参考資料１

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

目次

1. 背景と目的

1.1. 背景

産業活動のサービス化に伴い、産業に占めるソフトウェアの重要性は高まる傾向にある。特に、近年は、産業機械や自動車等の制御にもソフトウェアの導入が進んでおり、IoT 機器・サービスや 5G 技術においても、汎用的な機器でハードウェア・システムを構築した上で、ソフトウェアにより多様な機能を持たせることで、様々な付加価値を創出していくことが期待されている。

ソフトウェアを利活用した製品・サービスの安全・安心を担保するには、利活用するソフトウェアの脆弱性の管理が求められる。企画・設計段階で脆弱性を含めないようソフトウェアが構成されていたとしても、製品出荷後に脆弱性が発見されることがあり、その場合、ソフトウェアを利活用する側でのソフトウェア更新等の対応が求められる。また、自社の製品・サービスで利活用しているソフトウェアの保守・サポートが終了する場合、それ以降に発見された脆弱性の管理について代替ソフトウェアへの変更を含めた検討が求められる。しかしながら、ソフトウェアサプライチェーンが複雑化し、オープンソースソフトウェア（OSS）の利用が一般化する中で、自社製品において利用するソフトウェアであっても、コンポーネントとしてどのようなソフトウェアが含まれているのかを把握することが困難な状況がある。組織内の IT システムで利用されているソフトウェアを資産管理している組織は多いが、開発者が直接利用している上位のコンポーネントのみが資産管理の対象となり、直接利用のコンポーネントに内包されて間接的に利用される下位のコンポーネントの多くは資産管理の対象外となっている。したがって、脆弱性情報と資産管理台帳を照らし合わせるだけでは、下位のコンポーネントとして利用される OSS のようなコンポーネントにおいて脆弱性が発見された場合に、間接的な脆弱性の影響を検知することができない。

このようなソフトウェアの脆弱性管理に関し、ソフトウェアの開発組織と利用組織双方の課題を解決する一つの手法として、Software Bill of Materials（SBOM：エスボム）を用いた管理手法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も含めた機械処理可能な一覧リストのことで、日本語では「ソフトウェア部品表」とも呼ばれる。SBOM は、米国商務省の電気通信情報局（NTIA）が 2018 年 7 月より開始した実証を通じて注目され始め、2021 年 5 月に米国バイデン大統領が署名した大統領令[1]を一つの起点として、世界的に普及が進みつつある。Linux Foundation が 2021 年の第 3 四半期にグローバルの 412 の組織を対象に実施した調査[2]

---

[1] Executive Order on Improving the Nation's Cybersecurity
https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

[2] Linux Foundation, The State of Software Bill of Materials (SBOM) and Cybersecurity Readiness
https://www.linuxfoundation.org/tools/the-state-of-software-bill-of-materials-sbom-and-cybersecurity-readiness/
（日本語版） https://www.linuxfoundation.jp/blog/2022/05/the-state-of-software-bill-of-materials-sbom-and-cybersecurity-readiness/

では、調査対象の 48%の組織が SBOM を導入していることが明らかになったほか、Linux Foundationは、調査対象組織のSBOM準備状況・計画状況を踏まえ2022年には78%、2023年には88%の導入率になると推測している。

国内では、経済産業省において「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」（ソフトウェアタスクフォース）を2019年9月より開催し、SBOMも含めたソフトウェア管理手法等に関して幅広い議論を行ってきた。ソフトウェアタスクフォースの議論を通じ、SBOM 導入に向けて検討すべき課題として、SBOM 導入による費用対効果の課題、サプライチェーン上でのSBOM 共有に関する課題、SBOM 管理に当たっての契約に関する課題、中小企業における SBOM 導入に関する課題等が提起された。このような課題を踏まえ、経済産業省は SBOM 導入に向けた実証を 2021 年以降に実施し、SBOM 導入にかかるコストや効果の評価を複数の産業分野で実施した。2021 年度の実証では自動運転システム開発向け検証基盤ソフトウェアを対象とし、2022 年度の実証では医療機器分野として歯科用 CT、自動車分野として自動車ヒーターコントローラー、ソフトウェア分野としてネットワーク脅威検知ソフト等を対象とした。これらの実証を通じて、以下に示すようなSBOMのメリット・効果が確認でき、特にソフトウェアの脆弱性管理のメリットやライセンス管理のメリットがあり、その結果、開発生産性向上のメリットが得られることが分かった。

- 手作業でのコンポーネント管理の工数と SBOM を活用した管理の工数を比較した場合、SBOM を活用した管理の方が工数は小さくなる。SBOM を導入する際、初期工数は大きいものの、SBOM ツール[3]を活用することで負担は軽減される。
- SBOM を作成・管理することで、ソフトウェアに含まれているコンポーネントの脆弱性が発見された際の影響有無の特定までのリードタイムを短縮可能であり、脆弱性が残留するリスクの低減や脆弱性対応工数の低減につながる。特に、有償の SBOM ツールを活用することで、OSS 間の依存関係や OSS の再帰的な利用（再利用部品）も効率的に検出・管理できる。
- SBOM を作成・管理することで、ソフトウェアに含まれているコンポーネントのライセンス情報を確認でき、コンプライアンス上の過失を防ぐことができるため、ライセンス違反リスクの低減やライセンス管理工数の低減につながる。特に、SBOM ツールを活用することで、各ライセンスの内容表示や注意が必要なライセンスの警告等、コンプライアンス遵守のための機能が活用できるため、より効率的なライセンス管理が可能となる。

他方で、SBOM 導入に関して以下に示すような課題が明らかとなった。

- 対象とするソフトウェア・システムの全体構成が把握できていない場合、SBOM ツールの適用範囲を適切に設定できず、効果的なリスク管理が実施できない。
- SBOMツールを導入するための環境整備や学習に工数を要する。
- 無償の SBOM ツールは、環境整備や学習に当たっての情報が不足しており、導入に大きな工

[3] 本手引では、SBOM の作成、共有、活用、管理することができるツールを総称して「SBOM ツール」と呼ぶ。SBOM 管理ツール、OSS 管理ツール、ソフトウェア構成解析（SCA）ツール等と呼ばれることもある。

数を要する。また、再帰的に利用される部品が十分に検出できない、取扱い可能な SBOM フォーマットに制限がある、ライセンスの検知漏れが発生する等、利用時に注意すべきことが多い。

- SBOM ツールを単に適用しただけでは、対象ソフトウェアに含まれるコンポーネントの検知漏れが発生する場合がある。
- SBOM ツールの出力結果について、コンポーネントの誤検出や検出漏れ、脆弱性情報の誤り等が発生する場合があるため、出力結果の精査が必要となる。
- サードパーティのコンポーネントについて、内部の構成や活用されている技術を把握できない状況で SBOM ツールの出力結果を精査することになるため、精査にかかる工数が大きくなる。
- 現状では、異なる SBOM ツールで生成した SBOM を読み込んで脆弱性管理に活用することができる SBOM ツールが少なく、異なる SBOM ツール間での SBOM の相互共有が困難である。
- 特定されたコンポーネントの脆弱性に対する対応要否・対応優先度の判断が困難である。

総論として、SBOM を活用することで、効率的なソフトウェア管理を実施できることが確認できた一方で、実際の SBOM 導入に際しては様々な課題が存在することが明らかとなった。

1.2. 目的

本手引では、ソフトウェア管理に向けた SBOM の作成・共有・運用・管理に関する様々な課題の解決するために、SBOM の概要や SBOM 導入のメリット等、SBOM に関する基本的な情報を提供するとともに、ソフトウェアサプライヤーにおける SBOM 作成に向けた環境構築・体制整備、SBOM 作成・共有、そして SBOM の運用・管理に至る一連の SBOM 導入に向けたプロセスを示す。そして、企業における効率的・効果的な SBOM 導入を支援するために、各フェーズにおける主な実施事項や SBOM 導入に当たって認識しておくべきポイントを示す。本手引は主にソフトウェアサプライヤーを対象としたものであるが、ソフトウェアを調達して利用する企業等においても、活用・参照することが可能である。なお、SBOM はソフトウェア管理の一手法であるため、作成することが目的ではなく、SBOM を用いたソフトウェアの適切な管理が重要となることに留意が必要である。近年のソフトウェア開発における OSS 活用増加の傾向を受け、ソフトウェアのセキュリティ対策に当たっては OSS の管理も重要となるが、OSS の管理に関するドキュメントとして、経済産業省は「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」[4]を公開しているため、本手引と合わせて参照することが望まれる。

[4] 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
https://www.meti.go.jp/policy/netsecurity/wg1/ossjirei_20220801.pdf

1.3. 主な対象読者

本手引では、ソフトウェアサプライヤーにおける開発・設計部門や製品セキュリティ担当部門（PSIRT 等）等のソフトウェアセキュリティに関わる部門と、経営層を主な対象としている。ソフトウェアセキュリティに関わる部門に対して、本手引では、ソフトウェア管理に向けた SBOM 導入の際に活用できる SBOM 導入に向けたプロセス、SBOM 導入に向けた主な実施事項及び SBOM 導入に当たって認識しておくべきポイントを記載している。また、ソフトウェア管理の一手法としての SBOM が、経営層に十分に認識されていないと考えられる場合には、「1.6 本手引のサマリー」を活用し、経営層と適切なコミュニケーションを行うことが期待される。経営層に対して、本手引では、SBOM 導入に関する意思決定を行う際に参照できる SBOM の効果・メリットや、SBOM に関する誤解と事実を示している。SBOM 導入に関する意思決定を行う際、「1.6 本手引のサマリー」の内容は認識しておくことが期待される。

いずれの内容も、ソフトウェアにおける脆弱性管理に課題を抱えている組織、SBOM という用語は聞いたことがあるが具体的な内容やメリットは把握できていない組織、SBOM の必要性は理解しているが、導入に向けた取組内容が認識できていない組織等、主に SBOM 初級者に向けた内容となっている。関連して、本手引のライセンス管理に関する内容は、組織の法務・知財部門でも活用できるほか、全般的な内容は、ソフトウェアサプライヤーに限らず、ソフトウェアを調達して利用する企業においても一部活用可能である。

1.4. 主な対象ソフトウェア

本手引では、主にパッケージソフトウェアや組込みソフトウェアに関する SBOM について、SBOM 導入に向けたプロセスを示すとともに、各プロセスにおける主な実施事項や SBOM 導入に当たって認識しておくべきポイントを記載する。

1.5. 活用方法

SBOM を導入する組織は、本手引に基づき、SBOM に関する基本情報を認識するとともに、SBOM 導入に向けたプロセスを確認することが望まれる。そして、各ステップにおける主な実施事項及び SBOM 導入に当たって認識しておくべきポイントを確認しつつ、SBOM 導入を進めることが期待される。なお、付録の 7.1 では、SBOM 導入の各ステップにおける実施事項をチェックリストとしてまとめているため、SBOM 導入に向けた取組時にあわせて参照することが望まれる。

1.6. 本手引のサマリー

《本手引のポイント》

- 企業経営へ影響を及ぼしうるソフトウェアのセキュリティ脅威が近年急激に増大している
- 脅威に対し、ソフトウェア管理の一手法である Software Bill of Materials（SBOM：エスボム）が注目を集めており、導入企業が世界的に増加している
- SBOM の活用により、ソフトウェアの脆弱性やライセンスの管理のリスク及びコストを低減可能である
- 本手引を活用し、ソフトウェア管理に向けた SBOM 導入の取組を進めることが期待される

《本手引の背景・概要》

【ソフトウェアサプライチェーンの脅威】

- ✓ ソフトウェアサプライチェーンが複雑化し、オープンソースソフトウェア（OSS）の利用が一般化する中で、ソフトウェアに対するセキュリティ脅威が近年急激に増大している。2021 年 12 月に発見された Apache Log4j の脆弱性は世界中に影響を及ぼしたほか、2019 年から 2022 年にかけてのソフトウェアサプライチェーン攻撃の年平均増加率が 742%に達したというデータもある。
- ✓ ソフトウェアに対するセキュリティ脅威は企業経営へ大きな影響を及ぼす。例えば、SolarWinds のサイバー攻撃の影響を受けた企業は、平均して年間収益額の約 11%の損害を被ったというデータもあるほか、製品に脆弱性が残存することで製品回収や販売停止につながった事例もある。
- ✓ ソフトウェアに対する脅威の状況に対し、ソフトウェアに含まれる脆弱性を適切に管理し、脆弱性が明らかになった際に迅速に対応するといったソフトウェア管理を効率的・効果的に実施していくことが重要である。

【ソフトウェア管理における SBOM 活用のメリット】

- ✓ このようなサプライチェーンを通じて開発されるソフトウェアの管理を効率化するための一手法として、Software Bill of Materials（SBOM：エスボム）を用いた管理手法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も含めた機械処理可能な一覧リストのことで、世界的に導入企業が増加しているほか、医療機器分野等の一部の分野では SBOM の推奨がされる等、規制や制度化も検討され始めている。
- ✓ 情報量が膨大となるソフトウェア管理に対し、機械処理可能な SBOM を導入することで、ソフトウェア管理に要する対応コストや人的コストを低減することができ、これにより開発生産性向上に繋がる。事実、経済産業省が実施した医療機器分野を対象とした実証では、SBOM を活用した脆弱性管理を行うことで、手動での管理と比較して、管理工数が 70%程度低減した。
- ✓ また、脆弱性管理上のメリットとして、SBOM を作成し、継続的に管理することで、ソフトウェアの透明性を高め、脆弱性残留リスクの低減が期待されるほか、サプライチェーンを通じた脆弱性対応の効率化にも繋がる。
- ✓ さらに、ライセンス管理上のメリットとして、SBOM を導入し、OSS のライセンス情報を管理することで、

ライセンス違反リスクの低減にも寄与する。

【本手引の活用ポイント】

- ✓ 本手引では、SBOM に関する基本的な情報を提供するとともに、企業の効率的・効果的な SBOM 導入を支援するために、SBOM 導入に向けた主な実施事項及び SBOM 導入に当たって認識しておくべきポイントを示す。
- ✓ 効率的・効果的なソフトウェア管理に向け、本手引を活用し、経営層においては、SBOM 導入に関する意思決定を行うとともに、ソフトウェアセキュリティに関わる部門においては、SBOM 導入に向けた具体的な取組を進めることが期待される。

コラム：ソフトウェアのセキュリティ脅威に関する重要指数

ソフトウェアサプライチェーンが複雑化し、オープンソースソフトウェア（OSS）の利用が一般化する中で、ソフトウェアに対するセキュリティ脅威が近年拡大している。以下では、近年のソフトウェアのセキュリティ脅威に関する現状を示す重要な数値について紹介する。

***81%*：脆弱性が含まれるコードベースの割合**

2022 年に Synopsys 社が発表した 2,409 のコードベースを対象とした調査結果によれば、OSS を含むコードベースの割合は 97%であった。そのうち、81%のコードベースに少なくとも一つの脆弱性が含まれていた[5]。

***62%*：2021 年にソフトウェアサプライチェーン攻撃を受けた企業の割合**

2022 年に Anchore 社が発表した北米・EU・英国の企業 428 社を対象とした調査結果によれば、過去 1 年間でソフトウェアサプライチェーン攻撃の影響を受けた企業は 62%であった[6]。

***+742%*：2019 年から 2022 年にかけてのソフトウェアサプライチェーン攻撃の年平均増加率**

2023 年に Sonatype 社が発表した調査結果によれば、2019 年から 2022 年の 3 年間でのソフトウェアサプライチェーン攻撃の年平均増加率は 742%であり、2022 年には 88,000 件を超えた。2015 年 2 月～2019 年 6 月までの攻撃件数は 216 件であり、近年、ソフトウェアサプライチェーン攻撃の件数が指数関数的に増加している[7]。

***-11%*：SolarWinds のサイバー攻撃による企業収益額への影響（損害）**

2021 年に IronNet 社が発表した米国・英国・シンガポールの企業 473 社を対象とした調査結果によれば、85%の企業が SolarWinds のサイバー攻撃の影響を受けた、それら企業は、平均して年間収益額の約 11%の損害を被った[8]。

---

[5] Synopsys, 2022 Open Source Security and Risk Analysis Report
https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html

[6] Anchore, 2022 Security Trends: Software Supply Chain Survey
https://anchore.com/blog/2022-security-trends-software-supply-chain-survey/

[7] Sonatype, 8th Annual State of the Software Supply Chain Report
https://www.sonatype.com/state-of-the-software-supply-chain/introduction

[8] IronNet, 2021 Cybersecurity Impact Report
https://www.ironnet.com/hubfs/IronNet-2021-Cybersecurity-Impact-Report-June2021.pdf

2. SBOM の概要

2.1. SBOM とは

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一覧リストである。SBOM には、ソフトウェアに含まれるコンポーネントの名称やバージョン情報、コンポーネントの開発者等の情報が含まれ、OSS だけではなく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できる。また、SBOM をソフトウェアサプライチェーンの上流から下流に向かって組織を越えて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を高めることが期待されており、特に、コンポーネントの脆弱性管理の課題に対する一つの解決策として期待されている。

SBOM のイメージをより具体化するために、以下のような簡易的なシナリオを考える。

- A 社は、B 社の Browser とコミュニティ P の Protocol という 2 つのコンポーネントを使用して、Application というソフトウェアを開発した。
- B 社の Browser は、C 氏が開発した Compression Engine のコンポーネントを使用している。
- B 社は、Browser に関する SBOM を自社で作成し、A 社に共有した。ただし、C 氏やコミュニティ P のコンポーネントに関する SBOM 情報を取得できなかったため、A 社にて、C 氏とコミュニティ P のコンポーネントの SBOM を作成した。

このシナリオにおけるプレイヤーやコンポーネントの関係性は図 2-1 のように表現できる。この図に示すとおり、多くの SBOM のエンティティは、ソフトウェアのサプライヤーの役割だけでなく、他者から共有された SBOM を利用する役割も担う。すなわち、別のエンティティから入手した SBOM の情報を活用するだけでなく、新たに開発したコンポーネントに関連する SBOM を作成し、その他のエンティティに SBOM を共有する役割も担うことがある。なお、ソフトウェアコンポーネントに関するサプライヤーと SBOM 作成者は一致することが理想的であるが、SBOM が完全に普及していない現状では必ずしも一致しない。今回のシナリオでは、B 社は自社内で SBOM を作成しているため、Browser のコンポーネントに関するサプライヤーと SBOM 作成者は一致しているが、Protocol の場合、コミュニティ P では SBOM 作成を行わず A 社にて SBOM 作成を行ったため、サプライヤーはコミュニティ P となるが、SBOM 作成者は A 社となる。

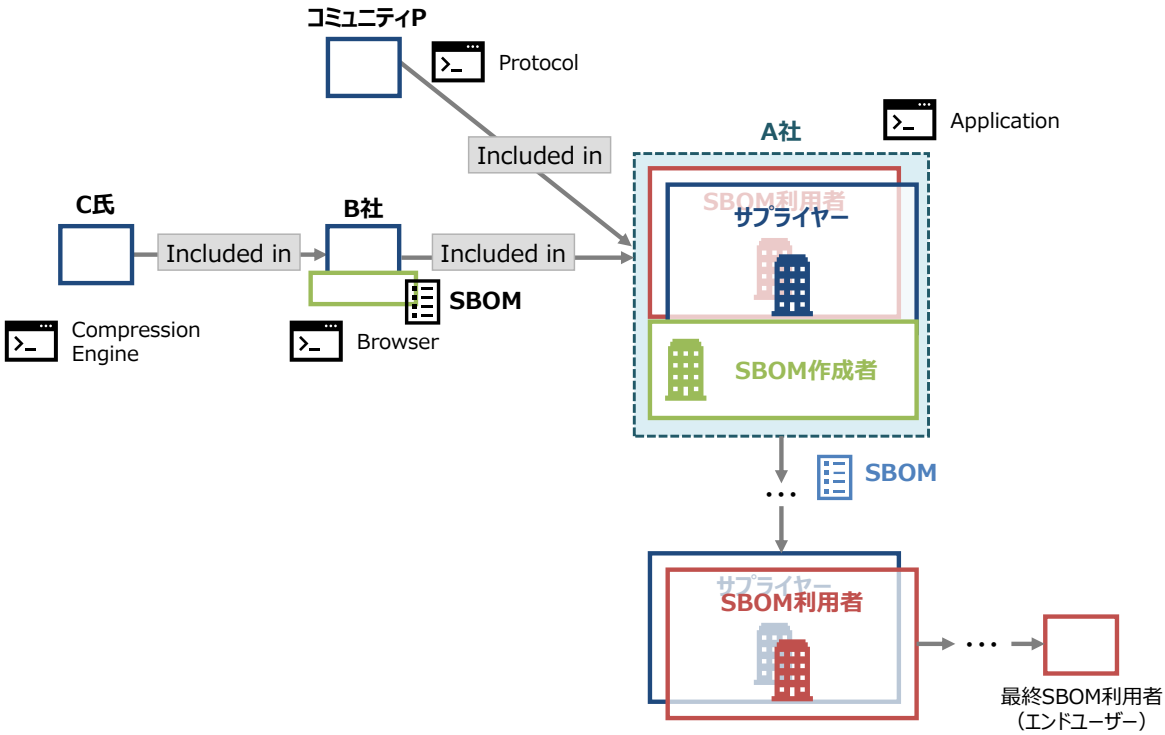

図 2-1 簡易シナリオにおけるプレイヤー間の関係性

上記のシナリオにおいて、A 社が作成する SBOM の概念的イメージは表 2-1 のように与えられる。このイメージでは、各コンポーネントについて、そのサプライヤーやバージョン、コンポーネント間の依存関係、SBOM 作成者等が一覧化されている。SBOM を作成することで、各コンポーネントがいつ、誰によって開発され、他のコンポーネントとどのような依存関係があり、当該コンポーネントに関する SBOM が誰によって作成されたかを特定・管理することが可能となる。これにより、特定のコンポーネントの脆弱性が明らかになったとき、その脆弱性の影響を受けるコンポーネントが含まれているかを即座に認識することができ、脆弱性に対する迅速な対応を行うことができる。そして、SBOM が組織を越えて相互共有されることで、各コンポーネントに関する情報が可視化されることとなり、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こととなる。

