GLOBALRAIN

Practices for Secure Software Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	June 18, 2023	Jonathan Miller	Initial Revision

Client

Developer
Jonathan Miller

1. Algorithm Cipher

a. Provide a brief, high-level overview of the encryption algorithm cipher.

The cryptographic encryption algorithm that I decided to use is the SHA3 – 256 algorithm cipher. I decided to use this algorithm cipher because it is the latest standard for cryptographic encryption according to the Federal Information Processing Standards Publication 202. SHA3 is the latest technology used to ensure that the original message was not altered to a different message with the same hash value (collision resistance). (Pritzker & May, 2015) It is important to avoid collisions so that the original message does not get intercepted and replaced with a malicious message that has the same hash, or checksum, value.

I have decided to use the 256-bit version of SHA3 that will always output a 256-bit long string for the digest, or checksum, value. A 256-bit digest is sufficient for the message that was encrypted in this assignment.

b. What is the purpose of the cipher's hash functions and bit levels?

A cipher uses hash functions to turn plaintext into ciphertext. The cipher will translate the data into another set of data using a function to encrypt the data. For each data translation a key will be generated to decrypt the translated data.

This generated key will have a bit level to determine the length of the key. The higher the bit level, the more secure the algorithm will be due to the higher number of permutations that a key can take on.

c. Explain the use of random numbers, symmetric versus non-symmetric keys, and so on.

Encryption algorithms will use cryptographic pseudorandom number generators to create secure keys for decryption. This will prevent the unlikely event of predicting a key.

Algorithm ciphers can use either one or two encryption keys for encryption/decryption. Symmetric ciphers will create one key for both encryption and decryption, while asymmetric ciphers will utilize both a public and private key for encryption and decryption, respectively. (Manico et al., 2015)

d. Describe the history and current state of encryption algorithms.

Encryption algorithms in the past, such as DES (Data Encryption Standard) generated keys of only 56 bits in length. This allowed for brute force attacks for generating the correct key to decrypt data. Since there were only 2^56 different permutations of a DES key, computers were able guess the correct key by trying all possible permutations until the correct one was found.

To combat this, 3DES (Triple Data Encryption Standard) was formed. This basically performed DES three times on the data and led to key lengths of 168 bits. The block size of 3DES is only 64-bits and it led to block collision attacks if it was used to encrypt large amounts of data with the same key. (Triple des 2023)

Currently, as of FIPS 197, AES is the new encryption standard. This can produce key sizes of up to 256 bits in length, and it is currently impossible to brute force a key. However, with the emergence of AI technologies, AES keys may be brute forced in the near future.

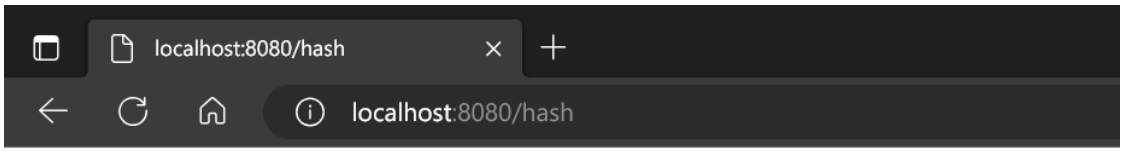
2. Certificate Generation

```
C:\Program Files\Java\jdk-20\bin>keytool.exe -printcert -file server.cer
Owner: CN=Jonathan Miller, OU=SNHU, O=SNHU, L=Urbana, ST=Ohio, C=US
Issuer: CN=Jonathan Miller, OU=SNHU, O=SNHU, L=Urbana, ST=Ohio, C=US
Serial number: f976eac4ace8f0e8
Valid from: Sun Jun 18 13:46:08 EDT 2023 until: Wed Jun 12 13:46:08 EDT 2024
Certificate fingerprints:
         SHA1: 94:3E:00:6D:D6:57:96:60:DA:09:7E:9C:CC:10:2E:16:41:39:6F:AC
         SHA256: 74:4A:62:51:D6:E8:57:D0:B8:22:40:E1:A0:A1:B7:DD:20:7D:2B:10:D2:23:87:02:BB:12:92:62:88:9F:B3:7B
Signature algorithm name: SHA384withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 51 87 05 E1 C0 84 4B FC   93 D7 52 5B 6D E3 E7 5C  Q.....K...R[m..\
0010: 6E E7 DD A0                                        n...
]
]
```

