Survey - Practitioner's Perspective on the MVN+ of Vulnerability Management in Maven Ecosystem

Dear Participant,

We are conducting a survey to understand practitioners' perceptions of vulnerability management in the Maven ecosystem. We are providing a vulnerability management tool, **MVN+**, and comparing it to the existing public platforms the Maven Repository and the Open Source Insights. This survey will take you around 10 minutes.

Background
Modern software often reuses functionality from other software libraries (i.e., dependencies) to provide a full working system. For Java projects, the Maven plays the central role of a software supply chain to manage software reuse.

Vulnerability propagation in the supply chain
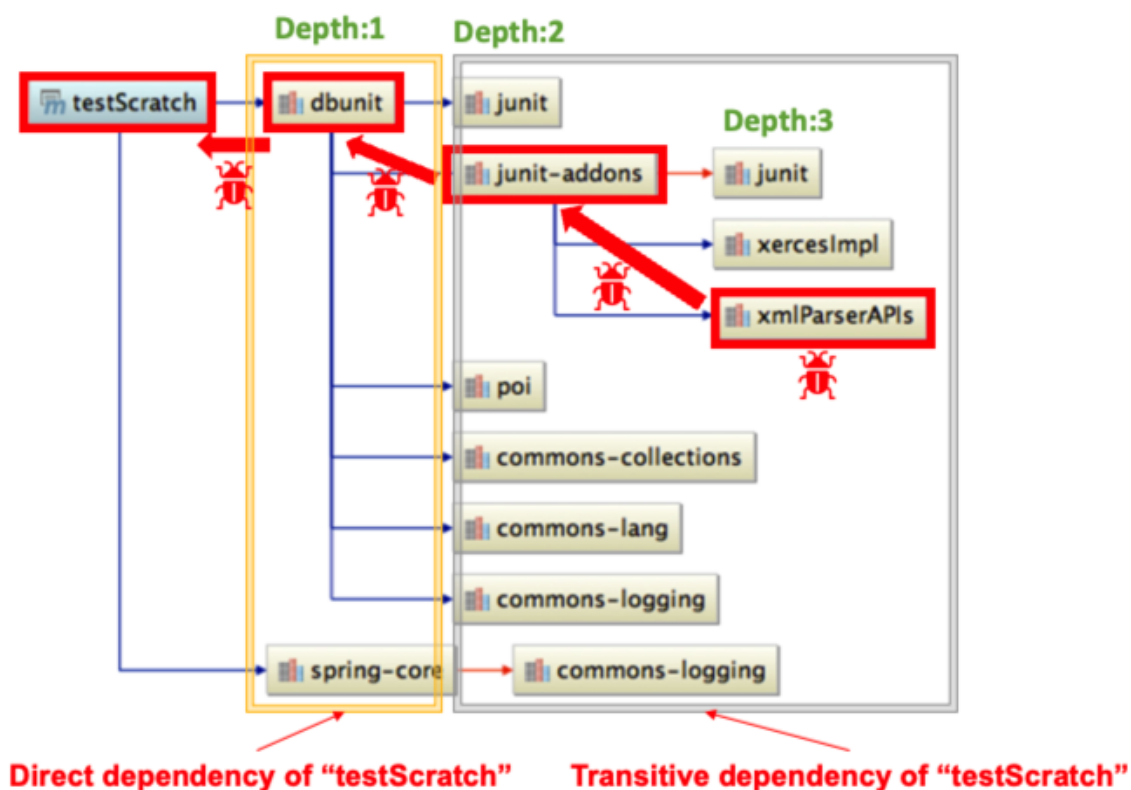
The figure above is an example of a Maven project **"testScratch"**.
- The dependency in the yellow box and the gray box are direct dependency and transitive dependency of **"testScratch"**, respectively.

- If we found a vulnerability from the dependency **"xmlParserAPIs"**, the vulnerability would **propagate** from the dependency chain to the project "**testScratch**".

Although Maven provides great convenience to developers, there are also potential risks related to the quality of the dependencies. So, for the developers who reused other libraries, they will try to avoid the vulnerabilities impacting the whole project.

Current *state-of-the-art* in Vulnerability Management

Below, there are three links showing the vulnerability management result one library (org.apache.spark：spark-core_2.11：1.2.2) on **Maven Repository**, **Open Source Insights** and our tool **MVN+.**

[Maven Repository result of library (org.apache.spark：spark-core_2.11：1.2.2)](#)

[Open Source Insights result of library (org.apache.spark：spark-core_2.11：1.2.2)](#)

[MVN+ result of library (org.apache.spark：spark-core_2.11：1.2.2)](#)

This library is the example we used in this questionnaire. You can click to visit each link to get the full results for the library on each platform. We also provide screenshots of relevant parts in the questions below.

The main goal of this survey is to investigate:
(1) **user satisfaction** of three platforms;
(2) examine the **importance** of each aspect;
(3) **user feedback** on our approach.

We will not reveal your identity in any form of the research outcomes. All of the information provided in the survey will be used for research purposes only. We sincerely appreciate your participation in our study.

