GLOBALRAIN

Practices for Secure Software Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	12/05/22	Quinnie Ho	

Client

ARTEMIS FINANCIAL

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
Quinnie Ho

1. Algorithm Cipher

The encryption algorithm cipher that I recommend deploying for Artemis Financial is the Advanced Encryption Standard (AES). This recommendation comes after considering security protection best practices to defend against various types of security attacks, the company's security vulnerabilities, and the most current government regulations.

The AES algorithm is used to encrypt and decrypt data. Encryption converts outgoing data to an unidentifiable format known as ciphertext, and then decryption is used on the receiving side to translate the ciphertext back to an identifiable format known as plaintext. This cryptography effort reduces the risk of hackers intercepting transmitted data and/or successfully impersonating to be the authorized receiver of the data.

Fundamentally, the AES algorithm meets the following general rules and security protection best practices for selecting ciphers (Manico & Detlefsen, 2015):

- Avoid ciphers that have "ANON" for authentication.
- Avoid ciphers that contain "NULL".
- Avoid ciphers that contain "EXPORT".
- Utilize ciphers with key sizes of 128 bits or more.
- Avoid ciphers that utilize "RC4", "DES", and "3DES".
- Opt for "ECDHE" and "DHE" for key agreement.

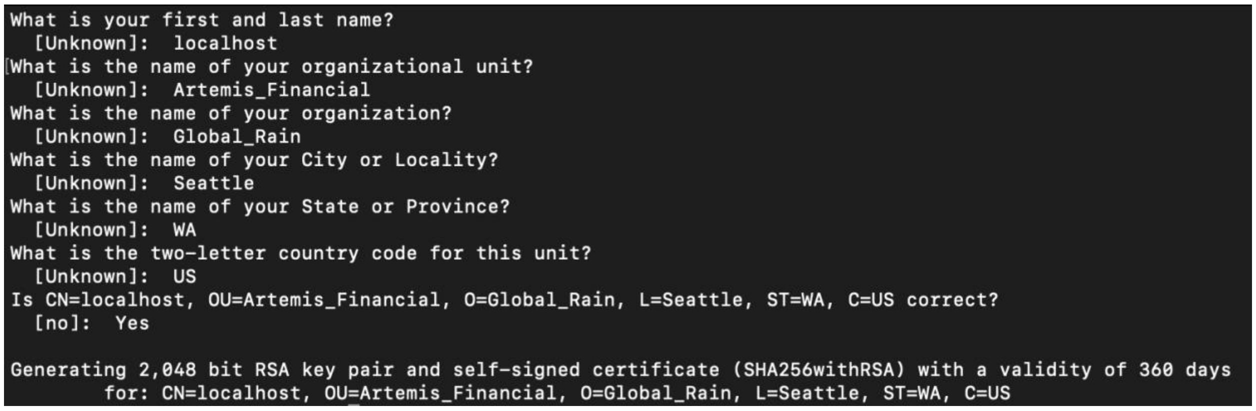
Moreover, this encryption algorithm also meets the most current government regulations since it is classified as a FIPS-approved (Federal Information Processing Standards) cryptographic algorithm that is issued by the National Institute of Standards and Technology and approved by the Secretary of Commerce (National Institute of Standards and Technology, 2001).

The main purpose of the cipher's hash functions is to determine if sent data was tampered with while in transit. This is done by, first, turning a plaintext input into a hash value output using a fixed length of bits. Then, the receiver runs their own hash function by using the known hash value output and encryption algorithm cipher. If the receiver's hash value output does not match the sender's hash value output, then there is a high risk that the data has been tampered with and should not be trusted (Crane, 2021). The recommended AES algorithm uses the fixed length of 128 bits, which is the minimum length listed in the general rule for security protection best practices for selecting ciphers.

The AES algorithm uses symmetric keys, which is the more commonly used type of cryptography. This type handles only encryption, and the term "symmetric keys" come from the fact that the same secret key is used to both encrypt and decrypt data (Manico & Detlefsen, 2015). Random numbers help provide an extra layer of protection by generating random numbers for the secret keys. Cryptography has come a long way, but it still coexists in a world where security threats are also growing and new attempts are constantly developed. Therefore, in addition to adopting the recommended encryption cipher and best practices mentioned in this document, it is best to also have a plan in place that details the frequency that Artemis Financial reviews and updates its cryptography practices.

