SCA ツールの精度評価に向けた 標準データセットと評価手法の検討

山本 遊大[1,a)]　木原 百々香[1,b)]　原 悟史[2,c)]　佐々木 貴之[3,d)]　吉岡 克成[3,4,e)]

概要：近年，ソフトウェア開発において Open Source Software（OSS）の利用が一般的になっているが，ライセンス違反や脆弱性などのリスクが存在する．これらのリスクを検出するツールとして，Software Composition Analysis（SCA）ツールが注目されているが，検出精度を客観的に評価するための標準的な手法やデータセットは確立されておらず，適切なツール選択や性能比較が困難である．そこで本研究では，SCA ツール評価用データセットとそれに基づく評価手法の確立に向けた第一歩として，ルータファームウェアで使用される代表的な OSS を対象とした作成手法の異なる 2 種類の評価用バイナリデータセットを試作し，商用ツール 1 つとオープンソースツール Syft の精度評価を実施した．その結果，今回の実験においては，データセットに含まれる OSS の選定基準がツールの検出精度に最も大きな影響を与えること，同一 OSS においてはバイナリデータセット作成手法による検出結果への影響は限定的であることを確認した．最後に，SCA ツール評価用データセットの作成と利用におけるエコシステムについて検討し，公平性の観点から SCA ツールベンダーと適切な距離をとりながら，現場のユースケースを反映してデータセットを構築することが必要であることを考察した．

キーワード：SCA ツール，データセット

Toward Accuracy Evaluation of SCA Tools: Standard Datasets and Evaluation Methodology

Yudai Yamamoto[1,a)]　Momoka Kihara[1,b)]　Satoshi Hara[2,c)]　Takayuki Sasaki[3,d)]
Katsunari Yoshioka[3,4,e)]

Abstract: In recent years, the use of open source software (OSS) has become commonplace in software development, but there are risks such as license violations and vulnerabilities. Software composition analysis (SCA) tools are attracting attention as a means of detecting these risks, but there are no established standard methods or datasets for objectively evaluating detection accuracy, making it difficult to select appropriate tools and compare their performance. In this study, as a first step toward establishing an evaluation dataset and evaluation methodology for SCA tools, we identified issues to be considered based on use cases and created two types of evaluation datasets of representative OSS used in router firmware, created with different methods. Furthermore, we evaluated the accuracy of one commercial tool and the open-source tool Syft using datasets. In this experiment, we confirmed that the selection criteria for OSS included in datasets have the greatest impact on tool detection accuracy, and that the influence of binary dataset creation methods on detection results was limited when using identical OSS. Finally, we examined the ecosystem for creating and using SCA tool evaluation datasets and concluded that it is necessary to maintain an appropriate distance from SCA tool vendors from the perspective of fairness while updating datasets to reflect real-world use cases.

Keywords: SCA tool, dataset

1. はじめに

近年，ソフトウェア開発において Open Source Software (OSS) の利用が一般的となっている．Black Duck 社の 2025 年の調査では，商用のコードベース 1658 件のうち 97%で OSS を利用しており，さらに 1 つのコードベースに平均して 911 個の OSS コンポーネントが含まれていることが判明している [1]．OSS は無償で利用でき開発コストの削減に役立つ一方で，ライセンス違反による訴訟のリスクや脆弱性によるリスクが存在すため，ソフトウェア内に含まれる OSS を把握し適切な管理を行うことが重要視されている．

このような OSS リスクに対処するツールとして近年注目されているのが，Software Composition Analysis (SCA) ツールである．SCA ツールはソフトウェア中の OSS コンポーネントを検出し，検出されたコンポーネントに報告されている既知の脆弱性を合わせて出力する．これを利用することで，サードパーティ製ソフトウェアのリスク検出や，近年導入が進んでいる Software Bill of Materials (SBOM) の自動作成を行うことができる．

しかし，同一の Web アプリケーションを複数の無償または有償 SCA ツールで解析した結果を比較した研究 [2] では，ツールごとに OSS コンポーネントの検出数や脆弱性検出数に大きな差があることが明らかとなっている．OSS コンポーネントの検出は SCA ツールを利用する主目的であることから，ユーザがツールを選択する際の指標の一つになると考えられる．検出精度を評価するためには，既知の OSS コンポーネントとそのバージョン，それらに紐づく脆弱性やライセンスといった属性情報に関する正解データ，すなわち評価用データセットの構築が必要となる．SCA ツールの OSS 検出ロジックはブラックボックスであるため，検出ロジックに基づいてテータセットを構築することは難しく，ユースケースごとにデータセットを用意する必要があると考えられる．しかしながら，既存の研究ではツールの OSS コンポーネント検出精度を客観的に評価するためのデータセットや，データセットを用いた評価手法は確立されておらず，ユーザが解析したい対象に合わせて性能比較を行うことは困難である．

