SBOM 導入・運用の手引き

2024 年 12 月

独立行政法人情報処理推進機構

産業サイバーセキュリティセンター

中核人材育成プログラム

SBOM プロジェクト

目次

前書き

本ガイドラインは独立行政法人情報処理推進機構の産業サイバーセキュリティセンターが開催している中核人材プログラムのカリキュラムである卒業プロジェクトの一環として作成されました。

昨今、Apache Log4j の脆弱性や SolarWinds に対するサプライチェーン攻撃をはじめとするソフトウェアサプライチェーンへの攻撃が非常に大きな影響を与えました。ソフトウェアサプライチェーンセキュリティに対して SBOM が効果的であることから、米国の大統領令や EU の欧州サイバーレジリエンス法案を契機に SBOM の導入が急速に進められています。日本においても経済産業省が先頭に立ち SBOM 導入に向けて検討が推し進められています。しかしながら、急速な SBOM 推進の流れに対し、導入する企業がそのスピードについていくことができず、SBOM の導入・運用に関する知見が足りないという課題が浮き彫りとなりました。

本ガイドライン作成のプロジェクトではこのような課題を解決したいというメンバーが集まり、プロジェクトの立ち上げに至りました。検討の結果、我々は SBOM の導入・運用に関して圧倒的に SBOM に関するドキュメントが少ないと結論づけました。日本においては SBOM に関する適切な書籍はなく、ドキュメントも圧倒的に少ないという状況です。そこで本プロジェクトでは、一人の担当者の視点で実践し、討論を重ね、その過程で実施すべきこと、気を付けるべきことを文書化しました。本ガイドラインが、これから SBOM を導入・運用する皆様の一助となることを願っております。

免責事項

- 本書は単に情報として提供され、内容は予告なしに変更される場合がある。
- 本書に誤りがないことの保証や、商品性または特定目的への適合性の黙示的な保証や条件を含め明示的または黙示的な保証や条件は一切ないものとする。
- 本書に記載の内容は、独立行政法人情報処理推進機構および産業サイバーセキュリティセンターの意見を代表するものではなく、著者の見解に基づいている。
- 本書の利用によるトラブルに対し、本書著者ならびに監修者は一切の責任を負わないものとする。
- 本書の有効期限は、発行日から 2 年間とする。

1. 背景と目的

1.1 背景

現在ソフトウェアは社会の多岐にわたる領域で不可欠な役割を果たしており、その影響力は将来においてもさらに拡大するものと見込まれる。産業界におけるデジタル化の波は加速の一途を辿り、人工知能（AI）、ビッグデータ、クラウドコンピューティングの利用が一般化している。例えばヘルスケア分野では、デジタル化による遠隔診断や診断結果の解析、治療方針策定などが進みつつあり、金融技術（フィンテック）分野では個人間決済の拡大や仮想通貨・ブロックチェーン技術による新規ビジネスが生まれている。また、自動車業界では自動運転技術の発展、コネクテッドカーの普及など、ソフトウェアは自動車の設計、製造、運用のすべての面で中心的な役割を果たしている。そして、他の製造業でも産業用ロボット、IoT（Internet of Things）、AI を利用したスマートファクトリーが登場し、生産効率の向上とコスト削減が進んでいる。

しかし、これに伴いソフトウェアの脆弱性管理の必要性も高まっており、製品・サービスの安全と安心を保証するためには、開発段階から製品出荷後に至るまでの脆弱性管理が求められている。また、ソフトウェアサプライチェーンの複雑化やオープンソースソフトウェア（以下 OSS と称する）利用の一般化により、自社製品に含まれるソフトウェアコンポーネントの把握が難しくなっている。

このような背景の下、「Software Bill of Materials（以下 SBOM と称する）」の活用が注目を集めている。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を含む機械処理可能な一覧リストのことである。日本語では「ソフトウェア部品表」とも呼ばれ、脆弱性管理やライセンス管理の課題解決に有効であると考えられている。

海外では、米国商務省の電気通信情報局（NTIA）において 2018 年 7 月より SBOM に関する実証が開始され、2021 年 5 月に米国バイデン大統領が署名した大統領令を一つの起点として、世界的に普及が進んでいる。さらに「欧州サイバーレジリエンス法案」においては、主要な項目として「すべてのサイバーセキュリティリスクについての文書化」が挙げられ、SBOM による管理が義務付けられる。

一方で、SBOM には以下のような課題が挙げられる。

(1) ソフトウェア・システムの全体構成が不明な場合、SBOM ツールの適用範囲の設定が困

難であり、効果的なリスク管理ができない。

(2) SBOM ツールの導入には環境整備や学習に多くの工数が必要となる。

(3) 無償の SBOM ツールの場合、サポートが少ないことにより環境構築や運用にあたっての情報が不足しており、導入・運用に大きな工数を要する。また、再帰的に利用されるコンポーネントの検出やライセンスの検知漏れが問題となる。

(4) SBOM ツールを使用しても、コンポーネントの検知漏れや誤検出が発生する場合があり、出力結果の精査が必須である。

(5) SBOM は脆弱性対応において、コンポーネントに対する脆弱性の特定までに効果を発揮するため、以降の特定されたコンポーネントの脆弱性に対する対応の可否や対応優先度の判断には大きな効果を発揮しない。しかし、特定できる脆弱性は増加するため、対応リソースが不足する恐れがある。

総じて、SBOM は効率的なソフトウェア管理を実施する上で有効であることが確認されているが、実際の導入に際してはさまざまな課題が存在し、これらを克服するためのさらなる検討と技術開発が求められている。これらの課題を踏まえ、本手引きでは SBOM の普及と導入に関連する主要な課題は以下と想定する。

(1) 普及の必要性

SBOM の効果を最大限に引き出すためには、広範囲にわたる普及が必要であり、これには業界全体の認識向上と協力が求められる。

(2) 導入の難しさ

現状、SBOM を導入するための具体的な知見が不足している上、有償の管理ツールが高価であるため、新規導入する企業にとって大きな負担となっている。

(3) 適切な管理の重要性

SBOM は導入しただけでは十分な効果は発揮せず、SBOM 情報を管理し運用することで脆弱性管理およびライセンス管理に効果を発揮する。すなわち SBOM を有効活用するためには、SBOM の知識に加えて脆弱性やライセンス管理に関する知識が必要になる。

(4) 導入に対する抵抗感

上記の(1)〜(3)の課題から、多くの企業にとって SBOM の導入は高いハードルと感じられている。

1.2 目的

本ガイドラインでは、背景で述べた SBOM の普及と導入に関する課題を解決するために、SBOM の導入・運用における現実的な取り組みの一例や知見を提供する。SBOM および SBOM ツールの使用によって得られた筆者らの経験をガイドラインとして共有することで、SBOM 導入・運用時のハードルを下げることを目的とする。なお、SBOM 運用・管理においては、運用開始直後から完璧を目指さず、まずは必要最低限のレベルを獲得し、その後よりよい SBOM 運用に向けて改善を行うことが重要である。

1.3 主な対象読者

本手引は、これから SBOM の導入を検討している企業や、直近に SBOM を導入したがより良い運用につなげることができていない企業を対象としている。SBOM に関係する部門としては、SBOM を作成する開発者に加えて品質管理部門及び PSIRT、SBOM を活用して脆弱性を管理するセキュリティ部門が想定される。SBOM を提供する者、利用する者が相互に SBOM に関する十分な知識を持つことで、SBOM を適切に利用することができる。

1.4 活用方法

本手引は、SBOM を導入・運用するための実施事項を記載している[1]。SBOM を適切に利用するための基本的な知識の認識や各フェーズの実施項目と具体的な方法を記載し、実業務担当者の SBOM の導入・運用を手助けする。

第 2 章では、SBOM の概要、第 3 章では、導入・運用に関する基本指針、第 4 章では、環境構築・体制整備フェーズにおける実施事項、第 5 章では、SBOM 作成・共有フェーズにおける実施事項、第 6 章では、SBOM 運用・管理フェーズにおける実施事項について具体的に記載していく。

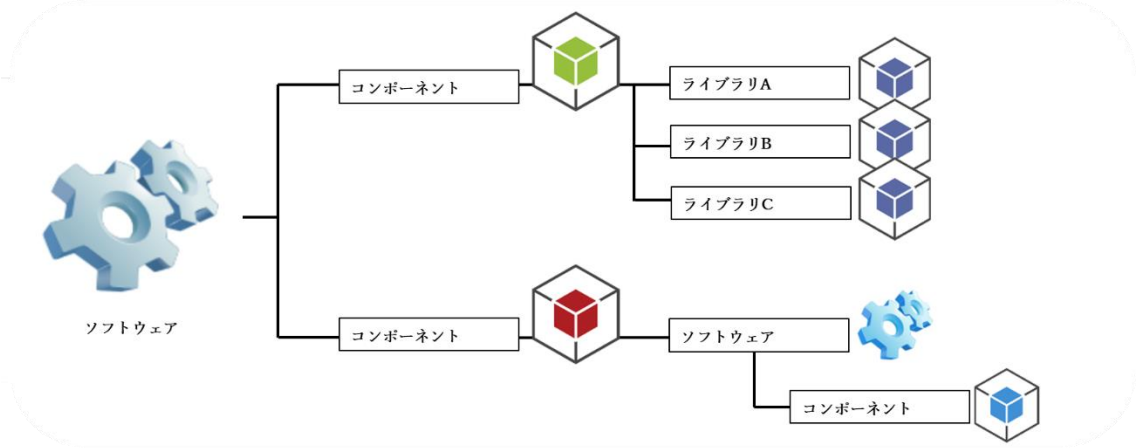
---

[1] 経済産業省「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」[1]。

2. SBOM の概要

2.1 SBOM とは

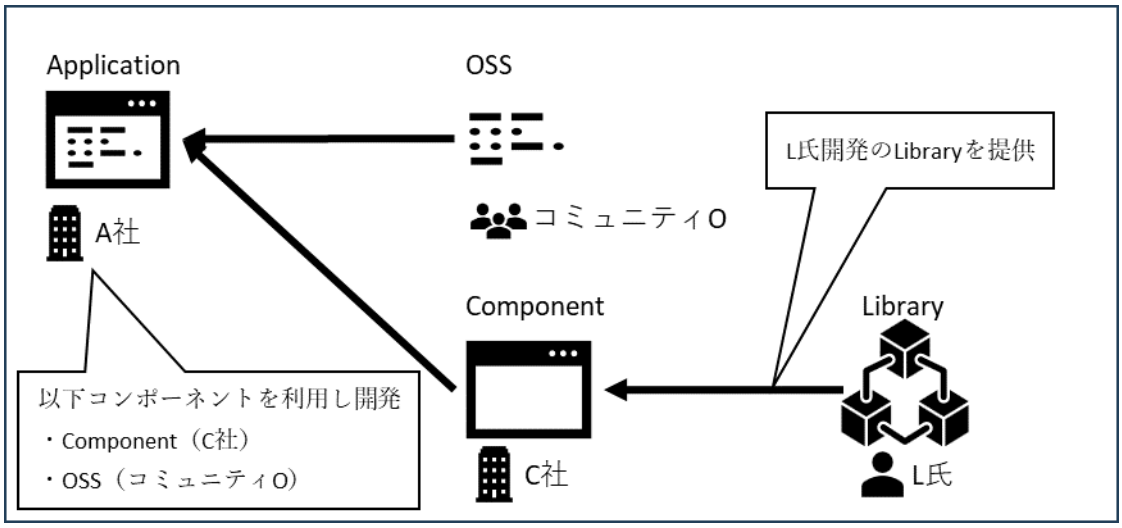
SBOM とは、図：2.1.1 で示すようにソフトウェアを構成するコンポーネントの名称やバージョン情報、依存関係、開発者情報などを含むソフトウェアの部品表を指す。情報には OSS やプロプライエタリソフトウェアに関する情報も含めることが可能であり、機械処理可能なフォーマットが提供されている。



図： 2.1.1 SBOM の概要

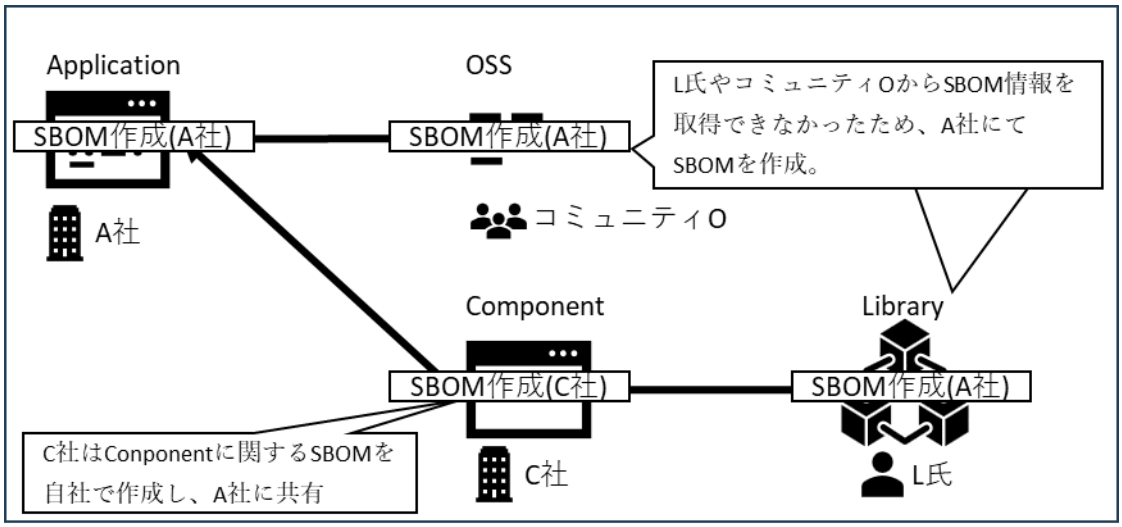
SBOM を関係者間で相互共有することで、ソフトウェアサプライチェーンの透明性の強化が期待されており、依存関係まで含むコンポーネントの詳細な情報を明らかにすることから脆弱性管理やライセンス管理の課題に対する解決策の一つとして注目されている。

SBOM 作成について仮想企業 A 社が開発したソフトウェアである Application を例にソフトウェアに関連する組織に関する SBOM の作成、共有の流れを説明する。まずは Application の構成イメージを図：2.1.2 に示す。



図： 2.1.2 Application の構成イメージ

上述の Application の SBOM を作成するケースでは、ソフトウェアのコンポーネントを開発している全ての組織が SBOM を作成できないことも考慮している。図：2.1.3 では Application に関するそれぞれの SBOM をどの組織が作成、提供しているかを示す。



図： 2.1.3 Application の SBOM 作成

図：2.1.3 に示したように SBOM を作成した結果、「Application」を構成するコンポーネントや、その依存関係、サプライヤー名、バージョンなどの情報を確認することができる。表：2.1.1 に本ケースにおける SBOM を作成した際の概念的イメージを示す。

表：2.1.1　Application における SBOM の概念的イメージ

サプライヤー名	コンポーネント名	バージョン情報	その他の一意の識別子	依存関係	SBOM 作成者	タイムスタンプ
A 社	Application	1.1	PURL	1 層目	A 社	05-30-2024 17：00：00
C 社	Component	2.2	CPE	2 層目	C 社	05-04-2024 12：00：00
コミュニティ O	OSS	1.3	PURL	2 層目	A 社	05-30-2024 17：00：00
L 氏	Library	3.1	PURL	3 層目	A 社	05-30-2024 17：00：00

表：2.1.1 はあくまでイメージであり、実際は機械処理可能なフォーマットで出力されており、管理ツールに取り込むことで、可視化され管理可能となる。SBOM を活用することより、特定のコンポーネントに脆弱性が発見された際に、対象のコンポーネントがソフトウェアに含まれているかを即座に認識する。この結果、発見された脆弱性に対して迅速な対応

が可能となる。SBOM を作成し、相互に共有することでソフトウェアサプライチェーンの透明性が高まり、自組織以外の脆弱性対応速度の向上につながる。

2.2 SBOM 導入のメリット

SBOM 導入による直接的なメリットとしては、大きく分けて脆弱性管理能力の向上、ライセンス管理能力の向上の 2 つが挙げられる。

ソフトウェア開発の現場では近年、他社や OSS コミュニティが開発したソフトウェアをコンポーネントとして用いた開発が多く見られるようになっている。また、その開発でコンポーネントとして使用したソフトウェア自体が、さらに他のソフトウェアをコンポーネントとして使用していることも多く、複雑な階層構造や依存関係が存在している。そのため、ソフトウェア開発におけるサプライチェーンは複雑化している。

ソフトウェア開発の複雑化したサプライチェーンにおいて、特に問題となるのが脆弱性管理とライセンス管理の問題である。従来のコンポーネント管理では自らが開発したソフトウェアで使用しているコンポーネントの管理のみを実施していた。しかし 2021 年 12 月に発見された Apache Log4j のゼロデイ脆弱性の事例では、ソフトウェアを構成するコンポーネントとして Apache Log4j を使用していた組織も多く、その場合、従来のコンポーネント管理の手法では検知・対応することが困難であった。これはライセンス管理においても同様である。

OSS のライセンス形態である GNU General Public License(以下 GPL と称する)では、GPL が適用されたソフトウェアをコンポーネントとして利用していた場合でも GPL が適用され、ソースコードの開示が求められる。従来までのライセンス管理ではコンポーネントとして GPL ソフトウェアを利用していた場合、意図せずに GPL 違反となる可能性がある。

SBOM を使用しコンポーネント管理を行うことにより、従来のコンポーネント管理では行うことができなかった複雑な階層構造の脆弱性やライセンスについても検知することが出来ることで、脆弱性管理やライセンス管理に大きく貢献する。なお、国内では経済産業省が 2019 年から「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」を開催し、さらに SBOM 導入に向けた実証を 2021 年以降に実施した。この実証を通じて、以下の SBOM のメリット・効果が確認できたと報告されている。