表 2-1 簡易シナリオにおける SBOM の概念的イメージ

ID	サプライヤー名	コンポーネント名	コンポーネントのバージョン	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
1	Company A	Application	1.1	234	Primary	Company A	05-09-2022 13:00:00
2	Company B	Browser	2.1	334	Included in #1	Company B	04-18-2022 15:00:00
3	Mr. C	Compression Engine	3.1	434	Included in #2	Company A	05-09-2022 13:00:00
4	Community P	Protocol	2.2	534	Included in #1	Company A	05-09-2022 13:00:00

上記の簡易シナリオに基づく表 2-1 の SBOM はあくまでイメージであり、このレベルの記載内容であれば、あえて SBOM として管理する必要はないかもしれない。しかしながら、実際のソフトウェアは、図 2-1 で描かれるようなシンプルなサプライチェーン構造ではなく、複雑な構造の下で開発される。また、自社で開発したプロプライエタリソフトウェアだけでなく、他者が開発したコンポーネントも含まれ、それぞれが複雑な依存関係を持つこととなる。したがって、ソフトウェアのリスク管理やソフトウェアサプライチェーンの透明性を高めるために、SBOM を用いて、ソフトウェアに含まれるコンポーネントの情報をその依存関係も含めて管理することが重要となる。

コラム：SBOM と食品表示とのアナロジー

SBOM は、食品の包装に記載されている食品表示に類似している。食品に含まれる原材料を可視化した食品表示を見ることで、アレルギー事故等による健康危害防止や食の禁忌への対応が可能となる。マカロニサラダを例として考えたとき、食品サプライチェーンを通じた製造・加工等の結果、各原材料を含んだ図 2-2 に示すような食品表示が作成される。SBOM は、食品表示のようにソフトウェアに含まれるコンポーネントの情報を示したリストであり、これらの情報が可視化されることで、脆弱性への対応やリスク管理が容易となる。食品表示が食品サプライチェーンの透明性向上に寄与していることと同様に、SBOM も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ものである。ただし、食品表示と異なり、SBOM はコンポーネント名だけでなく、そのバージョンや依存関係についても記載するため、食品表示より複雑なリストとなることに留意したい。また、多くの SBOM は作成後も動的に変更されるため、SBOM 利用者での管理が重要となることに留意したい。

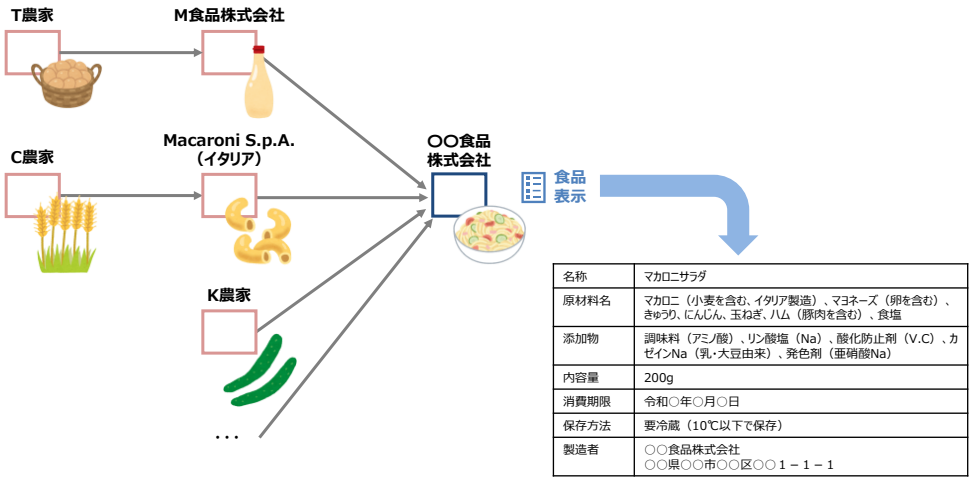

図 2-2 食品サプライチェーン及び食品表示の概念的イメージ

2.2. SBOM 導入のメリット

SBOM 導入による代表的なメリットとして、表 2-2 に示すとおり、脆弱性管理のメリット、ライセンス管理のメリット、そして、開発生産性向上のメリットの 3 つが挙げられる。それぞれのメリットにおいて、SBOM 導入組織に対する脆弱性管理・ライセンス管理・開発生産性向上の直接的なメリットのほか、製品価値や企業価値等に対する間接的なメリットも存在する。

表 2-2 SBOM 導入の主なメリット

メリット区分		メリット項目	主な内容
脆弱性管理のメリット	直接的メリット	脆弱性残留リスクの低減	脆弱性に関する情報を収集し、SBOM の情報と突合して脆弱性を検出することで、ソフトウェアにおいて脆弱性が残留するリスクを低減できる。
		脆弱性対応期間の短縮	SBOM ツール等を用いることにより新たな脆弱性をリアルタイムで検出し、影響を判断することで、初動期間を短縮できる。
		脆弱性管理にかかるコストの低減	SBOM ツールを用いた自動管理により、手動での管理と比較して、管理コストを低減できる。
	間接的メリット	製品価値・企業価値向上	製品に含まれる脆弱性の低減や脆弱性対応の迅速化により、製品や企業の価値が向上する。
		サイバー衛生の向	脆弱性の少ない製品が増えることで、サイバ

メリット区分		メリット項目	主な内容
		上（Cyber Hygiene）	ー空間全体のセキュリティが向上する。（踏み台悪用により攻撃を受けるリスクが低減できる。）
ライセンス管理のメリット	直接的メリット	ライセンス違反リスクの低減	OSS の特定漏れによるライセンス違反のリスクを低減できる。
		ライセンス管理にかかるコストの低減	SBOM ツールを用いた自動管理により、手動での管理と比較して、管理コストを低減できる。
	間接的メリット	製品価値・企業価値向上	製品のライセンス違反リスクの低減により、製品や企業の価値が向上する。
開発生産性向上のメリット	直接的メリット	開発遅延の阻止	コンポーネントに関する問題を早期に特定することで、開発遅延の発生を防ぐことができる。
		開発にかかるコストの低減	コンポーネントに関する問題を早期に特定することで、対応コストを低減できる。
		開発期間の短縮	使用するコンポーネントを選定する際、類似製品に関する過去の SBOM を参照することで、選定に関する工数を削減できる。

SBOM 導入のメリットのうち、最も注目されているのが脆弱性管理のメリット、すなわち、ソフトウェアにおける脆弱性を検出し、優先度付けを行った上で修正及び軽減するといった一連の脆弱性対応プロセスにおけるメリットである。近年のソフトウェアの多くは複雑なサプライチェーン構造の下で開発され、自社で開発したプロプライエタリソフトウェアだけでなく、他社や OSS コミュニティが開発したコンポーネントも多く含む。そして、これらのコンポーネントが複雑な階層構造や依存関係を持つことが多い。例えば、ある Java アプリケーションが Apache Log4j をコンポーネントとして用いている場合、Log4j は下位のコンポーネントに位置づけられ、通常のコンポーネント管理では特定が難しい場合がある。しかしながら、下位のコンポーネントであっても、当該コンポーネントが脆弱性を含んでいた場合にセキュリティ上の影響を受ける可能性がある。

脆弱性残留リスクの低減のために、利用しているコンポーネントに関する情報に基づき、脆弱性の継続的な監視を有効に実施することが重要となる。この点、SBOM を導入し、各コンポーネントについて脆弱性データベースとの突合を行うことで、既知の脆弱性の影響を受けないかを効率的に確認することができ、結果として、ソフトウェアにおいて脆弱性が残留するリスクを低減できる。また、あるコンポーネントについて新たな脆弱性が明らかになったとき、SBOM を用いた管理を行っていない場合、自社のソフトウェアがそのコンポーネントを含んでいるか否かも分からず、知らぬ間に脆弱性の影響を受ける可能性もある。図 2-3 に示すとおり、SBOM を導入していれば、あるコンポーネントにおいて脆弱性が明らかになった場合に、その

脆弱性の影響を即座に認識することができ、脆弱性対応にかかる期間を短縮することができる。加えて、パートナー企業、脆弱性の影響を受ける組織、ソフトウェア利用者等とソフトウェアに関する情報を組織間で共有することで、脆弱性対応にかかる期間を短縮できるほか、サプライチェーン上で第三者によりコンポーネントの書換えや不正な追加が行われた場合にも、その実態把握に資するものとなる。また、組織間共有にあたって SBOM を活用することで、ソフトウェア情報に関する共有に要する工数を低減することができる。

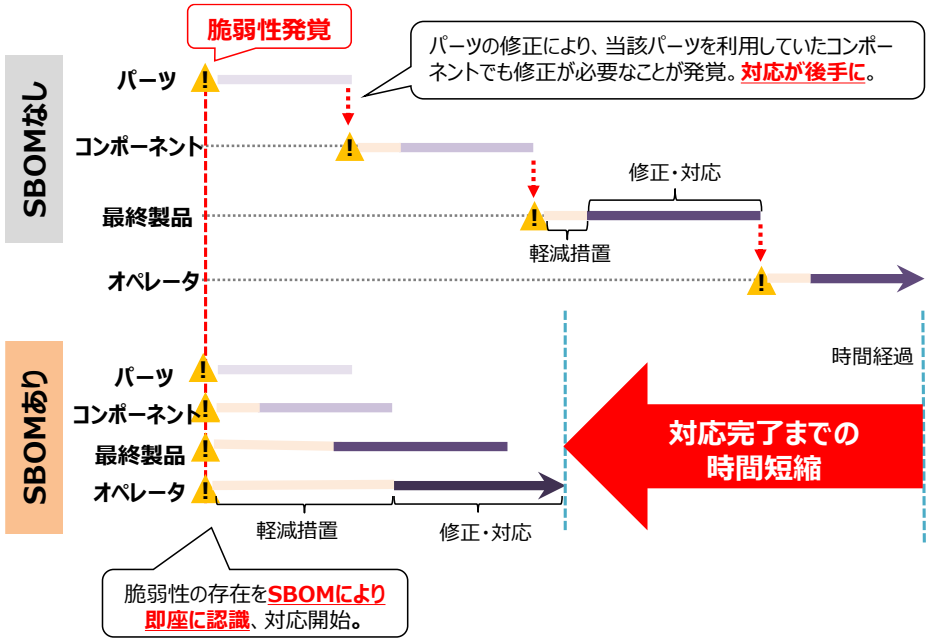

図 2-3 SBOM 導入による脆弱性対応期間短縮のメリット

そして、SBOM を導入することで、脆弱性管理にかかるコストを低減することができる。2022 年度に実施した医療機器分野の実証における SBOM を用いた脆弱性管理のコスト評価結果を図 2-4 に示す。ここでは、対象とするソフトウェアが約 80 のコンポーネントを有しており、工数単価について¥10,000/時間と仮定している。また、SBOM による管理は、SBOM ツールを用いて実施した。手動でのコンポーネント管理の場合、手動でコンポーネントのリストを洗い出す必要があるほか、各コンポーネントに脆弱性が含まれるか、脆弱性情報データベース（NIST NVD 等）を手動で検索して確認する必要がある。そして、脆弱性が明らかになった際に各コンポーネント情報と脆弱性情報とを突合して影響有無を確認する必要がある。他方、SBOM による管理の場合、SBOM ツールの環境整備やツールの学習のための工数が必要となるが、コンポーネントの解析・特定を自動で行うことができるため、コンポーネントの解析・特定自体にはほとんど工数はかからない。新たな脆弱性が明らかになった場合、SBOM ツールに自動で反映され、その影響を受けるかをリアルタイムで特定することができるため、解析・特定結果に関する確認は必要となるものの、脆弱性管理にかかるコストを大幅に削減することができる。実証では、SBOM を活用した場合に要する工数が、手動での脆弱性管理と比較して 30%

程度に低減されたことを確認した。なお、有償の SBOM ツールを用いた場合はツールのコストも追加される形となるが、対象とするコンポーネント数が多ければ多いほど、そのコストは按分されることに留意したい。

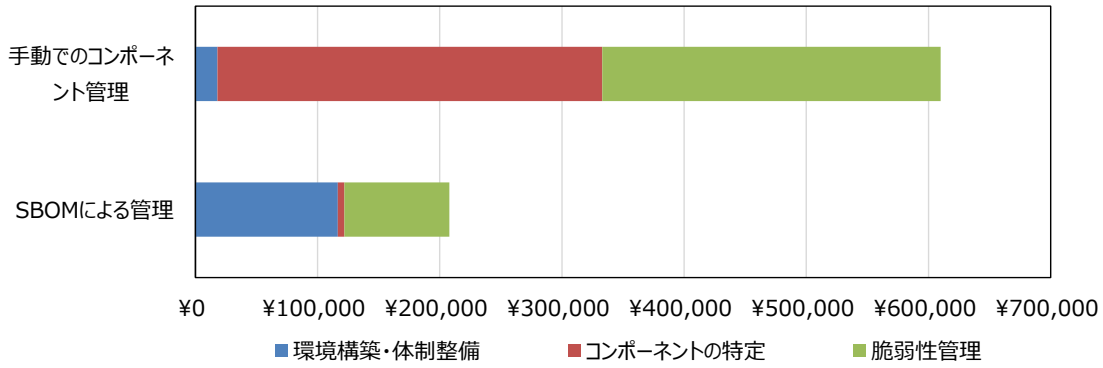

図 2-4 SBOM 管理による脆弱性管理コストの低減結果
（医療機器分野における 2022 年度の実証結果より）[9]

SBOM 導入による脆弱性管理に対する間接的なメリットとして、脆弱性のリスクが低減されることで製品価値や企業価値が向上するほか、大局的には、脆弱性の少ない製品が増えることで、サイバー空間全体のセキュリティが向上するというメリットも存在する。

SBOM 導入によるメリットの 2 つ目として、ライセンス管理上のメリット、すなわち、ソフトウェアに含まれるコンポーネントのライセンスを識別し、各ライセンスの要求事項に従った対応をするという一連のプロセスにおけるメリットが挙げられる。近年のソフトウェアの多くは OSS を含んでいるが、OSS のライセンスに違反した場合、ソフトウェアの販売停止や回収、罰金の支払い、企業ブランドイメージの低下等、大きな影響を受ける可能性がある。海外では、OSS ライセンス違反による訴訟事例が複数存在し、例えば、2010 年に家電メーカーら 14 社が GNU General Public License（GPL）違反で起訴された事例、2013 年にメディアプレイヤーメーカーが GPL 違反で起訴された事例、2021 年にテレビメーカーが GPL 違反で起訴された事例等が挙げられる。OSS を利用する場合、ライセンスの種類に応じた適切な対応が必要であり、例えば GPL の場合、派生物も GPL が適用されるほか、GPL を他のソフトウェアと組み合わせて新たなソフトウェアを作成した場合、当該ソフトウェアにも GPL が適用される。また、Mozilla Public License（MPL）の場合、GPL と同様に派生物も MPL が適用される一方で、組み合わせて作成した新たなソフトウェアに対しては MPL が適用されない。そのため、OSS を利用する場合、自らの責任ですべての OSS のライセンスを確認し、それぞれのライセンスに準拠する必要があるが、OSS のライセンス情報を抜け漏れなく管理することは容易ではない。SBOM を導入し、ライセンス情報も含めてコンポーネントを管理することで、ライセンス違反のリスクを低減することができるほか、脆弱性管理と同

[9] SBOM ツールのコストは含まず。また、「脆弱性管理」について、脆弱性の特定とリスク評価に関する工数までを考慮し、工数が SBOM の有無によって大きく変動しない脆弱性の修正作業や報告作業等は含めていない。

様に、ライセンス管理にかかるコストを低減することができる。さらには、ライセンス違反に起因する財務リスクから組織を保護することができ、製品価値や企業価値の向上に寄与する。

SBOM 導入によるメリットの 3 つ目として、ソフトウェア開発ライフサイクル（SDLC）が改善し、開発生産性が向上するというメリットが挙げられる。ソフトウェア開発初期段階から SBOM を生成することで、コンポーネントに含まれる既知の脆弱性やライセンスの問題等のコンポーネントに関する問題にあらかじめ対応することができる。これらの問題を早期に特定することで、開発遅延の発生を防ぐことができるほか、対応コストを低減することができる。また、社内で利用が承認されたコンポーネントの情報を SBOM として管理しておくことで、開発の際に毎度コンポーネントを調査・承認する必要がなくなり、結果として開発工数の低減が期待できる。開発生産性向上のメリットに関して、Linux Foundation が 2021 年の第 3 四半期にグローバルの 412 の組織を対象に実施した調査[10]によれば、SBOM のメリットについて、回答組織の 51%が「開発者がより広範で複雑なプロジェクト間の依存関係を理解しやすくなる」ことを挙げており、これは脆弱性管理のメリット（同 49%）やライセンス管理のメリット（同 44%）より高い割合となっている。

本節では、SBOM 導入による代表的なメリットとして、脆弱性管理のメリット、ライセンス管理のメリット、開発生産性向上のメリットの 3 つを挙げたが、それ以外にも想定されるメリットは存在する。例えば、SBOM による管理を行うことで、ソフトウェアの EOL 管理が容易となることも挙げられる。

コラム：Log4j の脆弱性（Log4Shell）に対する SBOM 導入の効果

2021 年 12 月、ログ出力ライブラリの Apache Log4j において任意コード実行の脆弱性（通称：Log4Shell）が発見された。OSS の Log4j は無償で利用可能であり、様々な機能が内包されていたことから、Java システムにおけるログ出力の定番的なモジュールとして様々な用途に用いられていた。しかしながら、発見された脆弱性を悪用し、Log4j が動作するアプリケーションに対して不正アクセスを行うことで、情報漏えいやマルウェア感染等の被害につながるおそれがある。米国 CISA 等が発表した「2021 年に頻繁に悪用された脆弱性」[11]では、2021 年 12 月に発見された脆弱性に関わらず Log4Shell が 1 位にランクインしており、この脆弱性の影響範囲は計り知れない。

Log4Shell の脆弱性が広く悪用されている理由として、多数のソフトウェアに導入されていることや攻撃が容易であることのほかに、コンポーネントとして組み込まれているため、サプライヤーやソフトウェア利用者が脆弱性の影響に気づかず、対策が実施されていないことも挙げられる。具体的には、図 2-5 に示すように、ソフトウェア利用者が確認できる（認識している）コンポーネントの範囲より深くに Log4j のコンポーネントが存在する場合、ソフトウェア利用者は認識していないものの Log4j の脆弱性が悪用され、ソフトウェア利用者に影響を及ぼす可能性がある。

---

[10] 脚注 2 参照。

[11] CISA, Alert (AA22-117A) 2021 Top Routinely Exploited Vulnerabilities
https://www.cisa.gov/uscert/ncas/alerts/aa22-117a

複数階層のコンポーネントを含む SBOM を導入することで、Log4j の脆弱性が発見された際に、利用しているソフトウェアが影響を受けるかを即座に確認でき、脆弱性対応を迅速化することができる。これにより、脆弱性が悪用されることのリスクが低減されるほか、脆弱性対応や影響範囲の特定に要するコスト低減にも寄与する。

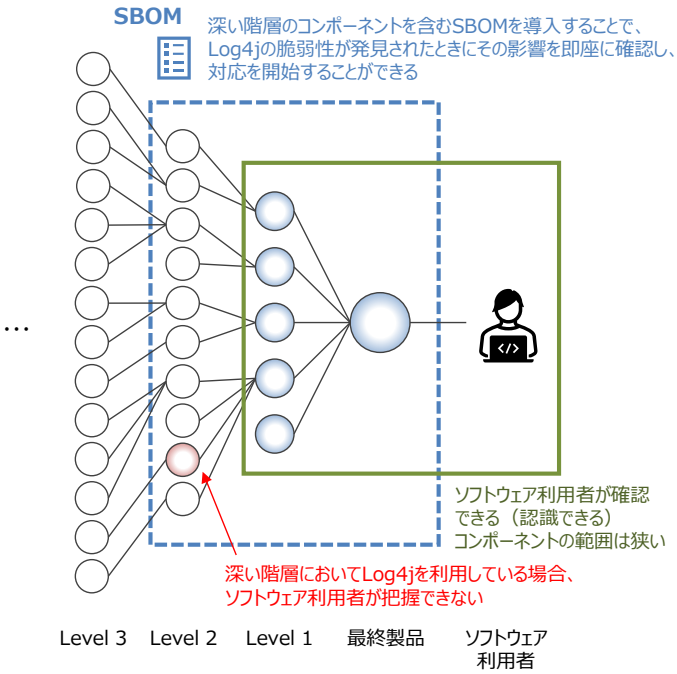

図 2-5 ソフトウェアのコンポーネント階層のイメージ

2.3. SBOM の「最小要素」

2021 年 5 月の米国大統領令を受け、NTIA は 2021 年 7 月に SBOM の「最小要素」の定義に関する文書を公開した[12]。NTIA が定める「最小要素」の定義には、SBOM に含めるべき情報に関するカテゴリーである「データフィールド」だけではなく、SBOM を導入する組織が考慮すべきカテゴリーとして「自動化サポート」、「プラクティスとプロセス」のカテゴリーも規定されている。具体的な「最小要素」のカテゴリーと定義は表 2-3 に示すとおりである。

