GLOBALRAIN

CS 305 Project Two
Practices for Secure Software Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	12/11/2021	Matt Zindler	

Client

ARTEMIS FINANCIAL

Instructions

Deliver this completed Practices for Secure Software Report documenting your process for writing secure communications and refactoring code that complies with software security testing protocols. Respond to the steps outlined below and replace the bracketed text with your findings in your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

Developer
Matt Zindler

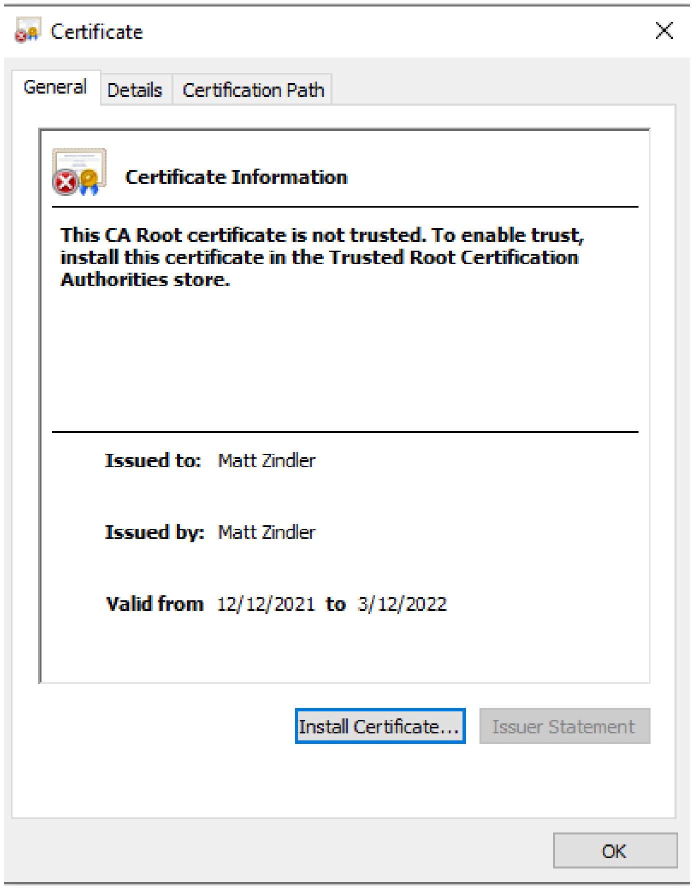
1. Algorithm Cipher
Determine an appropriate encryption algorithm cipher to deploy given the security vulnerabilities, justifying your reasoning. Be sure to address the following:

- 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 vs non-symmetric keys, and so on.
- Describe the history and current state of encryption algorithms.

- I recommend the use of AES for this project, as it has a flexible bit level, and a large key size to make brute force attacks near improbable. AES is commonly used worldwide, and for governmental purposes. AES is also currently uncracked, meaning malicious users have to resort to other methods to attempt to break the encryption.
- The hash function takes the message that is to be encrypted, and assigns each letter with a random number associated with its position on the hash table. The message is then scrambled several times to produce the resulting encrypted string. Bit levels relate by changing the message size that it scrambles, and the resulting encrypted string. The higher bit level you have, the larger the message size is, and in turn, the longer it takes to encode and decode the message.
- Random numbers and symmetric/non-symmetric keys lead to algorithms that are difficult to crack, whether it's by hand or by brute force. Random numbers come into play with a hash table that encryption algorithms use to transform messages. The message is put into the table which has random numbers associated with each position. The message is then scrambled multiple times depending on the cipher until the message is complete. Keys help in this way, by laying out the arrangement of numbers that scramble the message. Having an asymmetric key means that the sender and receiver of the message each have their own key to encrypt and decrypt the message sent. On the other hand a symmetric key means only one key is used to encrypt and decrypt messages.
- Before the advent of modern computers encrypting generally worked on a smaller scale, with less combination size, and often for military use.. During WW2, the German military used the Enigma machine to encode military messages to stop the Allies from being able to read them. The Allies were able to eventually crack the code, and encryption took a big step forward. In more recent years, the use of computers has increased the rate at which encryption has been adopted. With the increasing speed at which computers can attempt to brute force or attack encryptions in different ways.. Standards have changed as algorithms get longer and more complex, from DES to AES to ECIES.

2. Certificate Generation
Generate appropriate self-signed certificates using the Java Keytool, which is used through the command line.

- To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.