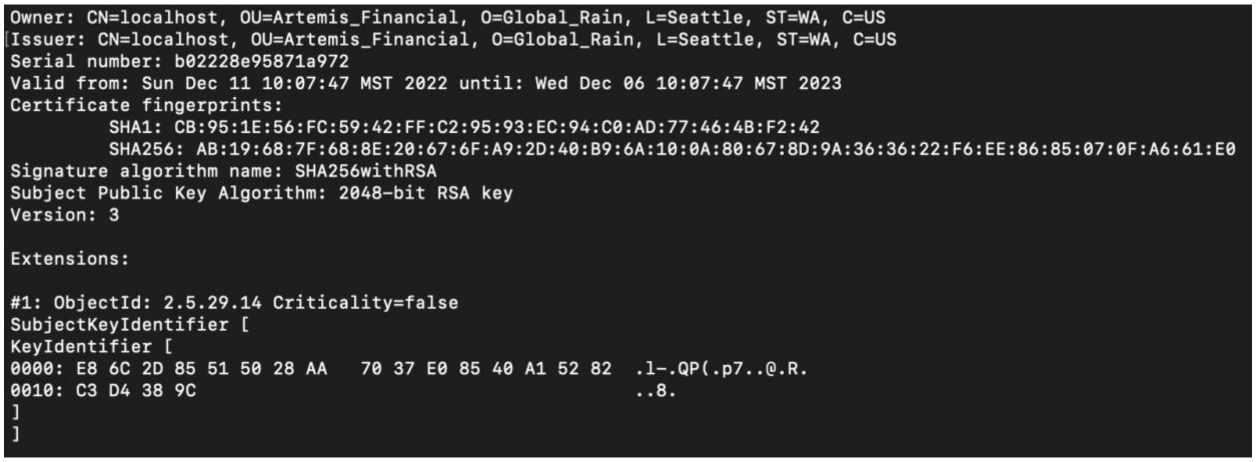
2. Certificate Generation

A self-signed certificate was generated using the following information:



```
What is your first and last name?
  [Unknown]:  localhost
What is the name of your organizational unit?
  [Unknown]:  Artemis_Financial
What is the name of your organization?
  [Unknown]:  Global_Rain
What is the name of your City or Locality?
  [Unknown]:  Seattle
What is the name of your State or Province?
  [Unknown]:  WA
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=localhost, OU=Artemis_Financial, O=Global_Rain, L=Seattle, ST=WA, C=US correct?
  [no]:  Yes

Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 360 days
        for: CN=localhost, OU=Artemis_Financial, O=Global_Rain, L=Seattle, ST=WA, C=US
```

The following is a screenshot of the CER file.

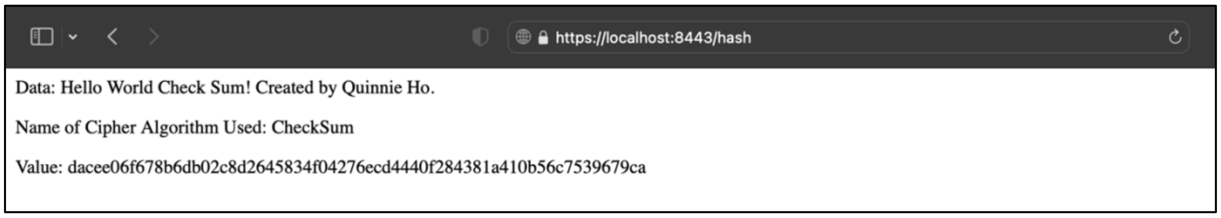


```
Owner: CN=localhost, OU=Artemis_Financial, O=Global_Rain, L=Seattle, ST=WA, C=US
Issuer: CN=localhost, OU=Artemis_Financial, O=Global_Rain, L=Seattle, ST=WA, C=US
Serial number: b02228e95871a972
Valid from: Sun Dec 11 10:07:47 MST 2022 until: Wed Dec 06 10:07:47 MST 2023
Certificate fingerprints:
         SHA1: CB:95:1E:56:FC:59:42:FF:C2:95:93:EC:94:C0:AD:77:46:4B:F2:42
         SHA256: AB:19:68:7F:68:8E:20:67:6F:A9:2D:40:B9:6A:10:0A:80:67:8D:9A:36:36:22:F6:EE:86:85:07:0F:A6:61:E0
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: E8 6C 2D 85 51 50 28 AA   70 37 E0 85 40 A1 52 82  .l-.QP(.p7..@.R.
0010: C3 D4 38 9C                                        ..8.
]
]
```

3. Deploy Cipher

The cryptographic hash algorithm SHA-256 was implemented in order to create a checksum verification. SHA-256 is a part of the Secure Hash Algorithm (SHA) family. The strength of the SHA-256 is that the hash value of this algorithm will always turn out to be 256 bits.

Here is a screenshot of the checksum verification that shows my name and a unique data string that has been created:



