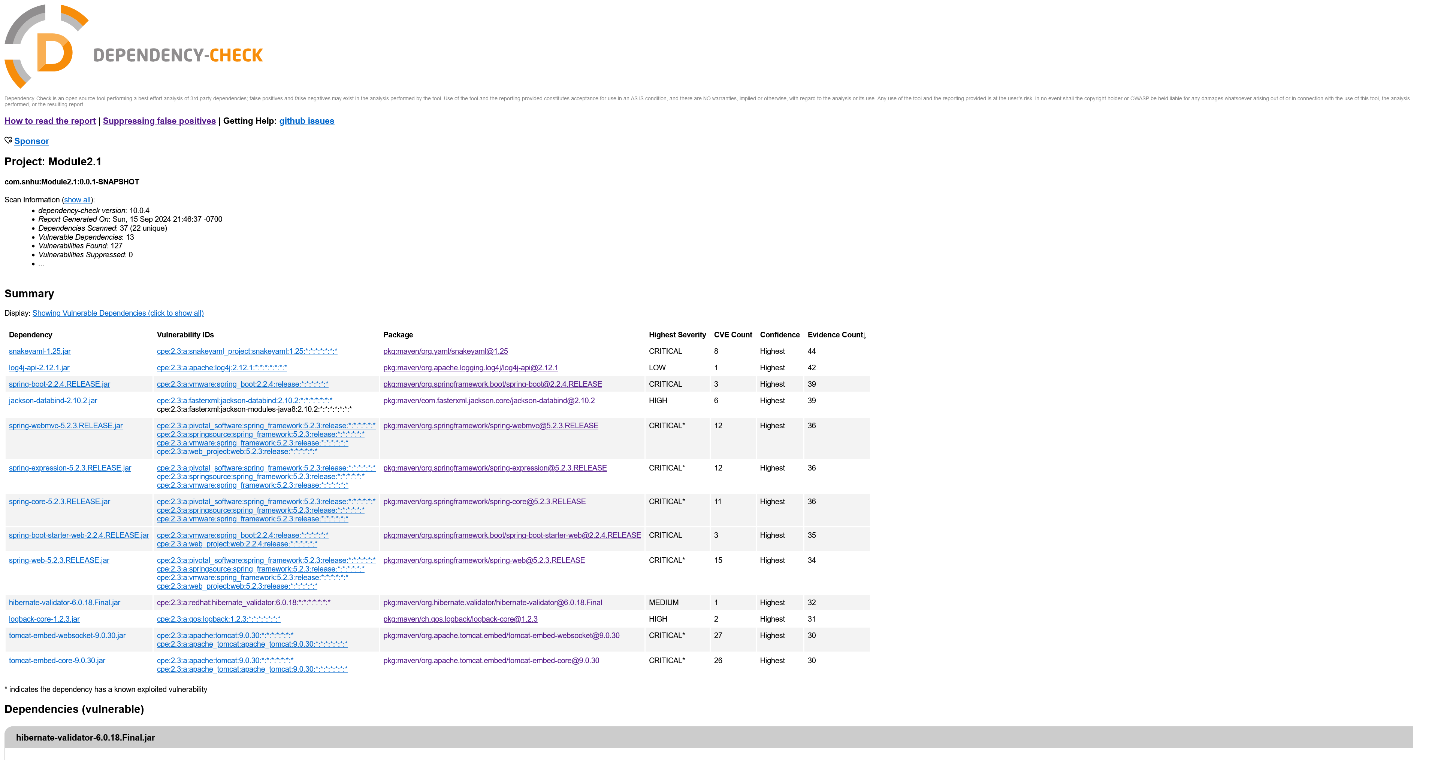
# CS 305 Module Two Coding Assignment Template