Certificate

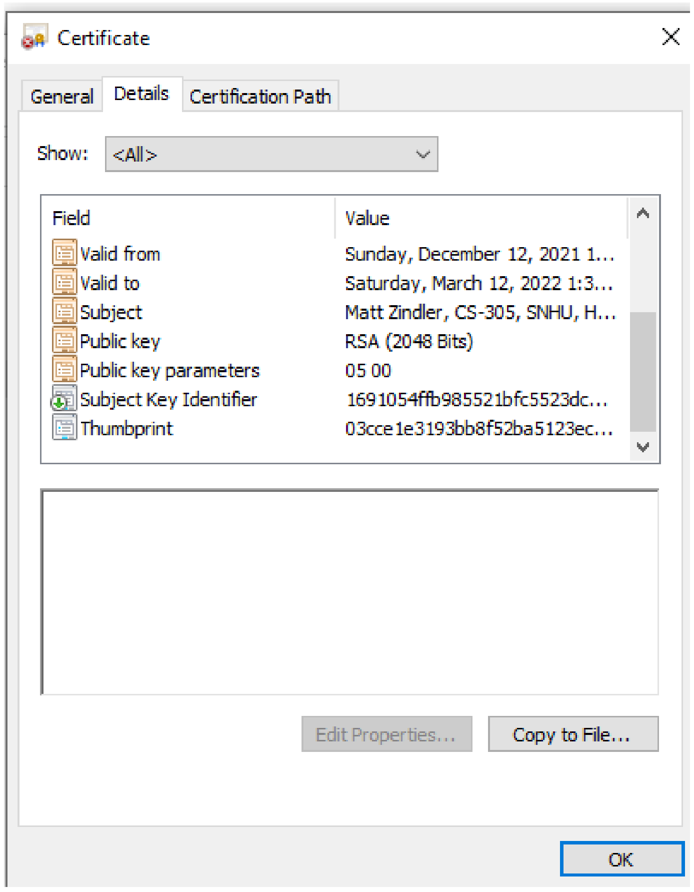
Certificate Information

This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

Issued to: Matt Zindler

Issued by: Matt Zindler

Valid from 12/12/2021 **to** 3/12/2022

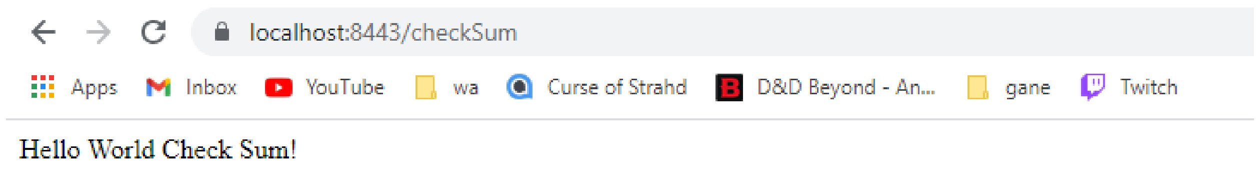

Certificate

Show: <All>

Field	Value
Valid from	Sunday, December 12, 2021 1…
Valid to	Saturday, March 12, 2022 1:3…
Subject	Matt Zindler, CS-305, SNHU, H…
Public key	RSA (2048 Bits)
Public key parameters	05 00
Subject Key Identifier	1691054ffb985521bfc5523dc…
Thumbprint	03cce1e3193bb8f52ba5123ec…

3. Deploy Cipher
Refactor the code and use security libraries to deploy and implement the encryption algorithm cipher to the software application. Verify this additional functionality with a checksum.

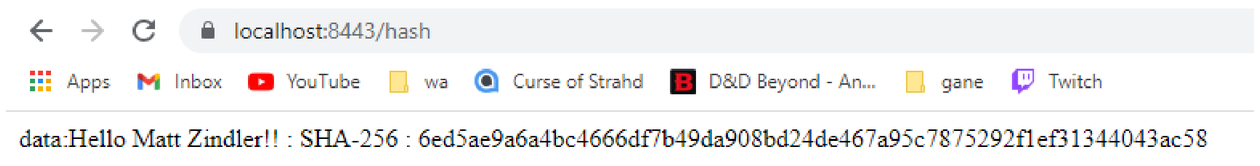
- Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.



Hello World Check Sum!

4. Secure Communications
Refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code to verify secure communication by typing **https://localhost:8443/hash** in a new browser window to demonstrate that the secure communication works successfully.

