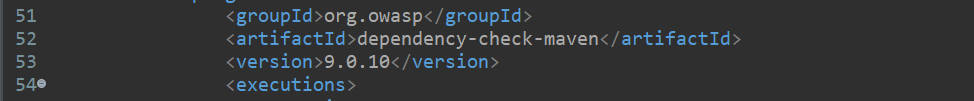
A screen shot of a computer

Description automatically generated

A close-up of a computer screen

Description automatically generated

After Updating the OWASP version to 9.0.10, the Dependency-Check is re-run:



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A group of text boxes

Description automatically generated

Then, after a long inspection, the suppression.xml file is filled with false-positives from the OWASP Dependency-Check. Finally, the Dependency-Check is run again to make sure that correct vulnerabilities were suppressed:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

A computer screen shot of a computer

Description automatically generated

## Functional Testing

The Updated SslServerApplication.java file:

A screenshot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generated

The SecurityConfig.java file:

A screenshot of a computer

Description automatically generated

A computer screen shot of a program

Description automatically generated

The ErrorController.java file:

A screen shot of a computer program

Description automatically generated

## Summary

Due to the sensitive nature of the data being handled and transferred at Artemis Financial, there were several areas of the original program that needed to be updated and refactored for security purposes. The first area of concern was the need for a secure algorithm to encrypt data and communications to and from the company. This was done by generating a checksum using the MessageDigest function and adding the parameters SHA-256 to implement the appropriate cipher. The hashed data was then converted to a hexadecimal value using the BytesToHex function, and then this encrypted string will be used to verify the integrity of the data when it is used by the company or their customers. Next, the company needed to make sure they were using secure API practices. HTTP traffic was rerouted through the 8843 port to ensure that it is automatically using HTTPS and is secured using the SSL communication protocol. A self-signed certificate was then created using an RSA key pair to establish a trustworthy relationship between the client and their customer base. In this project, I was not successful in accessing port 8843 from the local host. I believe this is due to the security of my personal device not trusting the source of the self-signed certificate. In the future, in a real-world scenario, a Certificate Authority will be used to avoid this issue. The next step was reviewing the quality of the code and examining the data structures to establish the level of security being provided. After the review, it was necessary to refactor the code by adding two new classes and a suppression file. The SecurityConfig.java class was added and used to push HTTPS requests through a centralized, secure login page. Within the class, two functions are used to handle the customer’s username and password information. The setDetailsService creates and modifies usernames and passwords, while the passwordEncoder method encodes the user’s password. The serviceController method is used to limit requests for access to the site. The default limit is 20 per minute. This method not only helps eliminate purposeful Denial-of-Service attacks, but also assists in successful scalability as a company, or website, grows in size. The second class that was needed was the ErrorController.java class. This class allows the application to access the application.properties file, where HTTPS using SSL can be implemented and its configurations can be tweaked. Finally, a Dependency-Check was initially run to see a list of vulnerable dependencies. A long inspection of these dependencies was needed to identify false positives and to verify which dependencies have already been patched or updated. A suppression list was then created in the suppression.xml file and the Dependency-Check is run again to confirm that the dependencies were properly suppressed.

## Industry Standard Best Practices

To adhere to industry standards and best practices, it is extremely important to implement security throughout the entire development life cycle. Waiting until the end of a project to concern yourself with security practices is extremely expensive and illogical. Data integrity should always be preserved by using a checksum to verify its contents. This ensures that the data is accurate and hasn’t been tampered with, reducing the chance of malware attacks that use steganography and other data hiding techniques. Input validation should always be excessively used to prevent the wrong information from being injected into the software, keeping the system safe from Denial-of-Service and Cross-Site Scripting attacks. Rate limiting should also be used to stop Denial-of Service attacks and unregulated consumption of system resources. This will allow the program to continually run at the desired speed and helps regulate access to a site as it grows larger and more scalable. Access Controls and Authentication will need to be carefully integrated throughout the system. This will allow only legitimate users access the site and will keep specific users and groups from being granted privileges beyond their needs. This is crucial for maintaining a secure and trustworthy system that users are comfortable using. Without the trust of the customer, there is no hope for a successful business.

**Citations**

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

**Spring. (2024). Spring Framework6.1.5. Spring Framework.**

[**https://spring.io/projects/spring-framework**](https://spring.io/projects/spring-framework)

**Oracle. (2023, May). Secure Coding Guidelines for Java SE.**

[**https://www.oracle.com/java/technologies/javase/seccodeguide.html**](https://www.oracle.com/java/technologies/javase/seccodeguide.html)

**SoftKraft. (2023, February 11). Web application architecture [complete guide & diagrams]. Medium.**

[**https://medium.com/@softkraft/web-application-architecture-complete-guide-diagrams-1b2d77fe3d2e**](https://medium.com/@softkraft/web-application-architecture-complete-guide-diagrams-1b2d77fe3d2e)

