

インパクト分析のためのオントロジーによる脆弱性情報の体系化

筒井 巧¹ 白石 善明¹ 森井 昌克¹

概要：脆弱性情報データベースには脆弱性が次々と報告され、これらはサイバー攻撃の対象となることがある。脆弱性情報データベースに公表されている5年分のデータを分析したところ、登録されている製品のうち半数以上が複数の脆弱性を持つことが確認でき、複数の脆弱性を持つ製品の47%が少なくとも1つ以上非常に危険度の高い脆弱性を持つことがわかった。また、複数の製品に共通する脆弱性も存在し、それらの65%以上が危険度の高いものであることが明らかになった。このような危険度の高い脆弱性に対して見落としなく対応するため、本研究ではオントロジーを用いて脆弱性情報の体系化に取り組み、特定のデータからオントロジーによって定義された関係性を利用することで関連する複数の脆弱性情報を取り出すことを可能としている。また既存方法と比較した結果、平均で6回程度の検索が必要であったところを1回の検索で目的の情報全てを得られている。

Systemization of Vulnerability Information by Ontology for Impact Analysis

TAKUMI TSUTSUI¹ YOSHIAKI SHIRAISHI¹ MASAKATU MORII¹

1. はじめに

ソフトウェアにセキュリティ上の欠陥である脆弱性が発見されると、その脆弱性は Common Vulnerability and Exposure 識別番号(CVE-ID)が付けられて公開され、その後 Common Vulnerability Scoring System(CVSS)という手法に基づき 0.0~10.0 の範囲で危険度が評価される。図1に示すように毎年膨大な数の脆弱性が公開されており、これらはサイバー攻撃の対象となる可能性がある。

脆弱性を悪用する攻撃の中には、特定の脆弱性を狙う攻撃のみではなく、脆弱性チェイニングと呼ばれる複数の脆弱性を組み合わせて利用する手法も確認されている。実際に Windows に関連する脆弱性である CVE-2020-1472 とその他複数の脆弱性を利用し、政府系機関への攻撃などが確認され、この攻撃の中で利用されている脆弱性の中には、同じベンダが作成している複数のソフトウェアに共通する脆弱性が利用されている。また、攻撃を受ける製品に注目すると、一つの製品が複数の脆弱性を持つ可能性がある。例えば、Ripple20 と呼ばれる脆弱性群は Treck 社のソフトウェアに関連する 19 個の脆弱性の総称であり、攻撃に利用された場合は何百万個の IoT 機器に影響を与えるとされている。

2016~2020 年度の間に National Vulnerability Database(NVD)[1]で公開された情報を分析すると、半数以

上の製品が複数の CVE を持っていることが分かった。この複数の CVE を持つ製品について CVSS の基本スコアの平均を取ったところ、約 66%の製品がスコア平均 7.0 以上であることが確認された。複数の CVE を持つ製品の内、基本スコア 9.0 以上の CVE を少なくとも一つ以上持つ製品の個数を図2に示す。複数の CVE を持つ製品の内、約 47%の製品が基本スコア 9.0 以上の CVE を少なくとも一つ以上持つことが確認された。また CVE に掲載されている製品という観点からは、複数の製品に共通する CVE は全体の 37%であり、このうち基本スコアが 7.0 以上のものは 65%を占めることが確認された。

このような実例やデータの概観から分かる通り、複数の脆弱性を同時に利用した攻撃や複数のソフトウェアに共通の脆弱性、複数の脆弱性を持つソフトウェアが存在しており、利用している製品の脆弱性に関連する情報を確認するには一つの脆弱性だけではなく、関連する複数の脆弱性とそれらのさらに関連する情報にも注目する必要がある。

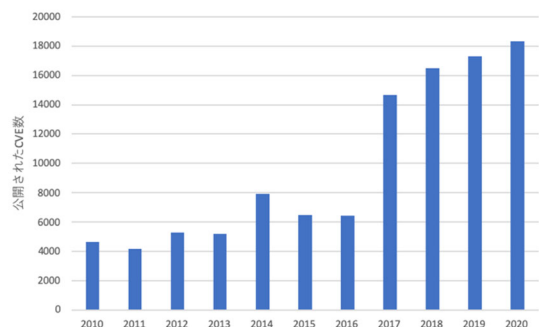


図1 National Vulnerability Database (NVD)に公開されている CVE 数。

¹ 神戸大学大学院工学研究科電気電子工学専攻
Dept. of Electrical and Electronic Engineering, Kobe University