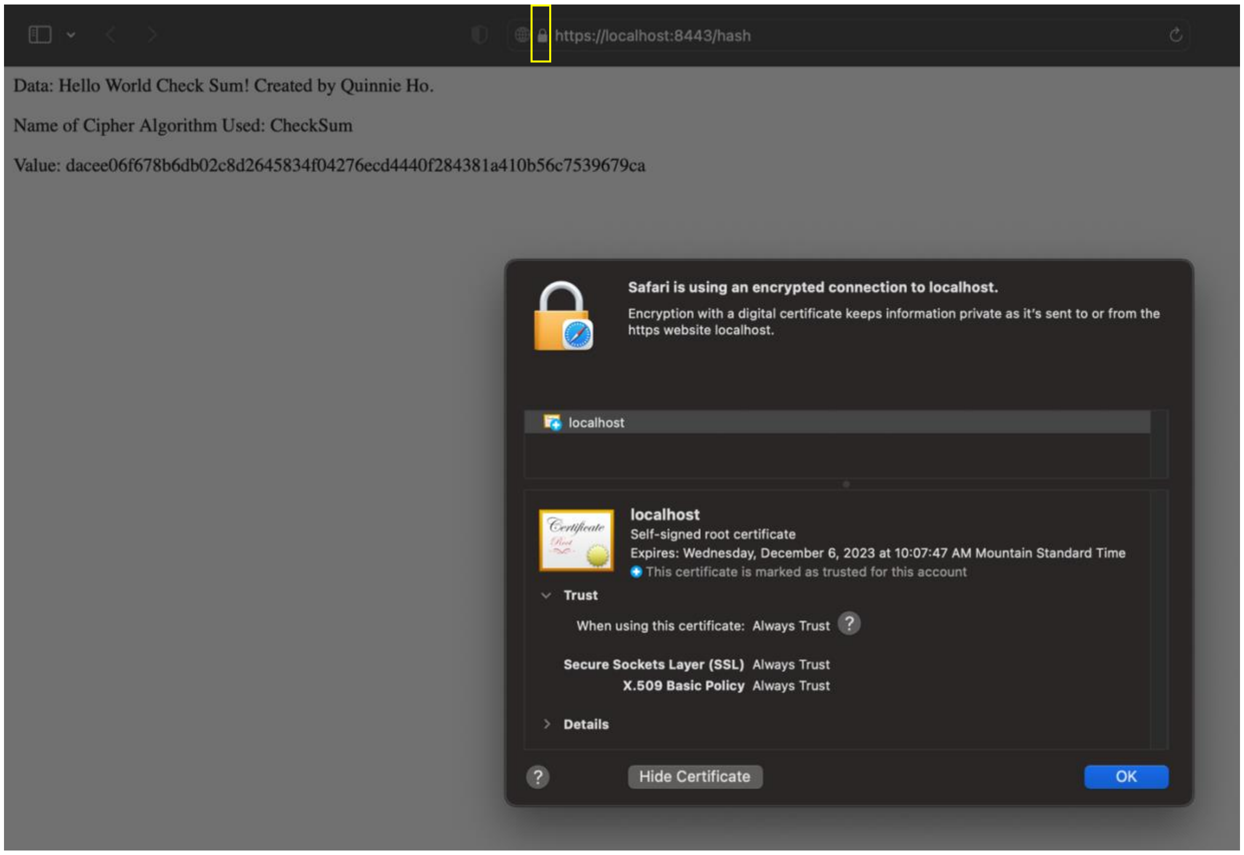
Data: Hello World Check Sum! Created by Quinnie Ho.

Name of Cipher Algorithm Used: CheckSum

Value: dacee06f678b6db02c8d2645834f04276ecd4440f284381a410b56c7539679ca

4. Secure Communications

The code in the "application.properties" file was refactor to covert HTTP to the HTTPS protocol. The screenshot below shows a padlock in the URL of the web browser (outlined in yellow) that symbolizes a secure webpage. Moreover, the screenshot shows that the self-signed certificate created for this project and discussed in the section above is trusted.


https://localhost:8443/hash

Data: Hello World Check Sum! Created by Quinnie Ho.

Name of Cipher Algorithm Used: CheckSum

Value: dacee06f678b6db02c8d2645834f04276ecd4440f284381a410b56c7539679ca

Safari is using an encrypted connection to localhost.

Encryption with a digital certificate keeps information private as it's sent to or from the https website localhost.

localhost

localhost
Self-signed root certificate
Expires: Wednesday, December 6, 2023 at 10:07:47 AM Mountain Standard Time
This certificate is marked as trusted for this account

