

A Large Scale Analysis of Semantic Versioning in NPM

Donald Pinckney
Northeastern University
Boston, USA
pinckney.d@northeastern.edu

Federico Cassano
Northeastern University
Boston, USA
cassano.f@northeastern.edu

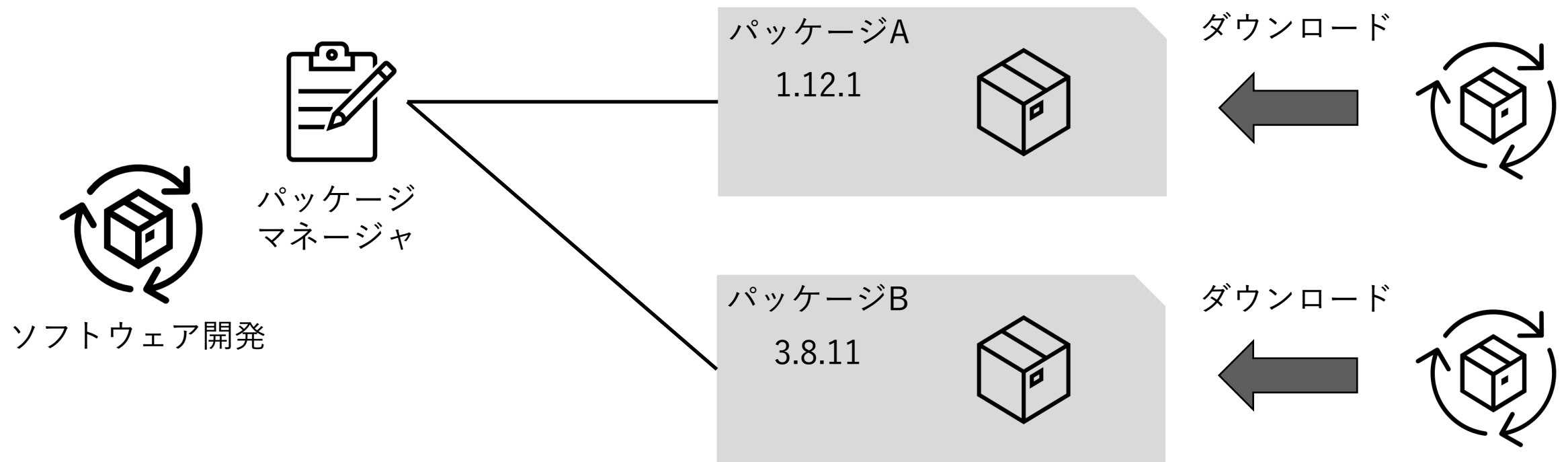
Arjun Guha
Northeastern and Roblox
Boston, USA
a.guha@northeastern.edu

Jonathan Bell
Northeastern University
Boston, USA
j.bell@northeastern.edu

Proceedings of 20th International Conference on Mining Software Repositories (MSR '23),
pp. ?? - ??

背景 | パッケージ管理

モダンソフトウェアは多くのオープンソースパッケージを活用し開発される



パッケージマネージャを利用することで、依存パッケージを柔軟に管理することが可能

- JavaScriptのNPM や PythonのPIP など

背景 | Semantic Versioning

下流の開発者がパッケージ更新による影響を容易に理解するためのバージョンングルール



- **Major** : 互換性がない破壊的なAPIの変更があった場合に更新
- **Minor** : 新しい機能の追加などがあった場合に更新
- **Bug** : バグ修正があった場合に更新

「バグ修正による更新は自動で行う」や「メジャーバージョンの更新はしばらく待つ」など判断可能

パッケージ更新における技術的遅延

原因 1

- パッケージ開発者のバージョンナンバリングが正確であることが保証されていないこと

原因 2

- 複数にまたがるパッケージの依存関係である場合、更新が伝播されるまでに時間がかかること

原因 3

- バグ修正の自動更新はマルウェアのセキュリティ攻撃が紛れることがあり危険性を孕んでいること

研究目的 & Research Question

目的

- Semantic Versioningや依存関係などの利用や、これらの要因が与える開発者の経験やセキュリティへの影響を理解する
 - NPMパッケージマネージャに関連するものを対象