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

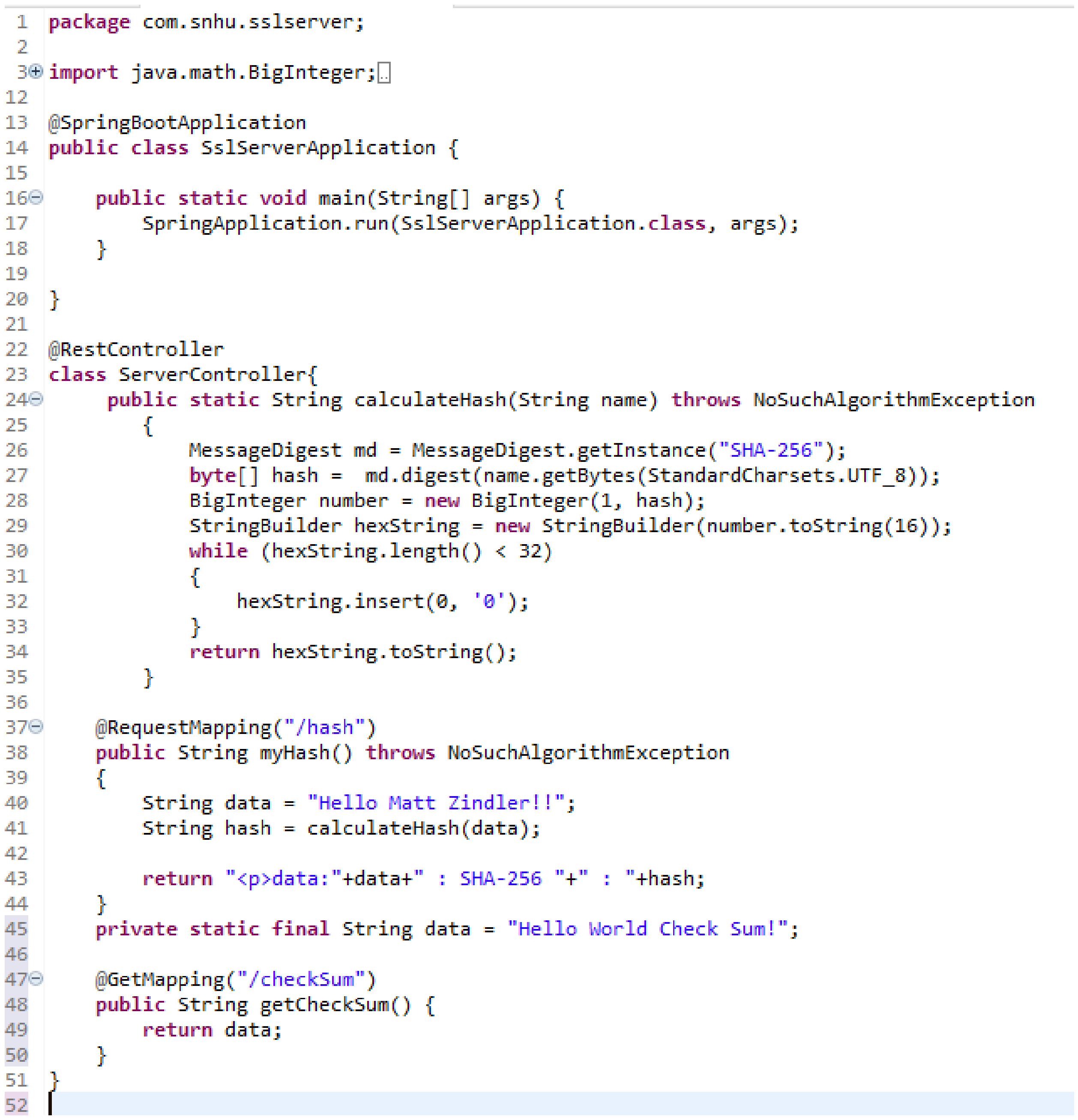


```
data:Hello Matt Zindler!! : SHA-256 : 6ed5ae9a6a4bc4666df7b49da908bd24de467a95c7875292f1ef31344043ac58
```

