GLOBALRAIN

Practices for Secure Software Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	**10/14/2023**	**Savion Peebles**	**Refactored code and performed a variety of security measures for Artemis.**

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.

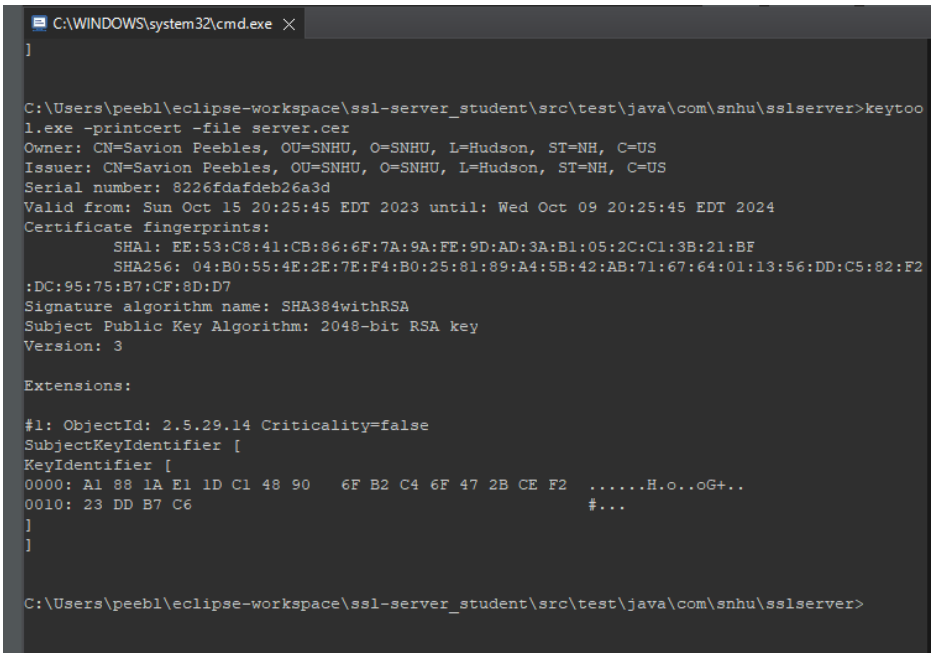
Developer
Savion Peebles

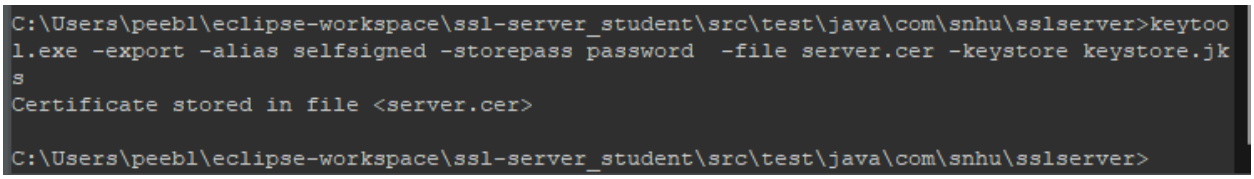
1. Algorithm Cipher

The algorithm that I will be choosing is SHA-256 because Artemis is a financial company that relies on their security software in order to protect their customers sensitive data safe from hackers. Since we will be offering programs around the world it is critical to make sure that we have the best possible encryption we can have. SHA-256 is a good recommendation because it allows critical information to be guarded completely from any outside influence SHA -256's hash functions are created randomly along with the bit levels, when the hash function is made the input value is compressed before it is put into use. Our hash value is the name of our compressed data, and the length of our encryption is based on the bit levels.

Symmetric keys are seen as the most simple form of encryption there is, they use plain text and then take a key that encrypts it. The keys are within the AES-256 standards, which is the most current form of encryption. On the other hand, Asymmetric keys are used to secure internet connections and a variety of other things, they are considered to be more secure than symmetric keys due to the fact that they use two keys instead of one. Simple forms of encryption have been used since ancient times such as The Caesar Shift Cypher which was used by the Roman Army to send encoded messages that normal citizens could not understand. Another major point in history that encryption was used was the Enigma Code in World War II that was created by the Nazis and then later deciphered by allied forces.