[12] NTIA, The Minimum Elements For a Software Bill of Materials (SBOM)
https://www.ntia.doc.gov/report/2021/minimum-elements-software-bill-materials-sbom

表 2-3 米国 NTIA による SBOM の「最小要素」の定義

カテゴリー名称	概要	定義
データフィールド（Data Fields）	各コンポーネントに関する基本情報を明確化すること	以下の情報を SBOM に含めること。 ● サプライヤー名 ● コンポーネント名 ● コンポーネントのバージョン ● その他の一意な識別子 ● 依存関係 ● SBOM 作成者 ● タイムスタンプ
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こと	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され、共有されること。現状では、国際的な議論を通じて策定された、SPDX、CycloneDX、SWID タグを用いること。
プラクティスとプロセス（Practices and Processes）	SBOM の要求、生成、利用に関する運用方法を定義すること	SBOM を利活用する組織は、以下の項目に関する運用方法を定めること。 ● SBOM の作成頻度 ● SBOM の深さ[13] ● 既知の未知[14] ● SBOM の共有 ● アクセス管理 ● 誤りの許容[15]

SBOM の利活用においては、コンポーネントに関する情報を収集し、一貫性のあるデータ構造を確立することが必要不可欠となる。そのため、データフィールドのカテゴリーでは、SBOM の対象となるコンポーネントを一意に特定するための情報を含めることが「最小要素」として位置づけられている。具体的なデータフィールドの定義は表 2-4 に示すとおりであり、SBOM の対象となるコンポーネントの名称やバージョン、その他の識別子に関する情報だけでなく、当該コンポーネントのサプライヤー及び SBOM 作成者の名称、コンポーネントの依存関係及びタイムスタンプに関する項目が含まれる。

[13] 図 2-9 に示すように、ソフトウェアのコンポーネントは階層構造となっていることが多い。SBOM の深さとは、この階層構造において、どの深さのコンポーネントまで SBOM に含めるか、ということを指す。

[14] 作成した SBOM において完全なコンポーネントの依存関係が未知である場合に、未知であるという事実を明示するということを意味する。例えば、依存関係の存在が不明であることの明示、部品を特定できていない範囲の明示等が挙げられる。

[15] NTIA は、「ソフトウェアサプライチェーンの管理方法は日々進化しているため、SBOM を導入・運用する初期フェーズにおいて完全性が欠如する可能性がある」としており、その上で、「偶発的な誤りに対しては明確に許容すべきである」としている。これにより、ツールの継続的な改善が促進されるとしている。

表 2-4 「最小要素」として SBOM に含めるべきデータフィールド

項目	説明
サプライヤー名（Supplier Name）	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（Component Name）	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。
コンポーネントのバージョン（Version of the Component）	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（Other Unique Identifiers）	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（Dependency Relationship）	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（Author of SBOM Data）	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（Timestamp）	SBOM データを作成した日付と時刻の情報。

2.4. SBOM フォーマットの例

SBOM の「最小要素」に規定されているとおり、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され、共有されることが求められる。この際、共通的なフォーマットを用いることで組織内の管理が効率化されるほか、組織を越えて 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。使用されうる SBOM フォーマットの例として、以下に示す 3 つのフォーマットが挙げられる。

(1) SPDX（Software Package Data Exchange）

(2) CycloneDX

(3) SWID タグ（Software Identification タグ）

SPDX の特徴として、スニペット、ファイル、パッケージ、コンテナ、OS ディストリビューション等の幅広いソフトウェア部品タイプをサポートしているほか、コンポーネントのライセンス情報を一意に特定するための識別子のリストが用意されていることが挙げられる。SPDX の項目のうち必要最低限の項目のみを含んだ SPDX-Lite という日本発のフォーマットも存在する。SPDX-Lite は、簡易的な SBOM 作成・管理を行う際に優れているほか、日本語で作成された仕様書等のドキュメントが豊富であることも特徴の一つであ

る。CycloneDX はセキュリティ管理を念頭に置いたフォーマットであり、対象となるソフトウェアの情報だけでなく、ソフトウェアに含まれる既知の脆弱性に関する情報やその脆弱性の悪用可能性に関する情報も記述することが可能である。最後に、SWID タグについて、ソフトウェアのライフサイクルに沿って SBOM を管理することができる特徴がある。

本節では、図 2-1 で示した簡易シナリオを再度考え、それぞれの SBOM フォーマットで A 社が作成する SBOM の例を示す。

(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 のフォーマットの構成、使用例・使用目的、特徴については付録の 7.3.3(1)を参照のこと。

前述した簡易シナリオにおいて、Tag-Value 形式の SPDX フォーマットを用いて A 社が SBOM を作成した場合、図 2-6 のような SBOM が作成される。ここで、色の関係は、表 2-1 で示した SBOM の概念的イメージと、SPDX フォーマットにおける項目との対応関係を示している。表 2-5 に示すとおり、SBOM の「最小要素」の各項目に対して、SPDX フォーマットの項目は対応可能である。

ID	サプライヤー名	コンポーネント名	コンポーネントのバージョン	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
1	Company A	Application	1.1	234	Primary	Company A	05-09-2022 13:00:00
2	Company B	Browser	2.1	334	Included in #1	Company B	04-18-2022 15:00:00
3	Mr. C	Compression Engine	3.1	434	Included in #2	Company A	05-09-2022 13:00:00
4	Community P	Protocol	2.2	534	Included in #1	Company A	05-09-2022 13:00:00

▼ SPDXフォーマット（tag-value形式）のSBOM

```
SPDXVersion: SPDX-2.2
DataLicense: CC0-1.0
DocumentNamespace: http://www.spdx.org/spdxdocs/8f141b09-1138-4fc5-aefb-fc10d9ac1eed
DocumentName: SBOM Example
SPDXID: SPDXRef-DOCUMENT
Creator: Organization: Company A
Created: 2022-05-09T13:00:00Z
Relationship: SPDXRef-DOCUMENT DESCRIBES SPDXRef-Application-v1.1

PackageName: Application
SPDXID: SPDXRef-Application-v1.1
PackageVersion: 1.1
PackageSupplier: Organization: Company A
PackageDownloadLocation: NOASSERTION
FilesAnalyzed: false
PackageChecksum: SHA1: 75068c26abbed3ad3980685bae21d7202d288317
PackageLicenseConcluded: NOASSERTION
PackageLicenseDeclared: NOASSERTION
PackageCopyrightText: NOASSERTION
ExternalRef: SECURITY cpe23Type cpe:2.3:a:company_a:application:1.1:*:*:*:*:*:*:*
Relationship: SPDXRef-Application-v1.1 CONTAINS SPDXRef-Browser-v2.1
Relationship: SPDXRef-Application-v1.1 CONTAINS SPDXRef-Protocol-v2.2

（以下省略）
```

図 2-6 簡易シナリオにおける SPDX フォーマット（Tag-Value 形式）の SBOM 例

表 2-5 SBOM「最小要素」に対応する SPDX の項目

SBOM「最小要素」のデータフィールド	対応する SPDX の項目
サプライヤー名（Supplier Name）	PackageSupplier
コンポーネント名（Component Name）	PackageName
コンポーネントのバージョン（Version of the Component）	PackageVersion
その他の一意の識別子（Other Unique Identifiers）	DocumentNamespace と SPDXID の組合せ、ExternalRef
依存関係（Dependency Relationship）	Relationship（DESCRIBES; CONTAINS による表現）
SBOM 作成者（Author of SBOM Data）	Creator

SBOM「最小要素」のデータフィールド	対応する SPDX の項目
タイムスタンプ（Timestamp）	Created

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。

SPDX の項目のうち、必要最低限の項目のみを含んだ SPDX-Lite という日本発のフォーマットも存在する。SPDX-Lite は、手作業によりライセンス情報を作成する組織において、SPDX 準拠のライセンス情報が膨大で運用が困難な場合に、必要な情報のみ授受する場合を想定して設計されている。OpenChain Japan Work Group (WG) のライセンス情報サブグループによって開発され、SPDX のサブセットとして、ISO/IEC 5962:2021 規格の一部にも含まれている。SPDX-Lite フォーマットにおける 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、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-Lite のフォーマットの構成、使用例・使用目的、特徴については付録の 7.3.3(2)を参照のこと。

前述した簡易シナリオにおいて、xls 形式の SPDX-Lite フォーマットを用いて A 社が SBOM を作成した場合、図 2-7 のような SBOM が作成される。xls 形式の SPDX-Lite フォーマットの場合、一つの xls ファイルに「Creation Information」と「Package Information」の 2 つのシートを含めることで、SBOM 情報を記載することができる。ここで、色の関係は、表 2-1 で示した SBOM の概念的イメージと、SPDX-Lite フォーマットにおける項目との対応関係を示している。表 2-6 に示すとおり、SBOM の「最小要素」の「依存関係（Dependency Relationship）」以外の項目に対して SPDX-Lite フォーマットの項目は対応可能である。

ID	サプライヤー名	コンポーネント名	コンポーネントのバージョン	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
1	Company A	Application	1.1	234	Primary	Company A	05-09-2022 13:00:00
2	Company B	Browser	2.1	334	Included in #1	Company B	04-18-2022 15:00:00
3	Mr. C	Compression Engine	3.1	434	Included in #2	Company A	05-09-2022 13:00:00
4	Community P	Protocol	2.2	534	Included in #1	Company A	05-09-2022 13:00:00

SPDX-Liteフォーマット（xls形式）のSBOM

Creation Information Sheet

SPDX Version	SPDX-2.2
Data License	CC0-1.0
SPDX Identifier	SPDXRef-DOCUMENT
Document Name	SBOM Example
SPDX Document Namespace	http://www.spdx.org/spdxdocs/8f141b09-1138-4fc5-aefb-fc10d9ac1eed
Creator	Company A
Created	05-09-2022 13:00:00

Package Information Sheet

PackageName	PackageSPDXIdentifier	PackageVersion	PackageFileName	PackageSupplier	PackageDownloadLocation	FilesAnalyzed	PackageHomePage	ConcludedLicense	DeclaredLicense	Comments on License	Copyright Text	PackageComment	External Reference field
Application	234	1.1	省略	Company A	省略								
Browser	334	2.1		Company B									
Compression Engine	434	3.1		Mr. C									
Protocol	534	2.2		Community P									

図 2-7 簡易シナリオにおける SPDX-Lite フォーマット（xls 形式）の SBOM 例

表 2-6 SBOM「最小要素」に対応する SPDX-Lite の項目

SBOM「最小要素」のデータフィールド	対応する SPDX-Lite の項目
サプライヤー名（Supplier Name）	PackageSupplier
コンポーネント名（Component Name）	PackageName
コンポーネントのバージョン（Version of the Component）	PackageVersion
その他の一意の識別子（Other Unique Identifiers）	SPDX Identifier と SPDX Document Namespace の組合せ、PackageSPDX Identifier
依存関係（Dependency Relationship）	-
SBOM 作成者（Author of SBOM Data）	Creator
タイムスタンプ（Timestamp）	Created

SPDX-Lite は、SPDX から必要最低限の項目のみを抽出したフォーマットであるため、運用性を重視した SBOM 管理が可能となる。SPDX は記述する必要がある項目が多く、自動処理による管理を目的としているが、SPDX-Lite は項目数が限られるため、手作業での管理も現実的に可能となる。ただし、SPDX-Lite には必要最低限の項目のみが含まれているため、例えば NTIA の「最小要素」で規定されている「依存関係」に関する項目等は表現できないことに注意が必要である。項目数が限定的であるため、サプライチェーン内で SBOM 共有を行う際に上流組織が求める要件に合致しない可能性もあり、SPDX-Lite の利用可否の判断においては、取引先への確認を行う等の対応が望まれる。さらに、SPDX-Lite フォーマットの SBOM を手作業で管理する場合、自動管理の場合と比較して管理工数が大きくなる場合があることにも注意が必要である。

(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 のフォーマットの構成、使用例・使用目的、特徴については付録の 7.3.3(3)を参照のこと。

前述した簡易シナリオにおいて、xml 形式の CycloneDX フォーマットを用いて A 社が SBOM を作成した場合、図 2-8 のような SBOM が作成される。ここで、色の関係は、表 2-1 で示した SBOM の概念的イメージと、CycloneDX フォーマットにおける項目との対応関係を示している。表 2-7 に示すとおり、SBOM の「最小要素」の各項目に対して CycloneDX フォーマットの項目は対応可能である。

ID	サプライヤー名	コンポーネント名	コンポーネントのバージョン	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
1	Company A	Application	1.1	234	Primary	Company A	05-09-2022 13:00:00
2	Company B	Browser	2.1	334	Included in #1	Company B	04-18-2022 15:00:00
3	Mr. C	Compression Engine	3.1	434	Included in #2	Company A	05-09-2022 13:00:00
4	Community P	Protocol	2.2	534	Included in #1	Company A	05-09-2022 13:00:00

▼ CycloneDXフォーマット（XML形式）のSBOM

```
<?xml version="1.0" encoding="utf-8"?>
<bom xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 serialNumber="urn:uuid:3e671687-395b-41f5-a30f-a58921a69b71" version="1"
 xmlns="http://cyclonedx.org/schema/bom/1.3">
 <metadata>
  <timestamp>2022-05-09T13:00:00Z</timestamp>
  <authors>
   <author>
    <name>Company A</name>
   </author>
  </authors>
  <component type="application">
   <name>Application</name>
   <version>1.1</version>
   <hashes>
    <hash alg="SHA-1">75068c26abbed3ad3980685bae21d7202d288317</hash>
   </hashes>
   <cpe>cpe:2.3:a:company_a:application:1.1:*:*:*:*:*:*:*</cpe>
   <externalReferences />
   <components />
  </component>
  <manufacture>
   <name>Company A</name>
  </manufacture>
  <supplier>
   <name>Company A</name>
  </supplier>
 </metadata>

 (中略)

 <dependencies>
  <dependency ref="pkg:maven/org.company_b/browser@2.1">
   <dependency ref="pkg:maven/org.c/CompressionEng@3.1" />
  </dependency>
  <dependency ref="pkg:maven/org.community_p/protocol@2.2" />
 </dependencies>

 (以下省略)
```

図 2-8 簡易シナリオにおける CycloneDX フォーマット（xml 形式）の SBOM 例

表 2-7 SBOM「最小要素」に対応する CycloneDX の項目

SBOM「最小要素」のデータフィールド	対応する CycloneDX の項目
サプライヤー名（Supplier Name）	component/supplier/name
コンポーネント名（Component Name）	component/name
コンポーネントのバージョン（Version of the Component）	component/version
その他の一意の識別子（Other Unique Identifiers）	serialNumber、component/cpe
依存関係（Dependency Relationship）	dependencies/dependency ref
SBOM 作成者（Author of SBOM Data）	metadata/authors/author/name
タイムスタンプ（Timestamp）	metadata/timestamp

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」が追加され、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 タグでは、ソフトウェアライフサイクルに沿ったソフトウェアのインストールプロセスの一貫として、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。SWID タグフォーマットによる SBOM では、SWID タグにしたがって作成された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パッチ等の情報が記載され、xml 形式がサポートされている。SWID タグは、対象とするデバイスのライフサイクルを把握するために、デバイスにインストールされたソフトウェアの情報を示すタグが規定されている。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。SWID タグのフォーマットの構成、使用例・使用目的、特徴については付録の 7.3.3(4)を参照のこと。

前述した簡易シナリオにおいて xml 形式の SWID タグを用いて A 社が SBOM を作成した場合、図 2-9 のような SBOM が作成される。ここで、色の関係は、表 2-1 で示した SBOM の概念的イメージと、SWID タグフォーマットにおける項目との対応関係を示している。表 2-8 に示すとおり、SBOM の「最小要素」の各項目に対して SWID タグフォーマットの項目は対応可能である。

ID	サプライヤー名	コンポーネント名	コンポーネントのバージョン	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
1	Company A	Application	1.1	234	Primary	Company A	05-09-2022 13:00:00
2	Company B	Browser	2.1	334	Included in #1	Company B	04-18-2022 15:00:00
3	Mr. C	Compression Engine	3.1	434	Included in #2	Company A	05-09-2022 13:00:00
4	Community P	Protocol	2.2	534	Included in #1	Company A	05-09-2022 13:00:00

SWIDタグフォーマット（XML形式）のSBOM

```
<SoftwareIdentity
 xmlns="http://standards.iso.org/iso/19770/-2/2015/schema.xsd"
 xmlns:sha512="http://www.w3.org/2001/04/xmlenc#sha512"
 name="application"
 tagId="Company A/application@1.1"
 version="1.1">
 <Entity name="Company A" role="tagCreatorsoftwareCreator" />
 <Meta title="Company A Application v1.1" timestamp="2022-05-09T13:00:00Z" />
 <Link href="swid:Company B/browser@2.1" rel="component" />
 <Link href="swid:Community P/ptotocol@2.2" rel="component" />
 <Payload >
  <File name="Company-A-application-1.1.exe"
sha512:hash="BC55DEF84538898754536AE47CC907387B8F61D9ACD7D3FB8B8A624199682C8FBE6D1631088AE6A322CDDC4252D3564655CB234D3818962B0B75C35504D55689"/>
 </Payload>
</SoftwareIdentity>

  （以下省略）
```

図 2-9 簡易シナリオにおける SWID タグフォーマット（xml 形式）の SBOM 例

表 2-8 SBOM「最小要素」に対応する SWID タグの項目

SBOM「最小要素」のデータフィールド	対応する SWID タグの項目
サプライヤー名（Supplier Name）	<Entity> @role(tagCreator) @name
コンポーネント名（Component Name）	<SoftwareIdentity> @name
コンポーネントのバージョン（Version of the Component）	<SoftwareIdentity> @version
その他の一意の識別子（Other Unique Identifiers）	<SoftwareIdentity>@tagId
依存関係（Dependency Relationship）	<Link> @rel @href
SBOM 作成者（Author of SBOM Data）	<Entity> @role(softwareCreator) @name
タイムスタンプ（Timestamp）	<Meta> @timestamp

SWID タグはソフトウェア識別に関するフォーマットであるが、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も含めることができるフォーマットである。

ここまで、SPDX、SPDX-Lite、CycloneDX、SWID タグにおける SBOM 例を示したが、多くのフォーマットは SBOM ツールを用いた自動処理・管理を目的としている。SBOM ツールを用いてソフトウェアのソースコードやバイナリファイルをスキャンし、ソフトウェアに含まれるコンポーネントを自動検出することで、自動で SBOM を作成することができる。加えて、SBOM ツールによっては、脆弱性情報やライセンス情報を継続的に把握することができるため、管理業務を効率化することができる。そのため、SBOMを導入する組織は、SBOMツールを用いたSBOMの作成・管理を行うことが現実的である。代表的なSBOMツールは付録の 7.3.2 に示すとおりであり、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いる。

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 SBOM ツールの選定の観点を整理した後、当該観点に基づき、複数の SBOM ツールを評価し、選定することが望まれる。SBOM ツールの選定に当たって実施すべき事項や認識しておくべきポイントは 4.2 を参照いただきたい。

SBOM ツールを用いた SBOM 管理を行う場合、図 2-6～図 2-9 に示したような Tag-Value 形式や xml 形式の SBOM ドキュメントはあまり意識せず、SBOM を作成・管理することができる。特に、多くの有償の SBOM ツールではダッシュボード機能が充実しているため、SBOM に含まれるコンポーネント一覧を簡単に表示できるほか、各コンポーネントの脆弱性やライセンスコンプライアンスに関する情報も一覧表示やグラフ表示することができる。

2.5. SBOM に関する誤解と事実

SBOM 導入のメリットがあるものの、国内における SBOM の普及率は高いとは言えない。この理由として、SBOM 導入にかかるコストの課題、技術的な課題、人材に関する課題等の様々な課題が考えられるが、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」[16]（SBOM の神話と事実）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。NTIA の文書で示された誤解と事実の概要は以下のとおりである。

誤解：SBOM は攻撃者を支援する

（事実）SBOM が攻撃に利用される可能性はあるものの、SBOM により透明性を確保することによる「攻撃者からの防御」におけるメリットの方が大きい。攻撃者にとって、SBOM やソフトウェアの

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

透明性に関する情報の効果は限定的であり、一般的に攻撃者は SBOM を必要としない。例えば、WannaCry によるランサムウェア攻撃は、攻撃のための前提条件として SBOM は必要ではない。

誤解：SBOM だけでは有用・実用的な情報を得ることができない

（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：SBOM は公開しなければならない

（事実）SBOM を公開する必要はなく、SBOM 作成者やサプライヤーの判断で SBOM の共有方法を判断することができる。米国大統領令においても、SBOM の公開は SBOM 作成者の判断であり、必須ではないことが明確に記載されている。

誤解：SBOM は知的財産や企業秘密を露呈する

（事実）SBOM はソフトウェアに含まれているコンポーネントの一覧リストであり、特許やアルゴリズムは含まれておらず、知的財産を公開するものではない。SBOM は単なる「材料の一覧」であり、特許やアルゴリズムのような「レシピ」とは異なる。また、SBOM には、ソフトウェアのソースコード自体は含まれない。第三者が開発したコンポーネントの特許やアルゴリズム等の知的財産は、コンポーネントの開発者又は著作権所有者に帰属することに留意する必要がある。

誤解：SBOM の導入を支援するプロセスは存在しない

（事実）ソフトウェア構成分析ツールは、一部の分野では、10 年以上にわたって企業内で使用されてきた実績がある。ソフトウェアの透明性に関しては、NTIA の活動、大統領令、SBOM フォーマットの標準化等の活動が進んでいるほか、一部の分野では、ソフトウェアの透明性について 5 年以上にわたって議論や実証の取組が進められており、他分野での導入をサポートしている。

また、国内で 2022 年度に実施した実証等を通じ、以下に示すさらに具体的な誤解と事実が明らかとなった。

誤解：対象ソフトウェアが直接利用しているコンポーネントのみ SBOM の管理対象とすればよい

（事実）対象ソフトウェアが直接利用しているコンポーネントだけでなく、そのコンポーネントが再帰的に利用するコンポーネントについても把握しないと、脆弱性対応が不十分となる可能性がある。どの階層のコンポーネントまで SBOM を作成するかという「SBOM の深さ」の観点に関しては、有識者による議論が進行中である。

誤解：SBOM 作成に用いる SBOM ツールの選定において、特に留意すべき点はない

（事実）SBOM 作成を支援するツールについて、有償のツール及び OSS として提供される無償のツールが既に複数公開されている。無償のツールを活用することで、ツール自体はコストをかけずに入手できるものの、有償ツールと比較して、導入・活用に関するマニュアルやサポートが限定的で

あることが多く、ツールの習得に多大なコストがかかる可能性がある。また、有償ツールと比較してサポート範囲や性能が限定的であることが多く、SBOM 導入の目的を達成できない可能性もある。SBOM の作成に当たっては、SBOM ツールを活用することで効率的に SBOM を作成することができるが、自社の SBOM 導入の目的を踏まえて使用するツールを選定する必要がある。

誤解：SBOM ツールを活用することで、対象ソフトウェアに含まれるコンポーネントを完全に特定することができる

（事実）SBOM ツールを用いることで効率的に SBOM を作成することができるが、SBOM 作成に当たってのコンポーネントの誤検出や検出漏れが発生し、正確な SBOM を作成することができない場合もある。そのため、例えば、SBOM ツールにより出力された SBOM をレビューする等の取組も検討することが大切である。また、ランタイムライブラリのような実行時に動的に追加されるライブラリは、SBOM ツールがライブラリの実体を解析しないため、特定することができない。そのような場合は、パッケージマネージャー等を用いてライブラリに対する構成情報と実行環境を個別に用意し、SBOM ツールにそれを認識させることで再帰的なコンポーネントを特定できるようにする必要がある。

誤解：SBOM ツールが出力したすべての脆弱性に対応する必要がある

（事実）SBOM ツールが出力した脆弱性に関する結果を踏まえて脆弱性へのリスク対応を行う際、必ずしもすべての脆弱性が悪用可能ではなく、影響を受けない脆弱性も存在することに留意する必要がある[17]。そのため、脆弱性の影響範囲、リスクの評価結果、対応に要するコスト等を踏まえ、優先度を踏まえた脆弱性対応が必要となる。なお、手動での SBOM 管理の場合、脆弱性データベースを活用して脆弱性の有無を手作業で特定する必要があるほか、個別に脆弱性を評価し、さらに対応方針を個々に検討する必要があるため、膨大な管理コストを要する可能性がある。

誤解：作成する SBOM のコンポーネントの粒度はサプライチェーン全体で共通化し、必要なコンポーネント情報だけを保持するべきである

（事実）現状では、JVN や米国 NVD のような脆弱性情報データベースにおける「影響を受けるソフトウェア」の粒度が体系化されていないため、コンポーネントの粒度を限定すると脆弱性の特定で漏れが生じる可能性がある。そのため、OSS のみならず、自社製品等も含めてコンポーネント情報を保持することが有効である。

誤解：SBOM の対象はパッケージソフトウェアや組込みソフトウェアのみである

（事実）ソフトウェアに限らず、IT システムも SBOM の対象となりうる。なお、コンテナイメージに対する SBOM、SaaS ソフトウェアに対する SBOM、クラウドサービスに対する SBOM 等のオンラインアプリケーションに対する SBOM の議論も米国を中心に行われている。

誤解：SBOM のフォーマットとして、SPDX、CycloneDX、SWID タグの 3 つのフォーマットのみが認められており、独自フォーマットに基づく SBOM は認められない

---

[17] ある製品が既知の脆弱性の影響を受けるかどうかを示す機械判読可能なセキュリティ勧告の一つとして、VEX（Vulnerability Exploitability Exchange）が米国を中心に開発されている。

（事実）米国 NTIA の定義に拠れば、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一覧リスト」のことであり、独自フォーマットであってもこの定義に合致する場合はSBOMとみなすことができる。ただし、2.2に記載のとおり、SBOMの「最小要素」として「自動化サポート」が位置づけられており、また、自動処理により効率化が図られることから、可能な限り、自動処理可能なフォーマットの採用を検討することが望ましい。

3. SBOM 導入に関する基本指針・全体像

3.1. SBOM 導入における基本指針

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するとともに、SBOM を導入することで解決したい自組織の課題と、それを踏まえた SBOM 導入の目的を明確化することが必要である。例えば、膨大な数のコンポーネントが存在する大規模製品について、コンポーネントの依存関係も含めた SBOM を作成し、共有するという目的があった場合、有償の SBOM ツールを用いて SBOM を作成・管理することが想定される。また、コンポーネント数が膨大ではない小規模な製品について、最低限の項目のみ手作業でコンポーネントのバージョンを管理したいという目的であれば、SPDX-Lite フォーマットを用いた SBOM を作成することが想定される。SBOM 導入の目的に応じて、作成すべき SBOM の項目、フォーマット、作成範囲、共有範囲等、SBOM の適用範囲が大きく異なるため、SBOM を導入する組織は、まず、SBOM 導入により解決したいソフトウェア管理に関する自社の課題を整理するとともに、SBOM 導入の目的を明確化したうえで、SBOM を作成・運用・管理することが求められる。

3.2. SBOM 導入プロセス

SBOM 導入に関するプロセスは主に 3 つのフェーズに分けることができる。具体的には、SBOM 導入に関する環境構築・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 つである。それぞれのフェーズにおける主な実施項目・実施概要を図 3-1 に示す。

環境構築・体制整備フェーズでは、SBOM の導入範囲を明確化するとともに、SBOM の作成・共有に向け環境や体制を構築する。SBOM 作成・共有フェーズでは、実際に SBOM を作成するとともに、必要に応じて作成した SBOM を外部に共有する。SBOM はソフトウェア管理の一手法であるため、作成することが目的ではなく、SBOM を用いた管理が重要となる。よって、SBOM 運用・管理フェーズとして、SBOM の情報に基づき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必要がある。

以降の章では、各フェーズにおける主な実施事項や SBOM 導入に当たって認識しておくべきポイントを示す。

図 3-1 SBOM 導入プロセス

4. 環境構築・体制整備フェーズにおける実施事項・認識しておくべきポイント

SBOM 導入に向け、まず SBOM に関する環境を整備するとともに、SBOM に関する体制を整備することが必要となる。本章では、環境構築・体制整備フェーズにおいて SBOM 導入組織が実施すべき事項や、SBOM 導入組織が認識しておくべきポイントを示す。

4.1. SBOM 適用範囲の明確化

【SBOM 導入に向けた実施事項】

- □ 対象ソフトウェアの開発言語、コンポーネント形態、開発ツール等、対象ソフトウェアに関する情報を明確化する。
- □ 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利用者及びサプライヤーとの契約形態・取引慣行を明確化する。
- □ 対象ソフトウェアの SBOM に関する規制・要求事項を確認する。
- □ SBOM 導入に関する組織内の制約（体制の制約、コストの制約等）を明確化する。
- □ 整理した情報に基づき、SBOM 適用範囲（5W1H）を明確化する。

【SBOM 導入に向け認識しておくべきポイント】

- 組織内外の開発者の知見を活用することで、対象ソフトウェアに関する効率的な情報収集を行うことができる。
- 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ことで、リスク管理の範囲を明確化することができる。

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 の適用範囲を明確化する必要がある。SBOM 適用範囲は表 4-1 に示す 5W1H の観点に分類することができ、各観点において複数の適用項目（選択肢）が存在する。

表 4-1 SBOM 適用範囲（5W1H）

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
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の共有
SBOM のフォーマット・項目（What）	・ 標準フォーマット（SPDX、SPDX-Lite、CycloneDX、SWID タグ、SPDX-Lite） ・ 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、これらの適用項目の組合せによって定まる。どの適用項目を選択するかにより、SBOM 導入に要するコストが異なることに留意が必要である。また、一つの観点に対し、複数の適用項

目を選択する可能性もある。適用項目の決定のために、SBOM の対象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ことが求められる。

対象ソフトウェアに関して、以下に関する情報をまず整理することが望まれる[18]。

- **開発言語**
  （例） Python、Java、Go、JavaScript、Rust、Swift、Objective-C、C、C++、VisualBasic 等
- **コンポーネントの形態**
  （例）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**
  （例）Visual Studio、Eclipse、Android Studio、Xcode 等
- **ビルドツール**
  （例）Jenkins、Circle CI、Github Actions、Gradle、Maven 等
- **構成管理ツール**
  （例）Github、Gitlab、Team Foundation Server、Ansible 等
- **自組織で取扱うデータ形式**
  （例）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**
  （例）OS、CPU アーキテクチャ 等

このような情報の整理においては、組織内外の開発者の知見を活用することが効果的である。特に、SBOM ツールを用いて SBOM を作成する場合、ツールによって対応している言語やコンポーネント形態が異なるため、開発言語とコンポーネントの形態については最低限把握することが必要である。そしてSBOM の対象とするコンポーネントの範囲を明確化するために、対象ソフトウェアの構成を可視化することが望まれる。具体的には、対象ソフトウェアにおいて自組織で開発した範囲、取引契約のあるサプライヤーが開発した範囲、取引契約のないサプライヤーが開発した範囲（OSS 等）を可視化した図を作成することが望まれる。一例として、2022 年度の実証で対象とした歯科用 CT では以下のような構成図を作成し、この構成図をベースに、リスク管理の範囲を明確化した。

[18] 各項目に対する例示は網羅的ではなく、例示の内容に限定されないことに留意。

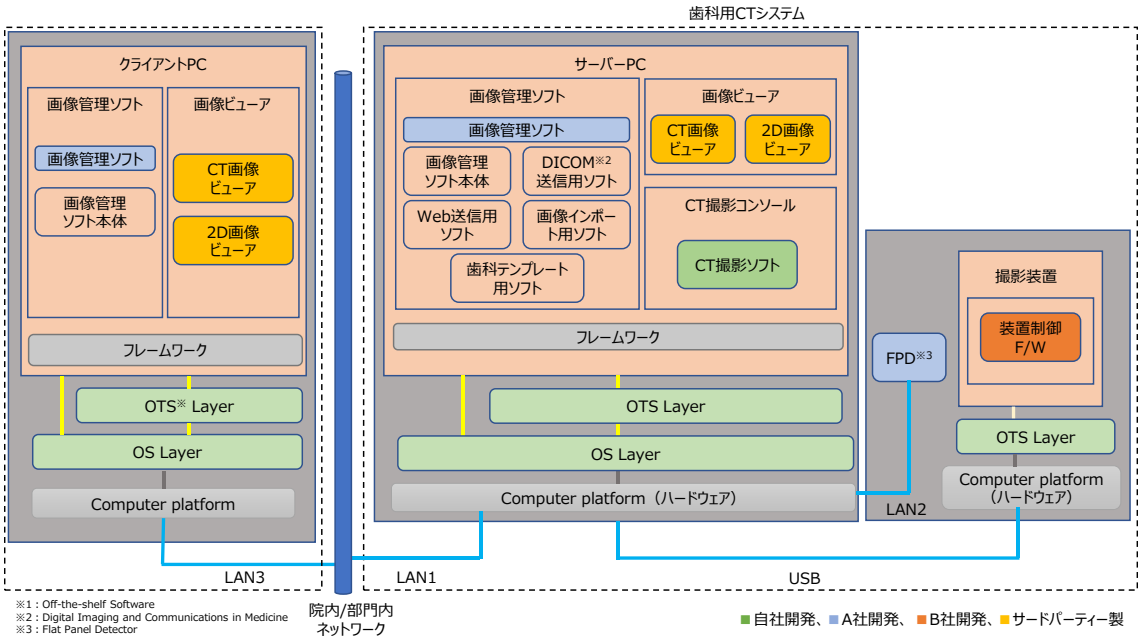

図 4-1 システム構成図の例（歯科用 CT の例）

あわせて、SBOM の対象とするコンポーネントの範囲対象を明確化するために、対象ソフトウェアの利用者及びサプライヤーとの契約形態・取引慣行を整理することが望まれる。具体的には、利用者・サプライヤーそれぞれについて、以下の各項目に関する対象ソフトウェアの情報を整理することが望まれる。

- **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**：提供なし、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 について申告 等
- **脆弱性の通知**：修正すべきと判断された脆弱性に関してのみ通知 等
- **脆弱性の修正**：修正すべきと判断された脆弱性に関してのみ修正 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、納入元に帰属 等
- **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 適用項目のうち、SBOM のフォーマット・項目や SBOM の活用範囲を決定するために、対象ソフトウェアの SBOM に関する規制・要求事項を確認し、整理することが望まれる。現状のところ、

SBOM の提供が義務付けられているソフトウェアは限定的であるが、例えば米国では、政府調達対象となるソフトウェアベンダーに対して SBOM の提供を推奨している[19]ほか、EU では、2022 年 9 月に草案が発表されたサイバーレジリエンス法において、EU 市場に上市するデジタル製品に対する SBOM に関する要求事項が含まれている[20]。医療機器分野では、IMDRF（International Medical Device Regulators Forum：国際医療機器規制当局フォーラム）から発行された「医療機器サイバーセキュリティガイダンス（IMDRF ガイダンス）」を薬機法による医療機器の規制に取り入れ 2023 年を目途に本格運用するとの方針が示されているところ、今後、規制の中で SBOM が要求される可能性もある。規制・要求事項において、SBOM のフォーマット・項目や SBOM の活用範囲が規定される可能性もあるため、対象ソフトウェアに関する規制・要求事項について随時情報収集し、求められる場合には具体的な要求事項を整理することが望まれる。

SBOM 適用項目の検討に当たっては、当然ながら、SBOM 導入に向けた組織内の制約も加味する必要がある。最も想定される制約としては、組織内の体制に関する制約やコストに関する制約が挙げられる。これらの制約が厳しい場合、限定的な SBOM 適用項目しか選択できない可能性もあるところ、SBOM 適用範囲の整理のために、あらかじめ組織内の制約を確認・整理することが望まれる。

整理したこれらの情報を踏まえ、上述した SBOM 適用範囲の 5W1H の各観点について、適用項目を検討・明確化することが望まれる。SBOM 適用範囲は対応したいリスクの範囲やレベルによって変わることに注意が必要である。例えば、将来的に規制・要求事項として求められる医療機器において SBOM を作成するとした場合に、自組織に限らず取引契約のあるサプライヤーにてソフトウェアビルド時に SBOM を作成し、その SBOM を利用者である医療機関が活用することが想定される。SBOM の対象とするコンポーネントの範囲としては、直接利用するコンポーネントに限らず、再帰的に利用されるコンポーネントも含めて対象とし、SBOM ツールを用いて自動で SBOM を作成することで、脆弱性管理及びライセンス管理に活用することが想定される。SBOM のフォーマット・項目としては、SPDX 等の自動処理可能なフォーマットに基づき、規制で求められる要求項目を含んだ SBOM を作成することが望まれる。

4.2. SBOM ツールの選定

【SBOM 導入に向けた実施事項】

□ 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 ツールの選定の観点を整理する。
（選定の観点の例：機能、性能、解析可能な情報、解析可能なデータ形式、コスト、対応フォ

[19] Office of Management and Budget, Enhancing the Security of the Software Supply Chain through Secure Software Development Practices https://www.whitehouse.gov/wp-content/uploads/2022/09/M-22-18.pdf

[20] European Commission, Cyber Resilience Act https://digital-strategy.ec.europa.eu/en/library/cyber-resilience-act

ーマット、コンポーネント解析方法、サポート体制、他ツールとの連携、提供形態、ユーザーインターフェース、運用方法、対応するソフトウェア開発言語、日本語対応等）

□ 整理した観点に基づき、複数の SBOM ツールを評価し、選定する。

【SBOM 導入に向け認識しておくべきポイント】

- 複数の SBOM ツールの使い分けは非効率となる場合があるため、目的に対して最小限の SBOM ツールを用いた運用となるかどうか等も考慮することが望ましい。
- 有償の SBOM ツールは一般に高価である。一方で、無償の SBOM ツールは、ツール自体のコストは無料であるものの、環境整備や学習に当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可能性がある。
- 有償の SBOM ツールと比較して、無償の SBOM ツールの機能・性能は限定的である場合が多く、例えば、再帰的な利用部品が検出できない、読み込み可能な SBOM フォーマットに制限がある、ライセンスの検知漏れが発生する、導入環境が限定される等の課題がある。
- オンプレミス型の SBOM ツールでは、導入環境が制約される場合がある。また、SaaS 型の SBOM ツールでは、機密性の高いソースコードの情報が外部に送信される構造になっていないかを確認する必要がある。
- SBOM 導入が原因で開発効率が著しく下がることのないよう、既存の開発プロセスへの組込が容易な SBOM ツールを選定し、開発者に負担をかけない運用を心がけることが必要である。
- SBOM ツールの選定にあたり、無償トライアル等を利用して実際の使用感を体験することが効果的である。観点の設定や選定に難しさを感じる場合には、複数の SBOM ツールを扱う販売代理店に相談し、各ツールの特徴や長所・短所を比較評価しながら選定することも一案である。

SBOM 導入組織は、明確化した SBOM 適用範囲に対応する SBOM を作成するための環境構築・体制整備を行う必要がある。SBOM 作成・管理の環境として最も重要な設備は SBOM ツールである。SBOM を作成・管理するとした場合に、SBOM ツールは必ずしも必須ではなく、例えば SPDX-Lite のように手動で SBOM を作成・管理できるフォーマットも用意されている。しかしながら、実証を通じて、SBOM ツールを活用することでコンポーネント管理にかかる工数は小さくなるほか、SBOM ツールを活用することで、OSS 間の依存関係や OSS の再利用も効率的に検出・管理でき、脆弱性発表から特定までのリードタイムを短縮可能であることが明らかとなった。そのため、SBOM ツールを用いた SBOM の作成・管理を行うことが現実的であり、本手引でも SBOM ツールを前提とした記載としている。

代表的な SBOM ツールは付録の 7.3.2 に示すとおりであり、SBOM ツールは大きく有償のツールと無償のツールに分かれる。有償の SBOM ツールは一般に高価であるが、ユーザーインターフェースが充実しており、直感的な SBOM 作成・管理が可能であるほか、サポート体制が整備されているため、ツール

の導入や運用に悩んだ際にベンダーや販売代理店に相談できるというメリットがある。さらには、各種開発ツールやコミュニケーションツールと連携可能な SBOM ツールも存在する。無償の SBOM ツールは、ツール自体のコストは無料であるものの、環境整備や学習に当たっての情報が不足していることが多い。そのため、ツールの導入・運用やエラー発生時の原因究明・対応に大きな工数を要する可能性がある。また、有償の SBOM ツールと比較して、無償の SBOM ツールの機能・性能は限定的であることが多く、例えば、再利用部品が検出できない、読み込み可能な SBOM フォーマットに制限がある、ライセンスの検知漏れが発生する、導入環境が限定される等の課題があるものの、無償の SBOM ツールは OSS コミュニティを中心に活発に開発されているため、機能・性能が向上していく可能性があることに留意する必要がある。なお、無償の SBOM ツールに関するサポートサービスを提供している企業もあり、無償の SBOM ツールを活用する場合に必要に応じて支援を受けることも想定される。

有償・無償の様々な SBOM ツールが用意されているところ、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選定の観点を整理し、この観点に基づいて SBOM ツールを評価・選定することが望まれる。想定される選定の観点の例として、表 4-2 に示す観点が挙げられる。

表 4-2 SBOM ツールの選定の観点

観点	説明
機能	SBOM ツールが有する機能として、コンポーネントの解析機能、脆弱性情報・ライセンス情報の自動マッチング機能、リスクの定量化機能、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、SBOM データのインポート機能等が挙げられる。SBOM ツールによって対応している機能が異なるため、SBOM 導入の目的や SBOM 適用範囲を踏まえ、どの機能が必要であるかを整理することが望まれる。
性能	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。SBOM 導入の目的や SBOM 適用範囲を踏まえ、どの程度の性能[21]を求めるかを整理することが望まれる。
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 導入の目的や SBOM 適用範囲を踏まえ、どの情報が必要であるかを整理することが望まれる。

[21] なお、ツールの性能を把握する方法として、例えば、無償トライアルを利用して実際に解析するソフトウェアや利用する SBOM 等をツールに読み込ませ、正確な情報をツールが出力可能か確認することやツールベンダーがデータベースに収録している OSS や脆弱性の数、脆弱性情報の情報元（JVN、NVD 等）、データベースの更新頻度等の仕様を開発元・代理店に確認すること等が想定される。

観点	説明
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。
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る。ツールによって料金体系は異なるが、多くのツールが年間のサブスクリプションモデルで提供している。オプションとして、複数の OSS 解析方法が利用できるツールがあるほか、問合せ対応に限らず OSS 管理に関する様々な相談が可能となるプランを提供しているツールも存在する。また、ライセンス費用の算出方法は、開発者数や組織の規模、解析コード量に応じた課金等、ツールによって様々であり、高額であっても会社全体で導入する場合にスケールメリットが出る場合もある。社内のコスト制約を踏まえ、SBOM ツールに対してどの程度のコストをかけることができるか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 SBOM フォーマット（SPDX、SPDX-Lite、CycloneDX、SWID タグ等）の SBOM のみインポート可能／作成可能な場合がある。SBOM の作成に関しては、大半の SBOM ツールが複数の SBOM フォーマットをサポートしているが、SBOM のインポートに関しては、複数の SBOM フォーマットに対応している製品は多くない。SBOM 適用範囲を踏まえ、どの SBOM フォーマットに対応する必要があるかを整理することが望まれる。
コンポーネント解析方法	ソフトウェアに含まれるコンポーネントの解析方法は、コードマッチング、依存関係検出、文字列検出の大きく 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。複数の解析方法を組合せて OSS 解析を行っている SBOM ツールもあれば、一部の解析方法のみに対応しているツールもあるため、SBOM 作成に当たって用意できるコード情報等を踏まえ、どの OSS 解析方法を採用すべきかを整理することが望まれる。

観点	説明
サポート体制	有償ツールに関しては、ツールの導入や運用についてベンダーに問合せが可能なSBOM ツールが存在するほか、オプションとして、ツールに関する問合せに限らずOSS 管理に関する様々な相談が可能となるプランを提供しているツールも存在する。また、無償ツールについても、サポートサービスを提供している企業もあり、必要に応じて支援を受けることも想定される。SBOM 適用範囲や組織内の SBOM 導入に関わる担当者の知識レベル等を踏まえ、どの程度のサポートが必要であるかを整理することが望まれる。
他ツールとの連携	開発環境、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可能な SBOM ツールが存在する。SBOM 作成の自動化等、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く、どのようなツールとの連携が必要であるかを整理することが望まれる。
提供形態	SBOM ツールの提供形態にはパッケージ版とクラウド版がある。パッケージ版のSBOM ツールを導入する場合、ツール料金のほかに、サーバーの維持管理費用が発生する可能性があるほか、導入できる環境が制約される可能性がある。クラウド版の場合、パッケージ版と比較して初期導入にかかるコストや SBOM 共有に関する工数を低減できる。ただし、機密性の高い自社のソースコード情報が社外に送信されるおそれがないかをあらかじめ確認する必要がある。組織内のシステム制約を踏まえ、どちらの提供形態がふさわしいかを整理することが望まれる。
ユーザーインターフェース	SBOM ツールによって、CLI（コマンドラインインターフェース）のみ提供している場合と、GUI（グラフィカルユーザーインターフェース）も提供している場合がある。GUI 対応のツールの場合、直感的な SBOM の作成や出力結果の可視化が可能となる。SBOM 導入に関わる担当者の知識レベル等を踏まえ、どのようなユーザーインターフェースを備えたツールが求められるかを整理することが望まれる。
運用方法	開発者が自身で SBOM ツールを実行する場合は、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 ツールを選定することで、開発者の負担を軽減することができる。一方、解析チームのような専門部隊が取り纏めてSBOM ツールを実行する場合は、ポリシー機能やライセンス等の補足情報が充実している SBOM ツールを選定することで、解析結果の精査がしやすくなる。
対応するソフトウェア開発言語	SBOM ツールによって対応しているソフトウェア開発言語が異なる。C、C++、Java、Python、Ruby、Swift、Go 等の代表的な言語であれば多くのツールが対応しているが、一部の言語については、対応しているツールが限定的な場合がある。対象ソフトウェアの情報整理結果を踏まえ、どの開発言語に対応したSBOM ツールを導入する必要があるかを整理することが望まれる。

観点	説明
日本語対応	現状のところ、海外で開発された SBOM ツールがほとんどである。そのため、取扱説明書や README ファイルが英語のみで提供されているケースがあるほか、ツール自体も日本語対応していない場合がある。英語のみで提供されているツールの運用が困難である場合には、自組織の SBOM 導入の目的やその他の観点を踏まえて検討したうえで[22]、日本語対応しているツールの優先度を高めることが考えられる。なお、有償ツールの販売代理店や無償ツールのサポートを提供している企業において、SBOM ツールに関するドキュメント類を日本語に翻訳して提供している場合もある。

SBOM 導入の目的や SBOM 適用範囲を踏まえ、各観点について、どの程度の内容を求めるか、あらかじめ整理した上で SBOM ツールを評価・選定することが望まれる。例えば、SBOM 導入に際して利用できる社内の予算が限定的である場合、無償の SBOM ツールから、自社の開発言語に対応し、所望のフォーマットで SBOM を出力できるツールを選定することが想定される。有償ツールを導入できる予算が充てられている場合、機能、性能、コスト等を総合的に勘案し、複数のツールを評価した上で、ツールを選定することが想定される。なお、事業部門や開発プロジェクトごとに求める最適なツールの観点が異なる場合等を除き、複数の SBOM ツールの使い分けが非効率となる場合があることに留意が必要である。

ツールの評価・選定に当たっては、SBOM ツールを扱う代理店に相談することも想定される。SBOM ツールの導入前に無償トライアル[23]等を利用して実際の使用感を体験し、操作学習の難易度及び必要期間を評価することで、自社における典型的な製品や適用を想定しているプロジェクトのソースコードを試験的に解析し、期待通りの結果が得られるかどうかを確認することができる。また、観点の設定や選定に難しさを感じる場合には、複数の SBOM ツールを扱う販売代理店に相談し、多くのツールに関する情報を把握しつつ、各ツールの特徴や長所・短所を比較評価しながら選定することも想定される。

4.3. SBOM ツールの導入・設定

【SBOM 導入に向けた実施事項】

□ SBOM ツールが導入可能な環境の要件を確認し、整備する。

[22] 例えば、自社海外拠点、海外提携先、外資サプライヤー等と SBOM ツールを共同運用する可能性がある場合、日本語対応の優先度より、ツールの機能、性能、運用方法等を踏まえたツール選定が望まれる等、導入目的や他の観点も踏まえた検討が必要である。

[23] なお、トライアルを実施する前に評価したい機能やユースケースを整理し、具体的なトライアル計画を策定することが効果的である。

□ ツールの取扱説明書や README ファイルを確認して、SBOM ツールの導入・設定を行う。

【SBOM 導入に向け認識しておくべきポイント】

- サポート体制が整備されている有償の SBOM ツールにおいては、販売代理店やツールベンダーに対して問合せを行い、支援を受けることで、効率的にツールの導入・設定を行うことができる。
- 無償の SBOM ツールでは、ツールの構築や設定に関する情報が不足している場合があるため、試行錯誤的に設定を行うための負担を強いられる可能性がある。必要に応じて、無償ツールに関するサポートサービスを提供している企業の支援を受けることで、効果的な無償 SBOM ツールの導入・設定が可能となる。
- SBOM ツールを脆弱性管理に活用する場合、障害等の影響で SBOM ツールが停止し、脆弱性の検知が滞ることのないよう、稼働監視やデータの定期的なバックアップを実施する必要がある。

SBOM ツールによって導入可能な環境が異なり、例えば、ツールが動作する PC について、インターネット接続があること、一定以上のマシンスペックを有すること、特定の OS であること、特定のブラウザをインストールしていること、Java や Python の実行環境が整っていること等が求められる場合がある。また、一部の SBOM ツールでは導入可能な OS が Linux に限定されており、Windows 端末にインストールする場合、別途仮想マシン環境が必要となる場合がある。そのため、SBOM ツールの導入・設定に当たっては、まず当該ツールの導入要件を確認し、導入できる環境を整備することが必要である。

導入できる環境が整備できた後、実際に SBOM ツールを導入し、SBOM 作成に向けた初期設定を行う。基本的には、取扱説明書や README ファイルを確認して導入・設定を行うこととなるが、サポート体制が整備されている有償の SBOM ツールにおいては、販売代理店やツールベンダーの支援を受けることで、効率的にツールの導入・設定を行うことができる。環境構築や初期設定の代行サービスを提供している販売代理店もあるため、必要に応じてそれらのサービスの活用を検討するのも一案である。特定の SBOM ツールでは、ツールの構築や設定に関する情報が不足している場合がある。また、多くの無償 SBOM ツールは海外で開発されているため、参考となるドキュメントが英語のみであることが多い。そのため、サンプルコードを入力して所望の SBOM が出力されるかを確認する等、試行錯誤的に設定を行うことも想定される。必要に応じて、無償ツールに関するサポートサービスを提供している企業の支援を受けることで、効果的な無償 SBOM ツールの導入・設定が可能となる。

なお、SBOM ツールを脆弱性管理に活用する場合、障害等の影響で SBOM ツールが停止し、脆弱性の検知が滞ることのないよう、稼働監視やデータの定期的なバックアップを実施する必要がある。

4.4. SBOM ツールに関する学習

【SBOM 導入に向けた実施事項】

□ ツールの取扱説明書や README ファイルを確認して、SBOM ツールの使い方を習得する。

□ ツールの使い方に関するノウハウや各機能の概要は記録し、組織内で共有する。

【SBOM 導入に向け認識しておくべきポイント】

- サポート体制が整備されている有償の SBOM ツールにおいては、販売代理店やツールベンダーに対して問合せを行うことで、効率的にツールの使い方を習得することができる。
- サンプル SBOM の作成等を通じて試行錯誤的にツールを使うことで、効率的にツールの使い方を習得できる。

SBOM ツールが導入・設定できた後、当該ツールの使い方を習得することが望まれる。上記のツールの導入・設定と同様に、基本的には、取扱説明書や README ファイルを確認してツールの使い方を習得することとなるが、サポート体制が整備されている有償の SBOM ツールにおいては、販売代理店やツールベンダーに対して問合せを行うことで、効率的にツールの使い方を習得することができる。無償ツールと比較して有償ツールは高機能であるため、すべての機能に関する習得に時間がかかる可能性がある。自組織が作成したい SBOM 作成に必要な機能を販売代理店やツールベンダーに確認し、当該機能に絞って使い方を習得することも想定される。また、サンプル SBOM の作成等を通じて、試行錯誤的にツールの使い方を習得することも効果的である。特に、使い方に関する情報が不足しているツールの場合に効果的である。なお、ツールの具体的な使い方は組織によって異なるところ、学習プロセスを通じて明らかとなったツールの使い方に関するノウハウや各機能概要を記録し、組織内で共有することが望まれる。

5. SBOM 作成・共有フェーズにおける実施事項・認識しておくべきポイント

構築した環境や体制に基づき実際に SBOM を作成し、必要に応じて SBOM を提供することが求められる。本章では、SBOM 作成・共有フェーズにおいて SBOM 導入組織が実施すべき事項や、SBOM 導入組織が認識しておくべきポイントを示す。

5.1. コンポーネントの解析

【SBOM 導入に向けた実施事項】

□ SBOM ツールを用いて対象ソフトウェアのスキャンを行い、コンポーネントの情報を解析する。

□ SBOM ツールの解析ログ等を調査し、エラー発生や情報不足による解析の中断や省略がなく、解析が正しく実行されたかを確認する。

□ コンポーネントの解析結果について、コンポーネントの誤検出や検出漏れがないかを確認する。

【SBOM 導入に向け認識しておくべきポイント】

- SBOM ツールを用いることで、手動の場合と比較し、効率的にコンポーネントの解析及び SBOM の作成を行うことができる。SBOM ツールを用いることの効果はコンポーネント数が多いほど大きい。
- パッケージマネージャーの構成情報を活用することが効果的な場合がある。また、パッケージマネージャーを用いることで、SBOM ツールでは特定できない粒度の細かいコンポーネントを特定できる場合がある。
- コンポーネントの誤検出や検出漏れが生じる場合がある。例えば、シンボリックリンクやランタイムライブラリ等のコンポーネント、深い階層のコンポーネント、特定分野でのみ利用されているコンポーネント等を検出できない場合があるほか、コンポーネントを特定できてもバージョン情報が誤っている場合がある。
- SBOM ツールにおけるコンポーネント解析方法によって、出力結果が異なる。依存関係に基づく解析の場合、誤検出の発生可能性は極めて低いが、その他の解析方法の場合、誤検出・検出漏れが発生する可能性がある。また、バイナリファイルに基づく解析の場合、ソースコードを利用できない場合でもバイナリファイルのみで解析可能というメリットがある一方で、バイナリファイルのみを用いた場合、解析精度が下がる可能性がある。
- コンポーネントを解析する環境（実行環境、開発環境等）によって、解析結果が異なる場合がある。

- SBOM ツールのデータベースに存在しない OSS は検出できないため、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、追加対応が必要となる場合がある。
- 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場合があるため、適切な設定を施した上で解析する必要がある。
- サブ階層のコンポーネントやサードパーティコンポーネントに関する誤検出や検出漏れの確認には特に大きな工数を要する。検出漏れの保証は困難であるため、正確性の度合いと対応工数とのトレードオフを踏まえた確認が必要となる。
- SBOM ツールの解析方式を考慮することで、効率的に誤検出や検出漏れを確認することができる。

実証を通じて、SBOM ツールを用いることで、手動の場合と比較し、効率的にコンポーネントの解析及び SBOM の作成を行うことができることを確認した。例えば、医療機器分野の歯科用 CT を対象とした実証では、手動での SBOM 作成には 30 人時間以上の工数が必要になった一方で、SBOM ツールを用いた場合、0.15 人時間で SBOM 作成できることを確認でき、ツールを用いることで 99%以上の工数削減につながることを確認した。そのため、SBOM ツールを用いたコンポーネントの解析及び SBOM の作成・管理を行うことが現実的であり、本節でも SBOM ツールを前提とした記載を行う。なお、パッケージマネージャーを用いることで、SBOM ツールでは特定できない粒度の細かいコンポーネントを特定できる場合があり、パッケージマネージャーの構成情報を活用することで、効果的に SBOM を作成できる場合がある。また、可能な場合に、ソフトウェアサプライヤーから SBOM を受領することで、効率的に SBOM を作成できる。

SBOM 作成に向け、まず、SBOM ツールに基づいて対象ソフトウェアをスキャンし、コンポーネントの情報を解析する。SBOM ツールによってスキャン方法が異なり、GUI 上から対象ソフトウェアを指定して解析する場合もあれば、CLI にて解析を行う場合もあるため、解析の方法について、導入した SBOM ツールの取扱説明書や README ファイルを確認いただきたい。解析を行うことで、対象ソフトウェアに含まれるコンポーネントの名称、サプライヤー名、バージョン、コンポーネント間の依存関係等を洗い出すことができる。ただし、コンポーネントの誤検出や検出漏れが生じる場合があることに留意する必要がある。事実、実証において、以下の点が明らかとなった。

- シンボリックリンクやランタイムライブラリ等、実体が 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ことが分かった。
- 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用い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となった。
- 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。

よって、SBOM ツールの出力結果をそのまま用いるのではなく、誤検出や検出漏れに関して出力結果を確認することが重要である。結果に対する誤検出や検出漏れの確認の観点及び確認方法は図 5-1 に示すとおりである。なお、誤検出や検出漏れの確認は基本的に人手で実施する必要があるため、網羅的に確認することが現実的に難しい場合がある。実証では、誤検出や検出漏れの確認に 0.50 人時間を要したコンポーネントも存在し、コンポーネントが膨大なソフトウェアの場合、誤検出や検出漏れの確認に関して多大な工数が必要となる。特に、サブ階層のコンポーネントやサードパーティコンポーネントに関する誤検出や検出漏れの確認には大きな工数を要する。検出漏れの保証は困難であるため、正確性の度合いと対応工数とのトレードオフを踏まえた確認が必要となる。

確認の観点		確認方法
ツールが正しく動作しているか		① ツールの実行ログにエラーが無いかを確認する。
コンポーネントの解析結果が正しいか	コンポーネントを誤検出していないか	② 出力されたコンポーネントが、対象ソフトウェアに含まれているかを確認する。マッチタイプが完全一致でない場合、コンポーネントが改変されている可能性が無いかを確認する。 ③ （②で判断がつかない場合）出力されたコンポーネント名を元に、GitHub等の外部ソースから入手したソースコードと対象ソフトウェアのソースコードとを比較する。
	使用しているコンポーネントが含まれているか（検出漏れが無いか）	④ 対象ソフトウェアに含まれるコンポーネントが出力結果に含まれているかを確認する。
	改変されたコンポーネントを検出できているか	⑤ スニペット解析の結果をもとに、対象ソフトウェアのソースコードと元のソースコードとを比較する。
	検出できていないコンポーネントが無いか	⑥ 対象ソフトウェアのCopyrightを抽出し、検出したコンポーネントが出力結果に含まれているかを確認する。

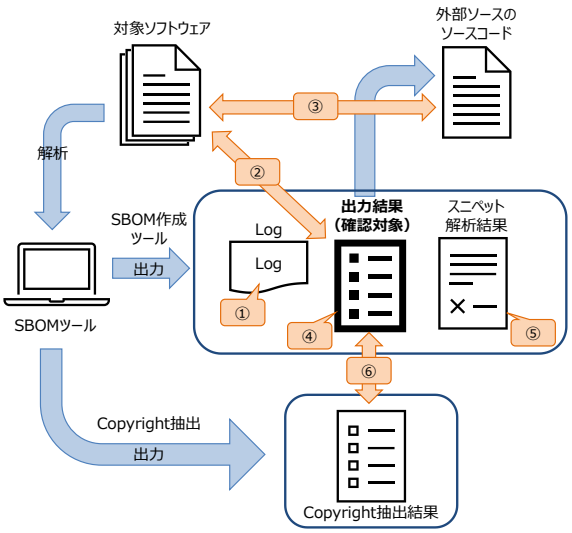

図 5-1 コンポーネント解析結果の確認の観点及び確認方法

誤検出や検出漏れの確認に当たっては、SBOM ツールの解析方式を考慮することが重要である。ツールにおけるコンポーネントの解析方法には、コードマッチング、依存関係検出、文字列検出の大きく 3 つの方法が存在する。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。一方で、コードマッチングや文字列検出による解析の場合、誤検出や検出漏れが発生する可能性がある。また、バイナリファイルに基づくスキャンの場合、多くの検出漏れが発生したことが実証で明らかとなった。コンポーネントの解析方式によって誤検出や検出漏れの発生度合いが異なるところ、使っている SBOM ツールの解析方式を踏まえた誤検出や検出漏れの確認が望まれる。例えば、バイナリファイルに基づく解析の場合、ソースコードを利用できない場合でもバイナリファイルのみで解析可能というメリットがある一方で、バイナリファイルのみを用いた場合、誤検出や検出漏れが多数発生する可能性があるということ等も踏まえながら、誤検出や検出漏れを確認することが想定される。また、SBOM ツールのパラメータ設定の不足や、パッケージマネージャーの実行失敗等の理由により解析が正常に実施できておらず、誤検知や検出漏れが発生している可能性もある。表面上は正常終了しているように見えても、エラーにより内部的な解析プロセスの一部をスキップして終了している場合もあるため、ツールの実行ログを確認し、そのようなエラーが発生していないかを確認する必要がある。

誤検出や検出漏れの確認の結果、未知の情報や不明の情報が含まれていることが分かった場合、その情報を「既知の未知」として把握することが望まれる。「既知の未知」とは、不明ではあるものの、不明である事実を既知としておくことであり、表 2-3 で記載のとおり、SBOM の「最小要素」においても言及されている。作成した SBOM を対象ソフトウェアの利用者や納入先に共有する場合、「既知の未知」な情報も合わせて共有することで、情報の透明性を高めることができる。

5.2. SBOM の作成

【SBOM 導入に向けた実施事項】

□ 作成する SBOM の項目、フォーマット、出力ファイル形式等の SBOM に関する要件を決定する。

□ SBOM ツールを用いて、当該要件を満足する SBOM を作成する。

【SBOM 導入に向け認識しておくべきポイント】

- SBOM 作成と共有の目的を鑑み、正確な情報を不足なく SBOM に記載することが望ましい。
- サードパーティや OSS コミュニティ等の第三者から提供されたコンポーネントを使用している場合は、当該コンポーネントの SBOM の提供を受けることができる場合もある。ただし、そのコンポーネントを自組織にて改変して使用している場合は、提供を受けた SBOM をそのまま利用できなくなるので注意が必要である。

- SBOM 内の名称について、SBOM 利用者の視点で名称設定を行うことで、SBOM 共有後の手戻りをなくすことができる。

解析したコンポーネントの情報に基づき、SBOMを作成する。SBOMの作成に当たっては、SBOMに含める項目、フォーマット、出力ファイル形式等の SBOM に関する要件を事前に決定する必要がある。これらの要件について、規制・要求事項において、SBOM のフォーマットや項目が定められている場合がある。SBOM のフォーマットでは、情報なし（NOASSERTION）も許容されるが、SBOM 作成と共有の目的を鑑み、正確な情報を不足なく SBOM に記載することが望まれる。なお、サードパーティや OSS コミュニティ等の第三者から提供されたコンポーネントを使用している場合は、当該コンポーネントの SBOM の提供を受けることができる場合もある。第三者から SBOM の提供を受けることで、効率的に SBOM が作成できるほか、自社で作成した SBOM の精査に当たって活用することも想定される。ただし、コミュニティや個人に対して SBOM の提供を依頼すべきか否かについては、契約やライセンスの問題があることに注意が必要である。また、第三者から提供されたコンポーネントを自組織にて改変して使用している場合は、提供を受けた SBOM をそのまま利用できなくなるため注意が必要である。加えて、SBOM の共有先となりうるソフトウェアの利用者及びサプライヤーから指定される場合もあるため、自社の状況を鑑み、SBOM の要件を決定することが必要となる。具体的な SBOM 作成方法はツールによって異なるため、導入した SBOM ツールの取扱説明書や README ファイルを確認いただきたい。