そこで本研究では，SCA ツール評価用データセットとそれに基づく評価手法の確立に向けた第一歩として，SCA ツールのユースケースを想定して検討事項を洗い出し，それを踏まえて 2 種類の評価用データセットを試作した．

データセット試作を通じて，ユースケース基づく様々な検討事項があることが分かった．少なくとも以下の項目を検討する必要があった．

ソースコード/バイナリ．ソフトウェア開発では，開発中の製品のソースコードに対して SCA ツールを適用する一方で，サードパーティから入手したコンポーネントを分析する際はバイナリファイルに対して SCA ツールを適用する場合があり，それらのユースケース対応してソースコードやバイナリのデータセットが考えられる．

OSS の選定．SCA ツールで解析する対象に基づいて，データセットに含める OSS を選定する必要がある．特定分野 (例えば，通信機器や自動車分野など) の製品を対象に SCA ツールを使用する場合には，当該分野で広く使用される OSS を正確に分析できることが望まれると思われる．さらに，古い機器を対象とするか，新しい機器を対象とするかによってもデータセットに含めるべき OSS が異なる．

バイナリの作成手法．バイナリのデータセットを構築する場合，その作成方法を検討する必要がある．バイナリ作成手法として，OSS のソースコードを公式サイトから取得しコンパイルしてバイナリを生成する方法が考えられる．その際は，さらに，コンパイル環境 (CPU アーキテクチャ，コンパイラのバージョン，最適化オプションなど) や，バイナリ中の文字列削除 (strip) の有無を検討する必要がある．また，ソースコードをコンパイルするのではなく，実際の機器のファームウェアに含まれているバイナリを抽出することで，バイナリを用意する手法が考えられる．

これらの項目は 2 種類のデータセットを試作する過程で検討したものであり，より多様なユースケースを想定した場合にはさらなる考慮が必要になると考えられる．従って，今後，IoT 機器メーカや脆弱性診断を第 3 者の立場で行っている組織などの SCA ツールのユーザにインタビューを行い，ユースケースの収集と，どのユースケースやそれに基づくデータセットの作成を優先すべきかを検討したい．

本研究では，上記の項目を踏まえ，第三者の評価機関がルータの脆弱性診断を行うユースケースを想定して，ルータに利用される OSS を対象としたバイナリデータセットの試作を行った．バイナリの生成/収集は，実際のファームウェアに含まれる OSS のソースコードを公式サイトから取得し仮想環境でコンパイルするソースコードコンパイル手法と，ファームウェアから抽出したバイナリを複数の SCA ツールで解析し，全ツールで一致したコンポーネント名・バージョンを抽出するコンセンサス手法を用いた．

[1] 横浜国立大学大学院環境情報学府
Graduate School of Environment and Information Sciences, Yokohama National University
[2] 富士ソフト株式会社
FUJI SOFT INCORPORATED
[3] 横浜国立大学先端科学高等研究院
Institute of Advanced Sciences, Yokohama National University
[4] 横浜国立大学大学院環境情報研究院
Graduate School of Environment and Information Sciences, Yokohama National University
[a)] yamamoto-yudai-jx@ynu.jp
[b)] kihara-momoka-dw@ynu.jp
[c)] hara-satoshi-bs@ynu.ac.jp
[d)] sasaki-takayuki-yv@ynu.ac.jp
[e)] yoshioka@ynu.ac.jp

データセットの特性が SCA ツールの検出精度に与える影響を分析するために，これらの 2 つのデータセットに加え，既存研究 [3] で使用されていた実行検証手法 (ファームウェアから実行ファイルを取り出し仮想環境上で実行して標準出力からコンポーネント名やバージョンなどの情報を取得する手法) で作成したデータセットを，商用ツール 1 つとオープンソースの SCA ツールである Syft[4] にそれぞれ適用した．その結果，商用ツールにおいて，ソースコードコンパイル手法データセットでは 50.0%, 実行検証手法データセットでは 36.7%の検出率を示した．Syft の精度評価において，ソースコードコンパイル手法データセットでは 8.3%，コンセンサス手法データセットでは 25.8%，実行検証手法データセットでは 11.4%の検出率を示した．以上のように，商用 SCA ツールと Syft のどちらの場合でも，データセット構築手法によって検出精度が変動することがわかった．また，SCA ツールに検出された OSS とされなかった OSS を比較したところ，SCA ツールに検出された OSS はそうではない OSS に比べて，幅広い製品に用いられている OSS (例: OpenSSL など) であった．

最後に，SCA ツール評価用データセットの作成と利用におけるエコシステムについて考察する．例えば，アンチウイルスソフトの評価では，評価機関はアンチウイルスソフトベンダーと独立して運用されている [5]．SCA ツール評価用データセットでも同様に，ツールベンダーなどの評価対象組織との情報交換において適切な距離を保つことが重要である．具体的には，作成したデータセットをツールベンダーに提供することや，ツールベンダーが内部で使用している評価データセットを研究用データセットに組み込むことは，評価の客観性を損なう可能性がある．また，作成したデータセットは学術研究コミュニティでの共有を基本とし，特定のツールベンダーの競争優位性に影響を与えないよう配慮する必要がある．これらの配慮により，公正で透明性の高い SCA ツール評価環境の構築に貢献することが期待される．適切な距離を取る一方で，現場のユースケースの情報を収集し，データセットをアップデートしていく必要がある．そのために，データセット構築における検討項目については公開し，それに対して適宜フィードバックを得る施策などが考えられる．今後はユーザーやベンダーにインタビューをすることにより，SCA ツール評価用データセットのエコシステムについて検討する．

本論文の貢献は以下である．

- SCA ツールの評価データセットの試作を通じて，データセット作成の際に検討すべき項目は SCA ツールのユースケースに依存して数多くあることを議論した．
- SCA ツールで複数のデータセットを解析し，今回の実験において，OSS の選定基準がツールの検出精度に最も大きな影響を与えていたこと，及び，同一 OSS においてはソースコードをコンパイルする手法とファームウェアから抽出する手法の間で検出結果の一致率が高いことを確認した．
- SCA ツール評価用データセットの作成と利用におけるエコシステムについて検討を行い，評価対象組織と適切な距離をとりながら，現場のユースケースを反映してデータセットをアップデートすることが必要であることを考察した．

2. 背景

2.1 Software Bill of Materials(SBOM)

近年，OSS 由来の脆弱性の把握やライセンス違反リスク低減のため，Software Bill of Materials(SBOM) の導入が進められている．SBOM とは，ソフトウェアに含まれる OSS コンポーネントの一覧で，コンポーネントの名前やバージョン，サプライヤーなどの詳細情報が含まれる．SBOM には複数のフォーマットがあり，代表的なものに SPDX と CycloneDX がある．SBOM は 2021 年の米国大統領令 [6] や，2024 年に EU により発令されたサイバーレジリエンス法 [7] において要件として含まれており，各国で導入の動きが活発となっている．日本では 2023 年に経済産業省により「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き ver1.0」[8] が公表され，医療機器や政府調達機器に関して導入が活発に進められている．

2.2 Software Composition Analysis (SCA) ツール

SCA ツールは，ソフトウェア中の OSS コンポーネントを検出し，SBOM やコンポーネントのライセンス情報，脆弱性情報を合わせて出力するツールである．ツールの解析対象としては，バイナリファイルだけでなくソースコードも解析できるツールもある．SCA ツールの OSS 検出メソッドは基本的にブラックボックスで不明瞭であるが，バイナリ解析の際は OSS を特定するためのフィンガープリントを用いて検出を行っているという調査結果もある [9]．

2.3 SCA ツールのユースケース

商用・フリーともに多くの SCA ツールが存在するが．解析対象（バイナリ・ソースコード）や出力する SBOM のフォーマット，レポートの形式，クラウド・オンプレミス，REST API の有無など，機能はツールによって異なる．これらの選択は，ソフトウェアサプライチェーンにおけるユースケースと関連している．

例えば，機器メーカが自社製品を開発する際にはソースコードへのアクセスが可能であるため，ソースコード解析が可能な SCA ツールが適している．この段階では，開発プロセスに OSS チェックを組み込むことが主要なユースケースとなる．一方，機器メーカがチップメーカからバイナリ形式で SDK を受け取り，その SDK に含まれる OSS

を解析する場合や，第三者機関が完成した製品の評価を行う場合には，ソースコードへアクセスができないため，バイナリ解析が可能な SCA ツールが必要となる．この段階では，サードパーティ製コンポーネントのリスク評価，既存システムの脆弱性診断などが主要なユースケースとなる．

3. 関連研究

SCA ツールの OSS コンポーネント検出に関する調査はこれまでにも複数行われている．同一の Web アプリケーションを 9 つの SCA ツールで解析し，出力結果を比較した研究 [2] では，ツールによって出力された OSS コンポーネントや脆弱性が異なることが明らかとなっている．また，4 つのプログラミング言語のライブラリを複数の無償 SCA ツールで解析し，出力された SBOM を調査した研究 [10] では，ツールにより依存関係や OSS サプライヤ等，出力される情報に差異があることが判明している．他にも，4 つの SCA ツールにおいて C・C++ライブラリの検出能力を調査した研究 [11] では，汎用ライブラリの検出精度が 70%以上を示す一方で，自動車ファームウェアに採用されているライブラリの精度は 45%以下に減少することが明らかとなった．さらに，SCA ツールのユーザである開発者 20 名にインタビューを行った研究 [9]，SBOM の採用状況についてステークホルダーにインタビューを行った研究 [12] では，既存の SCA ツールはバイナリ解析精度や完全性に課題があり，出力のレビューに時間コストがかかるということが判明している．

これらの研究では，SCA ツールの OSS コンポーネント検出精度に関する問題提起が行われているが，ツールのユースケースに着目し，ユーザが解析したい対象に合わせて性能比較を行うための評価用データセットの構築は行われていない．そのため本研究では，SCA ツールの評価用データセットの構築手法や，それを用いた評価の手法についての検討を行う．

4. SCA ツール評価用データセット構築における検討事項

SCA ツール評価用データセットの構築には，ツールの利用目的や評価対象となるソフトウェアの特性に応じて多岐にわたる検討事項が存在する．本研究では，2 種類のデータセットを作成する過程において，データセット構築時に考慮すべき主要な検討軸として，ソースコード/バイナリの選択，対象 OSS の選定，およびバイナリの作成手法の 3 つが少なくとも存在することを特定した．

4.1 ソースコード/バイナリ

SCA ツール評価用データセットの構築において検討すべき項目の一つ目は，ソースコードとバイナリのいずれを対象とするかである．この選択は，SCA ツールの利用ユースケースと密接に関連している．

ソースコードを対象とするデータセットは，主に開発段階での SCA ツール利用を想定したユースケースに適している．開発チームが自社製品に SCA ツールを適用する場合，ソースコードへのアクセスが可能なため，開発プロセスに統合された継続的な脆弱性監視や，CI/CD パイプラインでの自動チェックなどが主要な利用目的となる．

一方，バイナリを対象とするデータセットは，より幅広いユースケースに対応可能である．第三者機関による脆弱性診断では，評価対象ソフトウェアのソースコードが入手困難な場合が多く，バイナリ解析による OSS 検出が必要となる．また，サードパーティ製ソフトウェアの導入前評価，既存システムの現状把握など，多様な場面でバイナリ解析が必要となる．

4.2 対象 OSS の選定

評価用データセットに含める OSS の選定は，SCA ツールの使用目的と対象とする分野に応じて行う必要がある．

特定分野の製品を対象に SCA ツールを使用する場合には，当該分野で広く使用される OSS を正確に分析できることが望まれると考えられる．例えば，通信機器を対象とする SCA ツール評価では，ネットワーク関連ライブラリ（OpenSSL、libpcap、iptables 等）や組み込み Linux 環境で使用される基本ツール群（BusyBox、uClibc 等）を含める必要がある．自動車分野であれば，車載システム特有のリアルタイム処理ライブラリや通信プロトコル実装が重要となる．このような分野固有の選定により，実際の利用場面により適した評価が可能となる．

また，古い機器を対象とするか，新しい機器を対象とするかによってもデータセットに含めるべき OSS が異なる．古い機器では，メンテナンスが停止している古いバージョンの OSS が使用されている場合が多く，これらのレガシーコンポーネントに対する SCA ツールの検出能力を評価することが重要である．一方，新しい機器では最新バージョンの OSS が使用されることもあるため，最新のコンポーネントに対する検出能力の評価をすることも重要である．

セキュリティ評価に特化したデータセットでは，重大な脆弱性が報告された OSS や，CVE 登録数の多いライブラリを積極的に含めることが有効と考えられる．これにより，SCA ツールの脆弱性検出能力を重点的に評価できる．

4.3 バイナリデータセットの作成手法

バイナリのデータセットを構築する場合，その作成方法を検討する必要がある．本研究では主に 2 つのアプローチを検討した．

4.3.1 ソースコードコンパイル手法

1 つ目のアプローチは，OSS のソースコードを公式サイトから取得し，コンパイルすることでバイナリを生成する

手法である．この手法の利点は，コンポーネント名とバージョンが確実に既知であるバイナリをデータセットに組み込むことができる点である．また，コンパイルオプションやビルド設定を変更することで多様なバイナリパターンを網羅的に収集できるため，異なるビルド環境での検出精度評価や，特定のコンパイル条件下での性能測定といったユースケースに適している．この手法を採用する際は，さらに，コンパイル環境（CPU アーキテクチャ，コンパイラのバージョン，最適化オプションなど）や，バイナリの中の文字列が削除（strip）するか否かを検討する必要がある．

4.3.2 ファームウェア抽出手法

2 つ目のアプローチは実際の機器のファームウェアに含まれているバイナリを抽出する手法である．この手法の利点として，実際のファームウェアに組み込まれたバイナリをそのままデータセットにしているため，実運用環境での検出性能をより正確に評価できることが挙げられる．また，ソースコード入手やコンパイルが困難なコンポーネントもデータセットに組み込むことができるため，より包括的なデータセット構築が可能となる．この手法は，ソースコードの一部変更やパッチの適用などが行われている，実際の製品のバイナリの検出精度を評価したいといったユースケースに適している．

5. 評価用データセットの作成手法

第 4 章で議論した検討事項を踏まえ，本研究では実際に 2 種類の評価用データセットを作成した．本研究では，第三者の評価機関がルータの脆弱性診断を行うユースケースを想定し，ルータファームウェアで使用される代表的な OSS を対象としたバイナリデータセットを作成した．データセット作成においては，異なるアプローチによる特性の違いを検証するため，ソースコードコンパイル手法とコンセンサス手法の 2 つの手法を採用した．コンセンサス手法はファームウェア抽出手法の一種である．また，比較対象として既存研究 [3] で使用された実行検証手法によるデータセットも活用した．実行検証手法もファームウェア抽出手法の一種である．

5.1 ソースコードコンパイル手法

作成プロセス：実際のファームウェアに含まれている OSS のソースコードを公式サイトからダウンロードし，環境でコンパイルすることによりバイナリデータセットを構築した．コンパイル環境として，ホストシステムに Ubuntu22.04（AMD64）を使用し，ターゲットアーキテクチャは AArch64（ARM64）を採用した．コンパイラには aarch64-linux-gnu-gcc を使用し，ビルド環境を構築した．

対象 OSS の選定：ルータファームウェアで使用頻度の高い OSS を特定するため，我々の先行研究 [13] で行ったルータファームウェア 126 機器の解析結果を元にランキングを作成した．この先行研究では，商用ツールを用いて各ファームウェアに含まれる OSS を特定し，検出頻度に基づいて使用頻度ランキングを作成している．ランキング作成に使用した SCA ツールは本研究における評価対象のツールと異なるため，評価対象ツールに有利になるようなことは避けている．ただし，ランキング作成自体が SCA ツールの検出結果に依存しているため，SCA ツールが検出しやすい OSS に偏ってしまう可能性がある．次に，実際のファームウェアに含まれていることがわかった上位 51 個の OSS を選定した．

データセット：最終的に構築されたデータセットは，OSS51 種類，総データセット 60 件（複数バージョンを含む），個別バイナリファイル数 693 個から構成される．

本手法の特徴として，4.3 節で述べたように，コンポーネント名とバージョンが確実に既知であるバイナリをデータセットに組み込むことができる点，および，コンパイルオプションやビルド設定を変更することで多様なバイナリパターンを網羅的に収集できる点が挙げられる．一方，実際のファームウェアに組み込まれているようなバイナリとは異なるものを生成してしまうことも考慮する必要がある．

5.2 コンセンサス手法

作成プロセス：同一ファームウェアを対象に複数の SCA ツールで解析を実施し、全ツールで一致したコンポーネント名・バージョンの組み合わせを抽出することによりバイナリデータセットを構築した．解析対象としたファームウェアは 10 種類のルータのファームウェであり，国内市場における主要なルータメーカ 6 社から選定している．SCA ツールは 3 種類の商用ツールを使用した．このデータセットを使用する際の注意点として，評価用のデータセットを評価対象のツール群の検知結果を参照して作成するのは適切でないため，データセット作成に使用する SCA ツール群と評価対象の SCA ツール群は分離する必要がある．

評価用データセットとしての意義：本手法は (1) ファームウェア抽出手法の利点と (2) コンセンサス取得による利点の 2 つの価値を提供できる．

(1) ファームウェア抽出手法の利点として，4.3 節で述べたように，実運用環境での検出性能をより正確に評価できる点や，ソースコード入手やコンパイルが困難なコンポーネントもデータセットに組み込むことができる点がある．

(2) コンセンサス取得による利点として，まず，多くの SCA ツールが解析可能な OSS をデータセットに組み込むことができる点が挙げられる．複数ツールで検出される OSS は，他のツールでも検出されることが期待され，ツールごとの検出制度の比較において，適切な評価対象となる．また，複数ツールで検出結果が一致していることにより，正解データとしての信頼性が向上される点が挙げられる．

本手法特有のデメリットとして，SCA ツールの解析結

果に依存するため正解データの信頼性は保証されない点，データセット作成に使用した SCA ツールの精度評価が実施できなくなる点が存在する．

データセット：最終的に構築されたデータセットは，OSS11 種類，総データセット 31 件（複数バージョンを含む），個別バイナリファイル数 40 個から構成される．

5.3 実行検証手法

本研究では，作成した 2 つのデータセットに加え，データセットごとの特性を相対的に評価するため，既存研究 [3] で使用されていた手法によるデータセットを比較対象として使用した．

作成プロセス：ファームウェア内の ELF ファイルを QEMU 上で実行し，得られた標準出力中からコンポーネント名とバージョンを取得することでデータセットを構築している．QEMU は ARM や MIPS，Power PC のような IoT 機器でよく使用される CPU アーキテクチャの仮想環境を提供している．既存研究では QEMU をユーザーモードエミュレーションと呼ばれる，実行対象バイナリをプロセスとして起動し実 PC 上で実行する機能を利用してデータセットの構築を行っている．

