SBOMを活用した脆弱性管理の取り組み

2023年3月3日

NTTコミュニケーションズ株式会社

イノベーションセンター テクノロジー部門

西野 卓也

SBOM（Software Bill of Materials）の利用目的

サプライチェーンセキュリティ

- サプライチェーンセキュリティの透明性※1を確保する技術
  Supply Chain Security Transparency
  （国内では *Security Transparency* と呼ぶ人も）
- ソフトウェアコンポーネントの透明性※1を確保する技術
  Software Component Transparency

※1 NTIA （米国商務省電気通信情報局）は、SBOMの発行による透明性向上で**サプライチェーンにおけるサイバーセキュリティリスクと総コストを下げられる**と主張している。また、NTIAはSBOMの最小要素に関するレポートも公開している。

01 ライセンス管理

- SBOMの最初の利用目的
- SPDXフォーマットが開発された元々の理由

02 脆弱性管理

- SBOMの応用的な利用
- セキュリティ用途のCycloneDXフォーマット

03 ポリシー&コンプライアンス管理

- 調達に関する制限のチェックなど
- SBOMの新しい注目領域

今回のテーマ

サプライチェーンセキュリティ

- サプライチェーンセキュリティの透明性※1を確保する技術
  Supply Chain Security Transparency
  （国内では *Security Transparency* と呼ぶ人も）
- ソフトウェアコンポーネントの透明性※1を確保する技術
  Software Component Transparency

※1 NTIA （米国商務省電気通信情報局）は、SBOMの発行による透明性向上でサプライチェーンにおけるサイバーセキュリティリスクと総コストを下げられると主張している。また、NTIAはSBOMの最小要素に関するレポートも公開している。

