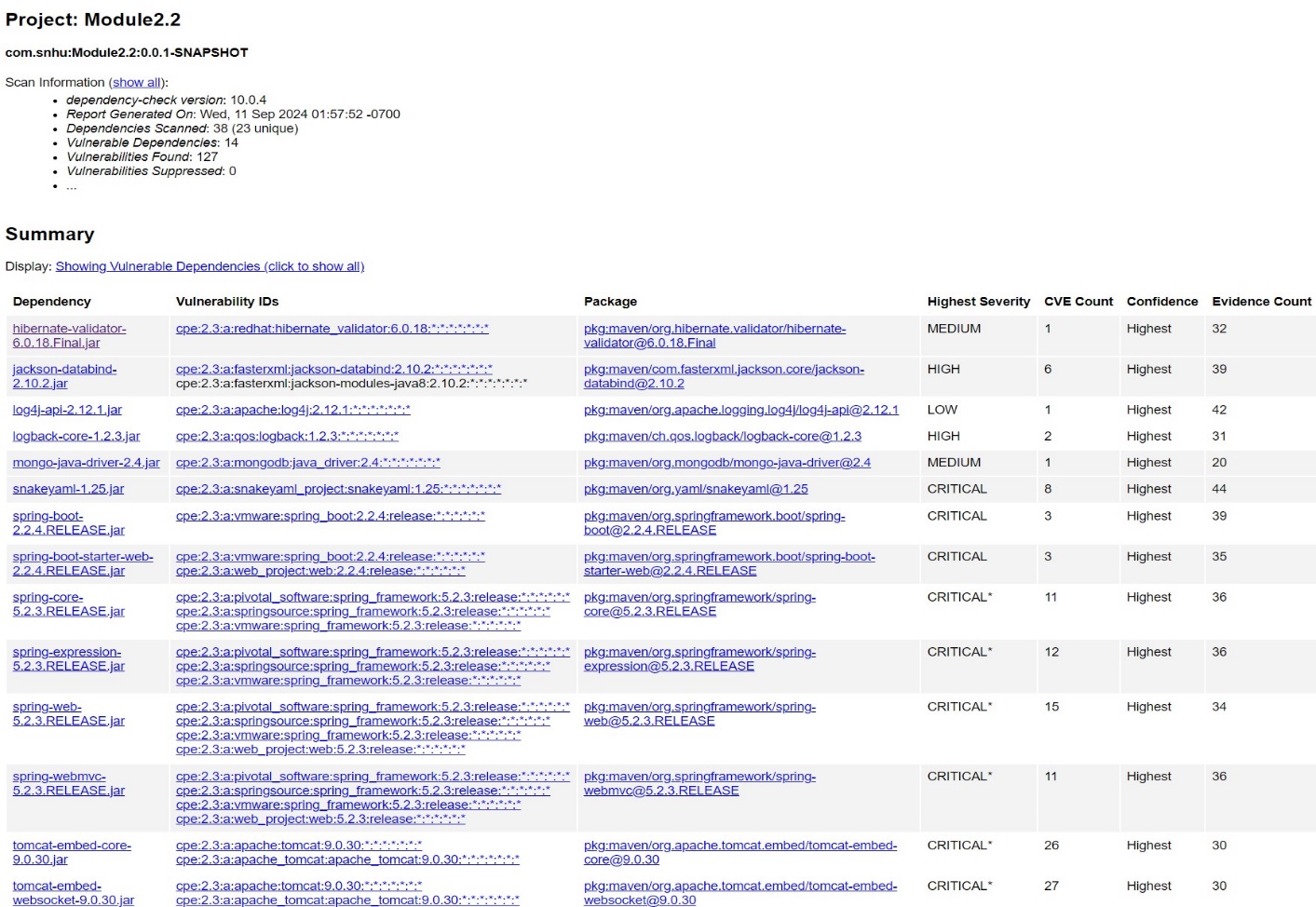
# CS 305 Module Two Coding Assignment

## Run Dependency Check



## Document Results

Overall, in our dependency verification we discovered 14 dependencies in our code that were vulnerable to security issues, with 127 total vulnerabilities discovered from those 14 dependencies:

1. hibernate-validator-6.0.18.Final.jar – 1
   1. Description: Hibernate's Bean Validation (JSR-380) reference implementation.
2. jackson-databind-2.10.2.jar
   1. Description: General data-binding functionality for Jackson: works on core streaming API
3. log4j-api-2.12.1.jar
   1. Description: The Apache Log4j API
4. logback-core-1.2.3.jar
   1. Description: logback-core module
5. mongo-java-driver-2.4.jar
   1. Description: Java Driver for MongoDB
6. snakeyaml-1.25.jar
   1. Description: YAML 1.1 parser and emitter for Java
7. spring-boot-2.2.4.RELEASE.jar
   1. Description: Spring Boot
8. spring-boot-starter-web-2.2.4.RELEASE.jar
   1. Description: Starter for building web, including RESTful, applications using Spring

MVC. Uses Tomcat as the default embedded container.

1. spring-core-5.2.3.RELEASE.jar
   1. Description: Spring Core
2. spring-expression-5.2.3.RELEASE.jar
   1. Description: Spring Expression Language (SpEL)
3. spring-web-5.2.3.RELEASE.jar
   1. Description: Spring Web
4. spring-webmvc-5.2.3.RELEASE.jar
   1. Description: Spring Web MVC
5. tomcat-embed-core-9.0.30.jar
   1. Description: Core Tomcat Implementation
6. tomcat-embed-websocket-9.0.30.jar
   1. Description: Core Tomcat Implementation

## Analyze Results

With this many vulnerabilities, the obvious solution would be to look for updated versions of our dependencies that we could utilize from Maven’s repository that are more secure. Taking the list from the last section, I’ve sorted the depdencies by CVE count, and have also only displayed the Criticals and Highs, as they are the most crucial to address:

Tomcat Embed WebSocket – 27 CVEs with Criticals

Current Version: 9.0.30

Updated Available Version: 10.1.20 – No CVEs Listed

Tomcat Embedded Core – 26 CVEs with Criticals

Current Version: 9.0.30

Updated Available Version: 10.1.20 – No CVEs Listed

Spring Web – 15 CVEs with 15 Criticals

Current Version: 5.2.3

Updated Available Version: 6.0.20 – No CVEs Listed

Spring Expression – 12 CVEs with Criticals

Current Version: 5.2.3

Updated Available Version: 6.0.8 – No CVEs Listed

Spring Web MVC – 11 CVEs with Criticals

Current Version: 5.2.3

Updated Available Version: 6.0.14 – No CVEs Listed

Spring Core – 11 CVEs with Criticals

Current Version: 5.2.3

Updated Available Version: 6.0.14 – No CVEs Listed

Snake Yaml – 8 CVEs with Criticals

Current Version: 1.25

Updated Available Version: 2.0 – No CVEs Listed

Jackson Databind – 6 CVEs with Highs

Current Version: 2.10.2

Updated Available Version: 2.15.0 – No CVEs Listed

Spring Boot Starter – 3 CVEs with Criticals

Current Version: 2.2.4

Updated Available Version: 2.7.0 – No CVEs Listed

Logback Core – 2 CVEs with Highs

Current Version: 1.2.3

Updated Available Version: 1.5.0 – No CVEs Listed

The remaining hits on our summary are for medium and low severities. Considering that rebuilding a project utilizing higher versions of dependencies can be a tricky task in making everything work together, I would recommend we consider ensuring that our High and Critical vulnerabilities are addressed first. Then, we can save the Mediums and Lows for after the major vulnerabilities have been sorted out, and the majority of the application is secure.

When looking at updating these dependencies, it’s important to look at the nearest version to our current one that hasn’t had issues in its builds. Finding a version series that has no known CVEs should be the goal, as it ensures there aren’t any other known problematic issues with the update series that could demonstrate future potential issues with other updates. We should also look at determining if the dependency we require is actually needed for our project – if it’s not, we shouldn’t include it.

It’s also important to update dependencies that are on similar versions to each other to also similar updated versions. As an example, Spring MVC, Core, Expression, and Web were all using version 5.2.3. In an effort to ensure compatibility, we should make sure that all updated versions fall into the same larger series (6.0.0) to have the best chance of being compatabile, assuming that these dependencies align together.

Finally, there’s the case of false positives and low confidence. Some of the identifiers for our depdencies are tagged with a confidence value of “Low”. In an effort to remain efficient, we should suppress these identifiers, and re-observe our depency hits to notice any changes in our report. In our case, suppressing identifiers that have low confidence wouldn’t meaningfully change our report.