(1) SBOM を作成・管理することで、ソフトウェア内のコンポーネントの脆弱性を早期に特

定し、リスクを低減するとともに、対応に必要な工数も減少する。

(2) SBOM を用いることでライセンス情報を正確に把握し、コンプライアンス違反のリスクや管理工数を削減できる。特に、SBOM ツールの利用により、ライセンスの内容表示や必要な警告機能が提供され、より効果的なライセンス管理が可能となる。

(3) 有償の SBOM ツールを使用することで、OSS の依存関係や再帰的利用（再利用ソフトウェア部品）の検出・管理が効率的に行える。

また、経済産業省の報告では、これらの直接的なメリットの他にも既知の脆弱性やライセンスの問題を開発時に早期に特定することによる「開発生産性の向上」や SBOM ツールの機能も活用した「ソフトウェアの EOL 管理」、SBOM 対応であることによる「製品価値や企業価値向上」のメリットについても取り上げられている。

2.3 SBOM の最小要件

2021 年 5 月の米国大統領令を受け、米国商務省の電気通信情報局（NTIA）は 2021 年 7 月に SBOM の「最小要素」の定義に関する文書[2]を公開した。SBOM の「最小要素」の定義は、以下の 3 つの分野に分類される。

- データフィールド（Data Fields）
- 自動化サポート（Automation Support）
- プラクティスとプロセス（Practices and Processes）

データフィールド（Data Fields）

データフィールドでは、ソフトウェアを構成するコンポーネントを一意に特定するための情報として以下の表：2.3.1 の情報を含めることが定義されている。

表：2.3.1 最小要素におけるデータフィールドの定義

項目	説明
サプライヤー名	コンポーネントを開発、定義及び識別するエンティティの名称。
コンポーネント名	サプライヤーによって定義された、ソフトウェアのある単位に対する名称。

[2] 米国 NTIA：The Minimum Elements For a Software Bill of Materials (SBOM)[3]

バージョン情報	コンポーネントを識別するために使用されるバージョンに関する識別子。
その他の一意の識別子	コンポーネントを識別するために使用される又は関連するデータベースの検索キーとして機能するその他の識別子。
依存関係	コンポーネントがあるソフトウェアに含まれているという関係性の特徴づけの情報。
SBOM 作成者	コンポーネントの SBOM を作成するエンティティの名称。
タイムスタンプ	SBOM データを作成した日付と時刻の情報。

自動化サポート（Automation Support）

SBOM の自動生成や可読性等の自動化をサポートするために定義されており、SBOM データは機械判読可能かつ相互運用可能なフォーマットを用いて作成・共有される。現状では、国際的な議論を通じて策定された表：2.3.2 のタグを用いることが推奨される。

表：2.3.2 最小要素におけるタグ情報の定義

タグ名	概要
SPDX	コンポーネントのライセンス情報を標準化された形式で記述した情報を含むフォーマットであり、RDF、JSON、YAML 等で表現可能。
CycloneDX	脆弱性の追跡やリスク評価に重点を置きコンポーネント間の依存関係を詳細に記述するフォーマットであり、XML と JSON で表現可能。
SWID	ソフトウェアの識別情報を提供し、インストールされているソフトウェアの管理を容易にするフォーマットであり、XML で表現可能。

プラクティスとプロセス（Practices and Processes）

SBOM の「最小要件」では SBOM の要求、生成、利用に関する運用方法を定義することとされており、SBOM を利活用する組織は、以下の表：2.3.3 の項目に関する運用方法を定めることが必要となる。

表：2.3.3 最小要素における運用方法の定義

項目	概要
SBOM の作成頻度	定期的またはイベント駆動で更新し、最新の情報を維持できる作成頻度を定める。
SBOM の深さ	ソフトウェアのレベルに応じて、作成する SBOM の深さを定める。
既知の未知	作成した SBOM において完全なコンポーネントの依存性が未知である場合、それが未知であることを記載する。
SBOM の共有	SBOM のフォーマット、セキュリティとアクセス制御を意識した共有方法を定める。
アクセス管理	認証、認可、監査ログで SBOM へのアクセスを管理する運用を定める。
誤りの許容	情報の正確性を確保し、許容範囲を定める。

2.4 SBOM フォーマット

SBOM は機械判読可能かつ相互運用可能なフォーマットを用いて作成し、共有する必要がある。これは SBOM に共通的なフォーマットを用いることで SBOM 作成の自動化サポート、組織内の管理が効率化されるほか、他組織への SBOM を共有する際の相互運用性が高まり、ソフトウェアサプライチェーンの透明性向上に寄与する為である。SBOM のフォーマットとして、以下に示す 3 つのフォーマットが代表的なフォーマットとして推奨されている[3]。

(1) SPDX（Software Package Data Exchange）

SPDXはLinux Foundationの傘下のプロジェクトによって開発されたSBOMフォーマットで、2021年9月にはISO/IEC 5962:2021として国際標準化された。SPDXフォーマットにおけるSBOMでは、SPDX Specificationにしたがって作成されたコンポーネントやライセンス、コピーライト等の情報が記載され、Tag-Value(txt)形式、RDF形式、xls形式、json形式、YAML形式、xml形式がサポートされている。

SPDXは、OSSのライセンスコンプライアンスに関する情報を効果的に扱うために開発されたフォーマットであり、ファイルのレベルまで構造化して詳細な情報を表現できる点が特徴である。また、対象となるコンポーネントについて、スニペットやファイルに留まらず、パッケージ、コンテナ、OSディストリビューションまで拡張可能である。自動処理を意図して開発されたフォーマットであり、前述のとおり、ISO/IEC 5962:2021として国際標準化されていることも大きな特徴である。

(2) CycloneDX

CycloneDXは、セキュリティに特化したSBOMフォーマットの標準を開発することを目標とし、OWASPコミュニティのプロジェクトによって開発されたSBOMフォーマットである。CycloneDXフォーマットによるSBOMでは、コンポーネントやライセンス、コピーライト等の情報が記載され、json形式、xml形式、Protocol Buffers (protobuf)形式がサポートされている。

CycloneDXの特徴として、セキュリティ管理を念頭に置いたSBOMフォーマットであることが挙げられる。2022年1月にリリースされたCycloneDX Version 1.4ではオブジェクトモデルに「Vulnerabilities」が追加、SBOMに含まれるサードパーティソフトウェアやOSSに存在する既知の脆弱性と、その脆弱性の悪用可能性を記述できるフォ

[3] 米国 NTIA：Survey of Existing SBOM Formats and Standards - Version 2021[4]

ーマットとなっている。またSPDXと同様に、ツールによる自動処理を目的としたフォーマットである。

(3) SWIDタグ（Software Identificationタグ）

SWIDタグは、組織が管理対象とするデバイスにインストールされたソフトウェアを追跡することを目標として開発された。2012年にISOで定義され、2015年にISO/IEC 19770-2:2015として更新された。SWIDタグによるSBOMでは、デバイスにインストールされたソフトウェアやソフトウェアに適用したコンポーネントのライセンスの情報や、パッチやアップデートに関する情報、脆弱性や脅威に関する情報等、セキュリティに関する情報が記載され、xml形式がサポートされている。

SWIDタグでは、デバイスにソフトウェアがインストールされるとタグと呼ばれるインストールされたソフトウェアの情報がデバイスに付与され、アンインストールされるとタグが削除される。各タグでは、タグの作成者、デバイスにインストールされるソフトウェア、他のソフトウェアへのリンクによる依存関係等の情報を提示することができ、対象とするデバイスのSBOMとして使用することが可能である。

上記のフォーマット以外にも簡易的なSBOM作成、管理を目的としたSPDX-Liteやツール独自のフォーマットが存在する。しかし、自動化、相互共有の観点から上記の3つのフォーマットが広く普及している。

2.5 SBOMに関する誤解

2.2節「SBOM導入のメリット」でも述べたようにSBOM導入によるメリットは大きいが、2024年時点で普及するために様々な課題がある。例えばSBOM導入にかかるコストの課題や技術的な課題、人材に関する課題等が考えられるが、このほかにもSBOMの効果や位置づけが適切に認知されていない課題も存在する。このような課題に対し、米国NTIAは2021年に「SBOM Myths vs. Facts」(SBOMの神話と<事実>)[4]という文書を発表し、SBOMに関する誤解と事実を明らかにした。以下に誤解と真実について記載する。

誤解例①：SBOMは攻撃者へのロードマップである。
<事実>
攻撃者はSBOMに含まれる情報を活用することができる。しかし、SBOMは「防御者のためのロードマップ」として機能するため、透明性の防御上の利点が上回る。また、攻撃者はSBOMの情報がない場合でも、攻撃可能であるため、防御上の利点を重視すべきである。

[4] SBOM Myths vs. Facts[5]

誤解例② ：SBOM だけでは有益な情報も実用的な情報も得られない。
<事実>
SBOM は、ソフトウェアのサプライヤー、利用者、運用者をサポートする。例えば、利用者がソフトウェアに対する攻撃を受けたとき、SBOM を使用することで攻撃の影響を受けているか、攻撃の影響範囲はどこかを容易に判断できる。また、SBOM に基づくコンポーネント情報があることで、ソフトウェアの透明性が向上し、管理が容易となる。

誤解例③ ：SBOM は公開する必要がある。
<事実>
SBOM は必ずしも公開する必要はない。SBOM を作成する行為は、このデータを建設的に使用できる人々と共有することとは別のものである。作成者は、自らの裁量で SBOM を宣伝および共有することができる。

誤解例④ ：SBOM は知的財産／企業秘密を暴露する。
<事実>
SBOM は含まれるソフトウェアコンポーネントの概要であり、知的財産（IP）を公開するものではない。特許およびアルゴリズムは含まれない。

誤解例⑤ ：SBOM の導入を支援するプロセスは存在しない。
<事実>
NTIA のマルチステークホルダープロセスや大統領令、概念実証などが影響し、商用およびオープンソースツールのリストが増え、プロセス化や統合が進んでいる。特定の業界はすでに 5 年以上先行しており、他業界での採用を支援している。

　また、SBOM の導入支援をサービスとして提供している企業にヒアリングしたところ、以下のような誤解と事実の例も出ている。

誤解例⑥ ：同一のフォーマットを利用することで SBOM は情報の漏れなく連携できる。
<事実>
同一の SBOM フォーマットを利用していた場合でも、SBOM 出力のジェネレータの実装が異なる場合、出力される情報に実装特有の傾向が生まれ、共有時の情報漏れやエラーが発生する場合がある。

誤解例⑦：SBOM の最小構成要素を満たしていれば、SBOM 利用に問題はない。
<事実>
利用目的によっては正しくない。2024 年 6 月時点での最小構成要素には、ライセンス情報などは含まれておらず、SBOM 利用の目的にライセンス管理が含まれていれば、必要な情報が取得できていない。導入の目的から必要な情報を整理し、SBOM で収集が必要な項目を精査することが必要である。

誤解例⑧：SBOM で脆弱性検知ができれば、パッチ適用などの脆弱性対応が容易になる。
<事実>
重要インフラ関連の環境やアプリケーションへのパッチ適用等には他システムへの影響調査やパッチ適用後の動作テストなどの考慮が必要となる。そのため SBOM の利用によって脆弱性の検知に要する工数は削減できても、パッチ適用などの脆弱性対応に必要な作業は変わらないため、一概に容易になるとは言えない。

3. SBOM 導入・運用に関する基本指針

3.1 SBOM 導入・運用の基本指針

SBOM 導入に先立ち、SBOM を作成するソフトウェアの範囲を決定し自組織の課題とそれを踏まえた SBOM 導入の目的を明確にすることが重要である。目的の具体例としては膨大なコンポーネントの把握、脆弱性管理工数の削減、正確なライセンス管理などがある。

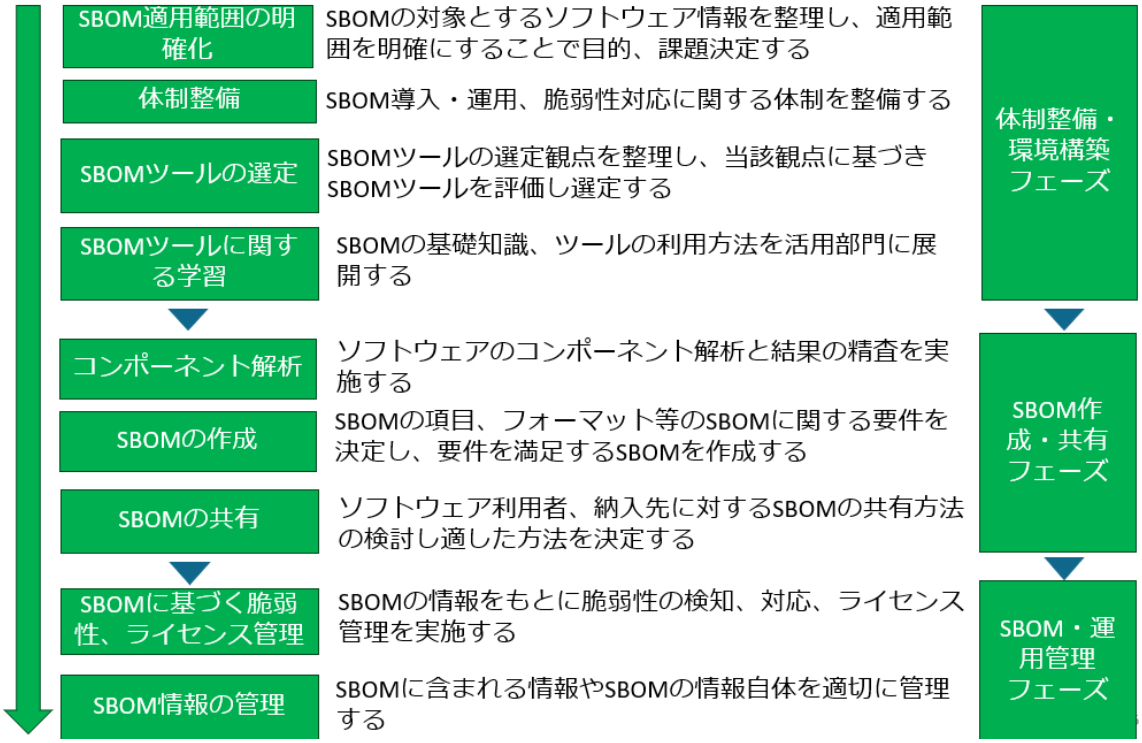
これらの目的を達成するために、脆弱性管理に強い SBOM ツールを導入するのか、SBOM 管理の容易性に優れた SBOM ツールを導入するのか、また、有償なのか無償なのかを選定し、さらにどのようなフォーマットを採用するのか、などの導入方針を決定する。

加えて、SBOM を導入・運用するための適切な体制づくりが重要である。SBOM 導入・運用するための体制づくりでは導入する部門にとどまらず、運用すると予想される PSIRT や品質管理部門、セキュリティ部門、実際にソフトウェア開発で SBOM ツールを利用する部門、そして取引先と SBOM を共有する部門を含めた体制を採用する。

このような体制において SBOM を導入することで導入後の SBOM における脆弱性管理や開発部門との情報連携、SBOM の管理をスムーズに実施することが可能となる。そのため、SBOM を導入する組織は、まず、SBOM 導入により解決したい自組織の課題、目的を整理し適切な体制構築したうえで SBOM を作成・運用管理することが重要である。

3.2 SBOM 導入・運用に関するプロセス

経済産業省の「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」[5]によると、SBOM 導入に関するプロセスは環境構築・体制整備フェーズ、SBOM 作成・共有フェーズ、SBOM 運用・管理フェーズの 3 つにわけることができる。環境構築・体制整備フェーズでは、SBOM の導入範囲や SBOM 導入の目的を明確化するとともに導入範囲、目的に合わせたツール導入などの SBOM の作成・運用に向けた環境、体制の構築をする。SBOM 作成・共有フェーズでは SBOM の作成、必要に応じて外部に SBOM を共有する方法を検討、構築する。SBOM 運用・管理フェーズでは作成した SBOM を活用し脆弱性管理やライセンス管理を行うとともに SBOM 自体を適切に管理する。各フェーズにおける実施事項を図：3.2.1 に示す。



図： 3.2.1 SBOM 導入・運用に関するプロセス

[5] 経済産業省「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.2.0」［1］

4. 環境構築・体制整備フェーズにおける実施事項

4.1 SBOM 適用範囲の明確化

SBOM 導入組織は、SBOM 導入により解決したい自社の課題と SBOM 導入の目的を踏まえ、SBOM の適用範囲を明確にする必要がある。以下表：4.1.1 に示す。なお、表：4.1.1 中の参考ページは、本ドキュメントにおける参照先を表している。

表：4.1.1 SBOM 適用範囲の明確化における実施事項

No	チェック項目	内容	参考ページ
1	SBOM 導入により解決したい自社の課題と SBOM 導入の目的を決める。	脆弱性管理やライセンス管理における自社の課題を明確にすることで目的を定める。	P18-19
2	SBOM の対象とするソフトウェアを決定する。	自社管理するシステムやソフトウェアに関する情報を整理し、SBOM の対象とするソフトウェアを決定する。	P19
3	SBOM の対象ソフトウェアの開発言語、コンポーネント形態などのソフトウェアに関する情報を明確にする。	対象ソフトウェアの情報を整理し、SBOM ツール選定時の検討項目として活用する。	P19-20
4	対象ソフトウェアの正確な構成図を作成し、SBOM 適用の対象を可視化する	対象ソフトウェアの構成図を作成することで、構成図をベースにリスク管理の範囲を明確にする。	P20
5	SBOM 導入に向けた組織内の制約（コストの制約、環境の制約等）を明確化する。	コストや環境の制約をはじめとする組織内の制約を整理することで以降のフェーズをスムーズに進める。	P20-21
6	整理した情報に基づき、SBOM 適用範囲（5W1H）を明確化する。	整理した情報を改めて 5W1H の形にまとめ SBOM の適用範囲を明確にすることで以降の導入フェーズに役立てる。	P21-22

