CONFIDENTIAL

日本におけるソフトウェアサプライチェーンと SBOM のこれから
SBOM の継続的な運用フローとリスク管理手法

2023/08/25

脆弱性管理クラウド yamory プロダクト責任者
Software ISAC OSS委員会 副委員長
鈴木康弘

yamory

鈴木 康弘 @yappy727

株式会社アシュアード
脆弱性管理クラウド yamory プロダクト責任者
Software ISAC OSS委員会 副委員長

【略歴】

- 2010年、ITエンジニアとして株式会社ビズリーチに入社
- 5つの事業立ち上げ、プロダクト開発組織の組成に携わる
- 2018年 脆弱性管理クラウドyamoryを起案し、その後、事業化
- 現在は「脆弱性管理クラウドyamory」のプロダクト責任者
- 2023年 Software ISAC OSS委員会 副委員長

個別での対応が必要だったITシステムの脆弱性対策とリスク管理を
オールインワンで実現する、日本唯一のクラウドサービスです

1

ソフトウェア・クラウドの
利用／設定状況を自動で可視化

- **アプリライブラリ**（開発言語、SBOM）
- **ホスト／コンテナイメージ**
- **クラウド**（AWS, Microsoft Azure, GCP）
- **IT資産**（ネットワーク機器、商用MWなど）

2

独自の脆弱性データベースや
チェックルールと自動照合

3

複数プロダクト／レイヤーの
リスクを危険度別に一元管理

- **ソフトウェア脆弱性管理**
- **OSSライセンスの違反リスク**
- **クラウド設定不備（CSPM）**
- **EOL管理**

ご利用企業様

IT・Web系だけでなく、大手企業や非IT企業の導入も加速

一部抜粋／2023年5月現在

経産省ガイドラインに掲載（国産のツールでは唯一）

yamoryを「SBOMの作成や運用・管理に資する代表的なツール」として紹介

P.71

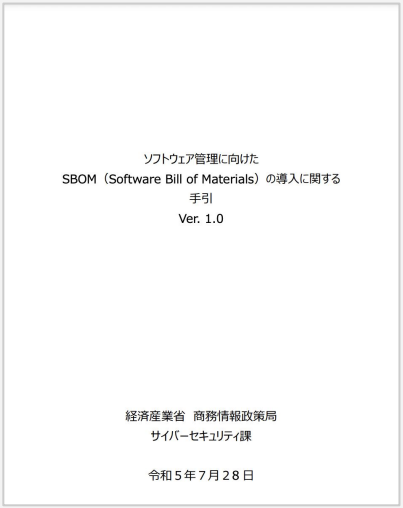
7.3.2. SBOM に関するツール

本節では、SBOM の作成や運用・管理に資する代表的な SBOM ツールを紹介する。有償の SBOM ツールだけでなく、無償の SBOM ツールも公開されており、ツールそれぞれに特徴がある。SBOM を導入する組織は、自組織の SBOM 導入の目的や SBOM 適用範囲を踏まえて、適切な SBOM ツールを選定することが望まれる。

(1) 有償ツール

※ アルファベット順

No.	名称	開発者	特徴
			・ コードマッチング、コンテナ解析、バイナリ解析などの複数のスキャンアプローチにより、正確で効率的な解析が可
			・ 脆弱性等に対する実用的な修正バージョンを提示
6	yamory	株式会社アシュアード	・ 対象となるITシステムで利用されるソフトウェアの脆弱性を検知・管理可能 ・ オートトリアージ機能で脆弱性の優先度を自動で判断 ・ 日次で脆弱性データベースを更新し、緊急性が高い脆弱性を早期に検知可能 ・ OSS のライセンス違反リスクを可視化



ソフトウェア管理に向けた
SBOM（Software Bill of Materials）の導入に関する
手引
Ver. 1.0

経済産業省 商務情報政策局
サイバーセキュリティ課

令和５年７月２８日

経済産業省 商務情報政策局 サイバーセキュリティ課
SBOM（Software Bill of Materials）の導入に関する手引
hhttps://www.meti.go.jp/press/2023/07/20230728004/20230728004.html

