脆弱性情報の関連付け遷移グラフを用いた影響把握ツール ssch

神宮真人 (masato-j@is.naist.jp)

2011 年 12 月 31 日

1 概要

既知の脆弱性を悪用した不正アクセスは，提供された修正パッチを適用することで未然に防ぐことができる．しかし，自身が管理しているシステムが脆弱性により受ける影響を理解できないユーザは脆弱性に対しての危険性を認識できないため対策を行わない．そこで，システムに含まれるソフトウェアの脆弱性から脆弱性情報を関連付けることで，システムの脆弱性および危険性を認識しやすくなるのではないかと考え，発生するインパクトをトレース可能な遷移グラフを作成する ssch (Security Status Checker) を作成し，sourceforge.net にて公開した [1]．また，ssch の設計に関してはコンピュータセキュリティシンポジウム（CSS2011）[2] にて発表した．

2 はじめに

インターネットの普及に伴い，インターネットは我々の生活やビジネスの基盤となってきている．2010 年末のインターネット利用者数は 9,462 万人，人口普及率は 78.2 %となった [3]．モバイル端末や家電製品もインターネット端末となり，情報収集行動や余暇行動ではインターネットが重要な情報源と認識されている．さらに電子商取引利用の発展による経済社会の変化により，インターネットは我々の生活に不可欠な存在となっている．

一方で，システムを利用する者がその者に与えられた権限によって許された行為以外の行為をネットワークを介して意図的に行うことである不正アクセスによる被害が後を絶たない．

代表的な不正アクセスには，ソフトウェアの弱点を悪用してファイルを盗み見たり削除・改変する行為や，盗聴や総当たり攻撃によるパスワード窃取，メールサーバを悪用した迷惑メールのばらまきなどがある．情報処理推進機構（IPA）[1]による情報セキュリティ事象被害状況調査報告書 [4] によると，2009 年度のマルウェア遭遇率は 57.6%と報告されている．近年のサイバー犯罪はボットネットを中心としたアンダーグラウンドビジネスの確立が背景にあるとされており [5]，不正アクセスの被害を受けると，そのシステムの利用者だけでなく，攻撃の踏み台として悪用されることによってネットワーク上の他のユーザにも被害を与える可能性がある．

不正アクセス対策は，不正アクセスによる被害の予防，発見，及び復旧並びに拡大及び再発防止の四つの基準が挙げられる [6]．このうち，不正アクセスによる被害の予防は，不正アクセスの被害を受ける頻度を減少させるために重要な対策である．ソフトウェアの弱点である脆弱性は，攻撃の入り口として不正アクセスに悪用される．そのため，脆弱性を排除することは被害を抑えるために有効な予防対策であると言える．

既知の脆弱性を悪用した不正アクセスについては，ベンダーが提供するセキュリティ修正パッチを適用することで未然に防ぐことができる．しかし，既知の脆弱性を利用した不正アクセスによる被害が後を絶たないことから，脆弱性が公開されても対策を行わないユーザが多いことがわかっている [7]．

脆弱性対策を行わない要因は，二種類あると考えられる．一つ目の要因は，セキュリティに対する意識はあるが，ソフトウェアのアップデートに伴って発生するシステムの可用性の低下や，システムに発生する不具合を懸念し，対策を行えないことである．もう一つの要因は，脆弱性に関する情報を持っていないため，システムに発生する可能性のある影響を認知していないことである．

[1]IPA 独立行政法人 情報処理推進機構．日本のソフトウェア分野における競争力の総合的な強化を図る組織．

一つ目の要因は，ソフトウェアの安全性の維持に投資するコストと企業の利益のトレードオフによって引き起こされる要因であると考えられる．多くの企業では，業務の都合上所定の OS やソフトウェアを使用する必要があることが多く，脆弱なソフトウェアを使用せざるを得ないこともある．例えば，社内向けの Web ページやシステムが OS やブラウザに依存することがある．社内の情報を得るためのシステムとインターネット上の情報を得るためのシステムを使い分けることは，業務効率低下の原因となる可能性がある．そのため，脆弱なシステムを用いてインターネットにアクセスしている状況は多いと考えられる．また，インターネットユーザに提供しているサービスは，脆弱性の修正パッチを適用する際にサービスを停止する場合がある．サービスの停止期間中は可用性が失われるため，利益が得られないことになる．そのため，セキュリティ修正パッチの提供頻度や適用に要する時間によっては，セキュリティ修正パッチの適用を見送り，脆弱性のリスクを犠牲にすることで利益を維持する状況が発生することがあると考えられる．

二つ目の要因は，情報が不足している企業や専門知識を持たないコンシューマにおいて発生することの多い要因であると考えられる．多くの大企業のシステム管理者は，エンドポイントのマシンを次章で述べるセキュリティ製品を用いて管理していると考えられる．一方，中小企業においては，管理するマシンの数が少ないため，エンドポイントマシンのセキュリティ管理に必要なコストを削減するために手動で管理している企業が多いと考えられる．そのため，脆弱性情報を取得する時間的コストが高く，さらに必要な脆弱性情報を取得し逃す可能性があるため，本要因が発生すると考えられる．また，専門知識を持たないコンシューマは，自身のシステムに存在する脆弱性を認知していない場合が多いと考えられる．さらに，脆弱性が存在することを認知しても，それによってどのような危険性が存在するのかを理解していない場合が多いと考えられる．そのため，対策する意義を見出せず，セキュリティ対策を行わないユーザが減少しない要因の一つとなっていると考えられる．

エンタープライズとコンシューマ，及びセキュリティ意識の有無で隔てた各ユーザ層において，対策をしない理由を図 1 に示す．また，定義した各ユーザ層において有効な対策手法を図 2 に示す．図から，セキュリティ意識が低いエンタープライズに関しては，脆弱性がシステムに与える影響を認知させることが必要であることが分かる．

本研究では，低コストで脆弱性の有無とシステムの危険性を認知することが可能であることを示すことを目的とし，公開されている脆弱性情報とオープンソースのソフトウェアを用いることで，システムの危険性の認知に必要な情報の提示を実現するツールである ssch (Security Status Checker) を作成し、SourceForge.Net にて公開した．

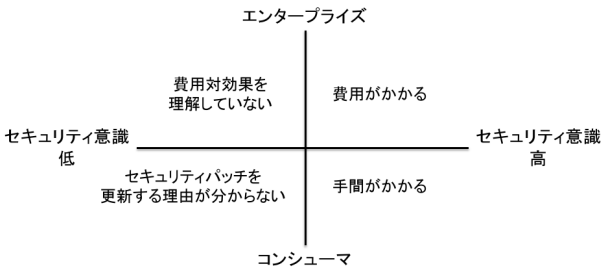

図 1: 対策をしない理由

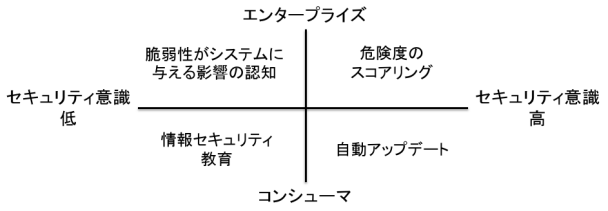

図 2: 各ユーザ層に必要な対策

3 関連技術

脆弱性の有無とシステムの危険性を認知するためには，システムに存在する脆弱性を管理する必要がある．しかし，多くのシステムは様々なベンダが提供するソフトウェアが混在している．それぞれのベンダが公開する脆弱性情報を情報源に技術的対応を行うには多くの時間と労力を必要とする．そのため，脆弱性管理の自動化を行う必要がある．

本章では，情報セキュリティ対策の自動化と標準化を実現する仕様について述べる．また，前章で述べた

各ユーザ層の対策のうち，危険度のスコアリングを試みた研究として文献 [8] がある．さらに，エンタープライズ向けのリスク管理に関連する製品について述べる．

3.1 情報セキュリティ対策の標準化を目的とした仕様

情報セキュリティにおける重要な対策の一つは，ソフトウェアの脆弱性や設定ミスを排除することである．しかし，管理対象のシステムが巨大化，及び複雑化すると，脆弱性の管理やシステムの設定ミスを排除するためには多くの時間と労力を必要とする．脆弱性の管理を自動化することは，この問題を解決する有効な手段である．そのため，脆弱性の管理を自動化することで時間と労力を削減することが求められている．

多くのシステムは，様々なベンダが提供するソフトウェアが混在しており，それぞれのベンダが公開する脆弱性情報は独自の記述形式を持っている．そのため，脆弱性管理の自動化を実現するためにはそれぞれの情報源に対して情報取得方法を定義する必要があり，新たな情報源の追加は困難である．したがって，脆弱性の検出や排除，及び設定ミスの検出や設定の修正を自動化するためには，脆弱性情報や脆弱性に関わる情報の記述ルールを標準化する必要がある．SCAP (Security Content Automation Protocol) とは，NIST[2]により開発された情報セキュリティにかかわる技術面での自動化と標準化を実現するための技術仕様である．この仕様に沿って情報セキュリティ対策の自動化を図ることで，管理コストの削減が期待できる．現在，SCAP は表 1 に示す 6 つの仕様から構成されている．

また，脆弱性に関わる脅威情報を識別するために，MITRE 社[3]が中心となり，表 2 に示す仕様が策定されている．本節では，これらの仕様について詳述する．

CVE

CVE は，既知の脆弱性情報に一意の識別番号を付与し，XML ベースで構造化したディクショナリである．米国政府の支援を受けた非営利団体の MITRE 社が採番している．また，CVE は米国政府が推進している情報セキュリティに関わる技術面での自動化と標準化を実現する技術仕様である SCAP の構成要素の一つである．CVE によって付与される識別子である CVE 識別子は，”CVE-西暦-連番”という構成で生成される．CVE 識別子には，表 3 に示す情報が付随している．

Description には，脆弱性に関する簡潔な説明が記述されている．Status には，脆弱性識別子として承認された状態である”entry”，もしくは検討中の状態である”candidate”が記述されている．Reference には，脆弱性情報の参照元である報告書や勧告の，ID や URL が記述されている．CVE の識別子をデータの提供者と利用者の両者が用いることで，共通のセキュリティ情報の交換が可能となる．

CVSS

CVSS は，脆弱性が引き起こす影響を数値化するためのスコアリングシステムである．現在は，FIRST[4]の CVSS-SIG(Special Interest Group) で適用推進や仕様改善が行われている．また，CVSS は米国政府が推進している情報セキュリティに関わる技術面での自動化と標準化を実現する技術仕様である SCAP の構成要素の一つである．情報システムの脆弱性に対するオープンで汎用的な評価手順であり，共通の評価方法を提供する．CVSS を用いることで，脆弱性の深刻度を同一の基準の下で定量的に比較することが可能となる．CVSS は表 4 に示す要素の影響度を考慮して評価している．

Base Metrics では，脆弱性そのものの特性を評価する基準である．情報システムに求められる 3 つのセキュリティ特性である機密性，完全性，可用性に対する影響を評価し，CVSS 基本値を算出する．この値は，ベンダや脆弱性を公表する組織などが，脆弱性の固有の深刻度を表すために評価する基準となる．Temporal Metric では，脆弱性の現在の深刻度を評価する基準である．攻撃コードや対策の有無や情報の信頼性を評価し，CVSS 現状値を算出する．この値は，ベンダや脆弱性を公表する組織などが，脆弱性の現状を表すために評価する基準となる．Emviromental Metrics は最終的な脆弱性の深刻度を評価する基準である．攻撃を受

[2]National Institute of Standards and Technology : (米国) 標準技術局．連邦政府の機関であり，工業技術の標準化を支援している．

[3]MITRE Corporation．米国政府向けの技術支援や研究開発を行う非営利組織．

[4]FIRST : Forum of Incident Response and Security Teams. インシデント対策組織の国際的な連合体

仕様名
CVE (Common Vulnerabilities and Exposures)[9]
CVSS (Common Vulnerability Scoring System)[10]
CCE (Common Configuration Enumeration)[11]
CPE (Common Platform Enumeration)[12]
XCCDF (eXtensible Configuration Checklist Description Format)[13]
OVAL (Open Vulnerability and Assessment Language)[14]

表 1: SCAP を構成する 6 つの仕様

仕様名
CWE (Common Weakness Enumeration)[15]
CAPEC (Common Attack Pattern Enumeration and Classification)[16]

表 2: 標準化を目指した仕様

要素名	概要
Description	脆弱性に関する概要
Status	脆弱性情報の状態
Reference	脆弱性に関連する参照情報

表 3: CVE 識別子に付随する情報

要素名	概要
Base Metrics	脆弱性自体の特性など
Temporal Metric	攻撃コードや対策の有無や，情報の信頼性など
Environmental Metrics	対象システム範囲など

表 4: CVSS において影響度を評価するための要素

けた際の二次的な被害の大きさや，組織での対象製品の使用状況を評価し，CVSS 環境値を算出する．この値は，製品利用者が脆弱性への対応を決めるために評価する基準となる．

CCE

CCE は，コンピュータのセキュリティ設定項目に一意の番号を付与する仕様である．MITRE 社が中心となり仕様が策定された．また，CCE は米国政府が推進している情報セキュリティに関わる技術面での自動化と標準化を実現する技術仕様である SCAP の構成要素の一つである．CCE は，セキュリティ設定項目に問題のない設定が施されているかどうかをチェックするためのチェックリストとして利用することができる．CCE は，セキュリティ設定項目に一意の番号である CCE 識別番号を付与する．CCE によって付与される識別番号である CCE 識別番号は，”CCE-番号-チェック番号”という構成で生成される．セキュリティ設定項目には，ユーザの権限，監査とアカウントのポリシー，ネットワーク・サービス等がある．現在は，表 5 に示すアプリケーションや OS のセキュリティ設定項目に CCE 識別番号が割り当てられている．