SBOM を導入する組織は SBOM を導入する目的･解決する自社の課題を明確にすることでどのようなツールを選定するのか、どのような体制で導入プロジェクトに臨むのか等、今後の導入フェーズをスムーズに実施することができる。また、組織によっては規制により SBOM 導入を検討している場合もある。米国では、政府調達対象となるソフトウェアに対して SBOM の提供を求める大統領令が発令された。EU のサイバーレジリエンス法では、EU 市場におけるデジタル製品に SBOM による管理が義務付けられた。これに伴い日本でも経済産業省が「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」で SBOM に関する検討が実施され、今後規制として SBOM が要求される可能性がある。そのような組織でも SBOM 導入や SBOM の効率的な運用に効果

的であるため、SBOM 導入の目的・課題の明確化は必須である。

2.2 節「SBOM 導入のメリット」で述べた通り、SBOM を導入するメリットは脆弱性管理の強化とライセンス管理の強化の 2 つに大別できる。これらのメリットを踏まえ、まずは自社における課題を明確にする。

まず脆弱性管理の強化では、脆弱性に関するリスク分析を実施することで、資産の構成管理ができていない、脆弱性対応のリソース不足、脆弱性発見能力不足などの課題を明確にする。SBOM 導入部門が脆弱性へのリスク分析ができない場合は、脆弱性に関する業務を実施している部門に依頼し課題を明確にする。次にライセンス管理強化では、過去に自社で起きたライセンスに関するトラブルや過去の他社事例などから自社におけるライセンスに関する課題を明確にする。これらの課題を SBOM 導入により解決することを目的と定める。

SBOM を導入するにあたり、SBOM を作成し運用する対象を決める。SBOM をどのシステム、ソフトウェアを対象として適用するのか明確にすることで導入・運用体制やツール選定、作成フェーズでの検討項目が決まる。対象とするシステムやソフトウェアを決めるにあたり、自社のシステムやソフトウェアに関する情報を整理する必要がある。以下のような項目を整理することで SBOM を導入するシステムやソフトウェアの優先度を決定することができる。

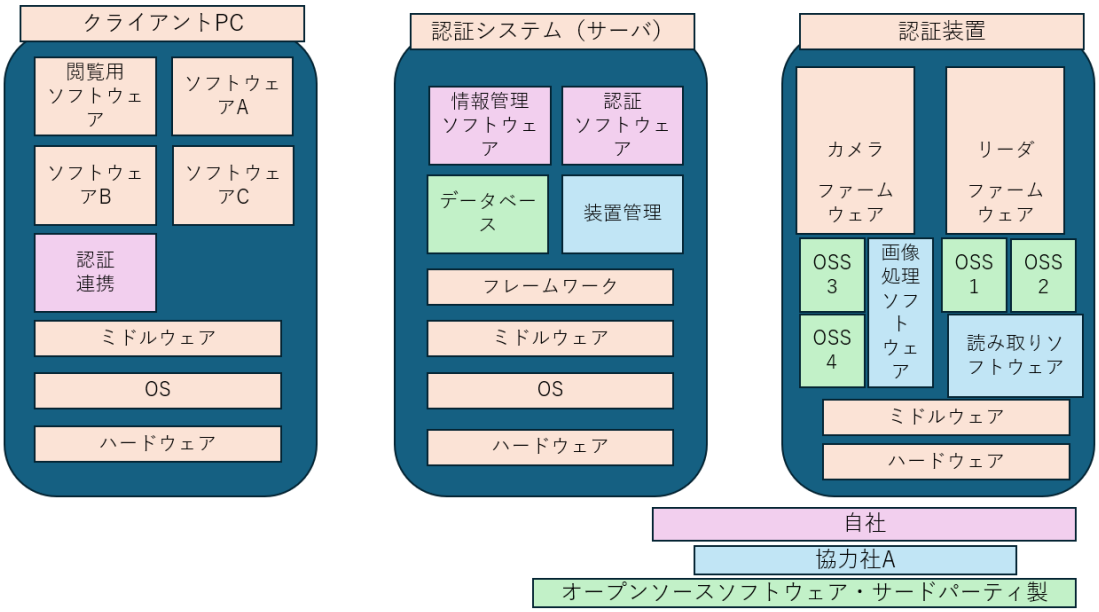
- 規制や要求：対象ソフトウェアの業界の規制や海外の法令 等
- 契約形態： 開発委託、製品販売 等
- コンポーネント情報の提供：提供無、無償提供、要望された場合に提供可能 等
- 第三者コンポーネントの申告：すべての OSS について申告、ライセンスを踏まえて一部の OSS について報告 等
- 脆弱性の通知・修正：修正すべきと判断された脆弱性に関してのみ通知 等
- 納品形態：バイナリパッケージ、機器組み込み、ライセンス情報 (SaaS 等)、実行モジュール 等
- 損害賠償責任
- 知的財産権の帰属：自社に帰属、納入先に帰属、納入基に帰属 等
- コード改変の有無：サードパーティから提供されたソフトウェアをそのまま使用している、自社にて改変して使用している 等

SBOM の対象となるソフトウェアを決定した後にそのソフトウェアに関する情報や、SBOM 導入に関する社内の制約を整理する。対象ソフトウェアに関する情報は以下のような項目に対して実施する。

- 開発言語：Java、Python、Go、Rust、C、C++ 等
- コンポーネントの形態：ライブラリ、アプリケーション、ミドルウェア、データベース サービス 等
- 開発環境ツール：Visual Studio、Eclipse、Android Studio、Xcode 等
- ビルドツール：Jenkins、Circle CI、Github Actions、Gradle、Maven 等
- 構成管理ツール：Github、Gitlab、Team Foundation Server、Ansible 等
- 取り扱うデータ形式：ソースコード、パッケージ、コンテナ、バイナリデータ 等
- 動作環境：OS、CPU アーキテクチャ 等

導入部門のみでこれらの情報を得られない場合は、対象ソフトウェアの開発部門に依頼し情報を整理する。SBOM ツールによっては自組織で利用している開発言語やビルドツール、データ形式が対応していない場合もある。これらの情報を整理することで SBOM ツール選定をスムーズに実施することができる。

また、図:4.1.1 のように対象ソフトウェアの構成図を作成し可視化することが望ましい。ソフトウェアは自社ですべて開発していないこともあり、一部機能を取引契約のあるサプライヤーに委託して開発している場合や、取引契約のないサプライヤーの既存ソフトウェアをコンポーネントにしている場合がある。構成図を可視化することで構成図をベースにリスク管理の範囲を明確にすることができる。



図： 4.1.1 可視化したソフトウェア構成図

次に対象ソフトウェアの SBOM 導入に向けた組織内の制約を確認する。組織内の制約として最も代表的なものはコストである。SBOM の導入・運用にかけることのできるコスト

を正確に把握する必要がある。また、構築する SBOM ツールをインターネットに接続可能か、クラウドサービスの SBOM ツールを利用可能かなどの環境的制約も確認する。これらの制約が厳しい場合、SBOM の活用が限定的になる可能性がある。そのため、あらかじめ組織内の制約を確認、整理する。

以上の情報を整理したうえで改めて SBOM の適用範囲を整理する。経済産業省より公開されている「SBOM（Software Bill of Materials）の導入に関する手引き」[6]に記載されている 5W1H の方法で整理することで今後のフェーズで活用しやすい形で情報を整理することができる。以下、表：4.1.2 に 5W1H による SBOM の適用範囲の整理方法を示す。

表：4.1.2 5W1H による SBOM 適用範囲

観点	主な適用項目（選択肢）
SBOM の作成主体（Who）	● 自組織で SBOM を作成する ● 取引契約のあるサプライヤーにて SBOM を作成する ● 取引契約のないサプライヤー（OSS コミュニティ等）にて SBOM を作成する
SBOM の作成タイミング（When）	● 製品計画又は開発計画時 ● プログラム開発時 ● ソフトウェアビルド時 ● ソフトウェア納入時 ● コンポーネントのバージョンアップ時
SBOM の活用主体（Who）	● ソフトウェア利用者 ● 最終製品ベンダー ● 開発ベンダー ● 最終製品ユーザー
SBOM の対象とするコンポーネントの範囲（What、Where）	● 開発主体が直接利用するコンポーネントのみを対象とする ● 既製品等開発委託契約のないコンポーネントから再帰的に利用されるコンポーネントも含めて対象とする
SBOM の作成手段（How）	● 構成管理情報を踏まえて手動で SBOM を作成する ● SBOM ツールを用いて自動で SBOM を作成する ● 一部は構成管理情報を踏まえて手動で SBOM を作成、一部は SBOM ツールを用いて自動で SBOM を作成等、手動作成と自動作成を併用する

[6] 経済産業省「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.2.0」［1］

SBOM の活用範囲（Why）	● 脆弱性管理 ● ライセンス管理 ● 開発生産性の向上 ● 資産管理、トレーサビリティ ● 利用者や納入先に対するコンポーネントに関する情報共有
SBOM のフォーマット・項目（What）	● 標準フォーマット（SPDX、CycloneDX、SWID タグ） ● 米国大統領令におけるデータフィールドの最小要素 ● 規制・要求事項や業界の慣行として使用される独自のフォーマット

SBOM 適用範囲は、これらの項目の組み合わせによって定まり、どの項目を選択するかにより、導入に要するコストが異なる。また、1 つの観点に対し、複数の適用項目を選択する場合もある。

4.2 SBOM 導入体制の整備

SBOM 導入にあたり、課題・目的に合わせたプロジェクト体制づくりが重要である。以下、表：4.2.1 に導入体制の整備における実施事項を示す。

表：4.2.1 SBOM 導入体制の整備における実施事項

No	チェック項目	内容	参考ページ
1	自組織における SBOM 導入・運用に係る役割と部門を精査する。	SBOM 導入・運用の役割を精査し、その役割を持つ自組織内の部門を明確にする。	P22-23
2	導入主管部門に加え、精査した部門を加えたプロジェクト体制を整備する。	自組織における SBOM 導入主管部門、SBOM 作成する部門、脆弱性、ライセンスを管理する部門、SBOM 自体を管理する部門を明確にし、SBOM 導入体制を整備する。	P22-23

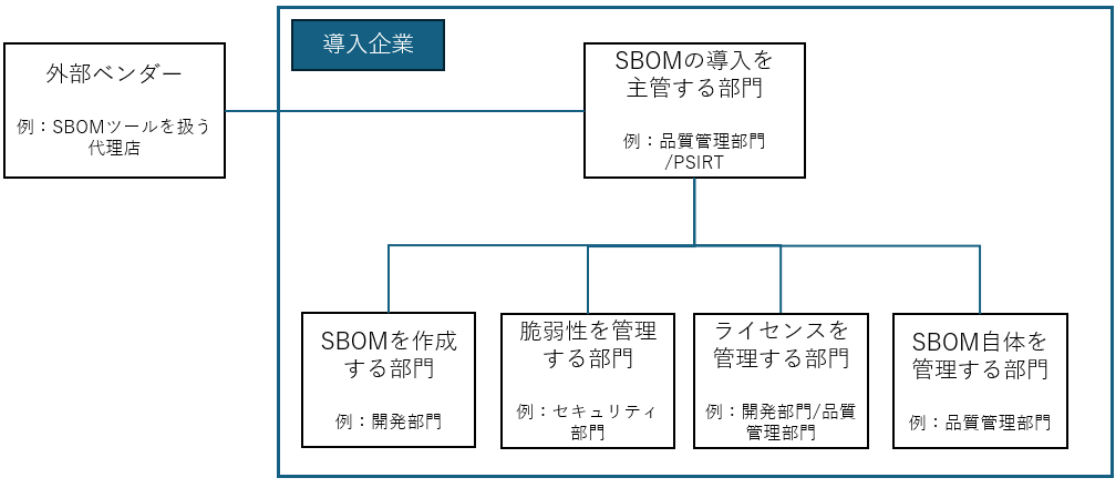
SBOM 導入のプロジェクト体制づくりにおいて、3.1 節「SBOM 導入・運用の基本指針」で説明したとおり、SBOM を活用し脆弱性対応、ライセンス管理、SBOM 情報の管理をする部門を含めた体制で導入を進めることで導入後の運用にスムーズに移行することができる。

そのためにまずは、SBOM の適用範囲に関係する役割を持つ自社の組織を明確にする。以下、表：4.2.2 に SBOM の導入・運用プロセスに関係する役割と関連する部門およびその実施内容を紹介する。

表：4.2.2 SBOM 導入・運用に関連する役割と部門

役割	部門	実施内容
SBOM の導入を主管する	品質管理部門/PSIRT	・企業内での SBOM 導入推進 ・外部ベンダーとの連携
SBOM を作成する	開発部門	・作成したアプリケーションのソースコードを SBOM ツールで解析し、SBOM フォーマットで出力する
脆弱性を管理する	セキュリティ部門	・出力された SBOM と脆弱性データベースを照合し、脆弱性を検知し、対応方針を決める。
ライセンスを管理する	開発部門/品質管理部門	・出力された SBOM コンポーネント情報からライセンス要件を確認する。 ・ライセンス違反のリスク評価を実施し、適切に遵守されているかを定期的に監査する。
SBOM 自体を管理する	品質管理部門	・追加開発や脆弱性対応による SBOM の更新に応じて、バージョンを適切に管理する。

また、SBOM の導入後に効果的な運用を実現するには、部門を超えた協力と連携が不可欠である。上記で説明した役割を持つステークホルダー間で密に連携をとることによって組織全体のセキュリティと製品の品質が大幅に向上する。そのために、導入時に導入の主管となる部門に加え実際に SBOM を運用していく部門を含めた体制でプロジェクトメンバーを構成する。SBOM の導入企業と外部ベンダーを含めた関係性を以下の図：4.2.1 に示す。



図： 4.2.1 SBOM 導入企業と外部ベンダーの関係性

※図：4.2.1 に表記されていない部門の協力が必要とされる可能性がある。

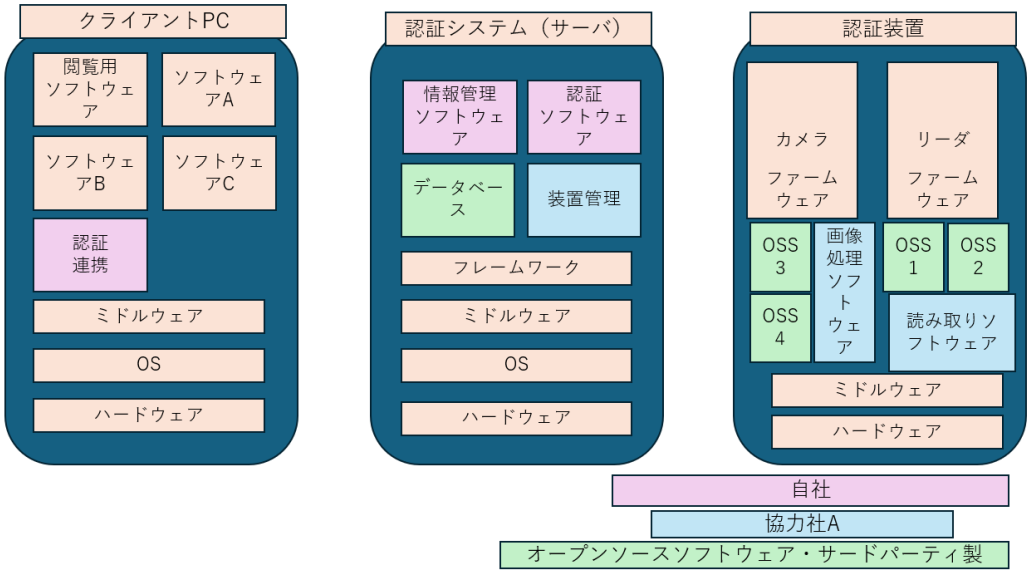
4.3　SBOM ツールの選定

本節では、4.1 節「SBOM 適用範囲の明確化」で決定した自組織における SBOM の適用範囲や課題・目的を基に、SBOM ツールを選定する方法を説明する。以下、表：4.3.1 に SBOM ツール選定における実施事項を示す。

表：4.3.1 SBOM ツール選定における実施事項

No	チェック項目	内容	参考ページ
1	ツール選定の基準を決める。	設定した課題や目的、自組織の制約、開発環境をもとに SBOM ツールの選定基準を定める。	P24-29
2	選定基準を基に複数のツールを比較する。	星取り表を作成し、自社の選定基準に合致する SBOM ツールを選定する。	P28-89
3	SBOM ツールの PoC を実施する。	選定された SBOM ツールを試験的に導入し、当初設定した課題や目的を解決・満たす性能を有するか確認する。	P29

以降では、SBOM ツールを選定する手順について、仮想的な企業を例として用いて説明する。まず、この仮想企業はぜい弱性管理を目的として、SBOM ツール導入することとする。次に、仮想企業のシステムは図：4.3.1 に示すものとする。そして、SBOM を作成する対象のソフトウェアは同図の認証連携、情報管理ソフトウェア、認証ソフトウェアとする。



図：　4.3.1 仮想企業のシステム構成図

次に、仮想企業の制約事項として以下の①～④を想定する。

- 制約①：
  機密性の高いソースコードが漏洩するリスクを踏まえ、情報機密性が担保できる運用を確保すること。

- 制約②：
  対象ソフトウェアに関する制約
  - 開発言語：Java
  - コンポーネントの形態：アプリケーション
  - ビルドツール：Maven
  - 構成管理ツール：Github
  - 取り扱うデータ形式：ソースコード

- 制約③：
  SBOM 運用開始期日が近いため、SBOM 導入担当者の学習が不要または短期間で完了できることを必須とする。

- 制約④：
  SBOM の導入・運用は GUI での操作を前提とする。CUI 版の SBOM ツールを採用する場合はベンダーからのサポートを必須とする。

では、ここまでに整理した導入目的と制約を基に、SBOM ツールの選定を進めていく。

対象ソフトウェアの開発言語や組織内の制約を考慮した SBOM ツールの選定基準を整理

- 選定基準①：SBOM ツールの機能要件
  - 提供形態
    SBOM ツールの提供形態には組織内にツール環境を構築するオンプレミス版とツールベンダーの環境に構築したものを利用するクラウド版がある。クラウド版では、オンプレミス版と比較して初期導入にかかる構築コストを削減できるが、機密性の高い自社のソースコードに関するシグネチャ情報が社外に送信されることを禁止する組織の制約がないか確認する必要がある。一方、オンプレミス版では、機密性の高いソースコードは自社の責任で管理できるが、ツール料金のほかにサーバの維持管理費用が発生する。制約①の機密情報保護の観点から、オンプレミス版を採用する。

- 解析可能なデータ形式

  SBOM ツールは、コンポーネント解析時に読込可能なデータ形式に条件がある。ファイル形式（拡張子別の対応可否）、対応するパッケージマネージャーの種類、ソフトウェアが動作可能な OS・CPU アーキテクチャ等、解析に使用するデータの形式を整理することが望まれる。

- 対応フォーマット

  SBOM ツールによって、特定のフォーマットの SBOM のみインポート可能／作成可能な場合がある。SBOM のインポートおよび SBOM のエクスポート（SBOM の作成）に関しては、運用を予定しているフォーマットに適応しているか、SBOM 提供先や顧客に確認する必要がある。現時点で普及している SPDX と CycloneDX に対応していれば広く対応が可能である。

- コンポーネントの解析方法

  ソフトウェアに含まれるコンポーネントの解析方法は、依存関係検出、コードマッチング、文字列検出の 3 つに大別できる。コードマッチングは、OSS データベースとのマッチングによって OSS を検出する方法であり、完全一致のコードマッチングのほか、スニペットマッチングと呼ばれる部分一致のコードマッチング方法も存在する。また、バイナリパターンによるマッチングを行う方法も存在する。依存関係検出は、パッケージマネージャーで取得する直接的・間接的な OSS を検出する方法であり、誤検出の発生可能性は低い。そして、文字列検出は、ソフトウェアのライセンス文字列を解析して、適用されているライセンスを検出する方法である。

- 解析可能な情報

  SBOM ツールによって、解析できるコンポーネントの情報が異なる。有償ツールの多くはコンポーネントに関する脆弱性情報やライセンス情報を自動で解析できるほか、脆弱性情報の解析に特化したツールや、ライセンス情報の解析に特化したツールも存在する。SBOM ツールの適用範囲、目的および制約②より、その SBOM ツールを導入することで、本来期待していた目的を達成するための情報が取得できるのかを確認する必要がある。

- 日本語対応

  SBOM ツールは海外で開発されたツールが多く、取扱説明書が英語のみで提供される場合や、SBOM ツール自体も日本語対応していない場合がある。制約③より、十分な学習時間が取れないため SBOM 担当者の母国語である日本

語に対応した SBOM ツールを採用する。

- サポート体制

  SBOM ツールの導入や運用に関する問い合わせが可能な SBOM ツールが存在する。また、無償ツールであっても、問い合わせが可能なサポートサービスを提供している企業が存在する。制約③より、SBOM ツール導入担当者は十分な学習時間が取れないため、サポート体制のあるツールを採用する。

- ユーザーインタフェース

  CLI のみ提供している場合と、GUI も提供している場合がある。CLI とはユーザーがテキストベースのコマンドを入力してコンピュータと対話する方法であり、慣れていない人にとっては扱いに慣れるために時間がかかる。GUI を提供している場合、直感的な SBOM の作成や出力結果の可視化が可能となる。

  制約④により直感的な操作が可能である GUI を提供している SBOM ツールでなければならない。

- 選定基準②：脆弱性管理機能の機能要件
  - 脆弱性管理機能の利便性

    SBOM ツールによって対応している機能が異なる。想定企業における SBOM の導入目的は「SBOM を活用した脆弱性管理」であるため、以下のような機能を持つ SBOM ツールを採用することで、脆弱性管理能力の強化を図ることができる。
    - 脆弱性情報の自動マッチング機能
    - 依存関係や脆弱性情報等の可視化機能
    - 脆弱性情報・ライセンス情報の自動追跡機能
    - 新たな脆弱性が検出された際のアラート機能
    - アドバイザリー情報の自動レポート機能

  - 脆弱性検知の精度・信頼性

    上述の「脆弱性管理機能の利便性」における OSS の検出や脆弱性情報・ライセンス情報のマッチングに当たって、どの程度の誤検出・検出漏れが生じうるかは一つの重要な指標である。加えて、新たな脆弱性が見つかった際に、どれほど迅速にツールに反映されるかという点も重要となるため、ベンダーや SBOM ツールを扱う代理店に質問し、状況を確認するとともに自社環境で入念に検証する。

- 他ツールとの連携

  SBOM ツールには開発環境、構成管理ツール、ビルドツール、ソフトウェアバージョン管理ツール、コミュニケーションツール等と連携する機能を持つツールがある。構成管理ツールやバージョン管理ツールとの連携により SBOM 作成の自動化やコミュニケーションツールへの通知等、ソフトウェア開発ライフサイクル全体の効率化を図る目的では、既に組織内で活用しているツールとの連携ができることが望ましい。そのため、どのようなツールとの連携が必要であるかを整理・確認する必要がある。

- 選定基準③：適切なコスト
  - 費用の妥当性検討　（コスト）

    有償の SBOM ツールの場合、ツールのライセンス費用が必要となり、料金体系はそれぞれ異なる。導入企業の環境や人材の能力に見合った製品を選定する。

  - 運用に必要な人的コスト

    開発環境と連携しバックグラウンドで自動的に解析が行われるような SBOM ツールやポリシー機能、ライセンス等の補足情報が充実している SBOM ツールも存在している。SBOM 関連の業務に割り当てられる人員が不足している場合には、ツールによる自動化や追加作業が少なくなるようツールを選定することが望ましい。

複数の SBOM ツールを評価し、適切なツールを選定

市場で利用可能な SBOM ツールをリサーチし、先ほど整理した選定基準に合致する候補ツールのリストを作成する。星取表を作成し、候補ツールの評価結果を一覧化することで客観的かつ体系的にツールを比較し、最適な選定が可能になる。表：4.3.2 に作成する星取表の例を示す。

表：4.3.2 SBOM ツール選定のための星取表

選定基準	SBOM ツール 候補 A	SBOM ツール 候補 B	・・・
情報の機密性確保 （制約①対応のため）	○ （オンプレ版）	△ （クラウド版）	・・・
サポート体制・日本語対応 （制約②対応のため）	○	△ （24h 対応×）	
コスト	△ （ランニングコスト高）	○	
・・・			

コストについては、SBOM ツールの有償・無償だけで判断せず、環境整備・学習に必要な情報が提供されるか、導入・運用、エラー発生時の原因究明・対応に要する工数、各種開発ツールとの連携可能性等も考慮する必要がある。実際に筆者が SBOM ツールを導入する際に感じた有償・無償ツールの違いについては「コラム：有償無償ツールの違い」に記載する。

SBOM ツールを評価して候補を選定した後は、PoC が実施できる場合には操作感を確認し、期待通りに動作するか、コンポーネント解析の精度が十分か、自社環境へ適合しているかを検証することが重要である。スペック上は問題ない場合でも自社環境にうまく適合しないなど、一見うまくいきそうに見えても実際には問題がある場合があるため、本格的に導入する前に期待通りに動作するかを確認する必要がある。スムーズに SBOM を活用できるよう、導入時は特に慎重に SBOM ツールを選定すべきである。SBOM ツールを扱う代理店や導入ベンダーに相談し、導入実績に基づく専門知識を活用するのも一つの方法である。

コラム：有償ツールと無償ツールの違い

SBOM ツール選定する際に、コストは重要な観点の一つである。ここでは、ライセンスコストが SBOM ツールの機能や環境構築コスト、学習コスト、ランニングコストにどの様な影響を与えるのか、SBOM の作成と管理に着目し、筆者の私見を交えながら記す。

まず、有償ツールと無償ツールの絶対的な差にベンダーサポートの有無がある。有償ツールであれば、導入のみならず使用方法や不具合に対してベンダーからサポートを受けることができ、問題を早期解決することができる。しかし、無償ツールの導入はサポートを受けず対応する必要がある場合が多く、使用方法や不具合は公式ドキュメントやフォーラムを参照することになる。この公式ドキュメントやフォーラムは有志によって運営されている事が多いため、ユーザーが少ないツールの場合は情報収集そのものが難しいことがあり、些細な対応であっても工数が増え、導入・運用コストが増加する要因になる。加えて、無償ツールは先述の通り有志により開発されていることが多いため、終了予告や終了宣言のないまま開発・運用が停止するリスクがあることを理解する必要がある。

SBOM 作成機能については有償ツールと無償ツールで精度の違いはあるが、SBOM 作成機能の本質に大きな差はない印象である。有償ツールは GUI 操作でソフトウェアを分析し SBOM を作成する機能を有する物が多いが、無償ツールは CLI 操作でソフトウェアを分析し SBOM を作成する機能を有する物が多く、結果として有償ツールの方がユーザビリティに優れており、ツールの学習コストも低くなる。

SBOM 管理機能については有償ツールと無償ツールで大きな差がある。有償ツールの場合はツール内で SBOM を作成・管理・共有することを前提として設計されている印象があり、特に管理機能は複数プロジェクトを横断して管理することができるユーザーインタフェースになっていることが多い。これに対して無償ツールは SBOM の作成・管理・共有機能を個別のツールで、特定の OS に対してのみ提供している事が多く、SBOM 管理機能を構築した結果、有償ツールと比較した際に複雑な構成になりがちである。

脆弱性管理機能については有償ツールと無償ツールにどれだけの差があるのか評価することは難しい。有償ツールは JVN や NVD の様な有名な脆弱性データベースの情報に加えて、付加価値としてツール提供元ベンダー独自の脆弱性情報データベースを提供している。この付加価値は無償ツールに対して優位であるように思えるが、無償ツールであってもセキュリティベンダーが提供するデータベースを参照している場合もあり、単純に脆弱性情報を管理する性能を比較することは難しい。しかし、有償として提供しているため一定の品質が担保されており、脆弱性情報の更新が止まる等のリスクは低い。脆弱性は悪用されるとビジネスに直接影響を与えるため一定以上の品質の保証は必須といってよい。

総括すると、有償ツールは金銭負担と引き換えにサポート等で省力化や構成の効率化、一定以上の品質保証が実現でき、無償ツールはツールの活用方法の調査や効率的な環境構築の努力と引き換えに金銭負担の軽減を実現できる。組織が管理するソフトウェアの規模感や重要度によってツールを選択する必要がある。

4.4　SBOM ツールの導入・設定

本節では、SBOM ツールの導入及び設計について説明する。以下、表：4.4.1 に SBOM ツールの導入・設計における実施事項を示す。

表：4.4.1 SBOM ツールの導入・設定における実施事項

No	チェック項目	内容	参考ページ
1	導入する SBOM ツールの導入に必要な環境を整理し、構築する。	SBOM ツールを導入するサーバの用意や、OS インストールなどの環境を構築する。	P31-32
2	ツールの取り扱い説明書や README ファイルを確認し、ツールを導入・設定をする。	構築した環境にツール取り扱い説明書、README ファイルを確認し、初期設定を実施する。必要に応じてサポートなども活用する。	P31-32

4.3 節「SBOM ツールの選定」で選定したツールはツールごとに導入方法や環境が異なる。クラウドで動作しているサービスであればツールを利用する環境設定は少ないが、オンプレミスでサーバを構築する場合、インターネット接続や特定やプログラミング言語が実行できる環境が整っていること、特定の OS がインストールされていること等の条件がある。

ツールを導入する前にまずはどのような環境が求められているのか確認し SBOM ツール導入の準備を実施する。導入環境を構築した後、SBOM ツールを導入し SBOM 作成に向けた初期設定を実施する。SBOM ツールの導入・設定では以下の情報を確認し進めていく。

公式のドキュメント・リファレンス・README を確認する

公式のドキュメント・リファレンス・README はツールの開発者によって提供される公式なドキュメントであり、ツールのインストール手順、設定方法、使用方法、トラブルシューティング情報が詳細に記載されている。公式のドキュメント・リファレンスを確認することで、ツールの基本的な使い方を理解し、一般的な問題を自力で解決することができる。

導入ベンダーサポートの活用

有償ツールの場合は、販売代理店やツールベンダーのサポートを受けることができる場合がある。自社の能力やスケジュール、コストを考慮してサポートを受けることでより効率的に導入・設定を実施することができる。

コミュニティサイトに質問する

オープンソースツールの場合、活発なユーザーコミュニティが存在することが多く、フォーラムやディスカッションボードで質問を投稿することで、他のユーザーや開発者から有

益なアドバイスを得ることができる。特に、README を読んでも解決できない問題や、特定の環境に依存する問題については、コミュニティの知識が役立つ。

4.5　SBOM ツールの学習

本節では、導入した SBOM ツールを効果的に利用するために必要な項目を説明する。以下、表：4.5.1 に SBOM ツールの学習における実施事項を示す。

表：4.5.1 SBOM ツール学習における実施事項

No	チェック項目	内容	参考ページ
1	導入した SBOM ツールの手順書を作成する。	導入した SBOM ツールの手順書を作成し、開発部門の教育に使用する。	P32
2	自組織の教育に SBOM に関する内容を盛り込む。	開発部門、セキュリティ部門の教育として SBOM に関する内容を盛り込むことで継続的な教育を実施する。	P32

SBOM ツールを効果的に利用するためには、SBOM を運用する開発部門やセキュリティ部門への適切な教育、引継ぎが不可欠である。教育、引継ぎが不十分な場合、SBOM 作成や管理が適切に実施できない可能性がある。不正確な SBOM が生成された場合、それに基づく脆弱性管理が適切に実施できないため、自社のみならず、SBOM を提供したサプライチェーンにも影響が及ぶ。したがって、SBOM ツールの効果的な利用のためには、従業員への徹底した教育と継続的なトレーニングが重要である。例えば以下のような方法が考えられる。

- SBOM の基礎知識の教育
  SBOM の概要とその重要性、構成要素について説明する。
- 導入・運用の手順書の作成
  導入・運用を行うための手順書を作成し、関係する従業員へ共有する。
- 利用者に対する継続的な教育
  定期的な勉強会や E ラーニングに SBOM に関する内容を盛り込み SBOM に関する知識、や運用方法について学習する。

いずれの教育・学習においても社内リソースにより、内製教育するか、外部教育を活用するか検討する必要がある。

5. SBOM 作成・共有フェーズにおける実施事項

5.1　コンポーネントの解析

本節では、SBOM の作成・共有フェーズにおけるコンポーネントの解析について説明する。以下、表：5.1.1 にコンポーネント解析における実施事項を示す。

表：5.1.1 コンポーネント解析における実施事項

No	チェック項目	内容	参考ページ
1	SBOM ツールを用いて対象ソフトウェアのコンポーネント解析を実施する	対象ソフトウェアのオブジェクトを SBOM ツールのスキャン方法に乗っ取りスキャンを実施する	P33-35
2	対象ソフトウェアの精査範囲を決める	対応工数とソフトウェアの重要度のトレードオフを考慮して精査対象範囲を決定する	P35-37
3	SBOM ツールが検知したコンポーネントを精査する	検知されたコンポーネントに抜け漏れや過検知がないか精査を行う。	P37-38

経済産業省の実証実験では、SBOM ツールを用いることで、手動で作成する場合と比較して効率的にコンポーネント解析、SBOM の作成ができることを確認している。医療機器分野の歯科用 CT を対象とした実証実験では手動と比較して 99%以上の工数が削減できる結果が出ている[7]。手動でのコンポーネント解析、SBOM 作成はソフトウェアの規模が大きくなるほど困難であり、小規模のソフトウェアであっても依存関係にある OSS が非常に多く SBOM ツールを用いることが現実的である。そのため本章ではコンポーネント解析、SBOM 作成は SBOM ツールを用いることを前提とする。

[7] 経済産業省「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.2.0」［1］

SBOM ツールを用いたコンポーネント解析手法は依存関係検出、コードマッチング、文字列検出の 3 つの手法がある。

- 依存関係検出
  依存関係検出は Maven や npm、pip などのパッケージマネージャーやビルドツールが提供するメタデータを解析して依存関係を特定する手法である。メタデータとは package.json、requirements.txt、pom.xml などのパッケージマネージャー、ビルドツールの設定ファイルである。これらに記載されている依存関係を解析することで直接的、間接的な依存関係を検出することが可能である。利点としては包括的な依存関係の情報を検出することができ、誤検出が少ない。

- コードマッチング
  ソースコードを解析して特定のシグネチャやハッシュ値を既知のライブラリやモジュールのシグネチャを持つデータベースとマッチングさせることでコンポーネントを検出する。利点としては、ソースコードのコピー&ペーストなどのパッケージマネージャーやビルドツールにない依存関係も検出できる。しかし、依存関係検出と比較して誤検出が発生する可能性がある。

- 文字列検出
  ソースコード内に特定のライブラリ名やバージョン情報などの文字列を検出してコンポーネントを検出する。コードマッチングと同様にコピー&ペーストやパッケージマネージャーやビルドツールにない依存関係を検出できる。しかし、依存関係検出と比較して誤検出が発生する可能性がある。

