

On the Effect of Transitivity and Granularity on Vulnerability Propagation in the Maven Ecosystem

坂本 廉也

ソフトウェアの再利用

| 現代のソフトウェア開発では当たり前

- プロジェクトで使用するライブラリ数は増加
- セキュリティ脆弱性は着実に増加^[2]

| ライブラリの再利用

- 開発工数を大幅に削減可能
- セキュリティ上の脅威をもたらす可能性^[3]

多くのソフトウェアが、脆弱性を持つ可能性の高いライブラリに依存

[2] Snyk, “State of Open Source Security Report,” 2022.

[3] N.H.Pham,T.T.Nguyen,H.A.Nguyen,andT.N.Nguyen,“Detection of recurring software vulnerabilities,” in *Proceedings of the IEEE/ACM international conference on Automated software engineering*, 2010.

背景

| 開発者が古い依存関係を保持

- ソフトウェアを脆弱にしている^[6]

| Why?

- 利用可能なアップデートに関する認識不足
 - 互換性の問題
- などが原因

| 攻撃者に脆弱性を突かれないためには. . .

- 素早く修正プログラムをリリース
- 脆弱性発見から修正パッチ作成までのエコシステムを理解

[6] R.G.Kula,D.M.German,A.Ouni,T.Ishio,andK.Inoue,“Dodevelopers update their library dependencies?”
Empirical Software Engineering, vol. 23, no. 1, pp. 384–417, 2018.

目的

| 関連研究

- セキュリティとその伝播による影響の分析^[9~12]
 - パッケージレベルの分析にとどまる
 - 実際に脆弱なコード部分を利用しているかどうかはわからない
- 依存関係を細かく分析^[13,14]
 - 計算コストがかかるため、少ないプロジェクトのみが対象

| 目的

- Mavenエコシステムにおける脆弱性の潜在的な影響に関する新たな洞察を提供

[9] A. Decan, T. Mens, and E. Constantinou, “On the impact of security vulnerabilities in the npm package dependency network,” in *Proceedings of the 15th international conference on mining software repositories*, 2018, pp. 181–191.

[10] M. Alfadel, D. E. Costa, and E. Shihab, “Empirical analysis of security vulnerabilities in python packages,” in *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2021, pp. 446–457.

[14] B.B.Nielsen, M.T.Torp, and A.Møller, “Modulargraphconstruction for security scanning of node.js applications,” in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2021, pp. 29–41.

Research Question

| RQ1

- Mavenにおいて脆弱性はどのように分布しているか

| RQ2

- パッケージレベルとメソッドレベルの分析で、脆弱性の伝播はどのように異なるか

| RQ3

- ルートパッケージへの脆弱性の伝播はどのようにされるか

| RQ4

- 全ての推移的依存関係を考慮する必要があるか

データセット

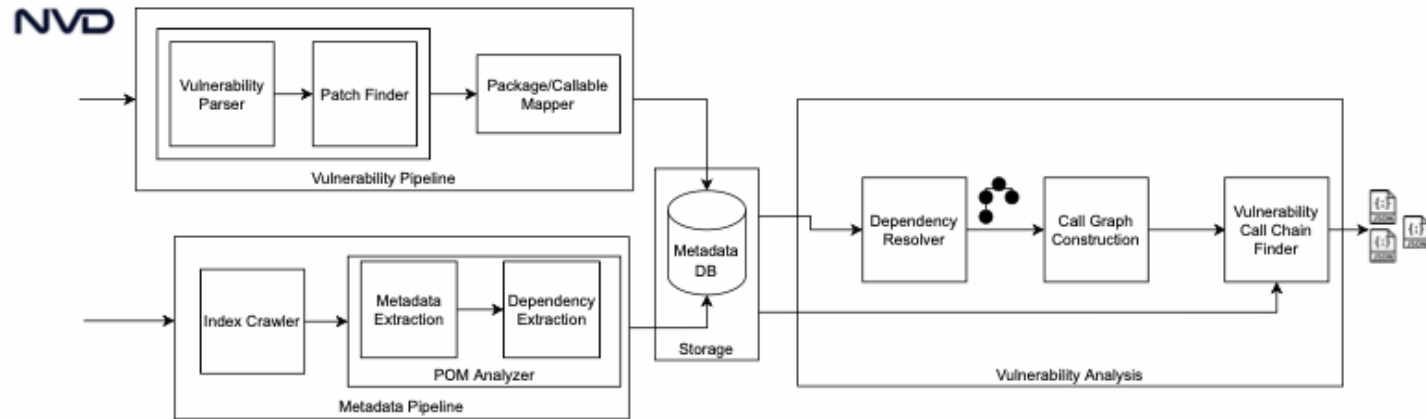
Source	License	Updates
National Vuln. Database (NVD)	Public Domain	2 hours
GitHub Advisories	Public Domain	Daily
project-kb (by SAP)	Apache License 2.0	n/a
oss-fuzz-vulns (by Google)	CC-BY-4.0	Daily