**Ruppert, S. (2024, January 16). Secure coding practices – input validation. Sven Ruppert.**

[**https://svenruppert.com/2023/12/13/secure-coding-practices-validation**](https://svenruppert.com/2023/12/13/secure-coding-practices-validation)**.**

**Microsoft. (2024). What is Data Protection? Microsoft security. What Is Data Protection? | Microsoft Security. (n.d.).**

[**https://www.microsoft.com/en-us/security/business/security-101/what-is-data-protection**](https://www.microsoft.com/en-us/security/business/security-101/what-is-data-protection)

**Wallis, C. (2024). How to perform a vulnerability assessment: Step-by-step. RSS.**

[**https://www.intruder.io/guides/vulnerability-assessment-made-simple-a-step-by-step-guide**](https://www.intruder.io/guides/vulnerability-assessment-made-simple-a-step-by-step-guide)

**Yasar, K. (2023, May 22). What is API security? Definition from TechTarget. App Architecture.**

[**https://www.techtarget.com/searchapparchitecture/definition/API-security**](https://www.techtarget.com/searchapparchitecture/definition/API-security)

**Benzel, T., Irvine, C., & Levin, T. (n.d.). (PDF) design principles for security. faculty.nps.edu.**

[**https://www.researchgate.net/publication/265224436\_Design\_Principles\_for\_Security**](https://www.researchgate.net/publication/265224436_Design_Principles_for_Security)

**Srinivas. (2020, December 1). How is cryptography used in applications? Infosec.**

[**https://www.infosecinstitute.com/resources/cryptography/how-is-cryptography-used-in-applications**](https://www.infosecinstitute.com/resources/cryptography/how-is-cryptography-used-in-applications)

**Esplanada, R. Y. (2021, July 27). @Controller and @RestController annotations in Spring Boot. Stack Abuse.**

[**https://stackabuse.com/controller-and-restcontroller-annotations-in-spring-boot**](https://stackabuse.com/controller-and-restcontroller-annotations-in-spring-boot)

**CSO Staff. (2023, September 12). Security and privacy laws, regulations, and compliance: The complete guide. CSO Online.**

[**https://www.csoonline.com/article/570281/csos-ultimate-guide-to-security-and-privacy-laws-regulations-and-compliance.html**](https://www.csoonline.com/article/570281/csos-ultimate-guide-to-security-and-privacy-laws-regulations-and-compliance.html)

**Recent Contributions To Cryptographic HASH Functions. (2009). Intel Technology Journal, 13(2).**

**Wikimedia Foundation. (2024, March 19). Secure hash algorithms. Wikipedia.**

[**https://en.wikipedia.org/wiki/Secure\_Hash\_Algorithms**](https://en.wikipedia.org/wiki/Secure_Hash_Algorithms)

**Keytool - Key and Certificate Management Tool. (2024). Oracle**

**GfG. (2021, September 10). A cryptographic introduction to hashing and hash collisions. GeeksforGeeks.**

[**https://www.geeksforgeeks.org/a-cryptographic-introduction-to-hashing-and-hash-collisions**](https://www.geeksforgeeks.org/a-cryptographic-introduction-to-hashing-and-hash-collisions)

**Dang, Q. H. (2015, August 4). Secure Hash Standard. NIST.**

[**https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf**](https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf)

**GfG. (2022, August 25). What Is Salted Password Hashing? GeeksforGeeks.**

[**https://www.geeksforgeeks.org/what-is-salted-password-hashing**](https://www.geeksforgeeks.org/what-is-salted-password-hashing)

**Piero. (2023, November 2). What is a cryptographic hash collision. Gate Learn.**

[**https://www.gate.io/learn/articles/what-is-a-cryptographic-hash-collision/800**](https://www.gate.io/learn/articles/what-is-a-cryptographic-hash-collision/800)

**Explaining SHA-512: A guide to cryptography. Info Institute. (2024, January 26).**

[**https://generalrelativity.org/blog/explaining-sha512-a-guide-to-cryptography**](https://generalrelativity.org/blog/explaining-sha512-a-guide-to-cryptography)

**Team, S. S. (2024, March 4). What is a Certificate Authority (CA)?. SSL.com.**

[**https://www.ssl.com/article/what-is-a-certificate-authority-ca**](https://www.ssl.com/article/what-is-a-certificate-authority-ca)

**Hoestje, J. (2018, June 14). OWASP dependency check for Vulnerability reporting. Keyhole Software.**

[**https://keyholesoftware.com/owasp-dependency-check-for-vulnerability-reporting**](https://keyholesoftware.com/owasp-dependency-check-for-vulnerability-reporting)

**Long, J. (2024, March 31). Suppressing False Positives. Dependency-Check – Suppressing False Positives.**

[**https://jeremylong.github.io/DependencyCheck/general/suppression.html**](https://jeremylong.github.io/DependencyCheck/general/suppression.html)