2016 年度　卒業論文

既知脆弱性データベースの調査および パターン化による再利用

2017 年 2 月 3 日(金)提出

指導教官：鷲崎弘宜 教授

早稲田大学　基幹理工学部

情報理工学科　鷲崎研究室

学籍番号：1w130153-2

鎌田　夏実

0.目次

1.はじめに

近年 IT システム・サービスが増加･複雑化するにつれ、その脆弱性に対する攻撃も増大・多様化している。脆弱性攻撃および攻撃への対策は日々行われ、CVE や ICAT などの共通機関に対して次々と報告されている。過去の事例が保存されたデータベースは現状"製品に対する脆弱性の報告の集合"にとどまり、今後起こりうる脆弱性問題およびその対策における有効な情報を得ることが難しい。

・既知脆弱性の有効な整理手段が無く、開発者側が効率よく知識を得られていない。

・既知の脆弱性問題から想定される予防策をシステム開発の中に有効に組み込めていない。

これらの問題を解決すべく、本研究では既知脆弱性データベースを用いたセキュリティ対策への有効なデータ利用方法を考察・提案する。以下に具体的な手法を述べる。

・既知脆弱性データベースとして CVE に着目し、情報を抽出・半自動で分割した後、その特徴をテキスト解析を駆使して整理し、[1]のセキュリティメタモデルとの関連付けを行う。

・分割した情報を基に既知のセキュリティパターンとの関連付けを行う。

CVE から情報の抽出・分割を行うことで、各事例に対しての原因・脅威・ユーザ・因子などといった個別の特徴をパターンとして取り出しそれぞれに固有な関連要素を見出すことができる。加えて[1]のセキュリティメタモデルへの適用により、クラウド製品に対する新規脆弱性・既知脆弱性を一貫して扱うことができ、整理した情報から特徴や対応策を得る際にデータの扱いを体系的にまとめることができる。これらによってシステムやサービスにおいて、開発段階で注意すべき脆弱性とそれに関連する要素は何かを的確に押さえること、運用段階で新規脆弱性と類似する既知脆弱性から有効な情報を得ることを可能とする。またセキュリティパターンとの関連付けにより、設計段階において危惧すべき脆弱性に対抗できるセキュリティパターンを製品に導入することを可能とする。

また本研究は[1]で開発された”クラウドサービスの開発・運用におけるセキュリティ・プライバシを一貫して扱うためのメタモデル”の実現をきっかけとしている。これにより複雑なクラウドシステムおよびそのセキュリティや脆弱性を体系的に扱うことができるようになったが、日々報告される脆弱性や対策を扱うような最新化の実現には至らなかった。このメタモデルを実践的に活用する「セキュリティ・プライバシエコシステム」[2]の実現の提案のうち、既知脆弱性データベースを用いた情報抽出・体系化へのアプローチが本研究の発端であり狙いの一つである

2.背景、先行研究

2.1 クラウドにおけるセキュリティ･プライバシを扱うための共通メタモデル

[1]において開発されたクラウドサービスの開発・運用におけるセキュリティ・プライバシを一貫して扱うためのメタモデルについて説明する。

クラウドサービスとはインターネットを介して別の場所にあるサービスや資源を利用する技術の総称である。クラウド技術によってデータを一箇所に集めて場所・時間を問わず使用することや、自分の環境に合わせて必要な分だけのリソースやサービスを利用することを可能にする。クラウドサービスはサービスの使用形態によって SaaS(サービスの提供), PaaS(プラットフォームの提供), IaaS(インフラの提供)の 3 つに分類される。

クラウドにおけるセキュリティ・プライバシ問題を扱う際、このようにクラウドサービスとそのサービスを利用するシステムがまたがっているため、通常のセキュリティ・プライバシ問題よりもより複雑になりやすい。

そこでインフラ・プラットフォーム・サービスの形態ごとにレイヤ化して扱うことで、クラウドに関するセキュリティ・プライバシ問題を共通して扱えるようにしたものが[1]で紹介されている共通メタモデルである。図 1 にメタモデルを示す。

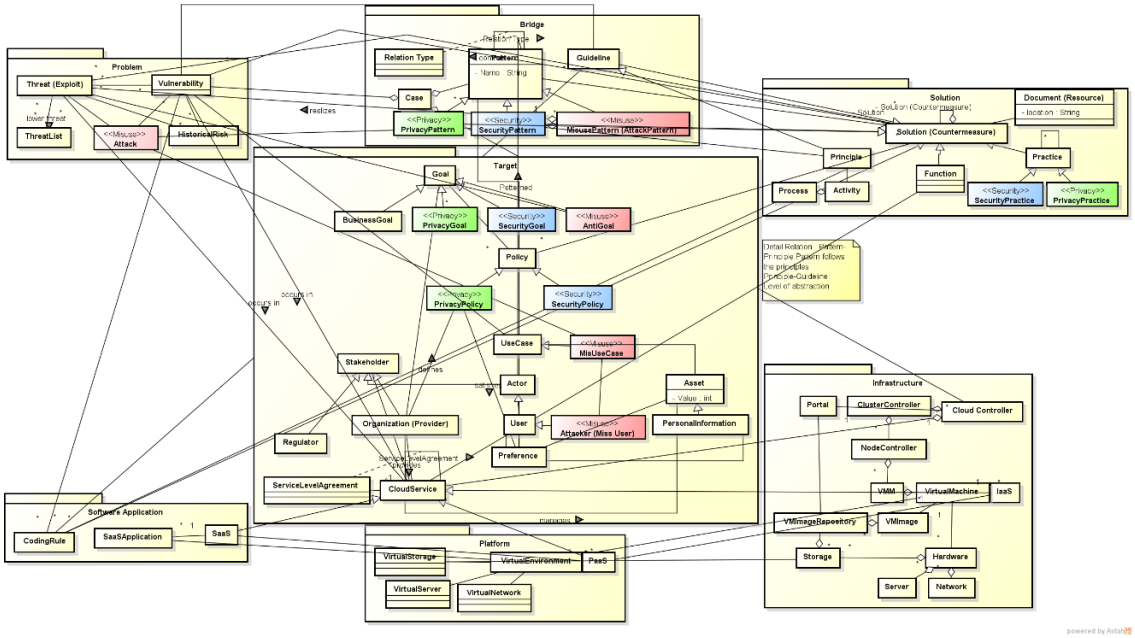


図 1　クラウドのセキュリティ・プライバシを扱う共通メタモデル

このメタモデルにはセキュリティ・プライバシを扱う上での専門概念や知識をクラスとして持たせている他、SaaS, PaaS, IaaS ごとにパッケージをまとめそれぞれを関連付けしている。これによりクラウドに特化したセキュリティ・プライバシ問題も扱うことができるようになった。

2.2 セキュリティ・プライバシエコシステム

2.1 で紹介したメタモデルを実用化するために[2]ではセキュリティ・プライバシエコシステムが提案された。システムの概念図を図 2 に示す。

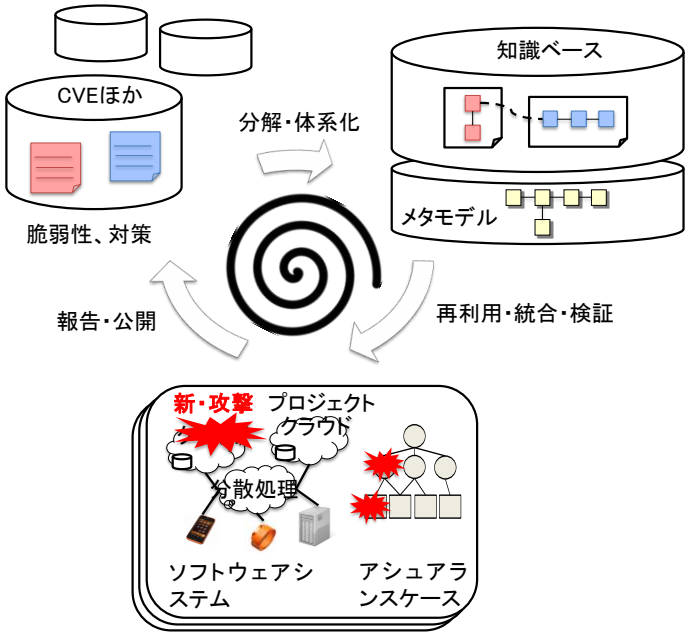


図 2　エコシステム概念図

エコシステムとは本来は生態系を意味する言葉であるが、ここでは「フローを循環させることによってシステムのセキュリティを強化していく」ことをコンセプトとしている。システムの循環は以下の通りである。

・既存の脆弱性問題やその対策法から必要なデータを抽出し、メタモデルと関連付けを行った上で知識ベースへ格納する。
・新しい脆弱性問題が発見された場合、知識ベースからその問題に関連する脆弱性問題を検索し、問題の箇所や対処法、対策を参照することで脆弱性問題の解決を効率的に行う。
・脆弱性やその対策を既存脆弱性問題として報告・登録する。

このシステムのうち、既存の脆弱性やその対策を分解・体系化し、メタモデルと関連付けした形でデータベースに格納する、という部分が本研究の発端である。
本研究では既存の脆弱性問題を扱うにあたって CVE(共通脆弱性識別子)を中心に取り上げ、データを抽出し自動で分類する方法を検討し、メタモデルとの関連付けを考察することで、メタモデルとの関連付けを行った知識ベースの開発を支援する。