01 ライセンス管理

- SBOMの最初の利用目的
- SPDXフォーマットが開発された元々の理由

02 脆弱性管理

- SBOMの応用的な利用
- セキュリティ用途のCycloneDXフォーマット

03 ポリシー&コンプライアンス管理

- 調達に関する制限のチェックなど
- SBOMの新しい注目領域

ソフトウェアの脆弱性管理について

脆弱性スキャナの結果をDBに格納して管理する仕組みが重要

脆弱性スキャナ

01 ソフトウェア構成分析

02 脆弱性DBの検索

03 結果を統合して出力

出力をDBに格納して管理

脆弱性スキャナとSBOMの関係

ソフトウェア構成分析(SCA)の結果をSBOMとしても収集可能

脆弱性スキャナ

01 ソフトウェア構成分析

02 脆弱性DBの検索

03 結果を統合して出力

SBOMの収集＝**脆弱性スキャナの中間生成物を保存しているだけ**

中間生成物を集めることに意味はある？

基本的にない

一般的なユーザーや開発者は
脆弱性の有無にしか興味がない

脆弱性スキャナとSBOM検討資料の不思議な関係

脆弱性スキャナを分割して再結合すると、脆弱性スキャナの単純利用より嬉しい状態に？？？

機能を分離

機能を分離

脆弱性スキャナ

01 ソフトウェア構成分析

02 脆弱性DBの検索

03 結果を統合して出力

SCAツール + 脆弱性DB < SBOM + 脆弱性DB ？

脆弱性スキャナを分割したら性能が変わるのか？

変わらない

SBOMの中身は**SCAツールの能力に強く依存**

- ライセンス管理
- 脆弱性管理
- ポリシー&コンプライアンス管理

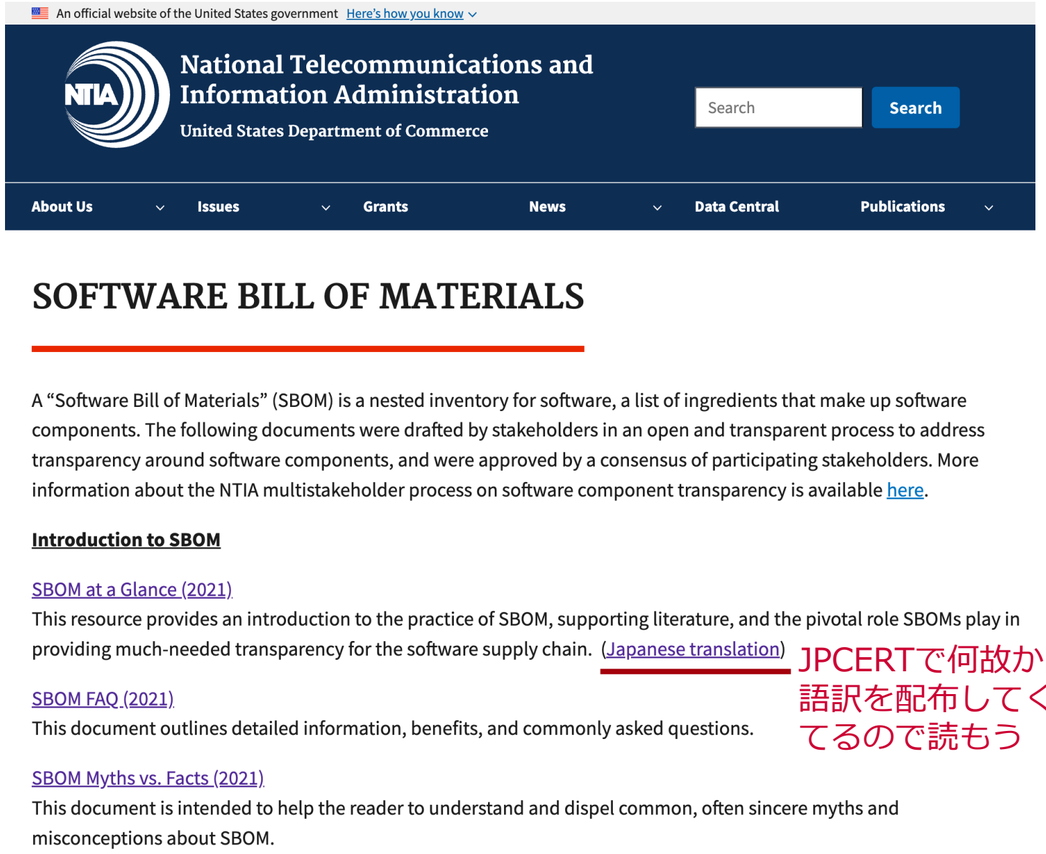
使う目的に合わせてSCAツールも最適化されているはず

NTIAの活動と開発現場の温度差

NTIAが考えるスマートなSBOM

SBOMとは、ソフトウェアコンポーネントやそれらの依存関係の情報も含めた、機械処理可能なインベントリーです。

- NTIA著「SBOM at a Glance」の一般社団法人JPCERTコーディネーションセンターによる翻訳から抜粋 -

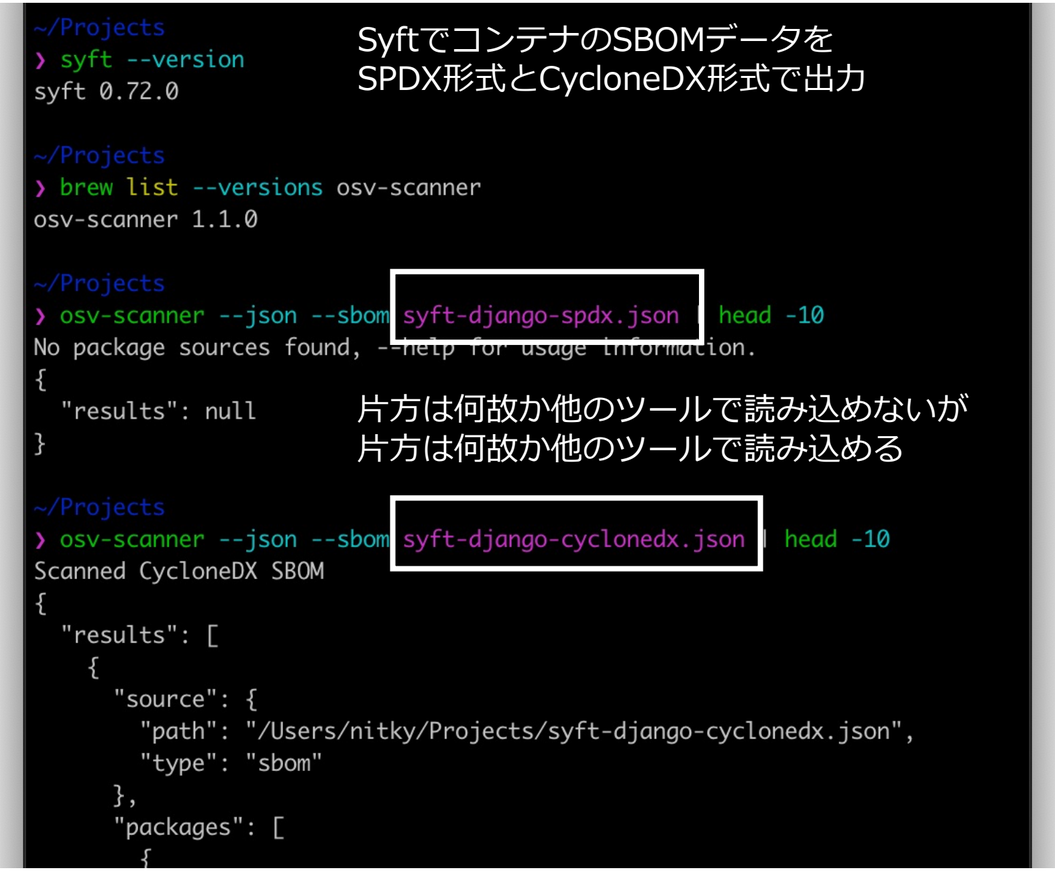


JPCERTで何故か日本語訳を配布してくれてるので読もう

https://ntia.gov/page/software-bill-materials

開発現場でのSBOM検証

「このツールの出力ファイルだと失敗しますが、他のヤツで試したらいけました！！というか、SBOM使う意味って？」



2023年2月21日当時の最新環境

社内のSBOM施策検討で実際に聞いた言葉

そもそもの話、
現場だけだと構成管理の
モチベーションはない

開発現場のモチベーション

現場のモチベーションは、**脆弱性スキャナによるSDLC※1の改善**でSBOMの出力ではない

※1 ソフトウェア開発ライフサイクル

SBOMが性能を改善？

01 ソフトウェア構成分析

02 脆弱性DBの検索

03 結果を統合して出力

SBOMデータの有無と脆弱性スキャナの性能は無関係

性能とは無関係でも、検証性とは関係あり

SDLCの成果物をスキャン
（コード、コンテナイメージ、バイナリ等）

脆弱性スキャナ

01 ソフトウェア構成分析

02 脆弱性DBの検索

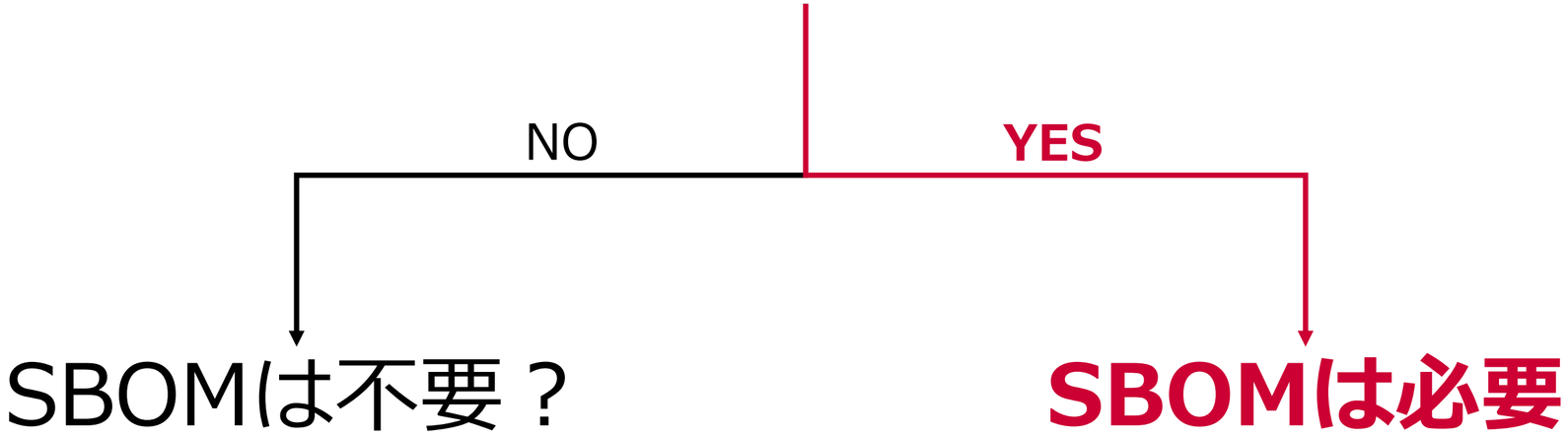
03 結果を統合して出力

脆弱性スキャナの妥当性判断にSBOMが欲しい！

SBOMにないコンポーネントは検査できていない可能性が高い

脆弱性検査の取り組みを評価したいのか？

SDLCを評価する必要が社内や社外にある



SDLCのセキュリティ的な観点を
客観的に検証するためにSBOMは必要

容易にできる？

SBOMがあれば容易に検証できるわけではない

2つのSBOMフォーマットと、３つのOSS脆弱性スキャナの観点から説明

SPDX

CycloneDX

Trivy

Grype（Syft※1）

OSV-Scanner

※1 SyftはGrype内部で使われているはSBOM用ライブラリ（単体でCLIツールとしても動作可）

脆弱性管理の観点でみたSBOMの主要フォーマット

	SPDX	CycloneDX
特徴	• **最も人気があるSBOMフォーマット** • **ISO/IEC 5962:2021** • **コンプライアンスに関する情報管理に強い** • **詳細な分析情報を提供できる** • **SBOM領域の全てをカバーする高い理想を持つ規格設計** • **少なくとも脆弱性検知の観点ではオーバースペック**	• **セキュリティ向けのSBOMフォーマット** • **サイバーリスク管理に十分な情報の管理** • **SPDXに比べてシンプルな構造** • **実用性を重視した実装ありきの規格設計** • **脆弱性検知の観点では扱いやすい**
最新バージョン (2023/2月現在)	**2.3**	**1.4**
主な開発支援団体	**Linux Foundation** Linux Foundationプロジェクトの1つであるSPDX Working Groupが開発	**OWASP Foundation** OWASP Foundationの支援を受けてCycloneDX Core Working Groupが開発
SBOMデータの出力例 (djangoコンテナを対象に M2 MBAで実行)	**Trivy v0.37.3のスキャン** • ファイル形式：SPDX 2.2 • 処理時間：4.892秒 • 行数：4,724行 • 検出されたアーティファクト:185 **Syft v0.60.1のスキャン** • ファイル形式：SPDX 2.2 • **処理時間：60.61秒** • **行数：121,218行** • 