Trust

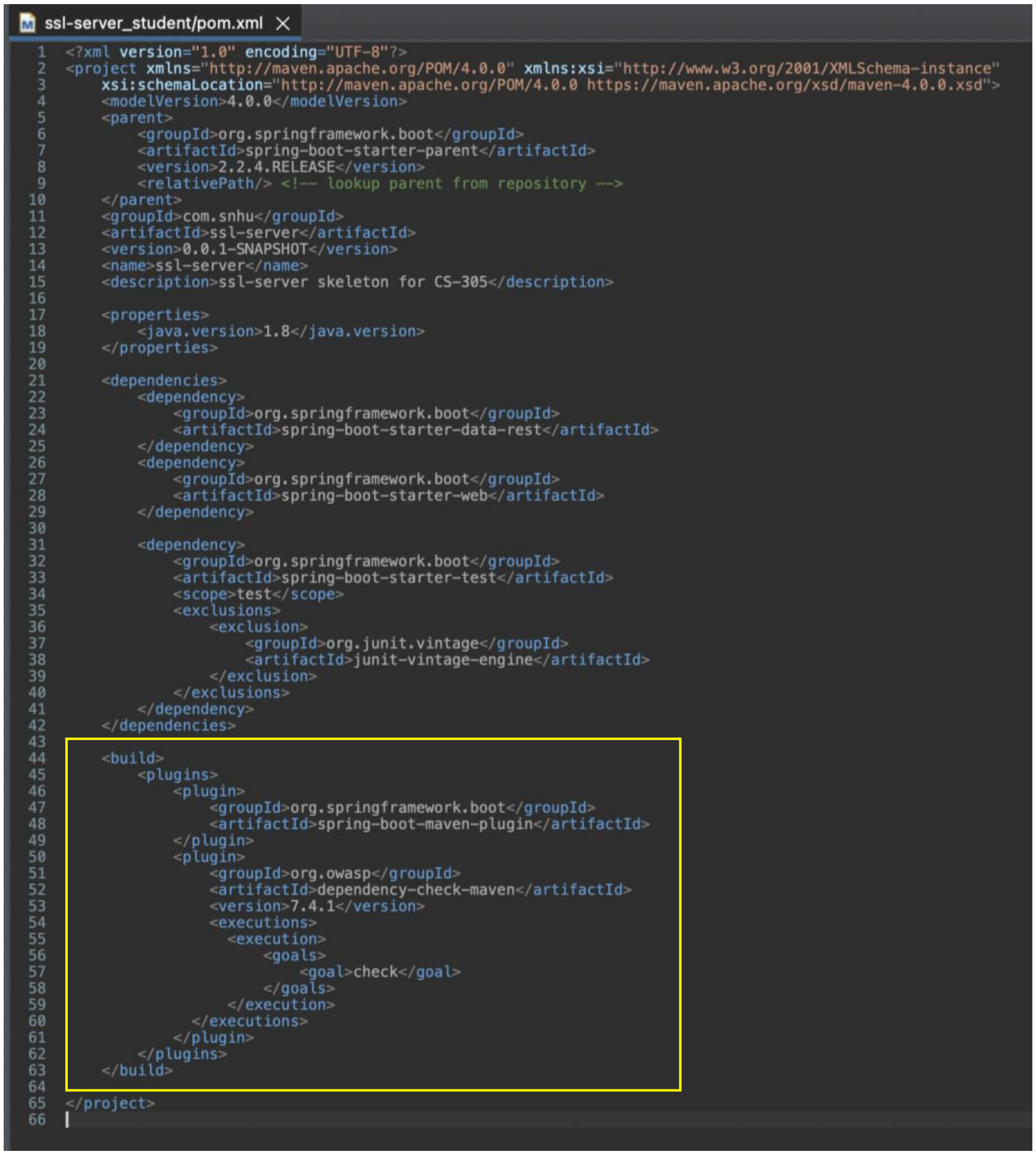
When using this certificate: Always Trust