SBOM は作成するだけでなく継続的に管理することが求められるところ、SBOM の作成日時は明確に記録することが求められる。また、ソフトウェアサプライチェーンの透明性を高めるために、必要に応じて作成した SBOM を対象ソフトウェアの利用者や納入先に共有することが望まれるが、共有に際して、作成した SBOM のフォーマットが有効であり、必要な情報が含まれているか、確認することが求められる。

作成された SBOM には、コンポーネントの情報だけでなく、プロジェクト名称等のツール上で設定した情報も含まれる場合がある。この情報について、SBOM 利用者が活用しやすい情報となっているかを検討することが望まれる。開発段階から SBOM ツールでコンポーネントを管理する場合、そこで使用しているプロジェクト名称やバージョン情報が SBOM に反映されるため、これまで社内のみで使われていた情報が SBOM 利用者に共有される可能性がある。SBOM 内の名称について、SBOM 利用者も理解できる名称設定を行うことで、SBOM 共有後の手戻りをなくすことができる。

5.3. SBOM の共有

【SBOM 導入に向けた実施事項】

□ 対象ソフトウェアの利用者及び納入先に対する SBOM の共有方法を検討した上で、必要に応じて、SBOM を共有する。

□ SBOM の共有に当たって、SBOM データの改ざん防止のための電子署名技術等の活用を検討

する。

【SBOM 導入に向け認識しておくべきポイント】

- 納入先が利用する SBOM ツールによって、採用可能な SBOM 共有方法が異なる。
- 利用者に対する SBOM 共有について、様々な方法が想定される。利用者に対して SBOM 共有を行う場合、それぞれの方法の長所短所を踏まえて検討する。

ソフトウェアサプライチェーンの透明性を高める観点で、必要に応じて、ソフトウェアの利用者や納入先に対して作成した SBOM を共有することが望まれる。規制・要求事項として SBOM の共有が求められている場合には、当該事項で規定された内容に則り、適切な対象者に対して、適切な方法で SBOM を共有することが必要となる。共有方法の検討に際して、多くの SBOM は、コンポーネントのバージョンアップ等により作成後も動的に内容が変わることに留意する必要がある。なお、2.5 で記載のとおり、SBOM の公開は必須ではなく、SBOM 作成者やサプライヤーの判断で SBOM の共有方法を判断することが望まれる。

納入先に SBOM を共有する場合、納入先が利用する SBOM ツールによって共有の方法が異なる。一般的に、自組織と納入先とが同じ SBOM ツールを利用している場合、SBOM の共有は比較的容易に実施することができ、特に有償の SBOM ツールの場合、自組織と利用者・納入先とがクラウド上で同一の SBOM ツールを使用することにより、SBOM を共有できるケースもある。一方、自組織と利用者・納入先とが異なる SBOM ツールを用いている場合、ツールによってインポートできる SBOM 形式やフォーマットに制約がある場合があり、事前に納入先と、SBOM 共有方法や共有する SBOM の内容について協議することが望まれる。現状では、ほかのツールで生成した SBOM をインポートして脆弱性管理等に活用することができるツールは限られているため、利用者・納入先との間の協議の際には注意が必要である。

利用者に対する SBOM 共有について、様々な方法が想定される。例えば、製品内から SBOM を確認できるよう製品に組み込む方法、利用者がアクセス可能なリポジトリにおいて公開する方法、Web ページ上で公開する方法、共通の SBOM ツールを用いて SBOM データを共有する方法等が想定される。利用者に対する SBOM 共有を行う場合、SBOM 対象ソフトウェアの特性や更新の頻度、利用者における SBOM 利用状況等を踏まえ、どの方法で共有するかを検討することが望まれる。なお、SBOM 共有時の SBOM データ自体の信頼性を確保する目的で、改ざん防止のための電子署名技術や分散型台帳技術等の活用を検討することが求められる。

6. SBOM 運用・管理フェーズにおける実施事項・認識しておくべきポイント

SBOM のメリットを享受するために、作成した SBOM を運用・管理することが求められる。本章では、SBOM 運用・管理フェーズにおいて SBOM 導入組織が実施すべき事項や、SBOM 導入組織が認識しておくべきポイントを示す。

6.1. SBOM に基づく脆弱性管理、ライセンス管理等の実施

【SBOM 導入に向けた実施事項】

□ 脆弱性に関する SBOM ツールの出力結果を踏まえ、深刻度の評価、影響度の評価、脆弱性の修正、残存リスクの確認、関係機関への情報提供等の脆弱性対応を行う。

□ ライセンスに関する SBOM ツールの出力結果を踏まえ、OSS のライセンス違反が発生していないかを確認する。

【SBOM 導入に向け認識しておくべきポイント】

- SBOM ツールが出力した脆弱性情報やライセンスに関する情報が誤っている場合があり、出力結果を確認する必要がある。
- SBOM ツールでコンポーネントの EOL を特定できない場合、別途個別に調査する必要がある。

SBOM 運用・管理フェーズでは、作成された SBOM に基づき、脆弱性管理やライセンス管理等を行う。前述のとおり、SBOM はソフトウェア管理の一手法であるため、作成することが目的ではなく、SBOM を用いて適切なソフトウェア管理を行うことが重要となる。よって、第三者から提供された SBOM データに対しても、脆弱性管理やライセンス管理等、同様の対応を実施することが求められる。脆弱性管理に当たっては、SBOM ツールの出力結果を踏まえ、ソフトウェアに含まれるコンポーネントの脆弱性有無を確認するとともに、脆弱性が確認された場合には、当該脆弱性に対する対応を実施することが必要となる。具体的な脆弱性対応として、脆弱性の箇所を特定し、影響範囲を分析するとともに、リスクの推定及び評価を行い、リスクの受容可能性を確認、脆弱性対応の優先付けが望まれる。そして、関連するセキュリティ問題の特定を行った上で、脆弱性の深刻度を評価し、緊急性の判断を行うことが望まれる。自社のプロプライエタリソフトウェアで脆弱性が確認された場合、関連するソフトウェア利用者に対して適切に通知するほか、OSS や汎用ソフトウェア等のサードパーティコンポーネントで脆弱性が確認された場合、当該コンポーネントのサプライヤーに対して脆弱性を通知することが望まれる。なお、脆弱性の影響範囲分析においては、ソースコードのみならず、要件定義書、仕様書、テスト仕様書等の開発文書の更新必要範囲の特定・分析も必要であることに留意すべきである。この点の対策例として、2021 年度

の実証では、SBOM ツールと既存の構成管理ツールを連携させることで、脆弱性の影響範囲特定の工数削減につながる可能性が確認された。

手動で SBOM を管理する場合、一つ一つの脆弱性の有無を手作業で特定するほか、個別に脆弱性を評価し、さらに対応方針を個々に検討する必要がある。脆弱性の情報が日々更新されることを踏まえると、手動での SBOM 運用・管理は非現実的である。よって、図 6-1 に示すとおり、脆弱性管理においても、SBOM ツールを用いた管理が想定される。有償・無償の SBOM ツールにより、脆弱性マッチングの範囲に差が大きいことに留意が必要である。無償ツールには脆弱性マッチングの機能がないものもあり、有償ツールでは、NVD、JVN のような公的な脆弱性情報データベース以外に、独自に脆弱性情報データベースを強化し、脆弱性マッチングの範囲を拡大したものがある。有償の SBOM ツールによっては、解析されたコンポーネントと脆弱性情報を自動でマッチングし、その脆弱性の深刻度やリスク、対処方法等を提示するため、脆弱性の発見から深刻度評価、対処方法の特定までを非常に迅速に行うことができるツールも存在する。ただし、脆弱性情報が特定されたとしても具体的な対処方法までは提示されない場合は、個々の脆弱性の内容を踏まえて対処方法を別途検討する必要がある。

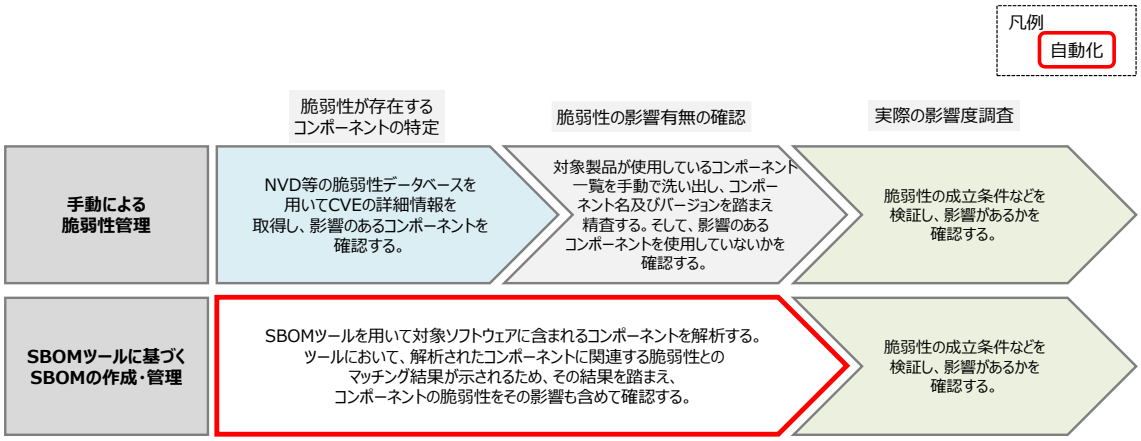

図 6-1 手動と SBOM ツールでの脆弱性管理手順の比較

SBOM ツールに基づく脆弱性管理に当たって認識しておくべきポイントとして、SBOM ツールが出力した脆弱性情報に誤りがあることが挙げられる。実証において利用した無償の SBOM ツールにおいて、脆弱性の深刻度が誤って出力されたケースがあり、手動で脆弱性情報を調査することが必要となった。コンポーネントの解析に当たって誤検出や検出漏れが生じる可能性があるところ、脆弱性情報の出力結果においても誤りが生じる可能性もあり、出力結果を確認することが必要となる。SBOM ツールによっては、NVD といった公開脆弱性情報データベースに掲載されている脆弱性情報だけでなく、ツールベンダー独自の調査に基づく脆弱性情報を活用して脆弱性のマッチングを行うツールもあり、幅広い脆弱性の情報に基づく脆弱性管理が可能となる場合がある。

ライセンス管理も同様である。SBOM ツールの出力結果を踏まえ、ソフトウェアに含まれるコンポーネントのライセンスコンプライアンスの状況を確認するとともに、当該コンポーネントの想定利用方法に対して遵守不可能又は遵守困難なライセンス条件が確認された場合には、利用コンポーネント自体の変更や

利用方法の変更等、対応を実施することが必要となる。脆弱性管理と同様に、手動での管理ではなくSBOM ツールでの管理が現実的である。

SBOM ツールを用いることで、対象ソフトウェアに含まれるコンポーネントにおける脆弱性やライセンスの情報を効率的に特定できる一方で、ツールを用いてコンポーネントの EOL を特定することは一般に難しく、手作業で特定することが必要となる。EOL に関する情報がないコンポーネントも存在するところ、可能な限りこのようなコンポーネントを使用しない等を設計時に考慮することが望まれる。

6.2. SBOM 情報の管理

【SBOM 導入に向けた実施事項】

□ 作成した SBOM は、社外からの問合せがあった場合等に参照できるよう、変更履歴も含めて一定期間保管する。

□ SBOM に含まれる情報や SBOM 自体を適切に管理する。

【SBOM 導入に向け認識しておくべきポイント】

- 自動で脆弱性情報が更新・通知される SBOM ツールを用いることで、新たな脆弱性に関する情報を即座に把握することができる。ツールを用いた自動管理ができない場合、担当者を別途設置する等運用面でカバーする必要があるが、対応工数を要する。
- SBOM の管理は、組織内の PSIRT に相当する部門が対応することが効果的である。PSIRT に相当する部門が存在しない場合、品質管理部門にて対応することが効果的である。

作成した SBOM は、社外からの問合せがあった場合等に参照できるよう、変更履歴も含めて一定期間保管する。保管期間は、一般的に対象製品が市場に流通している間や対象サービスが提供されている期間は最低限としつつ、販売終了後も、保証期間、サポート提供期間、交換部品の提供期間等に必要に応じて SBOM を参照する可能性があるため、あらかじめ参照できるように準備しておく必要がある。さらに、使用しているコンポーネントのライセンスの条件で製品提供終了後 3 年等の個別の指定がある場合にはその期間も考慮する必要がある。また、出荷済製品と SBOM 情報とを対応づけられるよう、SBOM の改変履歴も含めて資産管理システム等で保管することも想定される。

SBOM の対象としたソフトウェアの内容が動的に変わるほか、脆弱性の情報も日々更新されることを踏まえ、SBOM に含まれる情報は定期的に更新し、管理する必要がある。自動で脆弱性情報が更新・通知される SBOM ツールを用いることで、新たな脆弱性に関する情報を即座に把握することができる。ツールを用いた自動管理ができない場合、担当者を別途設置する等運用面でカバーすることとなるが、ツールでの管理と比較して対応工数を要することに留意が必要である。

SBOM の管理体制について、脆弱性管理という観点を踏まえると、組織内の PSIRT 又はそれに類する部門が主導して SBOM の管理を行うのが望ましい。また、作成された SBOM を PSIRT が活用することで、実際に利用者の環境で用いられている OSS の絞り込みに必要な工数が削減され、より効率的な脆弱性対応や監視が可能になる。PSIRT に相当する部門が存在しない場合でも、脆弱性管理は一定のポリシーの下で実施されることが必要であり、例えば、品質管理部門にて SBOM を管理することが望まれる。会社横断で品質管理を担うチームが存在するようであれば、品質管理の一貫として、成果物としての SBOM の定義や管理、SBOM 活用による脆弱性の対応を行うことで、一定のポリシーの下での運用が可能になる。もし、品質管理部門も存在しない場合、まずは特定の製品開発チームで SBOM ツールを導入し、SBOM の作成から運用・管理に至るノウハウを蓄積することから始めることが期待される。その後、得られたノウハウを別の開発チームに横展開し、チームごとで SBOM 導入を進めることで、社内の SBOM 導入レベルを向上することが望まれる。

7. 付録：チェックリスト・用語集等

7.1. SBOM 導入に向けた実施事項チェックリスト

SBOM 導入に関する環境構築・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 つのフェーズにおける実施事項をチェックリストとして以下にまとめる。

表 7-1 SBOM 導入に向けた実施事項チェックリスト

フェーズ	ステップ	実施事項	チェック
環境構築・体制整備フェーズ	SBOM 適用範囲の明確化	対象ソフトウェアの開発言語、コンポーネント形態、開発ツール等、対象ソフトウェアに関する情報を明確化する。	□
		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利用者及びサプライヤーとの契約形態・取引慣行を明確化する。	□
		対象ソフトウェアの SBOM に関する規制・要求事項を確認する。	□
		SBOM 導入に関する組織内の制約（体制の制約、コストの制約等）を明確化する。	□
		整理した情報に基づき、SBOM 適用範囲（5W1H）を明確化する。	□
	SBOM ツールの選定	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 ツールの選定の観点を整理する。 （選定の観点の例：機能、性能、解析可能な情報、解析可能なデータ形式、コスト、対応フォーマット、コンポーネント解析方法、サポート体制、他ツールとの連携、提供形態、ユーザーインターフェース、運用方法、対応するソフトウェア開発言語、日本語対応等）	□
		整理した観点に基づき、複数の SBOM ツールを評価し、選定する。	□
	SBOM ツールの導入・設定	SBOM ツールが導入可能な環境の要件を確認し、整備する。	□
		ツールの取扱説明書や README ファイルを確認して、SBOM ツールの導入・設定を行う。	□

フェーズ	ステップ	実施事項	チェック
	SBOM ツールに関する学習	ツールの取扱説明書や README ファイルを確認して、SBOM ツールの使い方を習得する。	□
		ツールの使い方に関するノウハウや各機能の概要は記録し、組織内で共有する。	□
SBOM 作成・共有フェーズ	コンポーネントの解析	SBOM ツールを用いて対象ソフトウェアのスキャンを行い、コンポーネントの情報を解析する。	□
		SBOM ツールの解析ログ等を調査し、エラー発生や情報不足による解析の中断や省略がなく、解析が正しく実行されたかを確認する。	□
		コンポーネントの解析結果について、コンポーネントの誤検出や検出漏れがないかを確認する。	□
	SBOM の作成	作成する SBOM の項目、フォーマット、出力ファイル形式等の SBOM に関する要件を決定する。	□
		SBOM ツールを用いて、当該要件を満足する SBOM を作成する。	□
	SBOM の共有	対象ソフトウェアの利用者及び納入先に対する SBOM の共有方法を検討した上で、必要に応じて、SBOM を共有する。	□
		SBOM の共有に当たって、SBOM データの改ざん防止のための電子署名技術等の活用を検討する。	□
SBOM 運用・管理フェーズ	SBOM に基づく脆弱性管理、ライセンス管理等の実施	脆弱性に関する SBOM ツールの出力結果を踏まえ、深刻度の評価、影響度の評価、脆弱性の修正、残存リスクの確認、関係機関への情報提供等の脆弱性対応を行う。	□
		ライセンスに関する SBOM ツールの出力結果を踏まえ、OSS のライセンス違反が発生していないかを確認する。	□
	SBOM 情報の管理	作成した SBOM は、社外からの問合せがあった場合等に参照できるよう、変更履歴も含めて一定期間保管する。	□
		SBOM に含まれる情報や SBOM 自体を適切に管理する。	□

7.2. 用語集

7.2.1. SBOM やソフトウェアに関する用語の定義

- EOL（End of Life）
  製品やサービスにおいて販売やサポートが終了し、それ以上継続して使用すべきでないとされる使用期限のこと。
- OTS（Off-The-Shelf）
  サプライヤーが一般的に利用するソフトウェアのコンポーネントであり、サプライヤーが完全なソフトウェアライフサイクル管理を主張できないコンポーネントのこと。
- OSS（Open Source Software）
  ソフトウェアのソースコードが公開され、利用や改変、再配布を行うことが誰に対しても許可されているソフトウェアのこと。
- SBOM（Software Bill of Materials）
  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（一覧表）のこと。コンポーネントやその依存関係をすべて表現している場合もある。また、どこの部分が欠けているかという情報が含まれている場合もある。OSS だけではなくプロプライエタリソフトウェアに活用することもでき、広く一般に公開するほか関係者だけに提示するという使用方法も存在する。
- SBOM 作成者（SBOM Author）
  SBOM を作成するエンティティのこと。必ずしも SBOM 作成者とサプライヤーが一致しないことに留意する必要がある。SBOM 作成者とサプライヤーが異なる場合、あるエンティティが、異なるサプライヤーによって作成又は含有された構成要素について主張した場合、当該エンティティが SBOM 作成者としてみなされる。
- SBOM 利用者（SBOM Consumer）
  SBOMを入手するエンティティのこと。ほとんどのエンティティは、サプライヤーであると同時にSBOM利用者でもあり、SBOM データを有するコンポーネントを自身のソフトウェアに使用し、そのデータを下流に渡すことが可能となる。
- SBOM 項目（SBOM Entry）
  SBOM のコンポーネントと関連する属性のこと。行列形式の SBOM の場合、行に該当する。
- SBOM システム（SBOM System）
  SBOM を作成、共有、活用、管理する能力を提供する一連の要素及びプロセスのこと。
- SBOM ツール（SBOM Tool）
  SBOM の作成、共有、活用、管理することができるツールのこと。SBOM 管理ツール、OSS 管理ツール、ソフトウェア構成解析（SCA）ツール等とも呼ばれることがあり、パッケージとして提供されるツ

ールのほか、クラウドソフトウェアとして提供されるツールも存在する。

- VEX（Vulnerability Exploitability Exchange）
  特定の製品が既知の脆弱性の影響を受けるかどうかを示すセキュリティアドバイザリーの一つの形態のこと。

- 依存関係（Dependency Relationship）
  ソフトウェア Y に、上流のコンポーネント X が含まれる関係性の特徴づけのこと。

- エンティティ（Entity）
  ソフトウェアやコンポーネントに関連する企業、団体、組織、個人のこと。

- 関係性表明（Relationship Assertion）
  ある作成者における、他サプライヤーのコンポーネントの知識の範囲のこと。Unknown, Root/None, Partial, Known の 4 つのカテゴリーが存在する。

- コードベース（Codebase）
  特定のソフトウェア、アプリケーション、コンポーネント等を構築するために使用されるソースコード全体のこと。

- コンポーネント（Component）
  サプライヤーによって定義されるソフトウェアの単位のこと。コンポーネントは、サプライヤーにより構築、パッケージ化又は納入される時点で定義される。ソフトウェア製品、機器、ライブラリ、単一ファイル等も一つのコンポーネントに位置づけられるほか、OS、オフィススイート、データベースシステム、自動車、自動車のエンジンコントロールユニット（ECU）、医療用画像処理装置、インストールパッケージ等のコンポーネントの集合体も、コンポーネントとなる。多くのコンポーネントがサブコンポーネントを含む。

- 最小要素（Minimum Elements）
  米国の大統領令（Executive Order 14028）に基づき NTIA より 2021 年 7 月 12 日に発表された SBOM に含めるべき最小要素のこと。データフィールド、自動化サポート、プラクティスとプロセスの 3 つのカテゴリーに基づく具体的な定義が示されている。

- サブコンポーネント（Subcomponent）
  コンポーネントに含まれるコンポーネントのこと。

- サプライヤー（Supplier）
  コンポーネントを開発、定義及び識別するエンティティで、理想的には当該コンポーネントに関連する SBOM を作成するエンティティのこと。サプライヤーは、製造者、ベンダー、デベロッパー、システムインテグレーター、保守事業者、サービスプロバイダーとも呼ばれる。ほとんどのサプライヤーは SBOM の利用者でもある。上流のコンポーネントを持たないサプライヤーは、ルートエンティティとも呼ばれる。