## Instructions

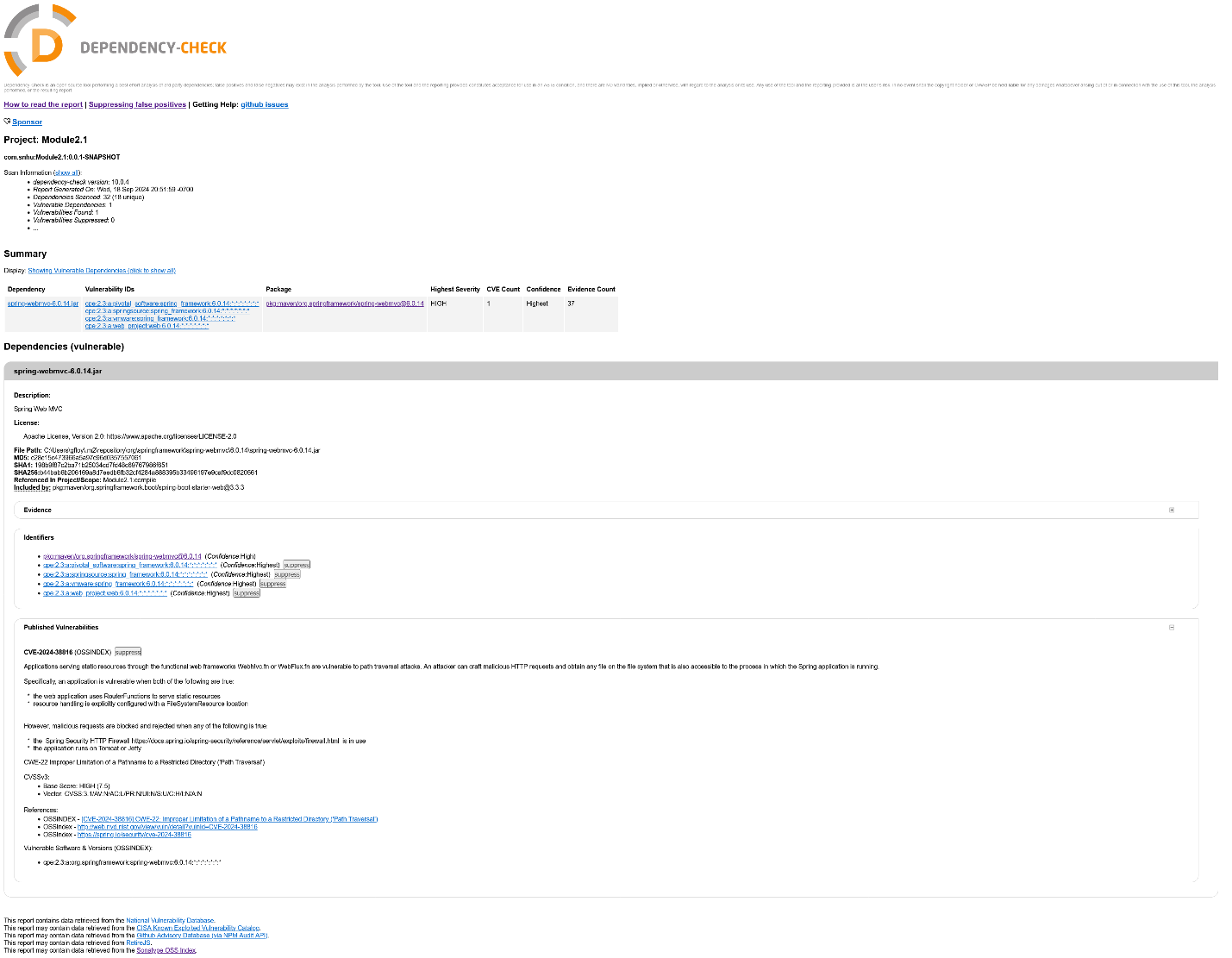
Replace the bracketed text with the relevant information in your own words. If you choose to include images or supporting materials, make certain to insert them in all the relevant locations in the document.

## Run Dependency Check

This image is before applying any fixes:



After investigating the POM file and updating dependencies:



## Document Results

The issue that persisted was with “spring-webmvc-6.0.14.jar”, as the code would not compile on non-vulnerable versions. The rest of the issues that were eliminated were as follows:

#### hibernate-validator-6.0.18.Final.jar

CVE-2020-10693 in Hibernate's Bean Validation (JSR-380) reference implementation which had a bug that allowed attackers to bypass input sanitization through web page interactions.

#### jackson-databind-2.10.2.jar

CVE-2020-25649, CVE-2020-36518, CVE-2021-46877, CVE-2022-42003, CVE-2022-42004, and CVE-2023-35116 in “jackson-databind-2.10.2.jar”, a data-binder for Jackson. It had a vulnerability to external entity attacks through XML, as well as multiple denial of service attack vectors, an out of bounds write issue, and deserializations of untrusted data in extreme circumstances.

#### log4j-api-2.12.1.jar

CVE-2020-9488 affects the logging functionality of this Apache API as it allows for a man-in-the-middle attack over SMTPS to reveal logs sent through this API.

#### logback-core-1.2.3.jar

CVE-2023-6378 allows an attacker to send poisoned data to the log system to mount a Denial of Service (DoS) attack.

CVE-2021-42550 affects systems where if you have poor access control, an attacker can edit a config file to execute arbitrary code.

#### snakeyaml-1.25.jar

This YAML parser has CVE-2022-1471, a remote code execution (RCE) vulnerability caused by the constructor class not restricting its data types during deserialization.

Snakeyaml also has seven different denial of service attack vectors with CVE-2017-18640, CVE-2022-25857, CVE-2022-38749, CVE-2022-38751, CVE-2022-38752, CVE-2022-41854, and CVE-2022-38750.

#### spring-boot-2.2.4.RELEASE.jar

Spring Boot, a part of the Spring framework that allows us to create Spring applications easier, has the following vulnerabilities: CVE-2023-20873, a false positive that requires Cloud Foundry, CVE-2022-27772 a false positive that affects only very old systems that share a temporary directory between all users, and CVE-2023-20883, another false positive that requires the use of a reverse proxy.