2. Certificate Generation
Insert a screenshot below of the CER file.





3. Deploy Cipher

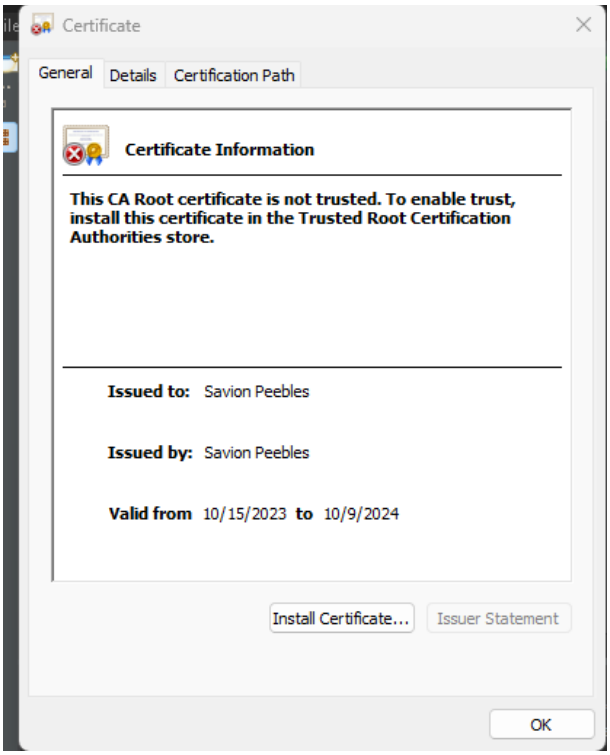
Insert a screenshot below of the checksum verification.

```java
//FIXME: Add route to enable check sum return of static data example:  String data = "Hello World Check Sum!";
@RestController
class ServerController{

    public static String getHash(String name) throws NoSuchAlgorithmException
    {
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[] hash =  md.digest(name.getBytes(StandardCharsets.UTF_8));
        BigInteger number = new BigInteger(1, hash);
        StringBuilder hexString = new StringBuilder(number.toString(16));
        while (hexString.length() < 32)
        {
            hexString.insert(0, '0');
        }
        return hexString.toString();

    }
    @RequestMapping("/Hash")
    public String algorithm(){

        String data = "Savion Peebles";
        String hash = "";
        try {
            hash = getHash(data);


        } catch (NoSuchAlgorithmException e) {

            e.printStackTrace();
        }


        return "<h3>Data</h3>"+data+" : <h3>Used Hash </h3> <br>SHA-256 <h3>"+"Result:  "+hash+"</h3>";
    }
}
```

While going through this portion of the project I faced some issues that prevented me from actually gaining my checksum verification. I believe I am on the right track based off of research and reviewing supporting materials, but I kept receiving “Failed to initialize component” and I could not figure it out. So, I submitted my code in hope that it would at least show you my thought process.

4. Secure Communications

Insert a screenshot below of the web browser that shows a secure webpage.



As I mentioned above I was having some issues with checksum verification But I managed to get my certificate, I took what I believed to be the next steps based off of our previous assignments and changed the properties file in order to go from http to https.

```
2
3 server.port=8443
4 server.ssl.key-alias=S
5 server.ssl.key-store-password=password
6 server.ssl.key-store=keystore.jks
7 server.ssl.key-store-provider=SUN
8 server.ssl.key-store-type=JKS
9
10
```