Secure Sockets Layer (SSL) Always Trust
X.509 Basic Policy Always Trust

Details

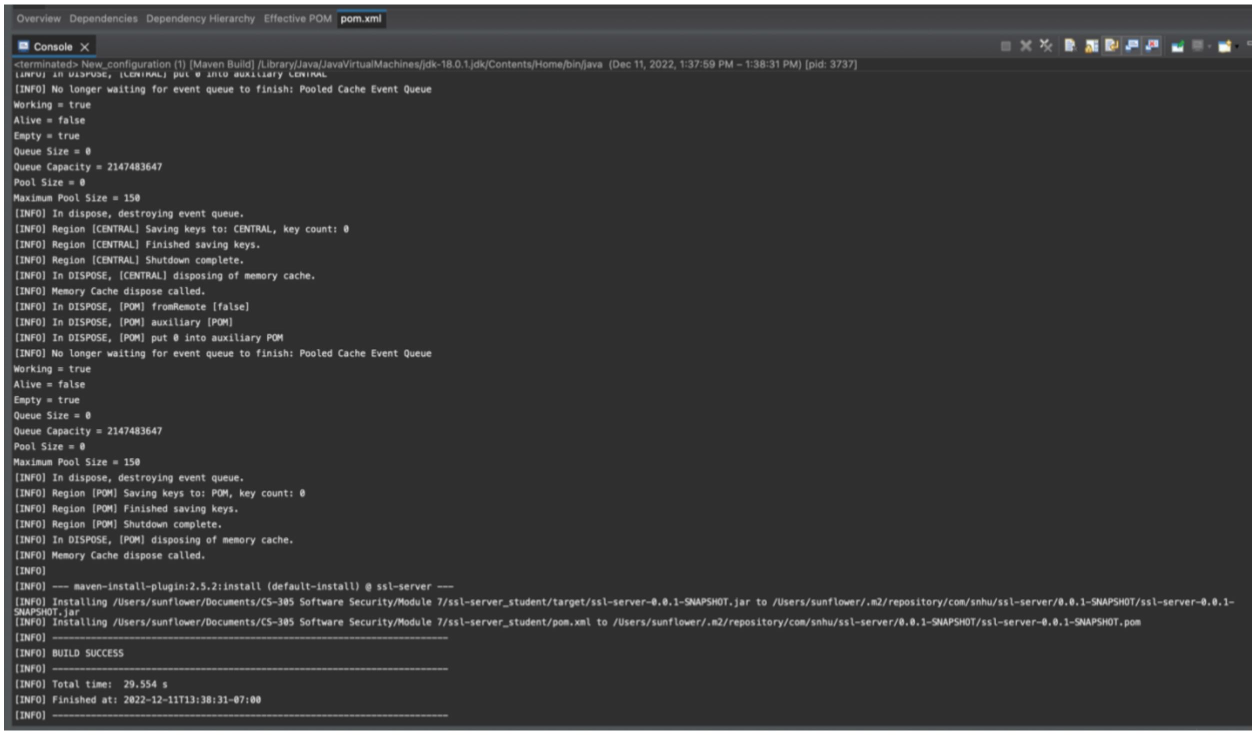
5. Secondary Testing

The version number of the dependency check was updated to the current number 7.4.1 that is available at the time of this implementation and documentation. The screenshot below shows the code in the "pom.xml" file that includes this updated version number in the box outlined in yellow.



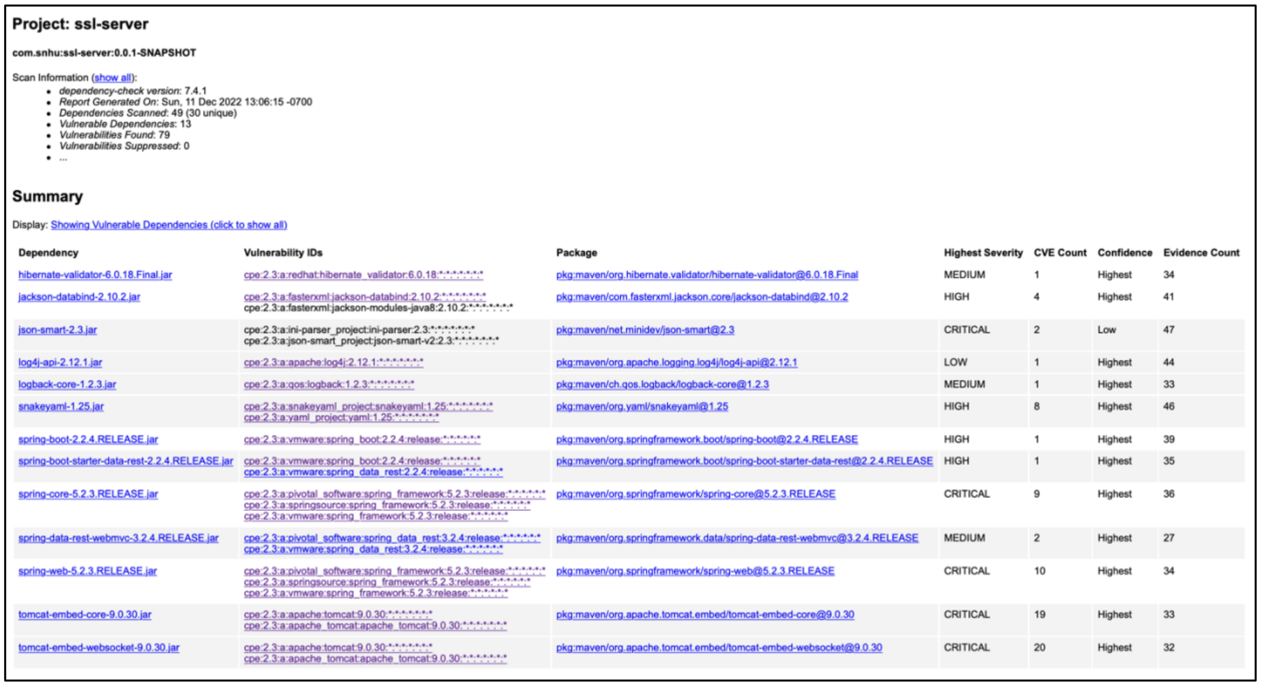
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.4.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.snhu</groupId>
    <artifactId>ssl-server</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>ssl-server</name>
    <description>ssl-server skeleton for CS-305</description>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-rest</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
            <exclusions>
                <exclusion>
                    <groupId>org.junit.vintage</groupId>
                    <artifactId>junit-vintage-engine</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
            <plugin>
                <groupId>org.owasp</groupId>
                <artifactId>dependency-check-maven</artifactId>
                <version>7.4.1</version>
                <executions>
                  <execution>
                      <goals>
                          <goal>check</goal>
                      </goals>
                  </execution>
                </executions>
            </plugin>
        </plugins>
    </build>

</project>
```

The screenshot below shows the result of the refactored code after executing without errors.