2.3 セキュリティパターン

セキュリティパターンとはセキュリティ特有の専門知識を含む再利用可能なパッケージのことである[3]。パターンとはそもそも繰り返しよく使用されるもの(構造や動作、プロセスなど)の総称であり、開発フェーズに関しても要求分析から設計、実装、テスト、運用まで多岐に及ぶ。セキュリティパターンはこれにセキュリティの概念が加わり、セキュリティに精通していない開発者でもセキュリティを各工程で扱うことを容易にする。しかし[3]では多くのセキュリティパターンが開発フェーズに結びついていないのも現状とされている。本研究では既知脆弱性とセキュリティパターンとの関連付けを行うことで、設計段階における脆弱性対策としてセキュリティパターンを組み込むことを目指す。なお設計段階に注目した理由は、本調査中に扱っていた脆弱性問題が特に設計・開発段階にあると考えたためである。

2.4 既存の脆弱性データベースについて

近年では脆弱性に関する情報を広く一般に公開するデータベースがいくつか存在しており、脆弱性を取り扱う問題が数多く格納されている。ここでは脆弱性問題今回扱う脆弱性データベースについて説明する。

○CVE(Common Vulnerabilities and Exposures・共通脆弱性識別子)

既知のセキュリティ脆弱性についての識別子であり CVE がつけられた脆弱性全体を「辞書」として管理している。CVE の対象は企業や組織のセキュリティツールであり、組織内外でのデータ共有を容易にすることが可能である。[4]

CVE サイト[4]に掲載されている個々の脆弱性についての主な情報は以下の通り。

・CVE-ID : CVE 内の識別子としての番号。CVE-[データが登録された年 4 桁]-[年内での通し番号 4 桁]で表される。ここに同脆弱性を扱う NVD のページへのリンクも載せられている。

・Description：概要部分。製品の名前、ベンダー、バージョン、脆弱性の種類、想定される攻撃、攻撃に必要なアクセス、重要なコンポーネントや入力などを含む。

・Reference：参考 URL。製品を管理する組織の URL の他、別のデータベースや GitHub での実際のコード修正ページなどがある。組織・製品によって様々であるが、ここから脆弱性への対策、修正情報を得られることもある。

・Data Entry Created：データが登録された日付について。

Description については企業側の編集ではなく CVE 作成チームの MITRE により関連情報の抽出・用語の標準化を行った上で書かれている。この規則については非公開である。

○NVD(National Vulnerability Database)

SCAP(Security Contents Automation Protocol)を使用して作成される脆弱性管理データのアメリカ政府のリポジトリである[5]。CVE のリストに基づいて作成されており、CVE の

脆弱性についてのより詳細な情報が得られる[4]。CVSS(Common Vulnerability Scoring System・共通脆弱性評価システム)によって脆弱性の深刻度を評価している他、CWE(Common Weakness Enumeration・共通脆弱性タイプ)による脆弱性の分類も行っている。

○JVN(Japan Vulnerability Notes)
「日本で使用されているソフトウェアなどの脆弱性関連情報とその対策情報を提供し、情報セキュリティ対策に資することを目的とした脆弱性対策情報ポータルサイト」[6]と記述されている。CVE レポートに関して日本語で対応しており、NVD 同様の深刻度・CWE 等詳細な情報が得られる。ここで CWE は NVD のものを引用している他、文面にて別の CWE が表示されていることもあることを言及しておく。

今回は CVE の description を利用するが、その際に NVD・JVD の CWE も脆弱性の分類としてデータに加えた。この時 NVD・JVD によって CWE が異なることがあったが両方を記述の上 NVD を使用した。

3.既知脆弱性データベースからの情報抽出・整理、分析

CVE の Description は MITRE の開発チームが標準化していることもあり規則的に書かれている。本研究ではこれを利用して自然言語処理を用いて半自動で CVE の分割を行い、個々の情報について KHcoder を用いたクラスター分析を行った。そしてその結果を用いて比較し特徴を調査した。その際に一定の明確な規則を持つものはパターン化を行った。本項では始めに CVE Description の記述規則について述べ、次に CVE からのデータ抽出・分割の方法について説明する。そして KHcoder によるクラスター分析の結果を示し、その結果を交えながら特定した規則性や特徴について述べる。最後に妥当性検証という形で最新の CVE データとの比較を記述している。

3.1CVE Description の記述規則について

調査の際に発見した CVE の Description の記述規則について説明する。
CVE Description は
・主語+in または before~(+when~)+動詞(allow 以外)+which makes it easier for または allow ~ ユーザ+ to 動詞~ + via~または by~
・主語+in または before ~(+when~)+ allow ~ ユーザ + to 動詞~ + via~または by~
といった形で文章が構成されている。when は場所が異なることがある。また要素が一部省略されることも少なくない。それぞれについて説明する。

主語：脆弱性の含まれている製品、対象もしくは名称のある脆弱性”〇〇vulnerability”
in または before~：その製品に関する情報。製品名やバージョンについてなど。脆弱性の含まれている対象がこちらに含まれる場合もある。
when～：脆弱性につながる状況説明。
動詞：脆弱性の原因を示す。does not で否定が来ることもある。
which makes it easier for または allow ~ユーザ：攻撃者となりうるユーザ。認証されているか、ローカルユーザなのかなど。
to 動詞~：脆弱性によって引き起こされる攻撃、脅威の説明。
via~または by~：攻撃の際に用いられる因子。

下記に[4]から分類例を紹介する。それぞれ脆弱性の原因および対象を赤、バージョンを青、状況を灰、ユーザを緑、攻撃や脅威を黄土、攻撃の際に用いられる因子を紫で表す。

CVE-2013-0307

Cross-site scripting (XSS) vulnerability in settings.php in ownCloud before 4.0.12 and 4.5.x before 4.5.7 allows remote administrators to inject arbitrary web script or HTML via the group input field parameter.

CVE-2012-3538
Pulp in Red Hat CloudForms before 1.1 logs administrative passwords in a world-readable file, which allows local users to read pulp administrative passwords by reading production.log.

CVE-2013-1851
Incomplete blacklist vulnerability in lib/migrate.php in ownCloud before 4.0.13 and 4.5.x before 4.5.8, when the user_migrate application is enabled, allows remote authenticated users to import arbitrary files to the user's account via unspecified vectors.

CVE-2014-3661
Jenkins before 1.583 and LTS before 1.565.3 allows remote attackers to cause a denial of service (thread consumption) via vectors related to a CLI handshake.

このようにキーワードや動詞の単純な規則によって文章を分解することができる。

3.2 CVE Description の分解

規則性がある程度確認できたことから、CVE Description を規則にしたがって分解したデータを作成することを検討した。それぞれを個別に扱うことで、各ブロックにおける類似性や更なる規則の考察、およびメタモデルへの適用を容易にすることを試みる。
扱うデータについて、CVE は"cloud"という単語を含むもののうち、～2014 年のもの 257 件を使用した。この時 NVD で得た CWE の情報もそれぞれに付与し、CVE1 つにつき「CVE-ID」「CWE-ID」「description」の情報を持つようにした。上記のルールについておおよそ単語を区切りとして分解ができることから、このデータを python を用いて半自動で分割するプログラムを作成した。またプログラム作成に当たって形態素解析を行うために TreeTagger[7]を使用した。形態素解析とは、自然言語でかかれた文を言語で意味を持つ最小単位「形態素」に分割する技術であり、この際辞書の情報を参照することで単語の品詞や活用形を取得できる[8]。これを用いて活用形を揃えた後にキーワードとの照合もしくは"動詞"指定の抽出などを行っている。
今回は全体を一括に分割するプログラムと個々の要素(脆弱性、原因、状況、ユーザ、to～・想定される脅威、via~・因子)のみを抽出するプログラムを両方作成した。全体の一括分割では各 CVE に対して CVE-ID, CWE-ID, description(分割前の情報), subject(主語、対象部分), vulnerability(主語部分に名称が来る場合の脆弱性名), version, verb(vulnerability),