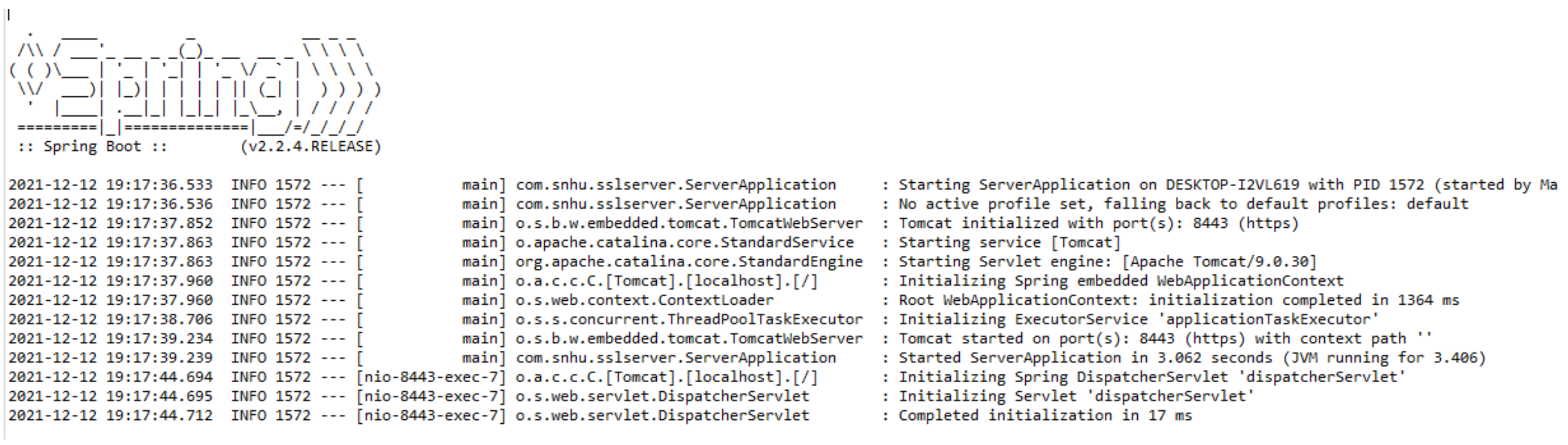
5. Secondary Testing
Complete a secondary static testing of the refactored code using the dependency check tool to ensure code complies with software security enhancements. You only need to focus on the code you have added as part of the refactoring. Complete the dependency check and review the output to ensure you did not introduce additional security vulnerabilities.

- Include the following below:
  - A screenshot of the refactored code executed without errors
  - A screenshot of the dependency check report