```
<terminated> New_configuration (1) [Maven Build] /Library/Java/JavaVirtualMachines/jdk-18.0.1.jdk/Contents/Home/bin/java  (Dec 11, 2022, 1:37:59 PM – 1:38:31 PM) [pid: 3737]
[INFO] In DISPOSE, [CENTRAL] put 0 into auxiliary CENTRAL
[INFO] No longer waiting for event queue to finish: Pooled Cache Event Queue
Working = true
Alive = false
Empty = true
Queue Size = 0
Queue Capacity = 2147483647
Pool Size = 0
Maximum Pool Size = 150
[INFO] In dispose, destroying event queue.
[INFO] Region [CENTRAL] Saving keys to: CENTRAL, key count: 0
[INFO] Region [CENTRAL] Finished saving keys.
[INFO] Region [CENTRAL] Shutdown complete.
[INFO] In DISPOSE, [CENTRAL] disposing of memory cache.
[INFO] Memory Cache dispose called.
[INFO] In DISPOSE, [POM] fromRemote [false]
[INFO] In DISPOSE, [POM] auxiliary [POM]
[INFO] In DISPOSE, [POM] put 0 into auxiliary POM
[INFO] No longer waiting for event queue to finish: Pooled Cache Event Queue
Working = true
Alive = false
Empty = true
Queue Size = 0
Queue Capacity = 2147483647
Pool Size = 0
Maximum Pool Size = 150
[INFO] In dispose, destroying event queue.
[INFO] Region [POM] Saving keys to: POM, key count: 0
[INFO] Region [POM] Finished saving keys.
[INFO] Region [POM] Shutdown complete.
[INFO] In DISPOSE, [POM] disposing of memory cache.
[INFO] Memory Cache dispose called.
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ ssl-server ---
[INFO] Installing /Users/sunflower/Documents/CS-305 Software Security/Module 7/ssl-server_student/target/ssl-server-0.0.1-SNAPSHOT.jar to /Users/sunflower/.m2/repository/com/snhu/ssl-server/0.0.1-SNAPSHOT/ssl-server-0.0.1-
SNAPSHOT.jar
[INFO] Installing /Users/sunflower/Documents/CS-305 Software Security/Module 7/ssl-server_student/pom.xml to /Users/sunflower/.m2/repository/com/snhu/ssl-server/0.0.1-SNAPSHOT/ssl-server-0.0.1-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  29.554 s
[INFO] Finished at: 2022-12-11T13:38:31-07:00
[INFO] ------------------------------------------------------------------------
```

The screenshot of the dependency-check report is below.



Project: ssl-server

com.snhu:ssl-server:0.0.1-SNAPSHOT

Scan Information (show all):

- *dependency-check version*: 7.4.1
- *Report Generated On*: Sun, 11 Dec 2022 13:06:15 -0700
- *Dependencies Scanned*: 49 (30 unique)
- *Vulnerable Dependencies*: 13
- *Vulnerabilities Found*: 79
- *Vulnerabilities Suppressed*: 0
- ...

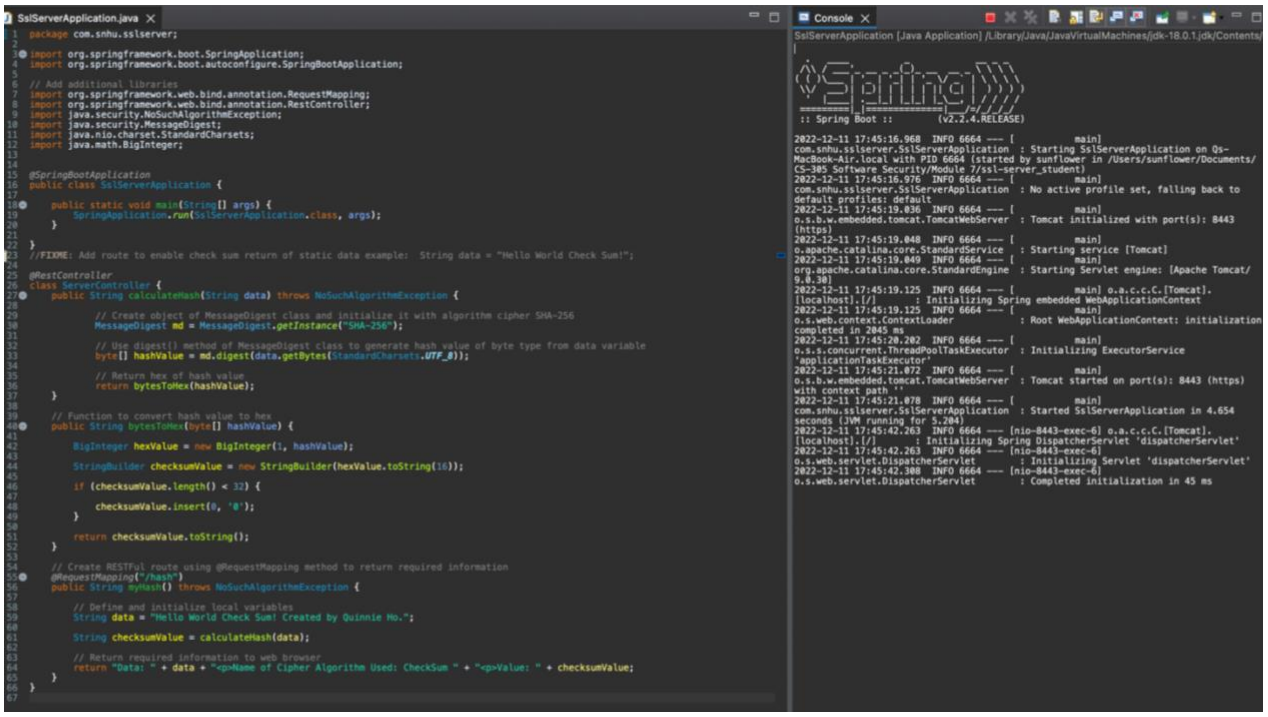
Summary

Display: Showing Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
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-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	4	Highest	41
json-smart-2.3.jar	cpe:2.3:a:ini-parser_project:ini-parser:2.3:*:*:*:*:*:*:* cpe:2.3:a:json-smart_project:json-smart-v2:2.3:*:*:*:*:*:*:*	pkg:maven/net.minidev/json-smart@2.3	CRITICAL	2	Low	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:*:*:*:*:*:*:* cpe:2.3:a:yaml_project:yaml:1.25:*:*:*:*:*:*:*	pkg:maven/org.yaml/snakeyaml@1.25	HIGH	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	1	Highest	39
spring-boot-starter-data-rest-2.2.4.RELEASE.jar	cpe:2.3:a:vmware:spring_boot:2.2.4:release:*:*:*:*:* cpe:2.3:a:vmware:spring_data_rest:2.2.4:release:*:*:*:*:*	pkg:maven/org.springframework.boot/spring-boot-starter-data-rest@2.2.4.RELEASE	HIGH	1	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	9	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:*:*:*:*:*	pkg:maven/org.springframework/spring-web@5.2.3.RELEASE	CRITICAL	10	Highest	34
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	19	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	20	Highest	32

6. Functional Testing

A functioning test was conducted by manually reviewing the code. The screenshot below shows the refactored code for the “SslServerApplicatioin.java” file in the left window and the execution of the code without errors in the right window.