3. Deploy Cipher

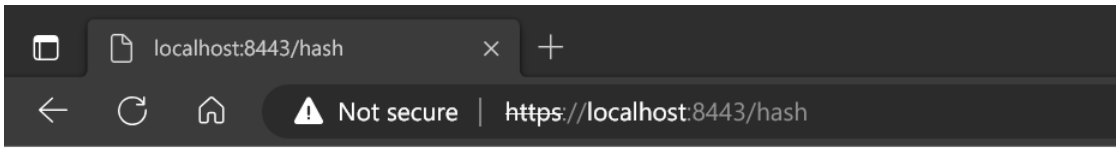


Data: Jonathan's Project 2 Check Sum!

Algorithm Cipher Used: SHA3-256

Checksum Value: 531e241f35c9d8dadd4f7f6c855142b1338385322f20df45dee324a279d8922e

4. Secure Communications



Data: Jonathan's Project 2 Check Sum!

Algorithm Cipher Used: SHA3-256

Checksum Value: 531e241f35c9d8dadd4f7f6c855142b1338385322f20df45dee324a279d8922e

5. Secondary Testing


```java
@RestController
class ServerController {
    /*
     * @description - Convert an array of bytes to a hex string.
     *
     * @param bytes - The array of bytes that will be converted to a hex string.
     * @return      - The converted hex string.
     */
    private static String bytesToHex(byte[] bytes) {
        // Define a StringBuilder object to build the hex string.
        StringBuilder hexStringBuilder = new StringBuilder();

        // For each byte in the bytes array...
        for (byte b : bytes) {
            // Use the format function to convert the current byte to a hex string value.
            hexStringBuilder.append(String.format("%02x", b));
        }

        // Return the converted hex string.
        return hexStringBuilder.toString();
    }

    @RequestMapping("/hash")
    public String myHash(){
        // The data to encrypt.
        String data = "Jonathan's Project 2 Check Sum!";

        // The algorithm cipher used to encrypt the data.
        String algorithmCipher = "SHA3-256";

        // Define a MessageDigest object that will be used to encrypt the data.
        MessageDigest messageObject = null;

        // Define a string that will hold the checksum value.
        String checksum = null;

        // Define a string to hold the text that will be displayed in the browser.
        String browserMessage = null;

        try {
            // Try to instantiate the message digest object with the defined algorithm cipher.
            messageObject = MessageDigest.getInstance(algorithmCipher);

            // Pass the data, as an array of bytes, to the created MessageDigest object.
            messageObject.update(data.getBytes());

            // Encrypt the data byte array from the MessageDigest object and store it in digest.
            byte[] digest = messageObject.digest();

            // Convert the digest byte array into a hex string and store it in the checksum string.
            checksum = bytesToHex(digest);
        } catch (NoSuchAlgorithmException e) {
            // If the algorithm cipher could not be used, print the stack trace.
            e.printStackTrace();
        }

        // Set the text to be displayed in the browser.
        browserMessage = "<p>Data: " + data + "\n" +
                         "<p>Algorithm Cipher Used: " + algorithmCipher + "\n" +
                         "<p>Checksum Value: " + checksum;

        return browserMessage;
    }
```

Project: ssl-server

com.snhu:ssl-server:0.0.1-SNAPSHOT

Scan Information (show all):

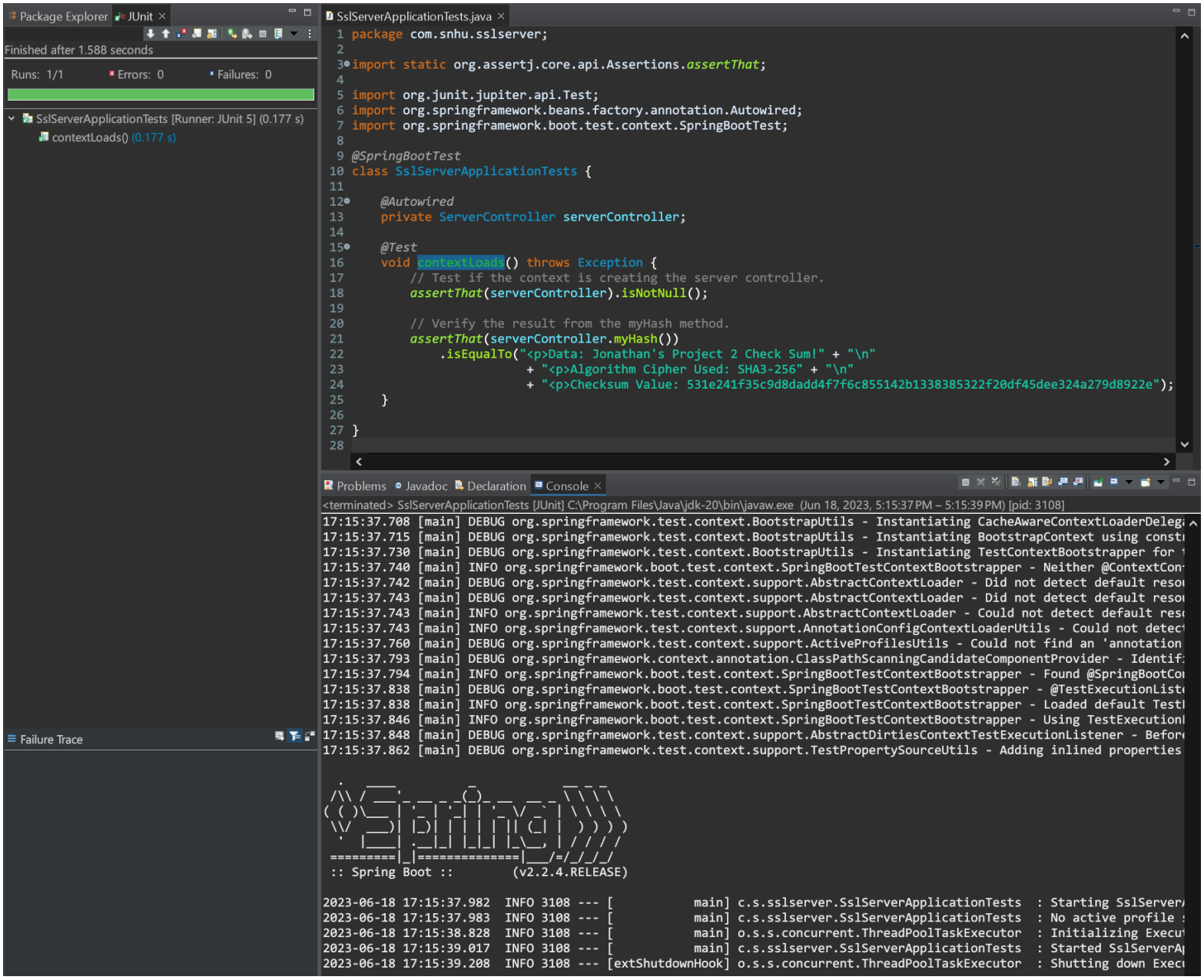
- *dependency-check version*: 8.1.0
- *Report Generated On*: Sun, 18 Jun 2023 15:04:00 -0400
- *Dependencies Scanned*: 49 (32 unique)
- *Vulnerable Dependencies*: 17
- *Vulnerabilities Found*: 104
- *Vulnerabilities Suppressed*: 0
- ...