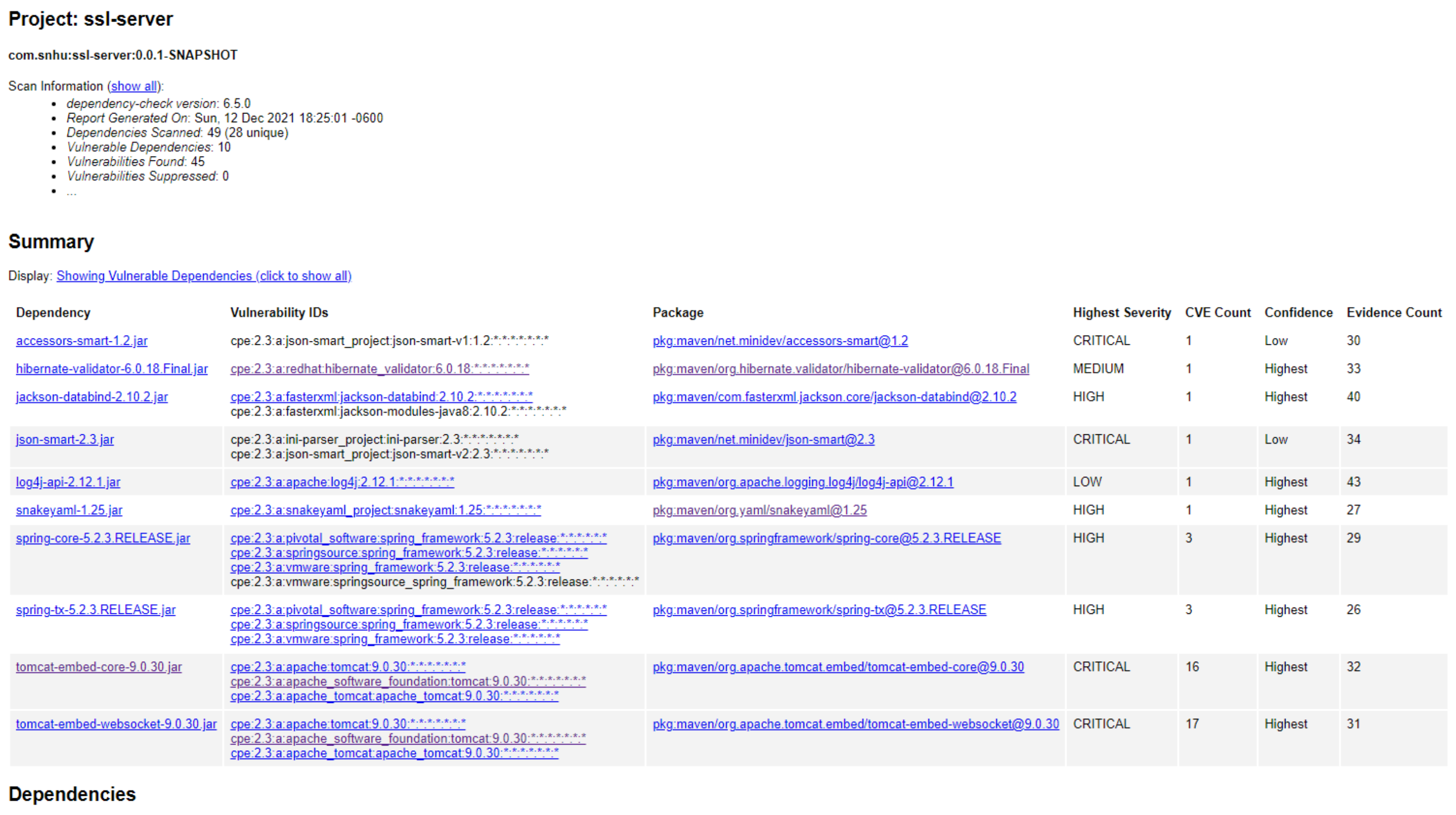
- Many of the vulnerabilities are linked to the version of Tomcat used, and some are not fixed by the most recent updates.



```java
package com.snhu.sslserver;

import java.math.BigInteger;

@SpringBootApplication
public class SslServerApplication {

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

}

@RestController
class ServerController{
     public static String calculateHash(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 myHash() throws NoSuchAlgorithmException
    {
        String data = "Hello Matt Zindler!!";
        String hash = calculateHash(data);

        return "<p>data:"+data+" : SHA-256 "+" : "+hash;
    }
    private static final String data = "Hello World Check Sum!";

    @GetMapping("/checkSum")
    public String getCheckSum() {
        return data;
    }
}
```



```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.4.RELEASE)

2021-12-12 19:17:36.533  INFO 1572 --- [           main] com.snhu.sslserver.ServerApplication     : Starting ServerApplication on DESKTOP-I2VL619 with PID 1572 (started by Ma
2021-12-12 19:17:36.536  INFO 1572 --- [           main] com.snhu.sslserver.ServerApplication     : No active profile set, falling back to default profiles: default
2021-12-12 19:17:37.852  INFO 1572 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8443 (https)
2021-12-12 19:17:37.863  INFO 1572 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2021-12-12 19:17:37.863  INFO 1572 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.30]
2021-12-12 19:17:37.960  INFO 1572 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2021-12-12 19:17:37.960  INFO 1572 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 1364 ms
2021-12-12 19:17:38.706  INFO 1572 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2021-12-12 19:17:39.234  INFO 1572 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8443 (https) with context path ''
2021-12-12 19:17:39.239  INFO 1572 --- [           main] com.snhu.sslserver.ServerApplication     : Started ServerApplication in 3.062 seconds (JVM running for 3.406)
2021-12-12 19:17:44.694  INFO 1572 --- [nio-8443-exec-7] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2021-12-12 19:17:44.695  INFO 1572 --- [nio-8443-exec-7] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2021-12-12 19:17:44.712  INFO 1572 --- [nio-8443-exec-7] o.s.web.servlet.DispatcherServlet        : Completed initialization in 17 ms
```



Project: ssl-server

com.snhu:ssl-server:0.0.1-SNAPSHOT

Scan Information (show all):

- *dependency-check version*: 6.5.0
- *Report Generated On*: Sun, 12 Dec 2021 18:25:01 -0600
- *Dependencies Scanned*: 49 (28 unique)
- *Vulnerable Dependencies*: 10
- *Vulnerabilities Found*: 45
- *Vulnerabilities Suppressed*: 0
- ...