Research Question

- RQ1：バージョン制約の利用
- RQ2：Semantic Versioningとリリース内容
 - パッチ → Bug
 - 脆弱性の混入 → Minor or Major
- RQ3：パッケージが含む依存関係の古さや更新の伝播
- RQ4：更新時のコード変更
 - Major > Minor > Bug の順にコードの変更が伴う確率が高い

データセット

2020年10月31日時点におけるNPMに含まれる全パッケージと全バージョンを対象

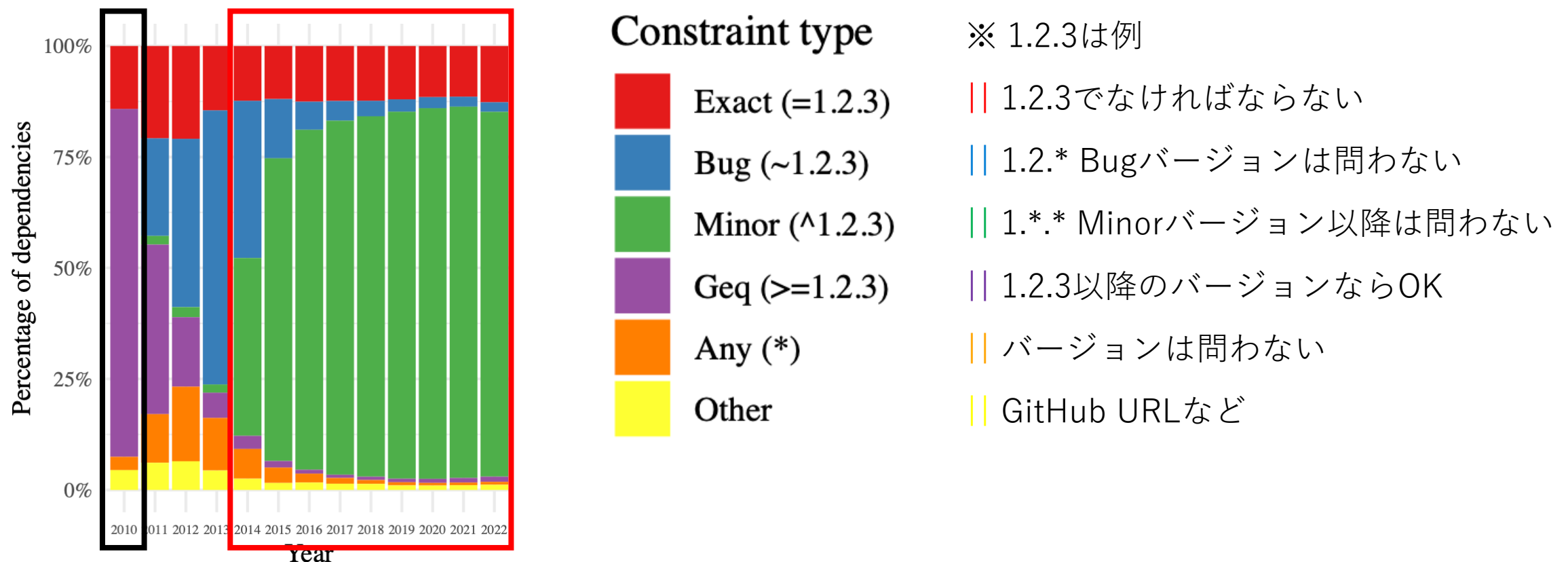
- 2,663,681パッケージ
- 28,941,927バージョン

セキュリティ情報としてGitHub Security AdvisoryからNPMに関するセキュリティデータを取得

RQ1 | バージョン制約の利用

近年の開発では**Minor記述が最も利用されている**

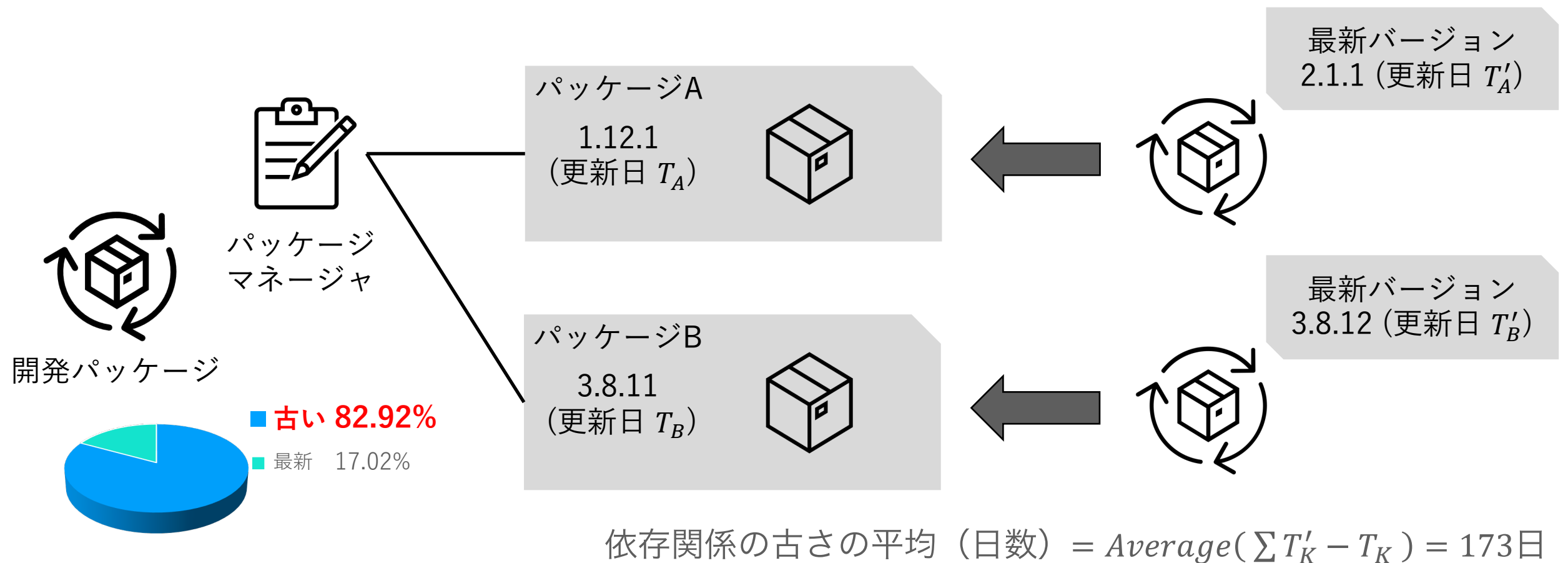
- 2010年：Geq記述が最も利用されている（78.36%）
 - 現在は後方互換性の問題から利用がほとんどされない
- **2014年以降**：Minor記述が大部分を占める
 - Minor更新は実質的にバグ修正を含む



RQ3| 依存関係の古さや更新の流れ

一つでも古い依存関係を持つパッケージは約83%

依存関係の古さは平均して173日



RQ3 | 依存関係の古さや更新の流れ

開発者はMinorやBug更新にほとんど関与せず，逐次更新を試みる（約90%）

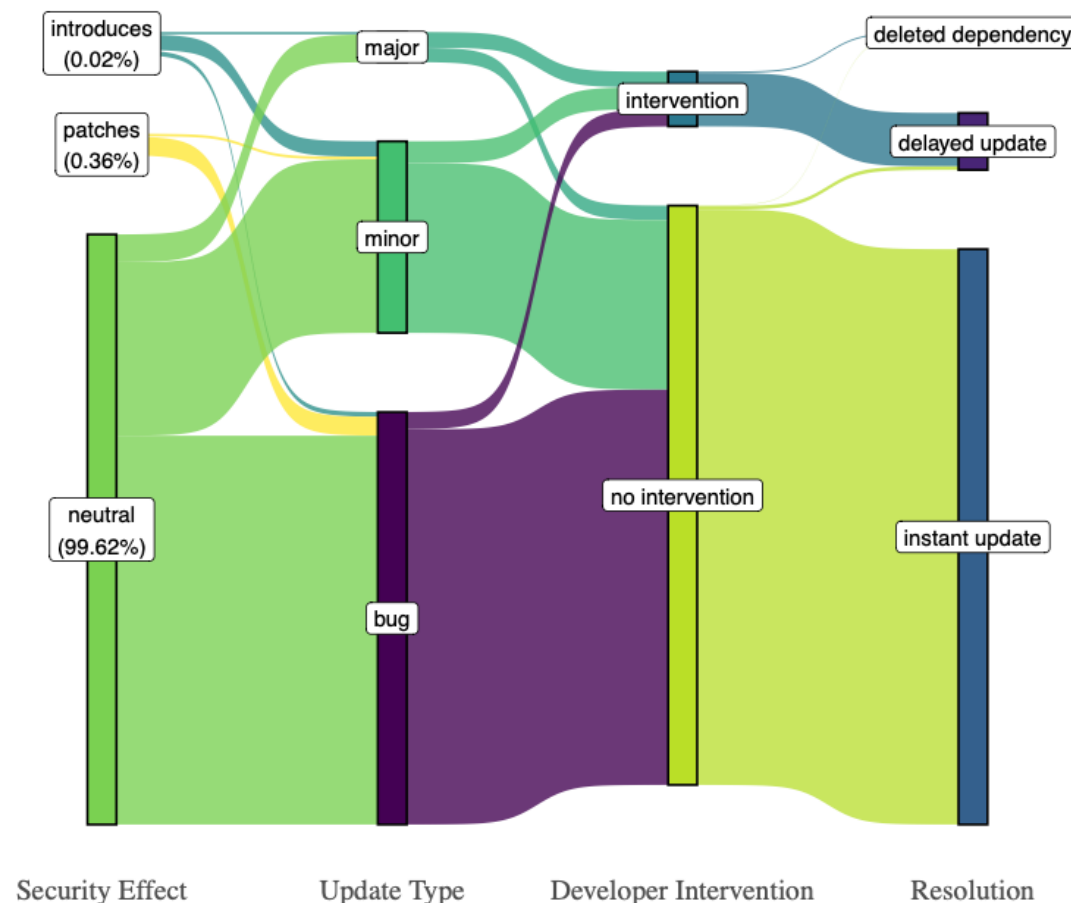
no intervention

instant update

更新が遅延する場合は開発者が意図的に遅らせている（約9%）

intervention

delayed update



議論 | 開発者とメンテナに向けて

開発者

- BugとMinorのリリースに違いがあるが、開発者は区別せずに自動更新する傾向にある
- しかしMinorの方が後方互換性を破壊する変更が含まれる可能性があるため、**自動更新はBug（“~”の利用）にとどめる方がいい**可能性がある
 - 実際にMinorの方が脆弱性を含む可能性も高い（RQ2）

メンテナ

- 複数のパッケージ推移型依存により更新の伝播を滞らせる可能性がある
- 自身のプロジェクトにとって重要なパッケージを理解し、協力をすることが重要である

議論 | 研究者に向けて

研究者

- 開発者は基本的にSemantic Versioningを遵守しようとする傾向にはあるが必ずしも守られているわけではない
- そのため、更新が行われた場合に、何が変更されたのかを理解するための仕組みが必要となる
 - マルウェアの混入防止のために
- また、更新時にコード以外の変更がそれなりに行われることがわかった（RQ4）ため、研究者はコード以外のものも対象に検知できるよう取り組むべきである

所感

後方互換性の厳密な定義を行うことが重要

- APIの引数や返り値の型の変化や，APIそのものの削除以外に後方互換性に関する現象はどのようなものか？
- JavaScriptは動的型付け言語であることから，型推論技術などの発展がより必要となる？

更新時にコード以外の変更がそれなりにある

- .jsonや.yamlなどのファイルであれば，出どころがわからないURLなどの記述を検知すればできそう
- バイナリファイルはどのように調べるのだろうか？