5. Secondary Testing

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

```java
package com.snhu.sslserver;

import java.math.BigInteger;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

import org.apache.catalina.connector.Connector;
import org.apache.tomcat.util.descriptor.web.SecurityCollection;
import org.apache.tomcat.util.descriptor.web.SecurityConstraint;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.web.embedded.tomcat.TomcatServletWebServerFactory;
import org.springframework.boot.web.servlet.server.ServletWebServerFactory;
import org.springframework.context.annotation.Bean;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import org.apache.catalina.Context;


@SpringBootApplication
public class SslServerApplication {

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

    @Bean
    public ServletWebServerFactory servletContainer() {
```

```java
    public static String getHash(String name) throws NoSuchAlgorithmException
    {
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[] hash =  md.digest(name.getBytes(StandardCharsets.UTF_8));
        BigInteger number = new BigInteger(1, hash);
        StringBuilder hexString = new StringBuilder(number.toString(16));
        while (hexString.length() < 32)
        {
            hexString.insert(0, '0');
        }
        return hexString.toString();

    }
    @RequestMapping("/Hash")
    public String algorithm(){

        String data = "Savion Peebles";
        String hash = "";
        try {
            hash = getHash(data);


        } catch (NoSuchAlgorithmException e) {

            e.printStackTrace();
        }


        return "<h3>Data</h3>"+data+" : <h3>Used Hash </h3> <br>SHA-256 <h3>"+"Result:  "+hash+"</h3>";
    }
}
```

Javadoc | Declaration | Console | Terminal | Debug | Coverage

```
<terminated> New_configuration (3) [Maven Build] C:\Program Files\Java\jdk-20\bin\javaw.exe  (Oct 15, 2023, 11:00:35 PM – 11:04:15 PM) [pid: 24064]
[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
```

Project: ssl-server

com.snhu:ssl-server:0.0.1-SNAPSHOT

Scan Information (show all):

- *dependency-check version*: 8.4.0
- *Report Generated On*: Sun, 15 Oct 2023 23:04:14 -0400
- *Dependencies Scanned*: 49 (31 unique)
- *Vulnerable Dependencies*: 15
- *Vulnerabilities Found*: 106
- *Vulnerabilities Suppressed*: 0
- ...

Summary

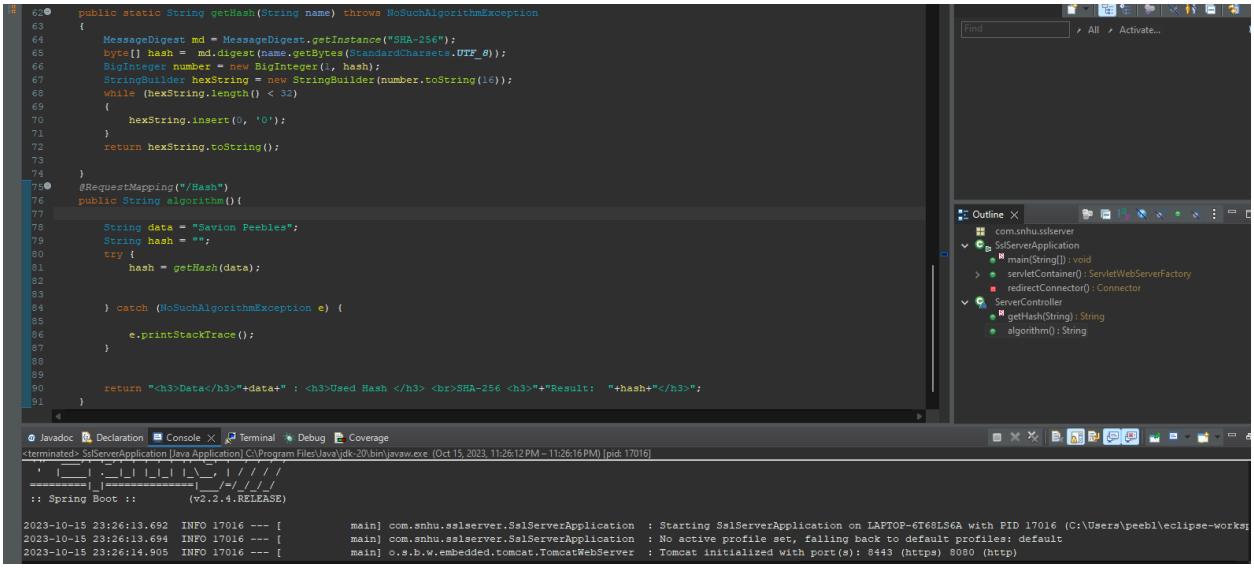
Display: Showing Vulnerable Dependencies (click to show all)