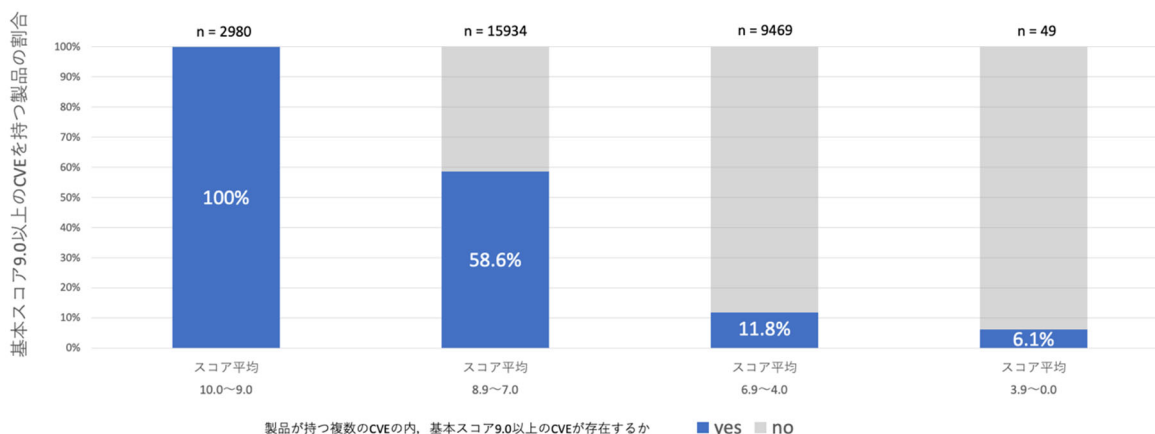


図 2 基本スコアの平均値による分布とスコア 9.0 以上の CVE を持つ製品の割合。

一般的に脆弱性に関する情報を集める方法として、Google のような汎用的検索エンジンによる情報収集や脆弱性情報データベースが提供している検索ツールの利用などが挙げられる。これらを利用し、製品名や CVE-ID などから検索を行うことで脆弱性の情報を得ることができるが、製品名から同様の脆弱性を持つ別の製品名を全て取得する検索や、CVE-ID から関連する製品が持つ別の CVE を検索するといった関連情報を取得するには複数回のページ遷移や検索のし直しなど手間がかかる場合が多く、網羅的であるかどうか判断しづらい。また、データの増加に伴い、検索の自動化などに取り組む際は、一度の検索で目的の結果を取得することが求められ、網羅的な結果は既存システムでは得られない。そこで本研究では関連する複数の脆弱性情報を取り出すことを目的として CVE とその周辺情報に対するオントロジーを構築し、それを用いて NVD から取得した脆弱性情報を体系化する。また、提案手法の評価を行うため、既存システムである NVD が提供する検索システムと体系化した情報を用いた検索システムの比較実験を行い、ケーススタディとして Ripple20 に関する分析を行う。

本稿の構成は以下の通りである。2 章では攻撃検知に関する研究について概説する。3 章では、一般的なオントロジーの定義やオントロジーを利用した研究について概説し、先行研究と関連する脆弱性情報を取得するという目的を踏まえた本研究のアプローチについて述べる。4 章では取り扱うデータである CVE とその関連情報について説明し、NVD が提供しているデータの分析結果について述べる。5 章で提案手法である脆弱性情報の体系化について述べる。6 章で提案手法を用いた検索と既存システムとの比較実験と、ケーススタディとして Ripple20 の分析を行った結果を 7 章で述べる。最後に 8 章で本稿のまとめと今後の展望について述べる。

2. 脆弱性情報の分析や検知に関する研究

脆弱性を狙った攻撃の分析や検知を目的とした研究として、攻撃の早期検知に関する研究[2,3,4,5,6]や脆弱性についての分析を行う研究[7,8,9,10]が存在する。[2]では標的型攻撃の被害にあう可能性がある SNS ユーザーを推定する分析モデルを提案している。[4]ではハッカー同士の通信に着目し悪意のあるメッセージや投稿を検出するモデルの提案をしている。また、新たな攻撃手法が構築される度に新しい語彙が作成されることに注目し、検出モデルにおけるコンセプトドリフトについての研究も行われている[11]。[3,5,6]は SNS 上の投稿から脆弱性などの脅威情報を含む投稿を検出するためのモデルの提案を行っている。このような攻撃の早期検知に関する研究は SNS のデータから単一の脆弱性や脅威情報などの情報をより素早く取得することを目的としている。

[7]は脆弱性についての議論が GitHub や Twitter, Reddit でどのように拡散するかを研究している。[8]は CVE が発表されてから CVSS スコアが付与されるまで時間がかかることに着目し、より早期にスコアを付与するため Twitter のデータから CVSS スコアを予測する手法を提案している。[9]は[8]と同様、CVE が発表されるタイミングと CVSS スコアが付与されるタイミングにギャップが存在することに注目し、Twitter のデータを用いて CVSS スコアを利用せずに CVE の悪用時期を予測するモデルを提案している。また、[10]では脆弱性がどのように発見されたかは重要であるとし、Open Source Software (OSS)のバグレポートの中で脆弱性発見戦略を含むレポートを検出するモデルを提案している。他にも関連する研究として OSS は秘密裏にセキュリティパッチを当てることがあるため、当てられたパッチからセキュリティパッチを識別する研究などが存在する[12]。これらの研究は攻撃に関する分析や、脆弱性の発見についての分析を目的としている。

3. オントロジーの利用

3.1 オントロジーの定義

オントロジーとは元々哲学用語で存在を体系的に説明する存在論を表すが情報科学の分野ではこれを借用し、「概念化の明示的な仕様」と定義される[13]. オントロジーはある事象に関連する概念に対して、他概念との関係性や型、プロパティを定義することによりデータの記述方法に規約を与えることができる。これによってデータの意味的解釈が可能となり、主にセマンティック Web や人工知能の分野で利用される。

3.2 オントロジーの利用例

オントロジーは様々な分野での利用が検討されている。例えば、データの記述方法に規約を与え、共通の語彙を規定するという点に注目し、異なる IoT デバイス間での相互運用を高めるために利用されることがある。[14]では医療分野の IoT デバイスの相互運用性を担保するために利用され、[15]ではスマートホームで利用されるシステムにおける異常検知や対応プロセスの構築を補助するためのフレームワークの一部としてオントロジーが利用されている。また教育分野での利用方法についても研究されており、[16]ではフォレンジック領域における専門性や認定・教育の定義が曖昧であるとして、それらの正しい分類を見出すためのオントロジーを作成し、カリキュラム作成などへの利用を提案している。検索対象に対してオントロジーを用いたアノテーションを付与することにより、意味論的な類似度に基づいた検索を行う研究も行われている[17]。また、オントロジーで定義した関係性などを基に論理推論を行うことができる。[18]ではデバイス、脆弱性、攻撃などの基本的なセキュリティ概念と関係性を定義するオントロジーを構築し、攻撃グラフを推論するセキュリティ評価方法を提案している。[19]では構築した攻撃オントロジーを用いて、脆弱性スキャナから確認した脆弱性について攻撃可能な手法を推論し、攻撃の評価を行っている。

このように、オントロジーは語彙や関係性などの規定による相互運用性の向上やデータの体系化、データの関連付けによる検索精度の向上、推論などに利用されている。

3.3 本研究のアプローチ

2 章であげた研究は一つの脆弱性に対する脅威情報や攻撃についての分析に関するものとなっており、本研究の目的である関連する複数の脆弱性を取り出すことは考えられていない。そこで本研究ではデータに関係性などを定義し、体系化することができるオントロジーに注目する。オントロジーの利用により、CVE と製品情報などの関連情報を関連付けて扱うことができ、特定の情報から他の情報へ辿り着くことができるようになる。また、本研究で構築するオ

ントロジーは関連する脆弱性情報を取り出すことを目的としているため、[18][19]のような攻撃をベースとしたオントロジーではなく、CVE とその関連情報に関するものとなる。

4. 利用データの詳細とデータ全体の分析

4.1 CVE とその関連情報

体系化の対象とする NVD の情報は以下のとおりである。**CVSS** CVE は公開後に分析され、CVSS スコア(Common Vulnerability Scoring System : CVSS)と呼ばれる値が与えられる。CVSS は情報システムの脆弱性に対する汎用的な評価手法であり、脆弱性の深刻度を共通の基準の下で定量的に比較するために作成された。CVSS は現在バージョン 2 とバージョン 3 が存在し、評価基準が一部異なっている。攻撃元区分や攻撃条件の複雑さなど、複数の評価基準を分析によって決定し、それをもとに深刻度である基本スコアを計算する。深刻度のスコアは 0.0~10.0 の範囲を取り、10.0~9.0 を緊急レベル、8.9~7.0 を重要レベル、6.9~4.0 を警告レベル、3.9~0.1 を注意レベルといったようにスコアに応じたレベル分けを行なっている。

CPE 脆弱性は製品に紐づく情報であるため、脆弱性対策の自動化や標準化を行う際には製品情報に関する統一された識別子が必要となる。このような考え方の下で Common Platform Enumeration(CPE)と呼ばれる構造化された命名体系が作成された。CPE 名はハードウェア、OS、アプリケーションといった製品の種別やベンダ名、製品名、バージョンなどの製品に関連する情報によって構成される。

CWE Common Weakness Enumeration(CWE)とは多種多様なソフトウェアの脆弱性を識別するため、脆弱性の種類を脆弱性タイプとして分類し、それぞれに CWE 識別子(CWE-ID)を付与して体系化したものである。例えば、クロスサイト・スクリプティング(XSS)を可能とするための脆弱性ならば CWE-79 といったような一意な識別子が与えられるため、NVD のような脆弱性データベースでの検索方法のキーワードとしても利用される。

4.2 NVD に公開されている情報の分析

本節では脆弱性データベースである NVD で 2016~2020 年に公開された情報に関する分析を行った結果について述べる。5 年間で公開された情報から、CVE を持つ製品は 55114 個、公開された製品に関連する CVE は 69929 件であることが分かった。以下の分析における基本スコアは CVSS バージョン 3 の値を扱う。

4.2.1 製品と CVE に関する分析

まず初めに製品が持つ CVE の基本スコアについて分析した。公開されている情報に含まれる製品情報の内、複数の CVE を持つ製品を取り出し、基本スコアに応じたレベ

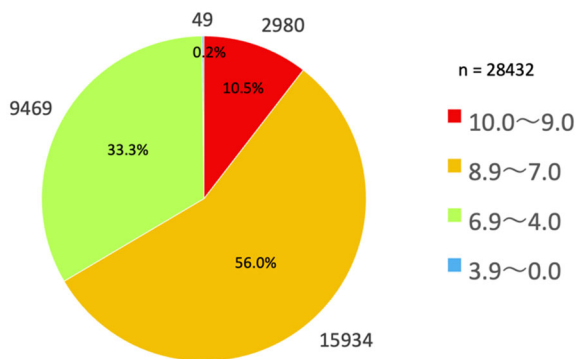


図 3 複数の CVE を持つ製品のスコア別割合。

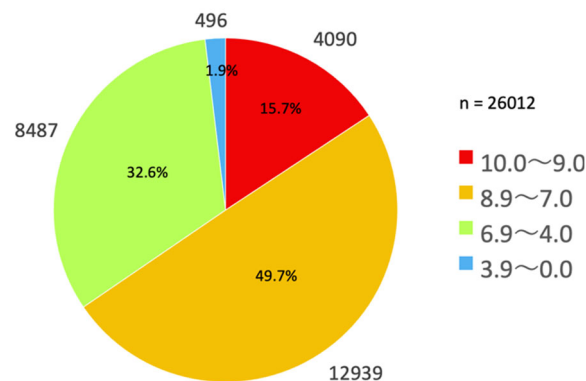


図 4 複数の製品に関連する CVE のスコア別割合

ル分けに基づいて、それぞれのレベルにおける製品数の全体に対する割合を求める。その結果を図 3 に示す。複数の CVE を持つ製品は 28432 個であり、公開されている情報に含まれる製品全体の半数以上を占めている。特に、複数の CVE を持つ製品は基本スコアの平均値 7.0 以上のものが約 66% 存在し、危険度が高い CVE を持っている可能性が高いと考えられる。また、図 2 で示したように複数の CVE を持つ製品に関して基本スコアの内訳を見ると、基本スコアが 9.0 以上である CVE を少なくとも一つ以上持っている製品は 13442 個存在することが分かり、これは複数の CVE を持つ製品の 47% 程度を占める。これらから複数の CVE を持つ製品が危険度の高い脆弱性を有する可能性は高いと言える。なお、複数の CVE を持つ製品は平均 10 件程度の CVE を持つことも分かっている。

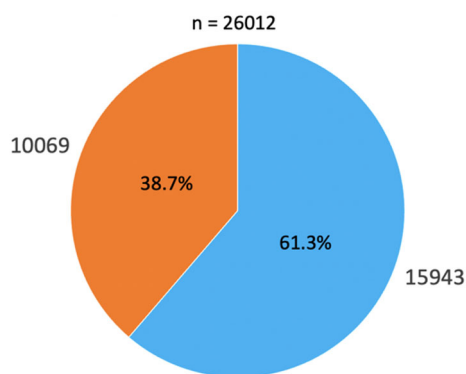
4.2.2 CVE のスコアに関する分析

5 年間で公開された CVE の内、複数の製品に関連するものを取得し、製品についての分析と同様に基本スコアの分布を調べた。結果を図 4 に示す。複数の製品に関連する CVE は 26012 件存在し、公開されている CVE の 37% であった。また、グラフから分かるようにスコア 7.0 以上のものが全体の 65% を占めている。

CVE が持つ複数の製品が同一ベンダーからリリースされたものかどうかを調べた結果を図 5 に示す。グラフより、複数の製品に関連する CVE は同じベンダーによる製品に関連することが多いことが分かる。

4.3 関連情報取得の必要性

図 6 のようなデータを考える。4.2.1 項の結果から複数の CVE を持つ製品 P1 は危険度の高い CVE を 1 つ以上持っている可能性が高いため、C2 のような CVE から製品 P1 を確認するだけでは危険度の高い脆弱性を見落とす可能性がある。また、複数の製品に関連する C1 のような CVE は 4.2.2 項の結果から危険度が高い可能性がある。ここで製品 P1 から脆弱性 C1,2 を確認するのみでは、製品 P2 は C1 と



同じベンダーからリリースされた製品のものに共通する CVE であるか yes no

図 5 同じベンダー製の製品に共通する CVE の割合。

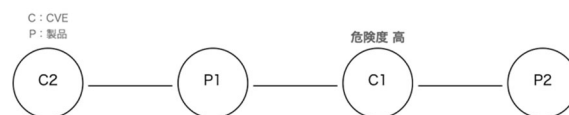


図 6 CVE と関連する製品のデータ例。

いう脆弱性を持ったままである上に危険度が高く悪用される可能性があるという危険な抜け漏れが生まれかねないということが言える。このような危険度の高い脆弱性を見落とすことを防ぐためには、図 6 のような関係において、脆弱性 C2 から P1 だけでなく C1 の情報を、そして P1 から脆弱性に関連する他の製品である P2 の情報を得ることが必要となる。

5. 提案手法

5.1 オントロジーを用いた脆弱性情報の体系化

4.3 節で述べたように、1 つの情報から関連性の最も高い情報のみを取り出すだけでは脆弱性に対して網羅的に対処することは難しいため、より広い範囲を検索できるシステムが求められる。しかし、現在公開されているデータの構造において CVE から製品情報、製品情報から CVE という

た関連性の高いデータは容易に取り出すことができるが、図 6 でいう C1, C2 のような、データ同士は直接関連していないが間接的に関係があるデータを取得することは難しい。また、毎年膨大な量の CVE が公開されることにより検索の自動化などを検討する際、検索速度という観点から見ても一度に目的とする情報を取得することが求められると考える。このような課題を解決するために、従来の検索範囲を損なわず、より幅広い検索を行うことのできるデータ構造を構築する必要がある。そこで本研究ではデータ間に関係性を定義することができるオントロジーを利用する。

本研究では CVE に関連する複数の情報について間接的に関係性が存在するものを検索できるようにし、脆弱性や脆弱性を持つ製品に対処できるよう情報を提示することを目的とする。ここで、過去に発生した攻撃の事例など多様な情報に対して関係性を定義すればそれだけ多くの情報を提示する検索が可能となるが、目的に対して冗長な関係性の定義は検索速度の低下や関連度の低い情報を取得することにつながる可能性がある。したがって、本研究では複数の脆弱性情報を取得することのみに着目し、直接的に関係があるデータ間に対して関係性を定義し、それをデータ全体に与えることでネットワーク状のデータを構築する。

このような考えをもとに構築したオントロジーの概要を図 7 に示し、各ノード間の関係性について表 1 に示す。各ノードとノードが持つプロパティについては以下の通りである。

CVE 脆弱性である CVE を表す。プロパティとして CVE-ID、公開日、最終更新日、脆弱性の概要に加え CVSS (バージョン 2,3) による各種評価を持つ。

CWE 脆弱性の種類である CWE を表す。プロパティとして CWE-ID を持つ。

Product 製品を表す。プロパティとして製品名その他、同一の製品名によるデータの混在を防ぐために”製品名_ベンダー名”という形式の一意な名称を持つ。

CPE 製品情報を表す命名体系である CPE を表す。CPE 名は複雑であり直接検索されることが少ないと考え、検索をより簡単に行えるよう、プロパティには CPE 名その他、製品名、ベンダー名、製品のバージョンを持つ。

Vendor 製品のベンダーを表す。プロパティとしてベンダー名を持つ。

このデータ構造を用いれば関連性の高いデータを取得できる他、関係性を辿ることによって直接関連していないが間接的に関係性が深いデータを取得することが可能となる。

5.2 実装とユースケース

体系化されたデータの構築にあたり、ネットワーク状のデータ構造を保存することに適したグラフデータベースである Neo4j を用いる。NVD が公開しているデータフィードのうち、2016~2020 年のデータを利用し、構築したオント

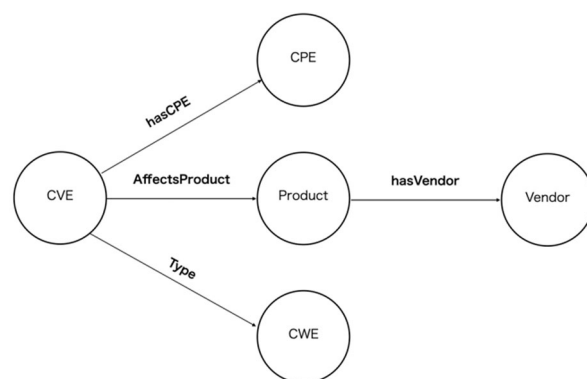


図 7 構築したオントロジーの概要

表 1 オントロジーで定義する関係性

始点ノード	関係性名	終点ノード
CVE	Type	CWE
CVE	affectsProduct	Product
CVE	hasCPE	CPE
Product	hasVendor	Vendor

ロジーを参照してデータ構造を決定することで Neo4j 内のデータを構築する。

ここで、いくつかのユースケースとクエリ例をあげて実装について述べる。以下の既存システムの例は NVD が提供する API を用いている。

ユースケース 1 特定のベンダーによる製品が持つ危険度の高い CVE の検索：あるベンダーによる製品が持つ危険度の高い CVE を取得する場合におけるクエリ例と実行結果を図 8 に示す。また、既存システムを用いて検索した場合のクエリ例と実行結果を図 9 に示す。オントロジーを用いて各ノードの関係性を定義したことで Vendor ノードから CVE ノードまでの関係性が Product ノードを経由する検索を行うことができ、またプロパティによる検索結果の絞り込みを行うことができる。例では、ベンダー「google」が持つ製品に関連する CVE の内、基本スコアが 9.0 以上のものを 3 件表示している。既存システムは図 9 に示した CVE に関する全ての情報を JSON 形式で返すが、提案システムではクエリで指定した特定のプロパティのみを出力することができる。

ユースケース 2 複数の製品に関連する CVE の検索：ある製品が持つ CVE の内、複数の製品に関連する CVE を検索する場合のクエリ例と実行結果を図 10 に示す。また、既存システムを利用した場合のクエリ例と実行結果を図 11 に示す。特定の製品から関係性が定義された CVE を経由し、別の製品を取得することができる。例では製品「android」が持つ CVE の内、複数の製品に関連する CVE と関連する製品名を 3 件取得している。

```
match (v:Vendor {name:"google"})—(p:Product)—(c:CVE)
where toInteger(c.baseScoreV3) >= 9.0
return c.name as CVE-ID, c.baseScoreV3 as BaseScore limit 3
```

CVE-ID	BaseScore
CVE-2020-8904	9.6
CVE-2020-7692	9.1
CVE-2020-7645	9.8

図 8 提案システムによる特定のベンダーの製品が持つ危険度が高い CVE の検索例。

```
https://services.nvd.nist.gov/rest/json/cves/1.0
?cvssV3Severity=CRITICAL&keyword=google&resultsPerPage=3
```

CVE-ID	BaseScore
CVE-2021-21201	9.6
CVE-2021-21226	9.6
CVE-2021-21223	9.6

図 9 既存システムによる特定のベンダーの製品が持つ危険度が高い CVE の検索例。

```
match (p:Product {product_name:"android"})—(c:CVE)—(p1:Product)
return c.name as CVE_ID, p1.product_name as Product,
c.baseScoreV3 as BaseScore limit 3
```

CVE-ID	Product	BaseScore
CVE-2020-8860	galaxy_s10	8.0
CVE-2020-6828	firefox_esr	7.5
CVE-2020-6563	backports_sle	6.5

図 10 提案システムによる複数の製品に関連する CVE の検索例。

```
https://services.nvd.nist.gov/rest/json/cves/1.0
?keyword=android&resultsPerPage=3&addOns=dictionaryCpes
```

CVE-ID	Product	BaseScore
CVE-2021-31815	google/ apple_exposer_notifications	3.3
CVE-2021-20715	hot_pepper_goument	4.3
CVE-2021-27941	chrome	6.5

図 11 既存システムによる複数の製品に関連する CVE の検索例。

ユースケース 3 複数の CVE を持つ製品の検索:ある CVE が関連する製品が持つ別の CVE を検索する場合のクエリ例と実行結果を図 12 に示す。また既存システムを用

```
match (c:CVE {name:"CVE-2020-8860"})—(p:Product)—(c1:CVE)
return c1.name as CVE_ID, p1.product_name as Product,
c1.baseScoreV3 as BaseScore limit 3
```

Product	CVE-ID	BaseScore
galaxy_s10	CVE-2019-17668	8.0
android	CVE-2020-8899	8.0
android	CVE-2020-7744	8.0

図 12 提案システムによる複数の CVE を持つ製品の検索例。

```
https://services.nvd.nist.gov/rest/json/cve/1.0/CVE-2020-8860
?resultsPerPage=5&addOns=dictionaryCpes
```

Product	CVE-ID	BaseScore
galaxy_s10	—	—
android	—	—

```
https://services.nvd.nist.gov/rest/json/cves/1.0
?keyword=android&resultsPerPage=3&addOns=dictionaryCpes
```

```
https://services.nvd.nist.gov/rest/json/cves/1.0
?keyword=galaxy_s10&resultsPerPage=3&addOns=dictionaryCpes
```

Product	CVE-ID	BaseScore
galaxy_s10	CVE-2019-17668	8.0
android	CVE-2020-8899	8.0
android	CVE-2020-7744	8.0

図 13 既存システムによる複数の CVE を持つ製品の検索例。

いた場合のクエリ例と実行結果を図 13 に示す。特定の CVE から関連する製品を経由し、その製品が持つ CVE を取得することができる。例ではユースケース 2 で出てきた CVE-2020-8860 を起点とし、それが関連する製品名と、その製品が持つ別の CVE を 3 件取得している。また既存システムでの検索は CVE-ID から検索を行い、製品名全てを取得した後、それらの製品名全てに対して再度検索を行い、同様の結果を得ている。

6. 提案手法の評価

6.1 既存システムによる検索方法

提案手法の評価を行うため、NVD が提供する検索システムとの比較を考える。これまで述べたように、NVD に公開されたデータの内、複数の製品に関連する CVE や複数の CVE を持つ製品は多く存在し、危険度が高い傾向にあるため、これらの関連する複数の製品・CVE を抜け漏れなく確認できるシステムが必要となる。また、データの増加に伴いシステムの自動化が求められる際には最小のクエリ数で

目的のデータを取得する必要がある。これらの観点から提案システムの評価を行うため、評価方法として以下の比較状況における必要クエリ数を比較する。また、各比較状況を説明するため、図 14 のようなデータを考える。P1~P3 はそれぞれ個別の製品を表し、C1、C2 は CVE を表す。ここで P1 は C1、C2 という CVE を持ち、C1 は P1 以外に P2、P3 という製品に関連しているとする。

比較状況 1 ある製品を起点として製品が持つ CVE を全て特定し、その CVE が他の製品と共通するものかどうかを判断する。また共通するならばその製品を取得する。
例：P1 は C1、C2 という CVE を持つ。この内、他の製品と共通する CVE は C1 であり、その製品は P2、P3 である。

比較状況 2 ある CVE を起点として、関連する製品名を特定し、その製品が他に CVE を持つかどうかを判断する。また他に CVE を持つならその CVE に関する情報を取得する。
例：C1 は製品 P1、P2、P3 に関連する。この内、他に CVE を持つ製品は P1 であり、その CVE は C2 である。

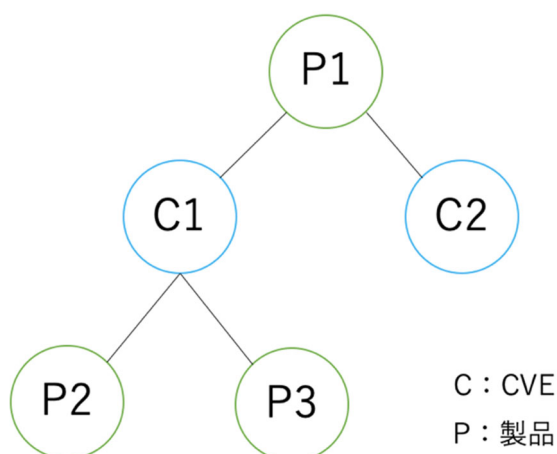


図 14 接続関係の例。

6.1.1 既存システムによる検索

NVD が提供する検索システムを用いた場合の検索について説明する。まず比較状況 1 において製品 P1 を起点として検索を行う場合について考える。既存システムと提案システムが一度のクエリで取得できるデータについて表 2 にまとめる。NVD が提供する検索システムでは P1 から検索を行うことで P1 に関連する CVE (C1、C2) を全て取得することができる。NVD が提供する API を用いることで CVE が持つ情報を JSON 形式で取得することができるため、そこに含まれる CPE から関連する製品が他にあるかどうかを確認することができる。一方、提供されているユーザーインターフェース(UI)を利用する場合、検索結果から得られる情報に製品情報は含まれないため、CVE が他の製品と関連するかどうかを調べるためには検索結果に含まれる

CVE 全ての詳細ページを確認する必要がある。これにより C1 は P1 以外の製品 P2、P3 に関連する CVE であり、C2 は P1 に固有の CVE であることが分かる。以上のように検索結果の取得、C1,2 の詳細情報の取得という合計 3 回のクエリにより、目的である C1、P2、P3 の情報を得ることができる。

表 2 比較状況 1 において一度の検索で得られるデータ

	C1	C2	P2, P3
既存システム	○	○	△
提案システム	○	○	○

次に比較状況 2 において C1 を起点として検索を行う場合について考える。既存システムと提案システムが一度のクエリで取得できるデータについて表 3 にまとめる。CVE-ID を用いた検索を行う場合、C1 に関する情報のみが取得できる。先ほどと同様に C1 の詳細情報を確認することで P1~3 の製品名を得ることができるが、それらの製品が持つ他の CVE に関する情報を得ることはできないため、P1~3 の製品名を用いて再度検索を行う必要がある。したがって、目的とする情報を得るためには C1 に関する検索の他に P1~P3 の製品名による検索を行うため合計 5 回のクエリが必要となる。

表 3 比較状況 2 において一度の検索で得られるデータ

	P1,2,3	C2
既存システム	○	×
提案システム	○	○

今回扱うデータにおいて、複数の CVE を持つ製品は平均 10 件程度の CVE を持ち、複数の製品に関連する CVE は平均 10 個の製品に関連する。比較状況において製品が持つ CVE 数や CVE が関連する製品数が増加することにより必要なクエリ数が増加するため、毎年膨大な量の CVE が公表されることも考慮すると、既存システムを用いて関連情報を取得することは非常に手間がかかると考えられる。

6.1.2 体系化したデータを用いた検索

構築したデータ構造を利用して検索を行う場合、データ間に定義された関係性を利用することで P1 が持つ CVE だけでなく、それらの CVE に関連する別の製品に関する情報も取得できる。また、C1 から検索を行う場合も同様に関連する製品が持つ全ての CVE を取得することが可能となる。それぞれの比較状況における提案システムのクエリ例をクエリ 1,2 として示す。

```
match (p1:Product {name:"製品名"})—(c:CVE)—(px:Product)
return c, px
```

```
match (c1:CVE {name:"CVE-ID"})—(p:Product)—(cx:CVE)
return p , cx
```

それぞれの比較状況における必要クエリ数を考える。必要クエリ数は製品が持つ CVE 数と CVE が関連する製品数によって計算される。それらのデータを表 4 に示す。これより、比較状況それぞれについて必要となるクエリ数を計算した結果を表 5 に示す。

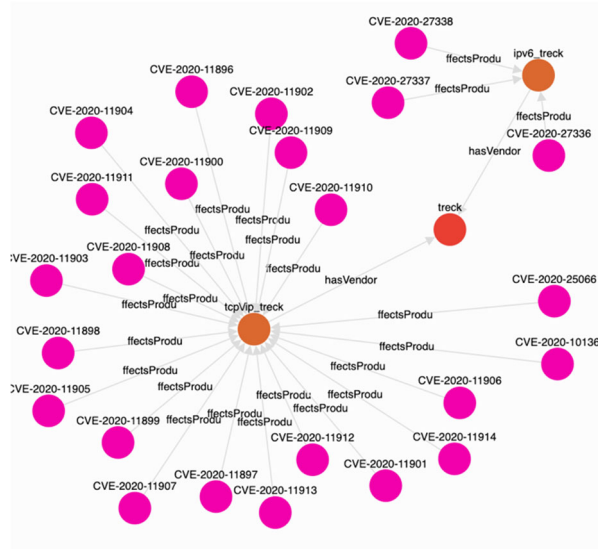
	平均値
製品が持つ CVE 数	5.9 件
CVE が関連する製品数	4.3 個