```java
package com.snhu.sslserver;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

// Add additional libraries
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import java.security.NoSuchAlgorithmException;
import java.security.MessageDigest;
import java.nio.charset.StandardCharsets;
import java.math.BigInteger;


@SpringBootApplication
public class SslServerApplication {

    public static void main(String[] args) {
        SpringApplication.run(SslServerApplication.class, args);
    }

}
//FIXME: Add route to enable check sum return of static data example:  String data = "Hello World Check Sum!";

@RestController
class ServerController {
    public String calculateHash(String data) throws NoSuchAlgorithmException {

            // Create object of MessageDigest class and initialize it with algorithm cipher SHA-256
            MessageDigest md = MessageDigest.getInstance("SHA-256");

            // Use digest() method of MessageDigest class to generate hash value of byte type from data variable
            byte[] hashValue = md.digest(data.getBytes(StandardCharsets.UTF_8));

            // Return hex of hash value
            return bytesToHex(hashValue);
    }

    // Function to convert hash value to hex
    public String bytesToHex(byte[] hashValue) {

        BigInteger hexValue = new BigInteger(1, hashValue);

        StringBuilder checksumValue = new StringBuilder(hexValue.toString(16));

        if (checksumValue.length() < 32) {

            checksumValue.insert(0, '0');
        }

        return checksumValue.toString();
    }

    // Create RESTFul route using @RequestMapping method to return required information
    @RequestMapping("/hash")
    public String myHash() throws NoSuchAlgorithmException {

        // Define and initialize local variables
        String data = "Hello World Check Sum! Created by Quinnie Mo.";

        String checksumValue = calculateHash(data);

        // Return required information to web browser
        return "Data: " + data + "<p>Name of Cipher Algorithm Used: CheckSum " + "<p>Value: " + checksumValue;
    }
}
```

```
SslServerApplication [Java Application] /Library/Java/JavaVirtualMachines/jdk-18.0.1.jdk/Contents/

 :: Spring Boot ::        (v2.2.4.RELEASE)