#### spring-boot-starter-web-2.2.4.RELEASE.jar

Part of the Spring framework, this dependency has a known exploited RCE through CVE-2022-22965 if you are running Tomcat as a WAR deployment with a non-default Spring Boot executable jar. It also has privilege escalation through CVE-2021-22118, Reflected File Download through CVE-2020-5421, denial of service through CVE-2022-22950, CVE-2022-22971, CVE-2023-20861, CVE-2023-20863, and CVE-2022-22970. It also has CVE-2022-22968 which is an issue with case sensitivity affecting the disallowedFields on DataBinder and CVE-2021-22096 for malicious insertion of log entries.

#### spring-core-5.2.3.RELEASE.jar AND spring-expression-5.2.3.RELEASE.jar AND spring-webmvc-5.2.3.RELEASE.jar

The Spring Expression Language (SpEL) has a known exploited RCE CVE-2022-22965. The vulnerabilities in these three jar files are reflected in the Spring Boot Starter Web part above.

#### spring-web-5.2.3.RELEASE.jar

The Spring Web jar also has an RCE that may or may not be present depending on how untrusted data is handled, documented in CVE-2016-1000027. It also has a known exploited vulnerability with CVE-2022-22965 as does the rest of the Spring family in these versions. It does have its own unique issues, including CVE-2024-38809 which is a regular expression denial of service (ReDoS), as well as CVE-2024-22243 and CVE-2024-22262 dealing with its URL parser.

#### tomcat-embed-core-9.0.30.jar AND tomcat-embed-websocket-9.0.30.jar

The WebSocket service Tomcat has a known exploited vulnerability CVE-2020-1938, a privilege management vulnerability, as well as CVE-2023-44487, a distributed denial of service (DDoS). It also has DoS attacks and increased risk of them with CVE-2020-11996, CVE-2020-13934, CVE-2020-13935, CVE-2021-41079, and CVE-2022-29885. It also had exposure of sensitive information with CVE-2020-17527, CVE-2021-25122, CVE-2021-24122, CVE-2024-21733, and CVE-2020-13943. Similarly, from improper cleanup information could leak from CVE-2023-42795. Furthermore, Tomcat had a HTTP request/response smuggling issue with CVE-2022-42252, CVE-2023-46589, CVE-2021-33037, CVE-2019-17569, and CVE-2020-1935. There is also a vulnerability with bypassing LockOut Realm with CVE-2021-30640 and similarly an input validation issue with CVE-2023-45648, as well as a cross-site scripting (XSS) vulnerability with CVE-2022-34305, and an open redirect with CVE-2023-41080. There was also a problem with session cookie handling with reverse proxies documented in CVE-2023-28708, and a hard to replicate race condition in CVE-2021-43980. Finally, under very specific conditions, CVE-2020-9484 would allow for the deserialization of untrusted data.

## Analyze Results

All the vulnerabilities except CVE-2024-38816 in “spring-webmvc-6.0.14.jar” were fixed by updating dependencies. I could not get a non-vulnerable version of Spring’s webmvc to compile from updating it in the POM file. The updated POM is attached in this document below. The vulnerability itself is entirely negated by using the Spring Security HTTP Firewall, or if our application will run on Tomcat or Jetty. Since we have the Tomcat modules, it is safe to assume that this is then a false positive and can be securely filtered out for now. We now have no **known** vulnerabilities found by the OWASP Dependency Checker from these dependency updates.

#### **←Click here to expand / collapse the POM**

<?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>3.0.13</version> <!-- WAS: 2.2.4.RELEASE -->

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.snhu</groupId>

<artifactId>Module2.1</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>Module2.1</name>

<description>Demo project for Spring Boot</description>

<properties>

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

</properties>

<dependencies>

<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-web -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

<version>3.3.3</version>

</dependency>

<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.17.2</version>

</dependency>

<!-- https://mvnrepository.com/artifact/ch.qos.logback/logback-classic -->

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.4.11</version>

<scope>test</scope>

</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>

<version>3.0.13</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework.data/spring-data-rest-webmvc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>6.1.6</version>

</dependency>

<dependency>

<groupId>org.yaml</groupId>

<artifactId>snakeyaml</artifactId>

<version>2.3</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>6.1.13</version>

</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>10.0.4</version>

<executions>

<execution>

<goals>

<goal>check</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

#### .

Discussing some of the fixed vulnerabilities, in logback-core-1.2.3.jar, CVE-2021-42550 is an example of a bug that on its own is quite harmless if access control is properly implemented, but any seasoned penetration tester will inform you that they rarely use just one vulnerability, but chains of them. So, while it may appear as a false positive since our program does not have such access, if somewhere else in the program we have a central authentication server built on LDAP it can be used as a stepping stone to ACE. Both RCE and ACE are vulnerabilities that should always raise maximum alarm, and since it is already patched, updating is truly a no-brainer even if we don’t use LDAP.

One more thing to note about false positive flags in code is also addressing the attitude of “not my problem” in software. If we ever reuse some of the code in this project somewhere else and bring a few dependencies over with it to save time, a previously suppressed vulnerability could abruptly become no longer a false positive. When the solution to the vulnerability is simply updating a library, we leave only ourselves to blame. These are fixed issues. In fact, every vulnerability save for the already fixed CVE-2024-38816 was addressed by simply updating our dependencies. No changes to our patterns are necessary at this time.