アジェンダ

1. SBOMの必要性
2. SBOM活用段階と運用例
3. SBOMの課題

SBOMの必要性

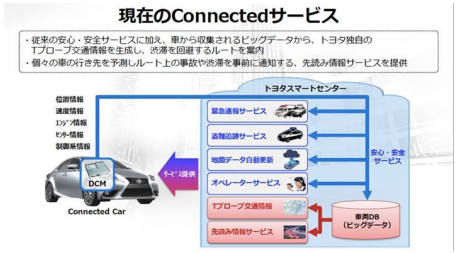
SBOMはソフトウェアの構成表

製品、ソフトウェア、ITシステムの内部のソフトウェア部品を把握するための仕組み

製品やソフトウェア、ITシステムの
リスク管理に活用する

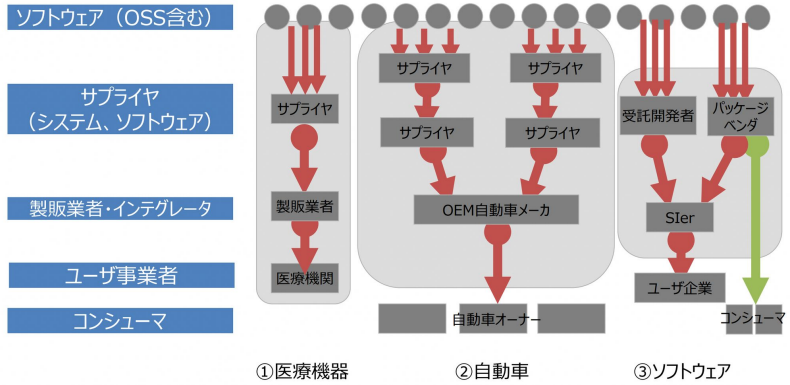
インターネットサービス

コネクテッドサービス



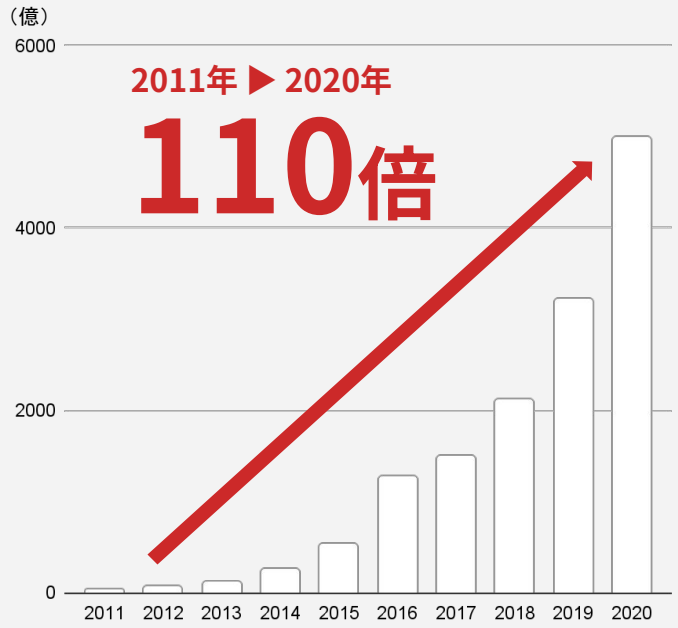
サプライチェーンの中で、
ソフトウェア部品情報を共有する

サプライチェーンの中で、さまざまなソフトウェアが作成され、最終製品の中に多く組み込まれている。



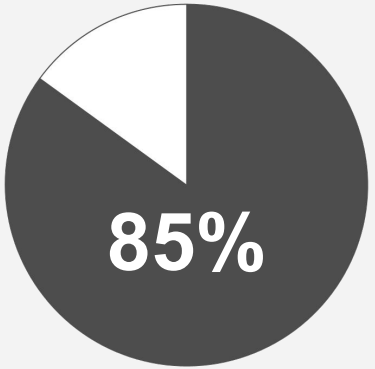
SBOMを管理できない場合のリスク

近年急増するサイバー攻撃



年間総観測パケット数
（NICTER観測レポート2020）

成功した攻撃のうち
既知の脆弱性を突いたものの割合



> **正しくパッチを当てていれば、攻撃のほとんどは防げたのです**
>
> ブライアン・サーティン
> （ベライゾンインシデント対応部門マネージャー）

参考：https://www.keyman.or.jp/kn/articles/1607/12/news170.html

SBOMを管理できない場合のリスク

ソフトウェアコンポーネントのリスク管理ができなければインシデントのリスクが高まる

