脆弱性管理のためのCPEマッチング手法の提案

佐々木 満春[1,a)]　山嵜 麿与[1]

概要：近年，ソフトウェアなどの脆弱性の報告数は増加傾向にあり，数多くの脆弱性に対処するためには，効率的に脆弱性情報を収集・管理する必要がある．脆弱性情報のデータベースとして，National Vulnerability Database (NVD) がよく知られている．脆弱性には，Common Platform Enumeration（CPE）とよばれるソフトウェアなどの製品を特定するための識別子が付与されており，脆弱性が影響を与える製品を判別することができる．CPE の一覧は CPE 辞書にて管理されているため，辞書内の CPE と脆弱性に付与された CPE を対応付ける CPE マッチングにより対処が必要な脆弱性を自動で収集することが可能となる．しかし，実際の脆弱性に付与される CPE には表記ゆれがあり，単純な文字列比較では CPE 辞書内の CPE と対応付けられず，対処が必要な脆弱性を収集できない恐れがある．そこで本稿では，CPE 文字列の特徴に加え，共起性や脆弱性の特徴を用いた高精度な CPE マッチング手法を提案する．また，有識者によって作成されたデータセットを用いて評価実験を行い，提案手法の有効性を確認した．

キーワード：脆弱性管理，CPE マッチング，CPE，NVD

CPE matching method for vulnerability management

Mitsuharu Sasaki[1,a)]　Mayo Yamasaki[1]

Abstract: In order to deal with increasing the number of vulnerabilities, it is necessary to collect and manage vulnerabilities efficiently. The National Vulnerability Database (NVD) is well known as a repository of standards-based vulnerability management data. Each vulnerability is assigned Common Platform Enumeration (CPE) to specify the software or other products and determine which products are affected by vulnerabilities. Since the list of CPEs is managed in the NIST's CPE dictionary, it is possible to automatically collect vulnerabilities by CPE matching. However, it is difficult to collect these vulnerabilities because the CPEs have orthographical variants and doesn't exactly match the ones in the CPE dictionary. In this paper, we propose a highly accurate CPE matching method that uses a co-occurrence of CPEs and features of vulnerability. We evaluated the proposed method by experiments using a dataset created by experts and confirmed the effectiveness of the proposed method.

Keywords: Vulnerability Management，CPE Matching，CPE，NVD

1. はじめに

IPA の情報セキュリティ 10 大脅威 [1] によれば，組織における脅威の上位には，ランサムウェアによる被害や機密情報の窃取などが挙げられており，攻撃手口の一例としてソフトウェアなどの脆弱性が悪用され，不正アクセスなどの攻撃の入り口となっていることが報告されている．また，NICT の NICTER 観測レポート 2020[2] によれば，Mirai 亜種をはじめとする IoT ボットの活動が観測されており，既知の脆弱性を持つ機器が依然として攻撃の被害を受けている状況が報告されている．したがって，脆弱なソフトウェアなどを未対策のまま使用することは組織にとって深刻なリスクであり，セキュリティインシデントを未然に防ぐためには，企業のセキュリティ担当者やシステム管理者が自組織で使用しているソフトウェアなどの脆弱性情報を

[1] NTT 社会情報研究所
NTT Social Informatics Laboratories
[a)] mitsuharu.sasaki.ry@hco.ntt.co.jp

収集し，脆弱性に対して適切に対応する必要がある．しかし，ソフトウェアなどの脆弱性の報告数は，2018 年には 16,511 件，2019 年には 17,305 件，2020 年には 18,356 件となっており，多くの脆弱性が報告されるとともに近年増加傾向にある [3]．全ての脆弱性情報を収集し，確認することは容易ではないことから，企業のセキュリティ担当者やシステム管理者は，自組織内で使用しているソフトウェア情報などを元に脆弱性データベースやベンダのウェブサイトなどから脆弱性情報を選別して収集しなければならない．

脆弱性情報を収集・管理するためのデータベースとして，National Vulnerability Database (NVD)[3] がよく知られている．NVD は National Institute of Standards and Technology(NIST) によって運営されており，世界中で報告された脆弱性情報が登録されている．脆弱性情報には Common Vulnerabilities and Exposures(CVE) と呼ばれる脆弱性の識別子（CVE-ID）が付与されている．例えば，ある組織の発行する脆弱性対策情報と，他の組織の発行する脆弱性対策情報とが同じ脆弱性に関する対策情報であることを判断したり，対策情報同士の相互参照や関連付けに利用することが可能である [4]．また，各脆弱性情報には情報セキュリティ対策の自動化と標準化を実現する技術仕様にて定義された SCAP[5] の Common Platform Enumeration(CPE) による識別子が付与されている．CPE にはハードウェア，オペレーティングシステム，アプリケーションなどのプラットフォームを識別するための構造化された識別子が記述されているため，CPE を参照することで脆弱性の影響を受ける製品を特定することが可能である [6]．CPE(バージョン 2.3) の構成要素を表 1 に示す．CPE は以下のフォーマットで表される．

$$cpe : 2.3 : \{Part\} : \{Vendor\} : \{Product\} : \{Version\} : \{Update\} : \{Edition\} : \{Language\} : \{SW\_Edition\} : \{Target\_SW\} : \{Target\_HW\} : \{Other\}$$

