GLOBALRAIN

CS 305 Project Two
Practices for Secure Software Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	06/19/2022	Craig O'Loughlin	Initial

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
Craig O'Loughlin

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.

Artemis Financial handles the sensitive personal information of clients on a web-based platform. A general encryption scheme is recommended to protect this information from unauthorized viewing. For this purpose, AES-128 or higher is recommended to be implemented at the database level if possible.

AES is an industry standard symmetric encryption algorithm that allows files to be locked or unlocked with the use of a 'key', which must be kept safe from unauthorized access. For this reason, keys should be generated and stored using a secure automated key generation tool. AES can currently be implemented using 128, 198, or 256-bits, each being more secure but less performant. This algorithm should be implemented using a cryptographically secure random number seed and a trusted library, it is not usually recommended to write your own encryption scheme as it is likely to be less secure.

Artemis Financial also seeks to verify that their files transferred over internet channels are unchanged in order to prevent file tampering and other related security threats. For this purpose, we recommend using a checksum feature implemented with an SHA-256 algorithm.

SHA-256 is currently considered the 'standard' hashing algorithm. It contains no known collisions, that is, no two known different files that will produce the same hash, thereby 'tricking' the verification system into thinking these two files are the same when in fact one may be malicious. SHA-256 produces a robust 256-bit hash of a given file or data object which contributes to the security of this algorithm. SHA-256 is a one-way hash used mainly for file verification, therefore the original file cannot be reproduced using the hash.

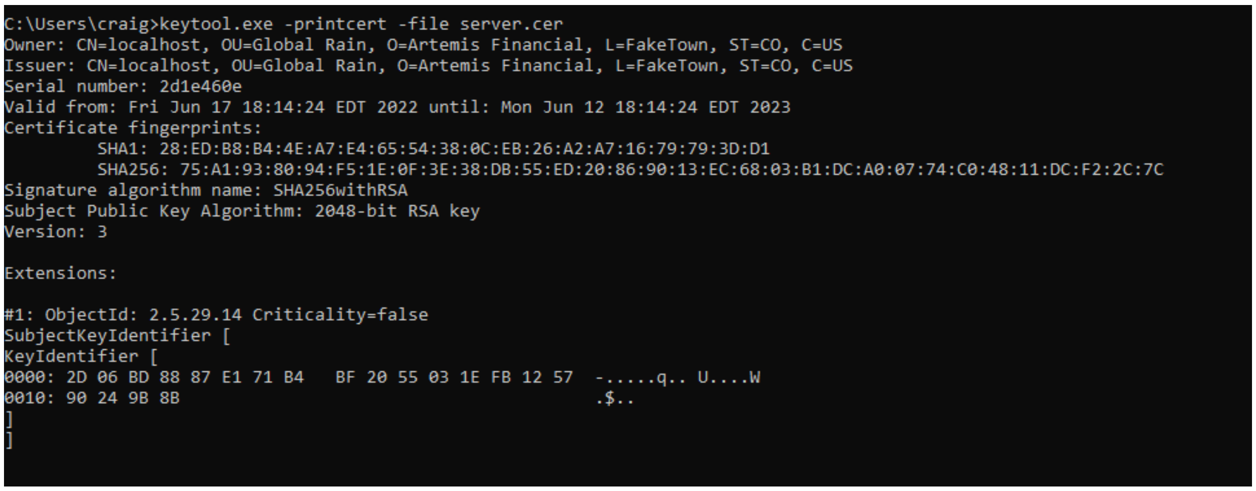
Finally, any remotely sensitive data transferred over the internet should be done using the widely adopted HTTPS protocol. This protocol uses SSL which implements the RSA algorithm, a non-symmetric cipher with both a public and private key to lock and unlock the data. This algorithm scrambles any data in transfer using a public key, with the only entity able to unscramble and view this data being the owner of the associated private key (the secure financial server).

The history of encryption technology has been a battle between cryptographic complexity and computer hardware performance. As computer technology gets better, the ability for a hacker to 'solve' more and more complex algorithms grows. For this reason, continuous assessment of an applications encryption suite is needed. For example, algorithms that have been considered safe in the recent past, such as MD5 and SHA-1, are currently considered trivial to break, and it is thought that current algorithms are at risk in the near future due to advances in quantum computing.

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

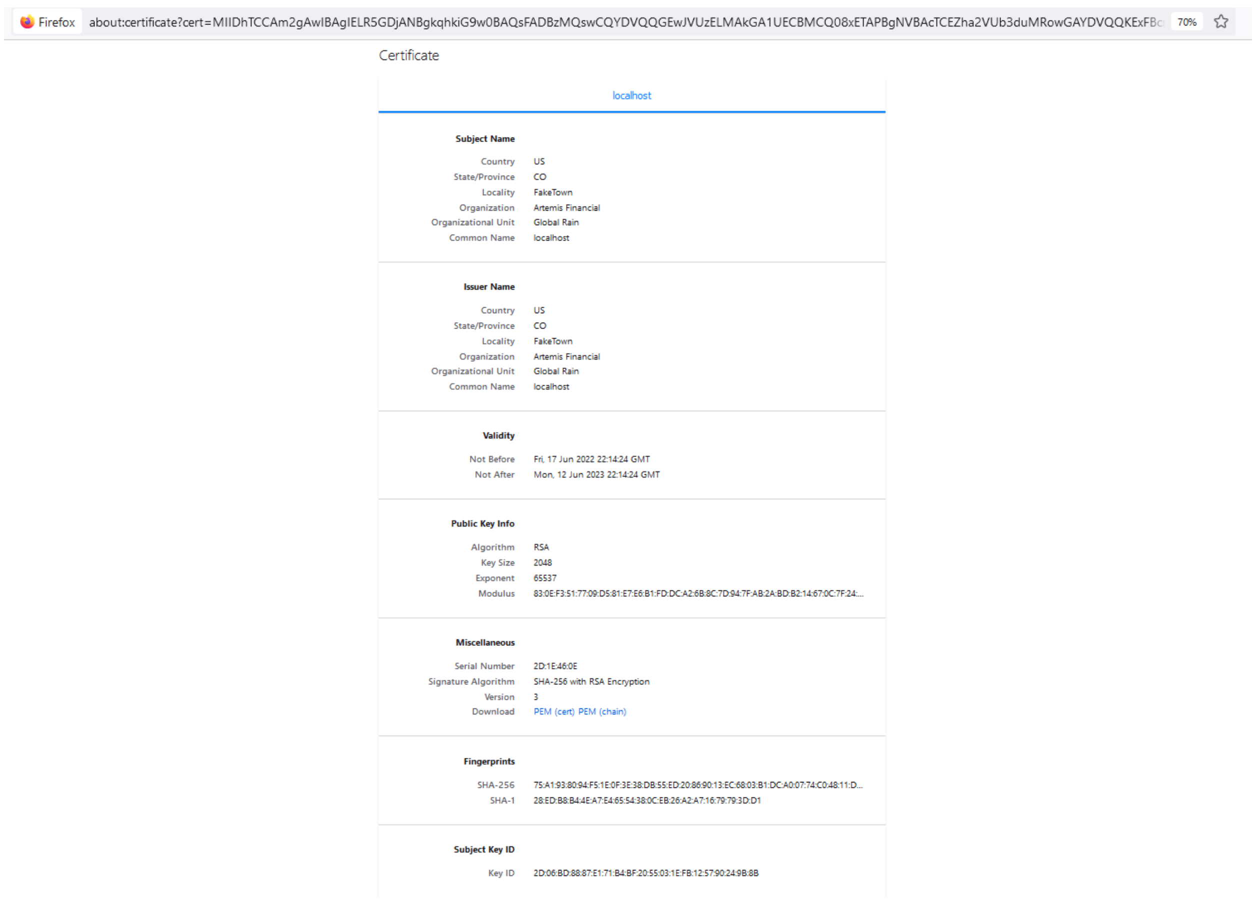
- To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.

HTTPS protocol implementation requires an RSA public/private key for use. HTTPS also requires the server to provide a certificate to verify the authenticity of the server that the client is connecting to. This certificate is typically signed by a trusted third-party after verifying the server information independently, although here we use a self-signed key for demonstration. The keytool provided by Oracle can generate both the RSA key and certificate needed for us to enable HTTPS communication.



```
C:\Users\craig>keytool.exe -printcert -file server.cer
Owner: CN=localhost, OU=Global Rain, O=Artemis Financial, L=FakeTown, ST=CO, C=US
Issuer: CN=localhost, OU=Global Rain, O=Artemis Financial, L=FakeTown, ST=CO, C=US
Serial number: 2d1e460e
Valid from: Fri Jun 17 18:14:24 EDT 2022 until: Mon Jun 12 18:14:24 EDT 2023
Certificate fingerprints:
         SHA1: 28:ED:B8:B4:4E:A7:E4:65:54:38:0C:EB:26:A2:A7:16:79:79:3D:D1
         SHA256: 75:A1:93:80:94:F5:1E:0F:3E:38:DB:55:ED:20:86:90:13:EC:68:03:B1:DC:A0:07:74:C0:48:11:DC:F2:2C:7C
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: 2D 06 BD 88 87 E1 71 B4   BF 20 55 03 1E FB 12 57  -.....q.. U....W
0010: 90 24 9B 8B                                        .$..
]
]
```

Figure 1: Certificate export view from the Oracle Keytool



Firefox about:certificate?cert=MIIDhTCCAm2gAwIBAgIELR5GDjANBgkqhkiG9w0BAQsFADBzMQswCQYDVQQGEwJVUzELMAkGA1UECBMCQ08xETAPBgNVBAcTCEZha2VUb3duMRowGAYDVQQKExFBc 70%

Certificate

localhost

Subject Name	
Country	US
State/Province	CO
Locality	FakeTown
Organization	Artemis Financial
Organizational Unit	Global Rain
Common Name	localhost
Issuer Name	
Country	US
State/Province	CO
Locality	FakeTown
Organization	Artemis Financial
Organizational Unit	Global Rain
Common Name	localhost
Validity	
Not Before	Fri, 17 Jun 2022 22:14:24 GMT
Not After	Mon, 12 Jun 2023 22:14:24 GMT
Public Key Info	
Algorithm	RSA
Key Size	2048
Exponent	65537
Modulus	83:0E:F3:51:77:09:D5:81:E7:E6:B1:FD:DC:A2:6B:8C:7D:94:7F:AB:2A:BD:B2:14:67:0C:7F:24:...
Miscellaneous	
Serial Number	2D:1E:46:0E
Signature Algorithm	SHA-256 with RSA Encryption
Version	3
Download	PEM (cert) PEM (chain)
Fingerprints	
SHA-256	75:A1:93:80:94:F5:1E:0F:3E:38:DB:55:ED:20:86:90:13:EC:68:03:B1:DC:A0:07:74:C0:48:11:D...
SHA-1	28:ED:B8:B4:4E:A7:E4:65:54:38:0C:EB:26:A2:A7:16:79:79:3D:D1
Subject Key ID	
Key ID	2D:06:BD:88:87:E1:71:B4:BF:20:55:03:1E:FB:12:57:90:24:9B:8B

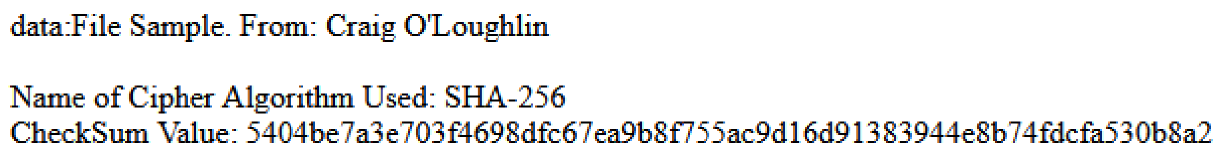
Figure 2: Certificate as the client receives from server

Firefox and many other browsers by default deny HTTPS connections to servers which are self-signed as in this case; we manually allow the connection since we know in this case exactly who our certificate authority is.

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.

Implementing an SHA-256 checksum requires choosing appropriate and well-established hashing libraries. The Java standard library contains a MessageDigest which is used here for hashing a piece of data, and Spring provides a Security library that among other things can convert the raw bytes output into a more friendly hexadecimal display. Again, the checksum here is a combination of 256 bits uniquely identifying a piece of data.

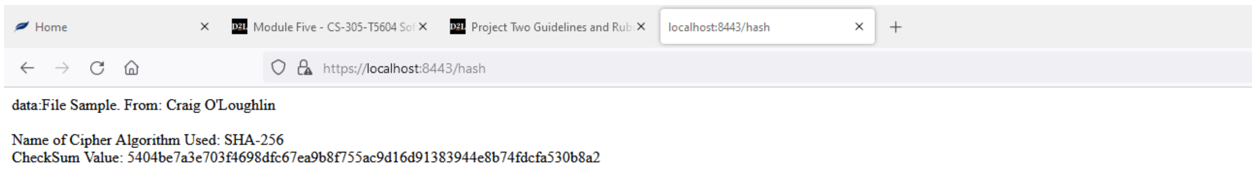


```
data:File Sample. From: Craig O'Loughlin

Name of Cipher Algorithm Used: SHA-256
CheckSum Value: 5404be7a3e703f4698dfc67ea9b8f755ac9d16d91383944e8b74fdcfa530b8a2
```

Figure 3: Server HTML response; sample data and subsequent checksum

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.


Home
Module Five - CS-305-T5604 So
Project Two Guidelines and Rub
localhost:8443/hash
https://localhost:8443/hash

```
data:File Sample. From: Craig O'Loughlin

Name of Cipher Algorithm Used: SHA-256
CheckSum Value: 5404be7a3e703f4698dfc67ea9b8f755ac9d16d91383944e8b74fdcfa530b8a2
```

Figure 4: HTTPS server connection from browser

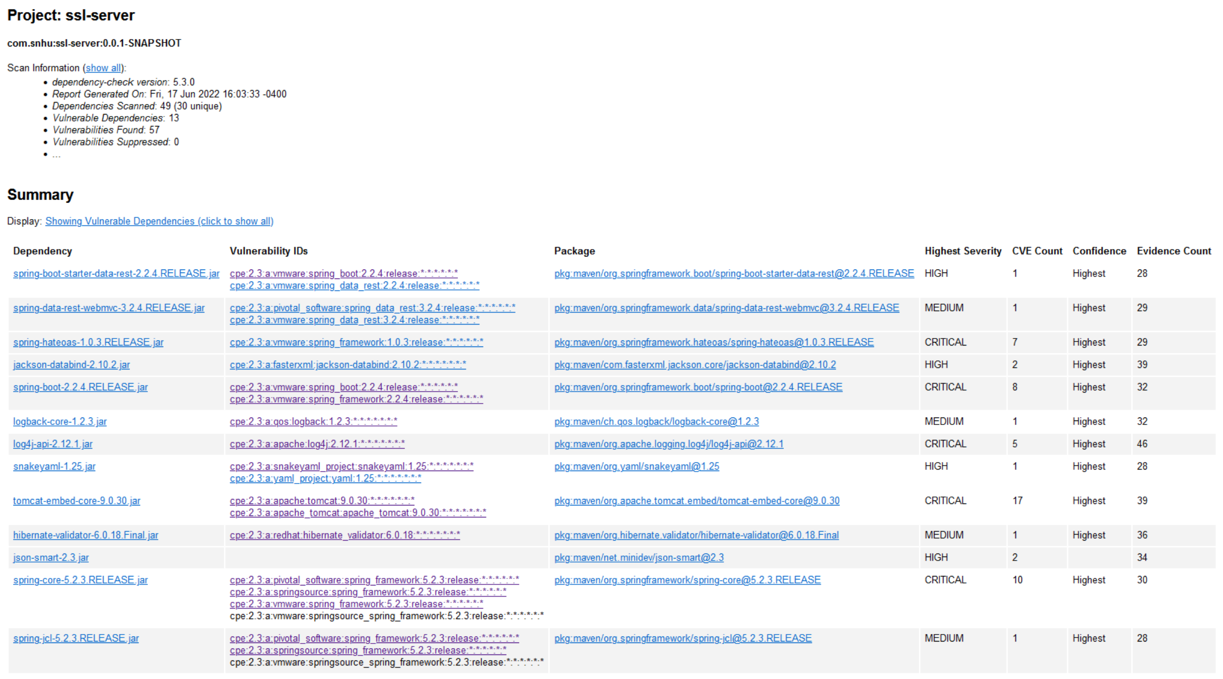
5. Secondary Testing

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

A dependency check ensures that the external libraries in use are considered safe. Vulnerabilities are always being identified and patched in reputable code libraries, a database of known vulnerabilities exists and can be searched against the libraries we are using.

An initial check of dependency vulnerabilities as the code is received at the outset of this project is performed.



Project: ssl-server

com.snhu:ssl-server:0.0.1-SNAPSHOT

Scan Information (show all):

- *dependency-check version*: 5.3.0
- *Report Generated On*: Fri, 17 Jun 2022 16:03:33 -0400
- *Dependencies Scanned*: 49 (30 unique)
- *Vulnerable Dependencies*: 13
- *Vulnerabilities Found*: 57
- *Vulnerabilities Suppressed*: 0
- ...

Summary

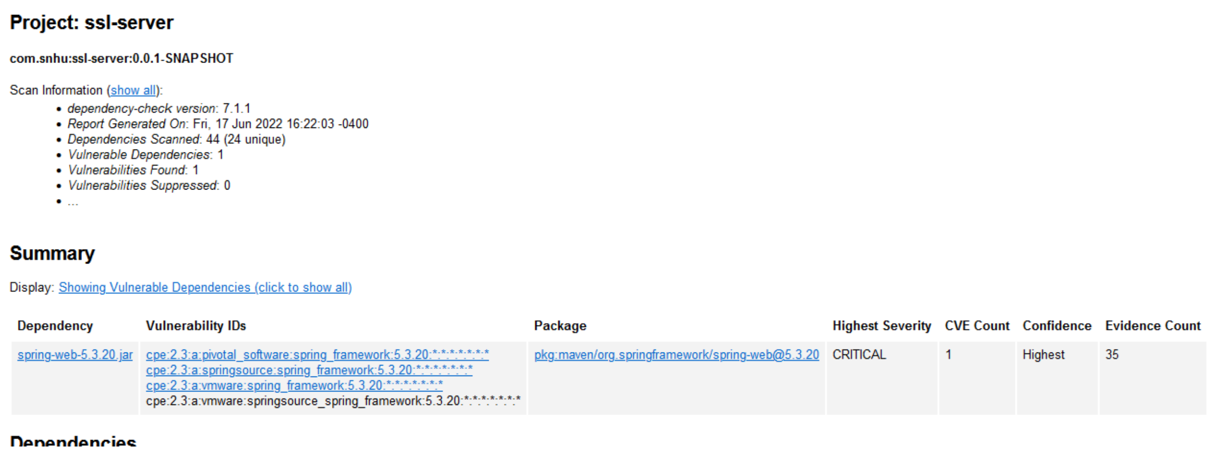
Display: Showing Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
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	28
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	1	Highest	29
spring-hateoas-1.0.3.RELEASE.jar	cpe:2.3:a:vmware:spring_framework:1.0.3:release:*:*:*:*:*:*	pkg:maven/org.springframework.hateoas/spring-hateoas@1.0.3.RELEASE	CRITICAL	7	Highest	29
jackson-databind-2.10.2.jar	cpe:2.3:a:fasterxml:jackson-databind:2.10.2:*:*:*:*:*:*:*	pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.10.2	HIGH	2	Highest	39
spring-boot-2.2.4.RELEASE.jar	cpe:2.3:a:vmware:spring_boot:2.2.4:release:*:*:*:*:*:* cpe:2.3:a:vmware:spring_framework:2.2.4:release:*:*:*:*:*:*	pkg:maven/org.springframework.boot/spring-boot@2.2.4.RELEASE	CRITICAL	8	Highest	32
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	32
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	CRITICAL	5	Highest	46
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	1	Highest	28
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	17	Highest	39
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	36
json-smart-2.3.jar		pkg:maven/net.minidev/json-smart@2.3	HIGH	2		34
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	CRITICAL	10	Highest	30
spring-jcl-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:springsource_spring_framework:5.2.3:release:*:*:*:*:*:*	pkg:maven/org.springframework/spring-jcl@5.2.3.RELEASE	MEDIUM	1	Highest	28

Figure 5: Initial Dependency Check (OWASP)

57 vulnerabilities were identified. The majority of existing vulnerabilities are addressed in subsequent software updates; it is imperative to use the latest versions of external libraries.

After updating Spring Framework and OWASP to the latest versions:



Project: ssl-server

com.snhu:ssl-server:0.0.1-SNAPSHOT

Scan Information (show all):

- *dependency-check version*: 7.1.1
- *Report Generated On*: Fri, 17 Jun 2022 16:22:03 -0400
- *Dependencies Scanned*: 44 (24 unique)
- *Vulnerable Dependencies*: 1
- *Vulnerabilities Found*: 1
- *Vulnerabilities Suppressed*: 0
- ...

Summary

Display: Showing Vulnerable Dependencies (click to show all)

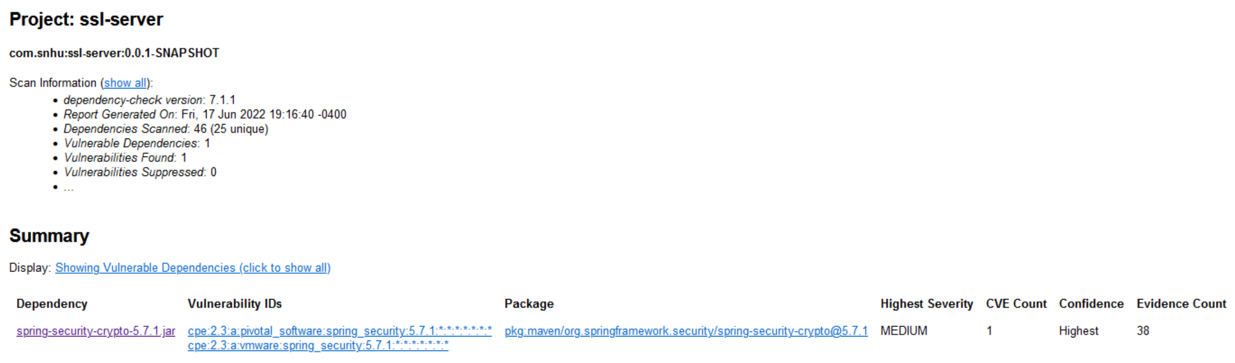
Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
spring-web-5.3.20.jar	cpe:2.3:a:pivotal_software:spring_framework:5.3.20:*:*:*:*:*:*:* cpe:2.3:a:springsource:spring_framework:5.3.20:*:*:*:*:*:*:* cpe:2.3:a:vmware:spring_framework:5.3.20:*:*:*:*:*:*:* cpe:2.3:a:vmware:springsource_spring_framework:5.3.20:*:*:*:*:*:*:*	pkg:maven/org.springframework/spring-web@5.3.20	CRITICAL	1	Highest	35

Dependencies

Figure 6: Dependency check after framework update

The check finds only a single vulnerability: CVE-2016-1000027, which Spring claims to have addressed in a previous version.

After adding the security groundwork as in the previous steps, we run the dependency check again in figure 7 to ensure no new vulnerabilities are introduced due to external libraries. We also suppress false positives to avoid confusion.



Project: ssl-server

com.snhu:ssl-server:0.0.1-SNAPSHOT

Scan Information (show all):

- *dependency-check version*: 7.1.1
- *Report Generated On*: Fri, 17 Jun 2022 19:16:40 -0400
- *Dependencies Scanned*: 46 (25 unique)
- *Vulnerable Dependencies*: 1
- *Vulnerabilities Found*: 1
- *Vulnerabilities Suppressed*: 0
- ...

Summary

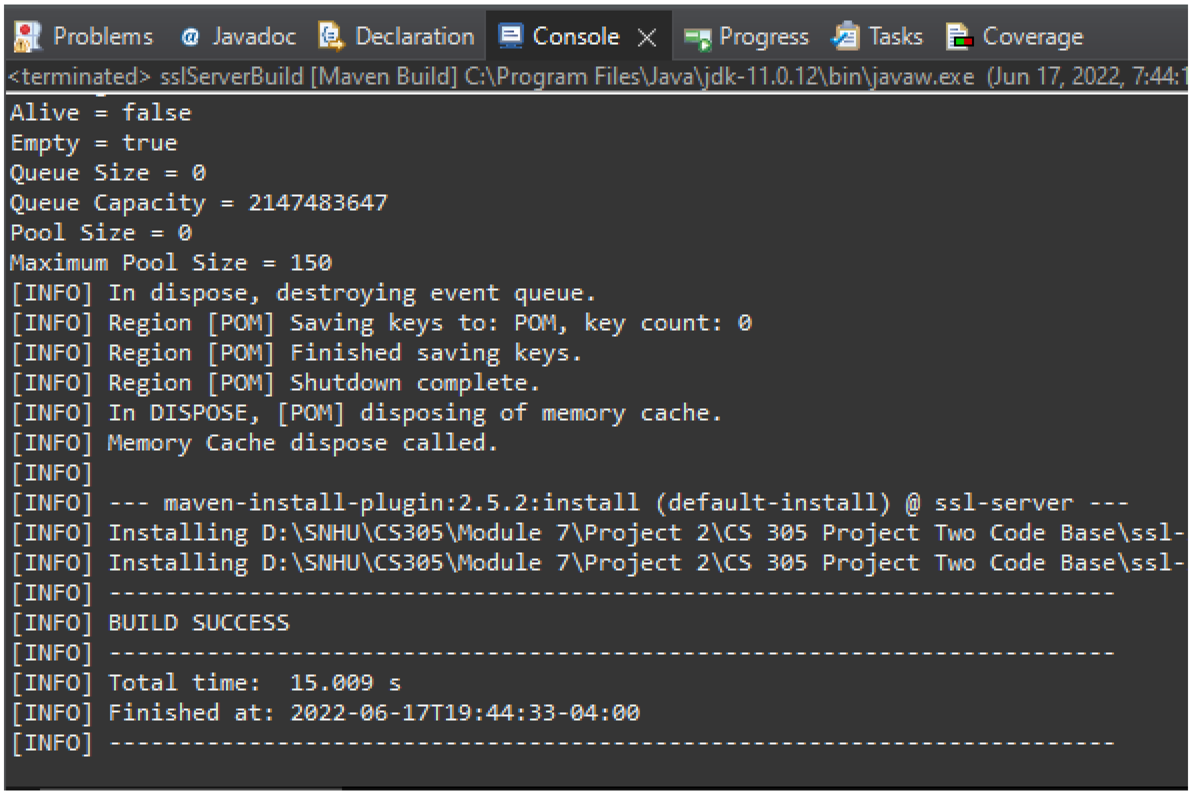
Display: Showing Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
spring-security-crypto-5.7.1.jar	cpe:2.3:a:pivotal_software:spring_security:5.7.1:*:*:*:*:*:*:* cpe:2.3:a:vmware:spring_security:5.7.1:*:*:*:*:*:*:*	pkg:maven/org.springframework.security/spring-security-crypto@5.7.1	MEDIUM	1	Highest	38

Figure 7: Dependency check after code additions

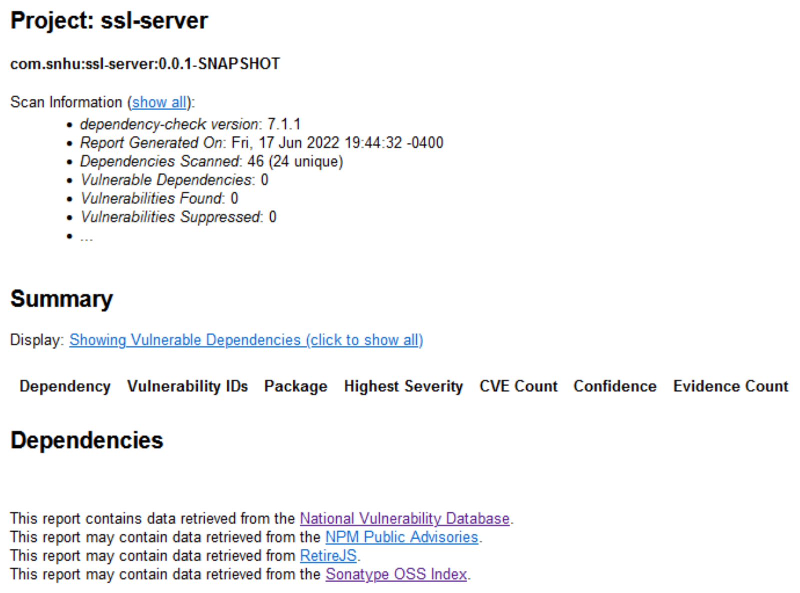
A new vulnerability is found with our chosen Spring Security v 5.7.1. However, CVE-2020-5408 was addressed two years ago with an update from Spring (https://spring.io/blog/2020/05/13/cve-reports-published-for-spring-security). This is known as a false positive.

A final check is run with all false positives suppressed.



```
<terminated> sslServerBuild [Maven Build] C:\Program Files\Java\jdk-11.0.12\bin\javaw.exe (Jun 17, 2022, 7:44:1
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 D:\SNHU\CS305\Module 7\Project 2\CS 305 Project Two Code Base\ssl-
[INFO] Installing D:\SNHU\CS305\Module 7\Project 2\CS 305 Project Two Code Base\ssl-
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  15.009 s
[INFO] Finished at: 2022-06-17T19:44:33-04:00
[INFO] ------------------------------------------------------------------------
```

Figure 8: Build Success



Project: ssl-server

com.snhu:ssl-server:0.0.1-SNAPSHOT

Scan Information (show all):

- *dependency-check version*: 7.1.1
- *Report Generated On*: Fri, 17 Jun 2022 19:44:32 -0400
- *Dependencies Scanned*: 46 (24 unique)
- *Vulnerable Dependencies*: 0
- *Vulnerabilities Found*: 0
- *Vulnerabilities Suppressed*: 0
- ...

Summary

Display: Showing Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count

Dependencies

This report contains data retrieved from the National Vulnerability Database.
This report may contain data retrieved from the NPM Public Advisories.
This report may contain data retrieved from RetireJS.
This report may contain data retrieved from the Sonatype OSS Index.

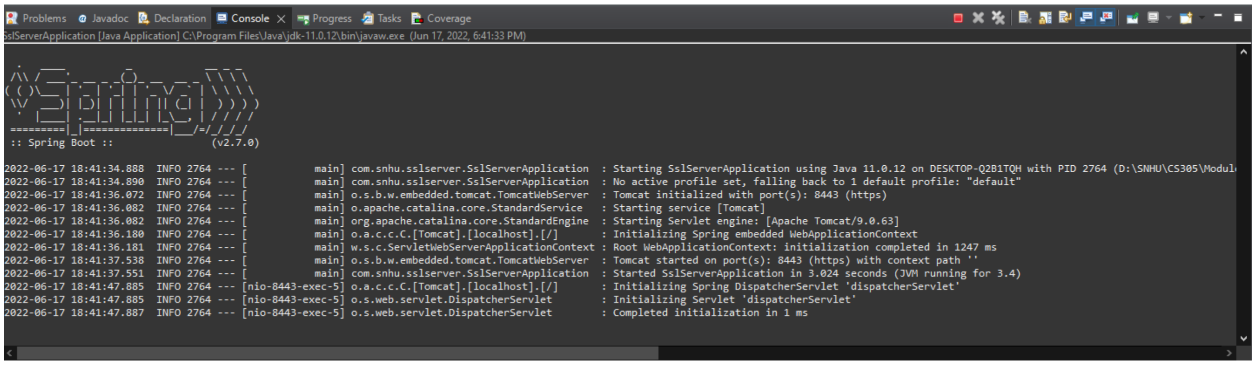
Figure 9: No known vulnerabilities in scanned dependencies

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.

We have provided the groundwork for a secure web application. The refactored code including the hash demonstration runs on the Spring framework.

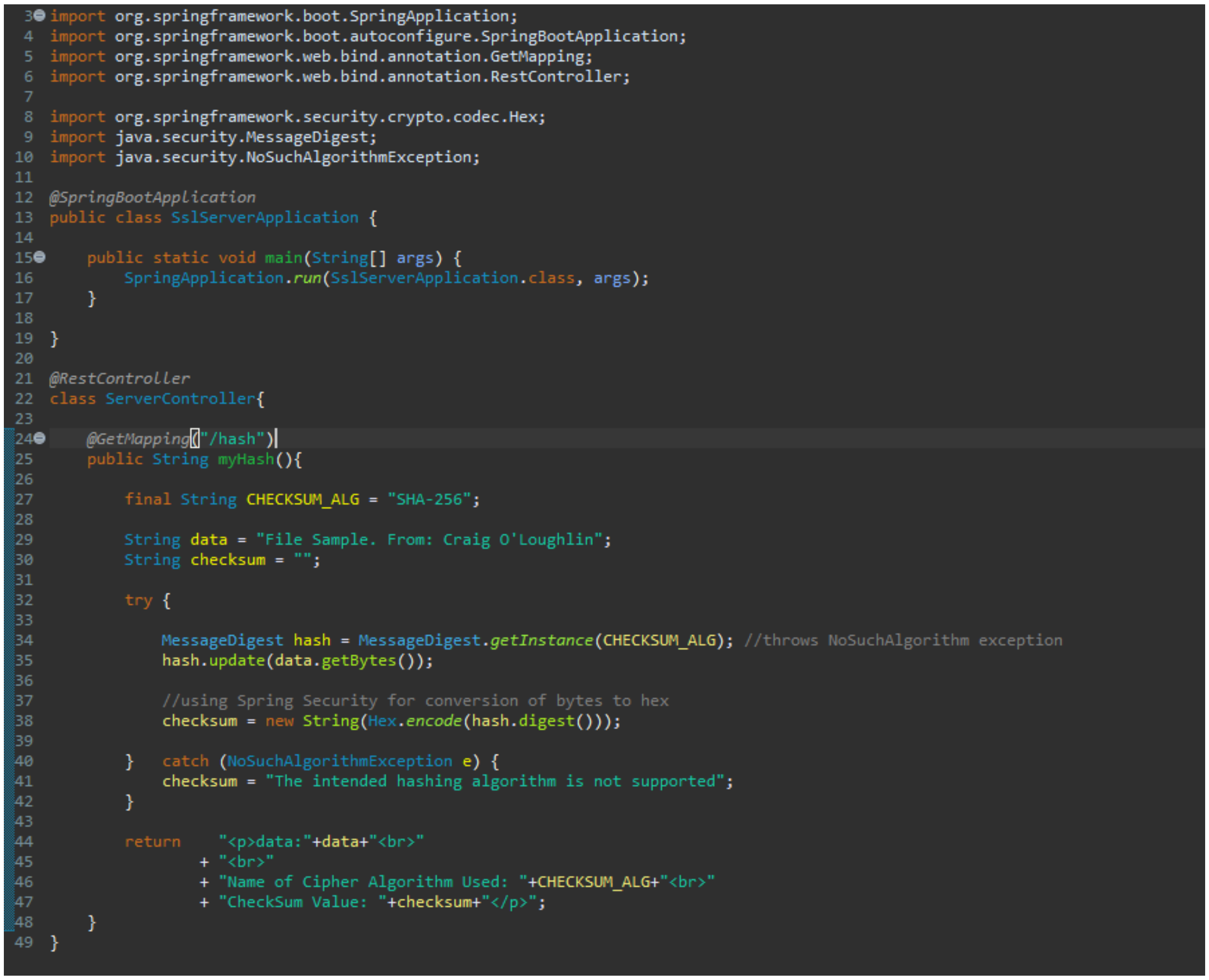


```
SslServerApplication [Java Application] C:\Program Files\Java\jdk-11.0.12\bin\javaw.exe  (Jun 17, 2022, 6:41:33 PM)

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.0)

2022-06-17 18:41:34.888  INFO 2764 --- [           main] com.snhu.sslserver.SslServerApplication  : Starting SslServerApplication using Java 11.0.12 on DESKTOP-Q2B1TQH with PID 2764 (D:\SNHU\CS305\Modul
2022-06-17 18:41:34.890  INFO 2764 --- [           main] com.snhu.sslserver.SslServerApplication  : No active profile set, falling back to 1 default profile: "default"
2022-06-17 18:41:36.072  INFO 2764 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8443 (https)
2022-06-17 18:41:36.082  INFO 2764 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-06-17 18:41:36.082  INFO 2764 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.63]
2022-06-17 18:41:36.180  INFO 2764 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-06-17 18:41:36.181  INFO 2764 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1247 ms
2022-06-17 18:41:37.538  INFO 2764 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8443 (https) with context path ''
2022-06-17 18:41:37.551  INFO 2764 --- [           main] com.snhu.sslserver.SslServerApplication  : Started SslServerApplication in 3.024 seconds (JVM running for 3.4)
2022-06-17 18:41:47.885  INFO 2764 --- [nio-8443-exec-5] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-06-17 18:41:47.885  INFO 2764 --- [nio-8443-exec-5] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2022-06-17 18:41:47.887  INFO 2764 --- [nio-8443-exec-5] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1 ms
```