脆弱性→ランサムウェア侵入、マルウェア（マリシャスパッケージ）

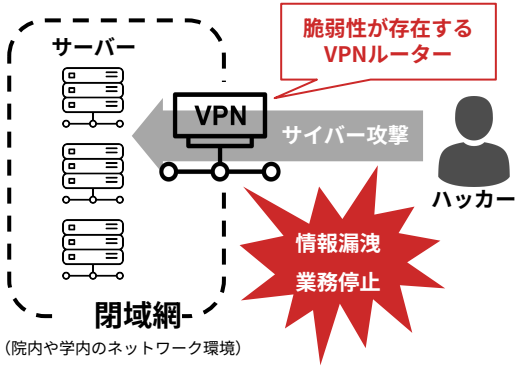
ランサムウェアなどへの感染や情報漏えいの危険性があるLog4Shell

オープンソースプロダクトとして公開されているソフトウェアである「Apache Log4j」において、2021年12月、任意のコードをリモートで実行できるゼロデイ脆弱性が発見されました。（中略）
*この深刻な脆弱性が悪用されると、ランサムウェアへの感染や個人情報の漏えいなどが起きる可能性があるうえ、**Log4Shellが残されたサーバーを踏み台に、社内の別のサーバーなどへ不正アクセス**するラテラルムーブメント（侵入拡大）も十分に考えられるため、迅速な対処が必要です。*

（X Managedコラムより抜粋）

事例 四国、大阪の医療機関

2021年、2022年、ネットワーク機器（VPN）の脆弱性を悪用され、そこを起因とし院内ネットワークへ侵入。閉域網内サーバー等についても適切に管理されていなかったため、各端末へランサムウェアが感染し、データを暗号化される。

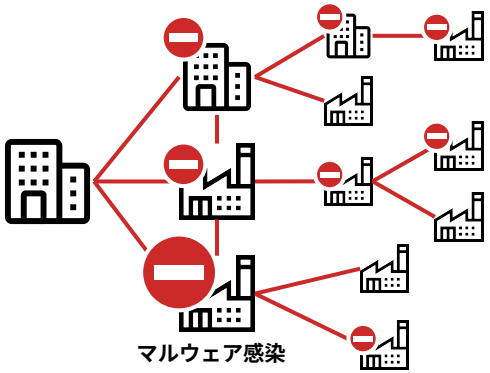


（院内や学内のネットワーク環境）

➡ **復旧まで 約2ヶ月**

事例 自動車製造サプライチェーン

2022年、自動車メーカー主要サプライヤーの1社がマルウエアによる感染被害を受け、**影響範囲の特定のため、社内サーバーを一旦全て停止**。自動車メーカーおよびグループ企業が一部生産を見合わせた。