Part 1: Information about the Participant

In this part, we will ask you for some background information related to your role and experience.
For each question, please type your answer in the corresponding area.

Q1: Which of the following best describes your primary job role?
    A. Developer B. Algorithm engineer C. Test engineer D. Project manager
    E. Researcher F. Data analyst G. Other (Please specify below)
Answer: E

Q2: How many years of experience do you have in software development/software maintenance/programming?
Answer: 4 years

Part 2: Vulnerability Listing

Maven Repository shows the IDs and the total number of vulnerabilities from each dependency. If you want to know details of the vulnerabilities, you should click the link and go to the homepage of the dependency library (e.g., go to **netty-all** to find the details).

Home » org.apache.spark » spark-core_2.11 » 1.2.2

Spark Project Core » 1.2.2

Core libraries for Apache Spark, a unified analytics engine for large-

License	Apache 2.0
Categories	Distributed Computing
Tags	computing distributed spark apache
HomePage	http://spark.apache.org/
Date	Apr 05, 2015
Files	pom (18 KB) jar (6.7 MB) View All
Repositories	Central BeDataDriven
Ranking	#199 in MvnRepository (See Top Artifacts) #1 in Distributed Computing
Used By	2,097 artifacts
Scala Target	Scala 2.11 (View all targets)
Vulnerabilities	**Direct vulnerabilities:** CVE-2022-33891 CVE-2021-38296 **Vulnerabilities from dependencies:** CVE-2022-23305 CVE-2022-23302 CVE-2021-4104 View 17 more ...

Network Framework Apache 2.0		io.netty » netty-all **2 vulnerabilities**		4.0.23.Final	4.1.79.Final
Logging Apache 2.0		log4j » log4j **4 vulnerabilities**		1.2.17	2.18.0

==

Open Source Insights shows a list of the vulnerabilities associated with a library. The vulnerabilities may or may not come from other dependencies.

Security Advisories   **100**

In this package

Low severity vulnerability that affects org.apache.spark:spark-core_2.10 and org.apache.spark:spark-core_2.11
9.8 CRITICAL · GHSA-phg2-9c5g-m4q7   **MORE DETAILS**

High severity vulnerability that affects org.apache.spark:spark-core_2.10 and org.apache.spark:spark-core_2.11
7.8 HIGH · GHSA-8rhc-48pp-52gr   **MORE DETAILS**

Sensitive data written to disk unencrypted in Spark
7.5 HIGH · GHSA-fp5j-3fpf-mhj5   **MORE DETAILS**

Low severity vulnerability that affects org.apache.spark:spark-core_2.10 and org.apache.spark:spark-core_2.11
4.7 MEDIUM · GHSA-6mqq-8r44-vmjc   **MORE DETAILS**

Moderate severity vulnerability that affects org.apache.spark:spark-core_2.10 and org.apache.spark:spark-core_2.11
6.1 MEDIUM · GHSA-r34r-f84j-5x4x   **MORE DETAILS**

Moderate severity vulnerability that affects org.apache.spark:spark-core_2.10 and org.apache.spark:spark-core_2.11
4.2 MEDIUM · GHSA-w4r4-65mg-45x2   **MORE DETAILS**

==

MVN+ shows a list of dependencies and their associated vulnerabilities grouped by different dependency types (e.g., direct vs transitive).

Direct Compile Dependency (39) (Priority:CRITICAL) Directly used

Depth	Group / Artifact	Version	Vulnerabilities
1	log4j » log4j	1.2.17	CVE-2019-17571 (9.8) CVE-2022-23305 (9.8) CVE-2022-23302 (8.8) CVE-2021-4104 (7.5)
1	org.eclipse.jetty » jetty-server	8.1.14.v20131031	CVE-2017-7657 (9.8) CVE-2017-7656 (7.5) CVE-2015-2080 (7.5) CVE-2019-10241 (6.1) CVE-2019-10247 (5.3) CVE-2021-34428 (3.5)

Direct Test Dependency (10) (Priority:LOW) Only be used in test scope

Depth	Group / Artifact	Version	Vulnerabilities
1	junit » junit	4.10	CVE-2020-15250 (5.5)
1	asm » asm	3.3.1	
1	com.novocode » junit-interface	0.10	

* Note: By default, test dependencies *will not* be included in the final binary. Hence, MVN+ shows that vulnerabilities in test dependencies have a low priority.

Q3: On a scale of 1 (negative) to 5 (positive), how would you rate Maven repository's vulnerability listing approach?
Answer: 3
Feedback (optional):

Q4: On a scale of 1 (negative) to 5 (positive), how would you rate Open Source Insights' vulnerability listing approach?
Answer: 3
Feedback (optional):

Q5: On a scale of 1 (negative) to 5 (positive), how would you rate MVN+'s vulnerability listing approach?
Answer: 5
Feedback (optional):

Q6: On a scale of 1 (negative) to 5 (positive), how useful is listing every single vulnerability that is *from each dependency* helpful for developers (e.g., DependencyA has Vulnerability1 and Vulnerability2)?