検出されたアーティファクト:188 検出差分パッケージ(purlで記述) • "pkg:generic/python@3.4.5" • "pkg:maven/gettext/gettext" • "pkg:maven/libintl/libintl"	**Trivy v0.37.3のスキャン** • ファイル形式：CycloneDX 1.4 • 処理時間：3.006秒 • 行数：7,219行 • 検出されたアーティファクト:185 **Syft v0.60.1のスキャン** • ファイル形式：CycloneDX 1.4 • 処理時間：56.49秒 • 行数：16,568行 • 検出されたアーティファクト:188 検出差分パッケージ(purlで記述) • "pkg:generic/python@3.4.5" • "pkg:maven/gettext/gettext" • "pkg:maven/libintl/libintl"
個人的な所感	国際オープンな標準(ISO)とは思えないほど自由度が高いフォーマット、SCAツールの性能に内容が大きく左右される。とにかく多くの情報を出力きでるため、パース機能に不具合があると取り回しが難しい。今後に期待。	脆弱性管理に必要な情報の記載が、SCAツールによって大きくブレないところが個人的に良い。機械だけでなく、頑張れば人間もギリギリ読めるレベルの情報量であるため、不具合が起こった場合にバグ特定が比較的容易。

SBOM対応のOSS脆弱性スキャナ

	Trivy	Grype (Syft)	OSV-Scanner

特徴	• **マルチOSで動作(Mac, Linux)**
	• **スキャンが高速**
	• **開発用パッケージの検知除去が可能**
	• **ローカルの脆弱性DBがBoltと呼ばれるKey-Valueストア**
	• **SPDXとCycloneDXのスキャンに対応**
	• **(SPDXとCycloneDXの出力に対応※1)**
		• **マルチOSで動作(Win, Mac, Linux)**
		• **GrypeのSBOM処理はSyftで実装されている**
		  • SyftはSBOM出力ツールとして非常に有名
		  • SyftもGrypeも開発は同じ会社
		• **ローカルの脆弱性DBがSQLite**
		• **独自形式（Syft）、SPDX、CycloneDXのスキャンに対応**
		• **CycloneDXの出力に対応※2**
			• **マルチOSで動作(Win, Mac, Linux)**
			• **スキャンが高速**
			• **Googleが開発した脆弱性スキャナ**
			• **オンラインの脆弱性DBに問い合わせ**
			• **SPDXとCycloneDXのスキャンに対応**
			• **Open Source Vulnerability format (OSV format)でレポートを出力※3**