2022-12-11 17:45:16.968  INFO 6664 --- [           main]
com.snhu.sslserver.SslServerApplication  : Starting SslServerApplication on Qs-
MacBook-Air.local with PID 6664 (started by sunflower in /Users/sunflower/Documents/
CS-305 Software Security/Module 7/ssl-server_student)
2022-12-11 17:45:16.976  INFO 6664 --- [           main]
com.snhu.sslserver.SslServerApplication  : No active profile set, falling back to
default profiles: default
2022-12-11 17:45:19.036  INFO 6664 --- [           main]
o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8443
(https)
2022-12-11 17:45:19.048  INFO 6664 --- [           main]
o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-12-11 17:45:19.049  INFO 6664 --- [           main]
org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/
9.0.30]
2022-12-11 17:45:19.125  INFO 6664 --- [           main] o.a.c.c.C.[Tomcat].
[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-12-11 17:45:19.125  INFO 6664 --- [           main]
o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization
completed in 2045 ms
2022-12-11 17:45:20.202  INFO 6664 --- [           main]
o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService
'applicationTaskExecutor'
2022-12-11 17:45:21.072  INFO 6664 --- [           main]
o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8443 (https)
with context path ''
2022-12-11 17:45:21.078  INFO 6664 --- [           main]
com.snhu.sslserver.SslServerApplication  : Started SslServerApplication in 4.654
seconds (JVM running for 5.204)
2022-12-11 17:45:42.263  INFO 6664 --- [nio-8443-exec-6] o.a.c.c.C.[Tomcat].
[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-12-11 17:45:42.263  INFO 6664 --- [nio-8443-exec-6]
o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2022-12-11 17:45:42.308  INFO 6664 --- [nio-8443-exec-6]
o.s.web.servlet.DispatcherServlet        : Completed initialization in 45 ms
```

7. Summary

There was a total of three instances when the code was refactored in this project. The first was to create a cryptographic hash algorithm. This action was taken to produce a checksum that can verify transferred data between Artemis Financial and their clients. Therefore, the area of security that is addressed by refactoring this code is cryptography.

The second refactoring of codes was to covert HTTP to the HTTPS protocol in order to secure communications. This was done by adding in code that incorporated information from the self-signed certificate created with the SSL keystore. This action allowed the server and client to transfer data in a secured manner, therefore, the refactoring in this scenario addressed the client/server security vulnerability.

The final code that was refactored was to update the version number of the dependency-check maven to the most current version. This ensures that the code complies with the most up-to-date software security enhancements. The APIs security vulnerability was addressed here since the dependency-check maven is external to Artemis Financial’s system. APIs are required to integrate with third-party software systems, therefore, it is good practice to ensure that secured API interactions are in place to reduce the risk of an unexpected third-party exposure to our client’s application. Ensuring that versions of external software systems are up to date is an easy way to reduce this security risk.

Last, but not least, the functional testing method of manually reviewing the software application's code is a good practice to address the code error security vulnerability. This is an important area of security to address in order to reduce the risk of errored codes being susceptible to security breaches, especially for codes related to API. A zipped project folder is attached to this document and contains the refactored codes outlined above.

Reviewing and adding the layers of security above one-by-one makes it easier to capture all the relevant security vulnerabilities related to Artemis Financial's application, and then ensure that they are addressed according to industry-standard best practices.

8. Industry Standard Best Practices

There are a few industry standard best practices that were implemented to enhance and maintain the security of Artemis Financial's software application. One was deploying the SHA-256 cryptographic hash algorithm to avoid collisions. Collisions have been found with previous popular hash algorithm such as MD5 and SHA-1 (Hoffman, 2019), however, SHA-256 is still collision-resistant and trusted enough to be supported and used by the United States Department of Defense (Khifer, 2022).

Another best practice that was implemented is ensuring that the application is synced with the most current version of external software and systems. This allows our client's application to benefit from updated security features. Moreover, it ensures that their application is not exposed to security breaches due to old, unmaintained versions of the software.

Running and reviewing the dependency-check report is also a best practice, because it gives a good overview of the vulnerabilities that are associated with the application. Moreover, the dependency check maven pulls from the Common Vulnerabilities and Exposures database as well as the National Vulnerability Database. These two databases are continuously updated with current security vulnerabilities and provide industry standard guidelines on how to deal with them. Therefore, performing a dependency-check ensures that any new security vulnerabilities added to Artemis Financial's application are made aware of as soon as possible, to mitigate the concerns before they escalate.

Writing concise, clear, and well documented code are also important industry standard best practices that should not be overlooked. This ensures that codes are easy to read and interpreted, which reduces code errors when changes need to be made to the original code. Moreover, well documented code makes it easier to troubleshoot by reserve engineering and, especially, when different developers are contributing to the same program.

Finally, it is important to ensure that there is a plan in place to review industry standard best practices as they evolve, and then implement any relevant updates to Artemis Financial's application. Overall, the value of applying industry standard best practices for secure coding to the company's overall wellbeing, is that it provides a proactive way to secure the company's software and systems.

References

Crane, C. (2021, January 25). *What is a hash function in cryptography? A beginner's guide*. The SSL Store. https://www.thesslstore.com/blog/what-is-a-hash-function-in-cryptography-a-beginners-guide/

Hoffman, C. (2019, September 30). *What is a checksum (and why should you care)?*. How-To Geek. https://www.howtogeek.com/363735/what-is-a-checksum-and-why-should-you-care/

Khifer, Z. (2022, March). *Checksum verification using an encryption algorithm cipher that avoids collisions*. Brogramo. https://brogramo.com/checksum-verification-using-an-encryption-algorithm-cipher-that-avoids-collisions/

Manico, J., & Detlefsen, A. (2015). *Iron-clad java: Building secure web applications*. McGraw-Hill Education. https://learning.oreilly.com/library/view/iron-clad-java/9780071835886/ch06.html#ch06lev2sec3

National Institute of Standards and Technology. (2001). *Advanced encryption standard (AES)* (Publication No. 197). Federal Information Processing Standards Publications. https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf