SBOMを用いた脆弱性検査における対応優先度付けを目的とした影響リスクの評価手法

荒川 玲佳[1,a)]　鐘本 楊[1]

概要：ソフトウェアサプライチェーンの複雑化により脆弱性対応の自動化が求められている．特にSBOMを用いた脆弱性検査では脆弱性を特定する工程は自動化されているが，優先度付けは依然人手に依存している．本研究は、ユーザへの人身被害を基準とした影響リスクに基づき優先度を判定する手法を提案する．提案手法はSBOMと外部情報から機能関連を特定し，誤動作が物理動作を伴うかを推定して影響リスクを4区分で評価する．自動車DCUを対象とした評価では，セキュリティ担当者の判定と比較して最大0.82の適合率と最低でも45 %以上の効率化を確認した．一方で，使用するLLMの種別により物理動作の判定精度に差があり，軽量モデルは過大判定の傾向が見られた．また，誤動作推定に必要な説明文の情報量は2017年以降増加しているものの依然不足していることが明らかとなった．

キーワード：脆弱性優先度付け，リスク評価，SBOM，SSVC

An Impact-Based Vulnerability Assessment Method for Prioritization in SBOM-Driven Vulnerability Scanning

Reika Nishimura.Arakawa[1,a)]　Yo Kanemoto[1]

Abstract: The growing complexity of software supply chains demands automation in vulnerability management. While SBOM-based scanning automates vulnerability identification, prioritization still relies heavily on manual analysis. This study proposes a method to prioritize vulnerabilities based on human safety impact. The approach derives functional dependencies from SBOMs and external sources, estimates whether malfunctions propagate to physical actions, and classifies risks into four levels. Evaluation using automotive DCU SBOMs showed up to 0.82 accuracy against human judgment and at least 45% reduction in assessment time. However, results varied by LLM type: lightweight models tended to overestimate physical impact. Furthermore, analysis of vulnerability descriptions revealed that although longer texts have increased since 2017, the information remains insufficient.

1. はじめに

ソフトウェアサプライチェーンは複雑化し，その結果として残存脆弱性の悪用リスクが増大している．この状況に対処するため，ソフトウェア構成の透明性を確保しセキュリティを強化する基盤として，Software Bill of Materials（SBOM）が注目されている．SBOMは，ソフトウェアを構成するライブラリやパッケージを機械可読な形式で体系化したドキュメントであり，自動生成が可能である．SBOMを脆弱性スキャンツールと連携することで，既知の脆弱性を迅速な同定・可視化できる．さらに，特定された脆弱性に対してステム影響度を評価し，パッチ適用や設定変更，緩和策の導入といった是正措置を講じることで，リスクの低減に繋がる．

SBOMを用いた脆弱性管理のプロセスは，(1) 脆弱性の特定，(2) 対応優先度付け，(3) 情報共有，(4) 脆弱性対応から構成される [17]. 第一段階の脆弱性の特定は，SBOMと脆弱性スキャンツールの照合により高い水準で自動化されている．一方で，対応優先度付けの自動化は依然として不

[1] NTT 株式会社
NTT Social Informatics Laboratories
3-9-11 Midori-cho Musasino-shi Tokyo
[a)] reika.arakawa@ntt.com

十分である．これは，脆弱性の影響を評価する際に，対象システムのアーキテクチャ，依存関係，運用要件，セキュリティ設定値，代替策の可用性といった要素を踏まえた専門的な分析・判断が必要となるためである．実務においては，セキュリティ担当者が Common Vulnerability Scoring System (CVSS) の基本値 [12] を起点とし，各システムへの影響を考慮して手作業で優先度付けが行っている．しかし，人手による対応にはスケーラビリティの限界があり，脆弱性対応の遅延を招くため，自動化技術の導入が強く求められている．

優先度付けの判断を支援する既存フレームワークとして，Stakeholder-Specific Vulnerability Categorization (SSVC) [11] がある．これはリスクを脆弱性・脅威・影響の三要素から捉え，システムへの影響を含めた 4 観点を定義しているが，リスク値を判定するロジックや実装は提供されていない．本研究では，システムへの影響の中でも人身被害に焦点を当て，影響リスクを評価する手法を提案する．提案手法は，SBOM の依存関係と外部情報から関連機能を特定し，機能間の関連性に基づき誤動作の波及を Large Language Model (LLM) を用いて推定し，最終的に影響リスクを 4 区分で評価する．提案手法のプロトタイプを実装し，自動車の ADAS と Electric Powertrain の DCU を想定した SBOM を用い，特定された 31 件の脆弱性で評価した．複数の LLM モデルを比較した結果，適合率は最大 0.82，判定処理時間はセキュリティ担当者と比較して最低でも 45%の効率化が確認された．本研究の貢献は以下の通りである．

- 脆弱パッケージに対し，人身被害を基準とする影響リスクに基づく優先度付け手法を提案した．手法はシステム機能との関連性を特定し，脆弱性悪用による誤動作が物理動作へ波及するかに基づき評価する．
- 自動車 DCU を想定した SBOM による評価では，セキュリティ担当者の判定と比較して適合率は最大 0.82，判定時間を 45 %以上削減できた．また，LLM の種別により精度差があり，軽量モデルは過大判定の傾向があることが確認された．
- 誤動作推定に必要な脆弱性説明文の分析では，2017 年以降情報量の多い脆弱性は平均 6.19 %に達し増加傾向にあるが，依然として不十分であることが明らかとなった．

2. 背景

2.1 脆弱性の対応優先度付け

脆弱性報告件数は毎年増加しており，2024 年は前年比約 13.89%増で，1 日平均約 105 件が公開されている[*1]．その約半数は緊急または重要と分類され，対応優先性が高い脆弱性が多い傾向にある．さらに，脆弱性の公開から悪用までの期間も短縮しており，2023 年は平均で公開後 5 日で悪用されるとの報告がある [4]．このような状況において，組織としては全ての脆弱性対応を人手のみで行うことは困難であり，自動化の導入が不可欠である．脆弱性対応における自動化の実現度合いを著者らが独自にレベル分けし，本研究が目指す自動化レベルを提示する．表 1 に，各レベルにおける自動化の範囲と概要を表す． 1 章で先述の通り，現時点では検知は自動化されているものの，評価や対応は依然として人手に依存しているため，自動化レベルは 1 に留まっている．本研究では，評価を自動化する手法を提案し，自動化レベル 2 の実現を目指す．

[*1] 著者らが CVE 番号が付与された既知の脆弱性 296,102 件を対象に集計・解析した結果である．

2.2 フレームワークと既存技術

従来，脆弱性評価には，CVSS が広く用いられてきた．CVSS の基本値は脆弱性そのものの深刻度を示すが，実際のシステムにおけるリスク評価には環境値の設定が必要であり，その設定は手動分析に依存する．また，CVSS ではビジネス上の影響や人命への影響を十分に反映できない課題がある．この問題に対し，リスクを脆弱性・脅威・影響の三要素から捉え，優先度判断を支援するフレームワークとして SSVC が提案されている [11]．SSVC は，ステークホルダーごとに意思決定を支援する決定木を提供する．例えば，パッチ適用を行う組織向けの決定木では，攻撃コードの公開状況（Exploitation），システムの露出度（Exposure），攻撃の自動化可能性（Utility），およびシステムへの影響（Human Impact）の 4 基準に基づき，分岐結果として優先度を判定する．

このうち Human Impact はシステムの特性を踏まえた運用者の分析を要するため，自動化が難しい．Human Impact は，システムの安全性への影響（Situated Safety Impact）と，企業のミッションにおける影響（Mission Impact）に分けられる．特に Situated Safety Impact は，身体的被害（Physical Harm），環境への影響（Environment），財務的影響（Financial），心理的影響（Psychological）の 4 要素から構成される．このうち Physical Harm に着目し，Situated Safety Impact に基づく影響リスクを評価対象とする．

SSVC を応用した市中製品 [3], [6], [10], [14] では，Human Impact の判定を簡略化したり，対象脆弱性を限定することで暫定的なリスク評価を行っている．また，一部製品はセキュリティ担当者による手動設定を前提とした設計で実現されている．

2.3 研究課題

本研究が想定する運用環境において，影響リスクの評価に利用可能な情報は，システム概要，生成された SBOM，および脆弱性検査ツールで検出された脆弱パッケージに限定される．この制約下で，検出された脆弱性がエンドユーザに人身被害を及ぼす可能性を判定する過程には 4 つの研究課題が存在する．図 1 は評価対象システムの模式図を示

表 1　著者らの独自に定義した脆弱性対応の自動化レベルと概要

レベル	自動化の範囲	自動化の概要	対応の主体
5	完全自動化	デジタルツイン等を活用し，予兆検知から脆弱性特定，対応策実行，検証まで全て自動化．人間の関与を必要としない	システム
4	高度な自動化	パッチ適用，設定変更，隔離など対応手段から最適策を自動で選択し実行．人間は例外時のみ介入	システム
3	条件付き自動化	フレームワークに基づき，システムが推奨対応を選択・提示し，人間の最終承認時のみ実施	人間＋システム
2	部分自動化	脆弱性情報の収集や影響範囲，既知の回避策などを自動で整理・提示するが，人間が必ずレビューし判断責任を持つ	人間＋システム
1	対応支援	脆弱性検知は自動化され，影響度や対応策の提示までをツールが支援するが，最終判断・実行は人手で行う	人間
0	なし	脆弱性の検知から対応まで全て人手で実施	人間

しており，システムは複数のソフトウェアとそれぞれに内包される複数のライブラリで構成される．脆弱パッケージはライブラリ内部に存在する場合もあれば，ライブラリを介さずソフトウェアに直接組み込まれる場合もある．

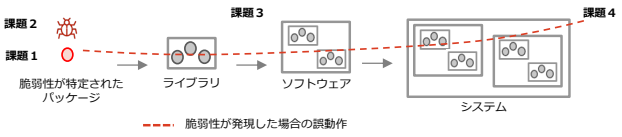

図 1　研究課題の模式図

課題 1：パッケージ機能の不確定性 SBOM 上で脆弱性が検出されたパッケージについて，提供機能や実行時の役割（API ／ライブラリ／プラグイン時のみ等）が不明確であり，影響の推定が困難である．

課題 2：誤動作様態の不明確性 当該脆弱性が発現した場合に，当該パッケージ内で生じ得る誤動作，例えば，任意コード実行，画像カラーリング失敗等の具体的動作が特定しにくい．

課題 3：システム内配置・関与機能の不明確性 脆弱パッケージがシステム内のどこに位置し，どの機能に関与するかが把握困難である．

課題 4：影響リスクのマッピング困難性 脆弱性の発現がシステム全体としてどのような挙動逸脱を引き起こし，最終的に人身被害へ至り得るかの判別が難しい．

3.　提案手法

3.1　提案手法の概要

本研究は，SBOM とシステム概要情報に基づく脆弱性管理を対象とし， 2.3 節で説明した対応優先度判定に伴う四つの課題を解決する手法を提案する．提案手法は四つのステップで構成され，課題 1,2,3,4 が処理ステップ 1,2,3,4 に対応している（図 2）．まず，検出された脆弱パッケージについて依存関係情報を用いて機能を特定し，脆弱性に起因する機能誤動作を推定する．次に，脆弱パッケージのシステム内での位置付けと関連機能を特定し，最後に評価結果を 4 区分の影響リスクを出力する．

3.2　各処理の詳細

Step-1 パッケージ機能の特定 SBOM を脆弱性スキャンツールで検査し，検出された脆弱パッケージの情報を抽出する．抽出対象は，パッケージの URL（PURL），依存ファイルパス，および脆弱性識別子（CVE-ID）である．次に課題 1 のパッケージの機能を特定するため，外部のパッケージ情報データベースを参照し，依存関係の一覧とメタ情報に記載された概要テキストを取得する．取得したテキストを基に機能キーワードリスト $K^p = \{k_1^p, ...k_n^p\}$ に変換する．例えば，コンパイラに関係するパッケージの例では，K^p は {`json parser`, `syntax tokenization`, `AST tree`} となる．

Step-2 脆弱性誤動作の推定 脆弱性が発動した場合に，Step-1 で特定したパッケージ機能に生じる誤動作を特定する．課題 2 に対処するために，各パッケージについて CVE-ID を基に外部の脆弱性情報データベースを参照し，説明文と脆弱性種別（CWE-ID）を取得する．説明文から以下に示す観点で誤動作に関する記述を抽出する．

- 脆弱性が見つかったソフトウェア名称
- 脆弱性が発現する操作条件
- 脆弱性が発現する関数の名称とファイルパス
- 脆弱性が悪用されたときに生じる誤動作

抽出される誤動作の例では，API キーの不正な読み取りや UI 上でのクレデンシャル情報の不正表示といった内容である．一部の CVE-ID では説明文が短く，上記観点で解析しても結果が得られない場合がある．この場合，過去の脆弱性情報を CWE-ID ごとに事前解析したデータを参照する．該当 CWE-ID に一致するレコードを抽出し，さらにソフトウェア名，関数名，ファイルパスが一致または類似するものに絞り込む．抽出レコードに記載された「脆弱性が悪用されたときに生じる誤動作」テキストを，当該 CVE-ID の誤動作として補完する．

Step-3 機能特定と誤動作の推定 脆弱性が検出されたパッケージについて，評価対象システム内での位置を特定し，誤動作が最も波及する機能を推定する．課題 3 に対処するために，まず Step-1 で得た依存ファイルパスに基づき，パッケージが記述された依存ファイル

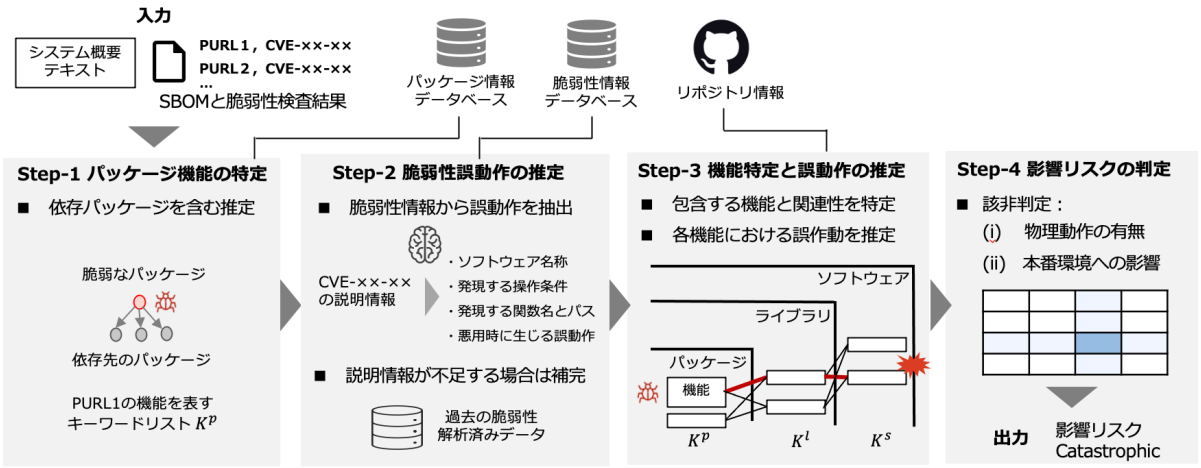

図 2 提案手法の全体像

の階層を確認し，上位階層に存在するライブラリやソフトウェアを特定する．例えば，依存ファイルパスが‘./apollo/modules/dreamview/frontend/yarn.lock’の場合，脆弱パッケージは yarn.lock に記載され，その上位の frontend，dreamview，modules をライブラリ，最上位の apollo をソフトウェアと見做すことで，システムに apollo が導入されていることを特定できる．

次に，ソフトウェアのリポジトリを特定し，ソフトウェアおよびライブラリの機能を推定する．リポジトリは，ソフトウェア名をもとにデータベースを検索し，候補となる所有者と機能を表すテキストを取得する．その後，各候補の所有者のレポジトリに対し，依存ファイルパスを含む URL を作成し，これがアクセス可能な場合に該当リポジトリと特定する．特定されたリポジトリにおいて，各階層の機能を推定するため，該当ディレクトリ内 README.md ファイルや package.json ファイルを探索し，機能概要に関わるテキスト情報を抽出する．抽出したテキストは Step-1 と同様に機能キーワードリストに変換し，パッケージ・ライブラリ・ソフトウェア間でキーワードの意味的類似度を算出する．各ペアにおいて最も類似度が高いキーワード同士を，機能の関連性があると判定する．

パッケージ p，ライブラリ l，ソフトウェア s の集合を $X \in \{p, l, s\}$，機能キーワードリストを $K^X = \{k_1^X, ...k_n^X\}$，埋め込み関数 $\phi : k \to \mathbb{R}^d$，類似度関数 $sim(x, y)$ とし，パッケージ p からソフトウェア s において最も高い機能関連性を持つキーワードペア $\mathcal{P}$ は，キーワードペア集合 $\mathcal{A}$ を用いて次の式で表される．

ライブラリ $l \in \mathcal{L}$ が存在する場合は，

$$\mathcal{A}_{pl} = \left\{ (k_i^{(p)}, k_j^{(\ell)}) \mid 1 \le i \le n_p,\ 1 \le j \le n_\ell \right\}$$
$$\mathcal{A}_{ls} = \left\{ (k_i^{(l)}, k_j^{(s)}) \mid 1 \le i \le n_l,\ 1 \le j \le n_s \right\},$$
$$\mathcal{P}_{pl} = \arg\max_{(i,j)\in\mathcal{A}_{pl}} sim(\phi(k_i^{(p)}), \phi(k_j^{(\ell)}))$$
$$\mathcal{P}_{ls} = \arg\max_{(i,j)\in\mathcal{A}_{ls}} sim(\phi(k_i^{(l)}), \phi(k_j^{(s)}))$$

ライブラリ $l \in \mathcal{L}$ が存在しない場合は，

$$\mathcal{A}_{ps} = \left\{ (k_i^{(p)}, k_j^{(s)}) \mid 1 \le i \le n_p,\ 1 \le j \le n_s \right\}$$
$$\mathcal{P}_{ps} = \arg\max_{(i,j)\in\mathcal{A}_{pl}} sim(\phi(k_i^{(p)}), \phi(k_j^{(s)}))$$

と表される．類似度が最も高い機能キーワードのペア $\mathcal{P}$ を Step-4 の影響リスク判定で利用する．

Step-4 影響リスクの判定 課題 4 を解決するために Step-1,2,3 で特定したパッケージ機能，システム構成，脆弱性悪用時の誤動作，および機能関連性の情報に基づいて，観点にわけて影響リスクを判定する．観点は次の二つで，両者の判定結果をクロスマッピングした結果を影響リスクとして出力とする．

(i) 脆弱パッケージがシステム本番環境に影響する可能性
(ii) 誤動作が物理動作を伴い安全機能に関与する可能性

(i) は，脆弱なパッケージを含む上位階層が本番環境に関係する場合は，影響リスクが高まると考える．そのため，パッケージが記述された依存ファイルのパス情報を用い，各階層名が定義済みの本番環境階層名との一致を判定する．

Level1. 本番環境で使用されない（CI/ドキュメント/サンプル/デモ），例. docs，demo，jenkins
Level2. 本番環境で使用される可能性は低い（テスト/ベンチ/実験/開発），例. mocks，dev，test
Level3. 本番環境で使用される可能性は低い（キャッシュ/一時生成物/仮想環境），例. .cache，temp
Level4. 本番環境で使用される可能性がある（ビルド/配布物），例. build，src，dist

また，(ii) は推定された誤動作の内容が物理動作を伴う場合，ユーザへの人身被害のリスクが高まる．誤動作の物理動作の有無を判定する．これは，Step-3 で特定された機能とその誤動作の内容，システム概要のテキスト情報を入力に，LLM を用いて最も類似する Level を判定させる．

Level1. 物理動作を伴わない
例. ログデータの削除，シミュレータ，データの改ざん，等
Level2. 物理動作を伴うが安全領域外
例. ランプ無効化，ディスプレイ誤表示，インフォテイン

メントのクラッシュ，表示の乱れ
Level3. 物理動作を伴い正常系から逸脱
例．安全設定値の改ざん，物体認識エラー，エンジン停止
Level4. 物理動作を伴い安全機能を無効化
例．ブレーキ無効化，リミッター無効化，インターロックバイパス，ABS 制御の無効化

(i) および (ii) の判定結果を表 2 でクロスマッピングし，その値を脆弱パッケージの影響リスクとする．影響リスクは，SSVC の Situated Safety Impact に基づき，リスクの大きい順に *Catastrophic*，*Hazardous*，*Major*，*Minor* の 4 区分で評価する．例えば，(i) が Level3 かつ (ii) が Level4 の場合，*Catastrophic* となる．併せて，判定理由も出力する．

表 2 (i) と (ii) の結果に基づく影響リスクの判定表

		(i) の結果			
		Level1	Level2	Level3	Level4
(ii) の結果	Level1	*Mi*	*Mi*	*Mi*	*Ma*
	Level2	*Mi*	*Mi*	*Ma*	*Ha*
	Level3	*Mi*	*Ma*	*Ha*	*Ca*
	Level4	*Ma*	*Ha*	*Ca*	*Ca*

Mi:Minor，*Ma:Major*，*Ha:Hazardous*，*Ca:Catastrophic*

4. 実験

4.1 実験設定

提案手法の有効性を評価するために，次の 3 つのリサーチクエスチョン（RQ）を設定した．
RQ1. 提案手法は，CVSS スコアなどの従来手法と比較して，影響リスクの優先度付けに相違をもたらすか．
RQ2. 提案手法による判定結果は，セキュリティ担当者による手動分析結果とどの程度一致するか．
RQ3. 提案手法を適用することで，脆弱性対応優先度付けにおける作業時間をどの程度短縮できる可能性があるか．

4.2 データセット

SBOM の作成と脆弱性スキャン結果の取得 脆弱性によって被害発生した場合，影響が甚大ですぐに対応すべきと考えられる例として，自動運転機能を備えた自動車を想定し，ドメイン型アーキテクチャを模した制御ファイルシステムを構築した．システムはコントロールゲートウェイユニットを中心に，ADAS ドメインコントロールユニット（DCU）および Electric Powertrain DCU 等が接続された構成とした．各 DCU のルートディレクトリには bin/，etc/，opt/，var/，data/，boot/を配置し，etc/ 配下に各機能のソースコードや設定ファイルを格納した．ADAS には Apollo，OpenADAS 等のオープンソースソフトウェア（OSS）を組み込むことを想定し，Electric Powertrain についても同様に Autoware 等の OSS を組み込むことを想定した．SBOM の取得にあたっては，各 DCU の etc/配下に設けた adas.d/および powertrain.d/をスキャン範囲とし，Trivy を用いて SBOM の生成と脆弱性スキャン結果を取得した．

```
/ ..... ルートディレクトリ
├─/opt/ ..... 実行体
│  └─ ... ..... 各 DCU のバイナリ
├─/etc/ ..... 設定/ソースコード
│  ├─/adas.d/ ..... ADAS DCU
│  │  ├─/openAdas/ ..... OSS
│  │  └─ ... ..... 以下省略
│  ├─/powertrain.d/ .... ElectricPowertrain DCU
│  │  └─/Autoware/ ..... OSS
└─ ... ..... 以下省略
```

脆弱性情報 2025 年 7 月 10 時点に NVD の脆弱性情報データベースから収集した．Step-2 において補完に用いた事前解析済みの脆弱性数は 250,045 件であった．

4.3 実装

提案手法はステップごとにモジュールを分離し，Python 3.11.7 によりプロトタイプを実装した．脆弱性情報は PostgreSQL データベースに記録・管理した．実験は MacBook Pro（Intel Core i5 2.0 GHz CPU，32GB RAM，4TB SSD）上で実行した．機能キーワードリストの類似度には，Embedding でベクトル化した値をコサイン類似度で算出した．
SBOM の作成 SBOM の生成および脆弱性スキャンは，Trivy バージョン 0.61.0 を用いた．
LLM のモデル OpenAI 社の API を使用し，gpt-o1 のモデルを用いた．

5. 評価

5.1 評価指標

4.2 節のデータセットから検出された 31 件の脆弱性について，セキュリティ運用業務に従事する担当者三名にリスク評価を依頼した．システム概要を表すテキスト文章（500 文字）と脆弱性検査結果を渡し，本研究のプロトタイプ実装の入力と同じ条件にした．セキュリティ担当者によって判定結果にばらつきがあったため，著者らで結果を選別して正解ラベルとした．具体的なばらつきの詳細は，6.3 節で後述する．評価指標には，4 区分それぞれで True Positive（TP），False Positive（FP），Precision($Pr = \frac{TP}{TP+FP}$) を計算し，マクロ平均適合率 ($MacroPrecision = \frac{1}{4}\Sigma_{i=1}^{4} Pr_K, K$ は *Catastrophic*，*Hazardous*，*Major*，*Minor*) を用いた．

5.2 4.1 節のリサーチクエスチョンに対する評価

RQ1.CVSS に基づく優先度付けとの相違
31 件の脆弱パッケージについて，CVSS スコアと影響リスク判定結果を比較した．CVSS スコアが 9.0-10 の「非常に深刻」と分類される脆弱性は 6 件存在したが，そのうち 1

件のみが *Catastrophic* と判定され，残りは *Major* あるいは *Minor* に判定された．具体例として，脆弱パッケージ @babel/traverse の脆弱性が挙げられる．この脆弱性はトランスコンパイル処理に起因し，任意コード実行の可能性があることから，CVSS スコアは 9.3 と設定されている．しかし，本手法による影響リスク判定では次のような理由に基づき *Minor* と評価された．「本脆弱性により HMI（Human-Machine Interface）機能に障害が生じ，歪んだ軌道のビジュアルや不正確な位置情報データが表示される可能性がある．この場合，ドライバーは不正確な表示に基づいて運転操作を誤り，接触事故により軽傷を負う可能性がある．ただし，安全制御機能を直接上書きまたは無効化する可能性は極めて低く，身体的損傷につながるリスクは限定的である．」この例は，CVSS スコアが高いが，脆弱パッケージの位置や機能関連性に基づく評価ではリスクが小さいと評価された例である．

RQ2. セキュリティ担当者の判定結果との一致度

セキュリティ担当者の結果に基づく正解ラベルとの比較を表 3 の Macro-Precision 列に示す．評価で異なるモデル種別を用いた結果，gpt-o1 で Macro-Precision が最大 0.82 を記録した．区分別では，*Catastrophic* の適合率が o1 で 1.0，gpt-4.1 で 0.85 と高かった一方で，*Major* や *Minor* は，0.63 や 0.71 に留まり，判定が難しい傾向が見られた．軽量モデル（o3-mini，4o-mini）は，*Hazardous* や *Catastrophic* を約 5 から 8 倍多く判定し，過大判定の傾向を示した．判定理由の分析では，軽量モデルが脆弱性の誤動作の内容をシステム概要に直接結びつけて過大評価する例が多かった．例えば，ファイルへの不正アクセスの脆弱性をセンサーデータの不正操作や安全機能無効化と関連づけて，リスクを過大に評価していた．これらの結果から，提案手法はモデル種別により精度が変動するため，LLM のプロンプト設計や判定基準の明確化が必要であることが示唆された

RQ3. 提案手法による評価時間の短縮可能性

プロトタイプを異なる LLM モデルを用いて 3 回ずつ実行して処理時間を取得した．その結果，o1 では 1 件あたりの脆弱性の判定に要した平均時間は約 120.79 秒，4o および 4o-mini の処理時間は，o1 と比較して約 27.04%速く，平均で 88.13 秒であった．また，セキュリティ担当者 3 名については，脆弱性ごとに判定に要した時間の計測を行った．3 名の結果を合わせて算出された 1 件あたりの判定時間の平均時間は約 221.82 秒であった．結果を表 5 にまとめる．提案手法はセキュリティ担当者と比較して，少なくとも約 45%の判定時間の削減効果が期待できる．

6. 考察

6.1 LLM モデル性能の違いによる判定への影響

実験では，使用する LLM の性能差が影響リスク評価に与える影響を検証した．入力とするシステム概要テキストは 500 字とし，比較対象はプロトタイプで利用可能な o1, gpt-4.1，o3-mini，GPT-4o，GPT-4o-mini の 5 モデルとした．31 件の脆弱性について判定結果を正解ラベルと比較した精度を表 3 に示す．表中の数値は，各区分に判定された脆弱性件数を表す．最も顕著な差は o3-mini であり，31 件中 20 件を *Catastrophic*，残り 11 件を *Minor* と判定した．モデル差を分析するため，影響リスク判定に直結する判定 (ii) の結果を比較した（表 4）．o3-mini では Level3（物理動作を伴い正常系から逸脱）に分類される件数が多く，*Catastrophic* 判定に偏った．o1 と比較すると，Level3, 4 ともに 3〜7 倍多く判定しており，物理動作を過大に評価する傾向が見られた．以上から，誤動作が物理動作を伴うかの判定ではモデル性能の影響が大きく，性能差が出にくい判定方法への改良が必要であることが明らかとなった．

6.2 入力で与える情報量の違いによる判定への影響

入力の評価対象のシステム概要テキストにおいて，文章量の違いが影響リスクの判定にどの程度影響するのかを検証した．文章量の条件を 3 つ設定し，一つはテキストの文章量を与えない 0 文字の場合，二つ目は簡潔な文で 77 文字の場合（例：家庭用の 5 人乗り自動車で自動運転機能を搭載），三つ目は各 DCU の機能仕様を反映した約 500 文字とした．比較した結果を表 6 で示す．3 条件での影響リスクの一致率は 31 件中 20 件で約 64.51%であった．概要テキストを与えない場合では，判定理由の文中に 31 件中 6 件でプロンプトで示していた例文に含まれる単語が観測された．また，77 文字の場合も誤動作の理由が抽象的な結果が多く，判定精度も 0.33 と低い結果となった．

6.3 セキュリティ担当者ごとの影響リスク判定のばらつき

セキュリティ担当者 3 名による 31 件の脆弱パッケージの判定では，11 件で結果が一致しなかった．中には，一方が *Catastrophic*，他方が *Minor* と評価するなど大きな差が見られた．また，判定時間にも差があり，業務経験年数の違いにより合計時間に 1.42 倍の差が確認された．これらの結果は，同一情報が与えられても評価が担当者の経験や判断基準に強く依存することを示している．判定が分かれた多くの事例は，説明情報が不十分で誤動作内容が明確でなく，被害シナリオを想定しにくいものであった．熟練者は説明文に加えて CWE 情報を重視していたが，システムに与える致命的影響の推定には難しさを指摘していた．すなわち，情報不足は解釈の差を拡大し，誤動作や機能関連の不確実性が判定を困難にしている．

このばらつきは，実運用で一貫した優先度付け基準の策定を難しくする．属人的判断に依存した評価では効率性や客観性を確保できないため，説明情報を補完し基準を明確化・標準化する自動化手法が求められる．自動化にあたっては，複数担当者の知識や判断を反映し，一貫した分析観点と判定基準を確立することが重要である．

表 3 LLM のモデル性能の違いによる影響リスク判定への影響

モデル種別	判定結果の分布 ※1				影響リスクの判定精度
	Mi	Ma	Ha	Ca	Macro-Precision
gpt-o1	3	24	1	3	0.82
gpt-4.1	4	21	2	4	0.75
gpt-o3-mini	0	11	0	20	0.08
gpt-4o	1	22	4	4	0.47
gpt-4o-mini	1	14	8	8	0.14

※1 Mi:Minor，Ma:Major，Ha:Hazardous，Ca:Catastrophic

表 4 LLM のモデル性能の違いによる (ii) の比較

モデル種別	(ii) の結果の分布 ※1			
	Level1	Level2	Level3	Level4
gpt-o1	25	2	3	1
gpt-4.1	26	1	4	2
gpt-o3-mini	7	0	21	3
gpt-4o	19	5	4	3
gpt-4o-mini	12	8	5	4

(ii) では，脆弱性による誤動作が物理動作を伴い安全機能に影響するほど，Level 値を大きく設定している．

表 5 影響リスクの判定に要した平均時間と削減率の比較

比較対象	平均時間 (sec)	削減率 (%)※1
セキュリティ担当者 3 名	221.82	0.00
o1	132.07	-45.55
4o，4o-mini	88.13	-60.27

※1 セキュリティ担当者 3 名の平均時間を基準に比較

表 6 システム概要テキスト情報量の違いによる影響 ※1

文章量	判定結果の分布				影響リスクの判定精度
	Mi	Ma	Ha	Ca	Macro-Precision
0 文字	0	25	2	4	0.17
77 文字	0	26	0	5	0.33
500 文字	3	24	1	3	0.82

※1 実行に用いた LLM のモデルは gpt-o1.