最新バージョン(2023/2月現在)	**0.37.3**	**0.57.1 (Grype)** 0.73.0 (Syft)	**1.2.0**
主な開発元	**AquaSecurity** クラウド系のセキュリティ企業	**Anchore** サプライチェーンセキュリティ系のセキュリティ企業	**Google** 検索エンジンで有名なあの会社
SPDX対応状況(2023/2月現在)	**2.2** SPDX公式が配布しているライブラリ(spdx/tools-golang)を利用	**2.3** SPDX公式が配布しているライブラリ(spdx/tools-golang)を利用。Syftの古いバージョン(0.60.1以下)を使えば2.2を利用可	**2.3** SPDX公式が配布しているライブラリ(spdx/tools-golang)を利用
CycloneDX対応状況	**1.4** CycloneDX公式が配布しているライブラリ(CycloneDX/cyclonedx-go)を利用	**1.4** CycloneDX公式が配布しているライブラリ(CycloneDX/cyclonedx-go)を利用	**1.4** CycloneDX公式が配布しているライブラリ(CycloneDX/cyclonedx-go)を利用
JSONとして入力できるSBOM形式	**SPDX(2.2), CycloneDX(1.0-1.4)**	**独自形式(Syft), SPDX(2.3), CycloneDX(1.0-1.4)** ドキュメント上はまだ2.2のままかも	**SPDX(2.3), CycloneDX(1.0-1.4)**
JSONとして出力できる形式	**独自形式, SPDX(2.2), CycloneDX(1.4)**	**独自形式, CycloneDX(1.4)**	**独自形式(OSV format)**
備考	※1 これらのSBOM形式を出力する場合は、脆弱性スキャンを同時に実行できない	※2 Syftは、SPDX(2.3)とCycloneDX(1.4)のどちらも出力可能	※3 すべての依存関係をオフラインで取得するPRがあり、近日中にSBOM的な使い方が可能に？

脆弱性管理に向けたSBOM戦略

検証性や動作速度の観点から**CycloneDX+Trivy**の組み合わせがオススメ(2023年2月時点)
SBOM標準を意識するならSPDX+Grype(Syft)だが、検証の複雑さや速度の点で脆弱性管理の検証に向かない

SPDX

CycloneDX

Trivy

Grype (Syft)

OSV-Scanner

ゴールはSDLCの改善と評価

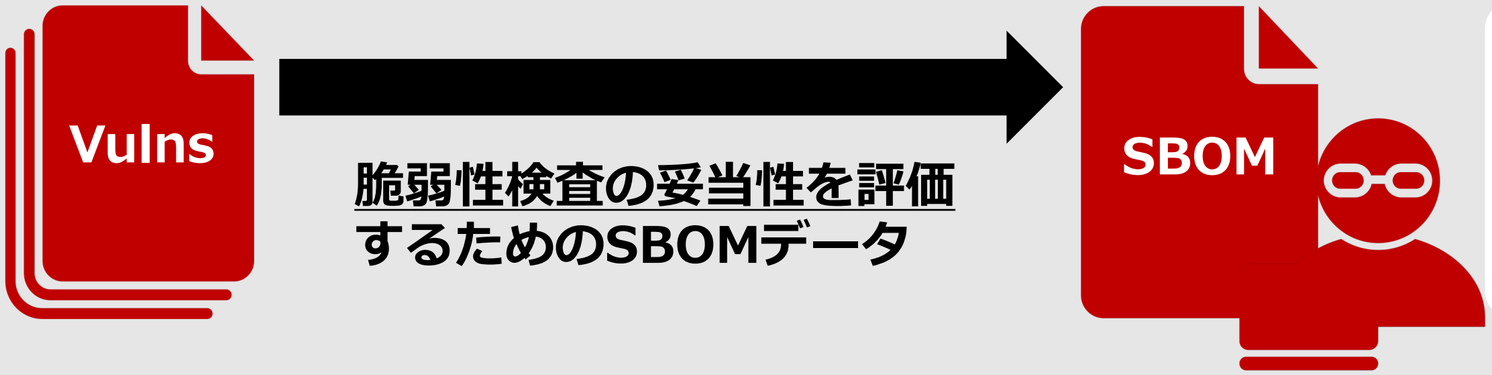
客観的に評価可能なセキュリティ運用

1. **脆弱性検査のレポート報告**
   - 脆弱性スキャンの実施
2. **SCAコンポーネントの動作保証**
   - SBOMデータの保存と分析
3. **検出された脅威の調査や対応**
   - パッチ管理など
4. **脅威の無力化に関する報告**
   - 対処レポートとSBOMデータの提出