SBOM を作成するためにはまず、ツールを用いてソフトウェアをスキャンし、利用しているコンポーネントを洗い出す。SBOM ツールのコンポーネント解析は必ずしも完全ではなく経済産業省の実証実験では以下のような事象が確認された[8]。

- シンボリックリンクやランタイムライブラリ等、実態が SBOM ツールのスキャン対象に含まれないコンポーネントが検出されなかった。
- 最上位のコンポーネントの検出結果と比較して、下位のコンポーネントに関する検出漏れ率が高かった。ただし、最上位のコンポーネントが検出されずに下位のコンポーネントのみが検出されるケースもあり、必ずしもコンポーネントの階層で検出率が変わるわけではない。
- 特定分野でのみ利用されている制御に関するコンポーネントが検出されなかった。
- バージョン情報が誤って検出されたコンポーネントが複数存在した。
- SBOM ツールにおけるコンポーネント解析方法によって、出力結果が異なった。バイナリファイルのみを持ったバイナリスキャンの結果について、通常スキャンで検出されたコンポーネント数と比較して 1 割程度しか検出されなかった。
- コンポーネントを解析する環境によって、解析結果が異なった。開発環境でスキャンを行った場合、実際に製品には使用されないアンインストールパッケージも検出された。
- SBOM ツールでのデータベースに存在しない OSS は検出できず、SBOM ツールのコンソールから手動でコンポーネントに関する情報を追加する等、解析結果の調整が必要になった。
- SBOM ツールのリポジトリや設定により、SBOM 中のコンポーネントの構成情報が異なり、SBOM ツールによって作成された SBOM におけるコンポーネントの関係性が、実際のソフトウェアの構成と異なるケースがあった。
- SBOM ツールで検出されたコンポーネントとパッケージマネージャーで抽出したコンポーネントが一致しないケースがあった。

SBOM ツールを利用する上でこれらの検出漏れや過検知のリスクが存在することを認識しておく必要がある。手動でコンポーネント解析結果を精査することで、SBOM の精度が向上し、これらのリスクを低減することができる。しかし、コンポーネント解析結果の精査には膨大な工数がかかり、運用負荷が増大する。実施する場合には自社の SBOM 運用が負荷に耐えられるか考慮したうえで実施することを推奨する。

---

[8] 経済産業省「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.2.0」［1］

コンポーネント解析結果の精査には、パッケージマネージャーやソフトウェアの詳細設計書などを組み合わせて実施する。コンポーネント解析結果の精査は手動で行う必要があるが、OSS の利用がほぼ不可欠な現代ではコンポーネントの数が膨大であり、規模の大きいソフトウェアでは数千ものコンポーネントを利用している場合もある。コンポーネントのすべてを手動で精査するには膨大な対応工数がかかるため、対応工数とソフトウェアの重要度（セキュリティ要件）とのトレードオフを考慮したうえで、納入先と協議しどの階層まで精査するかを決定する。以下に検討項目の例を示す。

- ソフトウェアの重要度
  対象のソフトウェアがどのようなシステムに導入されるか等のセキュリティ要件によって、どの程度の層まで精査するべきか検討する。
  （例）本ソフトウェアは認証システムに使用するソフトウェアのため、求められるセキュリティ要件のレベルが高い。そのため、直接部品の精査に加え間接部品の一部まで精査を実施する。

- セキュリティリスク
  そのソフトウェアが導入されるシステムが攻撃されやすい場所に配置される等、セキュリティリスクに応じてどの程度まで精査するべきか検討する。
  （例）本ソフトウェアは外部公開されインターネットよりアクセスが可能であるため攻撃されるリスクが高い。そのため、直接部品の精査に加え間接部品の一部まで精査を実施する。
- 依存関係の複雑さ
  大規模で依存関係が複雑なソフトウェアに関しては精査に大きな対応コストが生じるため、実現可能性を考慮した精査を検討する。
  （例）本ソフトウェアでは直接利用部品は開発チームが自覚的にコンポーネントを利用するため、パッケージマネージャーを用いてコンポーネントの突き合せが可能であるため精査を実施する。一方で間接利用部品については効率的な確認方法がなく、精査を実施するには膨大な対応コストが生じるためフィージビリティを考慮し実施しない。

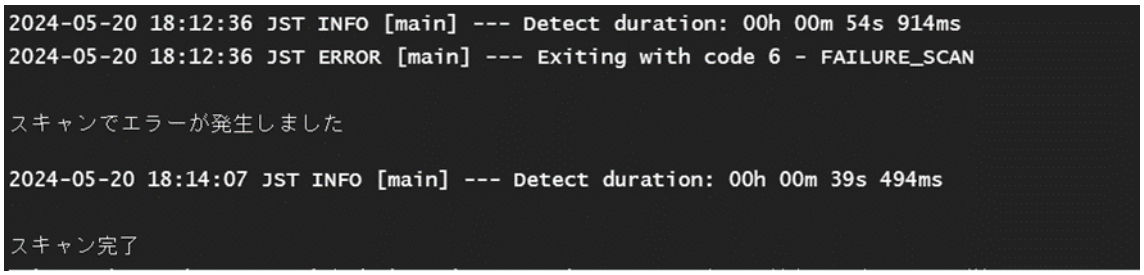
- 提供元の観点
  ソフトウェア開発におけるコードの提供元は自社、委託先サプライヤー、サードパーティサプライヤーの３つに分類できる。それぞれの提供元から得られる情報によってコンポーネントの精査の難易度が異なるため、提供元に応じた精査を検討する。
  （例）コード提供元が自社開発、委託先サプライヤーに関してはパッケージマネージャーにより依存関係が確認できるため、直接部品および間接部品の一部の精査を実施する。サードパーティサプライヤーに関しては依存関係の情報が不完全であり、精査が困

難なため精査は直接部品に限定する。

- 更新頻度
  更新頻度の高いソフトウェアに関しては、更新のたびにコンポーネントを解析する必要があり、精査を実施する場合は対応工数が大きくなる。そのため更新頻度と対応工数を考慮し、どの程度精査するか検討する。
  （例）本ソフトウェアは隔週のソフトウェアアップデートを実施しているため、間接部品の精査を実施する場合、対応工数が見合わないため精査は直接部品に限定する。ただし、１年に１回のメジャーアップデートの際にはパッケージマネージャーを基に一部の間接部品の精査を実施する。

SBOM の解析結果を精査するときの観点は大きく 2 つあり「ツールが正しく動作しているか」、「コンポーネントの解析結果が正しいか」に注意する必要がある。それぞれの観点について、具体的な精査方法を例示する。

- ツールが正しく動作しているか
  SBOM 作成ツールにソフトウェアを読み込ませ、SBOM を作成する。これが正しくできているか確認することは重要である。実際に SBOM 作成ツールを検証する中で「ファイルは作成されているが、内容が適切に作成されていない」ことがあった。ファイルが作成されている等の表面的な確認だけではなく、SBOM 作成ツールの実行ログを確認することで作成が確実に成功しているか確認することは重要である。SBOM 作成ツールの実行結果画面のスクリーンショットを図：5.1.1 に示す。



図： 5.1.1 SBOM 作成ツールの実行結果

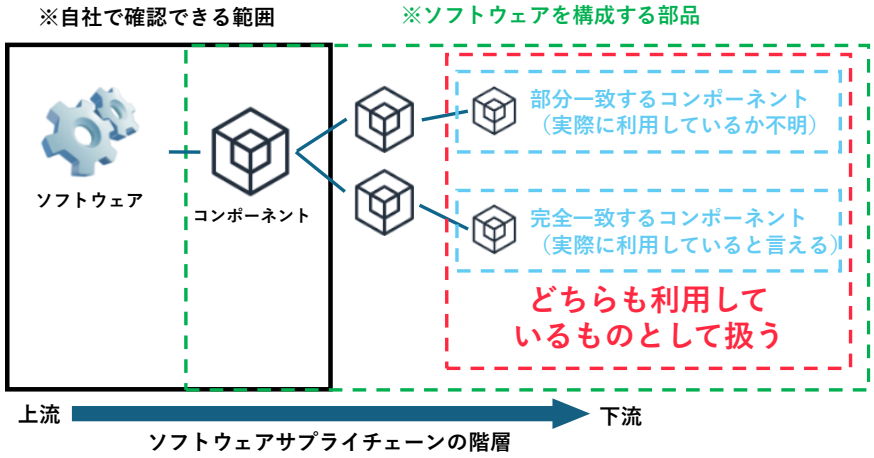
- コンポーネントの解析結果が正しいか
  コンポーネントの解析はソースコードのシグネチャを用いたパターンマッチ（スニペットマッチ）で行われる。アプリケーションに含まれるコンポーネントと OSS をスニペットマッチで解析したときに完全一致するか、部分一致するかで SBOM の解析結果の精査方法が変わる。完全一致したときは 1. の手法で SBOM の精査を行い、部分一致したときは 2. および 3. の手法で SBOM の精査を行う。

1. コンポーネントの解析結果に抜け漏れがないか

   SBOM 作成ツールが検出したコンポーネントが完全一致の場合、作成した SBOM に記載されたコンポーネント検出結果に抜け漏れがないか精査するために、アプリケーションに含まれるコンポーネントの一覧が必要になる。コンポーネント一覧の作成には、パッケージマネージャーで出力したプロジェクトの依存関係ツリーを用いると、パッケージマネージャーの管理範囲で完全なコンポーネント一覧のため信頼性が高い。

2. コンポーネントを過検出していないか

   以下、図：5.1.2 に示すように、SBOM 作成ツールが検出したコンポーネントが部分一致として検出した場合、開発したアプリケーションのソースコードと Github などで公開されているソースコードを比較することで、その同一性を検証することができる。しかし、ソースコードを直接比較するには多くの時間や人的コストが必要なので現実的には困難である。同一性を確認できないコンポーネントはソフトウェアで利用しているものとして扱い、検出されたコンポーネントのライセンスを確認し、ライセンス違反リスクを評価し対応することが妥当である。



図： 5.1.2 コンポーネント過検出とライセンス取り扱い

3. 既知の改変された OSS を検出できているか

   SBOM 作成ツールでは、改変された OSS は部分一致として検出する。場合によっては検出されない可能性がある。しかし、改変されたコンポーネントの検出を確認することを人が実施することは現実的に困難である。SBOM で検出された部分一致のコンポーネントに対して、開発時の詳細設計書で OSS の改変が指示されているか確認することが現実的な検出手段である。

5.2　SBOM の作成

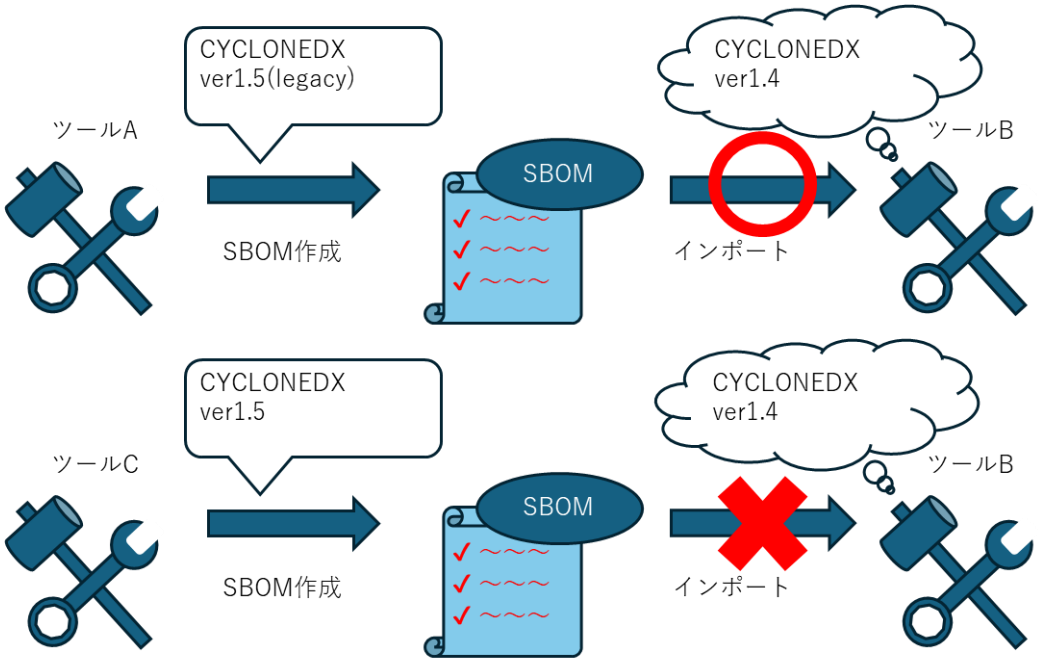
本節では、SBOM の作成・共有フェーズにおける SBOM の作成について説明する。以下、表：5.2.1 に SBOM の作成における実施事項を示す。

表：5.2.1 SBOM の作成における実施事項

No	チェック項目	内容	参考ページ
1	要件を踏まえて作成する SBOM のフォーマット、出力ファイル形式を決定する	提供先や自社で定めた要件を基に SPDX、CycloneDX などのフォーマットとバージョン、出力ファイル形式を決定する。	P39-40
2	SBOM ツールを用いて要件を満足する SBOM を作成する	SBOM ツールを用いて決定したフォーマット、バージョン、出力ファイル形式で SBOM を作成する。	P40-41

解析したコンポーネントの情報に基づき、SBOM を生成する。SBOM 作成にあたり事前に提供先の要件や規制を確認し、どのフォーマットを使用するのか、含める項目を決定する必要がある。項目に関しては 2.3 章の「SBOM 最小要件」に加え、提供先や規制によって定められた項目を含める必要がある。SBOM のフォーマットでは項目の情報として、情報なし（NOASSERTION）も許容されているが、SBOM 要件として必要な項目が情報なしとして記載されていないか注意が必要である。

SBOM フォーマットとそのバージョンを決定する際には、実際に提供先が使用しているツールに実際に取り込めるか確認することが望ましい。SBOM ツールにおける課題として、同フォーマット、同バージョンであっても SBOM を出力する際に表記方法に軽微な差異が存在することで別ツールへの取り込みに失敗するという課題がある。以下の図：5.2.1 に同フォーマット、同バージョンの SBOM フォーマットを利用した場合でも正常に取り込むことができなかった一例を示す。



ツールCとBでは同フォーマットでも軽微な表記方法の違いがあり
取り込みの際にエラーが発生する

図： 5.2.1 SBOM ツールの SBOM フォーマット取り込みエラー例

図：5.2.1 の例ではツール A、C ともに同じフォーマット（CYCLONEDX)、同じバージョン（ver1.5）で出力を実施しているがツール B の取り込みの際にエラーが発生する。ツール A、C の差異、図：5.2.1 の例では legacy フォーマットか、legacy フォーマットでないか、はどちらも標準で定められた CYCLONEDX ver1.5 フォーマットに準拠しており、ツールの不具合を原因とする事象ではない。そのため同フォーマット、同バージョン形式の SBOM でも正常に共有できない事象が起きる。原因は CYCLONEDX ver1.5(legacy)と CYCLONEDX ver1.5 の互換性の問題であり、CYCLONEDX ver1.5 のフォーマットが CYCLONEDX ver1.4 と前方互換が無いことに起因している。図：5.2.1 の例以外にも SBOM ツールごとに同バージョン、同フォーマットでも軽微な表記方法に違いが発生し、正常に取り込めない例もある。このようなインポートのエラーが起きないようにするためにも、SBOM の提供先のツールに正常にインポートできるか検証の上、採用するフォーマットを選定し SBOM を作成する必要がある。

また、SBOM はサードパーティサプライヤーなどの第三者が公開・提供しているものをインポートしてすることで効率的に作成することができる。提供された SBOM をインポートすることで、5.1 節「コンポーネントの解析」で実施したコンポーネント解析結果の精度を上げることができ、精査の際に活用することができる。

第三者から提供を受けたSBOMをインポートしてSBOMを作成する際には、自組織でコードを改変している場合、使用しているコンポーネントに差異が発生する場合があるため、間違った情報がSBOMに記載されることに注意する必要がる。

5.3　SBOMの共有

本節では、SBOMの作成・共有フェーズにおけるSBOMの共有について説明する。以下、表：5.3.1にSBOM共有における実施事項を示す。

表：5.3.1 SBOM共有における実施事項

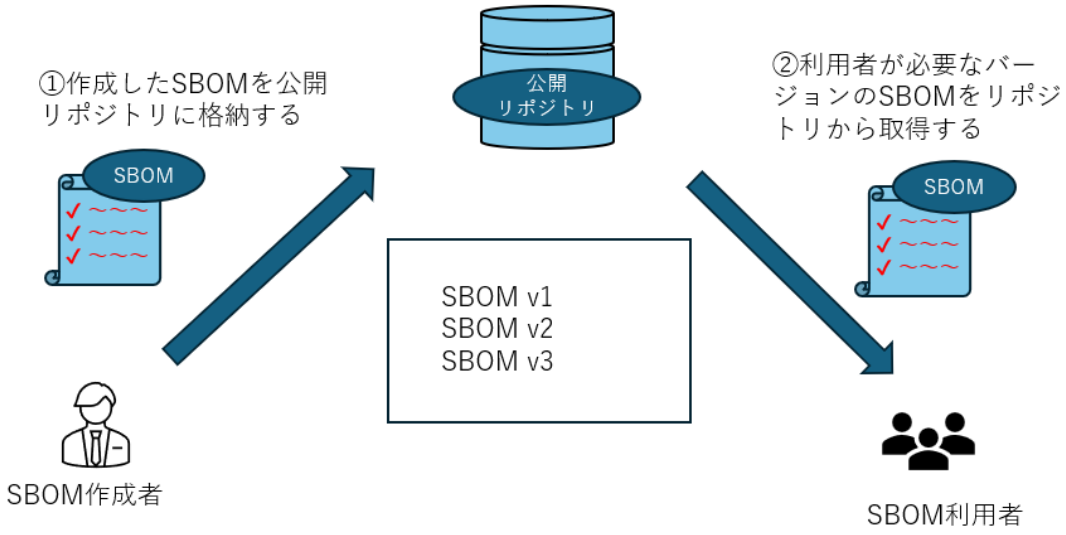
No	チェック項目	観点	参考ページ
1	規制や要求を確認しSBOMの公開範囲、証明書導入の有無を決定する	SBOMの共有を幅広く公開するのか、特定の範囲に限定するのかを決定する。また、改ざん対策として証明書を導入するのか検討する。	P41
2	SBOMの公開範囲、自社の特性を考慮し共有方法を検討する	幅広く公開、範囲を限定して公開するかを決定し、自社に適用する共有方法を決定する。	P41-45
3	決定した共有手法をもとにSBOMを共有する	決定した共有手法で公開する環境を整え公開する。	P41-45