	提案システム	NVD(UI)	NVD(API)
比較状況 1	1	6.9	1
比較状況 2	1	6.3	5.3

7. ケーススタディ Ripple20 の分析

提案システムを用いた分析例として Treck 社が開発した TCP/IP スタックに関連する脆弱性群である Ripple20 の分

提案システムでは、オントロジーを利用していることによりデータ間には定義された関係性が存在するため、特定のデータを起点としそこから関係性を辿ることで目的のデータを取得することができる。ここでは Treck 社の製品に関連する CVE を全て取得するために、ベンダー「Treck」から関係性が定義された製品を経由し、それらの製品に対して関係性を持つ CVE を全て取得する。その際のクエリ例をクエリ 3 に示す。検索結果のグラフ表示を図 15 に示す。検索結果より、Treck 社の TCP/IP スタックに関連する CVE の内、Ripple20 に含まれる 19 個の脆弱性を確認できた上、Ripple20 に含まれない二つの脆弱性である CVE-2020-25066、CVE-2020-10136 を確認することができた。また、Treck 社によるその他のソフトウェアに関する脆弱性も同時に確認することができた。これより、Treck 社が開発した製品には Ripple20 以外にも対応すべき脆弱性が存在し、他の製品に対しても対応すべき脆弱性が存在することがわ



— 890 —

かる。

クエリ 3 Treck 社製の製品に関連する CVE を取得する際のクエリ例

```
match (v:Vendor {name:"treck"})-[r1]-(p:Product)-[r2]-(c:CVE)
return v, r1, p, r2, c
```

7.3 複数の製品に関連する CVE の取得

Treck 社が開発した製品が持つ脆弱性の内、複数の製品に関連する CVE が存在するかどうかを確認する。この操作で提案システムはベンダー「Treck」から関係性が定義された製品を経由し取得した CVE の内、経由していない別の製品と関連がある CVE とその製品を取得する。その際のクエリ例をクエリ 4 に、検索結果をグラフ描画したものを図 16 に示す。この結果から Treck 社による TCP/IP スタックが持つ CVE の中には、複数の他社製品に関連する CVE が存在することが分かる。これより、検索結果に含まれている製品に対しても脆弱性が存在することが確認でき、対応を行える可能性がある。また、この CVE は前節で確認された Ripple20 に含まれない脆弱性である CVE-2020-10136 であることが分かるため、Ripple20 に含まれる CVE は TCP/IP スタックに固有のものであり、TCP/IP 以外の製品に関連していた三つの脆弱性に関してもその製品固有の脆弱性であることが確認できる。

クエリ 4 Treck 社製の製品が持つ CVE の内、複数の製品に関連する CVE を取得する際のクエリ例

```
match (v:Vendor {name:"treck"})-(p1:Product)-[r1]-(c:CVE)-[r2]-(p2:Product)
return p1, r1, c, r2, p2
```

7.4 既存システムとの比較

7.2 節では提案システムを用いて Treck 社の製品が持つ全ての CVE を取得したことで Ripple20 に含まれる脆弱性が確認できる他、Treck 社の別製品に対する脆弱性も取得でき、製品に対する脆弱性の確認漏れを防ぐことができることを示した。また、7.3 節では Treck 社の製品が持つ CVE の内、他の製品と共通する CVE を抽出して取得することで、他社製品への対策漏れを防ぐことができると共に検索結果の絞りこみができていることから、確認することなく他の CVE は該当する製品固有のものであることが確認できることを示した。これらについて、既存システムと同様の結果を一度の検索で得る場合について考え、提案システムとの比較を行う。比較する結果を以下にまとめる。なお、検索は 2016~2020 年の間に NVD で公表された CVE を対象として行う。

比較 1 Treck 社の製品に関する CVE を検索した際に全ての製品に関する CVE を取得し、その検索数を比較する。

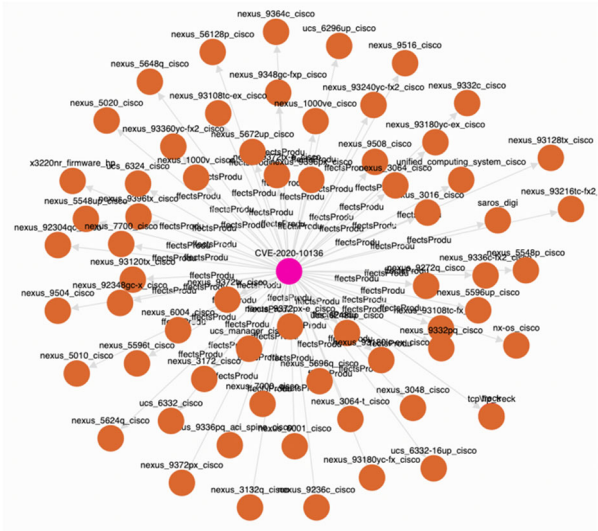


図 16 TCP/IP スタックが持つ複数の製品に関連する CVE。

比較 2 Treck 社の製品に関する CVE の内、他の製品に関連する CVE のみを抽出し、その関連製品を取得する。その際に取得できた製品数を比較する。

両比較に対する結果を表 6 に示す。比較 1 により、提案手法は既存システムと同様の結果を得ることができることが分かる。比較 2 において、既存システムでは複数の製品に関連する CVE から製品情報を直接検索する方法が存在しないため、目的の情報を一度の検索で取得することはできなかった。既存システムは CVE がどの製品に関連するかを確かめるために検索結果に含まれる CVE 全てについて詳細情報を得る必要があるのに対し、提案手法は複数の製品に関連する CVE とその製品情報を直接検索できることからより手間が少ない手法であると言える。

