GLOBALRAIN

Practices for Secure Software Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	04/13/2022	David DAgostino	04/15/23- Revised

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
David DAgostino

1. Algorithm Cipher

Recommend an appropriate encryption algorithm cipher to deploy, given the security vulnerabilities, and justify your reasoning. Review the scenario and the supporting materials to support your recommendation. In your practices for secure software report, be sure to address the following:

- Provide a brief, high-level overview of the encryption algorithm cipher.
- Discuss the hash functions and bit levels of the cipher.
- Explain the use of random numbers, symmetric versus non-symmetric keys, and so on.
- Describe the history and current state of encryption algorithms.

SHA-256 is a cryptographic algorithm used to convert data into a fixed-size string of characters called a hash. This algorithm generates a unique hash value for every input data, making it difficult to reverse engineer the original data. It uses a complex mathematical function that operates on the input data in blocks and generates a hash value that is 256 bits in length. SHA-256 is widely used for secure authentication, data integrity, and digital signatures in various applications such as blockchain, password storage, and SSL/TLS certificates.

The SHA-256 cipher is a hash function that takes input data of any size and outputs a fixed-size 256-bit message digest. In simpler terms, it converts a large amount of data into a smaller, unique, and fixed-size message digest that can be used for integrity checks and data authentication. The algorithm works by processing the input data through multiple rounds of operations that involve bitwise operations, logical operations, and modular arithmetic.

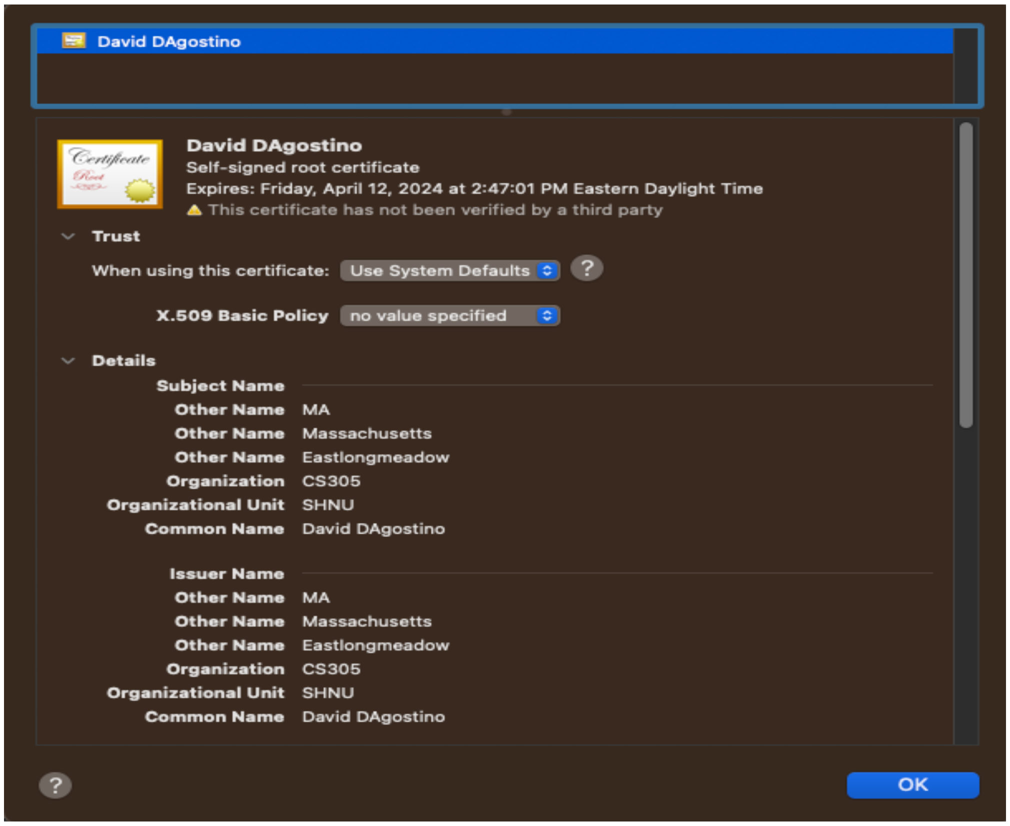
Random numbers play an essential role in cryptography as they are used to generate keys, which are used to encrypt and decrypt data. A key is a piece of information that is used in a cryptographic algorithm to transform plaintext into ciphertext and vice versa. There are two types of keys used in cryptography: symmetric and asymmetric.