when, user, to~(想定される攻撃), via~の 11 項目が与えられている。
分割した結果について一部を図 3 に示した。

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	CVE-ID	CWE-ID	description	subject	vulnerability	version	verb(vulner	when~	user	to~(attack)	via~		
2	CVE-2009-	CWE-79	Cross-site scripting (>	Cross-site		in the tag cloud search script (h			allows rem	to inject ar	via unspecified vectors .		
3	CVE-2010-	NVD-CWE-	Unspecified vulnerabili	Unspecifie		in the HP MagCloud app before 1			allows rem	to read and	via unknown vectors .		
4	CVE-2011-	CWE-20	Eucalyptus	Eucalyptus		before 2.0.3 and Eucalyptus EE be			allows man	to execute	by modifying a request , related to ar		
5	CVE-2011-	CWE-20	The RPC in	The RPC implementati		in the serv	does not properly initia		allows rem	to execute	by making RPC calls that leverage in		
6	CVE-2011-	CWE-89	SQL injection vulnerat	SQL inject		in replaced-dns <repdns text="" aj			allows rem	to execute	via the active parameter . NOTE : so		
7	CVE-2012-	CVE-310	The server	The server		in Crowbar	uses weak permissions for the replaced-dns <repdns text="" production.log" /> file						
8	CVE-2012-	CWE-352	Cross-site request fo	Cross-site		in IBM Maximo Asset Managemen			allows rem	to hijack th	via unknown vectors .		
9	CVE-2012-	CWE-89	SQL injection vulnerat	SQL inject		in IBM Maximo Asset Managemen			allows rem	to execute	via unspecified vectors .		
10	CVE-2012-	CWE-89	SQL injection vulnerat	SQL inject		in IBM Maximo Asset Managemen			allows rem	to execute	via unspecified vectors .		
11	CVE-2012-	CWE-79	Cross-site scripting (>	Cross-site		in IBM Maximo Asset Managemen			allows rem	to inject ar	via unspecified vectors .		
12	CVE-2012-	CWE-89	SQL injection vulnerat	SQL inject		in IBM Maximo Asset Managemen			allows rem	to execute	via unspecified vectors .		
13	CVE-2012-	CWE-89	SQL injection vulnerat	SQL inject		in replaced-dns <repdns text="" ch			allows rem	to execute	via the passw parameter . NOTE : Sc		
14	CVE-2012-	NVD-CWE-	Unspecified vulnerabili	Unspecifie		in the Net	has unknown impact and attack vectors .						
15	CVE-2012-	CWE-Othe	Session fixation vulne	Session fix		in IBM Maximo Asset Managemen			allows rem	to hijack w	via unspecified vectors .		
16	CVE-2012-	CWE-Othe	Session fixation vulne	Session fix		in IBM Maximo Asset Managemen			allows rem	to hijack w	via unspecified vectors .		
17	CVE-2012-	CWE-200	IBM Maxim	IBM Maximo Asset Ma		in SmartCloud Control Desk , Tiv			allows rem	to obtain s	via unspecified vectors .		
18	CVE-2012-	CWE-79	Multiple cross-site scr	Multiple cr		in ownCloud before 3.0.3			allow remo	to inject ar	via (1) an arbitrary field to apps / c		
19	CVE-2012-	CWE-20	Open redirect vulnerat	Open redir		in replaced-dns <repdns text="" in			allows rem	to redirect	via a URL in the redirect_url paramet		
20	CVE-2012-	NVD-CWE-	The Iomeg	The Iomega Home Me		before 2.104 , Home Media Netw			allow remo	to read or	via unspecified vectors .		
21	CVE-2012-	CWE-287	EMC Cloud	EMC Cloud Tiering Appliance (aka CTA , formerly FMA)					allows rem	to obtain C	by sending a crafted file during the a		
22	CVE-2012-	CWE-352	Cross-site request fo	Cross-site		in ownCloud before 3.0.3			allows rem	to hijack th	via vectors involving contacts .		
23	CVE-2012-	CWE-79	Cross-site scripting (>	Cross-site		in files / ajax / replaced-dns <re			allows rem	to inject ar	via the files parameter , a different v		
24	CVE-2012-	CWE-79	Cross-site scripting (>	Cross-site		in IBM Maximo Asset Managemen			allows rem	to inject ar	via unspecified vectors .		
25	CVE-2012-	CWE-79	Cross-site scripting (>	Cross-site		in the Tivoli Process Automation			allows rem	to inject ar	via unspecified vectors .		
26	CVE-2012-	CWE-264	IBM Smart	IBM SmartCloud Control		Desk 7.5			allows rem	to bypass i	via vectors involving an expired pass		
27	CVE-2012-	CWE-79	Cross-site scripting (>	Cross-site		in IBM Maximo Asset Managemen			allows rem	to inject ar	via vectors related to a display name		
28	CVE-2012-	CWE-264	IBM Maxim	IBM Maximo Asset Ma		before 6.2.8 , 7.1 before replaced			allows rem	to gain priv	via unspecified vectors .		

図 3　CVE 分割プログラムによる分割結果

完全に自動分割を行えたわけではない。主語直後の in～が製品や対象の場合、in や before で製品情報の詳細が述べられていない場合は主語+version の部分が混ざっている。when の後に原因の動詞がくる場合もそのまま when に分類されている。また to~の動詞部分において、or や and を用いて並列に記載している場合もリストを追加せず同じ部分に格納している。

3.3KHcoder を用いたクラスター分析

分割した情報から特徴を見出しパターン化を行うため、まず KHcoder を用いて階層的クラスター分析を行った。分割を行ってからクラスタリングをすることにより、全体で分析を行うよりも各ブロックでの特徴が掴みやすくなった。この時データは一括分割によるデータのうち製品名など固有な情報しか含まない subject(主語、対象部分)、version を除いた vulnerability(主語部分に名称が来る場合の脆弱性名), verb(vulnerability), when, user, to~(想定される攻撃), via~を使用した。以下に結果を示す。