対象 OSS の選定：IoT 機器 15 機種のファームウェアを対象としている．具体的には，スイッチ 2 機種 2 メーカ，ルータ 8 機種 8 メーカ，NVR2 機種 2 メーカ，IP カメラ 2 機種 2 メーカ，ワイヤレスアダプタ 1 機種 1 メーカから構成されており，ルータに限定されず幅広い IoT 機器をカバーしている．

データセット仕様：最終的に構築されたデータセットは，OSS75 種類，総データセット 128 件（複数バージョンを含む），個別バイナリファイル数 153 個から構成される．

6. 評価用データセットを用いた SCA ツールの評価

6.1 実験設定

評価対象ツール：本研究では，商用 SCA ツール 1 つとオープンソースツールである Syft（バージョン 1.29.0）を評価対象とした．商用ツールについては匿名化して表記する．

使用データセット：評価には第 4 章で説明した 3 つのデータセット（ソースコードコンパイル手法、コンセンサス手法、実行検証手法）を使用した．ただし，商用ツールの精度評価の際には，コンセンサス手法で作成したデータセットは使用していない．これは，当該データセットの作成過程で商用ツールを使用しているためである．

評価指標：SCA ツールで各データセットを解析した際に，データセットで定義されているコンポーネント名，バージョンの両方と一致していた割合をそのツールの検出精度とした．

表 1 データセット構築手法別の SCA ツール検出精度

SCA ツール	ソースコードコンパイル手法	コンセンサス手法	実行検証手法
商用ツール	50.0%	対象外	36.7%
Syft	8.3%	25.8%	11.4%

6.2 評価結果

各 SCA ツールのデータセット別検出精度を表 1 に示す．表中の数値は，各データセットに含まれる OSS について，SCA ツールの出力結果がデータセットで定義されているコンポーネント名とバージョンの両方と一致していた割合を示している．商用ツールについては，コンセンサス手法のデータセット作成に使用したため，評価対象外とした．

6.3 ツール間比較

商用ツールと Syft の検出精度を比較すると，全てのデータセットにおいて商用ツールが Syft を上回る結果となった．具体的には，ソースコードコンパイル手法では商用ツールが 50.0%に対して Syft が 8.3%，実行検証手法では商用ツールが 36.7%に対して Syft が 11.4%となった．

商用ツールは，Syft と比較してより幅広い種類の OSS を検出する傾向が確認された．両ツールともに多くのファームウェアで使用される汎用性の高いコンポーネントについては安定した検出性能を発揮したが，商用ツールは Syft では検出困難な比較的使用頻度の低いコンポーネントについても検出能力を示した．

両ツールの検出結果を分析したところ，共通して検出できた OSS として，BusyBox，curl，OpenSSL，util-linux が確認された．これらの OSS は組み込み Linux 環境において基本的な機能を提供する汎用性の高いコンポーネントである．

検出精度の詳細な分析では，両ツールともにコンポーネント名が検出できた場合には，バージョンまで正確に検出できていることが確認できた．ただし，商用ツールでは 1 つの OSS についてコンポーネント名は正しく検出されたものの，バージョンが誤って識別される事例があった．また，両ツールともに誤検出（コンポーネント名やバージョンの誤った検出，存在しないコンポーネントの検出）は確認されず，各 OSS に対して，正確な検出か未検出のいずれかの結果を示した．

6.4 データセット間比較

データセット構築手法による検出精度の違いを分析すると，同一の SCA ツールであってもデータセットによって異なる検出結果が得られることがわかった．

商用ツールにおいて，ソースコードコンパイル手法では 50.0%の検出率を示したのに対し，実行検証手法では 36.7%を示すことが確認された．また，Syft においても同様の傾

向が確認され，ソースコードコンパイル手法では 8.3%の検出率を示したのに対し，コンセンサス手法では 25.8%と最も高く，実行検証手法では 11.4%となった．

各データセットに含まれる OSS の特徴には大きな違いがある．コンセンサス手法で作成したデータセットでは，BusyBox や curl などの使用頻度の高い OSS が多く含まれていた．これは，複数の SCA ツールで検出される OSS が，広く利用されている汎用的な OSS に偏ってしまうためである．ソースコードコンパイル手法で作成したデータセットでは，ルータ使用頻度ランキングを参照して作成したため，ルータ分野で重要度の高い OSS が多く含まれていた．一方，実行検証手法で作成したデータセットにはマイナーな OSS が最も多く含まれており，全部で 198 ファイル中，ルータ使用頻度ランキング 500 位圏外の OSS が 46 ファイル含まれていた．