- **Symmetric key encryption** involves the use of the same key for both encryption and decryption. This means that the sender and receiver must share the same key in advance, and if the key is compromised, all encrypted data is at risk.
- **Asymmetric key encryption**, also known as public key encryption, uses two different keys: a public key and a private key. The public key is used to encrypt data, while the private key is used to decrypt it. This means that the sender can encrypt the data using the recipient's public key, and only the recipient, who has access to the corresponding private key, can decrypt it.

Encryption algorithms have a long and evolving history. In ancient times, secret messages were hidden through various techniques such as invisible ink, while in modern times, encryption algorithms are used to protect sensitive information. The first modern encryption algorithm was the Caesar cipher, invented by Julius Caesar in the Roman Empire. Since then, more complex encryption algorithms have been developed, including symmetric key algorithms such as DES and AES, and asymmetric key algorithms like RSA and Elliptic Curve Cryptography (ECC). As technology advances, encryption algorithms continue to evolve, with the introduction of quantum-resistant cryptography and the development of new algorithms to address emerging threats. Today, encryption algorithms are widely used to secure communications, financial transactions, and other sensitive information. However, as technology advances, so do the methods and tools used to break encryption, which makes it a constant battle to keep information secure.

2. Certificate Generation

Insert a screenshot below of the CER file.



David DAgostino

David DAgostino
Self-signed root certificate
Expires: Friday, April 12, 2024 at 2:47:01 PM Eastern Daylight Time
⚠ This certificate has not been verified by a third party

Trust

When using this certificate: Use System Defaults

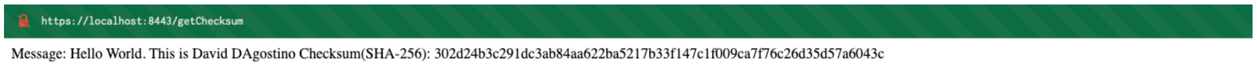
X.509 Basic Policy: no value specified

Details

Subject Name	
Other Name	MA
Other Name	Massachusetts
Other Name	Eastlongmeadow
Organization	CS305
Organizational Unit	SHNU
Common Name	David DAgostino
Issuer Name	
Other Name	MA
Other Name	Massachusetts
Other Name	Eastlongmeadow
Organization	CS305
Organizational Unit	SHNU
Common Name	David DAgostino

3. Deploy Cipher

Insert a screenshot below of the checksum verification.

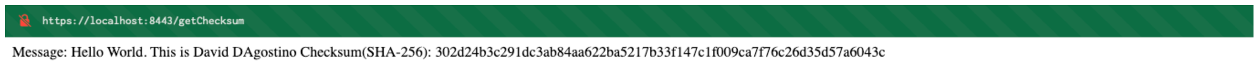


```
https://localhost:8443/getChecksum
```

Message: Hello World. This is David DAgostino Checksum(SHA-256): 302d24b3c291dc3ab84aa622ba5217b33f147c1f009ca7f76c26d35d57a6043c