Summary

Display: Showing Vulnerable Dependencies (click to show all)

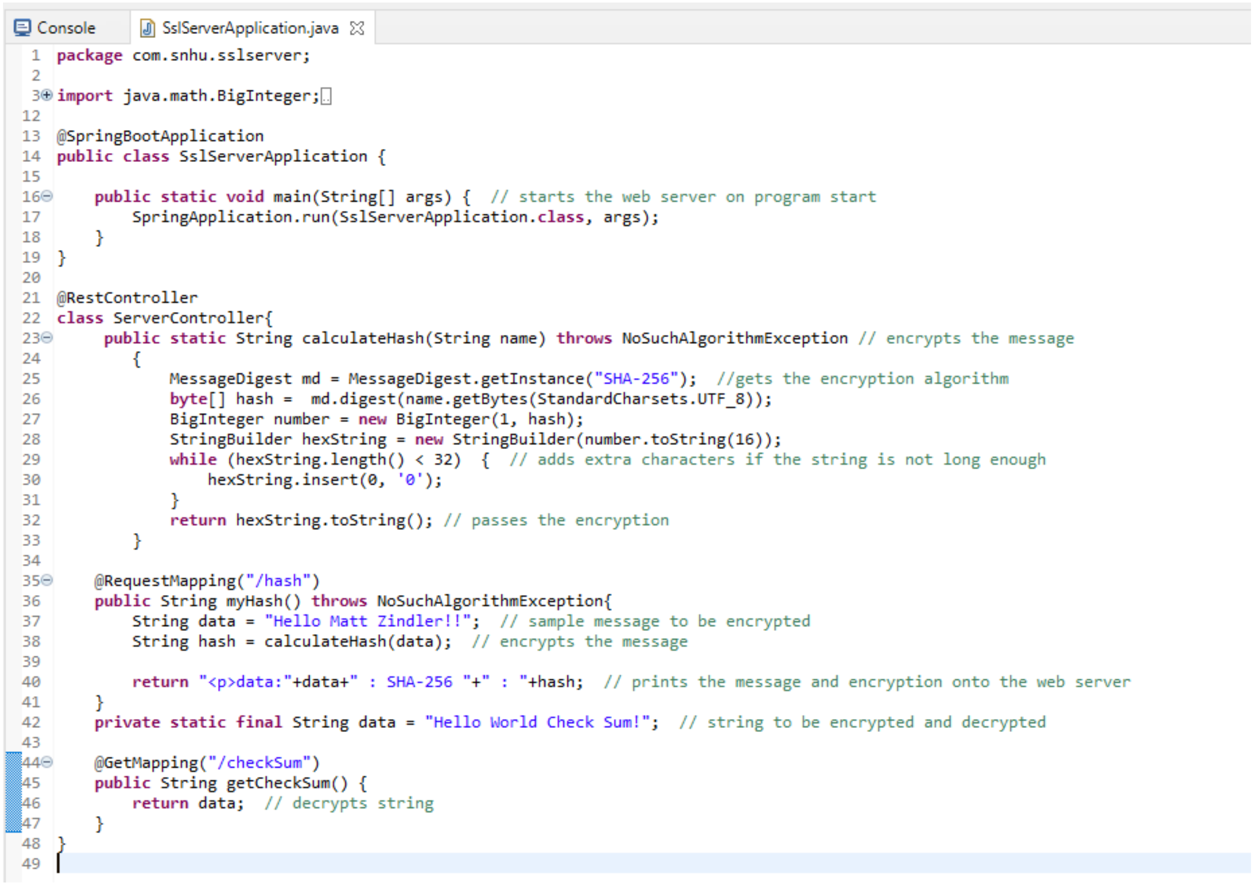
Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
accessors-smart-1.2.jar	cpe:2.3:a:json-smart_project:json-smart-v1:1.2:*:*:*:*:*:*:*	pkg:maven/net.minidev/accessors-smart@1.2	CRITICAL	1	Low	30
hibernate-validator-6.0.18.Final.jar	cpe:2.3:a:redhat:hibernate_validator:6.0.18:*:*:*:*:*:*:*	pkg:maven/org.hibernate.validator/hibernate-validator@6.0.18.Final	MEDIUM	1	Highest	33
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	1	Highest	40
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	1	Low	34
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	43
snakeyaml-1.25.jar	cpe:2.3:a:snakeyaml_project:snakeyaml:1.25:*:*:*:*:*:*:*	pkg:maven/org.yaml/snakeyaml@1.25	HIGH	1	Highest	27
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:*:*:*:*:*:* cpe:2.3:a:vmware:springsource_spring_framework:5.2.3:release:*:*:*:*:*:*	pkg:maven/org.springframework/spring-core@5.2.3.RELEASE	HIGH	3	Highest	29
spring-tx-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-tx@5.2.3.RELEASE	HIGH	3	Highest	26
tomcat-embed-core-9.0.30.jar	cpe:2.3:a:apache:tomcat:9.0.30:*:*:*:*:*:*:* cpe:2.3:a:apache_software_foundation: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	16	Highest	32
tomcat-embed-websocket-9.0.30.jar	cpe:2.3:a:apache:tomcat:9.0.30:*:*:*:*:*:*:* cpe:2.3:a:apache_software_foundation: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	17	Highest	31