例えば，Microsoft Internet Explorer 8.0.6001 Beta を表現する場合は，

$$cpe : 2.3 : a : microsoft : internet\_explorer : 8.0.6001 : beta : * : * : * : * : * : *$$

と表記される．ここで，*は任意の値を表している．CPE の一覧は NIST にて管理され，CPE 辞書として一般公開されている [7]．したがって，セキュリティ担当者やシステム管理者は自組織のシステムが利用しているソフトウェアリストなどに基づき，脆弱性を収集したい製品の CPE を CPE 辞書から事前に選択しておき，NVD で公開された脆弱性の CPE と対応付けすることで，必要な脆弱性のみを自動で取得することが可能となる．しかし，脆弱性に付与される CPE には表記ゆれがあり，CPE 辞書内の CPE と一致しない問題が指摘されている [8]．また，Ashokkumar ら [9] の研究によれば，2020 年 3 月時点において，NIST の CPE 辞書には 478,120 件の CPE が登録されており，2002 年から 2020 年 3 月までの NVD の CVE フィードには 140,300 件を超える脆弱性があり，2,454,708 件の CPE が登録されている．Ashokkumar らが，単純な文字列マッチングを用いて CPE の対応付けを行ったところ，CVE フィード中の脆弱性の少なくとも 50%に対して，CPE 辞書上の CPE と対応付けすることのできない CPE が一つ以上付与されていた．さらに，CPE におけるベンダ名の約 4.1%が一致するベンダ名を持っておらず，CPE の 10.2%はベンダ名は一致していたが，製品名とは一致していなかったことが報告されている．つまり，既存の CPE 辞書を用いた CPE による脆弱性情報の収集を行った場合，本来収集しなければならない脆弱性の収集漏れが生じる可能性があり，既知の脆弱性が放置されることによって，セキュリティインシデントの発生リスクが高まる恐れがある．そこで本稿では，製品の特定のために最も重要となる {Vender} および {Product} に対する表記ゆれを解決するため，CPE 辞書内の CPE と脆弱性に付与された未知の CPE を対応付ける CPE マッチング手法を提案する．提案手法では，CPE の文字列間の類似度や共起性および CPE が付与された脆弱性間の類似度を考慮することで CPE 辞書内に登録されている CPE と脆弱性に付与された未知の CPE を対応付ける．評価用のデータセットを作成することで定量的な評価実験を行い，提案手法の有効性を確認した．

本研究の主な貢献は，文字列間の類似度に加え，共起性および CPE が付与された脆弱性間の類似度を考慮した高精度のマッチング手法を考案したことである．また，評価用データセットを作成し，従来手法および提案手法の定量的な精度評価により約 20%の対応付け精度向上を示している．以下，2 章で関連研究，3 章でデータセット，4 章で提案手法について述べ，5 章で評価実験，6 章でまとめと今後の課題について述べる．

2. 関連研究

これまで様々な CPE マッチングに関する研究が報告されている．Luis ら [8] は，レーベンシュタイン距離を用いて CPE 文字列を比較し，類似度を算出することで対応付けを行っている．また，Ashokkumar ら [9] はジャロ・ウィンクラー類似度を用いて CPE 文字列を比較し，類似度を算出することで対応付けを行っている．いずれの手法も CPE の文字列表記にのみ着目した手法である．CPE のタイプミスなど対応付け元となる CPE からおおきく変化しない場合には非常に有効である．しかし，CPE の {Product} 部分にベンダ名と製品名をアンダースコアなどの記号で連結した文字列を使用された場合などに誤って対応付けしてしまうことも多く，従来の CPE 文字列の類似度のみに着目した手法では対応付けの精度が低い．また，従来研究で

表 1　CPE の構成要素

構成要素	内容
Part	製品種別（h:ハードウェア，o:OS，a:アプリケーション）を判別する情報を記述する．
Vendor	製品を製造・作成した人または組織を記述する．
Product	製品名を記述する．
Version	製品のバージョン情報を記述する．
Update	製品のアップデートやサービスパックの情報を記述する．
Edition	製品のエディション（ソフトウェアなどの提供方法である無償版やプロフェッショナル版など）を記述する．
Language	製品で使用している言語を特定したい場合に記述する．
SW_Edition	製品が市場やエンドユーザー向けカスタマイズされたモデルがあれば記述する．
Target_SW	製品が動作するプラットフォーム環境を記述する．
Target_HW	製品が動作する命令セットアーキテクチャを記述する．
Other	ベンダーや製品に固有であり，他の構成要素に適合しない情報があれば記述する．

は CPE 辞書内の CPE とを対応付ける定量的な精度比較は行われていない．CPE 辞書には CPE の表記は異なるものの同じ製品を意味する CPE の存在が指摘されているため [10]，本研究ではセキュリティ業務経験を有する有識者により，予め CPE 辞書内の同一製品の CPE をグルーピングすることで評価用データセットを作成し，評価実験を行っている．CPE 文字列以外の要素を考慮した研究も報告されており，Ying ら [11] は脆弱性のサマリーとセキュリティベンダなどの外部レポートを参照することで，付与された CPE の矛盾を検知する手法を提案している．Ying らの研究は脆弱性のサマリーと外部レポート間の矛盾を検知することを問題としており，CPE 辞書との対応付けを問題設定としている点において本研究とは異なっている．

3. データセット

本研究にて使用するデータセットについて説明する．本研究では，CPE 辞書内に登録されていない CPE である未知の CPE が与えられたときに CPE 辞書内の CPE と正しく対応付ける．したがって，評価を行うためには未知の CPE とそれに対応する CPE 辞書内の CPE の組が必要となる．NIST で公開されている CPE 辞書内には同一製品に対して複数の CPE 表記が存在するため [10]，ある未知の CPE に対して，対応付けられる辞書内の CPE は複数存在することになる．また，データセットの作成においては，本来すべての CPE に対して分類したデータセットを作成し，評価すべきであるが，CPE 数は膨大であり，すべてを手動で分類することは現実的ではないため一部の製品に限定している．

データセットの作成は以下の手順により実施した．(ⅰ)NVD から全ての脆弱性をダウンロードする（2020 年 7 月 8 日時点で取得した脆弱性：146,498 件）．(ⅱ) 各脆弱性毎に付与された CPE を抽出し，{Vender}:{Product} 単位で出現頻度をカウントする．(iii) 出現数上位の製品を調査対象として，表記ゆれをすべての CPE から調査する．(iv)(ⅰ)〜(iii) までの作業を 3 年以上のセキュリティ関連業務経験がある有識者 3 名にて実施する．(ⅴ) 有識者 3 名による調査結果を取りまとめ，3 名の合意がとれた CPE の組を評価用のデータセットに登録する．なお，分類時はベンダが異なれば別製品として扱い，またプラグインについても別製品として扱っている．評価用データセット例を表 2 に示す．表 2 は，Cisco IOS における CPE の組を表しており，cisco:cisco_ios は過去の脆弱性で付与されたことのある CPE であるが，CPE 辞書に登録されていない．一方，cisco:ios は過去の脆弱性に付与されたことがあり，かつ CPE 辞書に登録されている CPE である．本手順にて作成したデータセットは，26 製品に対して対象の CPE が抽出されており，総 CPE 数が 151 件，CPE 辞書に登録されていない CPE が 70 件含まれている．なお，本データセットの作成に伴う作業稼働として約 3 週間程度の日数を要している．

表 2　データセット例（Cisco IOS）

CPE	CPE 辞書の登録有無
cisco:cisco_ios	なし
cisco:ios	あり

対象 CPE の表記ゆれを調査するためには，付与された脆弱性の概要，関連するセキュリティレポート，同時に付与された CPE と関連する脆弱性，対象製品のコンポーネントやプラグインなどを網羅的に調査する必要があった．したがって，CPE の文字列的特徴のみに基づく表記ゆれの解消は困難であるため，本稿では文字列特徴に加えて，CPE の共起性と脆弱性の類似度を用いた CPE マッチング手法を提案する．

4. 提案手法

本章では，提案手法の概要について述べた後，提案手法の各要素について詳細に説明する．

4.1 概要

提案手法のイメージ図を図 1 に示す．提案手法では，

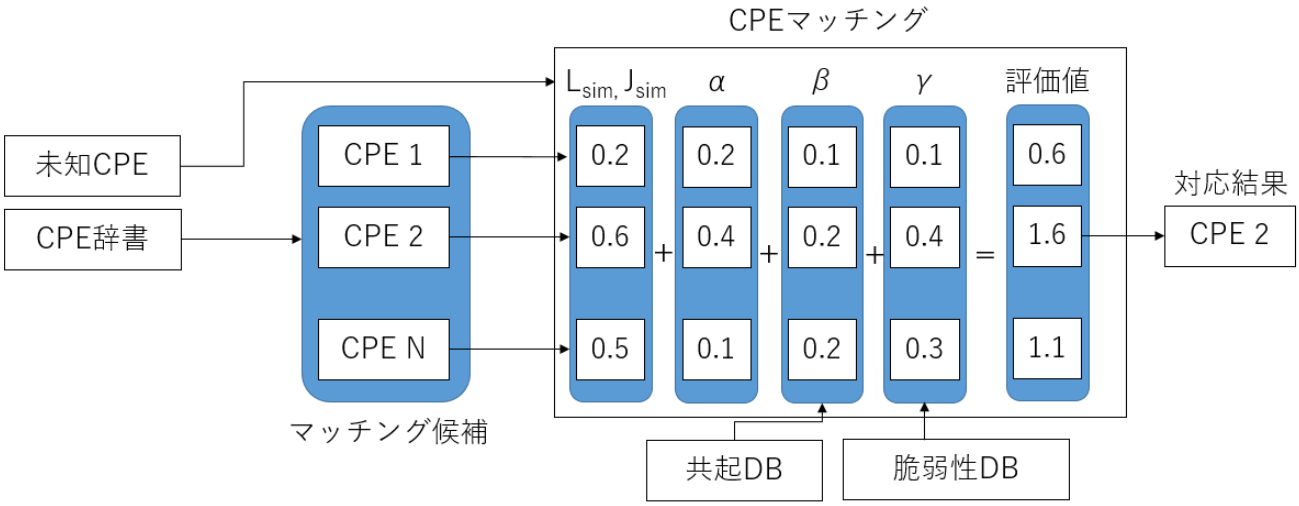

図 1　提案手法のイメージ図

未知の CPE と CPE 辞書を入力することで，CPE 辞書内で対応する CPE を出力する．評価値の計算過程では，Vender と Product の表記ゆれを考慮するため CPE 中の {Vender}:{Product} のみを使用し，それ以外の要素は除去している．CPE マッチングは 4 つの評価尺度から構成されている．(ⅰ) 文字列類似度: CPE 文字列間の類似度を評価する尺度であり，レーベンシュタイン距離を用いた類似度 L_{sim} もしくはジャロ・ウィンクラー類似度 J_{sim} を用いる．(ⅱ) 単語重複度: CPE 文字列間に存在する単語の重複度合を評価する尺度であり，CPE 文字列内に出現する同一単語をカウントすることで単語重複度 α を算出する．(iii) 共起度:　脆弱性における CPE の共起性を評価する尺度であり，脆弱性に付与された CPE の分布情報を用いることで共起度 β を算出する．(iv) 脆弱性類似度: CPE が付与された脆弱性の類似度を評価する尺度であり，脆弱性の説明文から得られた特徴量を用いることで脆弱性類似度 γ を算出する．それぞれの評価尺度から得られた値を足し合わせることで，入力された CPE を対応付けするための評価値を算出し，最も高い評価値が得られた CPE 同士を対応付ける．各評価値の算出過程では，未知の CPE や対応付け候補となる CPE の分布情報が格納された共起 DB や関連する脆弱性情報が格納された脆弱性 DB を参照している．また，提案手法においては CPE 辞書に登録されている CPE が対応付けの候補となるが，CPE 辞書内の CPE 数が多いため，対応付けの処理時間が非常に長くなる．そこで，評価前にレーベンシュタイン距離を用いた類似度やジャロ・ウィンクラー類似度を用いて得られた値により，一定の閾値以上の類似度を有する CPE のみを対応付け候補とし，後続の評価値を算出している．表記ゆれの傾向として，未知の CPE と対応付けるべき CPE における {Vender} および {Product} の文字列表記が同時に大きく異なることはないため，一定の類似度を有する CPE を対応付け候補としたほうが，対応付け精度を維持しつつ処理時間を削減できる．以下，各要素について詳細に説明する．

4.2　文字列類似度

提案手法では，従来手法で使用されているレーベンシュタイン距離を用いた類似度 L_{sim} もしくはジャロ・ウィンクラー類似度 J_{sim} を評価値として使用する．L_{sim} と J_{sim} は対応付けの評価値だけでなく，後続の処理を実行するかどうかを判断するための閾値としても用いている．

公開された脆弱性に付与されたある未知の CPE_i と CPE 辞書内に登録されたある CPE_j におけるレーベンシュタイン距離を用いた類似度 L_{sim} は (1) 式で表される．

$$L_{sim} = 1 - \frac{L(CPE_i, CPE_j)}{max\big(|CPE_i|, |CPE_j|\big)} \tag{1}$$

$L(CPE_i, CPE_j)$ は CPE_i から CPE_j への編集距離を表し，挿入・削除・置換の文字操作により，一方の CPE 文字列からもう一方の文字列に変形するために要する最小の操作回数である．また，$|CPE_i|$ は CPE_i の文字列長を表す．

ジャロ・ウィンクラー類似度 J_{sim} は (2) 式で表される．

$$J_{sim} = J_d + l * p * (1 - J_d)$$
$$J_d = \begin{cases} 0 & \textbf{if } r = 0 \\ \frac{1}{3}\left(\frac{r}{|CPE_i|} + \frac{r}{|CPE_j|} + \frac{r-\tau}{r}\right) & \textbf{otherwise} \end{cases} \tag{2}$$

ここで，l は文字列の先頭から共通する文字の長さを表しており，p は先頭で一致する文字に対する重みを表しており，通常 0.1 が使用される．また，r は CPE_i と CPE_j において一致する文字数を表しているが，位置の差が $\lfloor \frac{max(|CPE_i|,|CPE_j|)}{2} \rfloor - 1$ 以内にあれば 2 つの文字が一致しているとみなしカウントする．τ は転置の数を表しており，文字が一致しているとみなされた 2 つの文字の位置が入れ替わっている場合にカウントされる．例えば，house と

huose という 2 つの文字列を考慮すると，o と u の位置の差は 1 文字であるため一致しているとみなされるが，文字の位置が入れ替わっている．この場合 τ をカウントする．レーベンシュタイン距離を用いた類似度とジャロ・ウィンクラー類似度を比較するとレーベンシュタイン距離は文字列全体を均一に考慮するのに対して，ジャロ・ウィンクラー類似度は先頭の文字列のマッチングを重視した手法である．なお，類似度 L_{sim} および J_{sim} の算出時には CPE 文字列の {Vender} 部分と {Product} 部分のそれぞれに対して計算し，その平均値を評価値とする．L_{sim} および J_{sim} は 0 から 1 の値をとり，CPE 文字列が一致するほど，同一の CPE である可能性が高く，L_{sim} および J_{sim} の値も大きくなる．

4.3 単語重複度

提案手法では，CPE の文字列における単語の重複度合に着目している．CPE が同一製品の場合は，ベンダ名や製品名が CPE 文字列中で重複しているケースが多い．従来手法で用いられているレーベンシュタイン距離やジャロ・ウィンクラー類似度により CPE 文字列間の類似度を算出することで対応付けすることができる．しかし，例えば CPE の {Product} 部分に対して，ベンダ名と製品名を記号（アンダースコア）で連結した文字列が入力されてしまうケースが存在するため，レーベンシュタイン距離やジャロ・ウィンクラー類似度のみを単純に使用しただけでは対応付け精度が低くなる．そこで，レーベンシュタイン距離やジャロ・ウィンクラー類似度の近似文字列比較に加えて，CPE の {Vender} および {Product} 内で表記される文字列をアンダースコアで分割し，得られた単語の一致数から算出される単語重複度 α を導入する．

CPE_i と CPE_j における単語重複度 α は (3) 式で表される．

$$\alpha = \frac{|\boldsymbol{A}_i \cap \boldsymbol{B}_j|}{max(|\boldsymbol{A}_i|, |\boldsymbol{B}_j|)} \tag{3}$$

ここで，$\boldsymbol{A}_i = \{a_{i_0}, \ldots, a_{i_M}\}$，$\boldsymbol{B}_j = \{b_{j_0}, \ldots, b_{j_N}\}$ であり，それぞれ CPE_i，CPE_j 内に含まれるユニークな単語集合を表す．M，N はそれぞれの集合の単語数である．単語重複度 α は 0 から 1 の値をとり，一致する単語が多ければ同一の CPE である可能性が高く，α の値も大きくなる．

4.4 共起度

提案手法では，脆弱性に付与される CPE の共起関係に着目している．製品間では共通する機能やソフトウェアなどが存在するため，共起する CPE の分布が類似する CPE は，同一の CPE である可能性が高いと考えられる．表記ゆれにより生じた未知の CPE は頻度が少ないため，共通する脆弱性の有無だけに基づき，CPE 間の類似度を定義することは困難である．このため，提案手法では，共通する CPE の有無と頻度に基づき個々の脆弱性の特徴ベクトルを定義し，CPE に紐づく脆弱性の特徴ベクトルを用いて，CPE 間の類似度を考慮した共起度 β を導入する．

CPE_i と CPE_j における共起度 β は (4) 式で表される．

$$\beta = \frac{1}{|\boldsymbol{X}_i||\boldsymbol{Y}_j|} \sum_{\boldsymbol{x} \in \boldsymbol{X}_i} \sum_{\boldsymbol{y} \in \boldsymbol{Y}_j} CosineSimilarity(\boldsymbol{x}, \boldsymbol{y}) \tag{4}$$

ここで，$CosineSimilarity$ はコサイン類似度を表しており，$\boldsymbol{X_i}$ および $\boldsymbol{Y_j}$ はそれぞれ CPE_i および CPE_j が付与された脆弱性から算出される特徴ベクトルの集合である．特徴ベクトルは脆弱性毎に $\{Vendor\} : \{Product\}$ 単位で正規化された CPE の頻度に基づいて算出される．例えば，脆弱性 CVE-2020-9848 には apple:ipad_os と apple:iphone_os の CPE が一つずつ付与されているが，この時 apple:ipad_os の頻度は 0.5，apple:iphone_os の頻度は 0.5 となり，その他の CPE の頻度は 0 となる．各脆弱性毎に特徴ベクトルが計算されているため，CPE_i に紐づく脆弱性の集合が得られると，CPE_i に紐づく脆弱性の特徴ベクトルの集合が求められる．CPE_i および CPE_j から得られた特徴ベクトルの集合を用いて特徴ベクトル毎のコサイン類似度の値を足し合わせ，その算術平均を求めることで共起度 β が算出される．共起度 β は 0 から 1 の値をとり，共起性の高い CPE 同士であれば β の値も大きくなる．脆弱性毎の特徴ベクトルは予め取得した脆弱性情報に基づき算出し，共起 DB に算出結果を格納している．

4.5 脆弱性類似度

提案手法では，CPE に紐づく脆弱性の類似度に着目している．脆弱性情報には脆弱性の概要を記した説明文が付加されており，通常，脆弱性情報と同時に公開される．脆弱性の説明文が類似の内容であれば，付与される CPE も類似の CPE が付与される可能性が高いと考えられる．そこで，CPE に紐づく脆弱性の説明文から類似度を求める脆弱性類似度 γ を導入する．

CPE_i と CPE_j における脆弱性類似度 γ は (5) 式で表される．

$$\gamma = CosineSimilarity(\boldsymbol{U_i}, \boldsymbol{V_j}) \tag{5}$$

ここで，$CosineSimilarity$ はコサイン類似度を表しており，CPE_i における脆弱性の特徴ベクトル $\boldsymbol{U_i}$ と CPE_j における脆弱性の特徴ベクトル $\boldsymbol{V_j}$ から求められ，0 から 1 の値をとる．また，脆弱性の特徴ベクトルは，bag-of-words モデル [12] と Tf-Idf[13] を用いて算出される．文書 d における単語 t の Tf-Idf 値の算出式を (6) 式に示す．

$$\begin{aligned} \text{Tf-Idf} &= tf * idf \\ tf &= \frac{k_{t,d}}{\sum_{s \in d} k_{s,d}} \\ idf &= \log\left(\frac{D}{df(t)} + 1\right) \end{aligned} \tag{6}$$

ここで，$k_{t,d}$ はある単語 t の文書 d 内における出現回数を表す．文書 d は過去の公開された脆弱性から該当の CPE が付与された全ての脆弱性の説明文を連結し，生成された文書である．単語ごとの出現回数を文書から得られたすべての単語の出現回数の和で割ることで tf が求められる．また，$df(t)$ はある単語 t が出現する文書数を表し，D は全文書数を表す．これら tf と idf をかけあわせることによって，ある文書において文書内での出現回数が多く，他の文書であまり出てこない単語を抽出することができる．共起度 γ は 0 から 1 の値をとり，対象の CPE が付与された脆弱性間の類似度の高ければ γ の値も大きくなる．

5. 評価実験

本章では，実験方法について述べた後，従来手法および提案手法の CPE マッチング結果について説明する．

5.1 実験方法

評価実験では有識者 3 名にて作成したデータセットを使用する．CPE 辞書に登録されていない未知の CPE である 70 件を用いて，対応付けの正答率を比較することで従来手法および提案手法の精度評価を行う．表 2 を例にすると，入力として cisco:cisco_ios が与えられたときに，CPE 辞書の中から cisco:ios と対応付けられた場合に正解と判断し，それ以外の CPE と対応付けられた場合は不正解と判断する．従来手法では，Luis らのレーベンシュタイン距離を用いた類似度による対応付け手法および Ashokkumar らのジャロ・ウィンクラー類似度を用いた対応付け手法を使用する．提案手法では，従来手法の評価値である文字列の類似度に加えて単語重複度 α，共起度 β，脆弱性類似度 γ をそれぞれ導入した場合の精度を評価する．対応付け候補を絞り込むためのレーベンシュタイン距離を用いた類似度およびジャロ・ウィンクラー類似度の閾値は 0.5 としている．また，未知の CPE が付与された脆弱性に対して，対応付け先の CPE が当該脆弱性に含まれていなかったケースをカウントすることで提案手法の有効性を評価する．未知の CPE が CPE 辞書内の CPE に対応付けされることで脆弱性に付与される CPE が更新されることになるが，対応付け先の CPE が当該脆弱性にとって未だ付与されていない新規の CPE であれば，対応付けにより脆弱性の収集が容易になると考えられる．なお，CPE 辞書は 2021 年 2 月時点で NIST 公開されている辞書 [7] を使用しており，時間差による CPE の登録漏れの影響を低減するため，脆弱性を取得した日から一定期間経過後に取得した．

5.2 結果と考察

CPE マッチングの評価結果を表 3 に示す．レーベンシュタイン距離を用いた類似度を評価値とした場合の正答率が 0.64，ジャロ・ウィンクラー類似度を評価値とした場合の正答率は 0.70 となった．これに対して，レーベンシュタイン距離を用いた類似度および単語重複度 α，共起度 β，脆弱性類似度 γ を評価値とした場合の正答率が 0.84，ジャロ・ウィンクラー類似度を用いた類似度および単語重複度 α，共起度 β，脆弱性類似度 γ を評価値とした場合の正答率が 0.86 となり，対応付け精度がそれぞれ 20%および 16%と大きく向上していることがわかる．また，提案手法 1a および提案手法 1b は，従来手法に単語重複度 α のみを組み合わせた手法であり，提案手法 2a および提案手法 2b は，従来手法に単語重複度 α と共起度 β を組み合わせた手法である．すべての評価尺度を組み合わせた提案手法 3a および 3b をみると，提案手法 1a および 1b から提案手法 2a および 2b への精度向上度合に比べて，提案手法 2a および 2b から提案手法 3a および 3b への向上度合いが大きいため，評価尺度としては脆弱性類似度が最も有効である可能性が高い．脆弱性の説明文には製品を表す単語が含まれていることも多く，bag-of-words と Tf-Idf により有用な特徴量を抽出できたことが示唆される．

次に従来手法 2 による対応付け誤りの差分を表 4 に，提案手法 3a による対応付け誤りの差分を表 5 に示す．それぞれの表には提案手法と従来手法の差を明確にするため，一

表 3 CPE マッチングの評価結果

手法	正答率
従来手法 1(L_{sim})	0.64
従来手法 2(J_{sim})	0.70
提案手法 1a(L_{sim}+α)	0.73
提案手法 1b(J_{sim}+α)	0.70
提案手法 2a(L_{sim}+α+β)	0.77
提案手法 2b(J_{sim}+α+β)	0.77
提案手法 3a(L_{sim}+α+β+γ)	**0.84**
提案手法 3b(J_{sim}+α+β+γ)	**0.86**

表 4 従来手法 2 による対応付け誤りの差分

評価対象の CPE	対応付け誤りの CPE
cisco:cisco_ios	cisco:ciscoworks
ibm:runtimes_for_java_technology	ibm:tivoli_storage_flashcopy_manager
sun:openjdk	sun:openwindows
sun:java	sun:javamail
apple:mobile_safari	apple:mail
apache_tomcat:apache_tomcat	apache:apache_test
ubuntu:ubuntu	ubports:ubuntu_touch
asterisk:asterisk	estatik:estatik
asterisk:asterisk_business_edition	asthis:universal_website_asthis
digium:asterisk_business_edition	digium:asterisk_gui
asterisk:s800i	huawei:s7800
digium:asterisk_digiumphones	digium:asterisk_gui
digium:open_source	digium:asterisknow
sgi:samba	sgi:fam

表 5 提案手法 3a による対応付け誤りの差分

評価対象の CPE	対応付けあやまりの CPE
adobe:acrobat_2017	adobe:acrobat_dc
imagemagick:libmagick_library	pythonware:python_imaging_library
microsoft:windows_2000_advanced_server	microsoft:windows_2000_terminal_services

表 6　従来手法および提案手法により CPE 数が増加した脆弱性件数

手法	脆弱性件数
従来手法 1(L_{sim})	198
従来手法 2(J_{sim})	250
提案手法 3b(L_{sim}+α+β+γ)	**268**
提案手法 3b(J_{sim}+α+β+γ)	**279**

方の手法でのみ対応付け誤りがあった CPE を示している．結果から従来手法と提案手法を比較すると，多くの CPE の対応付け誤りが改善していることがわかる．特徴的な例として，従来手法では asterisk:asterisk と estatik:estatik とが対応付けられている．asterisk はオープンソースの IP-PBX 用のソフトウェアであるが，estatik は Wordpress のプラグインである．対象の CPE は異なるソフトウェアであるが，CPE の文字列が類似してしているため高い評価値が算出され，誤った対応付け結果が得られた．一方，提案手法では，CPE の文字列類似度に加え，CPE の共起性や脆弱性の類似度を考慮しているため正しく対応付けすることができた．しかし，従来手法では正しく対応付けられているが，提案手法では誤って対応付けられた結果もいくつか見られた．提案手法では adobe:acrobat_2017 と adobe:acrobat_dc が対応付けられている．adobe の acrobat 製品は複数存在しているが，基本的には PDF ファイルを作成・編集・加工・管理するためのソフトウェアであり，脆弱性も複数の製品で指摘されることも多い．したがって，CPE の文字列の類似度の高さに加え，CPE の共起性や脆弱性の類似度も高く，誤った対応付け結果が得られた．提案手法では bag-of-words と Tf-Idf を用いた特徴ベクトルのコサイン類似度を用いて評価値を算出しているが，近年の文書分類に関する研究では，セキュリティ文書などの特定ドメインで学習された単語ベクトルと LSTM などの学習モデルを組み合わせることで良好な結果が得られることが報告されている [14]．より表現力のある特徴ベクトルを生成することでさらなる改善が可能であると考えられる．しかし，本研究ではいまだ検討できておらず今後の課題である．