4. Secure Communications

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

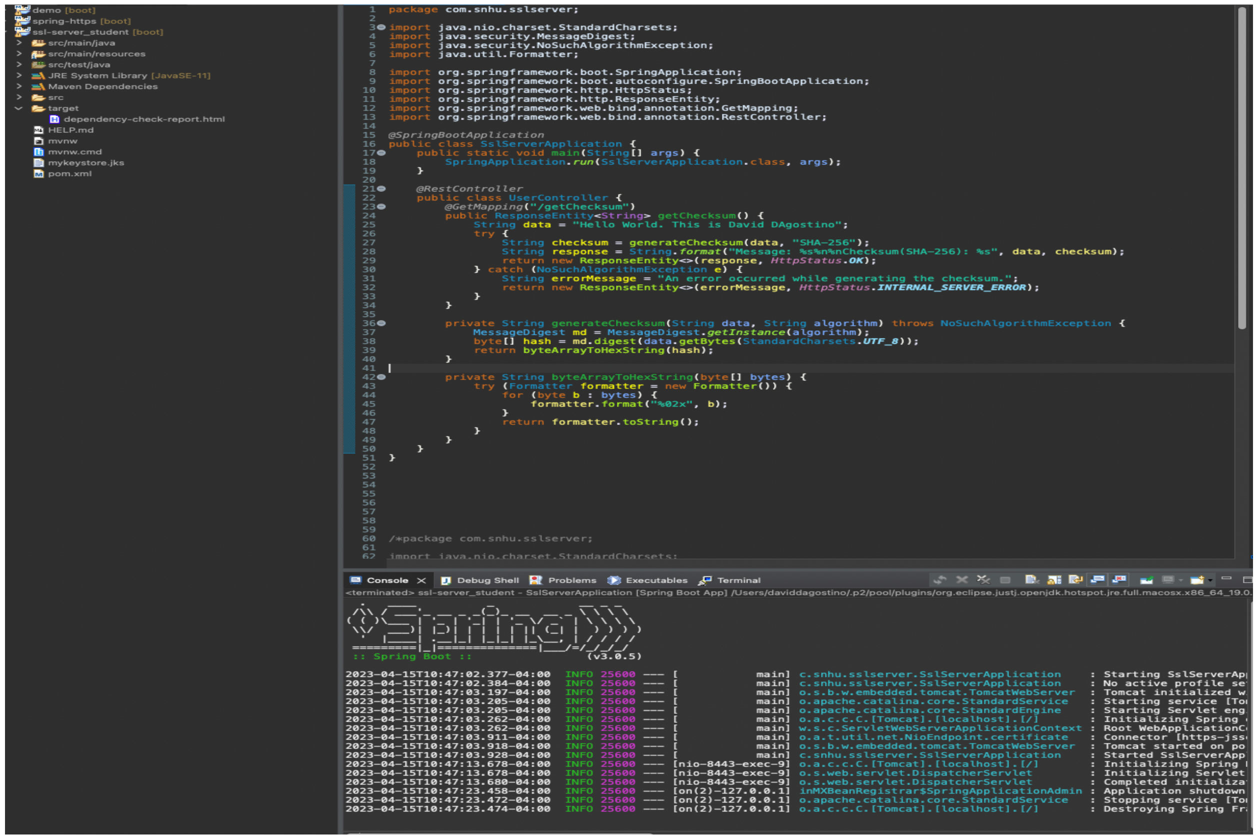


https://localhost:8443/getChecksum

Message: Hello World. This is David DAgostino Checksum(SHA-256): 302d24b3c291dc3ab84aa622ba5217b33f147c1f009ca7f76c26d35d57a6043c

