Debian パッケージ間の依存関係を表す SPDX ドキュメント自動生成ツールの開発

田邉　傑士[†a)]　神田　哲也[†]
眞鍋　雄貴[††]（正員）　井上　克郎[†††]（正員：フェロー）
肥後　芳樹[†]（正員）

Development of an Automatic SPDX Document Generation Tool to Show Dependencies between Debian Packages

Taketo TANABE[†a)], Tetsuya KANDA[†], *Nonmembers*, Yuki MANABE[††], *Member*, Katsuro INOUE[†††], *Fellow*, and Yoshiki HIGO[†], *Member*

[†] 大阪大学大学院情報科学研究科，吹田市
Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565–0871 Japan
[†††] 南山大学理工学部，名古屋市
Faculty of Science and Technology, Nanzan University, Nagoya-shi, 466–8673 Japan
[††] 福知山公立大学情報学部，福知山市
Faculty of Informatics, The University of Fukuchiyama, Fukuchiyama-shi, 3370 Japan
a) E-mail: tk-tanab@ist.osaka-u.ac.jp
DOI:10.14923/transinfj.2023JDL8004

あらまし　Debian パッケージを対象にした推移的なパッケージ間の依存関係を表す SPDX (Software Package Data Exchange) ドキュメント自動生成ツール debiantospdx を開発した．ツールの背景と実証について述べる．

キーワード　SBOM，SPDX，依存関係，ソフトウェアライセンス，脆弱性

1. まえがき

ソフトウェア開発にサードパーティ製のソフトウェアが利用される機会は多い．Contrast Security 社によると，今日のソフトウェアの大部分 (79%) にサードパーティ製のライブラリが使用されている [1]．サードパーティ製のソフトウェアの利用は開発にかかるコストを削減できる一方で，サードパーティ製のソフトウェアとの依存関係がソフトウェアライセンス（以降単にライセンス）や脆弱性などの問題を引き起こすことがある．これらの問題を受けて SBOM (SPDX) を利用する動きが広まりつつある．

SBOM とは「Software Bill of Materials」の略称であり，対象とするソフトウェアの機械可読なメタデータのことを指す．ここでいうメタデータとは，ソフトウェアのコンポーネントとその依存関係，及びライセンスデータなどであり，SBOM にはそれらが一意に識別できる形式でまとめられる [2]．2021 年 5 月に署名された「Executive Order on Improving the Nation's Cybersecurity」という大統領令で SBOM について言及されて以降，様々な組織で活用が検討されている [3]．また SPDX はソフトウェアパッケージ（以降単にパッケージ）の SBOM を実現する形式の一つであり，ISO/IEC JTC1 標準となった [4]．SPDX 形式で書かれた SBOM を SPDX ドキュメントと呼ぶ．

SPDX を活用する上では SPDX ドキュメントを自動生成するツールが必要である．このようなツールは SPDX ドキュメント生成における人為的なミスを防ぐことができる上に，SPDX 活用にかかるコストを大幅に減らしたという実証もある [3]．

そこで本研究では Debian パッケージに対して推移的な依存関係を解析して，SPDX ドキュメントを自動生成するツール debiantospdx を作成した．Debian パッケージは現在 Linux ディストリビューションのシェアで 1，2 位を占める Ubuntu と Debian で使われているパッケージである(注1)．しかしながら我々の知る限り，提案ツールと同等の機能を有する無償ツールはまだない．またこのツールの効果を調べるため，既存のコマンドを実行して推移的な依存関係を調査する場合や簡易ツールで推移的な依存関係を調査する場合と比較した．その結果，提案ツールが依存関係特定のための工数を大幅に削減すること，及び，他の調査方法で多く出現する不要な依存関係を削除できることを確認した．

2. 提案ツール

提案ツールはインストール済みの全ての Debian パッケージをパッケージ間の依存関係を辿りながら推移的に解析し，SPDX ドキュメントを生成するコマンドラインツールである．ツールは Python で記述されており，PyPI で公開している(注2)．

提案ツールは図 1 に示す五つのステップから構成される．まずステップ 1 として，Debian パッケージのパッケージ名から Debian パッケージのインストールを制御する control ファイルに記載されている情報を取得し，SPDX ドキュメントに必要な情報を抽出する．

次にステップ 2 としてパッケージによってインストールされたファイルのリストを取得した後，ライセンスやコピーライトが記載された copyright ファイルを解析して情報を抽出し，パッケージを構成する全てのファイルのハッシュ値を計算する．この copyright ファイルの解析には ScanCode Toolkit という既存のツールを使った．

(注1)：https://w3techs.com/technologies/history_details/os-linux
(注2)：https://pypi.org/project/debiantospdx/