作成したSBOMはソフトウェアの利用者や納入先に共有することでソフトウェアサプライチェーンの透明性を高めソフトウェアサプライチェーンリスクの低減につながる。しかし、2.5節の「SBOMに関する誤解と事実」で述べたようにSBOMは必ずしも公開する必要があるわけではない。SBOMは規制や要求事項として共有が求められている場合やサプライヤー、作成者の判断で必要に応じて適切な対象者に共有する。

SBOMを共有する方法は事前に納入者と協議して適切な方法を選択する必要がある。SBOMの一般的な共有方法は大きく3つあり、①製品に組み込む、②リポジトリに格納、③Web公開、である。これらの方法からSBOMを幅広く公開したいのか、特定の利用者に限定して公開したいのかなどの要件に合わせて選択する。以下にSBOM共有方法の例を示す。

幅広く公開する場合

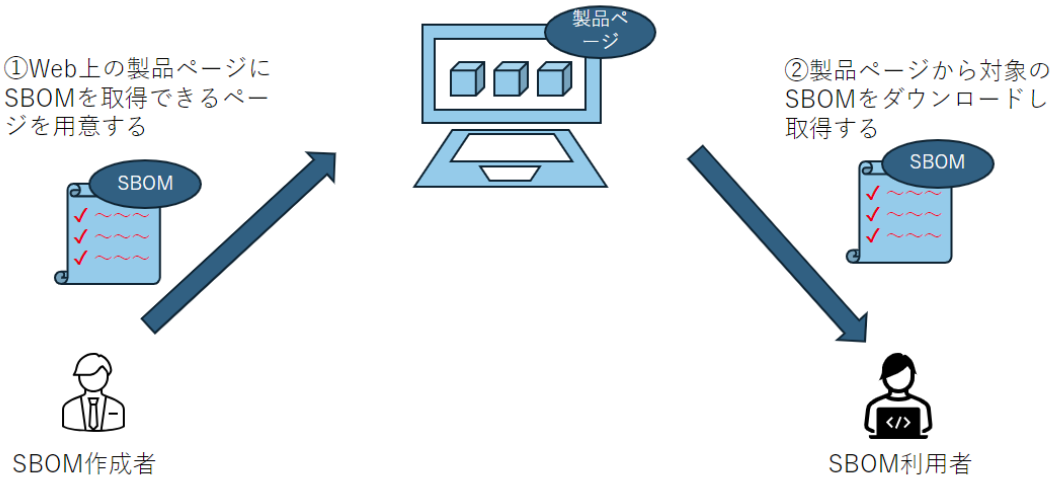
1. 公開リポジトリに格納して共有する



図： 5.3.1 公開リポジトリに格納する共有方法

図：5.3.1 に示すように、公開リポジトリに格納して共有する方法では利用者を限定せず一般に公開しているリポジトリを利用して誰でも SBOM を取得できる。Web ページでの公開と比較して管理性に優れており、SBOM の更新頻度が高い場合などで効果を発揮する。

2. 製品の Web ページに公開する

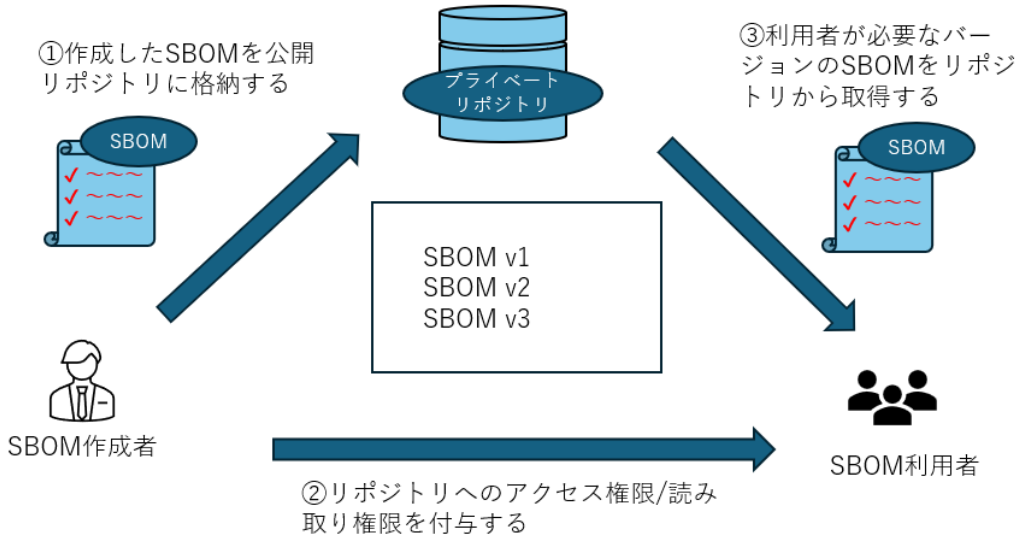


図： 5.3.2 製品の Web ページに公開する共有方法

図：5.3.2 で示すように、製品の Web ページに公開する方法では、自社で運用している製品ページに SBOM を取得する機能を持たせた方法である。SBOM を更新するためには Web ページの改変が必要なため公開リポジトリによる共有方法と比較して管理面は劣るが、製品と紐づけて SBOM を取得できるためユーザーが直感的に利用することができる。また専用のリポジトリを容易する必要がない。

特定の利用者に限定して公開する方法

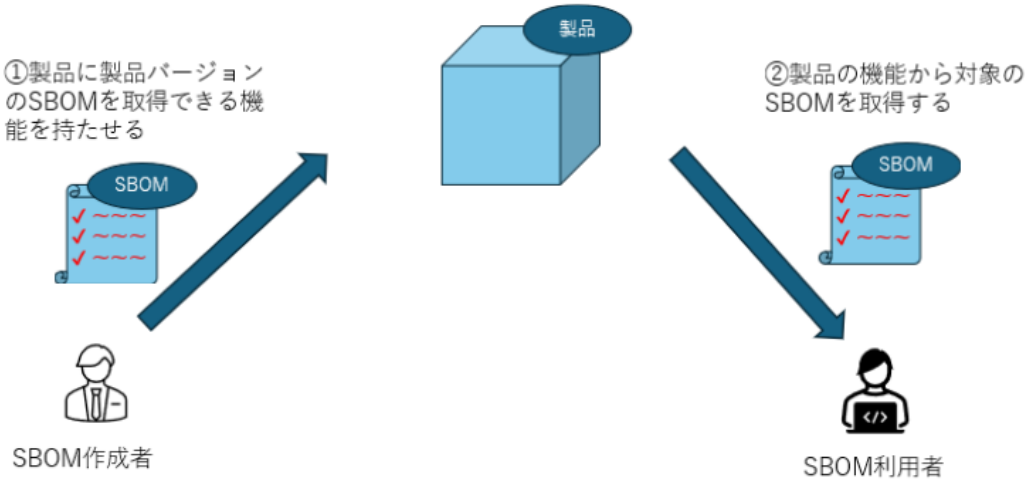
1. プライベートリポジトリに格納して共有する



図： 5.3.3 プライベートリポジトリに格納する共有方法

図：5.3.3 に示すように、プライベートリポジトリに格納して共有する方法の基本的な方法は公開リポジトリに格納して共有する方法と同様であるが、SBOM を取得する利用者を制限するために事前にリポジトリへのアクセス権、読み取り権限を適切に共有する必要がある。

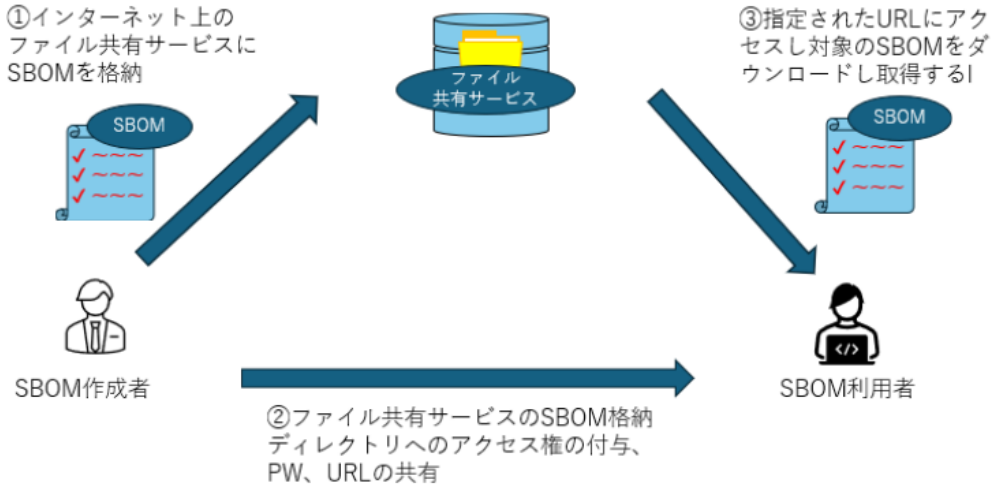
2. 製品に組み込む



図： 5.3.4 製品に組み込む共有方法

図：5.3.4 に示すように、製品に組み込む方法は、ソフトウェアや対象のソフトウェアが乗っている製品に SBOM を取得する機能を持たせる方法である。製品に新たな機能を持たせる必要がある一方で、製品のバージョンと同じ SBOM を取得することができるようにすることでユーザーに製品のバージョン情報を意識させることなく SBOM を共有することができる。また、インターネットに接続する必要がなくクローズドな環境で運用したいなどのニーズを満たすこともできる。

3. クラウドのファイル共有サービスで共有する



図： 5.3.5 クラウドを用いたファイル共有方法

図：5.3.5 に示すように、クラウドのファイル共有サービスで共有する方法では、クラウドのファイル共有サービスの特定のディレクトリに SBOM を格納したのちにディレクトリの URL、パスワードの共有、アクセス権限、読み取り権限の付与をして SBOM 利用者が SBOM を取得できるようにする。ファイル共有サービスの契約のみで実施できるため比較的簡易に実施することができる。

SBOM のデータ自体のセキュリティ要件として改ざん防止などの完全性の確保が要件に含まれる場合、電子署名などの技術活用を検討する。

6. SBOM 運用・管理フェーズにおける実施事項

6.1 SBOM に基づく脆弱性管理

本節では、SBOM の運用・管理フェーズにおける SBOM に基づく脆弱性管理について説明する。以下、表：6.1.1 に SBOM に基づく脆弱性管理における実施事項を示す。

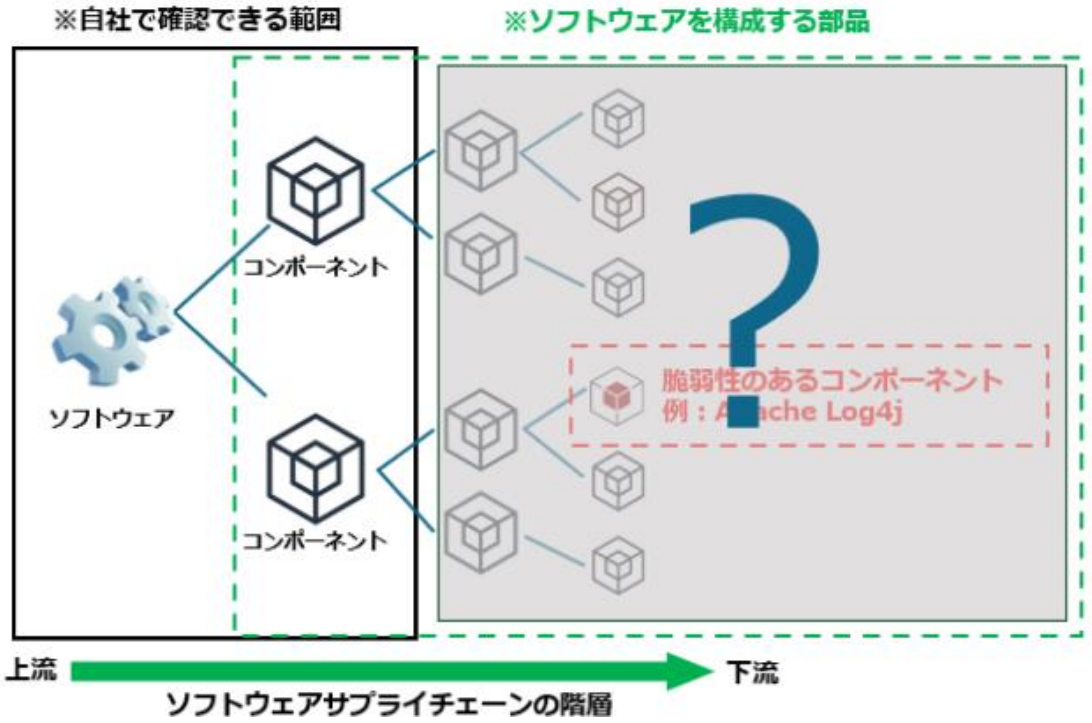
表 6.1.1 SBOM に基づく脆弱性管理における実施事項

No	チェック項目	観点	参考ページ
1	脆弱性管理方法を決定する	導入した既存ツールで脆弱性管理を実施するのか、自前で構築するのかを決定する。	P45-51
2	脆弱性 DB の選定を実施する。	Web UI を利用する方法等で「発見」に用いる脆弱性 DB を決定する。	P48-51
3	環境を構築する	API を利用する場合では、コーディングや必要があれば脆弱性 DB の構築を実施する。	P49-50
4	発見した脆弱性への対応基準方針を決定する。	優先度づけのための方針、基準を設定し対応にかかるコストをできるだけ削減する。	P51-52

脆弱性対応フローと SBOM の関連性

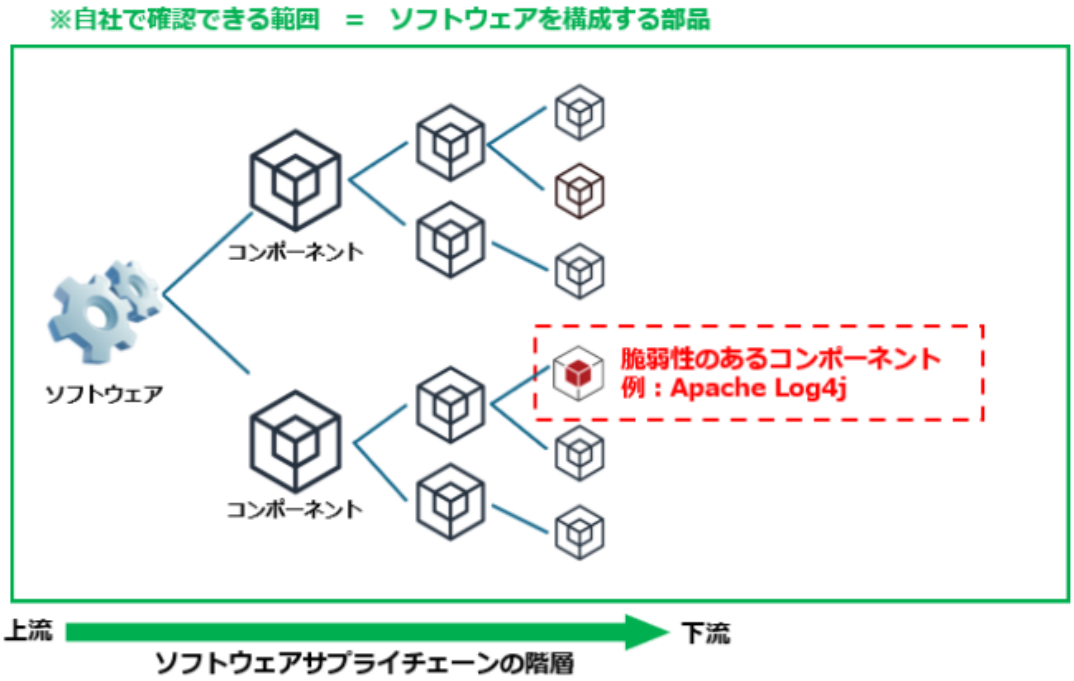
SBOM はあくまでも「ソフトウェアの部品表」であるため、作成しただけでは十分な効果は発揮しない。SBOM を作成し、ソフトウェアを構成するコンポーネントと脆弱性を照合することで脆弱性管理に効果を発揮する。SBOM を用いない従来の手法では図：6.1.1 のように自社で把握できていない範囲で脆弱性が発見された場合、ソフトウェアに脆弱性となるコンポーネントが含まれるかどうかを確認するために、システムやそのコンポーネントを調査する必要があった。発見された脆弱性が「Apache log4j」のような広く利用されて