5. Secondary Testing

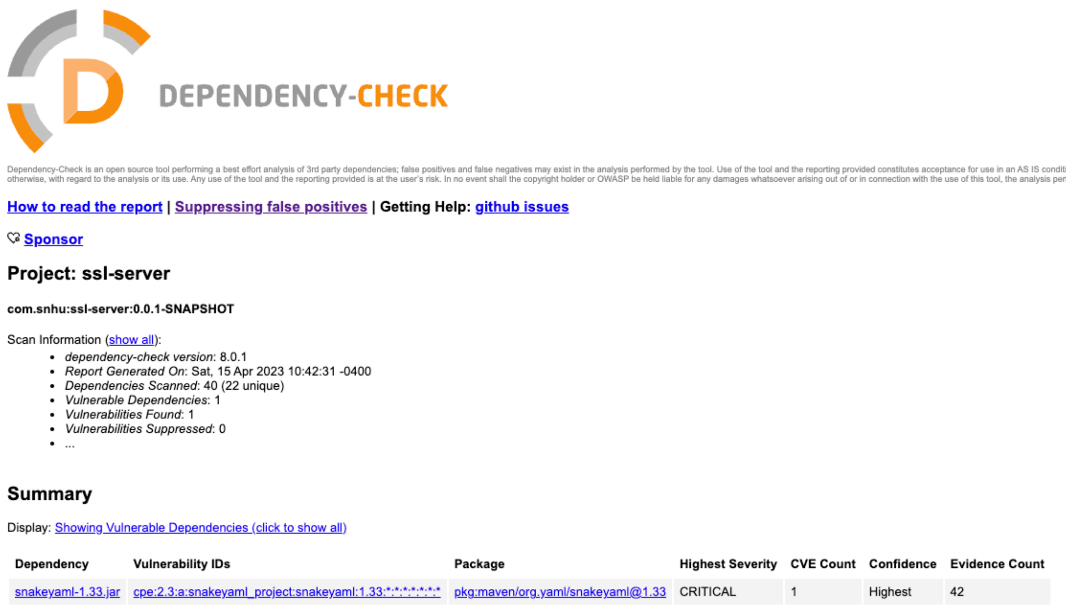
Insert screenshots below of the refactored code executed without errors and the dependency-check report.



```
demo [boot]
spring-https [boot]
ssl-server_student [boot]
  > src/main/java
  > src/main/resources
  > src/test/java
  > JRE System Library [JavaSE-11]
  > Maven Dependencies
  > src
  v target
      dependency-check-report.html
    HELP.md
    mvnw
    mvnw.cmd
    mykeystore.jks
    pom.xml
```

```java
package com.snhu.sslserver;

import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Formatter;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

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

    @RestController
    public class UserController {
        @GetMapping("/getChecksum")
        public ResponseEntity<String> getChecksum() {
            String data = "Hello World. This is David DAgostino";
            try {
                String checksum = generateChecksum(data, "SHA-256");
                String response = String.format("Message: %s%n%nChecksum(SHA-256): %s", data, checksum);
                return new ResponseEntity<>(response, HttpStatus.OK);
            } catch (NoSuchAlgorithmException e) {
                String errorMessage = "An error occurred while generating the checksum.";
                return new ResponseEntity<>(errorMessage, HttpStatus.INTERNAL_SERVER_ERROR);
            }
        }

        private String generateChecksum(String data, String algorithm) throws NoSuchAlgorithmException {
            MessageDigest md = MessageDigest.getInstance(algorithm);
            byte[] hash = md.digest(data.getBytes(StandardCharsets.UTF_8));
            return byteArrayToHexString(hash);
        }

        private String byteArrayToHexString(byte[] bytes) {
            try (Formatter formatter = new Formatter()) {
                for (byte b : bytes) {
                    formatter.format("%02x", b);
                }
                return formatter.toString();
            }
        }
    }
}








/*package com.snhu.sslserver;

import java.nio.charset.StandardCharsets;
```

```
<terminated> ssl-server_student - SslServerApplication [Spring Boot App] /Users/daviddagostino/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_19.0.

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.0.5)

2023-04-15T10:47:02.377-04:00  INFO 25600 --- [           main] c.snhu.sslserver.SslServerApplication    : Starting SslServerAp
2023-04-15T10:47:02.384-04:00  INFO 25600 --- [           main] c.snhu.sslserver.SslServerApplication    : No active profile se
2023-04-15T10:47:03.197-04:00  INFO 25600 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized w
2023-04-15T10:47:03.205-04:00  INFO 25600 --- [           main] o.apache.catalina.core.StandardService   : Starting service [To
2023-04-15T10:47:03.205-04:00  INFO 25600 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet eng
2023-04-15T10:47:03.262-04:00  INFO 25600 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring 
2023-04-15T10:47:03.262-04:00  INFO 25600 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationC
2023-04-15T10:47:03.911-04:00  INFO 25600 --- [           main] o.a.t.util.net.NioEndpoint.certificate   : Connector [https-jss
2023-04-15T10:47:03.918-04:00  INFO 25600 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on po
2023-04-15T10:47:03.928-04:00  INFO 25600 --- [           main] c.snhu.sslserver.SslServerApplication    : Started SslServerApp
2023-04-15T10:47:13.678-04:00  INFO 25600 --- [nio-8443-exec-9] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring 
2023-04-15T10:47:13.678-04:00  INFO 25600 --- [nio-8443-exec-9] o.s.web.servlet.DispatcherServlet        : Initializing Servlet
2023-04-15T10:47:13.680-04:00  INFO 25600 --- [nio-8443-exec-9] o.s.web.servlet.DispatcherServlet        : Completed initializa
2023-04-15T10:47:23.458-04:00  INFO 25600 --- [on(2)-127.0.0.1] inMXBeanRegistrar$SpringApplicationAdmin : Application shutdown
2023-04-15T10:47:23.472-04:00  INFO 25600 --- [on(2)-127.0.0.1] o.apache.catalina.core.StandardService   : Stopping service [To
2023-04-15T10:47:23.474-04:00  INFO 25600 --- [on(2)-127.0.0.1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Destroying Spring Fr
```