アプリケーション及び OS 名
AIX 5.3
HP-UX 11.23
Internet Explorer 7/8
Microsoft Office 2007/2010
Red Hat Enterprise Linux 4/5
Sun Solaris 8/9/10
Oracle WebLogic Server 11g
Windows 2000/XP/Vista/7
Windows Server 2003/2008/2008 R2

表 5: CCE 識別番号が割り当てられているアプリケーション及び OS

CPE

CPE では，ハードウェア，オペレーションシステム，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に，構造化された名称体系を規定している．CPE は，MITRE 社が中心となり仕様が策定された．また，CPE は米国政府が推進している情報セキュリティに関わる技術面での自動化と標準化を実現する技術仕様である SCAP の構成要素の一つである．CPE を用いることで，ベンダ，セキュリティ専門家，管理者，ユーザ間で脆弱性の存在する対象となるプラットフォームを共通の言葉で情報交換することが可能となる．CPE は Uniform Resource Identifiers(URI) に基づいた記述フォーマットを用いている．CPE は，cpe:/{ 種別 }:{ ベンダ名 }:{ 製品名 }:{ バージョン }:{ アップデート }:{ エディション }:{ 言語 } という構成で定義される．{ 種別 } は，製品種別を判別する情報である．'h' はハードウェア，'o' は OS，'a' はアプリケーションを示している．{ ベンダ名 } は，ドメイン名を利用して記述する．アプリケーションにベンダや組織が存在しない場合，開発者名を使用する．{ 製品名 } は，製品名を記述する．複数の単語で構成されている製品名の場合は，スペースをアンダースコアに置き換えた名前を記述する．{ バージョン } は，製品のバージョン情報を記述する．バージョンの記述は，製品名の記述で用いる形式と同様の形式で記述する．{ アップデート } は，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．バージョンとアップデートの違いは，ベンダや製品が出す情報に依存する．{ エディション } は，製品のエディションを記述する．エディションとはソフトウェアの提供方法等を示す．{ 言語 } は，製品で使用している言語を特定する場合に記述する．CPE 名は，短く表記するために略称を用いる場合がある．

XCCDF

XCCDF は，セキュリティのチェックリストとその関連情報を XML 形式で記述するための仕様である．XCCDF は，セキュリティ関連機関の協力により，NSA[5]が主体となって開発し，NIST によって公表されている．また，米国政府が推進している情報セキュリティに関わる技術面での自動化と標準化を実現する技術仕様である SCAP の構成要素の一つである．参照する各種ガイドライン，その推奨値，使用する検査データなどを記載する．XCCDF の目的は，設定ガイドの表現に単一の基盤を提供し，適切なセキュリティ対策の幅広い適用を促進することである．

OVAL

OVAL は，システムにインストールされている OS やアプリケーションの情報や，セキュリティ設定情報の収集方法を記述するための言語である．OVAL は，MITRE 社が中心となり仕様が策定された．また，米国政府が推進している情報セキュリティに関わる技術面での自動化と標準化を実現する技術仕様である SCAP の構成要素の一つである．OVAL 言語で作成された脆弱性対策情報である OVAL 定義データは，その OVAL 定義データを解釈するプログラムである OVAL インタプリタを用いて機械的に処理することで脆弱性対策のための確認作業の自動化を可能とする．また，OVAL 定義データは，OVAL リポジトリとしてデータベース化することにより，利活用の促進が図られている．

CWE

CWE は，脆弱性の種類を識別するための基準である．MITRE 社が中心となり仕様が策定された．多種多様な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）を付与して階層構造で体系化している．上位層に近いほど抽象的な脆弱

[5]NSA：National Security Agency．国家安全保障局

性タイプを表し，下位層にいくほど具体的な脆弱性タイプや個々の脆弱性を表している．CWE を用いることで，下記の利点が得られる．

- ソフトウェアのアーキテクチャ，デザイン，コードに内在する脆弱性に関して，共通の言葉で議論できるようになる．
- 脆弱性検査ツールなど，ソフトウェアのセキュリティを向上させるための，ツールの標準の評価尺度として使用できる．
- 脆弱性の原因を認識し，脆弱性の低減を行い，再発を防止するための共通の基準として活用できる．

CWE は多種多様な脆弱性の特性を脆弱性タイプとして分類し，それぞれに一意の識別子である CWE 識別子を付与する．CWE 識別子は，”CWE-識別番号”という構成で定義されている．CWE 識別子は階層構造で体系化されており，上位層に近いほど抽象的な脆弱性タイプを表し，下位層にいくほど具体的な脆弱性タイプを表す．脆弱性タイプはビュー，カテゴリ，脆弱性，複合要因の 4 種類に分類される．現在，ビューとして 27 個，カテゴリとして 157 個，脆弱性として 693 個，複合要因として 9 個，合計 886 個の脆弱性タイプが分類され一覧となっている．

ビューは，ある観点からいくつかの脆弱性タイプを選択して集めたものである．例えば，NIST では実際に公表されている脆弱性を考慮し，CWE の中から 19 個の脆弱性タイプを選択して NVD に掲載している．この 19 個の脆弱性タイプを集めたビューに CWE-635 の CWE 識別子が割り当てられている．また，開発者の観点から脆弱性タイプを集めたビューに CWE-699，研究者の観点のものに CWE-1000，C 言語に起因するものに CWE-658，Java 言語に起因するものに CWE-660 などが割り当てられている．

カテゴリは，共通の特性を持つ脆弱性タイプをグループ化したものである．例えば，CWE-310 の暗号の問題に関連する脆弱性，また CWE-355 のユーザインターフェースに関連する脆弱性が該当する．

脆弱性は，個々の脆弱性を表したもので，クラス，ベース，バリアントの属性が付与されている．クラスは，最も抽象的な脆弱性の属性である．例えば，CWE-362 の競合状態の脆弱性が該当する．ベースは，特定のリソースや技術に依存しない脆弱性の属性である．例えば，CWE-567 の共有データへの非同期アクセスの脆弱性が該当する．バリアントは，個々のリソースや技術，コンテキストなどが特定できるような脆弱性の属性である．例えば，CWE-488 の異なるセッション間で適切にセッションを識別できないことによる情報漏えいの脆弱性が該当する．

複合要因は，複数の要因が複合した脆弱性を表したもので，コンポジットとチェインの属性が付与されている．コンポジットは，複数の脆弱性が混合して発生する脆弱性の属性である．例えば，CWE-352 のクロスサイト・リクエスト・フォージェリ (CSRF) の脆弱性が該当する．チェインは，ある問題が原因で別の問題が連鎖して発生する脆弱性の属性である．例えば，CWE-680 の整数オーバーフローの発生によるバッファオーバーフローの脆弱性が該当する．

CWE が提供する脆弱性タイプ一覧のリストには，CWE 識別子ごとに様々な情報が記載されている．脆弱性の概要，攻撃の受けやすさ，一般的な脅威，脆弱性の軽減策，脆弱性の発生する具体的なコーディング例，当該脆弱性に起因する具体的な事例の紹介などの情報が整理されている．利用者は，脆弱性を識別し，脆弱性の低減を行い，再発を防止するための辞書として活用することができる．

CAPEC

CAPEC は，攻撃の定型である攻撃パターンを特定し，記述し，列挙するための表現を XML/XSD ベースで規定する．これにより，攻撃パターンのカタログをわかりやすい構文と分類に沿って広く公開できるようにすることを目的としている．Mitre 社が中心となり仕様が策定された．攻撃パターンは，攻撃による影響の緩和や，ソフトウェアを開発する上での助言を提供するための共通の記述方法である．

攻撃パターンに付与される識別子である攻撃パターン識別子は，”CAPEC-識別番号”という構成で定義される．CWE 識別子は階層構造で体系化されており，上位層に近いほど抽象的な攻撃パターンを表し，下位層にいくほど具体的な攻撃パターンを表す．CAPEC のリストはビュー，カテゴリ，攻撃パターンの 4 種類に分類される．現在，ビューとして 6 個，カテゴリとして 68 個，攻撃パターンとして 386 個，合計 460 個の要素が分類され一覧となっている．

ビューは，ある観点からいくつかの脆弱性タイプを選択して集めたものである．カテゴリは，共通の特性を持つ脆弱性タイプをグループ化したものである．攻撃パターンは，個々の攻撃の定型を表したものである．

3.2 脆弱性情報データベース

脆弱性に関する報告書や注意喚起の情報は，様々な組織が独自の公開形式を持っており，脆弱性情報を横断的に取得，及び検索することは多くの時間と労力を必要とする．米国政府では，脆弱性情報をデータベースとして管理するための NVD (National Vulnerability Database)[17] を公開している．脆弱性情報の提供者はデータベースに脆弱性情報を登録し，脆弱性情報の利用者はデータベースから情報を取得することで，脆弱性情報の提供者と利用者の間で効率よく情報共有することが可能となる．

日本においては，脆弱性対策情報ポータルサイトとして JPCERT/CC[6] と情報処理推進機構 (IPA) が共同で運営する JVN (Japan Vulnerability Notes)[18] が公開されている．JVN は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ている．本節では，NVD と JVN について述べる．

NVD

NVD は，米国政府の機関である NIST が提供する脆弱性データのリポジトリである．NVD には，セキュリティチェックリストやセキュリティに関連するソフトウェアの欠点や設定ミス，製品名，及び影響測定基準のデータベースが含まれている．NVD に含まれるデータは，脆弱性管理やセキュリティ対策，コンプライアンスの自動化に利用することができる．NIST の Web サーバから表 6 に示すデータフィードが XML 形式で提供されている．

また，CVE のデータフィードは表 7 に示すデータが提供されている．

[6]一般社団法人 JPCERT コーディネーションセンタ．コンピュータセキュリティ情報を収集し，インシデント対応の支援，コンピュータセキュリティ関連情報の発信などを行なう組織．

データの名称
CVE のデータフィード
CCE のデータフィード
CPE のディクショナリ
CCE 参照データ
NCP[7]チェックリスト

表 6: NIST の Web サーバから提供されるデータフィード

データの名称
一年間に公開された CVE の XML フィード
最近一週間に公開された CVE の XML フィード
最近修正された CVE の XML フィード
最近公開されたすべての CVE の RSS フィード
最近公開され，完全に解析された CVE の RSS フィード

表 7: CVE のデータフィードが提供するデータ

JVN

JVN は，JPCERT/CC と情報処理推進機構 (IPA) が共同で運営する脆弱性対策情報ポータルサイトである．JVN では様々な脆弱性関連情報を収集し，原則として製品開発者との調整を通じて対策方法を準備した上で掲載している．製品開発者の対応状況には，脆弱性に該当する製品の有無，回避策や対策情報も含まれる．定期的な情報収集を可能とするため，蓄積した情報は JS 形式及び RSS 形式で提供されている．

また，脆弱性対策情報データベースとして JVN iPedia [19] が公開されている．JVN iPedia は，国内で利用されるソフトウェア等の製品の脆弱性対策情報を中心に収集及び蓄積するデータベースである．本データベースの公開時点で約 200 社の製品に対する脆弱性対策情報が蓄積されている．JVN iPedia の蓄積対象となる脆弱性対策情報は，JVN に掲載される情報の他，NVD および国内ベンダの情報である．JVN iPedia では，目的の脆弱性対策情報を容易に探すために検索機能が用意されている．特定の製品に存在する脆弱性の確認など，入手したい情報が特定されている場合，検索機能によって効率的に検索することが可能である．

3.3 ネットワークシステムにおける脆弱性影響度の定量化

これまでに，脆弱性の影響度を分析する研究がいくつかおこなわれている．文献 [8] では，ネットワークシステムの表現モデルである NSQ モデル [20] を用いて，ネットワークシステムにおける脆弱性の影響度を定量化した．

原田らは，FIRST が推進する共通脆弱性評価システムである CVSS[10] を用いて，ネットワークシステムにおける脆弱性影響度範囲の検討をおこない，冗長化効果および中継機器における可用性影響の大きさを測定した．これらの研究は，ネットワークシステムにおける脆弱性の波及範囲を定量化している．

本研究では，端末ベースでの影響度を特定することで，単一システム内で発生する影響を特定することを目的とする．

3.4 リスク管理に関連する製品

McAfee Vulnerability Manager

McAfee Vulnerability Manager[21] は，最新の脆弱性および脅威情報の収集と情報資産の状況管理を自動化し，効率的なセキュリティ対策を支援するソリューションである．主な特徴として，下記が挙げられる．

- 最新の脆弱性や脅威に関する情報を定期的に提供する．
- 定期的な検査でネットワークに接続された資産を把握，管理する．
- 重要資産に深刻度の高い脆弱性が検出されると自動で担当者に作業指示を発行する．
- 対応状況，リスクレベル，対策作業の効果を継続的に把握する．

Microsoft Forefront Endpoint Protection

Microsoft Forefront Endpoint Protection[22] は，不正アクセスの脅威に対してエンドポイントの OS 保護を簡素化する管理ツールである．企業や大学などの組織のネットワーク内のマシンの一元管理が可能となる．下記の４つの機能を持っている．

- 結合コンソール
- 一元的なポリシー作成
- 可視性の向上
- エージェントの自動置き換え

結合コンソール機能によって，エンドポイントの管理とセキュリティ保護に単一のインターフェイスを提供する．この機能により，管理の複雑を低減している．一元的なポリシー作成機能により，管理者はエンドポイントに関するすべてのポリシーを一元的に作成して適用することが可能となる．可視性の向上機能によって，エンドポイントの保護と構成に共有ビューを持つ．この機能により，管理者は脆弱性のあるコンピュータの特定および修復をより簡単に行える．エージェントの自動置き換え機能によって，コンフリクトの原因となる他社等の一般的なエンドポイントセキュリティエージェントを自動的に検出して削除する．この機能により，冗長な管理コストを削減する．

Symantec Endpoint Protection

Symantec Endpoint Protection[23] は，エンドポイントのマシンを保護する管理ツールである．Symantec 社が持つデータベースと用いることで，エンドポイントマシンにダウンロードやインストールするソフトウェアの普及度が低い場合警告を出す．一方，普及度が高いソフトウェアに対してはスキャン対象から除外し，マシン全体のスキャン時間を短縮する．また，プロセスの動きを観察することで，定義ファイルで検出できない未知のマルウェアを識別している．

Qualys Guard

Qualys Guard[24] は，セキュリティリスクおよびポリシーコンプライアンス管理を単一のソリューションで実現することで，リスクの低減とコンプライアンスプロセスの管理を可能にしている．企業全体の脆弱性管理とポリシーコンプライアンスのプロセスを自動化し，ビジネスリスクに基づいてネットワーク検出とマッピング，資産の優先順位付け，脆弱性評価レポート，改善の追跡を行う．また，ポリシーコンプライアンス機能により，セキュリティマネージャは社内セキュリティ

ポリシーおよび社外の規制のコンプライアンスを監査，実施，文書化することが可能となる．

4　遷移グラフ提示システム ssch

脆弱なシステムを使用することによってユーザ自身が受けるインパクトを段階を追って紐付けることで，脆弱性から引き起こされるインパクトがイメージでき，脆弱なシステムを使用することの危険性を周知させることができると考えられる．

本研究では，システムに含まれるソフトウェアの脆弱性から，発生するインパクトをトレース可能な遷移グラフと，遷移グラフを生成するツールである ssch の設計および実装に関して説明する．

4.1　要件定義

本研究ではユーザが自身のシステムに起こり得る影響を理解することを目的とするため，提案システムでは下記の要件を満たす必要がある．

- システムに内在する脆弱性が抽出できる
- 脆弱性とシステムへの影響が関連付けられる

4.2　遷移グラフ提示システム ssch の設計

4.2.1　システムの概要

本節では，我々が提案するシステム ssch の概要を述べる．ssch は，検査対象システムに内在する脆弱性をスキャンし，検査対象システムが受ける可能性のある影響を提示することが可能である．ssch のシステム概要を図 3 に示す．

検査対象システムに内在する脆弱性は，インストールされているソフトウェアに起因する．そのため，ssch のスキャン部では，インストール済みソフトウェアの情報を取得する．

次に，リストアップされたソフトウェアの脆弱性情報を取得する．脆弱性情報には，NIST が管理する脆弱性情報データベースである NVD[17] を利用する．NVD は，MITRE 社が管理するベンダ非依存な共通脆弱性識別子である CVE[9] で命名された脆弱性情報を扱っている．CVE では，CVE 識別番号によって脆弱性が

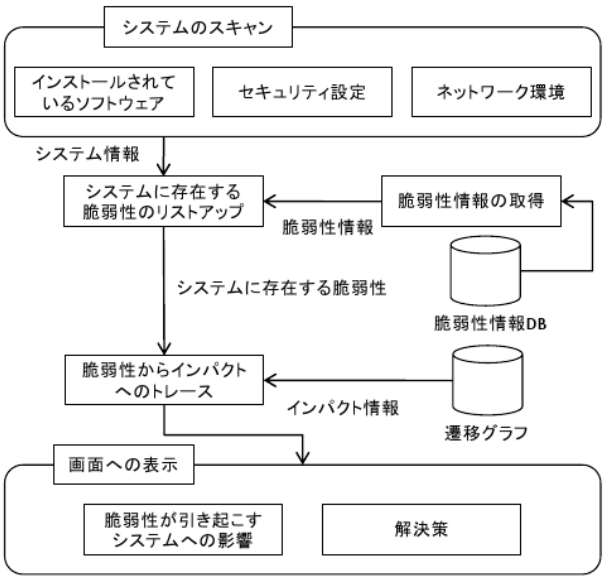

図 3: システムの概要

一意に識別されている．NVD では，識別された脆弱性の種類や重要性，脆弱性が発見されたソフトウェア名とバージョン，ソフトウェアを開発した企業名などが登録されており，毎日更新されている．このデータは一般に公開されており，XML 形式で取得が可能である．これらの脆弱性情報から，後述する遷移グラフを作成する．ssch の画面への表示部では，遷移グラフによってトレースされた結果と解決策をユーザに提示する．

4.2.2　遷移グラフ

遷移グラフとは，脆弱性と，脆弱性が持つ特性，及びシステムへ与える可能性のある影響を関連付けたグラフである．遷移グラフを用いることで，システムをスキャンした結果から，システムへ与える可能性のある影響を特定することが可能となる．遷移グラフのイメージを図 4 に示す．

遷移グラフは，脆弱性情報と脆弱性タイプ情報，攻撃パターン情報で成り立っている．それぞれの情報が関連付けられており，脆弱性から脆弱性タイプと攻撃パターンをトレースすることが可能となる．図中の各ノードは，識別番号の他にユーザに提示すべき情報を持っている．

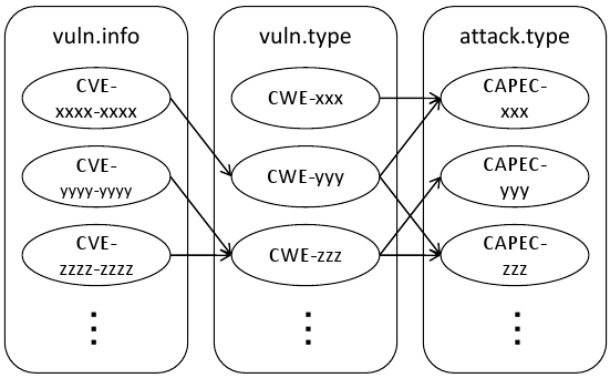

図 4: 遷移グラフのイメージ

脆弱性情報には，NVD を利用する．NVD からは下記の情報を取得する．

- vuln:cve-id (CVE 識別番号)
- vuln:cwe-id (CWE 識別番号)
- vuln:vulnerable-software-list (ソフトウェア名とバージョン)

脆弱性に伴う影響の原因や対策を一意に識別するために，脆弱性が持つ特性をカテゴライズする必要がある．脆弱性のカテゴライズには，MITRE 社が中心となり仕様策定が行われた共通脆弱性タイプ一覧である CWE[15] を利用する．CWE は，SQL インジェクション，クロスサイト・スクリプティング，バッファオーバーフローなど，多種多様にわたるソフトウェアの脆弱性を脆弱性タイプとして分類している．また，それぞれに CWE 識別子を付与し，階層構造で体系化している．CWE からは，脆弱性タイプ情報として下記の情報を取得する．

- Name (脆弱性タイプ名)
- Description_Summary (概要)
- Common_Consequences (一般的な影響)
- Detection_Methods (被害の暖和策)
- Relationship (関係性)
- Affected_Resource (影響を受けるリソース)
- Related_Attack_Patterns (関連する攻撃パターン)

脆弱性の存在によって被害を受ける可能性のある攻撃手法を識別するために，攻撃手法に一意の識別子が必要となる．攻撃手法の識別には，MITRE 社が推進する攻撃パターン一覧である CAPEC[16] を利用する．また，それぞれに CWE 識別子を付与し，階層構造で体系化している．CAPEC からは，攻撃パターン情報として下記の情報を取得する．

- Summary (概要)
- Solutions_and_Mitigations (被害の暖和策)
- Payload_Activation_Impact (ペイロードが実行されることにより受ける影響)
- CIA_Impact (機密性，保全性，可用性に与える影響)

4.3 ssch の試作

本節では，4.2 節で述べた設計を基に実装した ssch と，ssch の実行結果について詳述する．

4.3.1 ssch の実装

本節では，実装した ssch について詳述する．ssch は，FreeBSD, Windows の二つの OS に対応する．ssch は Perl で実装し，表に挙げるモジュールを利用している．

Perl モジュール名
File::Basename
HTTP::Request::Common
LWP::UserAgent
XML::Simple
XML::Parser
XML::Parser::Expat
XML::SAX
Win32::Registry
Data::Dumper
GraphViz
HTML::Template

表 8: ssch で利用した Perl モジュール

ssch は下記の機能を実装している．

機能 1 システムにインストールされているソフトウェアを抽出する

機能 2 NVD からソフトウェアに含まれる脆弱性を抽出する

機能 3 CWE 及び CAPEC から脆弱性の特性とシステムが受ける可能性のある攻撃パターンを抽出する

機能 1 では，システムにインストールされているソフトウェアをスキャンするために，パッケージ管理ソフトを用いた．パッケージ管理ソフトでインストールされたソフトウェアに限り，ソフトウェア名，バージョン情報を取得することができる．機能 2 では，スキャンしたソフトウェアに対して脆弱性の有無を確認するために，XML で記述された NVD から，ソフトウェア名とバージョンを正規表現を利用して特定する．機能 3 では，XML で記述された CWE の情報から，該当する CVE 識別番号の脆弱性タイプを特定する．また，XML で記述された CAPEC の情報から，該当する脆弱性タイプが受ける攻撃パターンを特定する．

次に ssch の擬似コードをアルゴリズム 1 に示す．まず，検査対象システムにインストールされているソフトウェアのソフトウェア名，バージョンを抽出する．Windows 環境に実装したシステム情報抽出部では，インストールされているソフトウェアの情報をレジストリから取得している．Windows のレジストリはソフトウェアによってバージョン情報が保存されている階層が異なるため，対応するソフトウェアを脆弱性の多いソフトウェアに限定した．対応するソフトウェアの選定には，Secunia 社が公開している Secunia Half Year Report 2010[25] の調査結果を用いた．2009 年 6 月から 2010 年 6 月の間に CVE に登録された脆弱性の中で，登録数の上位 10 個のソフトウェアとして報告されたサードパーティプログラムと，上位 5 個の Microsoft 製品を対象としている．選定したソフトウェアを表 9 に示す．FreeBSD 環境に実装したシステム情報抽出部では，インストールされているソフトウェアの情報を portaudit[26] を用いて取得している．次に，NVD から取得した XML データに含まれるタグ vuln:vulnerable-software-list の要素に，システムにインストールされていたソフトウェアが含まれているか正規表現を用いて確認する．vuln:vulnerable-software-list の要素と，検査対象システムにインストールされているソフトウェア名とバージョンが一致した場合は，該当エントリの

Algorithm 1 システム情報から攻撃パターン情報へのトレース

$SoftList \leftarrow$ パッケージ管理ソフトのリスト
$nvd \leftarrow$ NVD の XML データ
for all *entry* in *nvd* **do**
  if $SoftList$ is $< vuln : vulnerable-software-list >$ **then**
    $cve \leftarrow < vuln : cve-id >$
    $cwe \leftarrow < vuln : cwe-id >$
  end if
end for
$cwe \leftarrow$ CWE の XML データ
for all *entry* in *cwe* **do**
  if *cwe* is $< CWE-ID >$ **then**
    $capec \leftarrow < CAPEC-ID >$
  end if
end for
$capec \leftarrow$CAPEC の XML データ
for all *entry* in *capec* **do**
  if *capec* is $< CAPEC-ID >$ **then**
    return *entry*
  end if
end for

vuln:cve-id と vuln:cwe-id の要素を抽出する．そして，vuln:cwe-id の要素をキーに，CWE から取得した XML データのタグ CAPEC_ID の要素を抽出する．最後に，CAPEC_ID の要素をキーに，CAPEC から取得した XML データに含まれる情報を抽出する．

4.3.2 ssch の実行結果

本節では，ssch を実行した結果について述べる．ssch を実行することで，下記のデータが出力される．

- 遷移グラフ (図 5)
- ソフトウェアのバージョンリスト (図 6)
- 発見した CVE のリスト (図 7)
- 発見した CVE に関連する CWE のリスト (図 8)
- CWE に関連する CAPEC のリスト (図 9)

#	Rank Program	Vendor	CVEs
1	Mozilla Firefox	Mozilla Foundation	96
2	Apple Safari	Apple	84
3	Sun Java JRE	Sun(Oracle)	70
4	Google Chrome	Google	70
5	Adobe Reader	Adobe	69
6	Adobe Acrobat	Adobe	69
7	Adobe Flash Player	Adobe	51
8	Adobe AIR	Adobe	51
9	Internet Explorer	Microsoft	49
10	Apple iTunes	Apple	48
11	Excel Viewer	Microsoft	37
12	Mozilla Thunderbird	Mozilla Foundation	36
13	Excel	Microsoft	30
14	Visual Studio	Microsoft	15
15	.NET Framework	Microsoft	13

表 9: Windows 環境で情報を取得するソフトウェア

本結果は，脆弱性が存在するソフトウェアである Apple Safari 5.0.1，および Mozilla Firefox 6.0 がインストールされたマシンで実行した結果である．NVD が提供する 2011 年の CVE を用いて調査した結果，58 個の CVE，6 個の CWE，63 個の CAPEC の情報が関連付けられた．

4.3.3 公開レビュー

現在 SourceForge.NET にて試作した ssch を公開し，ユーザの意見を募っている [1]．また，ssch の設計に関してはコンピュータセキュリティシンポジウム（CSS2011）[2] にて発表した．

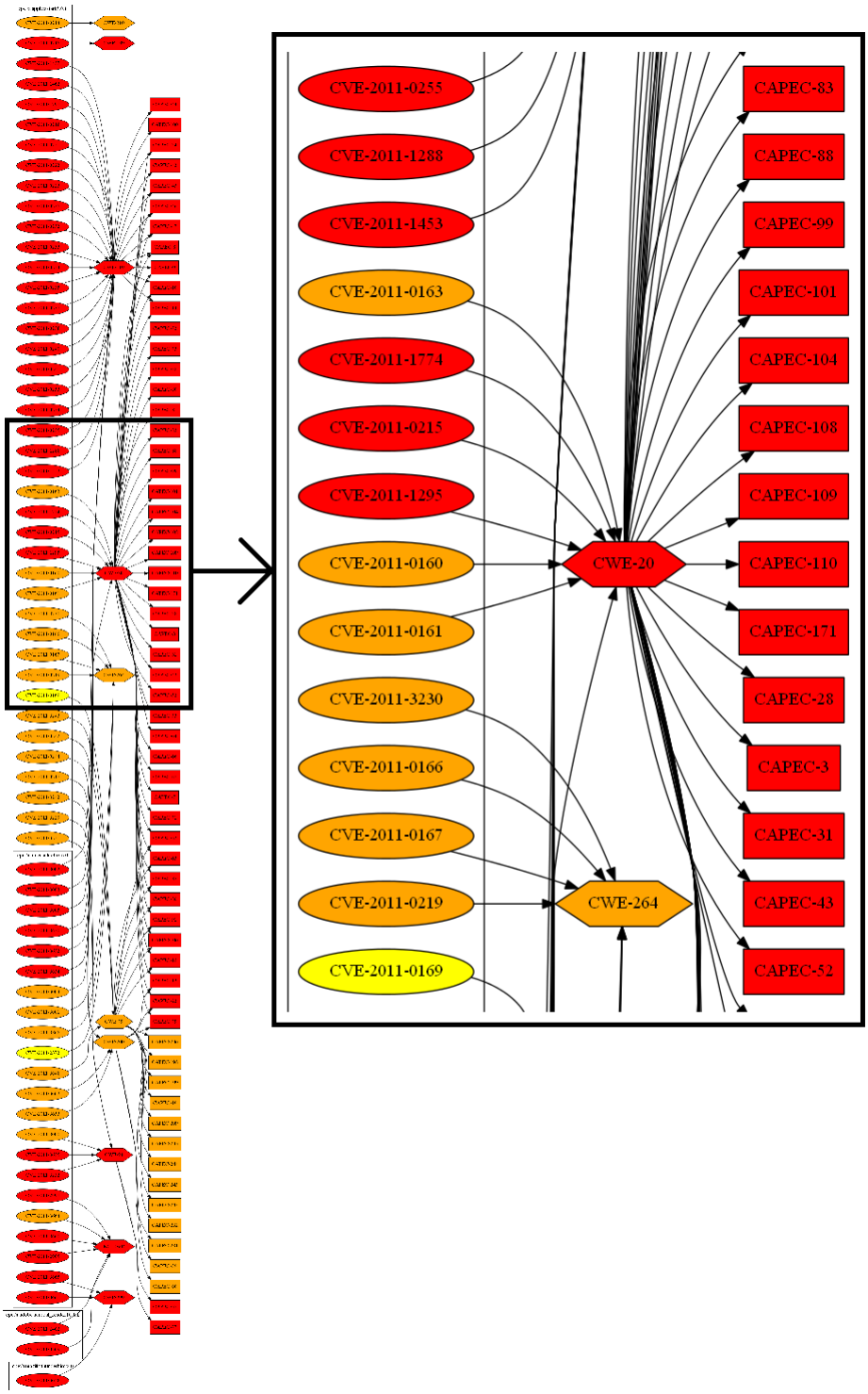
CVE-2011-0255
CVE-2011-1288
CVE-2011-1453
CVE-2011-0163
CVE-2011-1774
CVE-2011-0215
CVE-2011-1295
CVE-2011-0160
CVE-2011-0161
CVE-2011-3230
CVE-2011-0166
CVE-2011-0167
CVE-2011-0219
CVE-2011-0169
CWE-20
CWE-264
CAPEC-83
CAPEC-88
CAPEC-99
CAPEC-101
CAPEC-104
CAPEC-108
CAPEC-109
CAPEC-110
CAPEC-171
CAPEC-28
CAPEC-3
CAPEC-31
CAPEC-43
CAPEC-52

図 5: 遷移グラフ

```
cpe:/a:adobe:adobe_air 3.0.0.4080
cpe:/a:google:chrome 15.0.874.121
cpe:/a:adobe:acrobat_reader 10.1.1
cpe:/a:apple:itunes 10.5.1.42
cpe:/a:apple:safari 5.0.1
cpe:/a:microsoft:internet_explorer 9.0.8112.16421
cpe:/a:mozilla:firefox 6.0
cpe:/a:mozilla:thunderbird 8.0
cpe:/a:adobe:flash_player 10,3,183,5
cpe:/a:microsoft:excel_viewer 12.0.6612.1000
cpe:/a:microsoft:excel 12.0.6612.1000
cpe:/a:sun:jre 1.6.0_29-b110
```

図 6: ソフトウェアのリスト

```
CVE-2011-0160,cpe:/a:apple:safari:5.0.1
CVE-2011-0161,cpe:/a:apple:safari:5.0.1
CVE-2011-0163,cpe:/a:apple:safari:5.0.1
CVE-2011-0166,cpe:/a:apple:safari:5.0.1
CVE-2011-0167,cpe:/a:apple:safari:5.0.1
CVE-2011-0169,cpe:/a:apple:safari:5.0.1
CVE-2011-0214,cpe:/a:apple:safari:5.0.1
CVE-2011-0215,cpe:/a:apple:safari:5.0.1
CVE-2011-0216,cpe:/a:apple:safari:5.0.1
CVE-2011-0217,cpe:/a:apple:safari:5.0.1
CVE-2011-0218,cpe:/a:apple:safari:5.0.1
CVE-2011-0219,cpe:/a:apple:safari:5.0.1
CVE-2011-0221,cpe:/a:apple:safari:5.0.1
CVE-2011-0222,cpe:/a:apple:safari:5.0.1
CVE-2011-0223,cpe:/a:apple:safari:5.0.1
CVE-2011-0225,cpe:/a:apple:safari:5.0.1
CVE-2011-0232,cpe:/a:apple:safari:5.0.1
CVE-2011-0233,cpe:/a:apple:safari:5.0.1
CVE-2011-0234,cpe:/a:apple:safari:5.0.1
CVE-2011-0235,cpe:/a:apple:safari:5.0.1
CVE-2011-0237,cpe:/a:apple:safari:5.0.1
CVE-2011-0238,cpe:/a:apple:safari:5.0.1
CVE-2011-0240,cpe:/a:apple:safari:5.0.1
CVE-2011-0241,cpe:/a:apple:safari:5.0.1
CVE-2011-0242,cpe:/a:apple:safari:5.0.1
CVE-2011-0244,cpe:/a:apple:safari:5.0.1
CVE-2011-0253,cpe:/a:apple:safari:5.0.1
CVE-2011-0254,cpe:/a:apple:safari:5.0.1
CVE-2011-0255,cpe:/a:apple:safari:5.0.1
CVE-2011-1288,cpe:/a:apple:safari:5.0.1
CVE-2011-1295,cpe:/a:apple:safari:5.0.1
CVE-2011-1453,cpe:/a:apple:safari:5.0.1
CVE-2011-1457,cpe:/a:apple:safari:5.0.1
CVE-2011-1462,cpe:/a:apple:safari:5.0.1
CVE-2011-1774,cpe:/a:apple:safari:5.0.1
CVE-2011-1797,cpe:/a:apple:safari:5.0.1
CVE-2011-2372,cpe:/a:mozilla:firefox:6.0
CVE-2011-2995,cpe:/a:mozilla:firefox:6.0
CVE-2011-2997,cpe:/a:mozilla:firefox:6.0
CVE-2011-3000,cpe:/a:mozilla:firefox:6.0
CVE-2011-3001,cpe:/a:mozilla:firefox:6.0
CVE-2011-3002,cpe:/a:mozilla:firefox:6.0
CVE-2011-3003,cpe:/a:mozilla:firefox:6.0
CVE-2011-3004,cpe:/a:mozilla:firefox:6.0
CVE-2011-3005,cpe:/a:mozilla:firefox:6.0
– snip –
```