いる OSS であった場合、非常に多くのシステムやそのコンポーネントを調査する必要があるため、発見までに膨大な工数を要し、脆弱性対応に多くの時間がかかっていた。そして最悪の場合、把握しているコンポーネントのみで脆弱性の有無を判断し脆弱性があることを認識できず、脆弱なソフトウェアを利用し続けることになる。



図： 6.1.1 従来手法のコンポーネント管理範囲

SBOM を活用した手法の場合、図：6.1.2 のように上流から下流に至るまでソフトウェアを構成するコンポーネントが可視化できる。これにより脆弱性のあるコンポーネント発見までにかかる時間、工数を大幅に削減し脆弱性対応にかかる時間を短縮することができる。



図： 6.1.2 SBOM を活用したコンポーネント管理範囲

ただし、SBOM を導入し有効活用したとしても脆弱性対応のすべてが完結するわけではない。脆弱性対応において SBOM を効果的に活用できる範囲は脆弱性対応フローにおける一部である。図：6.1.3 に、新たな脆弱性が発見および公表されてから脅威を取り除くまでの脆弱性対応の一連のプロセスを示す。

図： 6.1.3 脆弱性対応の一連のプロセス

本プロセスは「発見」、「分類と優先順位付け」、「解決」、「再評価」、「レポート作成」の 5 つのステップに分かれている。以下にそれぞれのステップで実施する事項を説明する。

- 発見
  組織が持つ IT 資産に既知および潜在的な脆弱性がないかを検査する。

- 分類と優先順位付け

発見された脆弱性の特性(例：不十分な暗号化、デバイスの設定ミスなど)で分類を行い、そのリスクを評価し、脆弱性対応の優先順位を決定する。

- 解決
  脆弱性への対応として「修復」、「軽減」、「受容」のいずれかを選択する。「修復」は、脆弱性を取り除くために脆弱性に対する修正パッチの適用または、脆弱な IT 資産を取り除く。「軽減」は、脆弱性を悪用されるリスクを低減するためにセグメントの分割や特定の機能の利用停止等のワークアラウンドを実施し、悪用条件となる要因を減らす。「受容」は、脆弱性を悪用する難易度、悪用された際の影響を考慮し、対応を実施しない選択をすることである。

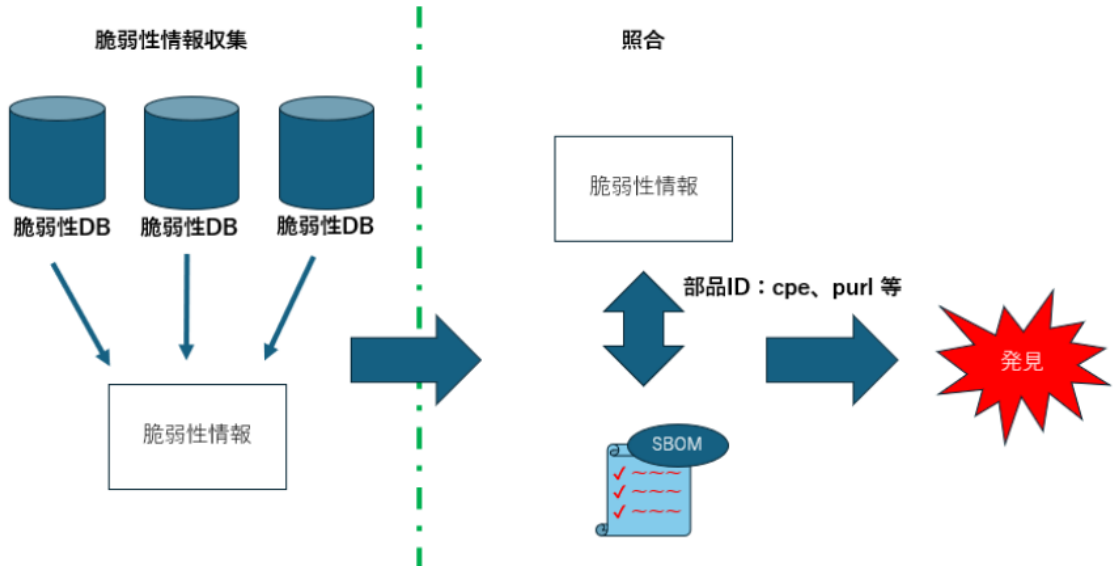
- 再評価
  「解決」を実施した後に脆弱性悪用に対するリスクが低減し、修復や軽減が機能していることを確認する。

- レポート作成
  脆弱性管理プロセス全体を記録し、関係者に報告する。この文書化は将来のセキュリティ対策に役立つことが期待できる。

SBOM は脆弱性対応フローのうち、「発見」と「分類と優先順位付け」において有効である。ソフトウェアを構成するコンポーネントに存在する脆弱性を検知し、脆弱性の分類を簡略化できるため、脆弱性対応にかかる工数の削減につながる。ここでは「解決」以降のプロセスは SBOM の活用が影響を与えるものではないため、本ドキュメントのスコープ外とする。

SBOM を活用した脆弱性の発見

SBOM を活用して脆弱性を「発見」するためには、図：6.1.4 に示すように脆弱性に関する情報を集める「脆弱性情報の収集」、SBOM と収集した脆弱性情報を照合する「照合」の 2 プロセスを実施する必要がある。



図： 6.1.4 SBOM と脆弱性 DB を連携した脆弱性管理

SBOM を導入した企業は、SBOM 導入前と比較してより多くのコンポーネントに対して脆弱性管理を実施する必要がある。そのため、脆弱性対応フローの「発見」フェーズは自動で実施することが望ましい。ここでは、一部手動で実施する方法を含めた、「発見」フェーズにおける 3 つの手法について説明する。

- Web UI の利用

  脆弱性 DB を持つ機関が公開している Web UI の検索ページから脆弱性情報を収集し、利用する部品 ID を基にソフトウェアのコンポーネント情報と照合する。Web UI を利用する手法では、情報収集、照合は手動で実施することとなる。そのため、本手法はこれから説明する自動化手法を実施するための検証に用いる。

  ＜メリット＞

  ・ 検索対象とする脆弱性 DB を柔軟に変更できるため、自動化に向けた検証に向いている。

  ＜懸念事項＞

  ・ 情報収集、照合ともに手動で実施する必要があるため、脆弱性情報の件数が増加した場合に実運用に耐えられない可能性がある。

- API を利用した連携

  脆弱性 DB から提供される API を利用し、脆弱性を収集する。また、cpe や purl といった部品 ID をキーとし、脆弱性情報とソフトウェアコンポーネントの情報をマッチン

グすることで照合することができる。脆弱性情報の収集と照合、どちらにおいてもスクリプトを作成し自動化することが可能である。

＜メリット＞
- 「発見」フェーズを自動化することができるため、工数を削減できる。
- 脆弱性 DB を柔軟に組み合わせることができる。

＜懸念事項＞
- API 利用や SBOM との照合等の実現のためにコーディングスキルと仕組みづくりの工数がかかる。
- 脆弱性 DB の選定のための調査、検証、その運用に専門的なスキルと膨大な工数がかかる。

- 既存ツールの利用

  脆弱性管理機能を持っている SBOM ツールは公的機関もしくは民間企業が提供する脆弱性情報を収集し、独自の脆弱性 DB を構築している。独自の脆弱性 DB とソフトウェアのコンポーネント情報を照合し、脆弱性を発見する。また、既存ツールの中に、脆弱性の分類、優先度を示すツールも存在する。

  ＜メリット＞
  - SBOM ツールのサービスの一部として提供しているため、収集した脆弱性情報は一定の網羅性、品質が担保されている。
  - コーディングなどの技術的スキルを必要としない
  - 「脆弱性情報収集」、「照合」を自動で実施し、その仕組みづくりも必要としないため限られたリソースで実施することができる。

  ＜考慮すべき事項＞
  - 利用する SBOM ツールによっては、脆弱性 DB と紐づけを行わないものがある。
  - 有償ツールの場合、継続的なライセンス費用が必要となる。
  - 独自の脆弱性 DB が利用するコンポーネントを含んでいない恐れがある。

ここでは 3 つの手法について例示したが、「発見」フェーズではまず、既存ツールの利用検討を推奨する。Web UI を利用する手法は、手動による脆弱性情報の収集と照合の工数が膨大になり、リソース不足となる可能性がある。API を用いた手法は、導入に専門的なスキルや知識、工数がかかることに加え、更新や改修などの運用が必要である。推奨する既存ツールの利用は、ツールベンダーの知見を活かし、様々な脆弱性 DB から脆弱性情報を収集

し、独自の DB を構築している。また、データベースの更新などの保守の必要もない。脆弱性情報との照合もツール内で解析したコンポーネントや取り込んだ SBOM に対して自動で実施される。そのため、一定以上の品質が担保でき、脆弱性を発見するために新たに構築する仕組みづくりも必要ない。従って、 導入に至るまでの人的・時間的リソースと脆弱性を検知する精度を考慮すると、既存のツールを活用することが好ましい。

SBOM 導入により生じる脆弱性管理の課題

SBOM を活用することで、これまで以上に多くのコンポーネントが可視化され、それに伴い、検知するコンポーネントの脆弱性も増加する。そのため、日々公表される脆弱性を迅速に把握し、これまで以上に多く検知する脆弱性への対応が求められる。SBOM の活用により検知する脆弱性は増加するが、検知した脆弱性に対応するためのリソースには限りがある。検知したすべての脆弱性に対し対応できることが理想だが、現実的ではない。今後、「解決」として「受容」を選択する場面が増えるため、適切な「分類と脆弱性の優先付け」が重要である。

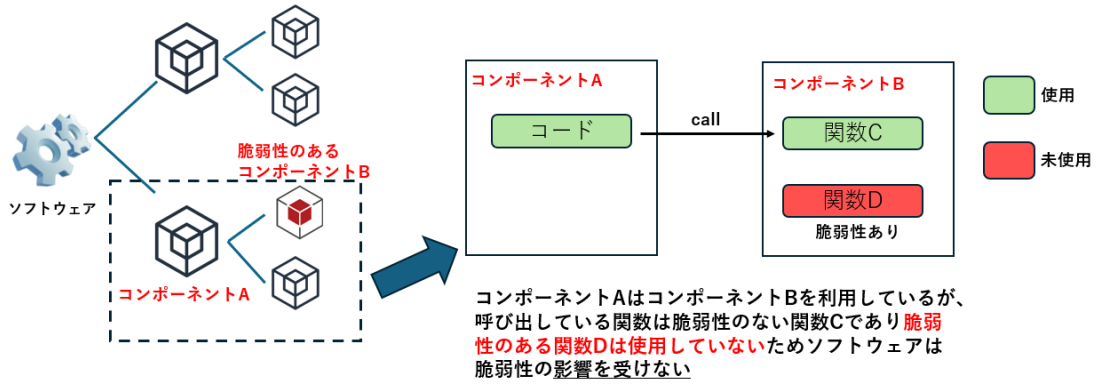
コラム：VEX を用いた脆弱性管理

SBOM を活用した脆弱性管理では、従来のソフトウェア製品や OS パッケージを対象とする脆弱性管理と比較して多くの脆弱性を検出する。理論上はそのソフトウェアに含まれる既知の脆弱性をすべて検出できる。SBOM を活用することにより、これまで以上に多くの脆弱性が検出され、脆弱性の優先付けや修正にかかるリソースは従来の脆弱性管理においても不足気味であったが、そのリソース不足は加速することが予想される。

検知された脆弱性を効率的に優先付けするためのフォーマット「VEX」が提唱された。この VEX が浸透することにより、“必要最小限の対応を迅速に実施すること”が可能となる。VEX は”Vulnerability Exploitability eXchange”の略であり米国商務省電気通信情報局（NTIA）によって開発されたセキュリティアドバイザリのフォーマットの一つである。VEX は脆弱性に関する情報を記載するフォーマットであり、ソフトウェアサプライチェーンの SBOM の使用に関するニーズを満たすために開発された。VEX は SBOM との使用に限定されるものではなく、脆弱性管理に関するフォーマットとして幅広く利用することができる。VEX は特定の脆弱性が製品に与える影響を追加情報として提供するために標準化されたフォーマットである。これにより脆弱性が含まれるコンポーネントが見つかった際に、その脆弱性により影響を受けるのか、影響がある場合にどのような対策が推奨されるのかを理解することができる。

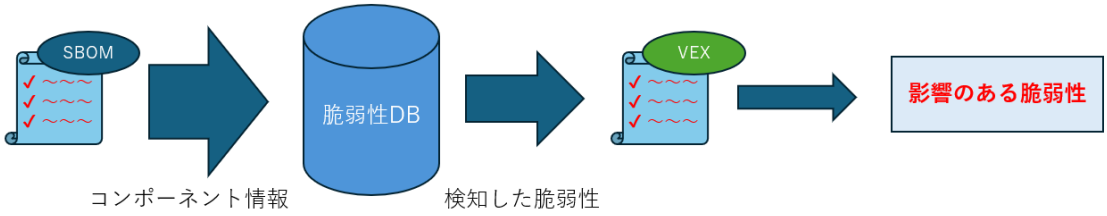
ソフトウェアのコンポーネントで検知した脆弱性は単体では悪用することが可能だが、

そのソフトウェアで必ず悪用可能というわけではない。上流のコンポーネントに存在する脆弱性が、最終的にはソフトウェアにおいて「悪用不可」な場合が多い。例えば、以下の図：6.1.5 に示す例では脆弱性のあるコンポーネントを使用しているが、そのコンポーネントの脆弱性箇所である特定のライブラリを使用していないため「悪用不可」になる。



図： 6.1.5 ソフトウェアに脆弱性を含むが悪用はできない例

SBOM と脆弱性 DB を用いて検知した脆弱性に対し VEX を適用することで、ソフトウェアが悪用される可能性のある脆弱性を正確に把握することができるようになる。VEX を用いた脆弱性管理の概観を図 6.1.6 に示す。



図： 6.1.6　VEX を用いた脆弱性管理

VEX を用いて、本来影響を受ける脆弱性を正確に把握することで、優先付けにかかる工数を大幅に削減することができる。

6.2　SBOM に基づくライセンス管理[9]

本節では、SBOM の運用・管理フェーズにおける SBOM に基づくライセンス管理について説明する。以下、表：6.2.1 に SBOM に基づくライセンス管理における実施事項を示す。

表 6.2.1 ライセンス管理における実施事項

No	チェック項目	観点	参考ページ
1	利用 OSS の洗い出し	SBOM ツールを活用して利用している OSS を一覧化する。	P53-54
2	利用 OSS のライセンスが持つ利用条件を確認する。	利用している OSS ライセンスの利用条件を確認する。利用条件を違反していないか確認し、適宜対応を行う。	P53-54

SBOM を活用することで利用しているコンポーネントのライセンスを可視化できる。SBOM を活用したライセンス管理では可視化したライセンスについて使用条件や再配布条件を理解している必要がある。以下に OSS ライセンスの一例と注意点を示す。

使用に注意が必要なライセンス例

- GPL
  コピーレフトライセンスであり、派生物も GPL で配布の必要がある。注意点としては、GPL で配布されたソフトウェアを改変した場合でも GPL で配布する必要があるため、改変したソフトウェアの使用、コピー、改変、再配布などにおいて他者の使用を制限してはならない。
- LGPL
  GPL の一種であり、動的リンクされたライブラリの場合、派生物は GPL を適用しなくてもよい。静的リンクされた場合には、派生物でも GPL で配布する必要がある場合がある。注意点として、ライブラリをほかのソフトウェアとリンクする際には、そのリンク方法に注意が必要である。
- MIT License
  MIT ライセンスは寛容なライセンスであり、ほぼ制限がない。ソフトウェアの使用、コピー、改変、再配布が自由に行えるが、元のライセンス情報を含める必要がある。注意点としては、ライセンス条文を必ずソースコードに含める必要がある。
- Apache License2.0
  特許権の明示的なライセンス許諾が含まれている。ソフトウェアの仕様、改変、再配布

[9]経済産業省：OSS の利活用及びそのセキュリティ確保に向けた 管理手法に関する事例集，令和 4 年[6]

が可能であり、変更を加えた場合には、その変更内容をファイルに記載する必要がある。注意点として、変更内容を明確にし、元の著作権表示を保持することが求められる。

- MPL
  MPL はファイル単位でのコピーレフトを提供する。変更を加えたファイルのみを公開する義務があるため、プロプライエタリなコードと混在させることが可能。注意点として、変更を加えたファイルの公開義務を理解し、適切に対応する必要がある。
- AGPL
  AGPL は GPL のサーバ利用版であり、サーバ経由で利用された場合でもソースコードの公開義務が発生する。注意点として、サーバアプリケーションに使用する場合、公開義務が厳格になる点に留意する必要がある。

上記の例以外にも OSS に関するライセンスは数多く存在する。SBOM で可視化したライセンスを理解したうえで、開発する必要がある。SBOM 導入により意図せずライセンス違反している OSS が発見される可能性があり、対応が必要であれば法務部や同等の機能を持つ組織と連携して対応することで法的リスクを低減する。開発段階から OSS 利用の方針を取り決めておくことで、法的リスクを根本的に低減させることができる。