➡ **14工場の28ラインが停止**

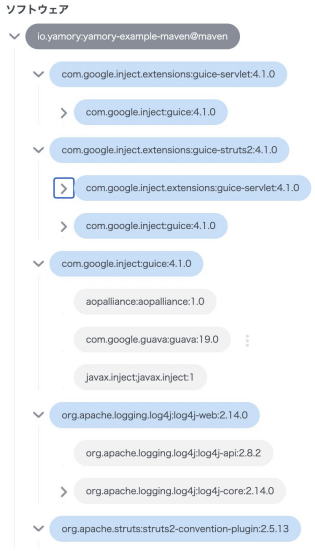
ソフトウェア構成管理の難しさ

ソフトウェアやシステムの構成が複雑になりすぎて正確に管理するのが難しい

一つのソフトウェアや製品やITシステムで数万-十万のコンポーネントに依存している

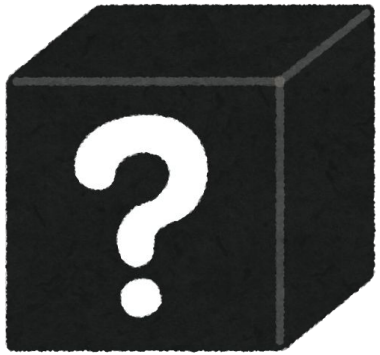
OSSの普及とソフトウェア依存構造

一つのソフトウェアは、数百から数千のソフトウェアコンポーネントに依存。



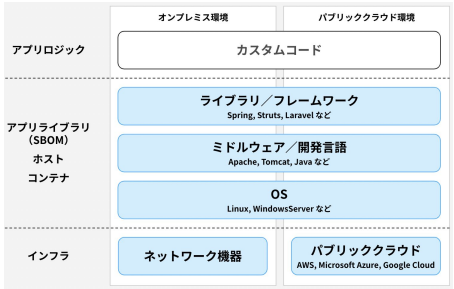
納品されたソフトウェアやシステムの中身はブラックボックスだった

サプライチェーンの中で、さまざまなソフトウェアが作成され、最終製品の中に多く組み込まれているが、中身はブラックボックスとなっている。



システムレイヤやコネクテッドの複雑性

最終製品は、複数のシステムレイヤや、さまざまなIoT（コネクテッド）で動く。



SBOMを利用することで、

複雑になりすぎたソフトウェア製品やITサービス内部のコンポーネントを

正確に把握しリスク管理を行えるようになる！

のではないか？と、期待されている

ガイドラインの制定や改訂

アメリカ政府を中心に、政府や各業界団体が出すガイドラインによって、
重要産業・業界からSBOMの運用が開始されている

	概要	動向
SBOM （Software Bill of Material）	• ソフトウェア部品表 • ソフトウェアサプライチェーンのなかで利用されているソフトウェア部品を正確に把握するための手法	• **米国大統領令**にてSBOMに関連するガイドラインが整備され、国内でも**経済産業省**を中心に**ガイドライン制定**に向けた動きがスタート
PCI-DSS v4.0	• クレジットカード業界におけるグローバルなセキュリティ基準 • カード情報の漏えいを防止するためにPCI SSC（大手ブランドによって設立）によって策定された	• システムコンポーネントの構成情報管理や脆弱性管理に関する要件が**新要件（6.3.2）に含まれている** • **2025年3月31日まで**にv4.0で新しく追加された要件に対応しなければならない
IMDRF **サイバーセキュリティガイダンス**	IMDRF（International Medical Device Regulators Forum）は国際医療機器規制当局フォーラム	• 医療機器のサイバーセキュリティに関する国際整合を図るため、一般原則とベストプラクティスを提供することを目的とした**ガイダンスが公開される** • メーカーと医療機関でのやりとりにおいてSBOM活用や、脆弱性の対応管理が**強く求められる**

SBOM活用段階と運用例

ガイドラインや法令遵守のため、まず、SBOMを作り、出せる状態にする企業が多い

同時に、継続的なSBOM運用やリスク管理運用の仕組み化が重要

1. **とりあえず手動でSBOMを作成した**

2. **SBOMを継続的に生成できる**
   a. OSSツールを組み合わせ、開発/運用のプロセスに組み込むことで可能
   b. 正確なソフトウェアコンポーネントの把握が困難なケースもあり、手動が残ってしまうケースもある

3. **SBOMからソフトウェア資産DBを作り、リスクDBとの自動照合とリスク管理ができる**
   a. 有償ツールを利用し、開発/運用のプロセスに組み込むことで可能

4. **取引先からSBOMを受け取り、ソフトウェア資産DBに取り込み、リスク管理できる**
   a. 有償ツールで対応可能
   b. ただ、有償ツールの中でも限られたツールのみが対応している