6.4 脆弱性に関する説明情報の分析

提案手法の Step-2 では，脆弱性に関する説明情報が不足している場合，過去の類似する脆弱性情報を用いて補完を行った．本節では，補完手法の有効性を検証するために，既知の脆弱性に付随する説明文を対象とした解析を実施した．具体的には，2025 年 7 月 10 日時点で CVE 番号が割り当てられている 296,102 件の脆弱性について，`description` フィールドに記載された説明文を収集し，年ごとに文書をスペースで分割した単語数に基づいて分類した．その集計結果を図 3 に示す．30 単語未満の割合を灰色，30 単語以上 100 単語未満を水色，100 単語以上を黄色で表す．脆弱性の説明文に含まれる単語数の分布を分析した結果，30 単語未満である割合の平均は全期間で約 37.93%，30 単語以上 100 単語未満である割合の平均は約 69.7%であった．これらの結果は，説明文の大部分が 100 単語未満にとどまっており，脆弱性に関する十分な記述としては情報量が不足している可能性を示唆している．

一方で，100 単語以上である割合は，1999 年から 2016 年までの平均が 0.44%であったのに対し，2017 年以降 2025 年までの平均は 6.19%と顕著な増加を示した．特に 2024 年および 2025 年には 7.5%以上に達している．しかしながら，依然として全体に占める割合は低く，脆弱性に関する説明情報が十分とは言い難い状況にある．説明情報が不足する場合，脆弱性分析に要する時間の増大や，影響リスク

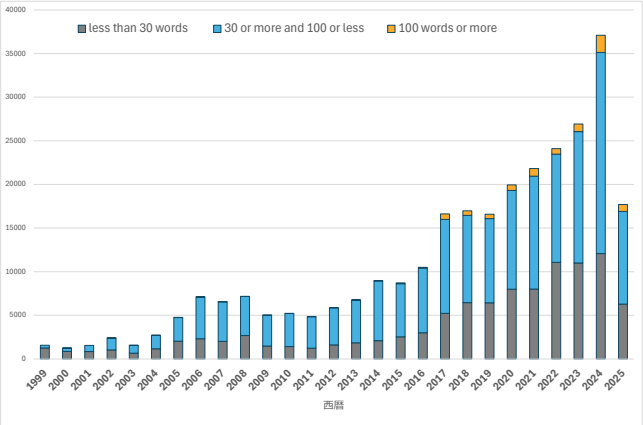

図 3 年度別に既知脆弱性の説明情報を解析した結果（単語数の割合）

の適切な判定を困難にする．したがって，個々の脆弱性情報が不十分である場合に，類似する脆弱性情報を補完的に活用することは，有効なアプローチであると考えられる．

6.5 研究倫理

提案手法の実装および評価においては，外部公開されている脆弱性情報データベース，パッケージ情報データベース，リポジトリ，OpenAI GPT に対して多数の API リクエストを送信した．しかし，いずれのアクセスにおいても，リミット上限を設定し，適切な範囲で実施した．

7. 関連研究

本研究で注目する SSVC は，Exploitation ／ Exposure ／ Utility ／ Human Impact の 4 観点に基づく，本節では各観点に関連する研究を整理し，本研究との差異を述べる．

Exploitation（公開状況） この観点は，脆弱性の実際の悪用状況や悪用確度に基づいている．Exploit Prediction Scoring System (EPSS) [7] は，公開済み脆弱性の悪用確率を機械学習で推定する手法であり，PoC コードや地下市場の取引情報を考慮している．Suciu ら [15] は，時間経過に伴う攻撃コードの高度化を反映した新たな指標を提案した．Jacobs ら [8] は EPSS のモデル設計を改良し，悪用／非悪用の識別性能を向上させた．

Exposure（システムの露出度） 脆弱性を含む資産の外部露出度に基づく観点である．ZMap [5] のような高速ス

キャン手法により，資産がインターネットからアクセス可能かを判定し，露出が確認されれば優先度を高める．企業内資産について攻撃グラフを用いてノードを資産や設定，エッジを到達可能性として表現し，攻撃経路を可視化して優先度を評価する研究がある [13]．さらに，和田ら [18] は，eBPF によりプロセスを一定時間監視し外部通信を行うソフトウェアを抽出，検査結果の `Attack Vector` との照合で，実際に外部通信が確認された脆弱性を特定する手法を提案している．