Dependencies

6. Functional Testing

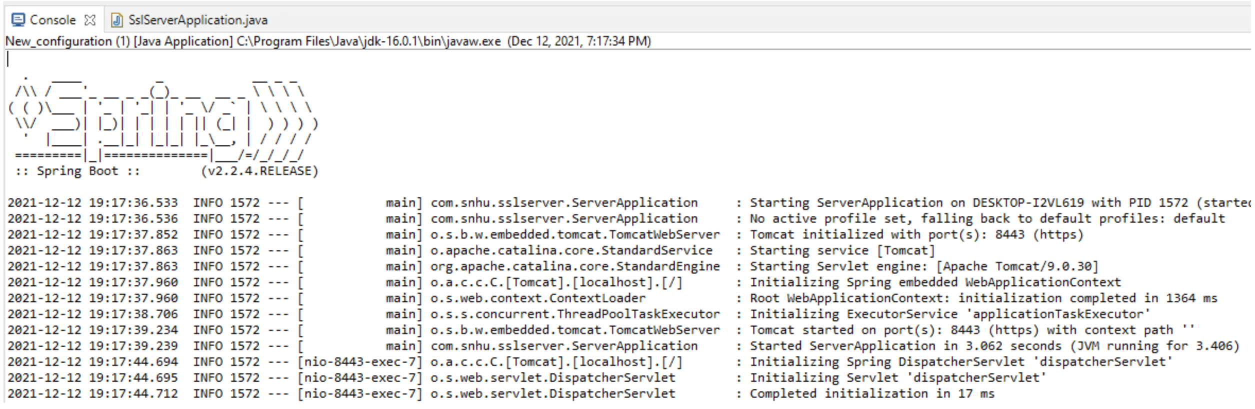
Identify syntactical, logical, and security vulnerabilities for the software application by manually reviewing code.

- Complete this functional testing and include a screenshot below of the refactored code executed without errors.



Console | SslServerApplication.java

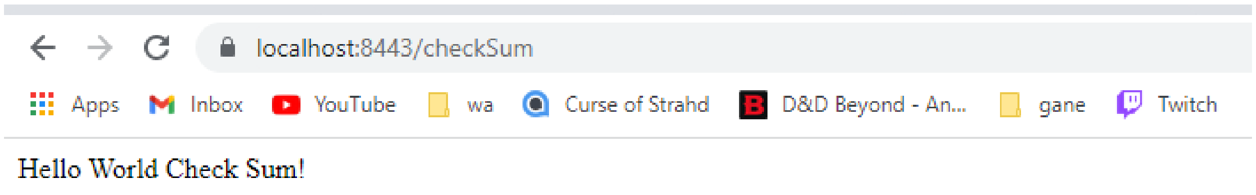
```java
package com.snhu.sslserver;

import java.math.BigInteger;

@SpringBootApplication
public class SslServerApplication {

    public static void main(String[] args) {  // starts the web server on program start
        SpringApplication.run(SslServerApplication.class, args);
    }
}

@RestController
class ServerController{
     public static String calculateHash(String name) throws NoSuchAlgorithmException // encrypts the message
        {
            MessageDigest md = MessageDigest.getInstance("SHA-256");  //gets the encryption algorithm
            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)  {  // adds extra characters if the string is not long enough
                hexString.insert(0, '0');
            }
            return hexString.toString(); // passes the encryption
        }

    @RequestMapping("/hash")
    public String myHash() throws NoSuchAlgorithmException{
        String data = "Hello Matt Zindler!!";  // sample message to be encrypted
        String hash = calculateHash(data);  // encrypts the message

        return "<p>data:"+data+"  : SHA-256 "+" : "+hash;  // prints the message and encryption onto the web server
    }
    private static final String data = "Hello World Check Sum!";  // string to be encrypted and decrypted

    @GetMapping("/checkSum")
    public String getCheckSum() {
        return data;  // decrypts string
    }
}
```