DEPENDENCY-CHECK

Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes acceptance for use in an AS IS condition, and there are NO warranties, implied or otherwise, with regard to the analysis or its use. Any use of the tool and the reporting provided is at the user's risk. In no event shall the copyright holder or OWASP be held liable for any damages whatsoever arising out of or in connection with the use of this tool, the analysis performed, or the resulting report.

How to read the report | Suppressing false positives | Getting Help: github issues

Sponsor

Project: ssl-server

com.snhu:ssl-server:0.0.1-SNAPSHOT

Scan Information (show all):

- *dependency-check version*: 8.0.1
- *Report Generated On*: Sat, 15 Apr 2023 10:42:31 -0400
- *Dependencies Scanned*: 40 (22 unique)
- *Vulnerable Dependencies*: 1
- *Vulnerabilities Found*: 1
- *Vulnerabilities Suppressed*: 0
- ...

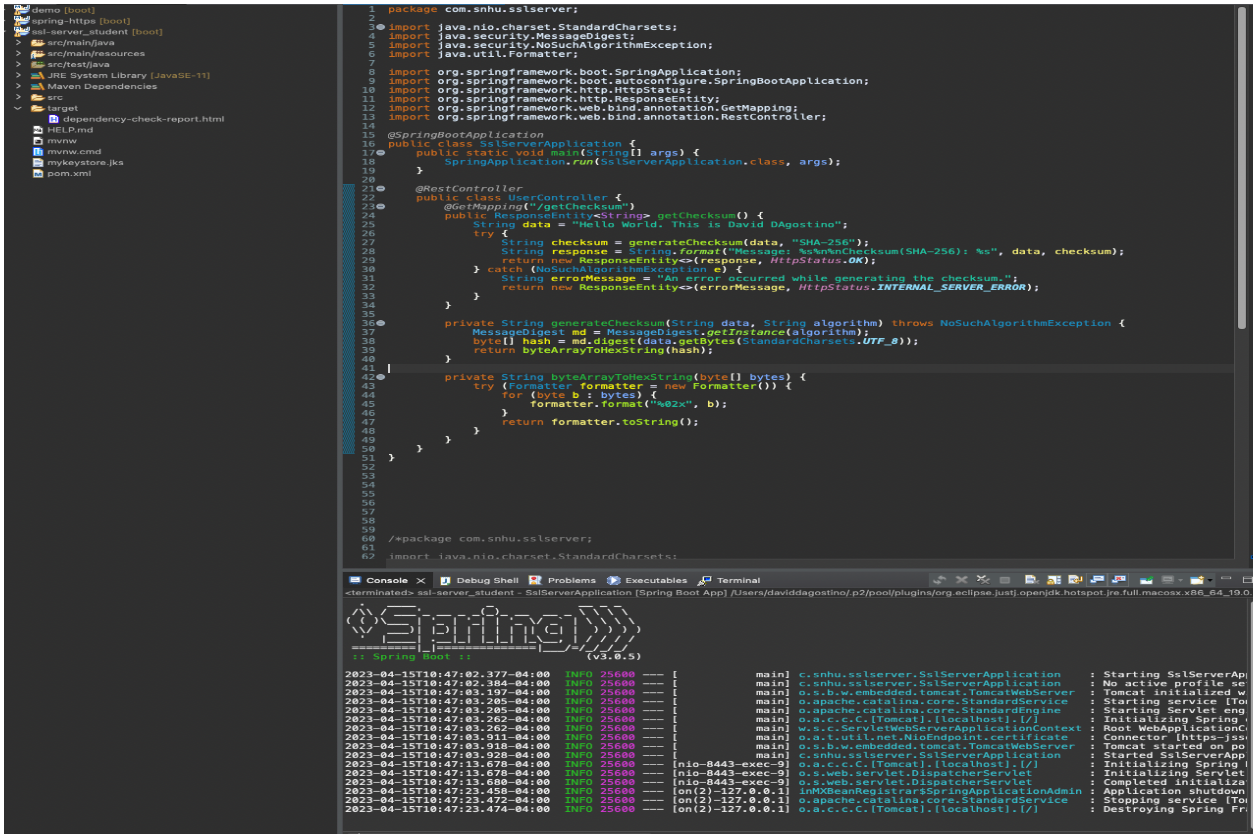
Summary

Display: Showing Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
snakeyaml-1.33.jar	cpe:2.3:a:snakeyaml_project:snakeyaml:1.33:*:*:*:*:*:*:*	pkg:maven/org.yaml/snakeyaml@1.33	CRITICAL	1	Highest	42