単一フォーマット
へ変更

JSON Field	Description
ID	A unique id (E.g. CVE-2018-9159)
Purls	Universal URLs that represent vulnerable packages [23]
CPE	A structured naming scheme that represents information technology systems, software, and packages [24]
CVSS score	A numeric value for showing the severity of software vulnerabilities from 0.0 to 10.0 [25]
CWE	A list of software weakness types [26]
Severity level	Qualitative severity rating scale based on CVSS scores [25]
Published date	The date that a vulnerability is publicly announced
Last modified date	The date that a vulnerability's metadata is updated
Description	An English description of what software systems are affected by a vulnerability
References	Extra web links that provide more information about a vulnerability
Patch	Links to patch information and commits
Exploits	Links to how to exploit a vulnerability

| 1,306件のセキュリティレポート

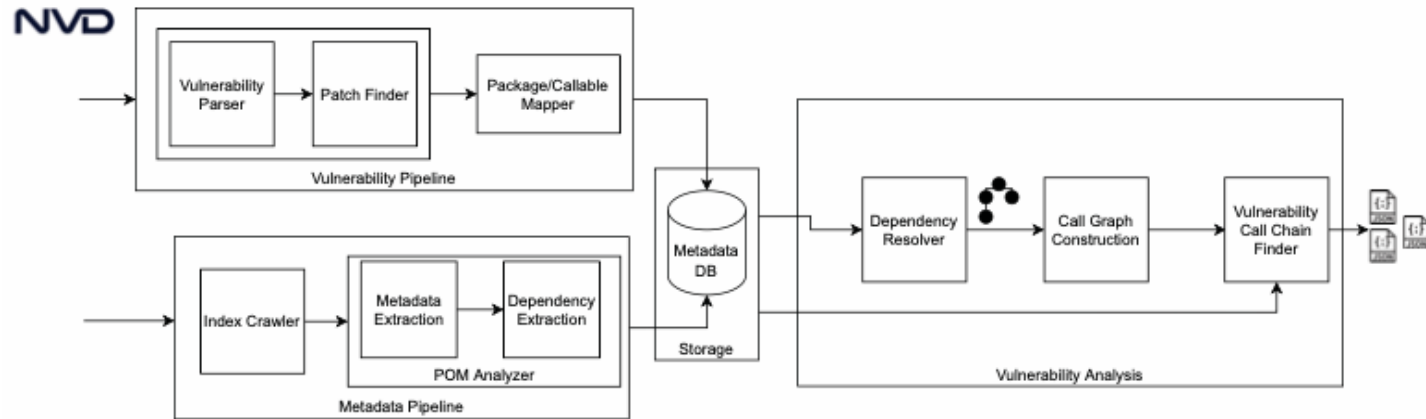
脆弱なメソッドの特定



脆弱性修正で変更されたメソッドを特定

- GitHub, GitLab, BitBucketから取得する場合
 - PR：マージするコミットを特定し，メソッドを取得
 - Issue：リンク先のPRなどから探す
- Bugzilla, Jira
 - 添付ファイルやIssuenoコメント内の参照先を探す

脆弱なパッケージの特定



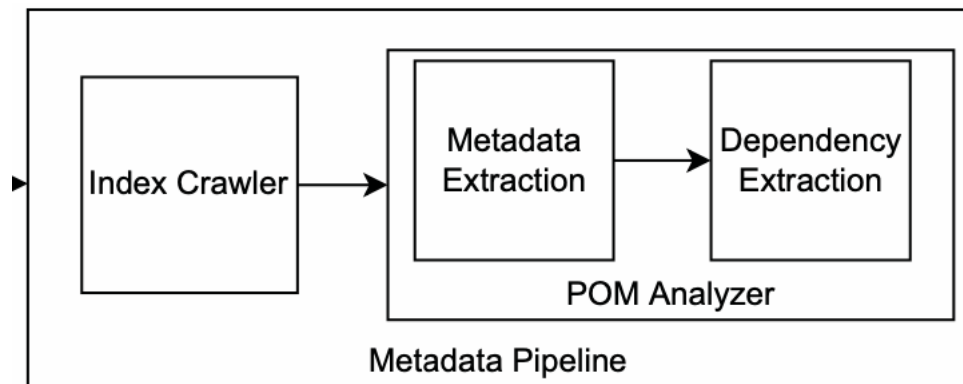
脆弱性パイプライン

- 脆弱性の影響を受けるプロジェクトのリリースを特定
 - Maven Central上のプロジェクトとそのバージョンを収集
 - セキュリティレポートから脆弱性制約を抽出
 - 影響を受けるプロジェクトのバージョンを取得
 - 脆弱性が含まれるファイルを特定

メタデータの収集

| メタデータパイプライン

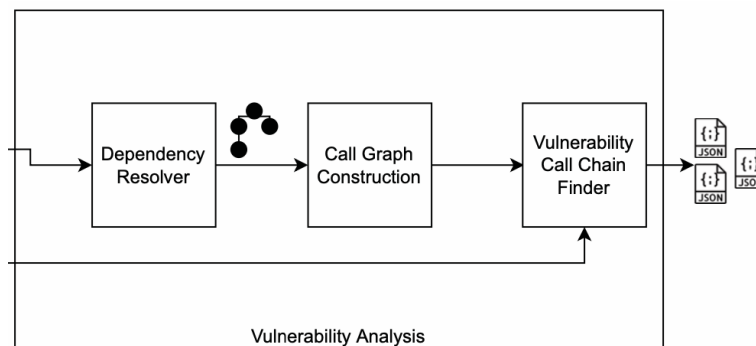
- Maven Centralからバージョン管理されたパッケージを収集
 - 2021年9月～2022年9月にリリースされたパッケージ
- POM Analyzer
 - pom.xmlで定義されている依存関係のリストなどのメタデータを抽出