6.3　SBOM 情報の管理[10]

本節では、SBOM の運用・管理フェーズにおける SBOM の情報管理について説明する。以下、表：6.3.1 に SBOM の情報管理における実施事項を示す。

表 6.3.1 SBOM 情報の管理における実施事項

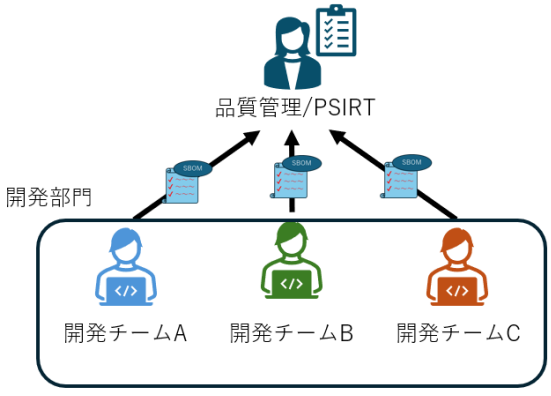
No	チェック項目	観点	参考ページ
1	SBOM 管理体制の整備	主管部署で一元管理するための体制を整える。	P54-55
2	SBOM の提供期間を定める	ソフトウェアやコンポーネントの提供期間を考慮し、SBOM の提供期間を定める。 SBOM の提供期間を公表する方法を定める。 必要に応じて顧客と協議を行う。	P55-56
3	SBOM の更新条件や時期を決める。	SBOM の更新条件や時期を定め、開発する部門に事前に周知する。	P56

[10] 経済産業省「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.2.0」を参考に構成や実施項目を検討[1][2]

SBOM は脆弱性管理やライセンス管理を目的とした社内利用のみならず、社外からの問い合わせに応じて参照・提供する情報である。そのため、SBOM の変更履歴は適切に管理し信頼性を担保する必要がある。

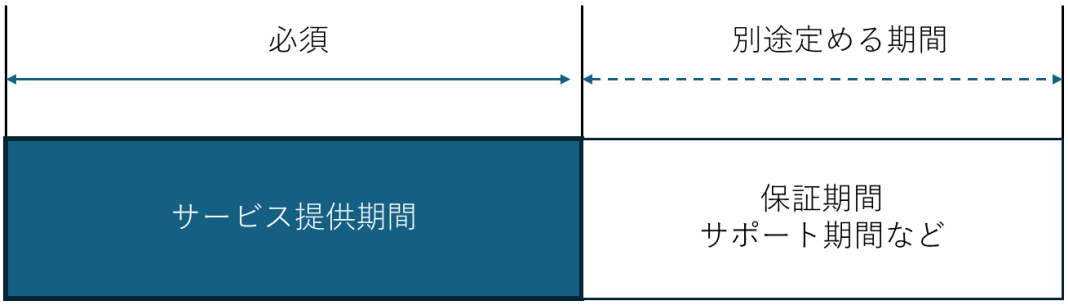
そのため、SBOM を管理していく中で次のような取り決めを行っておくことを推奨する。
・管理体制
・提供期間
・更新頻度など

- 管理体制
  ソフトウェアのコンポーネントを把握できるのは、開発部門もしくは品質管理/PSIRT である。開発時に利用するコンポーネントに詳しいのは開発チームとなりえるが、自組織が提供するコンポーネント等のすべてを一つの開発チームが手掛けるとは限らない。加えて開発チームそのものが複数存在しており管理すべき SBOM の数が膨大な場合もある。効率的な SBOM 管理体制を構築するには、自社製品の管理を担う品質管理/PSIRT が SBOM を全社で一括管理する体制が望ましい。図：6.3.1 に示すように、品質管理/PSIRT が開発チームと連携しながら SBOM を一括管理する手段として SBOM ツールの活用を推奨する。SBOM ツールを活用することで、複数のプロジェクトを横断したバージョン管理や変更履歴を管理することができ、SBOM を全社レベルで効率的に管理することができる。



図： 6.3.1 効率的な SBOM 管理体制例

- 保管期間
  SBOM の保管期間を設定するにあたり、最低限、製品のサポート期間中は SBOM を保管する必要がある。SBOM を最低限保管する期間の例として、「サービスを提供する期間」や「一般に流通する期間」「サポート・延長保証期間」「ライセンスによって定められた期間」などが挙げられる。SBOM 保管期間の検討例を図 6.3.2 に示す。



図： 6.3.2 SBOM 保管期間の検討例

- 更新する条件
  開発期間とサービス提供期間で更新する条件が異なる。開発期間であれば、日時による定期的な更新や開発ロードマップのマイルストーン毎、結合テストなどの開発期間の節目毎の更新を推奨する。サービス提供期間ではソフトウェアの脆弱性対応、バグ修正、機能追加などのソフトウェアバージョン変更に伴って SBOM を更新する必要がある。

7. まとめ

本書では、SBOM の導入・運用のハードルを下げることを目的として、検証に基づき実務者が実施すべきこと、知っておくべきことを記載した。「環境構築・体制フェーズ」、「SBOM 作成・共有フェーズ」、「SBOM 運用・管理フェーズ」の 3 章に分けて紹介した。

「SBOM 作成・共有フェーズ」では、第一に自組織における課題と目的を明確にする。明確化した課題や目的を基に、SBOM の対象とするソフトウェアを決定し、ソフトウェアに関する情報を整理する。ここで整理した情報をもとに体制整備、ツール選定を実施する。体制整備では、SBOM 導入の主管となる部門に加えて、SBOM を作成する部門、脆弱性管理やライセンス管理をする運用部門の体制で実施することが導入からスムーズな運用を実現するために重要である。SBOM ツール選定では、明確にした課題と目的、自組織における制約、ソフトウェアの開発環境、脆弱性管理機能など多くの項目に関して検証を実施し自組織に適したツールを導入する。特に SBOM 導入の目的になる可能性の高い脆弱性管理は SBOM ツールの機能に大きく依存するところが大きいため慎重に選定する必要がある。

「SBOM 作成・共有フェーズ」では、コンポーネントの解析と SBOM の作成、SBOM の共有について記載した。コンポーネント解析前に、ソフトウェアの重要度とコンポーネント精査の工数のトレードオフを考慮し、精査を実施するのか、実施する場合、どの階層まで精

査するのかを決める。コンポーネントの精査は膨大な工数がかかるため、実運用に耐えられるか見極めることが重要である。コンポーネントの解析はツールで解析することが前提とされる。コンポーネント解析後は、必要であれば解析したコンポーネントの精査を実施する。SBOM ツールは完璧ではなく、検出漏れ、過検出が起きることを認識しておく必要がある。SBOM の共有では、幅広く公開する場合と公開範囲を限定する場合の共有方法を示した。自組織に適した方法を選定し、共有方法を決めることが重要である。

「SBOM 運用・管理フェーズ」では、SBOM を活用した脆弱性管理とライセンス管理、そして SBOM 情報の管理について記載した。SBOM は主に脆弱性対応フローの「発見」フェーズで効果を発揮する。脆弱性管理機能を持った SBOM ツールを利用することで、効率的かつ正確に脆弱性を発見できる。しかし、これまでよりも多くの脆弱性を発見するため、それに伴い脆弱性対応も増加する。それを踏まえた体制整備、優先付けの基準を決めることが重要である。ライセンス管理では、どのライセンスがどのような条件で利用可能なのか理解しておくことで法的リスクを低減させることができる。SBOM ツールにて可視化したライセンス情報を活用することで、開発段階からライセンス管理をしていくことも効果的である。また、SBOM の信頼性を高めるためにも、SBOM 自体の管理は必須である。保管期間、定期的な更新、管理体制の整備を適切に行うことが重要である。

本書では SBOM の導入・運用に関して、検証結果や企業、公的機関へのヒアリングをもとに作成したが、あくまで本プロジェクトメンバーで検討した結果であり、ヒアリングにご協力いただいた企業の意見とは異なる点もある。必ずしも唯一無二の正解とは言えないが、SBOM 導入・運用をするために、何から始め、どう考え、どう導入し運用していくのかを検討する材料は詰め込んでいる。

なお、本書は 2024 年 6 月時点での製品・サービスの検証結果をもとに執筆したものであり、二年後にも本書の内容がそのまま通用するとは考えられない。SBOM は現在進行形で進化しているため、常に最新の情報や技術にアップデートし続ける必要がある。

本ガイドラインが、SBOM 導入・運用の助けになれば幸いである。

Appendix

用語集

用語	意味・解説
AI	人間の知能を模倣するコンピューターシステムやプログラムのことである。
API	Application Programming Interface の略。ソフトウェアと別のソフトウェアをつなぐためのモジュールなどのインターフェース
CLI	コンピュータとのやり取りをすべてコマンドラインによって行う方式
DevOps	ソフトウェア開発（Development）と IT 運用（Operations）を統合する一連のプラクティスである。 これにより、開発と運用チームが協力して、ソフトウェアの開発サイクルを短縮し、品質と信頼性を向上させることを目指す。
EOL	（正式名称：End of Life）ソフトウェアやハードウェア製品のメーカーによる公式なサポートと更新の提供が終了する時点を指す。 EOL を迎えた製品は、セキュリティ更新や技術サポートが受けられなくなるため、使用を続ける際には注意が必要である。
General Public License	（略称：GPL）フリーソフトウェア財団（FSF）が作成したソフトウェアライセンスである。 GPL は、ソフトウェアの自由な使用、改変、再配布を保証し、派生作品も同じライセンスで公開することを義務付けることで、ソフトウェアの自由とオープン性を保護する。
GNU	フリーソフトウェア運動の一環として 1983 年にリチャード・ストールマンによって開始されたプロジェクトであり、 Unix 互換のフリーオペレーティングシステムを作成することを目的としている。 GNU プロジェクトにより開発されたソフトウェアは、自由に使用、改変、再配布することができ、オープンソースの理念を推進している。
GUI	コンピュータから図形や画像を用いて情報を提示し、マウス操作などによる入力を行う方式

IoT	（正式名称：Internet of Things）インターネットに接続された物理デバイスやセンサーが相互に通信し、データを収集・交換する技術である。
ISO/IEC 5962:2021	ソフトウェア部品表（SBOM: Software Bill of Materials）に関する国際標準規格である。 この規格は、ソフトウェア製品に含まれるすべてのコンポーネントを記載し、ソフトウェアサプライチェーンの透明性とセキュリティを向上させることを目的としている。
json	（正式名称：JavaScript Object Notation）データを構造化して表現するための軽量なテキストフォーマットである。 主にキーと値のペアでデータを表し、データの交換や保存に広く利用されており、人間にも機械にも読みやすい特徴がある。
Apache Log4j	Apache Software Foundation が提供する Java ベースのロギングライブラリである。 アプリケーションのログ出力を効率的に管理するために広く使用されている。
PoC	Proof of Concept の略。概念実証。施策や技術が実現できるか確認するための簡易な試行
PSIRT	（正式名称：Product Security Incident Response Team）製品のセキュリティインシデントに対応する専門チームである。 PSIRT は、脆弱性の報告を受け取り、調査し、修正対応やユーザーへの情報提供を行い、製品のセキュリティを維持・向上させる役割を担っている。
SBOM	（正式名称：Software Bill of Materials）ソフトウェア製品に含まれるすべてのコンポーネントのリストを詳細に記載した文書である。
SolarWinds	IT インフラストラクチャ管理ソフトウェアを提供するアメリカの企業である。 2020 年には、同社の製品「Orion」のアップデートにマルウェアが混入し、多くの企業や政府機関が影響を受ける大規模なサイバー攻撃が発生した。
xml	（正式名称：Extensible Markup Language）データを階層的に構造化して表現するためのマークアップ言語である。 タグを用いてデータを記述し、柔軟で拡張性が高いため、データの交換や保存、構造化文書の作成に広く利用されている。
アーキテクチャ	ソフトウェアの基本設計や共通仕様、設計思想

オープンソースソフトウェア	（略称：OSS）ソースコードが公開され、誰でも自由に利用、改良、再配布できるソフトウェアである。
オンプレミス	社施設の構内に機器を設置してシステムを導入・運用すること
クラウド	外部が提供するサーバやサービス・システムを導入・運用すること
コンポーネント	システムやソフトウェアの機能を構成する個別の部品や要素である。 これらの部品は、独立して機能しつつ、他のコンポーネントと連携して全体のシステムやアプリケーションを実現する。
コンポーネント解析	コンポーネントをシステムの他の部分から分離して検証すること。
サプライチェーン	製品やサービスが原材料の調達から最終消費者に届くまでの一連のプロセスである。 供給元から生産、流通、販売までの流れを管理し、効率的かつ安定した供給を目指す。
シグネチャ	マルウェアの特徴的なパターンや挙動を特定するためのデータ
シンボリックリンク	別のファイルやディレクトリへの参照を示すファイルシステムのオブジェクト
スニペット	特定の機能やタスクを実行するための小さなコードの断片である。 開発者が効率的にプログラムを書くために再利用されることが多く、コードの繰り返し入力を省き、開発時間を短縮するのに役立つ。
トレーサビリティ	追跡可能性。過程や来歴などが追跡可能である状態
バイナリファイル	コンピュータが判読しやすい 2 進数で表現されたファイル
パターンマッチ/スニペットマッチ	他のプログラムと一致する部分を抽出する手法

パッケージマネージャー	ソフトウェアのインストール、更新、構成を管理するツール
ハッシュ値	元データをあらかじめ定められた計算手順により求められた値
ビッグデータ	従来のデータ処理方法では扱いきれない大量かつ多様なデータセットである。 これらのデータを解析することで、新たな知見や価値を生み出し、さまざまな分野での意思決定や戦略立案に活用される。
ビルドツール	実行ファイルを作成する工程を行うソフトウェア。
フィージビリティ	実現可能性
ブロックチェーン	分散型デジタル台帳技術である。取引データを暗号化してチェーン状に連結されたブロックに記録する。 改ざんが困難で透明性が高く、ビットコインなどの暗号通貨をはじめ、さまざまな分野での取引やデータ管理に利用されている。
プロプライエタリソフトウェア	開発元が所有権を保持し、ソースコードが公開されていない商業用ソフトウェアである。 利用や改変、再配布にはライセンス契約が必要であり、開発元がソフトウェアの使用条件や制限を定めている。
メタデータ	あるデータを説明するための情報データ
モジュール	ソフトウェアが提供する機能を実現するため部品
ライブラリ	特定の機能やタスクを実行するための再利用可能なコードの集合である。 開発者はライブラリを利用することで、共通の機能を効率的に実装し、アプリケーションの開発を迅速化することができる。
ランタイムライブラリ	プログラム実行時に必要となるモジュールの集まり

リポジトリ	データやプログラムを体系立てて保管するソフトウェア
欧州サイバーレジリエンス法案	欧州連合（EU）においてサイバーセキュリティを強化するための規制案である。 この法案は、デジタル製品およびサービスのセキュリティ要件を厳格化し、サイバー攻撃への対応能力を高めることを目的としている。
産業サイバーセキュリティセンター	（略称：ICSCoE）IPA 内の部門であり、産業分野におけるサイバーセキュリティの強化を目的としている。 企業のセキュリティ対策支援や情報共有、セキュリティ人材の育成を行っている。
脆弱性	コンピューターシステムやネットワーク、ソフトウェアに存在するセキュリティ上の欠陥や弱点である。 これらが悪用されると、不正アクセスやデータ漏洩、システムの停止などのセキュリティインシデントが発生する可能性がある。
中核人材プログラム	IPA が提供する、産業界で活躍するサイバーセキュリティの専門人材を育成するための教育プログラムである。 セキュリティ対策のリーダーシップを発揮できる人材の育成を目的としている。
電気通信情報局	米国大統領に対して電気通信および情報政策に関する助言を行う行政機関。
電子署名	文書やメッセージなどのデータの真正性を証明するために付加データ
独立行政法人情報処理推進機構	（略称：IPA）日本の情報技術の推進とセキュリティ対策を支援する政府機関である。

謝辞

本ガイドラインの作成にあたりまして、経済産業省 飯塚智様、株式会社日立ソリューションズ 渡邊歩様、森下大輔様、SOMPO ホールディングス株式会社 小中俊典様には SBOM の導入・運用に関する貴重なご意見を賜りました。いただいたご意見は、本ガイドラインの検討に大いに参考にさせていただきました。ここに厚く御礼申し上げます。

また検証を行うために、各製品ベンダーの方々に製品・ライセンス貸出しをしていただくとともに 多大なるご支援・ご尽力を賜りました。諸般の事情により 全ての方のお名前をこ

こに挙げることはできませんが お世話になりました皆様にこの場を借りて心より御礼申し上げます。

また産業サイバーセキュリティセンター中核人材育成プログラムの講師であられる満永拓邦先生、門林雄基先生には 本ガイドラインの元となる SBOM 導入・運用に関するガイドライン作成プロジェクトのメンター・講師として ご指導・ご助言とともに 各検証のご支援を賜り続けてきました。改めて御礼申し上げます。

そして本ガイドラインの作成や本プロジェクトをともに実施した 下記メンバーの皆様にも感謝を伝えたいと思います。

【リーダー】

繁田 大輝

【サブリーダー】

櫻井 健太　　間木平 伊織

【メンバー】

小山 千尋　　中野 貴裕　　永淵 亘

山崎 禎章　　山中 晋爾　　横道 太志

参考文献

[1] 経済産業省：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, 令和 5 年
https://www.meti.go.jp/press/2023/07/20230728004/20230728004-1-2.pdf

[2] 経済産業省：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. 2.0,令和 6 年
https://www.meti.go.jp/press/2024/08/20240829001/20240829001-1r.pdf

[3] 米国 NTIA：The Minimum Elements For a Software Bill of Materials (SBOM)
https://www.ntia.doc.gov/files/ntia/publications/sbom_minimum_elements_report.pdf

[4] 米国 NTIA：Survey of Existing SBOM Formats and Standards - Version 2021
https://www.ntia.gov/files/ntia/publications/sbom_formats_survey-version-2021.pdf

[5] 米国 NTIA：SBOM Myths vs. Facts
https://www.ntia.gov/files/ntia/publications/sbom_myths_vs_facts_nov2021.pdf

[6] IBM, 脆弱性管理とは,令和 6 年
https://www.ibm.com/jp-ja/topics/vulnerability-management

[7] 経済産業省：OSS の利活用及びそのセキュリティ確保に向けた 管理手法に関する事例集, 令和 4 年
https://www.meti.go.jp/press/2022/05/20220510001/20220510001-1-2.pdf