活用段階1：手動でSBOMを作成した

メリット：ガイドラインや法令遵守のため、取り急ぎ、SBOMを作り、提出できる

課題：ソフトウェアアップデート毎に毎回手動でSBOMを作る必要がある

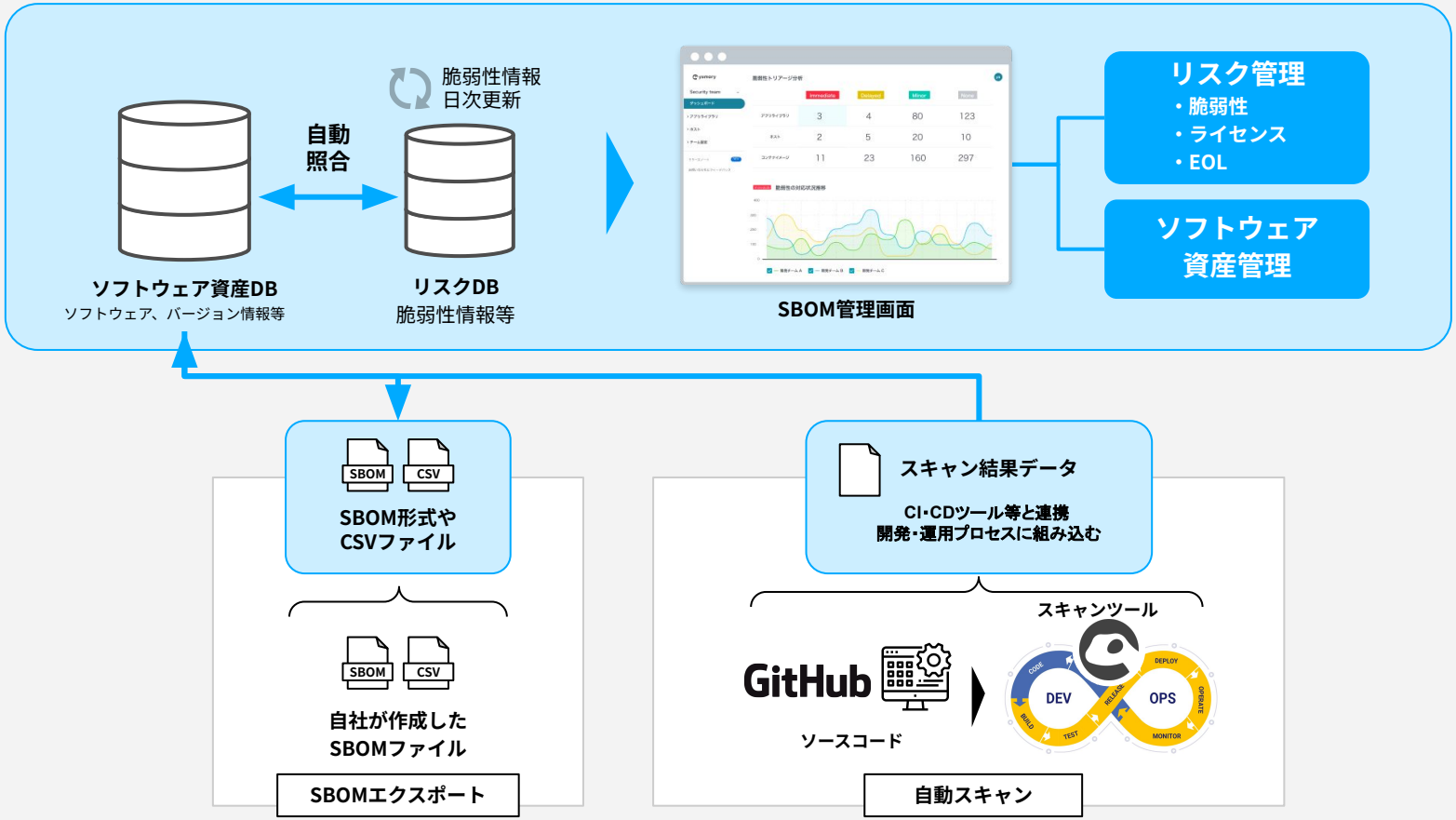
活用段階2：SBOMを継続的に生成できる

メリット：OSSを利用し、安く、継続的にSBOMを生成することができる

課題：リスクを管理するという本来の目的は達成できない

活用段階3：SBOMをDB化、リスク照合も自動化、リスク管理

メリット：ソフトウェア資産DBをベースにリスク管理まで行える



活用段階4：SBOMもインポート管理できる→目指すべき姿

SBOMの課題

SBOMの課題

1. **パッケージ管理システムが利用されない場合、ソフトウェアを正確に検出できない**
    a. パッケージ管理システムは、ソフトウェアの依存関係を管理するため、ソフトウェアの内部依存を正確に把握し、そこから正確な内部ソフトウェア検出が行える
    b. ただ、C/C++の開発現場では、パッケージ管理システムを利用しない開発が未だに主流
    c. 既存のスキャンツールでは正確にソフトウェア検出できないため、手動での確認修正フローが発生する
2. **ソフトウェアの表記揺れにより、他社が作成したSBOMのリスクDBとの照合は難しい**
    a. 表記揺れの問題があり、現状完全自動化は難しい→一定のオペレーションが必要
    b. パッケージ管理システムで管理されているソフトウェアに限定できれば、各パッケージ管理システムの中でソフトウェア名称の一意性が確保され、purlをベースにSBOMのリスク評価が自動化できる