脆弱性検査とSBOM出力
例：Trivyスキャン＆CycloneDX出力

SBOMデータを利用した
脅威の発見とSDLC評価

脆弱性管理

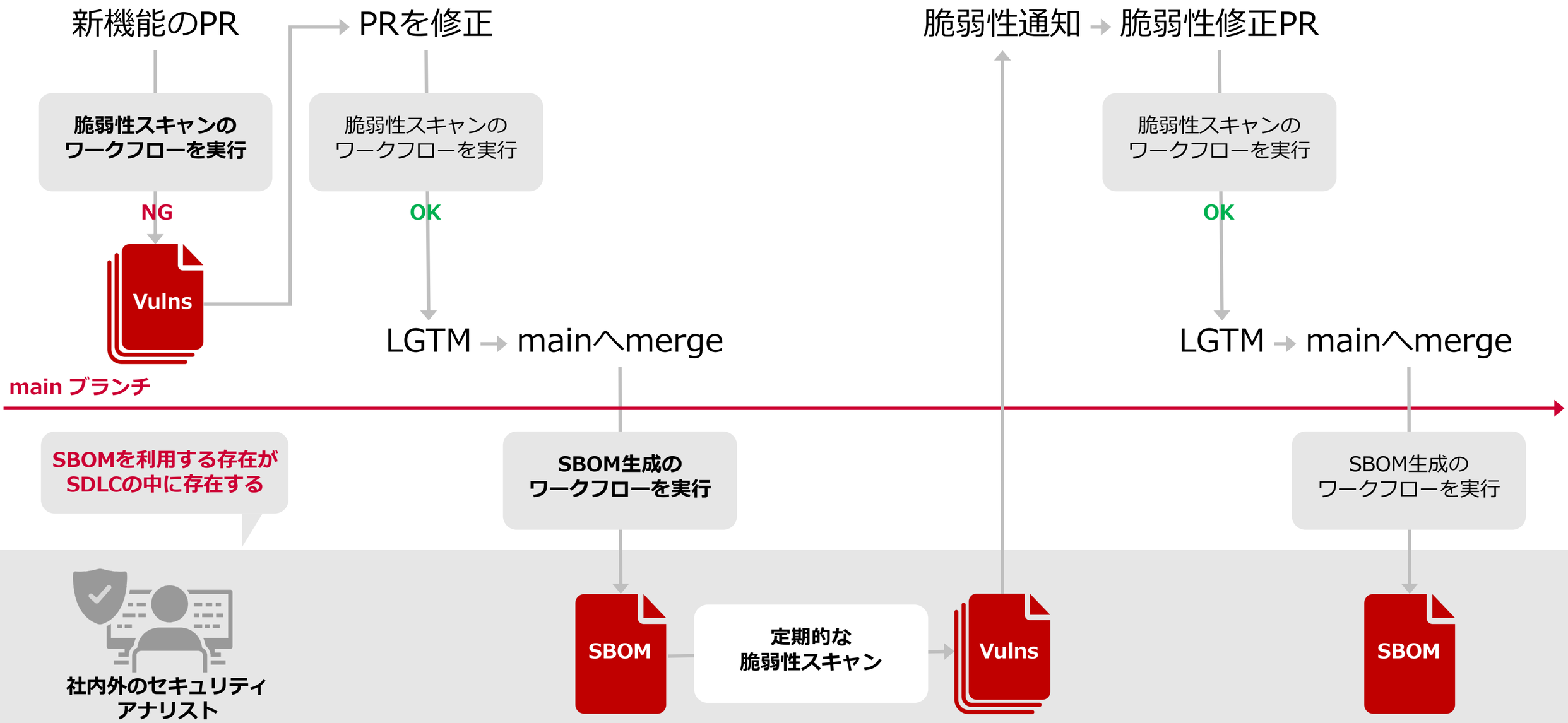


脆弱性検査の妥当性を評価するためのSBOMデータ

脆弱性検査の実施とSBOM利用のどちらを優先すべき？

少なくとも脆弱性管理の観点では、
脆弱性スキャナのCI/CD導入が先

GitHub上でのSDLCと脆弱性スキャナ



GitHub上でのSDLCと脆弱性スキャナ（Trivyを例に）

新機能のPR → PRを修正

Trivyで
脆弱性スキャン

NG

Trivyで
脆弱性スキャン

OK

LGTM → mainへmerge

脆弱性通知 → 脆弱性修正PR

Trivyで
脆弱性スキャン

OK

LGTM → mainへmerge

main ブランチ

TrivyでCycloneDX
ファイルを出力

SPDXとCycloneDXの
両方を生成して配布す
る場合が多いかも？

TrivyでCycloneDX
ファイルを出力

社内外のセキュリティ
アナリスト

定期的な
脆弱性スキャン

SDLCの評価は？

脅威の発見とSDLC評価

脆弱性検査をしてSBOMデータを出力するだけでは、SBOMの活用とはいえない

1. 脆弱性検査のレポート報告
   - 脆弱性スキャンの実施
2. SCAコンポーネントの動作保証
   - SBOMデータの保存と分析

脆弱性検査とSBOM出力
例：Trivyスキャン＆CycloneDX出力

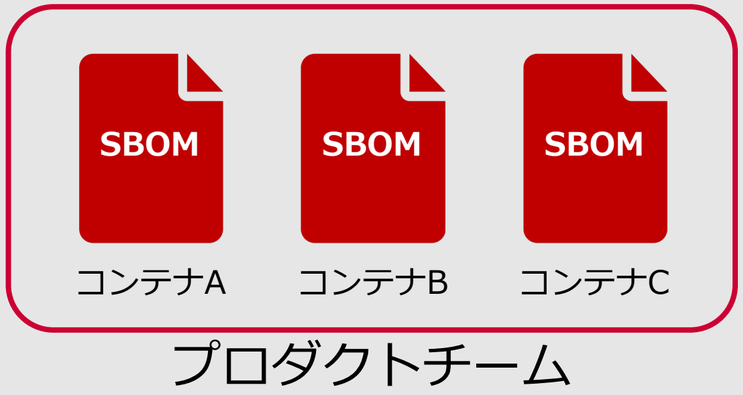
3. 検出された脅威の調査や対応
   - パッチ管理など

4. 脅威の無力化に関する報告
   - 対処レポートとSBOMデータの提出

SBOMデータを利用した脅威の発見とSDLC評価

コンテナ環境（K8s/Docker）であれば、
OSSの**KubeClarity**が比較的試しやすい
https://github.com/openclarity/kubeclarity

脆弱性管理



プロダクトチーム

- プロダクトチームは**複数のSBOMデータを提出することがある**
- プロダクトチームで**パッチアップデートが容易とは限らない**
  - アップデートの容易さは業種や契約、提供サービスに依存
- アナリストは**複数のプロダクトチームを対象**に脆弱性管理を行う