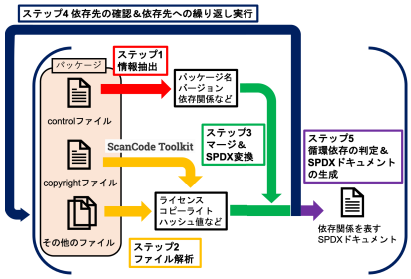

図 1　処理の概要

そしてステップ 3 としてステップ 1 とステップ 2 から得られた情報をマージして，依存関係に関する情報以外の情報を SPDX の形式に変換する．

更にステップ 4 として依存関係を処理する．control ファイルには依存先として可能性がある全てのパッケージを記載しており，実際にインストールされていないものを含む．そこで依存先候補のパッケージが実際に存在するか，存在する場合はバージョンの制約を満たしているか，依存関係を満たすパッケージがない場合は依存関係に指定されているパッケージの機能を提供する別のパッケージがないかを確認し，依存関係を満たしているパッケージが見つかった場合には現在解析中のパッケージに対する SPDX ドキュメントは生成せず，依存先のパッケージに対してステップ 1 からステップ 5 までを繰り返し実行する．

最後にステップ 5 として依存先の全てのパッケージの解析が終わると現在解析中のパッケージが他のパッケージと循環依存していないかを検証する．本ツールでは基本的に一つのパッケージにつき一つの SPDX ドキュメントを生成する．その際，依存先のパッケージは SPDX ドキュメントのハッシュ値を用いて参照する．しかし，その依存先を示すハッシュ値も SPDX ドキュメントの一部であるため，パッケージが循環依存している場合は双方の SPDX ドキュメントのハッシュ値を同時に計算することはできない．よって，循環依存している場合は循環依存している全てのパッケージの解析を終えてから循環依存に含まれるパッケージの情報をまとめて一つの SPDX ドキュメントとして出力する．

3.　実証・評価

実証・評価は Windows10 の WSL2 に新しくインストールした Ubuntu22.04 上で行った．システム内のインストール済みパッケージの数は 517 個である．

3.1　既存コマンドによる依存関係取得との工数比較

1 組のパッケージ (A,B) に対して A が B の推移的な依存先にあるかを apt や dpkg などの既存コマンドを実行して調べる場合に必要となるコマンドの実行回数をインストール済みパッケージの全ての組み合わせに対して調べた．

調査の結果，平均 100.6 回のコマンドの実行が必要であることが分かった．この原因は依存先にないことを確かめるためには依存関係を最後まで追う必要があるからだと考えられる．一方，提案ツールを用いれば 1 回のコマンド実行で済むため，工数削減となる．

3.2　簡易ツールとの比較

提案ツールはインストール済みパッケージとの推移的な依存関係を取得する．そこで control ファイルに記載されている依存先になる可能性がある全てのパッケージを表示するコマンド (apt depends) を繰り返すことで推移的な依存関係を取得する簡易ツールを作成し，提案ツールが生成する SPDX ファイルから取得できる推移的な依存関係との違いを調べた．

調査の結果，提案ツールからは 1 パッケージ当り平均 250.6 個，簡易ツールからは平均 9906.4 個の推移的な依存関係が検出された．よって提案ツールを用いれば不要な依存先の検査を減らすことができる．

4.　む　す　び

Debian パッケージに対して推移的な依存関係を解析して，SPDX ドキュメントを生成するツール debiantospdx を作成し，簡単な評価を実施した．この評価は比較的断片的であったため，今後は実際のユースケースに応じた評価を実施していきたい．

謝辞　本研究はJSPS科研費(JP19K20239,JP21K02862, JP23H03375)，及び 2023 年度南山大学パッヘ研究奨励金 I-A-2 の助成を得て行われた．

文　　献

[1] J. Williams and A. Dabirsiaghi, “The unfortunate reality of insecure libraries,” https://cdn2.hubspot.net/hub/203759/file-1100864196-pdf/docs/contrast/_-/_insecure/_libraries/_2014.pdf, March 2012.

[2] Linux Foundation and its Contributors, “A common software package data exchange format,” 2.0 edition, 2015.

[3] 経済産業省商務情報政策局サイバーセキュリティ課, “サイバー・フィジカル・セキュリティ確保に向けたソフトウェア管理手法等検討タスクフォースの検討の方向性,” https://www.meti.go.jp/shingikai/mono_info_service/sangyo_cyber/wg_seido/wg_bunyaodan/software/pdf/006_03_00.pdf

[4] ISO, “ISO/IEC 5962:2021 - Information technology – SPDX Specification V2.2.1,” https://www.iso.org/standard/81870.html, Aug. 2021.

（2023 年 3 月 16 日受付，5 月 11 日早期公開）