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	32
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	39
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	45
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	42
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	31
snakeyaml-1.25.jar	cpe:2.3:a:snakeyaml_project:snakeyaml:1.25:*:*:*:*:*:*:*	pkg:maven/org.yaml/snakeyaml@1.25	CRITICAL	8	Highest	44
spring-boot-2.2.4.RELEASE.jar	cpe:2.3:a:spring:spring:2.2.4:release:*:*:*:*:*:* cpe:2.3:a:vmware:spring:2.2.4:release:*:*:*:*:*:* cpe:2.3:a:vmware:spring_boot:2.2.4:release:*:*:*:*:*:*	pkg:maven/org.springframework.boot/spring-boot@2.2.4.RELEASE	CRITICAL	3	Highest	39
spring-boot-starter-web-2.2.4.RELEASE.jar	cpe:2.3:a:spring:spring:2.2.4:release:*:*:*:*:*:* cpe:2.3:a:vmware:spring:2.2.4:release:*:*:*:*:*:* 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	CRITICAL	3	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:spring:spring:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:springsource:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:vmware:spring: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:spring:spring:3.2.4:release:*:*:*:*:*:* cpe:2.3:a:vmware:spring: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-hateoas-1.0.3.RELEASE.jar	cpe:2.3:a:spring:spring:1.0.3:release:*:*:*:*:*:* cpe:2.3:a:vmware:spring:1.0.3:release:*:*:*:*:*:* cpe:2.3:a:vmware:spring_hateoas:1.0.3:release:*:*:*:*:*:*	pkg:maven/org.springframework.hateoas/spring-hateoas@1.0.3.RELEASE	MEDIUM	1	Highest	43
spring-web-5.2.3.RELEASE.jar	cpe:2.3:a:pivotal_software:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:spring:spring:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:springsource:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:vmware:spring: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:spring:spring:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:springsource:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:vmware:spring: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*	21	Highest	30
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*	22	Highest	30

* indicates the dependency has a known exploited vulnerability

6. Functional Testing

Insert a screenshot below of the refactored code executed without errors.



Code

Pom file after the version was changed to 8.4.0

```
      <plugins>
           <plugin>
            <groupId>org.owasp</groupId>
            <artifactId>dependency-check-maven</artifactId>
            <version>8.4.0</version>
            <executions>
                <execution>
                    <goals>
                        <goal>check</goal>
                    </goals>
                </execution>
            </executions>
           </plugin>
      </plugins>
  </build>

project>
```

7. Summary

After I refactored my code, I then attempted to encrypt it by using hash functions. This allowed for the security of the program to be increased mightily compared to what it was. In order to make sure that customers felt safe using the program and that data was watched over more effectively I created a certificate. The next step was making sure the system was fully monitored when it came to security vulnerabilities and doing a dependency check gave me a view on what vulnerabilities we were dealing with and how I should mitigate their risks, this increased the overall security of the program immensely. All the needs that Artemis had made clear that they need addressed were taken care of and the program and company will greatly benefit from it both in a business and security perspective.

8. Industry Standard Best Practices

Using industry standard practices gave me a great base to start my work from. Using these guidelines, I could go through the program and address the issues that stuck out to me, if I were just getting into programming it would be essential to learn these standards because they make a dramatic difference in the work you do and how you view other people's work. Industry

practices give us a baseline that allows us to determine how efficient a developer can be and how their work is received by the rest of the development community. If developers apply industry standards it will allow for a company to operate at the most efficient level possible, developers can review each other's code, add things to it and assist each other because everything is done to standard, and confusion can't be caused by a certain developer having a unique coding style.