脅威の発見と対処に特化したツールの内製

確認できる範囲のリポジトリを統計調査して、既存のツールで対応が可能かどうかを判断する

特殊なアプライアンスを扱う業種や、パッチアップデートが困難なサービスでは内製開発がほぼ必須？

Threatconnectome

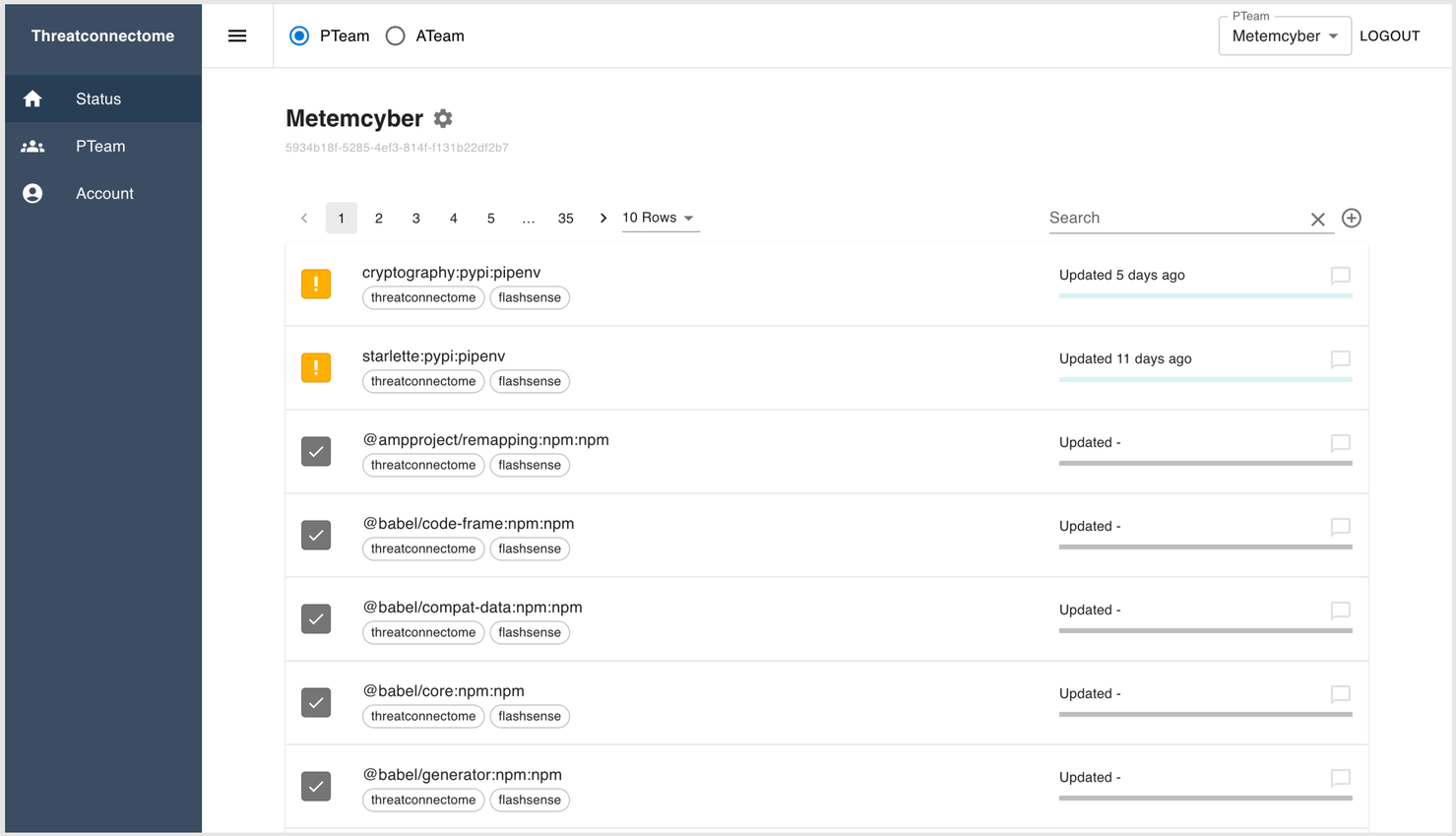
脅威分析からサービス影響を把握し、
その優先度に応じた予防措置を管理する

01 SBOMを利用した攻撃領域管理で通知の無駄や確認作業を削減

02 脅威制御アクションを利用した明快な脅威影響のコントロール

03 対処ログからチームのSDLCとセキュリティスキルを評価

今日紹介した全てのOSS脆弱性スキャナ(Trivy, Grype/Syft, OSV-Scanner)に対応済み



2023年7月～9月にOSS公開予定

SBOMを実運用するための課題は山積み

新機能のPR → PRを修正

脆弱性通知 → 脆弱性修正PR

アーティファクトの分析能力はSCA機能に依存

NG

Vulns

SCA機能に最適化したパッケージ管理手法の導入

OK

内製開発ライブラリが多い場合は**OmniBOR(GitBOM)**を用いたアーティファクト解析＆追跡補助も検討

アップデート対処できない場合はどうするのか？

OK

purlで問題を通知して作業者は対処可能？

LGTM → main へmerge

LGTM → main へmerge

main ブランチ

人員をどうやって確保して評価体制をつくるのか？

SBOMとして出力したいデータと利用目的の一致

そもそも現在の成果物がSBOM管理できる対象なのか、SBOM対応のためにコードを完全に書き直す必要はあるのか？

継続的にSBOMデータを活用できる状態なのか？

社内外のセキュリティアナリスト

内製開発ライブラリが多い場合や、マイナーなOSSライブラリを利用している場合は、外部の脆弱性DBの利用では不十分な場合も？

SBOM

脆弱性情報の管理はpurlベースで十分か？

Vulns