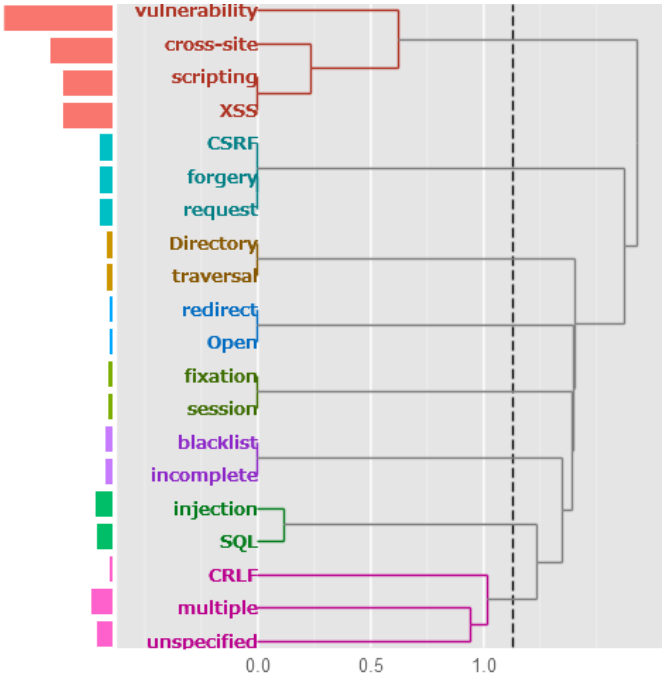


図 4　Vulnerability(主語部分に名称が来る場合の脆弱生命)のクラスター分析

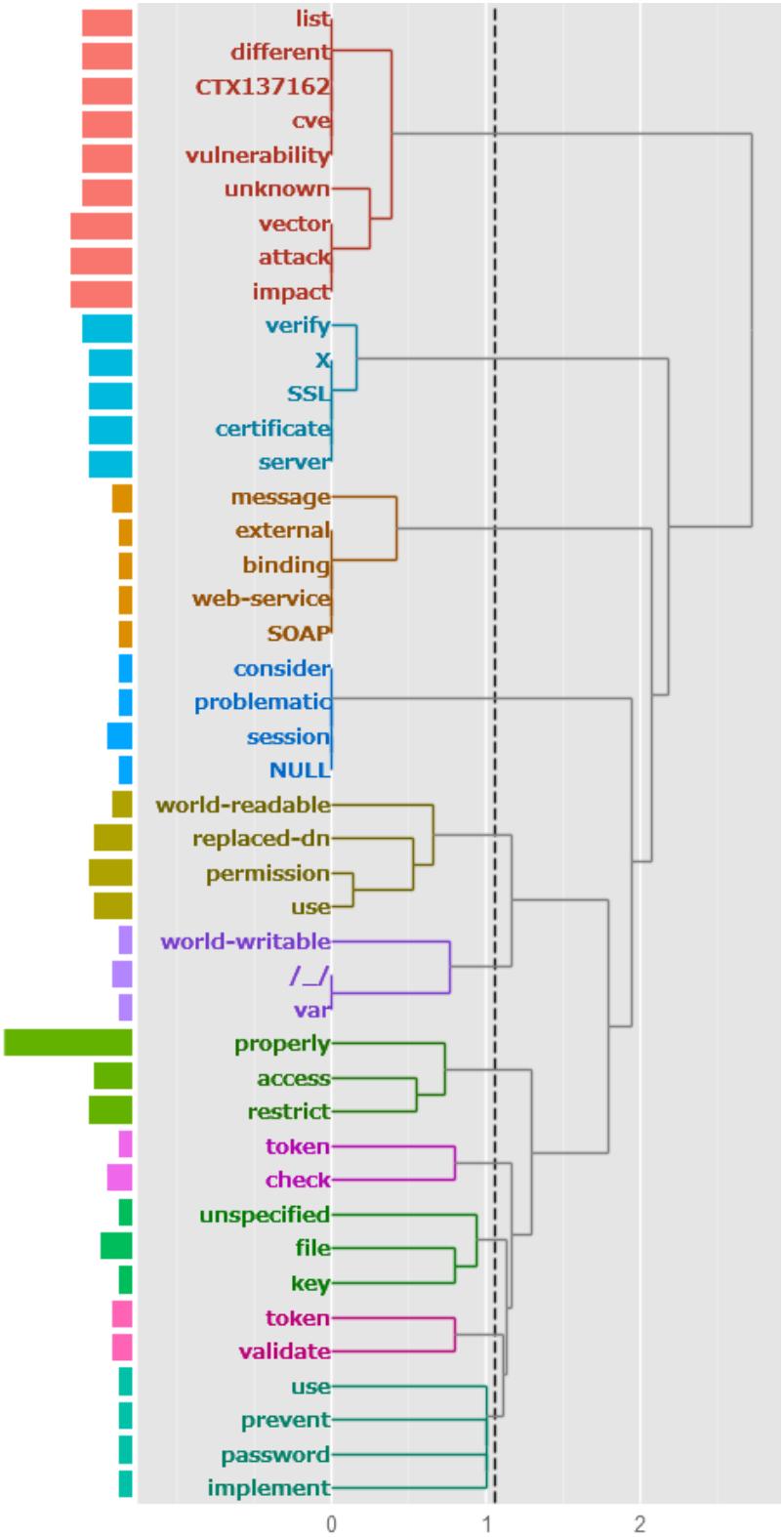


図 5 動詞で表される脆弱性のクラスター分析

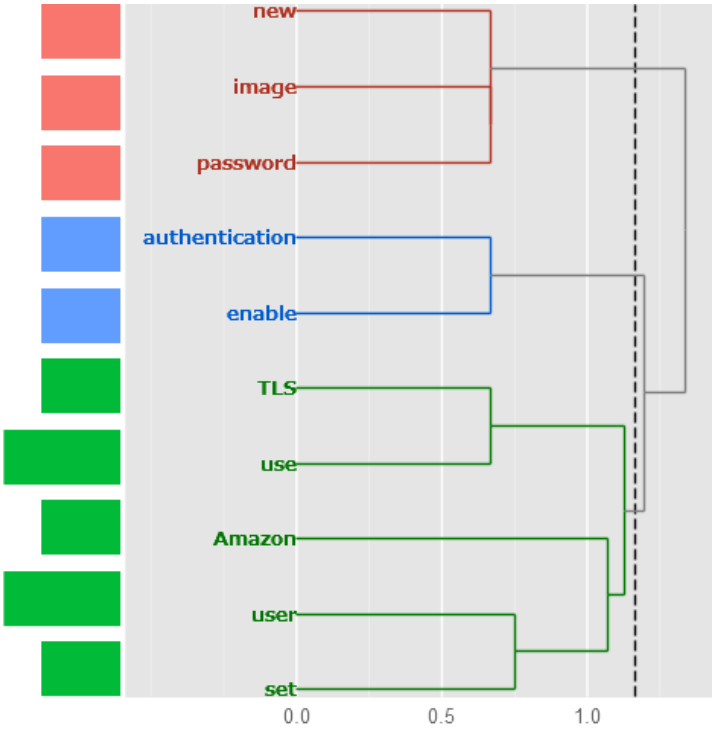


図 6 when(状況)のクラスター分析

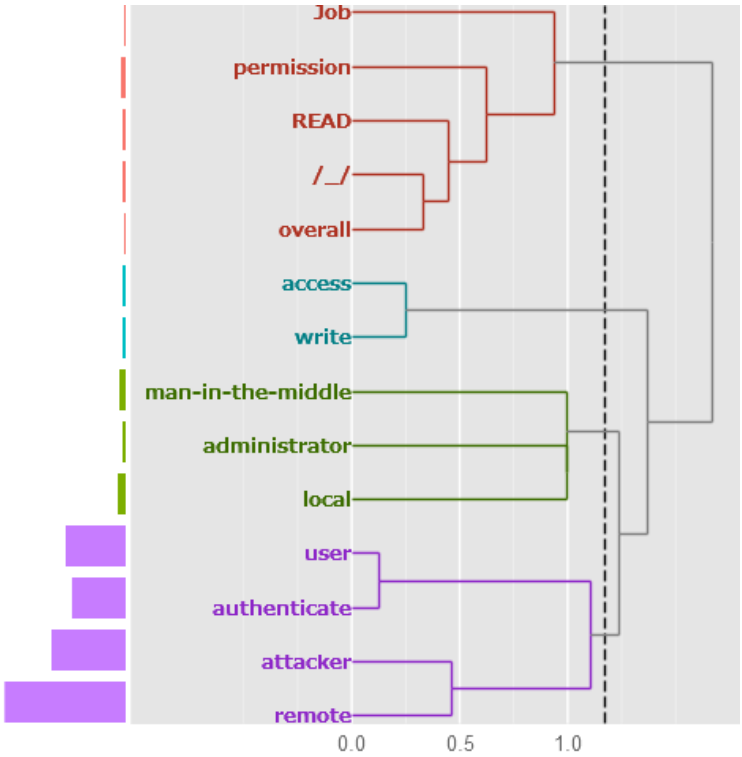


図 7 ユーザのクラスター分析

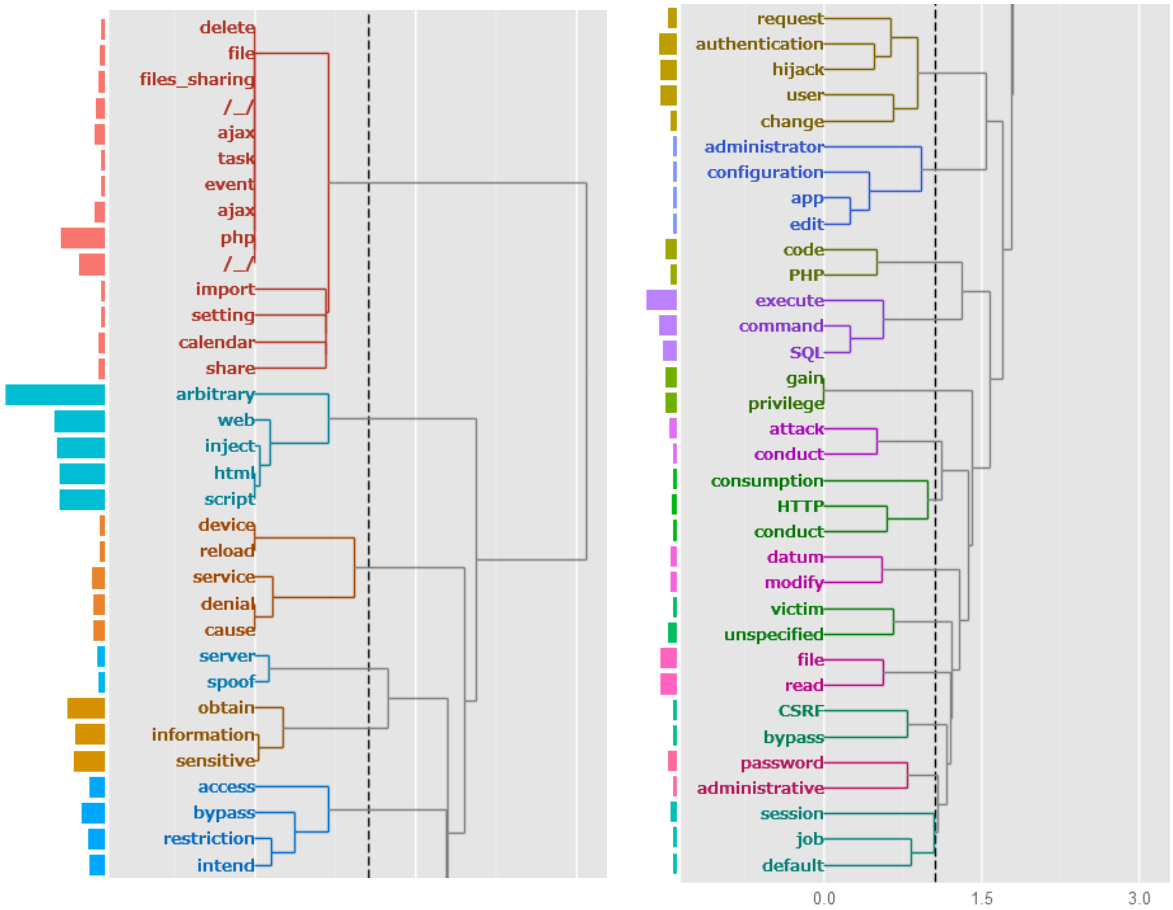


図 8･9 to ~(想定される攻撃)のクラスター分析

※縦に長いため中央で 2 分した

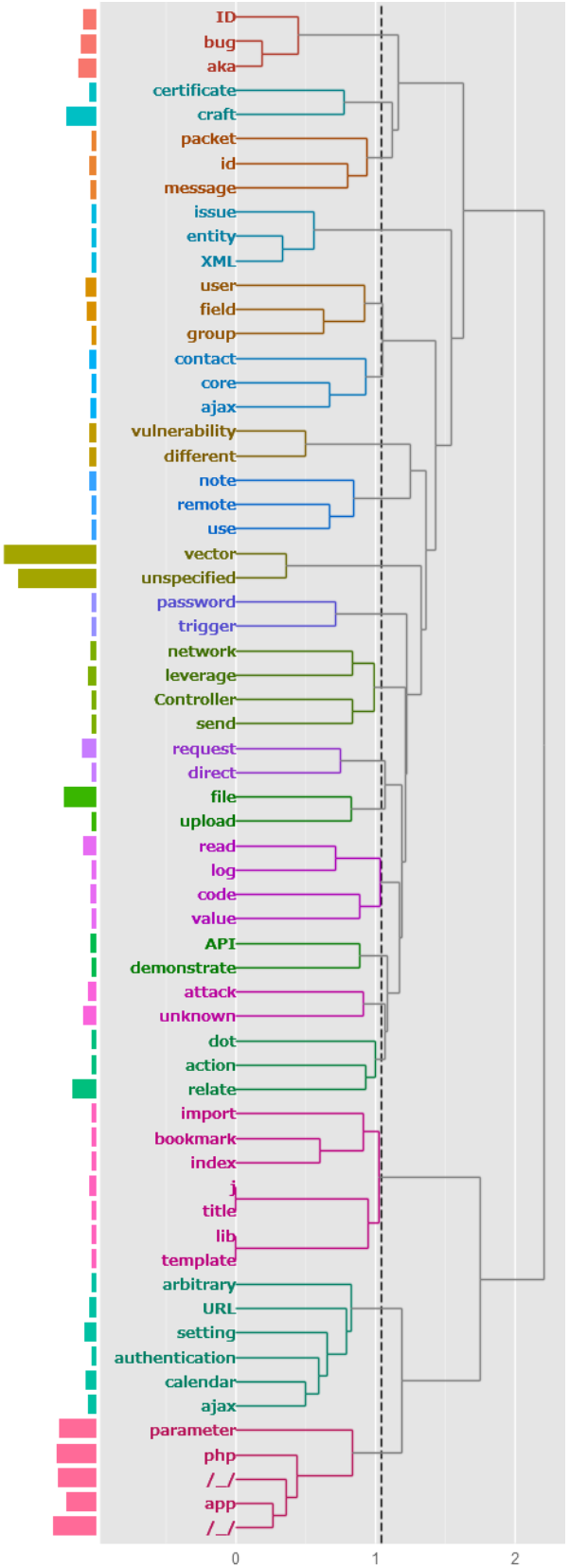


図 10 via~のクラスター分析

それぞれの結果を踏まえながら個々の特徴についてまとめる。

・vulnerability(脆弱性の名称)