Utility（攻撃の自動化可能性） 攻撃者が得る取得権限の大きさや攻撃容易性に基づく観点である．Allodi ら [2] は，攻撃者が有限のリソースの中で手法を選択するとの前提から，多くの脆弱性は悪用されないとの仮説を検証した．100 万台超の PC から収集したマルウェアログの分析で，確立した攻撃は平均 2 年間継続し，新規脆弱性が直ちに標的化されるわけではなく，攻撃コード開発時期も脆弱性公開よりソフトウェア更新周期に依存することを示した．さらに彼らは [1] は，サイバー犯罪フォーラムを調査し，攻撃コードの価格・取引量・投稿数とマルウェア感染統計を突合した．その結果，地下市場の活動指標は実際の攻撃発生率と強く相関し，議論が活発なコードは悪用されやすく，高額なコードはコスト障壁から大規模利用されにくい傾向が確認された．すなわち，脆弱性の武器化や流通を巡る経済的インセンティブが，広範に悪用される脆弱性を決定づけている．

Human Impact（システムへの影響） この観点は，人命，健康，社会基盤，業務継続への影響に基づいている．Keskin ら [9] は，情報資産間の依存関係をモデル化し，脆弱性が伝播して業務プロセスを阻害する度合いを評価・スコア化する手法を提案した．Walkowski ら [16] は，CVSS では考慮されない組織固有の状況を反映し，副次被害や影響範囲，資産重要度を加味して優先度を算出する手法を示した．本研究も類似の観点を持つが，アプローチは異なる．既存研究は資産全体を対象とし，資産価値の人手定義を前提とする．一方，本手法はシステム内の機能関連を厳密に推定し影響範囲を限定するとともに，公開ソフトウェア情報から資産価値を推定することで，人手による詳細定義を不要とする点に特徴がある．

8. おわりに

本研究は，脆弱性の対応優先度決定を目的とし，システム利用者への人身被害の可能性に基づく影響リスク自動評価手法を提案した．手法は SBOM 情報からシステム構成と機能の関連を特定し，脆弱性悪用による誤動作が物理動作に波及するかを推定して影響リスクを判定する．実験では，物理動作の判定において LLM の性能差により結果が大きく変動する課題が確認された．今後は処理の改良と判定基準の明確化により精度向上を図る．

参考文献

[1] Luca Allodi. Economic Factors of Vulnerability Trade and Exploitation. In *ACM CCS*, 2017.

[2] Luca Allodi and Fabio Massacci. The Work-averse Attacker Model. In *ECIS*. Association for Information Systems, 2015.

[3] ASSURED. 脆弱性対策からリスク管理までオールインワンで実現するクラウドサービス，2025. `https://yamory.io/service`.

[4] Casey Charrier and Robert Weiner. How Low Can You Go? An Analysis of 2023 Time-to-Exploit Trends, 2024.

[5] Zakir Durumeric, David Adrian, Phillip Stephens, Eric Wustrow, and J. Alex Halderman. Ten Years of ZMap. In *ACM IMC*, page 139–148, 2024.

[6] Future. FutureVuls Document, 2025. `https://help.vuls.biz/manual/`.

[7] Jay Jacobs, Sasha Romanosky, Benjamin Edwards, Idris Adjerid, and Michael Roytman. Exploit Prediction Scoring System (EPSS). *Digital Threats: Research and Practice*, 2021.

[8] Jay Jacobs, Sasha Romanosky, Octavian Suciu, Ben Edwards, and Armin Sarabi. Enhancing Vulnerability Prioritization: Data-Driven Exploit Predictions with Community-Driven Insights. In *EuroS&P*, 2023.

[9] Omer Keskin, Nick Gannon, Brian Lopez, and Unal Tatar. Scoring Cyber Vulnerabilities based on Their Impact on Organizational Goals. In *Systems and Information Engineering Design Symposium (SIEDS)*, 2021.

[10] MACNICA. 開発部門が直面する脆弱性管理の課題に対策を！ 〜SSVC や脆弱性統合管理ツール活用による成功へのヒント〜, 2024. `https://mnb.macnica.co.jp/2024/10/DevSecOps/SSVC.html`.

[11] NIST. PRIORITIZING VULNERABILITY RESPONSE: A STAKEHOLDER-SPECIFIC VULNERABILITY CATEGORIZATION (VERSION 2.0), 2021. `https://www.sei.cmu.edu/documents/606/2021_019_001_653461.pdf`.

[12] NIST. Vulnerability Metrics, 2024.

[13] Steven Noel, Sushil Jajodia, Lingyu Wang, and Anoop Singhal. Measuring security risk of networks using attack graphs. *International Journal of Next-Generation Computing*, 2010.

[14] SecPod. Understanding SanerNow Risk Prioritization Engine, 2025. `https://www.secpod.com/blog/understanding-sanernow-risk-prioritization-engine/`.

[15] Octavian Suciu, Connor Nelson, Zhuoer Lyu, Tiffany Bao, and Tudor Dumitraș. Expected exploitability: Predicting the development of functional vulnerability exploits. In *USENIX Security*, 2022.

[16] Michał Walkowski, Maciej Krakowiak, Marcin Jaroszewski, Jacek Oko, and Sławomir Sujecki. Automatic CVSS-based Vulnerability Prioritization and Response with Context Information. In *IEEE SoftCOM*, 2021.

[17] 経済産業省 商務情報政策局 サイバーセキュリティ課. ソフトウェア管理に向けた SBOM（Software Bill of Materials）の導入に関する手引 ver 2.0, 2024. `https://www.meti.go.jp/press/2024/08/20240829001/20240829001-1r.pdf`.

[18] 和田泰典, 荒川玲佳, 鐘本楊, and 上原貴之. 機器のソフトウェアやモジュールの通信動作の可視化データを用いた脆弱性リスク判定手法. Technical report, 2025 年暗号と情報セキュリティシンポジウム, January 2025.