Figure 10: Spring running Artemis Financial code and accepting connections

The majority of this project exists in external libraries. Our portion of the code is included below for review.

Our additions are straightforward and do not introduce security vulnerabilities. The @GetMapping is used instead of the more ambiguous @RequestMapping annotation; it is always important to define as close to intention as possible. All variables are initialized on use. No untrusted data is taken from the client in this example. Exceptions are caught close to source. See figure 11 below.



```java
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import org.springframework.security.crypto.codec.Hex;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

@SpringBootApplication
public class SslServerApplication {

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

}

@RestController
class ServerController{

    @GetMapping("/hash")
    public String myHash(){

        final String CHECKSUM_ALG = "SHA-256";

        String data = "File Sample. From: Craig O'Loughlin";
        String checksum = "";

        try {

            MessageDigest hash = MessageDigest.getInstance(CHECKSUM_ALG); //throws NoSuchAlgorithm exception
            hash.update(data.getBytes());

            //using Spring Security for conversion of bytes to hex
            checksum = new String(Hex.encode(hash.digest()));

        }   catch (NoSuchAlgorithmException e) {
            checksum = "The intended hashing algorithm is not supported";
        }

        return    "<p>data:"+data+"<br>"
                + "<br>"
                + "Name of Cipher Algorithm Used: "+CHECKSUM_ALG+"<br>"
                + "CheckSum Value: "+checksum+"</p>";
    }
}
```

Figure 11: Server boot and /hash URI

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 foundation of this web application has been secured through the use of HTTPS in communication with RSA encryption and certificate use, through a building block for checksum integration for file verification, and through verification of external libraries for known vulnerabilities.

We have considered vulnerabilities covering the topics of Cryptography, Client / Server security, and Code Quality. In the review of the code security, we have further considered Data Access, Services, and APIs.

Adding security to any computer program begins at the foundation. Security should be built into the program from the ground up and should not be an afterthought. The selection of the technology stack should be informed in large part by security principles and the application field that will be deployed to.

Going forward from this point, security considerations will continue to be especially important in securing and trusting client entered data and proper access to information databases. Even after deployment, security will need to be continuously considered and improved, monitored and patched for new threats and vulnerabilities as they are detected.

Encryption of customer data will need to be implemented as required by US law, and common best practices should be followed such as sanitization of all input data and gathering all input data over HTTPS via POST commands. Database security if hosted at a different server location should be independently inspected and verified for use. Using the most up to date versions of external dependencies will continue to be critical, and resources should be made available for checking and refactoring into the codebase if needed throughout the lifecycle of this application.