表 6 提案手法と NVD の検索結果。

	提案手法	NVD
比較 1	24	24
比較 2	62	-

8. まとめ

本研究では関連する脆弱性や製品情報を取得するための脆弱性情報の体系化を行った。また、構築したシステムを利用し、多数の IoT 製品に深刻な影響を与えるとされる脆弱性群である Ripple20 を分析した。結果として、構築したシステムは脆弱性データベースが提供する検索システムと同様の結果が得られる他、検索結果に含まれる脆弱性のうち、複数の製品に関連する脆弱性の抽出が行えることを示した。また、データを体系化したことによる効果を確かめるため、既存システムとの比較を行った。その結果、提案システムを用いることで一度のクエリで目的の情報を取得することができ、既存システムと比較すると検索速度を 1/6 程度に削減することが可能であることが確認できた。

本研究で構築したシステムをサイバー攻撃ハイブリッド分析プラットフォーム [20]のセキュリティ情報検索エンジンに脆弱性情報検索エンジンとして組み込むことを今後検討していく。検索エンジンとして運用していくためには自動実行可能なシステムであること、すなわち対象の情報を一回のクエリ実行で取得することが必要条件となる。既存システムでは実現できない、関連性のある製品群の抽出や、抽出されたデータ間の距離、データ間に存在する CVE を全て取得するといったより高度な検索を一度に行えるという点を活用していく予定である。

謝辞

本研究は総務省の「電波資源拡大のための研究開発 (JPJ000254)」における委託研究「電波の有効利用のための IoT マルウェア無害化／無機能化技術等に関する研究開発」によって実施した成果を含みます。

参考文献

- [1] National Institute of Standards and Technology: National Vulnerability Database(online), available from <<https://nvd.nist.gov/general>>, (accessed 2021-05-09)
- [2] Bo Feng, Qiang Li, Yuede Ji, Dong Guo, Xiangyu Meng: “Stopping the Cyberattack in the Early Stage: Assessing the Security Risks of Social Network Users”, Security and Communication Networks Volume 2019, 14pages, (2019)
- [3] A. Queiroz, Susan McKeever, Brian Keegan: “Eavesdropping Hackers: Detecting Software Vulnerability Communication on Social Media Using Text Mining”, The Fourth International Conference on Cyber-Technologies and Cyber-Systems, (2019)
- [4] A. Queiroz, Susan McKeever, Brian Keegan: “Detecting Hacker Threats: Performance of Word and Sentence Embedding Models in Identifying Hacker Communications”, 27th AIAI Irish Conference on Artificial Intelligence and Cognitive Science (AICS), (2019)
- [5] A. Queiroz, Susan McKeever, Brian Keegan: “Predicting Software Vulnerability Using Security Discussion in Social Media”, European Conference on Information Warfare and Security (ECCWS), (2017)
- [6] Ba Dung Le, Guanhua Wang, Mehwish Nasim, M. Babar: “Gathering Cyber Threat Intelligence from Twitter Using Novelty Classification”, International Conference on Cyberworlds (CW), (2019)
- [7] P. Shrestha, A.V. Sathanur, S. Maharjan, E. Saldanha, D. Arendt, S. Volkova: “Multiple social platforms reveal actionable signals for software vulnerability awareness: A study of GitHub, Twitter and Reddit”, PLoS ONE, (2020)
- [8] H. Chen, J. Liu, R. Liu, N. Park, V.S. Subrahmanian: “VASE: A Twitter-Based Vulnerability Analysis and Score Engine”, IEEE International Conference on Data Mining (ICDM), (2019)
- [9] H. Chen, R. Liu, N. Park, V. Subrahmanian: “Using Twitter to Predict When Vulnerabilities will be Exploited”, 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, (2019)
- [10] F.A. Bhuiyan, R. Shakya, A. Rahman: “Can we use software bug reports to identify vulnerability discovery strategies?”, 7th Symposium on Hot Topics in the Science of Security, (2020)
- [11] A. Queiroz, Susan McKeever, Brian Keegan: “Moving Targets: Addressing Concept Drift in Supervised Models for Hacker Communication Detection”, International Conference on Cyber Security and Protection of Digital Services, (2020)
- [12] X. Wang, K. Sun, A.L. Batcheller, S. Jajodia: “Detecting “0-Day” Vulnerability: An Empirical Study of Secret Security Patch in OSS”, 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), (2019)
- [13] T. Gruber: “A translation approach to portable ontology specifications”, Knowledge Acquisition, Vol.5, No.2, pp.199–221, (1993)
- [14] S. Jabbar, F. Ullah, S. Khalid, M. Khan, K. Han: “Semantic Interoperability in Heterogeneous IoT Infrastructure for Healthcare”, Wireless Communications and Mobile Computing Volume 2017, 10 pages, (2017)
- [15] E. Pardo, D. Espès, P. L. Parc: “A Framework for Anomaly Diagnosis in Smart Homes Based on Ontology”, Procedia Computer Science 83, pp.545 – 552, (2016)
- [16] A. Brinson, A. Robinson, M. Rogers: “A cyber forensics ontology: Creating a new approach to studying cyber forensics”, Digital Investigation, Vol.3, Supplement, September 2006, pp.37-43, (2006)
- [17] M. Fernández, I. Cantador, V. López, D. Vallet, P. Castells, E. Motta: “Semantically enhanced Information Retrieval: an ontology-based approach”, Journal of Web Semantics, Vol.9, No.4, pp.434–452, (2011)
- [18] S. Wu, Y. Zhang, W. Cao, “Network security assessment using a semantic reasoning and graph based approach”, Comput. Electr. Eng., Vol.64, pp.96–109, (2017)
- [19] J. Gao, B. Zhang, X. Chen, L. Zheng “Ontology-Based Model of Network and Computer Attacks for Security Assessment”, Journal of Shanghai Jiaotong University (Science) pp. 554-562, (2013)
- [20] T. Takahashi, Y. Umemura, C. Han, et al.: “Designing Comprehensive Cyber Threat Analysis Platform: Can We Orchestrate Analysis Engines?”, IEEE International Conference on Pervasive Computing and Communications (PerCom), (2021).