この対象 OSS の違いが検出精度に与える影響は顕著であった．商用ツールの場合でも，ランキング 500 位圏外のマイナー OSS 47 件のデータセットのうち 43 件のデータセット（91.5%）が検出できていなかった．一方で，実行検証手法のデータセットをルータ使用頻度ランキング 100 位以内の OSS に絞って分析したところ，65 件のデータセットのうち 39 件（60.0%）が検出できており，使用頻度の高い OSS では検出率が大幅に向上することが確認された．

また，同一の OSS が複数のデータセットに含まれた場合，バイナリ作成方法が異なっていても検出結果は高い一致率を示した．ソースコードコンパイル手法と実行検証手法のデータセットを比較したところ，16 件の同一 OSS（バージョンが異なっていても同一 OSS として扱う）が含まれており，そのうち 15 件（93.8%）で評価対象の商用ツールの検出結果が一致していた．この結果から，現時点での評価においては，バイナリ作成手法よりもデータセットに含める OSS の選定基準の方が精度評価により大きな影響を与える可能性がある．ただし，これは限られたデータセットでの結果であるため，今後データセットを拡充し，検証を行いたい．

7. 考察

7.1 データセット多様化の必要性と課題

SCA ツールの客観的で包括的な評価を実現するためには，多様な構築手法に基づく複数のデータセットが不可欠である．これは，SCA ツールが様々な利用環境や目的で使用されるため，単一のデータセットでは評価の網羅性や妥当性を確保することが困難であるためである．

第 4 章で提案したように，データセット構築には，対象とする分野，OSS 選定基準，バイナリ作成手法，バージョン選択など，多数の検討軸が存在する．これらの軸の組み合わせにより，異なる特性を持つデータセットを構築することができ，各データセットは特定のユースケースや評価目的に適応した特徴を有する．例えば，IoT 機器の評価を想定したデータセットと Web アプリケーションの評価を想定したデータセットでは，含まれる OSS の種類，対象アーキテクチャなどが異なってくる．

多様なデータセットを構築することで，SCA ツールの精度評価の結果が一致するかどうかの結果に関わらず，以下のような価値を提供することができる．ある SCA ツールの評価結果が複数のデータセット間で一致する場合，そのツールが多様な環境において安定した性能を発揮することを実証することができる．一方，データセットごとに検出精度が異なる場合，ツールの得意分野などを明確にすることができ，特定のユースケースにおける適性を判断することができるようになる．いずれの場合においても，単一のデータセットでは得られない価値を提供することができる．

しかし，多様なデータセットの構築は多くの困難を伴う．まず，第 4 章で提案した 3 つの検討軸（ソースコード/バイナリの選択，OSS 選定基準，バイナリ作成手法）に加えて，SCA ツールのユースケースに応じてさらに多様な検討軸が考えられる．これらの検討軸を包括的に収集・整理することは容易ではなく，IoT 機器メーカ，ソフトウェアベンダー，セキュリティ監査機関など，多様なステークホルダーに対するインタビューなどを通じて情報収集する必要がある．ステークホルダのニーズを適切に反映可能な検討軸を精査し，それぞれのニーズやリソースにあったデータセットを構築するための方法論を確立することが重要である．

7.2 SCA ツールの精度評価のあるべき姿

本研究の検討事項や結果を踏まえ，SCA ツールの精度評価における理想的な枠組みについて考察する．理想的な SCA ツールの評価環境は，多様なユースケースに対応した複数のデータセットが整備され，ツールの導入を検討する企業や組織がそれぞれの利用目的に応じて適切なデータセットを選択し，客観的な性能比較を実施できる仕組みであると考える．例えば，IoT 機器メーカであれば組み込みシステムに特化したデータセット，Web サービス企業であれば Web アプリケーション向けデータセットを選択することで，実際の利用環境に応じた評価が可能となる．

このような環境の実現には，データセットを分類して体系化することが重要である．7.1 節で提案したような多様な軸の組み合わせにより，異なる特性を持つデータセットを構築することが求められる．また，各データセットに，対象分野，想定ユースケースなどの情報を付与することで，適切なデータセットの選択を支援することも可能となる．

7.3 SCA 評価用データセットのエコシステム

公正な SCA ツール評価環境を実現するためには，適切なエコシステムの構築が必要である．このエコシステムで

は，評価の独立性と客観性を確保しつつ，実用的なフィードバックを取り入れる仕組みが重要となる．

アンチウイルスソフトウェアの評価では，AV-Comparatives[5] のような独立評価機関が確立されており，ベンダーから独立した立場で客観的な性能評価を実施している．SCA ツール評価においても同様に，ツールベンダーなどの評価対象組織との情報交換において適切な距離を保つことが重要である．具体的には，作成したデータセットをツールベンダーに事前提供することや，ベンダーが内部で使用している評価データセットを研究用データセットに組み込むことは，評価の客観性を損なう可能性があるため避ける必要がある．このようなことを行う場合，特定のベンダーに有利な評価結果をもたらす可能性があり，評価の公正性を損なってしまう恐れがある．また，作成したデータセットは独立評価機関での共有を基本とし，特定のツールベンダーの競争優位性に影響を与えないよう配慮する必要がある．