Summary

Display: Showing Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
accessors-smart-1.2.jar	cpe:2.3:a:json-java_project:json-java:1.2:*:*:*:*:*:*:*	pkg:maven/net.minidev/accessors-smart@1.2	HIGH	1	Low	37
hibernate-validator-6.0.18.Final.jar	cpe:2.3:a:redhat:hibernate_validator:6.0.18:*:*:*:*:*:*:*	pkg:maven/org.hibernate.validator/hibernate-validator@6.0.18.Final	MEDIUM	1	Highest	34
jackson-core-2.10.2.jar	cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:*:*:*:*:*:*:* cpe:2.3:a:json-java_project:json-java:2.10.2:*:*:*:*:*:*:*	pkg:maven/com.fasterxml.jackson.core/jackson-core@2.10.2	HIGH	1	Low	47
jackson-databind-2.10.2.jar	cpe:2.3:a:fasterxml:jackson-databind:2.10.2:*:*:*:*:*:*:* cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:*:*:*:*:*:*:*	pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.10.2	HIGH	6	Highest	41
json-path-2.4.0.jar	cpe:2.3:a:json-java_project:json-java:2.4.0:*:*:*:*:*:*:*	pkg:maven/com.jayway.jsonpath/json-path@2.4.0	HIGH	1	Low	36
json-smart-2.3.jar	cpe:2.3:a:json-smart_project:json-smart:2.3:*:*:*:*:*:*:* cpe:2.3:a:json-smart_project:json-smart-v2:2.3:*:*:*:*:*:*:*	pkg:maven/net.minidev/json-smart@2.3	HIGH	3	Highest	47
log4j-api-2.12.1.jar	cpe:2.3:a:apache:log4j:2.12.1:*:*:*:*:*:*:*	pkg:maven/org.apache.logging.log4j/log4j-api@2.12.1	LOW	1	Highest	44
logback-core-1.2.3.jar	cpe:2.3:a:qos:logback:1.2.3:*:*:*:*:*:*:*	pkg:maven/ch.qos.logback/logback-core@1.2.3	MEDIUM	1	Highest	33
snakeyaml-1.25.jar	cpe:2.3:a:snakeyaml_project:snakeyaml:1.25:*:*:*:*:*:*:*	pkg:maven/org.yaml/snakeyaml@1.25	CRITICAL	8	Highest	46
spring-boot-2.2.4.RELEASE.jar	cpe:2.3:a:vmware:spring_boot:2.2.4:release:*:*:*:*:*:*	pkg:maven/org.springframework.boot/spring-boot@2.2.4.RELEASE	HIGH	2	Highest	39
spring-boot-starter-web-2.2.4.RELEASE.jar	cpe:2.3:a:vmware:spring_boot:2.2.4:release:*:*:*:*:*:* cpe:2.3:a:web_project:web:2.2.4:release:*:*:*:*:*:*	pkg:maven/org.springframework.boot/spring-boot-starter-web@2.2.4.RELEASE	HIGH	2	Highest	35
spring-core-5.2.3.RELEASE.jar	cpe:2.3:a:pivotal_software:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:springsource:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:vmware:spring_framework:5.2.3:release:*:*:*:*:*:*	pkg:maven/org.springframework/spring-core@5.2.3.RELEASE	CRITICAL*	11	Highest	36
spring-data-rest-webmvc-3.2.4.RELEASE.jar	cpe:2.3:a:pivotal_software:spring_data_rest:3.2.4:release:*:*:*:*:*:* cpe:2.3:a:vmware:spring_data_rest:3.2.4:release:*:*:*:*:*:*	pkg:maven/org.springframework.data/spring-data-rest-webmvc@3.2.4.RELEASE	MEDIUM	2	Highest	27
spring-web-5.2.3.RELEASE.jar	cpe:2.3:a:pivotal_software:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:springsource:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:vmware:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:web_project:web:5.2.3:release:*:*:*:*:*:*	pkg:maven/org.springframework/spring-web@5.2.3.RELEASE	CRITICAL*	12	Highest	34
spring-webmvc-5.2.3.RELEASE.jar	cpe:2.3:a:pivotal_software:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:springsource:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:vmware:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:web_project:web:5.2.3:release:*:*:*:*:*:*	pkg:maven/org.springframework/spring-webmvc@5.2.3.RELEASE	CRITICAL*	11	Highest	36
tomcat-embed-core-9.0.30.jar	cpe:2.3:a:apache:tomcat:9.0.30:*:*:*:*:*:*:* cpe:2.3:a:apache_tomcat:apache_tomcat:9.0.30:*:*:*:*:*:*:*	pkg:maven/org.apache.tomcat.embed/tomcat-embed-core@9.0.30	CRITICAL*	20	Highest	33
tomcat-embed-websocket-9.0.30.jar	cpe:2.3:a:apache:tomcat:9.0.30:*:*:*:*:*:*:* cpe:2.3:a:apache_tomcat:apache_tomcat:9.0.30:*:*:*:*:*:*:*	pkg:maven/org.apache.tomcat.embed/tomcat-embed-websocket@9.0.30	CRITICAL*	21	Highest	32