Answer: 5
Feedback (optional):

Q7: On a scale of 1 (negative) to 5 (positive), how useful is grouping the dependencies/vulnerabilities based on dependency scope (e.g., direct dependency and test dependency)?

Answer: 4
Feedback (optional):

Part 3: Vulnerability Severity Score and Ranking

Maven Repository only shows the vulnerability from direct dependencies **(depth=1)** and does not show the vulnerability severity score. If you want to know the details of the vulnerabilities, you need to click the link and go to the homepage of the dependency library (e.g., go to **netty-all** to find the details) and search based on the CVE ID for the severity score by yourself.

Network Framework Apache 2.0		io.netty » netty-al 2 vulnerabilities	4.0.23.Final	4.1.79.Final
Logging Apache 2.0		log4j » log4j 4 vulnerabilities	1.2.17	2.18.0

Home » io.netty » netty-all » 4.0.23.Final

Netty/All In One » 4.0.23.Final

Netty is a NIO client server framework which enables quick and easy

License	Apache 2.0
Categories	Network App Frameworks
Tags	network socket framework netty io
Date	Aug 15, 2014
Files	pom (16 KB) jar (1.7 MB) View All
Repositories	Central BeDataDriven Redhat GA Sonatype
Ranking	#180 in MvnRepository (See Top Artifacts) #1 in Network App Frameworks
Used By	2,278 artifacts
	Direct vulnerabilities: CVE-2019-16869 CVE-2016-4970

===

Open Source Insights shows the vulnerability from direct dependencies **(depth=1)** and transitive **(depth>1)** dependencies. OSI does not show the dependency depth information. OSI ranks the vulnerabilities by severity score (as illustrated below by the red arrow).

Low severity vulnerability that affects org.apache.spark:spark-core_2.10 and org.apache.spark:spark-core_2.11 **9.8 CRITICAL** · GHSA-phg2-9c5g-m4q7	MORE DETAILS
High severity vulnerability that affects org.apache.spark:spark-core_2.10 and org.apache.spark:spark-core_2.11 **7.8 HIGH** · GHSA-hc-48pp-52gr	MORE DETAILS
Sensitive data written to disk unencrypted in Spark **7.5 HIGH** · GHSA-5j-3fpf-mhj5	MORE DETAILS
Low severity vulnerability that affects org.apache.spark:spark-core_2.10 and org.apache.spark:spark-core_2.11 **4.7 MEDIUM** · GHSA-6mqq-8r44-vmjc	MORE DETAILS
Moderate severity vulnerability that affects org.apache.spark:spark-core_2.10 and org.apache.spark:spark-core_2.11 **6.1 MEDIUM** · GHSA-r34r-f84j-5x4x	MORE DETAILS
Moderate severity vulnerability that affects org.apache.spark:spark-core_2.10 and org.apache.spark:spark-core_2.11 **4.2 MEDIUM** · GHSA-w4r4-65mg-45x2	MORE DETAILS

===

MVN+ shows the dependencies ranked by dependency depth. Then, MVN+ ranked the dependency based on the highest severity score.

Depth	Group / Artifact	Version	Vulnerabilities
3	net.sourceforge.htmlunit » htmlunit	2.14	CVE-2020-5529 (8.1)
3	org.apache.httpcomponents » httpclient	4.3.2	CVE-2014-3577 (5.8) CVE-2020-13956 (5.3) CVE-2015-5262 (4.3)
3	commons-io » commons-io	2.4	CVE-2021-29425 (4.8)
4	commons-collections » commons-collections	3.2.1	CVE-2015-6420 (7.5)
4	xerces » xercesImpl	2.11.0	CVE-2012-0881 (7.5) CVE-2022-23437 (6.5)
4	com.google.guava » guava	15.0	CVE-2018-10237 (5.9) CVE-2020-8908 (3.3)
5	org.eclipse.jetty » jetty-io	8.1.14.v20131031	CVE-2021-28165 (7.5)

Q8: On a scale of 1 (negative) to 5 (positive), how would you rate Maven repository's vulnerability listing approach?
Answer: 1
Feedback (optional):

Q9: On a scale of 1 (negative) to 5 (positive), how would you rate Open Source Insights' vulnerability ranking approach?
Answer:3
Feedback (optional):

Q10: On a scale of 1 (negative) to 5 (positive), how would you rate MVN+'s ranking approach?
Answer:5
Feedback (optional):

Q11: On a scale of 1 (negative) to 5 (positive), how is *depth* and *rank of dependency by depth*, helpful for developers?
(e.g.,
    (1) {Dependency A (depth=2) [CVE-1 (9), CVE-2 (5)]},
    (2) {Dependency B (depth=3) [CVE-3 (9.8)]},
    (3) {Dependency C (depth=3) [CVE-4 (8), CVE-5 (6)]}
)
Answer: 5
Feedback (optional):

Q12: On a scale of 1 (negative) to 5 (positive), how would you rate the *ranking of vulnerabilities by their severity score*? (e.g.,
    (1) CVE-3 (9.8)
    (2) CVE-1 (9)
    (3) CVE-4 (8)
    (4) CVE-5 (6)
    (5) CVE-2 (5)
)
Answer: 5
Feedback (optional):