- シンボリックリンク（Symbolic Link）
  OS のファイルシステムにおける機能の一つで、特定のファイルやディレクトリを示す別のファイルのこと。

- 推移的依存関係（Transitive Dependency）

ソフトウェア X にコンポーネント Y が含まれ、コンポーネント Y にコンポーネント Z が含まれる場合に、ソフトウェア X にコンポーネント Z が含まれる関係性の特徴づけのこと。

- スニペット（Snippet）
  ソースコード内のコード断片のこと。
- 属性（Attribute）
  コンポーネントに関する特性や情報のこと。行列形式の SBOM の場合、列に該当する。
- ソフトウェア構成解析（SCA：Software Composition Analysis）
  狭義には、製品が利用しているコンポーネントを識別すること。一般的には、特定した各コンポーネントの脆弱性やライセンスリスクを管理することを指す。
- 中間サプライヤー（Intermediate Supplier）
  上流のコンポーネントを、下流工程として新たなコンポーネントに加工するサプライヤーのこと。多くのサプライヤーは中間サプライヤーとして扱われる。
- プライマリーコンポーネント（Primary Component）
  SBOM によって記述される対象のコンポーネントのこと。
- プロプライエタリソフトウェア（Proprietary Software）
  ソフトウェア配布者がその知的財産を保持しており、改変や複製が制限されているソフトウェアのこと。
- 要素（Element）
  SBOM システムの一部のこと。
- ランタイムライブラリ（Run-time Library）
  プログラムの実行時に必要なライブラリのこと。

7.2.2. サイバーセキュリティに関する用語の定義

- CVSS（Common Vulnerability Scoring System）
  FIRST（Forum of Incident Response and Security Teams）が管理する脆弱性の深刻度を同一の基準の下で定量的に比較できる評価方法のこと。0.0～10.0 の間でスコアが定まる。
- CWE（Common Weakness Enumeration）
  Common Weakness Enumeration の略でソフトウェアにおけるセキュリティ上の弱点（脆弱性）の種類を識別するための共通の基準のこと。米国非営利団体 MITRE を中心として仕様策定。
- ISMS（Information Security Management System）
  組織のマネジメントとして、自らのリスクアセスメントにより必要なセキュリティレベルを決め、プランを持ち、資源を配分して、システムを運用するための仕組みのこと。国際規格 ISO/IEC 27001 に要求事項が定められている。

- OWASP（Open Web Application Security Project）
  Web をはじめとするソフトウェアのセキュリティ関する情報共有と普及啓発を目的とした、オープンソースソフトウェアコミュニティのこと。
- PSIRT（Product Security Incident Response Team）
  自社製品のセキュリティの向上を担い、インシデントが発生した際に対応する組織のこと。
- 脅威（Threat）
  システム又は組織に損害を与える可能性がある、望ましくないインシデントの潜在的な原因のこと。[JIS Q 27000:2014]
- 脅威情報（Threat Intelligence）
  脅威からの保護、攻撃者の活動検知、脅威への対応等に役立つ可能性のある情報のこと。[NIST SP 800-150]
- 脅威分析（Threat Analysis）
  機器やソフトウェア、システム等に対する脅威を抽出し、その影響を評価すること。主に、製品の要件定義、設計フェーズにて行われる。
- サイバー攻撃（Cyber Attack）
  資産の破壊、暴露、改ざん、無効化、盗用、又は認可されていないアクセス若しくは使用の試みのこと。[JIS Q 27000:2014]
- サイバーセキュリティ（Cybersecurity）
  電子データの漏えい・改ざん等や、期待されていた機器、IT システム、制御システム等の機能が果たされないといった不具合が生じないようにすること。
- サプライチェーン（Supply Chain）
  複数の開発者間でリンクされたリソース・プロセスで、製品とサービスについて、調達にはじまり設計・開発・製造・加工・販売及び購入者への配送に至る一連の流れのこと。[ISO 28001:2007, NIST SP 800-53 Rev.4]
- 脆弱性（Vulnerability）
  一つ以上の脅威によって付け込まれる可能性のある、資産又は管理策の弱点のこと。[JIS Q 27000:2014]
- 認証（Authentication）
  エンティティの主張する特性が正しいという保証の提供のこと。[JIS Q 27000:2014]
- 認可（Authorization）
  アクセス権限に基づいたアクセス機能の提供を含む権限の付与のこと。 [ISO 7498-2:1989]
- プロトコル（Protocol）
  複数の主体が滞りなく信号やデータ、情報を相互に伝送できるよう、あらかじめ決められた約束事や手順の集合のこと。

- マルウェア（Malware）
  許可されていないプロセスの実施を試みることによって、情報システムの機密性・完全性・可用性に悪影響をもたらすソフトウェア又はファームウェアのこと。[NIST SP 800-53 Rev.4]
  セキュリティ上の被害を及ぼすウイルス、スパイウエア、ボット等の悪意を持ったプログラムを指す総称。
- リスク（Risk）
  目的に対する不確かさの影響のこと。[JIS Q 27000:2014]

7.3. 参考情報

7.3.1. SBOM に関する参考文書

本項では、国内外の政府関係機関が発表している SBOM に関する参考文書を列挙する。

- **米国 NTIA : Roles and Benefits for SBOM Across the Supply Chain（2019 年 11 月）**
  ソフトウェア開発者、購入者、利用者のそれぞれの視点で、SBOM を利用することのメリットをまとめた文書。メリットは、コスト、セキュリティ、ライセンス、コンプライアンス、サプライチェーンにおけるソフトウェアの安定性ごとに整理されている。
  https://www.ntia.gov/files/ntia/publications/ntia_sbom_use_cases_roles_benefits-nov2019.pdf
- **米国 NTIA : Software Bill of Materials (SBOM)（2020 年 8 月）**
  SBOM の検討背景、ソフトウェアエコシステムにおける SBOM の役割や有効性、SBOM の概要をまとめた文書。
  https://www.ntia.gov/files/ntia/publications/sbom_overview_20200818.pdf
- **米国 NTIA : SBOM FAQ（2020 年 11 月）**
  SBOM の概要、利用効果、SBOM の生成や配布等に関してまとめた FAQ 集。
  https://www.ntia.gov/files/ntia/publications/sbom_faq_-_20201116.pdf
- **米国 NTIA : Sharing and Exchanging SBOMs（2021 年 2 月）**
  SBOM データがサプライチェーン上でどのように共有されるかに関するオプションを説明した文書。SBOM データを作成したサプライヤーと SBOM の利用者の負担を最小化することを目的としている。
  https://www.ntia.gov/files/ntia/publications/ntia_sbom_sharing_exchanging_sboms-10feb2021.pdf
- **米国 NTIA : SBOM Tool Classification Taxonomy（2021 年 3 月）**
  SBOM ツールの分類を示した文書。ツールの利用目的を SBOM の作成・利用・変換の 3 つに分類し、それぞれの目的におけるツールのタイプが整理されている。

https://www.ntia.gov/files/ntia/publications/ntia_sbom_tooling_taxonomy-2021mar30.pdf

- **米国 NTIA：Software Identification Challenges and Guidance（2021 年 3 月）**
  ソフトウェアコンポーネントを国際的に一意に識別するための課題を説明した文書。課題を解決するための対処方法・ガイダンスを示すことを目的としている。
  https://www.ntia.gov/files/ntia/publications/ntia_sbom_software_identity-2021mar30.pdf

- **米国 NTIA：SBOM at a Glance（2021 年 4 月）**
  SBOM の活用方法、ソフトウェアサプライチェーンの透明性確保において SBOM が果たす役割、参考となる文書についてまとめた文書。SBOM に含めるべき情報も整理されている。なお、JPCERT/CC によって日本語訳された文書も公開されている。
  https://www.ntia.gov/files/ntia/publications/sbom_at_a_glance_apr2021.pdf
  https://www.ntia.gov/files/ntia/publications/sbom_at_a_glance_ja.pdf（日本語版）

- **米国 NTIA：SBOM Options and Decision Points（2021 年 4 月）**
  SBOM に関して現在の手法で実現可能なことや、SBOM のサプライヤー及び利用者の間でのニーズの明確化を支援することを目的とした文書。
  https://www.ntia.gov/files/ntia/publications/sbom_options_and_decision_points_20210427-1.pdf

- **米国 NTIA：The Minimum Elements For a Software Bill of Materials (SBOM)（2021 年 7 月）**
  SBOM の最小要素を定義した文書。最小要素は 3 つのカテゴリーに分けられ、各カテゴリーの概要や SBOM に含めるべき具体的な項目が定義されている。
  https://www.ntia.doc.gov/files/ntia/publications/sbom_minimum_elements_report.pdf

- **米国 NTIA：Vulnerability-Exploitability eXchange (VEX) – An Overview（2021 年 9 月）**
  特定のソフトウェアコンポーネントが脆弱性の影響を受けるかどうかを判断する指標である VEX について、その概要を説明した文書。VEX は、特定の製品に存在する脆弱性の状態を表すもので、文書では 4 段階で状態を表す方針が示されている。
  https://www.ntia.gov/files/ntia/publications/vex_one-page_summary.pdf

- **米国 NTIA：How-To Guide for SBOM Generation（2021 年 10 月）**
  SBOM 生成の手引として SBOM 生成のための情報収集方法と、具体的な SBOM 生成方法の 2 つの観点をまとめた文書。本手引は、NTIA によるヘルスケア分野の SBOM PoC を通じて策定

されたが、ヘルスケア分野だけでなくあらゆる業界における SBOM 生成においての活用が期待されている。
https://www.ntia.gov/files/ntia/publications/howto_guide_for_sbom_generation_v1.pdf

- **米国 NTIA：Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM)（初版：2019 年 11 月、改定：2021 年 10 月）**
  SBOM の概念や関連用語、ソフトウェアコンポーネントの表現に関する基本的な考え方を示すとともに、SBOM の作成プロセスを示した文書。
  https://www.ntia.gov/files/ntia/publications/ntia_sbom_framing_2nd_edition_20211021.pdf
- **米国 NTIA：SBOM Myths vs. Facts（2021 年 11 月）**
  SBOM のメリットを正しく示すことを目的に、SBOM に関する代表的な誤解（神話）と、その誤解を解くための事実を整理した文書。
  https://www.ntia.gov/files/ntia/publications/sbom_myths_vs_facts_nov2021.pdf
- **米国 NTIA：Software Suppliers Playbook: SBOM Production and Provision（2021 年 11 月）**
  ソフトウェアサプライヤーを対象とした SBOM 生成に関するプレイブック。本プレイブックでは、「SBOM 作成手順」、「SBOM 作成に当たって考慮すべき事項」及び「SBOM に関する補足事項」の 3 つの事項についてまとめられている。
  https://www.ntia.gov/files/ntia/publications/software_suppliers_sbom_production_and_provision_-_final.pdf
- **米国 NTIA：Software Consumers Playbook: SBOM Acquisition, Management, and Use（2021 年 11 月）**
  ソフトウェア利用者を対象とした SBOM 利用に関するプレイブック。本プレイブックでは、「サプライヤーから SBOM を取得する際の注意点」、「SBOM 活用のプロセス及びプラットフォーム」、「SBOM の知的財産及び機密保持」に関する注意点等がまとめられている。
  https://www.ntia.gov/files/ntia/publications/software_consumers_sbom_acquisition_management_and_use_-_final.pdf
- **米国 NTIA：Survey of Existing SBOM Formats and Standards - Version 2021（初版：2019 年、改定：2021 年）**
  既存の SBOM フォーマットや基準に関する調査結果や今後の課題に関して整理した文書。既存の SBOM フォーマットについては、SPDX、CycloneDX、SWID の 3 つについて、概要、ユースケース、特徴等がまとめられている。
  https://www.ntia.gov/files/ntia/publications/sbom_formats_survey-version-

2021.pdf

- **米国 NIST：SP 800-218 Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities（2022 年 2 月）**
  ソフトウェアの脆弱性を軽減するためのソフトウェア開発者向けの手法をまとめたフレームワーク文書。手法は 4 つのカテゴリーに分類され、各手法を実践するためのタスクが体系整理されている。
  https://csrc.nist.gov/publications/detail/sp/800-218/final

- **米国 CISA：Vulnerability Exploitability eXchange (VEX) – Use Cases（2022 年 4 月）**
  VEX ドキュメントに含めるべき最小要素を示した文書。また、VEX ドキュメントを作成するための具体的な事例としてユースケースが紹介されている。
  https://www.cisa.gov/sites/default/files/publications/VEX_Use_Cases_Document_508c.pdf

- **米国 CISA：Vulnerability Exploitability eXchange (VEX) - Status Justifications（2022 年 6 月）**
  VEX ドキュメントの最小要素における「脆弱性ステータス」のうち、「脆弱性の影響を受けない（NOT AFFECTED）」ステータスを正当化するための具体的な 5 つの主張を定義した文書。
  https://www.cisa.gov/sites/default/files/publications/VEX_Status_Justification_Jun22.pdf

- **米国 CISA、NSA、ODNI：Securing Software Supply Chain Series - Recommended Practices for Developers（2022 年 9 月）**
  安全なソフトウェアサプライチェーンを確保するため、ソフトウェア開発者に対する推奨事項を整理した文書。本文書は、ソフトウェア開発者、ソフトウェアサプライヤー、ソフトウェア利用者の 3 つの役割ごとに焦点を当てた 3 部のガイダンスシリーズのうち、第 1 部に該当する。文書では、サードパーティコンポーネントを含むソフトウェアの SBOM の作成、脆弱性の評価等が推奨されている。
  https://www.cisa.gov/uscert/sites/default/files/publications/ESF_SECURING_THE_SOFTWARE_SUPPLY_CHAIN_DEVELOPERS.PDF

- **米国 CISA、NSA、ODNI：Securing Software Supply Chain Series - Recommended Practices for Suppliers（2022 年 10 月）**
  安全なソフトウェアサプライチェーンを確保するため、ソフトウェアサプライヤーに対する推奨事項を整理した文書。本文書は、ソフトウェア開発者、ソフトウェアサプライヤー、ソフトウェア利用者の 3 つの役割ごとに焦点を当てた 3 部のガイダンスシリーズのうち、第 2 部に該当する。文書では、サプライヤーは、開発者と利用者の間の仲介役として、ソフトウェアの保護や脆弱性に関する対応・通知等が推奨されている。
  https://media.defense.gov/2022/Oct/31/2003105368/-1/-1/0/SECURING_THE_SOFTWARE_SUPPLY_CHAIN_SUPPLIERS.PDF

- **米国 CISA、NSA、ODNI : Securing Software Supply Chain Series - Recommended Practices for Customers（2022 年 11 月）**
  安全なソフトウェアサプライチェーンを確保するため、ソフトウェア利用者に対する推奨事項を整理した文書。本文書は、ソフトウェア開発者、ソフトウェアサプライヤー、利用者の 3 つの役割ごとに焦点を当てた 3 部のガイダンスシリーズのうち、第 3 部に該当する。文書では、サプライヤーへの SBOM の要求や SBOM を基にしたソフトウェアの脆弱性評価等が推奨されている。
  https://media.defense.gov/2022/Nov/17/2003116445/-1/-1/0/ESF_SECURING_THE_SOFTWARE_SUPPLY_CHAIN_CUSTOMER.PDF
- **米国 CISA : Software Bill of Materials (SBOM) Sharing Lifecycle Report（2023 年 4 月）**
  SBOM の共有ライフサイクルに関するレポート。SBOM が作成者から利用者に共有されるまでに 3 つの基本フェーズがあるとし、各フェーズの概要とフェーズごとの洗練度合いが示されている。洗練度合いは、各フェーズを実施するために必要なコスト、リソース等の相対量を表しており、低・中・高のいずれかで定義される。また、SBOM の共有に関する現況を理解するため、関係組織に対して、組織が SBOM をどのように共有しているかのインタビュー結果が紹介されている。
  https://www.cisa.gov/resources-tools/resources/software-bill-materials-sbom-sharing-lifecycle-report
- **米国 CISA : Minimum Requirements for Vulnerability Exploitability eXchange (VEX)（2023 年 4 月）**
  VEX ドキュメントの最小要件を示した文書。本文書では、VEX ドキュメントを構成する項目と各項目に含まれる要素が示され、それぞれにおける必須項目・必須要件が定義されている。文書では、必須要件を VEX ドキュメントの最小要件と位置づけている。
  https://www.cisa.gov/resources-tools/resources/minimum-requirements-vulnerability-exploitability-exchange-vex
- **米国 CISA : Types of Software Bill of Materials (SBOM)（2023 年 4 月）**
  SBOM のタイプを定義した文書。本文書では、ソフトウェアライフサイクルの各フェーズで生成される可能性がある SBOM をタイプ分類し、各タイプの一般的な SBOM 生成方法、利点、制約が示されている。
  https://www.cisa.gov/resources-tools/resources/types-software-bill-materials-sbom

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 SBOM ツールの一例を紹介する。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。なお、本項に記載のツールはあくまで参考情報として手引作成時点での一例を挙げているにすぎず、特定のツールの利用を推奨するものではない。適切なツール選定のために、本項記載のツールに限定せず、市場に存在する様々なツールについて、4.2 に記載している観点を踏まえて評価・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
1	Black Duck	Synopsys, Inc.	・ コードマッチング、コンテナ解析、バイナリ解析等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能 ・ 脆弱性管理に関して、NVD 及び独自ソースの脆弱性情報を活用し、迅速な脆弱性検出が可能 ・ セキュリティ、ライセンス、コンプライアンス、運用等の観点でリスクを定量分析して管理 ・ 日本語の GUI を提供
2	Checkmarx SCA	Checkmarx Ltd.	・ Github における多くのリポジトリをホストすることで、使用されている OSS の自動的な追跡が可能 ・ 脆弱性管理に関して、ソースコード内に存在する脆弱な OSS のパッケージを検出し、対処方法を提示 ・ OSS のライセンスリスクを可視化し、効果的なライセンス管理が可能
3	FOSSA	FOSSA, Inc.	・ 脆弱性検出と継続的なリスクモニタリングに加え、必要な解決方法の提示により、効果的な脆弱性管理を支援 ・ 高品質なポリシー機能と強力なスキャン、柔軟なレポートにより、コンプライアンス遵守及びライセンス管理を促進 ・ 開発環境との統合によりアジャイル・DevOps プロセスにおける SBOM 管理の自動化と効率化を実現 ・ SPDX 等の複数のレポート形式を選択可能なほか、複数の形式の SBOM をインポートして脆弱性管理が可能

No.	名称	開発者	特徴
4	FossID	FossID AB	・ コンポーネント、パッケージ、ライブラリだけでなく、OSS のスニペットレベルまで検出可能 ・ コンポーネントとバージョン情報に基づく解析ではなく、スニペットレベルの情報に基づいた解析によって、脆弱なソフトウェアを検出可能 ・ ライセンス、著作権、脆弱性等の情報を含む SPDX の形式で SBOM を生成し、管理が可能 ・ ライセンス管理に関して、強い/弱いコピーレフトや非商用のライセンス等の幅広い OSS のライセンス違反リスクを可視化
5	Insignary Clarity	Insignary, Inc.	・ バイナリファイルを解析して内包するコンポーネントを特定（ソースコードやリバースエンジニアリングは不要） ・ バイナリファイルのパターンを利用して解析するため、ビルド環境に依存せず解析可能 ・ クラウド及びオンプレミスのソフトウェアに対して適用可能 ・ クラウド型で提供されており、導入が容易
6	MEND SCA	WhiteSource Software, Inc.	・ クラウドサービス、デスクトップアプリ、組込ソフトウェア等で利用されている OSS のライブラリ・フレームワーク等を漏れなく検出可能 ・ 脆弱性管理に関して、常に最新状態に維持されている独自の脆弱性データベースを用いて脆弱性発生時に即時アラートを発行するほか、影響度や深刻度のスコアや解決方法の詳細情報も提供可能 ・ ライセンス管理に関して、IDE やパッケージマネージャー等の対象となるソフトウェアの開発環境へ本ツールを統合することにより、開発者が新しく OSS のコンポーネントを追加する都度、OSS のライセンス情報を自動で特定可能
7	Revenera SCA	Flexera Software LLC	・ ソースコード、バイナリ等のソフトウェア解析だけでなく、独自の OSS のナレッジデータベースやサードパーティの SBOM データを照合することで、正確な SBOM を作成可能 ・ NVD や独自の脆弱性データベース（Secunia Research）等の複数のソースを活用することで、効果的な脆弱性管理が可能

No.	名称	開発者	特徴
8	Snyk	Snyk, Ltd.	・ 既存の IDE、リポジトリ、ワークフローに組み込んで使用可能 ・ 高度なセキュリティインテリジェンスを使用し、対象となるソフトウェア開発中に脆弱性を監視 ・ 脆弱性管理に関して、脆弱性等に対する実用的な修正アドバイスを提供
9	Sonatype Lifecycle	Sonatype, Inc.	・ IDE やソースコード管理システム等の対象となるソフトウェアの開発環境へ本ツールを統合することで利用可能 ・ 脆弱性管理に関して、ソフトウェアやコンポーネントに含まれる脆弱性、脆弱性のリスクレベルを継続的に監視することで、迅速なアラートを発出可能
10	Veracode SCA	Veracode, Inc.	・ OSS のコンポーネント一覧として、CycloneDX 形式の SBOM を作成可能 ・ 脆弱性管理に関して、検出した脆弱性に対し、脆弱性の対処方法だけでなく、対処すべき優先順位を提示 ・ ライセンス管理に関して、OSS のライセンス違反リスクを検出し、ライセンスの遵守状況を管理することが可能
11	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 ・ 脆弱性管理に関して、オートトリアージ機能で脆弱性の優先度を自動で判断するほか、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 ・ ライセンス管理に関して、OSS のライセンス違反リスクを可視化

(2)　無償ツール

※　アルファベット順

No.	名称	開発者	特徴
1	Augur	CHAOSS	・ ソフトウェアリポジトリに関するデータを収集し、データモデルに正規化 ・ OSS プロジェクトに関するデータを多くの情報源から収集

No.	名称	開発者	特徴
2	BOM Doctor	Sonatype, Inc.	・ Github 上のプロジェクトの URL やパッケージ URL を指定することで、SBOM を生成可能 ・ 生成された SBOM は、コンポーネントの依存関係を含めてツリー上に可視化（既存の CycloneDX 形式の SBOM をアップロードすることでも可視化可能） ・ 対象ソフトウェアに関して、脆弱でないコンポーネントを利用しているか、ライセンス違反していないか等を評価することで、スコアリングを実施
3	Checkov	Bridgecrew, Inc.	・ IaC 用の静的コード分析ツールでもあり、画像や OSS パッケージ用の SBOM ツールとしても活用可能 ・ スキャン結果は、CLI、CycloneDX、JSON、JUnit XML、CSV、SARIF、Markdown 形式で表示可能
4	Daggerboard	NewYork-Presbyterian Hospital	・ SBOM 及び関連する脆弱性を一目で確認・管理できるダッシュボードであり、SPDX 又は CycloneDX のファイルをインポートして脆弱性の検出が可能
5	Dependency-Track	OWASP Foundation	・ NVD、GitHub Advisories 等の複数のソースを活用し、サードパーティ及びオープンソースコンポーネントの既知の脆弱性を特定し、管理することが可能 ・ ソフトウェアコンポーネントのライセンス情報を特定可能 ・ API ファーストの設計により、他システムとの連携が容易
6	FOSSology	Linux Foundation	・ OSS の名称やバージョンの特定はできないものの、対象ソフトウェアに含まれるコンポーネントのライセンス及び著作権を検出し、管理することが可能 ・ Web UI を用いたインポートや解析が可能
7	in-toto	Linux Foundation	・ ソフトウェアがサプライチェーン内で配布中に改ざんされていないことを確認し、ソフトウェアサプライチェーンの整合性を保護するためのフレームワークを提供
8	OSS Review Toolkit (ORT)	Linux Foundation	・ ビルドシステムプラグインの適用等、既存のプロジェクトソースコードを変更する必要なしで SBOM の作成が可能 ・ カスタマイズ可能なポリシールールとライセンス分類に基づき、使用されているソフトウェアのライセンスを評価可能

No.	名称	開発者	特徴
9	SBOM Tool	Microsoft Corporation	・ NPM、NuGet、PyPI 等の様々なパッケージ管理システムと統合して自動検出し、SPDX 形式の SBOM を作成することが可能 ・ Windows、Linux、macOS のプラットフォームで動作可能
10	ScanCode.io	nexB, Inc.	・ ソフトウェア構成分析（SCA）のプロセスをスクリプト化して自動化 ・ アプリケーションのコードベース内の OSS コンポーネントとそのライセンス情報を特定可能
11	Scancode Toolkit	nexB, Inc.	・ スタンドアローンのコマンドラインツールで、インストール、実行、CI/CD 処理パイプラインへの組み込みが容易 ・ スキャン結果を JSON、HTML、CSV、SPDX、独自の形式で保存可能 ・ ライセンス管理に関して、ユーザーによって拡張可能な独自の検出ルールを用いて、OSS コンポーネントのライセンス情報を特定し、管理することが可能
12	SW360	Eclipse Foundation	・ ソフトウェアコンポーネントにおけるセキュリティの脆弱性情報を特定し、管理することが可能 ・ ソフトウェアコンポーネントのライセンス情報を特定し、管理することが可能
13	SwiftBOM	CERT Coordination Center (CERT/CC)	・ 手動入力で SBOM 生成が可能 ・ 作成済み SBOM をインポートし、SBOM をツリー状に表示することが可能
14	Syft & Grype	Anchore Enterprise	・ Syft による SBOM 生成と、Grype による脆弱性検出機能をシームレスに連携可能 ・ CycloneDX、SPDX、Syft 独自のフォーマット等の SBOM フォーマット間で SBOM 情報を変換可能 ・ OS パッケージ、言語パッケージにおける主要な脆弱性を検知し、管理することが可能
15	Trivy	Aqua Security Software, Ltd.	・ 既知の脆弱性、IaC の構成ミス等の様々なセキュリティ問題を検知し、管理することが可能 ・ コンテナイメージ、ファイルシステム等の様々な対象をスキャン可能

7.3.3. SBOM のデータフォーマット

米国 NTIA の SBOM の「最小要素」には「自動化サポート」のカテゴリーが含まれ、SBOM の自動生成や可読性等の自動化をサポートすることが考慮されている。具体的なデータフォーマットとして、これまで国際的な議論がなされてきた SPDX（Software Package Data Exchange）、CycloneDX、Software Identification Tags（SWID タグ）の 3 つのフォーマットが位置づけられている。以降では、これら 3 つのフォーマットに加え、SPDX に基づき日本が開発したフォーマットである SPDX-Lite の概要を示す。なお、SBOM のデータフォーマットは組織を越えて SBOM をやりとりするための一つの規格であり、データフォーマットの選定と SBOM に含めるべきデータフィールドの決定は、SBOM 利用者とサプライヤーとの間で合意の上、決定することが望まれる。

(1) SPDX

SPDX は、Linux Foundation の傘下のプロジェクトによって開発された。2021 年 9 月、ISO/IEC 5962:2021 の規格として、SBOM フォーマットの国際的な標準として認められている。SPDX の詳細な仕様は Web サイト上に公開[24]されており、プロジェクトにおいて検討・更新され続けている。以降では、SPDX の v2.3.0 の概要として、フォーマットの構成、フォーマットの使用例・使用目的、フォーマットの特徴を示す。

1) フォーマットの構成

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[25]、xls 形式、json 形式[26]、YAML 形式[27]、xml 形式[28]がサポートされている。SBOM ドキュメントに含める内容として、セクションと各セクションに分類される項目が規定されている。各セクションの概要を以下に示す。なお、「SBOM ドキュメントの情報（Creation Information）」のセクションのみが必須と定義され、必須ではないほかのセクションについては、SBOM ドキュメント作成者が SBOM に含めるべきと判断する場合に使用する。また、各セクションに規定された項目のうち、該当セクションを使用する場合に必須で含めるべき項目も決められている。

- **SBOM ドキュメントの情報（Creation Information）【必須セクション】:**

---

[24] https://spdx.github.io/spdx-spec/v2.3/

[25] RDF 形式のファイルを解析する方法として、例えば、ファイルに記述されたデータの検索や操作を行うための SPARQL 言語を活用することが知られている。

[26] json 形式のファイルを解析する方法として、例えば、ファイルから必要情報を取得するための jq コマンドを活用する方法が知られている。

[27] Visual Studio Code や IntelliJ IDEA 等の YAML 形式のファイルに対応するツールを使用することで、ファイルの閲覧や解析が容易となる。

[28] xml 形式のファイルを解析する方法として、例えば、ファイルから必要情報を取得するための xmllint コマンドを活用する方法が知られている。

サプライヤーが SBOM ドキュメントを提供し、利用者が SBOM ドキュメントを活用するために必要な情報（SPDX バージョン、SBOM データライセンス、作成者等）を提示するセクション。本セクションは、SPDX による SBOM ドキュメントに必ず含める必要がある。

- **パッケージ情報（Package Information）：**
  SBOM 内にて、製品、コンテナ、コンポーネント等をグループ化するために必要な情報を提示するセクション。
- **ファイル情報（File Information）：**
  製品、コンテナ、コンポーネント等のファイルに関する情報（名称、チェックサム、ライセンス、著作権等）を提示するセクション。
- **スニペット情報（Snippet Information）：**
  あるファイルがほかのリソースから生成されている場合に使用するセクション。本セクションは、ファイルの一部が他のファイルからコピーされたことを示す際に有効である。
- **その他ライセンス情報（Other Licensing Information）：**
  SPDX では、ファイル情報のライセンスを提示するための SPDX ライセンス一覧と呼ばれるライセンス一覧が定義されている。「パッケージ情報」、「ファイル情報」、「スニペット情報」のセクションでは、説明対象となるパッケージ、ファイル、スニペットのライセンス情報として、SPDX ライセンス一覧から選択して使用する。ただし、SPDX ライセンス一覧は、パッケージ、ファイル、スニペットに関するすべてのライセンスを網羅していないため、本セクションで、SPDX ライセンス一覧以外のライセンス情報（プロプライエタリソフトウェアによる制限等）を提示することが可能である。
- **リレーションシップ（Relationships）：**
  SBOM 内における製品、コンテナ、コンポーネント等のファイルやパッケージの関係を提示するセクション。
- **注釈（Annotations）：**
  SBOM をレビューし、そのレビュー結果から得た情報を他者へ共有するために使用されるセクション。加えて、SBOM ドキュメントの作成者が、前述した他セクションや項目に当てはまらない情報を SBOM 内に保管したい場合でも本セクションを使用することが可能である。

2）使用例・使用目的

SPDX の使用例・使用目的として、以下が想定される。

- システムの構成要素と構成要素間の関係性を記述する
- ソフトウェアコンポーネントの知的財産（ライセンス、著作権）を管理する
- ソフトウェアサプライチェーンのリスクアセスメントとコンポーネントを検証する
- ソフトウェアコンポーネント、コンテナコンテンツ等のインベントリーを作成する

- ソースファイルやソーススニペットに遡って実行ファイルを追跡する
- ファイルに埋め込まれたコードの出所を特定する
- ソフトウェアを一意に特定するためのフォーマットである CPE、SWHID（SoftWare Heritage persistent IDentifiers）、パッケージ URL 等を特定のパッケージに関連付け、追加のセキュリティ分析を容易にする

3) データフォーマットの特徴

SPDX の特徴として、以下が挙げられる。

- SBOM の作成対象となるソフトウェアとして、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
- SBOM ドキュメントとして作成された成果物は、提供されたハッシュ値を使用することで SBOM データの改ざん有無を検証することが可能
- 知的財産とライセンス情報のための豊富なリスト（SPDX ライセンス）を持つ
- 他のパッケージ参照システムやセキュリティシステムと連携することが可能
- 複雑なシステムに関連するドキュメントを論理的に分割し、SBOM ドキュメントの各セクションや項目で管理することが可能

(2) SPDX-Lite

SPDX-Lite は、Linux Foundation の傘下のプロジェクトである OpenChain Project における日本企業を中心として活動する、OpenChain Japan Work Group (WG) のライセンス情報サブグループによって開発された日本発のフォーマットである。SPDX-Lite は、SPDX に関する ISO/IEC 5962:2021 規格の一部に含まれ、SPDX に内包される位置づけとして定義されている。SPDX-Lite の詳細な仕様は、SPDX の v2.3.0 の仕様の一部として、Web サイト上に公開[29]されている。以降では、SPDX-Lite の概要として、フォーマットの構成と具体的な項目、フォーマットの使用例・使用目的、フォーマットの特徴を示す。

1) フォーマットの構成と具体的な項目

SPDX-Lite フォーマットにおける 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、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。SBOM ドキュメントに含める内容として、前述した、SPDX における「SBOM ドキュメントの情報」と「パッケージ情報」のセクションに分類される必須項目とその他の基本情報で構成される。

[29] https://spdx.github.io/spdx-spec/v2.3/SPDX-Lite/

SPDX-Lite に求められる項目は以下に示すとおりである。

表 7-2 SPDX-Lite の項目と SPDX との対応関係

SPDX におけるセクション名	SPDX-Lite の項目名
SBOM ドキュメントの情報（Creation Information）	SPDX バージョン（SPDX Version）
	SBOM のデータライセンス（Data License）
	ドキュメント ID（SPDX Identifier）
	ドキュメント名（Document Name）
	ドキュメント名前空間（SPDX Document Namespace）
	ドキュメント作成者・作成ツール（Creator）
	ドキュメント作成日（Created）
パッケージ情報（Package Information）	パッケージ名称（Package Name）
	パッケージ ID（Package SPDX Identifier）
	パッケージバージョン（Package Version）
	パッケージファイル名（Package File Name）
	パッケージサプライヤー名（Package Supplier）
	パッケージのダウンロード場所（Package Download Location）
	パッケージファイル解析情報（Files Analyzed）
	パッケージのホームページ（Package Home Page）
	パッケージへ締結されたライセンス（Concluded License）
	パッケージ作成者が宣言する適用ライセンス（Declared License）
	ライセンスに関するコメント（Comments on License）
	パッケージの著作権（Copyright Text）
	パッケージに関するコメント（Package Comment）
	パッケージに関する外部参照情報（External Reference field）
その他ライセンス情報（Other Licensing Information）	ライセンス ID（License Identifier）
	ライセンス情報（Extracted Text）
	ライセンス名（License Name）
	ライセンスに関するコメント（License Comment）

2) 使用例・使用目的

SPDX-Lite の使用例・使用目的として、以下が想定される。

- 「SBOM ドキュメントの情報」と「パッケージ情報」の SPDX セクションに分類される必須項目のみを手動で管理する
- SPDX のレベルではなく、自動車業界や家電業界等で最低限要求された項目に対応し、運用

性を重視した SBOM を作成する

3) データフォーマットの特徴

SPDX-Lite の特徴として、以下が挙げられる。

- SPDX と比べて必要最低限の項目のみが含まれているため、運用性を重視した SBOM 管理が可能
- SPDX の「SBOM ドキュメントの情報」と「パッケージ情報」のセクションに分類される必須項目を含んでいるため、SPDX に対応した SBOM ツールと親和性が高い
- SPDX-Lite 形式の SBOM 作成に当たって、専用のツールは必要なく、手動で SBOM ドキュメントを作成することが可能

(3) CycloneDX

CycloneDX は、完全自動化可能で、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て、OWASP コミュニティのプロジェクトによって開発された。CycloneDX の詳細な仕様は、WEB サイト上に公開[30]されており、OWASP コミュニティのコアワーキンググループで管理、更新され続けている。以降では、CycloneDX の v1.4 の概要として、フォーマットの構成、フォーマットの使用例・使用目的、フォーマットの特徴を示す。

1) フォーマットの構成

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。SBOM ドキュメントに含める内容として、オブジェクトモデルと各オブジェクトモデルに分類される項目が規定されている。各オブジェクトモデルの概要を以下に示す。また、各オブジェクトモデルに規定された項目のうち、該当モデルを使用する場合に必須で含めるべき項目も決められている。なお、オブジェクトモデルには分類されないものの、SBOM ドキュメントが CycloneDX フォーマットであること、CycloneDX のバージョン、SBOM ドキュメントのバージョンの項目を含めることが、必須とされている。

- **SBOM のメタデータ（SBOM Metadata）：**
  サプライヤー、開発者、SBOM ドキュメントが対象とするソフトウェアの範囲、SBOM ドキュメントを作成するために使用したツール等の情報を提示するオブジェクトモデル。
- **コンポーネント（Components）：**
  ファーストパーティ及びサードパーティのソフトウェアコンポーネントのインベントリーを提示するオブジェクトモデル。本オブジェクトモデルには、ソフトウェアコンポーネントの種類、ID、ライセンス、著作

[30] https://cyclonedx.org/docs/1.4/json/

権、暗号的ハッシュ関数、系譜、来歴、加えられた変更内容等の情報を含めることが可能である。また、本オブジェクトモデルでは、コンポーネントの組合せを表現することができ、組合せられたコンポーネントは、一つのコンポーネントとして各種情報を保有することが可能である。さらに、コンポーネントや組合せられたコンポーネントにデジタル署名を適用することが可能である。

- **外部サービス（Services）：**
  SBOM ドキュメントが対象とするソフトウェアが呼び出す可能性のある外部 API の情報を提示するオブジェクトモデル。本オブジェクトモデルには、外部 API のエンドポイント URI、認証要件、外部 API との信頼境界線、サービス間とのデータの流れ・分類等の情報を含めることが可能である。さらに、サービスにデジタル署名を適用することが可能である。

- **依存関係（Dependencies）：**
  コンポーネントと他のコンポーネント間の依存関係を提示するオブジェクトモデル。コンポーネント同士だけでなく、サービスに依存するコンポーネントやサービスに依存するサービスも表現することが可能である。また、依存関係は、推移的依存関係を表現することが可能である。

- **構成要素の完全性（Compositions）：**
  SBOM内における各構成要素（コンポーネント、サービス、依存関係を含む）と構成要素の完全性を提示するオブジェクトモデル。各構成要素は、「完全である」、「不完全である」、「不完全なファーストパーティのみである」、「不完全なサードパーティのみである」、「不明である」のように表現することが可能である。本オブジェクトモデルによって、作成された SBOM がどの程度完全であるか、SBOM 内に完全性が不明な構成要素があるかを理解することが可能である。

- **脆弱性（Vulnerabilities）：**
  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提示するオブジェクトモデル。また、コンポーネントやサービスに影響を与える未知の脆弱性も提示することを可能とし、VEX 等のセキュリティアドバイザリーとして使用することが可能である。

- **拡張機能（Extensions）：**
  CycloneDX における新しい機能の試行、特殊なユースケースや将来のユースケースのサポートを可能にするオブジェクトモデル。CycloneDX プロジェクトでは、専門的や業界固有のユースケースを対象とする拡張機能に関するコミュニティへの参加や開発を推奨している。

2）使用例・使用目的

CycloneDX の使用例・使用目的として、以下が想定される

- システムの構成要素と構成要素間の関係性を記述する
- ソフトウェアコンポーネントの知的財産（ライセンス、著作権）を管理する
- ソフトウェアサプライチェーンのリスクアセスメントとコンポーネントを検証する

- ソフトウェアコンポーネント、コンテナコンテンツ等のインベントリーを作成する
- ソースファイルやソーススニペットに遡って実行ファイルを追跡する
- ファイルに埋め込まれたコードの出所を特定する
- ソフトウェアを一意に特定するためのフォーマットである CPE、SWID、パッケージ URL 等を特定のパッケージに関連付け、追加のセキュリティ分析を容易にする
- 署名されたコンポーネントや組合せられたコンポーネントと SBOM の整合性を検証する
- ソフトウェアのビルド時における作成・配布に便利なフォーマット、M2M（マシンツーマシン）でのバイナリフォーマットとして利用する

3) データフォーマットの特徴

CycloneDX の特徴として、以下が挙げられる。

- セキュリティ管理を念頭に置いた SBOM フォーマットであり、既知の脆弱性に関する情報や、その脆弱性の悪用可能性に関する情報を記載できる
- アプリケーション、コンポーネント、サービス、ファームウェア、デバイス等の様々な種類のソフトウェアに対応したセキュリティに関する SBOM フォーマットであり、幅広い業界で使用され商用利用に適している
- CycloneDX は、体系立てられたオブジェクトモデルで構成されるフォーマットであるため、学習と導入が容易
- 多くの開発エコシステムと統合することによって自動化を実現
- 仕様の拡張が可能なため、組織や業界特有の要件に応じた新しい機能の迅速な試行が可能

(4) SWID 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 タグでは、ソフトウェアライフサイクルに沿ったソフトウェアのインストールプロセスの一貫として、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。以降では、SWID タグの概要として、フォーマットの構成、フォーマットの使用例・使用目的、フォーマットの特徴を示す。

1) フォーマットの構成

SWID タグフォーマットによる SBOM では、SWID タグにしたがって作成された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パッチ等の情報が記載され、xml 形式がサポートされている。SWID タグは、対象とするデバイスのライフサイクルを把握するために、デバイスにインストールされたソフト

ウェアの情報を示すタグが規定されている。各タグの概要を以下に示す。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。

- **プライマリータグ（Primary Tag）：**
  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識別、提示するタグ。
- **パッチタグ（Patch Tag）：**
  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に対して、アップデート等で適用したパッチを識別、提示するタグ。
- **コーパスタグ（Corpus Tag）：**
  対象とするデバイスにインストールするソフトウェアを特定し、説明するタグ。本タグは、ソフトウェアのインストールパッケージ、インストーラー、ソフトウェアアップデート、パッチ等のソフトウェアのメタデータを表現するために使用される。
- **サプリメントタグ（Supplemental Tag）：**
  上記のタグに関する情報に、追加的な情報を付与するためのタグ。デバイスの利用者やソフトウェア管理ツールが、任意の情報を追加するために本タグが使用される。

2) 使用例・使用目的

SWID タグの使用例・使用目的として、以下が想定される。

- 組織で管理しているデバイスにインストールされたソフトウェアをコンポーネントとして、SBOM を作成する
- デバイスにインストールされたソフトウェアを継続的に追跡する
- エンドポイントにおける脆弱なソフトウェアを特定する
- デバイスにインストールされたソフトウェアが、適切にパッチを適用しているかを確認する
- 不正なソフトウェアや破損したソフトウェアのインストールを防止する
- 破損したソフトウェアの実行を防止する
- 管理対象のデバイスに関するユーザーの権利やアクセス権等を管理する

3) データフォーマットの特徴

SWID タグの特徴として、以下が挙げられる。

- ソフトウェアライフサイクルに合わせて各タグの情報を更新するため、ビルド時に作成されるソフトウェア ID の情報を正確にタグへ付与して提供することが可能
- ソフトウェアのインストール時に、サプライヤーと利用者の間で交換可能なソフトウェア情報を標準

化

- 関連するパッチやアップデート、構成設定、セキュリティポリシー、脆弱性や脅威の勧告等、ソフトウェアに関連する情報の関連付けが可能