図 7: CVE のリスト

```
119,Improper Restriction of Operations within the Bounds of a Memory Buffer
20,Improper Input Validation
200,Information Exposure
22,Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
79,Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
94,Improper Control of Generation of Code ('Code Injection')
```

図 8: CWE のリスト

```
10,Buffer Overflow via Environment Variables
100,Overflow Buffers
101,Server Side Include (SSI) Injection
104,Cross Zone Scripting
106,Cross Site Scripting through Log Files
108,Command Line Execution through SQL Injection
109,Object Relational Mapping Injection
110,SQL Injection through SOAP Parameter Tampering
13,Subverting Environment Variable Values
139,Relative Path Traversal
14,Client-side Injection-induced Buffer Overflow
171,Variable Manipulation
18,Embedding Scripts in Nonscript Elements
19,Embedding Scripts within Scripts
198,Cross-Site Scripting in Error Pages
199,Cross-Site Scripting Using Alternate Syntax
209,Cross-Site Scripting Using MIME Type Mismatch
22,Exploiting Trust in Client (aka Make the Client Invisible)
23,File System Function Injection, Content Based
24,Filter Failure through Buffer Overflow
243,Cross-Site Scripting in Attributes
244,Cross-Site Scripting via Encoded URI Schemes
245,Cross-Site Scripting Using Doubled Characters, e.g. %3C%3Cscript
246,Cross-Site Scripting Using Flash
247,Cross-Site Scripting with Masking through Invalid Characters in Identifiers
28,Fuzzing
281,Analytic Attacks
3,Using Leading 'Ghost' Character Sequences to Bypass Input Filters
31,Accessing/Intercepting/Modifying HTTP Cookies
32,Embedding Scripts in HTTP Query Strings
35,Leverage Executable Code in Nonexecutable Files
42,MIME Conversion
43,Exploiting Multiple Input Interpretation Layers
44,Overflow Binary Resource File
45,Buffer Overflow via Symbolic Links
46,Overflow Variables and Tags
47,Buffer Overflow via Parameter Expansion
52,Embedding NULL Bytes
53,Postfix, Null Terminate, and Backslash
59,Session Credential Falsification through Prediction
60,Reusing Session IDs (aka Session Replay)
63,Simple Script Injection
64,Using Slashes and URL Encoding Combined to Bypass Validation Logic
66,SQL Injection
67,String Format Overflow in syslog()
– snip –
```

図 9: CAPEC のリスト

参考文献

[1] Jingu Masato. Security status checker. http://sourceforge.net/projects/ssch/.

[2] 神宮真人, Gregory Blanc, 奥田剛, 山口英. 脆弱性がもたらす影響をトレース可能な遷移グラフの提案. コンピュータセキュリティシンポジウム 2011(CSS2011), pp. 205 – 210, 2011 年 10 月.

[3] 総務省. 平成 23 年版 情報通信白書. http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h23/pdf/23honpen.pdf.

[4] 情報処理推進機構セキュリティセンター. 2009 年国内における情報セキュリティ事象 被害状況と新たな脅威の実態に関する調査. http://www.ipa.go.jp/security/fy21/reports/isec-survey/documents/2009_isec_domestic.pdf.

[5] JPCERT コーディネーションセンター. ボットネットの概要. http://www.jpcert.or.jp/research/2006/Botnet_summary_0720.pdf.

[6] 経済産業省. コンピュータ不正アクセス対策基準. http://www.meti.go.jp/policy/netsecurity/UAaccessCMG.htm.

[7] サイバークリーンセンターボット対策プロジェクト. 平成 20 年度サイバークリーンセンター活動報告. https://www.ccc.go.jp/report/h20ccc_report.pdf.

[8] T. Harada, A. Kanaoka, E. Okamoto, M. Kato. Identifying Potentially-Impacted Area using CVSS for Networked Systems. *Proceedings of The First Workshop on Convergence Security and Privacy (CSnP)*, July 2010.

[9] MITRE Corporation. CVE. http://cve.mitre.org/.

[10] National Institute of Standards and Technology. National Vulnerability Database CVSS Scoring. http://nvd.nist.gov/cvss.cfm.

[11] MITRE Corporation. CCE. http://cce.mitre.org/.

[12] MITRE Corporation. CPE. http://cpe.mitre.org/.

[13] National Institute of Standards and Technology. XCCDF. http://scap.nist.gov/specifications/xccdf/.

[14] MITRE Corporation. OVAL. http://oval.mitre.org/.

[15] MITRE Corporation. CWE. http://cwe.mitre.org/.

[16] MITRE Corporation. CAPEC. http://capec.mitre.org/.

[17] National Institute of Standards and Technology. National Vulnerability Database Home. http://nvd.nist.gov/.

[18] JPCERT コーディネーションセンター & 独立行政法人情報処理推進機構 (IPA). Japan Vulnerability Notes. http://jvn.jp/.

[19] JPCERT コーディネーションセンター & 独立行政法人情報処理推進機構 (IPA). JVN iPedia. http://jvndb.jvn.jp/.

[20] 金岡 晃, 藤堂 伸勝, 加藤 雅彦, 岡本 栄司. ネットワークシステムの安全性定量化に向けた新たな表現モデルとアクセス制御解析. 2008 年 暗号と情報セキュリティシンポジウム (SCIS), 2008 年 1 月.

[21] マカフィー株式会社. Mcafee vulnerability manager リスク管理ソリューション. http://www.mcafee.com/japan/products/vulnerability_manager.asp.

[22] Microsoft Corporation. Forefront endpoint protection. http://www.microsoft.com/en-us/server-cloud/forefront/endpoint-protection.aspx.

[23] Symantec Corporation. Symantec endpoint protection. http://www.symantec.com/business/theme.jsp?themeid=sep-family.

[24] Qualys, Inc. Qualys guard. http://www.qualys.com/products/qg_suite/.

[25] Secunia. Secunia half year report 2010. http://secunia.com/gfx/pdf/Secunia_Half_Year_Report_2010.pdf.

[26] FreeBSD. portaudit. http://www.freebsd.org/ports/portaudit/.