6. Functional Testing

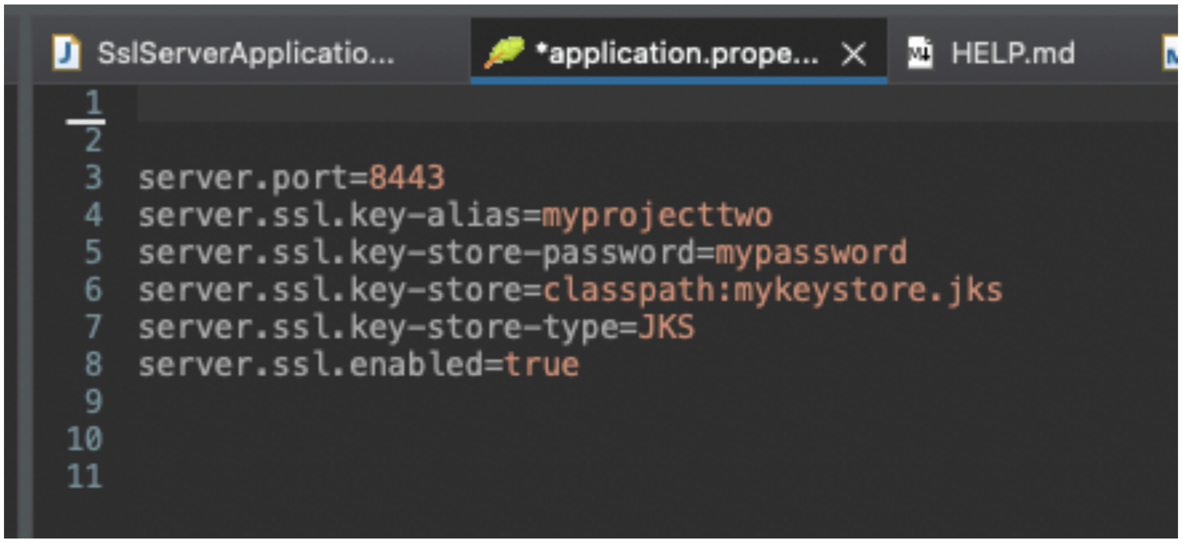
Insert a screenshot below of the refactored code executed without errors.



```
demo [boot]
spring-https [boot]
ssl-server_student [boot]
  > src/main/java
  > src/main/resources
  > src/test/java
  > JRE System Library [JavaSE-11]
  > Maven Dependencies
  > src
  ˅ target
      dependency-check-report.html
    HELP.md
    mvnw
    mvnw.cmd
    mykeystore.jks
    pom.xml
```

```java
package com.snhu.sslserver;

import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Formatter;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

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

    @RestController
    public class UserController {
        @GetMapping("/getChecksum")
        public ResponseEntity<String> getChecksum() {
            String data = "Hello World. This is David DAgostino";
            try {
                String checksum = generateChecksum(data, "SHA-256");
                String response = String.format("Message: %s%n%nChecksum(SHA-256): %s", data, checksum);
                return new ResponseEntity<>(response, HttpStatus.OK);
            } catch (NoSuchAlgorithmException e) {
                String errorMessage = "An error occurred while generating the checksum.";
                return new ResponseEntity<>(errorMessage, HttpStatus.INTERNAL_SERVER_ERROR);
            }
        }

        private String generateChecksum(String data, String algorithm) throws NoSuchAlgorithmException {
            MessageDigest md = MessageDigest.getInstance(algorithm);
            byte[] hash = md.digest(data.getBytes(StandardCharsets.UTF_8));
            return byteArrayToHexString(hash);
        }

        private String byteArrayToHexString(byte[] bytes) {
            try (Formatter formatter = new Formatter()) {
                for (byte b : bytes) {
                    formatter.format("%02x", b);
                }
                return formatter.toString();
            }
        }
    }
}









/*package com.snhu.sslserver;

import java.nio.charset.StandardCharsets;
```

```
<terminated> ssl-server_student - SslServerApplication [Spring Boot App] /Users/daviddagostino/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_19.0.

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.0.5)

2023-04-15T10:47:02.377-04:00  INFO 25600 --- [           main] c.snhu.sslserver.SslServerApplication    : Starting SslServerAp
2023-04-15T10:47:02.384-04:00  INFO 25600 --- [           main] c.snhu.sslserver.SslServerApplication    : No active profile se
2023-04-15T10:47:03.197-04:00  INFO 25600 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized w
2023-04-15T10:47:03.205-04:00  INFO 25600 --- [           main] o.apache.catalina.core.StandardService   : Starting service [To
2023-04-15T10:47:03.205-04:00  INFO 25600 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet eng
2023-04-15T10:47:03.262-04:00  INFO 25600 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring
2023-04-15T10:47:03.262-04:00  INFO 25600 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationC
2023-04-15T10:47:03.911-04:00  INFO 25600 --- [           main] o.a.t.util.net.NioEndpoint.certificate   : Connector [https-jss
2023-04-15T10:47:03.918-04:00  INFO 25600 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on po
2023-04-15T10:47:03.928-04:00  INFO 25600 --- [           main] c.snhu.sslserver.SslServerApplication    : Started SslServerApp
2023-04-15T10:47:13.678-04:00  INFO 25600 --- [nio-8443-exec-9] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring
2023-04-15T10:47:13.678-04:00  INFO 25600 --- [nio-8443-exec-9] o.s.web.servlet.DispatcherServlet        : Initializing Servlet
2023-04-15T10:47:13.680-04:00  INFO 25600 --- [nio-8443-exec-9] o.s.web.servlet.DispatcherServlet        : Completed initializa
2023-04-15T10:47:23.458-04:00  INFO 25600 --- [on(2)-127.0.0.1] inMXBeanRegistrar$SpringApplicationAdmin : Application shutdown
2023-04-15T10:47:23.472-04:00  INFO 25600 --- [on(2)-127.0.0.1] o.apache.catalina.core.StandardService   : Stopping service [To
2023-04-15T10:47:23.474-04:00  INFO 25600 --- [on(2)-127.0.0.1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Destroying Spring Fr
```