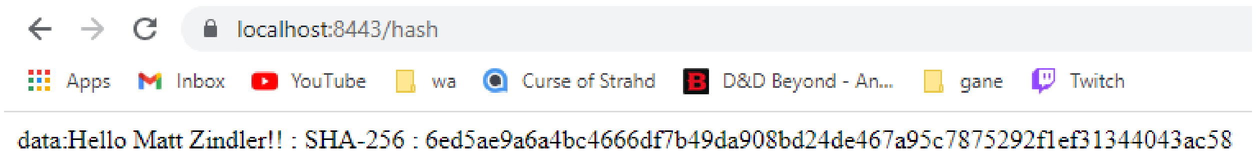
Console | SslServerApplication.java

New_configuration (1) [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (Dec 12, 2021, 7:17:34 PM)

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.4.RELEASE)

2021-12-12 19:17:36.533  INFO 1572 --- [           main] com.snhu.sslserver.ServerApplication     : Starting ServerApplication on DESKTOP-I2VL619 with PID 1572 (startec
2021-12-12 19:17:36.536  INFO 1572 --- [           main] com.snhu.sslserver.ServerApplication     : No active profile set, falling back to default profiles: default
2021-12-12 19:17:37.852  INFO 1572 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8443 (https)
2021-12-12 19:17:37.863  INFO 1572 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2021-12-12 19:17:37.863  INFO 1572 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.30]
2021-12-12 19:17:37.960  INFO 1572 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2021-12-12 19:17:37.960  INFO 1572 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 1364 ms
2021-12-12 19:17:38.706  INFO 1572 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2021-12-12 19:17:39.234  INFO 1572 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8443 (https) with context path ''
2021-12-12 19:17:39.239  INFO 1572 --- [           main] com.snhu.sslserver.ServerApplication     : Started ServerApplication in 3.062 seconds (JVM running for 3.406)
2021-12-12 19:17:44.694  INFO 1572 --- [nio-8443-exec-7] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2021-12-12 19:17:44.695  INFO 1572 --- [nio-8443-exec-7] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2021-12-12 19:17:44.712  INFO 1572 --- [nio-8443-exec-7] o.s.web.servlet.DispatcherServlet        : Completed initialization in 17 ms
```



Hello World Check Sum!



```
data:Hello Matt Zindler!! : SHA-256 : 6ed5ae9a6a4bc4666df7b49da908bd24de467a95c7875292f1ef31344043ac58
```

7. Summary

Discuss how the code has been refactored and how it complies with security testing protocols. Be sure to address the following:

- Refer to the Vulnerability Assessment Process Flow Diagram and highlight the areas of security that you addressed by refactoring the code.
- Discuss your process for adding layers of security to the software application and the value that security adds to the company's overall wellbeing.
- Point out best practices for maintaining the current security of the software application to your customer.

- The main vulnerabilities that were addressed in this code are: cryptography, client/server, and code quality. Cryptology takes place with the encryption method that converts the given string into random characters using a hash. This helps keep data secure from unauthorized users by hiding the data and requiring a key to decrypt it. Client/server security was taken care of by having a secure HTTPS connection to the web server, allowing users to know that their data has not been tampered with. Code quality is the last major vulnerability addressed, the code in this project leaves for minimal errors that malicious users can take advantage of.
- Adding multiple layers is a great line of defense for any program. Since there are many different kinds of attacks each security layer could have faults in it that others can catch before they do any damage. This particular program has security features such as certificates on the web browser, password protected certificates, and encrypted data. These security features can massively decrease the chance of a successful attack that could result in program crashes, and exposed data.
- Keeping security up to date and well maintained is the best defense possible. Ensure that any other software or application that this program uses is kept up to date, such as Tomcat, Maven, and other 1st or 3rd party additions. Keeping user data secure is also essential, ensuring that users know to keep their accounts secure, have strong unique passwords, and to be aware of user attacks like phishing.

Sources:

A brief history of encryption. Thales Group. (2016, April 18). Retrieved December 12, 2021, from https://www.thalesgroup.com/en/markets/digital-identity-and-security/magazine/brief-history-encryption.