依存関係の構築

| Analyzerパイプライン

- SHRINKWRAPを用いてプログラム全体のコールグラフを生成
 - メソッドレベル
- パッケージとその推移的依存関係にあるプログラムに対して幅優先探索を実行
 - グラフを走査し、パッケージのノードから脆弱なノード（メソッド）までの最短経路を計算
 - 脆弱なメソッドと脆弱性をリストアップ



RQ2 | 粒度別で含まれる脆弱性の数の違い

| 方法

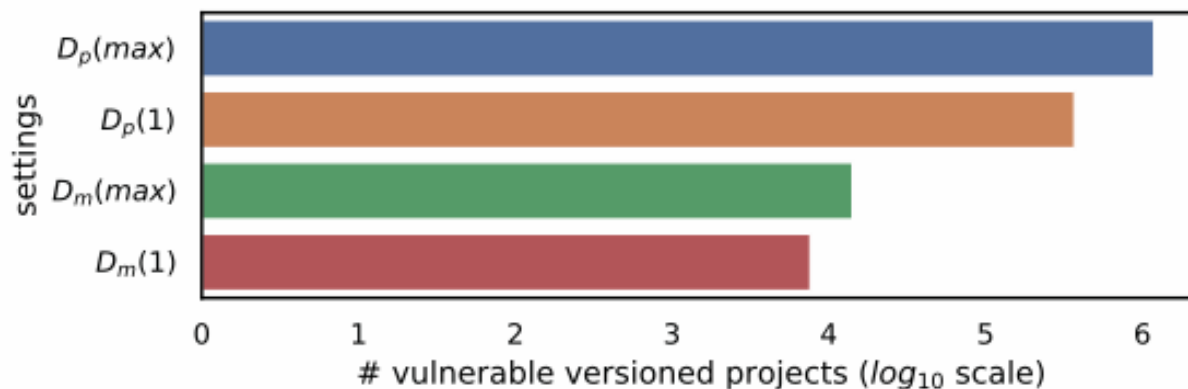
- $Dp(max)$: 全ての推移的な依存関係を含むパッケージレベルの分析
- $Dp(1)$: 直接的な依存関係のみを対象としたパッケージレベルの分析
- $Dm(max)$: 全ての推移的な依存関係を含むメソッドレベルの分析
 - 実際に影響を受けるバージョンが管理されたパッケージの数
- $Dm(1)$: 直接的に依存関係のみのメソッドレベルの分析

脆弱性の影響を受ける可能性のあるバージョン管理されたパッケージの数を調査

RQ2 | 粒度別で含まれる脆弱性の数の違い

結果

- メソッドレベルはパッケージレベルに比べて脆弱なパッケージの検出数が非常に少ない
- $D_p(\max)$ 分析： 10^6 個のパッケージが脆弱性の影響を受ける
- $D_m(1)$ 分析：369K(36.9×10^4)個



RQ3

| パッケージの推移的な依存関係を反復処理

- メソッドレベルにおいてパッケージに影響を与える脆弱性上位10個
- パッケージレベルでは上位10個が7886,921件のパッケージに影響
- メソッドレベルでは4,485件だけ

CVE ID	Project	Number of Packages		%Proportion ¹	
		Potentially Affected	Actually affected	$D_p(max)$	$D_m(max)$
CVE-2020-36518	com.fasterxml.jackson.core:jackson-databind	233,430	1,153	19.6	8.1
CVE-2022-24823	io.netty:netty-codec-http	142,177	90	11.9	0.6
CVE-2022-24329	org.jetbrains.kotlin:kotlin-stdlib	82,060	32	6.9	0.2
CVE-2021-37137	io.netty:netty-codec	57,535	525	4.8	3.7
CVE-2021-22569	com.google.protobuf:protobuf-kotlin	57,095	390	4.8	2.7
CVE-2018-1000632	dom4j:dom4j	47,820	1,438	4.0	10.1
CVE-2022-25647	com.google.code.gson:gson	47,372	171	4.0	1.2
CVE-2020-8908	com.google.guava:guava	42,084	84	3.5	0.6
CVE-2022-22965	org.springframework:spring-webflux	38,882	572	3.3	4.0
CVE-2018-20200	com.squareup.okhttp:okhttp	38,466	30	3.2	0.2

まとめ

- | 既存研究で言われていることとは異なり，実際にはあまり脆弱性の影響を受けない
- | パッケージ/メソッドレベルに関係なく，推移的依存関係によって脆弱性の影響を受ける
 - － 直接の依存関係だけでなく推移的依存関係を頻繁に監視することが重要