1. **パッケージ管理システムが利用されない場合、ソフトウェアを正確に検出できない**
   a. パッケージ管理システムは、ソフトウェアの依存関係を管理するため、ソフトウェアの内部依存を正確に把握し、そこから正確な内部ソフトウェア検出が行える
   b. ただ、C/C++の開発現場では、パッケージ管理システムを利用しない開発が未だに主流
   c. 既存のスキャンツールでは正確にソフトウェア検出できないため、手動での確認修正フローが発生する
2. **ソフトウェアの表記揺れにより、他社が作成したSBOMのリスクDBとの照合は難しい**
   a. 表記揺れの問題があり、現状完全自動化は難しい→一定のオペレーションが必要
   b. パッケージ管理システムで管理されているソフトウェアに限定できれば、各パッケージ管理システムの中でソフトウェア名称の一意性が確保され、purlをベースにSBOMのリスク評価が自動化できる

パッケージ管理システムとpurlが普及すれば、
SBOM生成→共有→リスク評価が全自動で行える

Appendix

パッケージ管理システムとSBOMに関連する特徴とは？

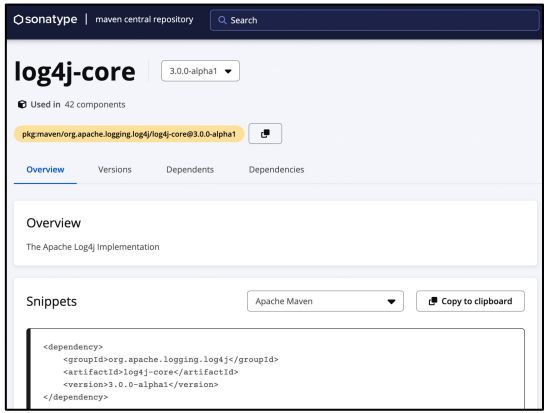
パッケージ管理システムは、必要なソフトウェア部品群をダウンロード、コンパイル、テスト、ビルドまでを自動化するソフトウェア開発/運用の便利ツール

言語別にパッケージ管理システムがある

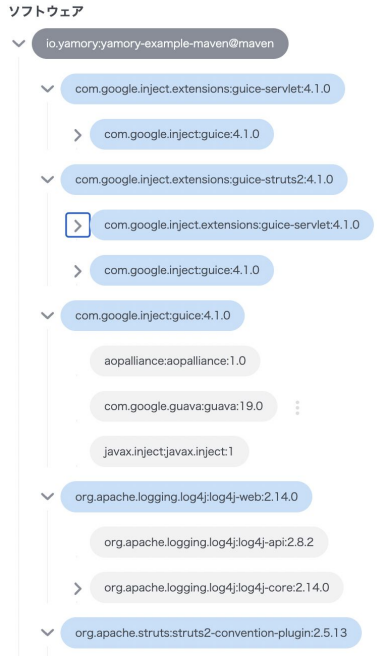
言語	パッケージシステム
C/C++	vcpkg, conan
Java/Kotlin	Maven
Java/Kotlin	Gradle
Java/Kotlin	Android (Gradle)
Scala	sbt
JavaScript/TypeScript	npm
	Yarn
PHP	Composer
Python	pip
Ruby	RubyGems
C#/Visual Basic/F#（.NET）	NuGet
Go	Go Modules
Rust	Cargo

パッケージ管理システムのリポジトリ内でソフトウェア名称が一意

例：Log4Shellのlog4j-coreパッケージは、Java開発でよく使われるMavenのセントラルリポジトリで、**org.apache.logging.log4j:log4j-core**という名称で一意に管理されている



依存関係を解決、管理



purlは、パッケージ管理システムの種別と
パッケージ管理システム内のソフトウェア名称により
一意なソフトウェアを示すことができる識別子

purlの仕様

pkg:{タイプ}/{名前空間}/{パッケージ名}@{バージョン}?{修飾子}#{パス}

Log4Shell（CVE-2021-44228）が発生するソフトウェアの例

pkg:maven/org.apache.logging.log4j/log4j-core@2.13.0以上、2.15.0未満