クラスタリングから Cross-site scripting (XSS), request forgery (CSRF), Directory traversal, Open redirect, session fixation, Incomplete blacklist, SQL injection, CRLF, その他に分類できる。特にクロスサイトスクリプティングが多く見受けられ、次点で SQL インジェクションやクロスサイトリクエストフォージェリなども多かった。実際に出現する脆弱性の形も Cross-site scripting (XSS) vulnerability, Cross-site request forgery (CSRF) vulnerability, Directory traversal vulnerability, Open redirect vulnerability, Session fixation vulnerability, Incomplete blacklist vulnerability, SQL injection vulnerability, CRLF injection vulnerability, Unspecified vulnerability の計 9 種であった。また Multiple が先頭につく場合もある。これらは明確にパターンとして使われている。

・verb(脆弱性の原因にあたる部分)

クラスタリングの結果から、unknown vector or attack or impact, verify certificates from SSL servers, use world-readable or world-writable permission, properly restrict access (または message), validate token または file などが挙げられるが、この部分は件数の少なさ(53 件)に対して同じ製品の脆弱性の並列によってクラスタが形成されていることもあったため、やや製品に固有であったり根拠の薄い分類がなされていたりしている。例えば最初に表れている list, different, CVE, vulnerability なども a different vulnerability than other CVEs listed in CTX137162 という形で同製品の並列部分から抜き出されたものである。これを踏まえてデータを整理し主観で規則性を作成したところ、has unknown impact and attack vectors, do not verify certificates, uses ~ permissions, do not properly restrict access の 4 つとなった。

・when(状況)

when 節が最も件数が少なく、15 件しかなかったためパターンを作ることが難しい。図左の棒グラフから見て 2〜3 件でクラスタが形成されているためこの結果からは規則性が見出しづらい。ただ use, user が 3 件ずつあったことからユーザの状況、使用状況がやや多いと言うこともできる。

・ユーザ

クラスタリングから write access, authenticated users, remote attacker などが挙げられる。また man-in-the-middle, local もユーザの種類として使われている。実際にはユーザ部分は種類がかなり限られており、列挙すると local users, man-in-the-middle attackers, remote attackers または administrators, remote authenticated attackers または administrators でほとんどが構成されている。他に context-dependent attackers, user-assisted remote

attackers が 1 件含まれていた。write access に関しては remote authenticated users に対する with を用いた修飾の部分で使われていたもので、with 句は write access と~(job など)permission が使われている。

・to～(想定される攻撃)
クラスタリングから inject arbitrary web script or HTML, cause a denial service (device reload), (spoof server and) obtain sensitive information, bypass intend (access) restriction, hijack authentication ~ request ~ user change, edit app configuration, execute PHP code, execute SQL command, gain privileges, conduct attack, conduct HTTP consumption, modify data, unspecified victim, read file などが挙げられる。この時1番目のクラスタは1文においての作用する対象が多量に含まれていたため出現回数が増えているだけで実際は CVE1 つのみにしか作用していなかったため省いた。またこれ以外にも主観で hijack web session, bypass authentication などを足して、以下をパターンとする。bypass authentication, bypass intended (access) restrictions, cause a denial of service (device reload または~consumption など), conduct ~ attacks, edit app configurations, execute arbitrary (Perl または PHP など) code, execute arbitrary (SQL) commands, gain privileges, hijack the authentication of users, victims または administrators (for requests that ~), hijack web sessions, inject arbitrary HTTP headers and conduct HTTP response splitting attacks, inject arbitrary web script or HTML, modify data, obtain sensitive information, read (arbitrary など) files, redirect users to arbitrary web sites and conduct phishing attacks, spoof servers and obtain sensitive information

・via~または by~
ファイルの名前などそれぞれの製品にあわせた因子も多く、規則性がつかみにくい。
クラスタリングのうち aka Bug ID, a crafted certificate, crafted packets, XML external entity, a different vulnerability than CVE~, unspecified vectors, password which triggers ~, a direct request, uploading a file, unknown (attack) vectors などが規則として確認できた他、表記揺れしていた a brute force attack もパターンとできる。aka Bug ID, a different vulnerability than CVE~については類似あるいは関連する他の脆弱性や問題の記述に使われていた。この部分に関しては決まった表現が使われるというよりそれぞれの因子を挙げるためあまりクラスター分析には向かないと感じた。

3.4 個々のパターン間の関連性について
ここまで分割したそれぞれのブロックについて特徴をまとめパターンを見出したが、次に個々のパターン間に関連性があるかどうかを調査した。方法としては特に明確なパターンが多い「脆弱性の名称」「想定される攻撃」に対してパターン化した文字列を含む CVE 情

報を抽出し、CWE や各パターン間に規則性が見受けられるかを調査した。

脆弱性の名称
想定される攻撃が名称によって固定されているパターンが存在した。以下の通りである。
CRLF injection vulnerability - to inject arbitrary HTTP headers and conduct HTTP response splitting attacks
Cross-site request forgery (CSRF) vulnerability - to hijack the authentication of users, victims または administrators (for requests that ~)
Cross-site scripting (XSS) vulnerability - to inject arbitrary web script or HTML
Open redirect vulnerability - redirect users to arbitrary web sites and conduct phishing attacks
Session fixation vulnerability - to hijack web sessions
SQL injection vulnerability - to execute arbitrary SQL commands
Directory traversal vulnerability に関しては、固定ではないが共通して file 操作に関する攻撃が想定されていた(read, overwrite, execute など)。Unspecified vulnerability は想定される攻撃・脆弱性の原因を has unknown impact and attack vectors.とまとめている(分類は想定される攻撃ではなく脆弱性の原因としている)他は件数も少なく規則性を見出せなかった。Incomplete blacklist vulnerability は execute arbitrary file または(PHP) code の形が 5/6 件あり完全な固定ではないが密接に関連していた。

CWE に関しても固定されていることが多い。名称に応じて CWE が作られている場合もあり、ここでは CWE-22 に Directory traversal vulnerability(パス・トラバーサル), CWE-79 に Cross-site scripting (XSS) vulnerability, CWE-89 に SQL injection vulnerability, CWE-352 に Cross-site request forgery (CSRF) vulnerability が名称として割り当てられている。割り当てがない場合でも Open redirect vulnerability は CWE-20(不適切な入力表現)で固定されていた。Unspecified vulnerability はそのまま NVD-CWE-noinfo へ分類されている。CRLF injection vulnerability は CWE-20 と CWE-Other、 Session fixation vulnerability は CWE-287(不適切な認証) と CWE-Other 、 Incomplete blacklist vulnerability は CWE-94(コード・インジェクション), CWE-Other, NVD-CVE-noinfo と複数に渡っているものもあった。
なお CRLF injection vulnerability の 2 件のうち 1 つは JVD の捕捉情報にて CWE-93(CRLF インジェクション)と分類されていた。

ユーザに関しては remote attackers か remote authenticated users のおおよそどちらかという形になった。それぞれの脆弱性の全件数に対する上記どちらかの割合が 0.8 以上のものを以下に示す。

remote attackers…CRLF injection(1), Cross-site request forgery (CSRF)(0.9375), Open redirect(1), Session fixation(1)
remote authenticated users…Directory traversal(0.8), Incomplete blacklist(0.8333), SQL injection vulnerability(0.8)
XSS に関しては remote attackers が 0.588 とやや多かったが、authenticated されているか否かは問わないようであった。

via~に関してはパターンが少ないため頻出語を調べて検証した。XSS は最頻出語が unspecified vectors であり、それ以外の頻出語として parameter, php, app, file, field,が挙げられる(DF5 以上/51)。CSRF も同様に unspecified, unknown vectors が最頻出でそれ以外では parameter が頻出であった。一方で Directry traversal では unspecified, unknown 以外では via a . . (dot dot)というように脆弱性の特徴が表れているものが複数見受けられた(0.4)他、SQL injection も parameter が unspecified vector をのぞくとよく出現していた(4/15,うち unspecified が 9)。parameter の種類は不定であった。Incomplete blacklist vulnerability は by uploading a (crafited) file が多く(5/6 件)明確な関連があった。Session fixation vulnerability は unspecified vectors が、Unspecified vulnerability は unknown vectors が多く出現していた。CRLF と Open redirect vulnerability に関してはそれぞれ 2 件ずつしかないため言及に至らなかった。

○動詞パターンについて
動詞パターンについても他と関連性があるかを調査した。作成したパターンの中にはほぼ完全一致するものとそうでない抽象化されたものが 2 種類あるが、調査の際には”そのパターン内の単語がすべて含まれているもの”を抽出してその中で件数の多さから特徴付けられるものを関連とみなした。
bypass authentication…ユーザは remote attacker のみ、via~では特にパスワードやリクエストを用いて認証を解除する記述が多かった(それぞれ 2/6 件)。CWE-287 がほとんど(5/6 件)。
bypass intended (access) restrictions…ユーザは access がつくと remote authenticated user が増える(8/11 件)がつかなければ remote attacker か remote authenticated user になる。CWE-264(認可・権限・アクセス制限)に分類されやすい(12/18 件)。
cause a denial of service (device reload または~consumption など)…ユーザは remote attacker がほとんど(9/10 件)。via~には一定の規則性は見られなかった。CWE にはばらつきがあり、CWE-20 や CWE-399(リソース管理の問題)、未分類のものが存在した。なお CWE-399 についてはこの脅威にしかつけられていなかった。
conduct ~ attacks…attacks の前にクリックジャックや CSRF など多様な攻撃が置かれる。remote attacker の方が多かった(3/4 件)が攻撃によると考えられる。via にも規則性はない。

CSRF attacks が CWE-352(CSRF)であることをのぞき 3 件とも NVD-CWE-noinfo となっている

edit app configurations…2 件のみ、同製品に対するものであったため規則として妥当でない可能性がある。原因に does not properly restrict access が共通していた以外に規則性はない。

execute arbitrary (Perl または PHP など) code…6/11 件が PHP に関するものであった。また前述の通り Incomplete blacklist vulnerability が 5 件を占め、4 件が PHP code を対象としていて by a crafted file が via~部分に来る。それ以外のものには規則性はなく、CWE もばらばらであった。

execute arbitrary commands…こちらも SQL commands に関しては前述の通りである。SQL 以外のものはコマンドに指定があるものは少なかった(shell commands が 1/6 件)。ユーザに local user や man-in-the-middle attacker など頻出でないユーザの種類の出現が目立った。それ以外では規則性はなく、CWE もばらけていた。

gain privileges…ユーザに authenticated users が多い(10/12 件)。via~に unspecified vectors が多く(7/12 件)それ以外の情報があまり得られなかった。CWE は NVD-CWE-noifo3 件, CWE-255 が 1 件をのぞきすべて CWE-264 であった。

hijack the authentication of users, victims または administrators (for requests that ~)…Cross-site request forgery (CSRF)に関する記述である。前述の通りユーザは remote attacker が多く、via~は unspecified vectors, unspecified vectors が多い。CWE は CWE-352 で固定である。

hijack web sessions…全て Session fixation に関する記述である。前述の通りユーザは remote attackers、via~は unspecified vectors が多い。CWE は CWE-287 と CWE-Other に分類されている。

inject arbitrary HTTP headers and conduct HTTP response splitting attacks…全て CRLF injection vulnerability に関する記述となっている。前述の通り 2 件しかないため規則性を推定することが難しい。

inject arbitrary web script or HTML…全て XSS に関する記述である。ユーザは remote attacker がやや多いが remote authenticated users も存在し、via~は unspecified vectors が最頻出、次点で parameter php, app, file, field が多い。CWE は CWE-79 で固定。

modify data…modify data 以外の攻撃と and や or でともに記述されていることが多い(3/4 件)。情報が定まっていない時に使われやすいのか全て via~が unknown vectors, unspecified vectors であるとともに CWE も NVD-CWE-noinfo に分類されていた。

obtain sensitive information…ユーザは remote attackers と remote authenticated users がそれぞれ 10/24 件ずつで他に local users(3/24 件)や context-dependent attackers(1/24 件)を含んでいた。via~は unspecified vectors が最も多く、それ以外は code, file, packet など統一性はなかった。一定のデータの読み込み(by reading~)を行っていることが多い。

CWE-200 が最も多かった(11/25 件)他、CWE-255 や CWE-264、NVD-CWE-noinfo も多く含まれていた。
read (arbitrary など) files…ユーザはやや remote attacker が多く(8/12 件)残りは remote authenticated users であった。この中に 3 件 Directory traversal vulnerability が含まれる。via~は unspecified vectors の他 request も多く見られた(ともに 4/12 件)。CWE については Directory traversal(CWE-22)を除くと CWE-20, CWE-200, CWE-264 が含まれていた。
spoof servers and obtain sensitive information…件数は一定あったがどれも同じ企業のアンドロイド向け製品であった。そのため原因部分 does not verify X.509 certificates from SSL servers、ユーザ man-in-the-middle attackers、via~の via a crafted certificate、CWE-310 が全て一致する形となった。

3.5 特徴・パターンの妥当性検証

～2014 年までのデータを用いて規則性や特徴を分析したが、この特徴が 2016 年のものに当てはまるかを検証した。検証方法としては 2016 年の CVE から同条件でデータ分割を行う("cloud”を含むものを検索し、CWE の情報を付与した上でプログラムを用いて分割する)。分割した個々のデータが作成パターンにどの程度一致するかを dice 係数を用いた類似度判定で検証する。dice 係数とは 2 つの集合の類似度を表現する係数の 1 つであり、両者の差集合*2/積集合をとるものである[9]。差集合が大きい場合にも類似度の再を表現しやすいため、抽象化されたパターンにもある程度対応していけると見込んでこの係数用いている。dice 係数の計算部分は同サイト[9]を参照した。脆弱性名がついているもの、動詞パターンの順で関連の規則性が 2014 年までのものと一致するかどうかを(手動で)確かめる。これらの動作の際、パターンや規則性から外れているものは別個に抽出した。

〇個々のパターンとの比較
・脆弱性名称…(multiple 含め)完全一致が 12/16 件、1 つは An open redirect vulnerability と表記揺れしていた。うち 3 つは unquoted windows search path vulnerability,が 2 つ、untrusted search path vulnerability が 1 つであった。また Blacklist vulnerability, Directory traversal vulnerability, Session fixation vulnerability は出現していなかった。

・ユーザ…完全一致が 37/43 件、うち with～permission または access に含められるものが 2 つ、他に新しく attacker が 2 つ, context-dependent attacker が 1 つ出現していた。また man-in-the-middle attacker は出現していなかった。

・動詞…プログラムによって類似度が 0.7 以上とでたものが 25/43 件、人手での一致の確認

を合わせると 30/43 件となった。出現したものは以下の通り。
bypass authentication, bypass intended access restrictions, execute arbitrary code, execute arbitrary (SQL) commands, cause a denial of service, inject arbitrary web script or HTML, gain privileges, hijack the authentication of users for requests that, inject arbitrary HTTP headers and conduct HTTP response splitting attacks, obtain sensitive information, read (arbitrary) files, redirect users to arbitrary web sites and conduct phishing attacks,
残りについてはパターンを作成していないが2014年までのデータに似たものがあるものも存在した。またaffect confidentiality(, integrity, and availability)が3件あり新しくパターンを作成できる可能性があった。

・脆弱性原因
全体件数 6 件のみ。プログラムで 1 件しか発見できなかったので主観で調査した結果、has unspecified impact and (remote) attack vectors と use ~ permissions が見つかった。なお 1 件文章が 3 文に及んでいるものがあり原因の中に区別できずに残っていた。

・via~
作成したパターン数が少ないのではあるが 11/43 件は一致し、3 件は unknown vectors related to ~と unknown vectors に相当した。また a brute-force approach と表記に変化があったようだった。

○パターン間の関連性について
・個々の脆弱性名称との関連
CRLF injection vulnerability, Cross-site scripting (XSS) vulnerability, Cross-site request forgery (CSRF) vulnerabilities, SQL injection vulnerability については想定される攻撃、CWE が 2014 年での関連と一致した(CSRF は CWE-20)。ユーザも XSS・Open directory vulnerability が半々で出ている以外はそれぞれ頻出だったユーザ(remote attackers, remote authenticated users)が出ていた。via~については件数が少ないので言及がし難い。Open redirect vulnerability も想定される攻撃が 2 件一致したが、CWE が CWE-601:URL Redirection to Untrusted Site ('Open Redirect')に変更されていた。もう 1 件は文書が 3 文に分かれており、全文において形式が異なっていた。Unspecified vulnerability は動詞部分が全て to affect confidentiality(, integrity , and availability)の形で書かれており、新しく規則性が作られたと考えられる。これ以外に出現した search path vulnerability は gain privileges で想定される攻撃が一定であり、ユーザは全て local users、 via も a Trojan horse ~ in ~ directory と一致、全て同企業の製品に対して言及していた。

・想定される動詞パターンとの関連

bypass authentication…1 件のみだがユーザは remote attacker, CWE は CWE-287 に一致していた。via~には原因部分の uses cookie-encryption を引き継ぎその key のナレッジ操作とかかれていたため想定される関連とは異なった。

bypass intended ~ restriction…2 件とも access がつかなかったためユーザは authenticated ではなく remote attacker だった。CWE の分類は CWE-264 ではなく CWE-20, CWE-254 と想定される関連と異なった。

cause a denial service…ユーザは remote attacker の他に attacker も存在。CWE も CWE-200 と CWE-284 であり、via~部分にも 2 件のみのため規則性を決定付けられなかった。

execute arbitrary ~ code…3 件あったが 2014 年同様にあまり規則性が見られなかった。

execute arbitrary commands…こちらも 2 件のみであまり規則性は見られなかった。

CWE-78(OS): Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')が 1 件あったがこの時 execute arbitrary OS commands as root となっており、OS コマンドインジェクションの分類を決定付ける可能性がある。

gain privileges…ユーザに remote authenticated users、remote attackers が 1 件ずつ存在した。CWE は CWE-264 と CWE-287 の 1 件ずつで CWE-264 の方が authenticated を要した。via~に関しては規則性を決定付けられなかった。

obtain sensitive restricts…ユーザがばらけている他、via~にも統一性はなかった。データ等の読み込み(by reading~)などのほか操作も因子として挙げられている。CWE-200 が多い(6/7 件)特徴はそのままであった。

read arbitrary files…1 件のみ。ユーザは remote attackers、via~は unspecified vectors となっており規則に一致していた。CWE は NVD-CWE-noinfo となっていた。

○全体的な結果の評価

パターンに分けて考えてしまうとそれぞれの件数が少なくなり、関連性の評価が難しくなってしまった。ただし個々の表現は 2016 年度においても一定量使われていたことがわかった。2016 年度のデータを見るだけでも目新しい CWE が増えていたり新しいパターンが使われていたり(brute-force attack/approach が CWE-254 や make it easier と関連しているなど)している。また CWE 分類が過去のものと異なっていたり、新たに分類されているものもあった。

4.情報の再利用方法

4.1 セキュリティ・プライバシを扱う共通メタモデルとの関連

セキュリティ･プライバシを扱う共通メタモデルと分割した CVE との関連を考察する。CVE-2012-4394 と CVE-2014-0592 を題材にとり、メタモデルに従ってオブジェクト図を作成した。それぞれについて図 11、12 に示す。

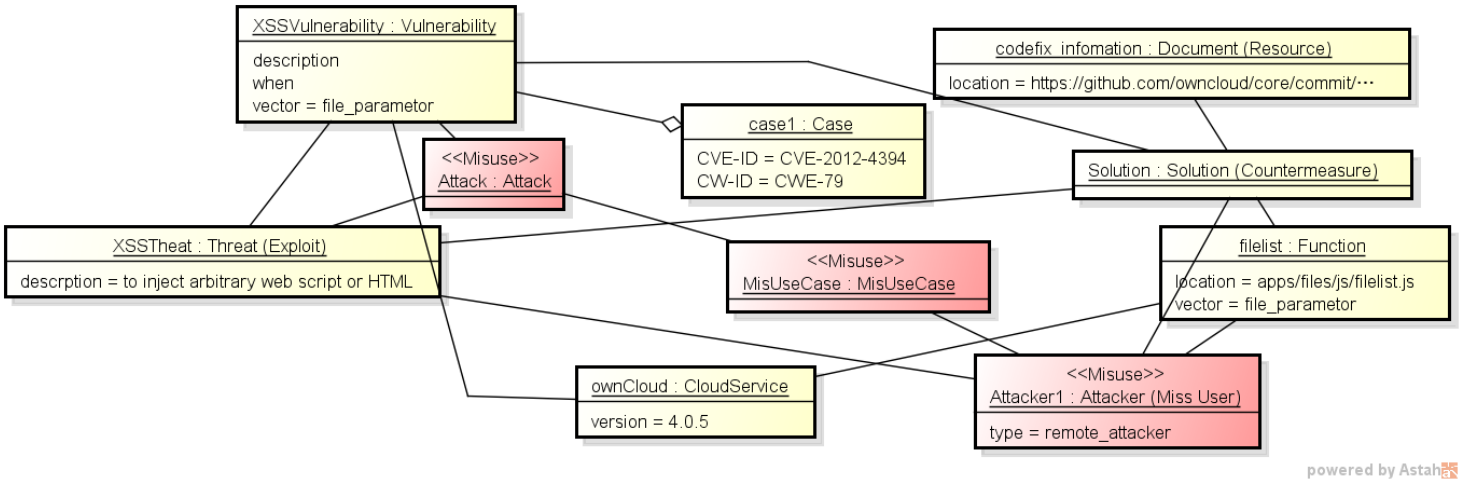


図 11　CVE-2012-4394 におけるオブジェクト図

図 12　CVE-2012-4394 におけるオブジェクト図

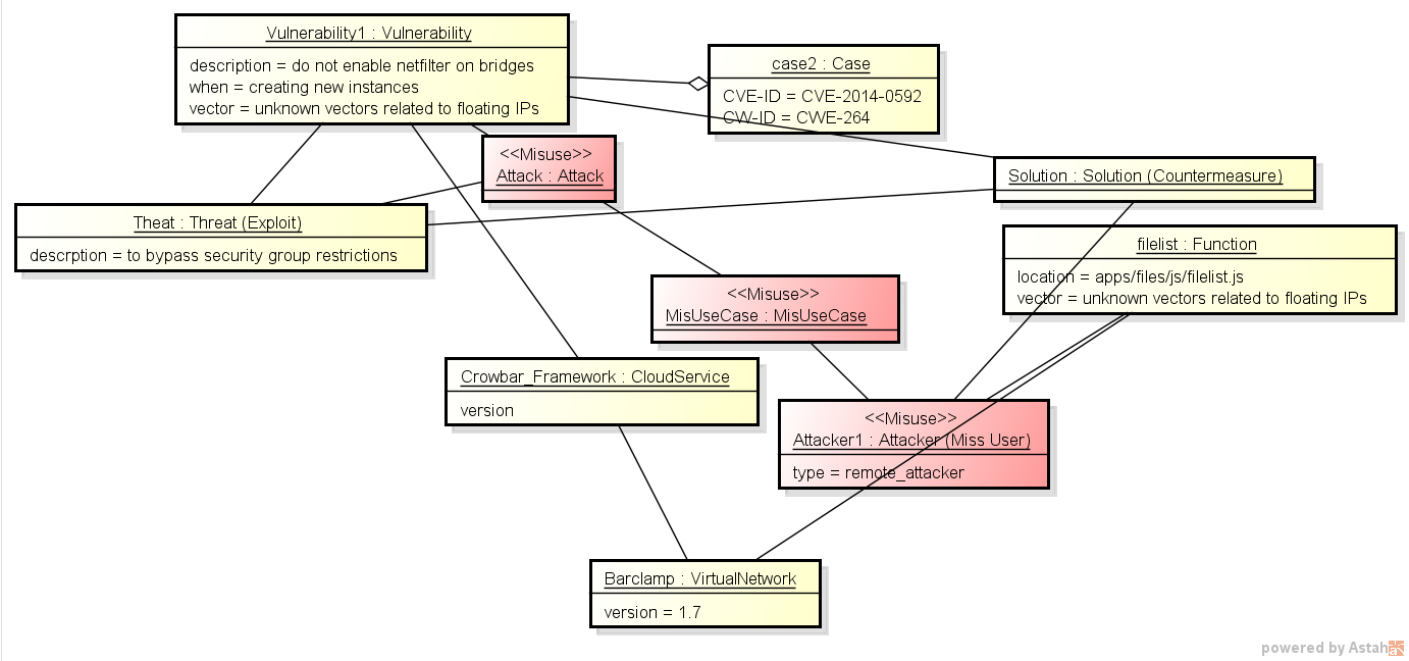
また、CVE の description は以下の通り。

CVE-2012-4394

Cross-site scripting (XSS) vulnerability in apps/files/js/filelist.js in ownCloud before 4.0.5 allows remote attackers to inject arbitrary web script or HTML via the file parameter.

CVE-2014-0592

Barclamp (aka barclamp-network) 1.7 for the Crowbar Framework, as used in SUSE Cloud 3, does not enable netfilter on bridges when creating new instances, which allows



remote attackers to bypass security group restrictions via unspecified vectors, related to floating IPs.

始めに CVE-ID と CWE-ID を case 内に格納することでそれぞれの事例を識別できるようにした。脆弱性の含まれるファイルの場所に関しては Function 内に情報を付与した。製品情報に関しては CloudService の他、後者の Barclamp はネットワークのクラウドサービスであったため VirtualNetwork としてインスタンスを生成した。脆弱性の原因については前者は名称を Vulnerability としてインスタンスを生成、後者は when における状況とともに Vulnerability へ持たせた。この時名称のついているものに関しては原因がある程度わかっており、詳細が省略されている。追加する場合は例えば CWE-79 の説明" does not neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users."[CWE サイト URL]とできる。ユーザが Attacker(Miss user)にあたるため remote_attacker とタイプ分けし、to~の想定される部分を Problem の Threat(Exploit)に、via～の因子は原因部分のクラスのインスタンスと対処後の修正された Function に持たせ、原因部分のクラスと関連付けた。また CVE の description にはないが、前者には参考 URL 部分にコードの修正情報が記載されていたためその情報を Solution 内の Document として格納した。

このように分割した CVE description をそれぞれインスタンスやクラスとしてメタモデルと関連付けして扱うことが出来る。具体的には CVE-ID,CWE-ID を case に、脆弱性名称を Vulnerability と Threat に、原因を Vulnerability の description に、ユーザを Attacker(Miss user)に、to～の想定される攻撃を Threat の description に、via～の因子を Vulnerability と Function の vector にしまう。ただし脆弱性の場所に関しては製品によって SaaS, PaaS, IaaS かどうかが異なるため特徴にあわせて分類する必要がある。

CVE から得られる情報をメタモデルに関連付けする場合に使用するメタモデルのクラスは以下の図 13 にまとめられる。

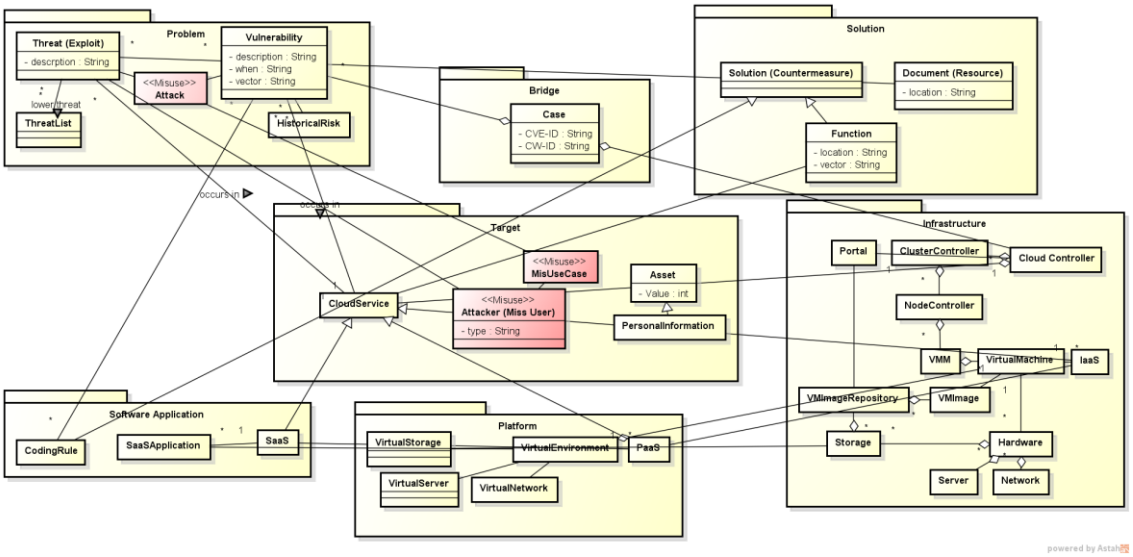


図 13 CVE を扱う上で必要なメタモデル要素

特に Bridge, Target の部分について、前者はセキュリティパターン・アタックパターンとの関連付け、後者は製品戦略などであるため情報を得ることができない。SaaS, PaaS, IaaS の部分は全 CVE の事象に対して使われる可能性として残した。また Threat, Vulnerability に対してそれぞれ ThreatList と HistricalRisk を残したが、ここに CVE の情報を蓄積させて関連付けを行う。