purl(package URL)のURLエンコードやsubpathの扱い（有無だけでなくdistroのように出力内容自体が違う場合あり）を完全に理解して処理しているか？

本当に必要なものはStatic SBOMとRuntime SBOMのどちらか？

まとめ

- SBOMの利用に関してはNTIAの動向を把握すると良い
  - 今後もサプライチェーンセキュリティに関しては、NTIAのドキュメントが標準になる可能性大
- NTIAの「理想的なSBOM運用」に現実はまだ追い付いていない
- **脆弱性管理はSBOMの理解しやすいユースケースだが、曖昧な知識でやると現場が混乱する**
  - **前提：「脆弱性検査の取り組みを評価」するためにSBOMデータの提出が必要**
    - 脆弱性管理の点において、**評価が必要でなければSBOMデータの提出は不要**
    - 開発者自身がセキュリティ対応をする場合は、セキュリティの取り組みが開発者の自己満足で良い場合も実際は多い
  - 開発の現場においては、「SBOMの導入」ではなく**SDLCの評価や改善**を目的にすること
    - 脆弱性スキャナ導入が先、SBOM利用は後
- **脆弱性スキャナ**や**SBOM出力用のSCAツール**対応には**開発チームの協力が必須**
  - プロダクトのコード管理やCI/CDのレベルでSBOM対応が必要になる場合がほとんど
    - 例：npmのようなパッケージ管理システムの導入、Github ActionのTrivy導入など
  - どんなものでもSBOMデータを正確に出してくれる魔法のようなツールは現状ない
- 提出されたSBOMデータを使って**脆弱性管理の負担軽減**や**脅威分析の改善**が可能
  - NTTコミュニケーションズでは業務の特殊性から内製開発に取り組んでいたが、KubeClarityなどのOSSツールで十分な場合も

SBOMのエコシステム自体が大きく変化している状態で、企業を超えた共有にはまだまだ課題感あり。攻撃者側にとって有益な情報も多く、業種によってはTLP※1のような仕組みがSBOM共有で必要とされる可能性も

※1 Traffic Light Protocol