SslServerApplicatio... *application.prope... HELP.md

```
server.port=8443
server.ssl.key-alias=myprojecttwo
server.ssl.key-store-password=mypassword
server.ssl.key-store=classpath:mykeystore.jks
server.ssl.key-store-type=JKS
server.ssl.enabled=true
```

7. Summary

Discuss how the code has been refactored and how it complies with security testing protocols. In the summary of your practices for secure software report, be sure to address the following:

- Refer to the Vulnerability Assessment Process Flow Diagram. Highlight the areas of security that you addressed by refactoring the code.
- Discuss your process for adding layers of security to the software application.

1. APIs
2. Cryptography
3. Client/Server
4. Code Error
5. Code Quality

The application's security was improved through several measures, including implementing HTTPS, utilizing encryption algorithm ciphers and hash functions for cryptography, verifying checksum, addressing client/server communication, and improving code quality through exception handling and code review. Additionally, self-signed certificates were added to ensure HTTPS usage, and the pom.xml file was refactored to resolve all identified vulnerabilities through dependency checks.

To ensure the certificates were properly created and HTTPS could be utilized, the first step was taken. This helps to secure the webpage and assures users that they are interacting with the intended organization, preventing imposter attacks. The next step involved verifying that the hashing function worked correctly and utilizing checksum to ensure data confidentiality. This ensures that users' data is protected and not easily retrievable. Finally, all vulnerabilities were addressed, ensuring that the application's inner workings are updated and functioning as intended, providing an additional layer of protection.

To maintain the application's security, best practices include patching software and systems regularly to keep everything up to date. This helps to prevent attacks by making it harder for attackers to exploit outdated systems. Enforcing least privilege is also crucial to prevent attacks within the group, ensuring users only have access to the resources they need. While it is not currently implemented, this measure will help to protect the organization in the future.

8. Industry Standard Best Practices

Explain how you applied industry standard best practices for secure coding to mitigate against known security vulnerabilities. Be sure to address the following:

- Explain how you used industry standard best practices to maintain the software application's current security.
- Explain the value of applying industry standard best practices for secure coding to the company's overall wellbeing.

Industry standard best practices for secure coding have been applied to maintain the software application's current security. Firstly, the use of HTTPS has been implemented to ensure secure communication between the client and server. Additionally, the SHA-256 encryption algorithm has been used to generate a checksum for the data, ensuring that the data is not easily retrievable. The code has been reviewed to ensure functionality and readability, and exceptions have been utilized to handle code errors.

Applying industry standard best practices for secure coding is crucial to the overall wellbeing of the company. It helps to mitigate the risk of cyber attacks, data breaches, and loss of sensitive information, which can have severe financial and reputational consequences. Implementing secure coding practices also increases customer trust and confidence, ultimately leading to increased business and revenue. Furthermore, adhering to industry standard best practices can ensure that the company meets regulatory compliance requirements and avoids legal and financial penalties.

References:

Jena, B.K. (n.d.). SHA-256 Algorithm - What is SHA-256 Algorithm? Simplilearn. Retrieved April 17, 2023, from https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm

N-able. (2021, January 7). SHA-256 Encryption: A Guide to What It Is and How It Works. Retrieved from https://www.n-able.com/blog/sha-256-encryption