4.2 既存セキュリティパターンとの関連

既存セキュリティパターンと分割した CVE との関連を考察する。こちらもいくつか例をとって紹介する。

・データインジェクション系(想定される攻撃)

[10]で紹介されている Intercepting Validator パターンを設計時に組み込むことで、SQL injection や XSS ほか、”to execute ~ commands”を始めとした”データ入力に対する脆弱性攻撃”の対策につながる。Intercepting Validator はクライアント側からのリクエストに対してその処理を行う前にサーバ側でデータ検証を行うものである。このパターンによって検証プロセスを単純化して扱うことができるのである。検証プロセスにより、例えば SQL クエリを含むフォームに対しての SQL 文字の検証によって SQL コマンドの実行を回避することが可能になる。また Perl で書かれたフォームに対しては J2SE パッケージを使うことで構造を単純化しつつデータ検証を行うことも可能である。ただしこのように言語によって対応が異なってしまうため、パターンを活用するためには対応策をそれぞれで更新していく必要がある。

・Denial of service(想定される攻撃)

web サービスであれば[10]で紹介されている Intercepting Web Agent パターンまたはその中の External Policy Server Strategy によってシステムの保護を行うことができる。認証と許可のプロセスの組み込みについての設計であるが、この時稼動するサーバとは別のサーバを設け、アプリケーションの外部で認証･許可を行っている。これにより内部サービスへの DoS 攻撃を防ぐことができる他、悪意のある要求からサービスを保護することにもつなげることができる。なおこのパターンは外部に認証・許可プロセスを設けるため、運用時に脆弱性が見つかった際の保守時にも有効に用いることができる。

・Session fixation vulnerability(脆弱性名称)

そもそもの脆弱性はセッション ID を取得されてしまうことであり、セッションの追跡や盗聴によってユーザになりすまし、ホスト間通信に介入することなく攻撃が行われる。原因は多岐に及びやすく、またセッション Cookie の取得に XSS など他の手法が用いられることもある。セキュリティパターンとの関連として、例えば[11]では Security Session パターンが紹介されている。ここでは[11]で紹介されている Single Access Point のインターフェースである Check Point パターンを用いて”セッションオブジェクト”を導入することでセキュアな通信を実現している。具体的には Single Access Point でアクセスポイントを作成し、確実に全てのユーザを通過させるようにしている。セキュリティポリシーに従ってクライアントの正当性をチェックさせると同時にログインしていないユーザのアクティビティを拒否、クライアントのログイン・ログアウト時間等の情報も保持する。セッションオブジェクトはユーザ関連の共有データを保持するものであり、このセッションオブジェクト生成時に Check Point を使用しアクセスを制限・正当性チェックのカプセル化をすることで、クライアント側からセッションを追跡させない通信を実現する。
また[10]では HTTP セッションの追跡への対策に関しては J2EE アプリケーションサーバの javax.servlet.http.HTTPSession が例に挙げられている。この HTTPSession クラスを使用することで、セッションクッキーをカプセル化して扱うことや、新しいセッションの不正な作成の抑制、属性の付与、既存セッションの削除、セッションの追跡・タイムアウトなどを行うことができ、クッキーの保護やアクティブでないセッションを不正に使用させない仕様を実現する。

5.考察

2014 年までのデータからパターンや規則性をもって名称・想定される攻撃、ユーザなどが扱われていることがわかった。名称や想定される攻撃によって CWE 分類やユーザなど他の要素に特徴を持つため、新しく攻撃が起こった際に少ない手がかりからでも名称や想定される攻撃、使われる因子などの特定に役立てることができると考えられる。対策情報を参照できることが脆弱性問題の早期解決に最も有効であるが、実際には CVE の対策情報は”アップデートによって脆弱性を回避する”といったものが多く、確信的なものは企業によっては秘匿情報となっている。

CVE の description に対して分割を行ったことで、CVE の過去情報を活用する際の手助けとなる。それぞれの特徴を掴むことが容易になり、個々に特徴をまとめると同時に他の情報との関連を考察することが出来た。この分割した情報を共通メタモデルへ機械的に適用することができ、CVE のデータ参照と対策時の扱いを統一化して行うことができるようになった。同様にセキュリティパターンへもいくつか関連付けを行うことができたが、全てのデータと結びつけることは難しい。理由としては想定される攻撃単体だけでは原因が特定できない場合や、情報が不鮮明である場合が挙げられる。CVE 自体が情報量に規定が無く、ただ”何らかの攻撃を受ける可能性がある”といった報告の記述も見られるため、推察ができないだけでなく他とのデータ比較も難しいケースも少なくない。ただ原因が特定できないような脅威であっても、今回のような手法でデータベースとしての情報の蓄積によって特徴的な原因や因子を特定することができれば既知のセキュリティパターンとの関連付けを容易にすることが可能になる。

また今回の調査で古い CVE に関する情報更新の必要性や、CWE とは別の”想定される攻撃”による分類の可能性を感じた。2016 年の CVE の分割データと 2014 年までの CVE 分割データの比較の際に、2016 年の方が CWE の分類が細かく、2014 年までのデータのうち、未分類にあたる CWE-Other と NVD-CWE-noinfo に属するものは 44/257 件であったのに対し 2016 年のものは 3/45 件と脆弱性の分類がしやすい環境になっている。また Open redirect vulnerability が CWE-601 に変更されているなど CWE の分類の増加・多様化から、CVE の利用の際にはその前に過去の CVE データを CWE で再分類することが必要である。

6.終わりに

本研究では CVE を題材にとり、既知の脆弱性データベースから得た情報の再利用方法について考察した。クラウド製品に限定して抽出した CVE 文書の集合から分割を行い、頻出表現のパターン化を行いながら特徴を要素に分けて見出すことができた。そして全てではないがその要素間で固有な関連性を特定することで、クラウド製品における脆弱性の情報を簡潔に整理することができた。また分割した情報を機械的にセキュリティメタモデルと関連付けることができた。これによって整理した情報を一貫して扱うことを可能とし、システムにセキュリティ・プライバシを組み込む際の足がかりとした。セキュリティパターンとの関連は全てを網羅することができなかったものの、今後の過去の脆弱性問題の蓄積によって関連付けを狙うことができ、備えるべき脆弱性への対策として製品への導入が期待できる。以降の研究としては同手法を用いたデータ源を CVE 内または他の脆弱性データベースへ拡張することによる情報量の拡大や、[2]のエコシステムにおける知識ベースの適用・検証へ繋げること、新規のセキュリティパターンやアタックパターンへの導出への昇華が挙げられる。

7.謝辞

本研究を進めるにあたり、ご指導いただいた鷲崎弘宜教授に深く感謝いたします。また日常の議論の中で多くの示唆や助力をくださった鷲崎研究室の皆様に感謝いたします。

8.参考資料

[1]Hironori Washizaki, Sota Fukumoto, Misato Yamamoto, Masatoshi Yoshizawa, Yoshiaki Fukazawa"A Metamodel for Security and Privacy Knowledge in Cloud Services", 12th IEEE World Congress on Services (IEEE SERVICES 2016)

[2]鷲崎弘宜, “SSR 平成 28 年度 調査研究プロポーザル”, 2016 年
http://www.iisf.or.jp/SSR/prpsl_pdf_2016/p16_02_washizaki.pdf

[3]Nobukazu YOSHIOKA, Hironori WASHIZAKI, Katuhisa MARUYAMA, “A survey on security patterns”, Progress in informatics, No.5 [35-47], 2008

[4]CVE
https://cve.mitre.org/index.html

[5]NVD
https://nvd.nist.gov/

[6]JVN iPedia -脆弱性対策情報データベース
http://jvndb.jvn.jp/

[7]TreeTagger
http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

[8]日本語形態素解析エンジン・言語郎 | 形態素解析とは？
http://gengoro.zoo.co.jp/

[9]文書の類似度計算（Python3.5）: Jaccard 係数、Dice 係数、Simpson 係数
https://media.accel-brain.com/jaccard-dice-simpson/

[10]Christopher Steel, Ramesh Nagappan, Ray Lai, ”core Security Patterns Best Practices and Strategies for J2EE, Web Services, and Identity Management”, Prentice Hall, [248-251], [560-569], [606-611], 2007

[11]Marulus Schumacher, Eduardo Fernandez-Buglioni, Duane Hybertson, Franc Buschmann, Peter Sommerlad, ”SECURITY PATTERNS Integrating Security and

Systems Engineering", John Wiley & Sons, Ltd, [279-303], 2007