一方で，適切な距離を保ちながらも，現場のユースケースや技術動向を適切に反映するため，ユーザ企業などとの協力関係は重要である．実用的で価値のあるデータセットを構築するためには，実際の利用環境や要求事項を深く理解する必要があり，これはステークホルダーとの協力なしに行うことは困難である．

7.4 研究倫理

本研究における，SCA ツールの精度評価の実験では，研究倫理への配慮を十分に考慮して実施した．商用 SCA ツールの検出精度の結果については，事前にメーカから許可を得た上で公開している．また，検出精度の公開がメーカのブランドイメージや市場競争力に与えるネガティブなインパクトを防止するため，本論文では商用ツール名を匿名化して表記している．さらに評価結果の解釈においては，限定的なデータセットを用いた結果であることを明示し，特定のツールの総合的な優劣を評価することは避けている．

8. まとめと今後の課題

本研究では，SCA ツールの OSS コンポーネント検出能力を評価するための第一歩として，ソースコードコンパイル手法，コンセンサス手法によるデータセットを作成し，フリー・商用の SCA ツール 2 種類で評価を行った．本研究の検討事項や結果を踏まえ，SCA ツールの精度評価には多様なユースケースを考慮したデータセットを構築し，さらに公平性を確保した評価エコシステムが必要であると結論付けた．今後は，IoT 機器メーカや脆弱性診断を第三者の立場で行っている組織などの SCA ツールのユーザにインタビューを行い，ユースケースを収集する．その上でどのユースケースに優先して対応するべきか検討し，それに基づいてデータセットを作成する．

謝辞 本研究の一部は，JST【経済安全保障重要技術育成プログラム】【JPMJKP24K2】の支援を受けたものです．

参考文献

[1] BlackDuck Software Inc. 2025 Open Source Security and Risk Analysis Report(OSSRA 2025). https://www.blackduck.com/resources/analyst-reports/open-source-security-risk-analysis.html(accessed 2025-07-16).

[2] Nasif Imtiaz, Seaver Thorn, Laurie Williams. A comparative study of vulnerability reporting by software composition analysis tools. ESEM ’21, October 11–15, 2021.

[3] 森井裕大, 木原百々香, 佐々木貴之, 秋山満昭, 植田宏, 吉岡克成, 松本勉. IoT ファームウェアに含まれる OSS の脆弱性に関する SCA ツールを用いた調査. コンピュータセキュリティシンポジウム 2023 論文集, 2023.

[4] Syft. https://github.com/anchore/syft.

[5] Av-comparatives. https://www.av-comparatives.org/.

[6] Executive Office of the President. Executive office of the president “improving the nation’s cybersecurity”. https://www.federalregister.gov/documents/2021/05/17/2021-10460/improving-the-nations-cybersecurity (accessed 2025-07-16).

[7] European Union. Regulation of the European Parliament and of the Counsil on Horizonal Cybersecurity Requirements for Products with Digital Elements(Cyber Resilience ACT). https://www.cyberresilienceact.eu/the-cyber-resilience-act/#chapterVII (accessed 2025-08-04).

[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. 1.0 . https://www.meti.go.jp/press/2023/07/20230728004/20230728004-1-2.pdf (accessed 2025-07-16).

[9] Elizabeth Lin, Sparsha Gowda, William Enck, and Dominik Wermke. Context Matters: Qualitative Insights into Developers’ Approaches and Challenges with Software Composition Analysis. In *34th USENIX Security Symposium, August 13-15*, 2025.

[10] Andreas Halbritter and Dominik Merli. Accuracy evaluation of sbom tools for web applications and system-level software. In *Proceedings of the 19th International Conference on Availability, Reliability and Security*, ARES ’24, New York, NY, USA, 2024. Association for Computing Machinery.

[11] Yuqiao Ning, Yanan Zhang, Chao Ma, Zhen Guo, and Longhai Yu. Empirical study of software composition analysis tools for c/c++ binary programs. *IEEE Access*, Vol. 12, pp. 50418–50430, 2024.

[12] Berend Kloeg, Aaron Yi Ding, Sjoerd Pellegrom, Yury Zhauniarovich. Charting the Path to SBOM Adoption: A Business Stakeholder-Centric Approach. In *ASIA CCS’24, July 1-5*, 2024.

[13] 木原百々香, 佐々木貴之, 吉岡克成. ルータのファームウェアに含まれる OSS 脆弱性に関する SCA ツールを用いた調査. 情報通信システムセキュリティ研究会（ICSS）, 2024.