最後に従来手法および提案手法を用いた対応付けにより，ユニークな CPE 数が増加した脆弱性件数を表 6 に示す．提案手法は従来手法に比べて，新規の CPE が付与された脆弱性件数が増加していることがわかる．脆弱性を自動で収集するためには，脆弱性を収集したい製品の CPE を CPE 辞書から事前に選択しておく必要がある．脆弱性情報収集の運用上，同一製品における CPE の一部が事前選択から漏れてしまう可能性もあるため，同一製品における CPE が複数付与されていた方が当該脆弱性の収集漏れを抑制できると考えられる．したがって，脆弱性情報収集に関して，提案手法は従来手法よりも優位な手法であることが示唆される．

6. まとめと今後の課題

本稿では，CPE 文字列の特徴，共起性，脆弱性の類似性に基づく評価により未知の CPE を CPE 辞書内の CPE と対応付ける CPE マッチング手法を提案した．また，セキュリティ業務経験を有する有識者により作成されたデータセットを用いた評価実験により提案手法の対応付け精度を定量的に求め，提案手法は従来手法よりも高精度に対応付けできることを示した．これにより，本来収集しなければならない脆弱性の収集漏れを抑制し，既知の脆弱性が放置されることによるセキュリティインシデントの発生リスクを低減することができる．

今後の課題としては評価値の見直しによる精度向上が挙げられる．評価モデルの改善に加えて，脆弱性の説明文のみだけでなく，セキュリティベンダなどが公開しているレポートなど脆弱性公開時の各種リファレンス情報も参照することで精度向上が期待できる [11]．また，本稿では，定量的な評価を行ったが，データセットにおける CPE の網羅性は限定的である．しかし，多大なデータセット作成コストにより現時点での拡張は難しい．提案手法を用いた脆弱性収集環境を構築し，実際に運用することで実企業における脆弱性漏れの影響などを評価するなど，本稿の評価実験とは異なる観点での評価方法を検討し，提案手法の有効性を検証していきたいと考えている．

参考文献

[1] IPA: 情報セキュリティ 10 大脅威 2021, https://www.ipa.go.jp/files/000088835.pdf (2021).

[2] NICT: NICTER 観測レポート 2020, https://www.nict.go.jp/cyber/report/NICTER_report_2020.pdf (2021).

[3] NIST: National Vulnerability Database, https://nvd.nist.gov/vuln/search.

[4] IPA: 脆弱性対策の効果的な進め方（実践編）第 2 版, https://www.ipa.go.jp/files/000071660.pdf (2019).

[5] IPA: セキュリティ設定共通化手順 SCAP 概説, https://www.ipa.go.jp/security/vuln/SCAP.html (2015).

[6] Cheikes, B., Waltermire, D. and Scarfone, K.: Common Platform Enumeration: Naming Specification Version 2.3, https://csrc.nist.gov/publications/detail/nistir/7695/final (2011).

[7] NIST: Official Common Platform Enumeration (CPE) Dictionary, https://nvd.nist.gov/products/cpe.

[8] Sanguino, L. A. B. and Uetz, R.: Software vulnerability analysis using CPE and CVE, *arXiv preprint arXiv:1705.05347* (2017).

[9] C, A., Mikami, S. and Ideguchi, K.: Semi-Automation of CPE matching for Vulnerability management, 2021 年暗号と情報セキュリティシンポジウム (SCIS2021) (2021).

[10] Fitzgerald, W. M. and Foley, S. N.: Avoiding inconsistencies in the security content automation protocol, *2013 IEEE Conference on Communications and Network Security (CNS)*, IEEE, pp. 454–461 (2013).

[11] Dong, Y., Guo, W., Chen, Y., Xing, X., Zhang, Y. and

Wang, G.: Towards the detection of inconsistencies in public security vulnerability reports, *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pp. 869–885 (2019).

[12] Zhang, Y., Jin, R. and Zhou, Z.-H.: Understanding bag-of-words model: a statistical framework, *International Journal of Machine Learning and Cybernetics*, Vol. 1, No. 1-4, pp. 43–52 (2010).

[13] Joachims, T.: A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization., Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science (1996).

[14] Xiao, L., Wang, G. and Zuo, Y.: Research on patent text classification based on word2vec and LSTM, *2018 11th International Symposium on Computational Intelligence and Design (ISCID)*, Vol. 1, IEEE, pp. 71–74 (2018).