6. Functional Testing



7. Summary

a. Refer to the Vulnerability Assessment Process Flow Diagram. Highlight the areas of security that you addressed by refactoring the code.

The areas of security that apply to the Artemis Financial's application include the following:

- APIs – This application uses the spring boot framework to utilize RESTful APIs.
- Cryptography – This application uses a cryptographic cipher to encrypt data.
- Client/Server – This application deploys a secure server that a client connects to.
- Code Quality – This application relies on secure coding practices and patterns to mitigate security vulnerabilities.

b. Discuss your process for adding layers of security to the software application.

Artemis Financial needs their application to be without security vulnerabilities. To accomplish this, the areas of security that were affected by this application were identified. Then, the code was reviewed for potential security vulnerabilities.

Artemis Financial stated that they needed data to be encrypted. This then led to the implementation of a hash function that utilized the latest standard for cryptographic algorithm ciphers, SHA-256. The data that was sent to the client was encrypted using the SHA-256 algorithm cipher to create a checksum for the data. They also needed their code to be secured with the HTTPS protocol, so a self-signed key was created and utilized in the spring boot application. Lastly, functional testing was performed to ensure that the code was secure and functioning as expected.

8. Industry Standard Best Practices

a. Explain how you used industry standard best practices to maintain the software application's current security.

This application utilizes industry standard best practices to defend against cross-site scripting, protect sensitive data, and prevent injection attacks. To do this, the Spring Boot framework was utilized to implement a secure server using HTTPS and a self-signed key. The application involved the implementation of a RESTful API with secure endpoints using industry standard coding best practices. Functions that were not needed by the end user were marked as private so they could only be accessed by the server controller class member function for hashing.

b. Explain the value of applying industry standard best practices for secure coding to the company's overall wellbeing.

Applying industry standard best practices for secure coding is essential for a company's overall wellbeing. Without secure coding best practices, company assets such as money, card numbers, trade secrets, personal information, and passwords could be accessed by attackers.

These attackers could use this obtained information to steal money from the company or its consumers. They could also use obtained trade secrets to gain an advantage in the market. Or they could DDOS attack your website and create downtime that results in major losses in revenue.

If any of these attacks successfully occur, there could be major regulatory fines and penalties that could cost your company great sums of money as well as consumer confidence that may not ever be re-obtained. Therefore, security vulnerabilities could ultimately lead to the collapse of the

company, and secure coding practices should be taken into consideration throughout the entire SDLC.

9. Citations

Information Technology Laboratory , Pritzker, P., & May, W., SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions 1–2 (2015). Gaithersburg, MD; National Institute of Standards and Technology.

Manico, J., & Detlefsen, A. (2015). Symmetric and Asymmetric Cryptography. In *Iron-clad java: Building secure web applications*. essay, McGraw-Hill Education.

Wikimedia Foundation. (2023, April 18). *Triple des*. Wikipedia. https://en.wikipedia.org/wiki/Triple_DES