情報処理学会 コンピュータセキュリティ 研究報告 Vol.2009 No.20, pp.283-288 (Mar. 5-6, 2009)

MyJVN を用いた脆弱性対策情報提供サービスの検討

寺田真敏[†1]　杉山　賢[†1]　山岸　正[†1]
小林偉昭[†1]　土居範久[†2]

[†1)] (独)情報処理推進機構 (IPA)
〒113-6591　東京都文京区本駒込 2-28-8
[†2)] 中央大学大学院 理工学研究科
〒112-8551　東京都文京区春日 1-13-27

概要：MyJVN は，処理の機械化や自動化を考慮した流通基盤上に，JVN の脆弱性対策情報を用いたサービスを構築するフレームワークである．しかしながら，流通基盤上で交換される情報は文書情報であり，脆弱性の有無をチェックして対策を促すなどの脆弱性対策に関わる処理の機械化については発展途上にある．本稿では，脆弱性対策情報の流通基盤の拡張として，セキュリティ問題をチェックする手続き仕様 OVAL (Open Vulnerability Assessment Language)を用いた脆弱性対策に関わる処理の機械化を提案すると共に，開発したプロトタイプシステムについて述べる．
キーワード：セキュリティ，脆弱性，Web API，JVN，OVAL

Feasibility study of vulnerability information service by MyJVN

Masato Terada[†1]　Ken Sugiyama[†1]　Tadashi Yamagishi[†1]
Hideaki Kobayashi[†1]　Norihisa Doi[†2]

[†1)] Information-technology Promotion Agency, Japan
2-28-8 Honkomagome, Bunkyo, Tokyo, 113-6591 Japan
[†2)] Graduate School of Science and Engineering, Chuo University.
1-13-27 Kasuga, Bunkyo-ku, Tokyo, 112-8551 Japan.

Abstract: MyJVN is framework for exchange of security information and automation of vulnerability countermeasure. Currently, MyJVN has a Web service API that is based on CPE (Common Platform Enumeration) as a structured naming scheme for products. And most of security information of MyJVN is deployed as human readable documents.
It is necessary to improve the security information service environment for automation of vulnerability countermeasure. In this paper, firstly we will explain the specification and applications of MyJVN. Secondly, we will introduce our feasibility study of OVAL (Open Vulnerability Assessment Language) for MyJVN.
Key words: Security, Vulnerability, Web API, JVN, OVAL

1　はじめに

2004 年の情報セキュリティ早期警戒パートナーシップ開始以降，国内においても，JVN (Japan Vulnerability Notes)，製品開発ベンダ，コミュニティなど様々な層での脆弱性対策情報の提供が充実してきている．しかし，対策情報の多くは主に文書として構成されているために，脆弱性の有無をチェックして対策を促すなどの脆弱性対策に関わる処理の機械化については発展途上にある．

米国では，2002 年の FISMA (Federal Information Security Management Act : 連邦情報セキュリティマネジメント法) の施行以降，セキュリティ規格やガイドラインに従い，情報システムにセキュリティ要件を反映する活動が進められている．特に，セキュリティ設定に関する作業を手作業で行なうと，設定ミスや設定者のセキュリティ知識の程度や判断の相違などによりセキュリティ要件を損なう可能性があることから，作業の自動化が試みられている．また，EU では情報セキュリティ推進機関である ENISA (European Network and Information Security Agency : 欧州ネットワーク情報セキュリティ庁)が中心となってセキュリティ関連情報の共有システムを実現するためのプロジェクトを推進している．

このような状況を踏まえ，JVN では，情報セキュリティの脆弱性対策が国内だけではなく，国際的に

も対応可能とするグローバルな JVN 実現に向け，脆弱性対策に関わる共通基準を積極的に採用すると共に(表 1)，脆弱性対策に関わる処理の機械化を目指すフレームワーク MyJVN を推進している[1].

本稿では，脆弱性対策情報の流通基盤の拡張として，セキュリティ問題をチェックする手続き仕様 OVAL (Open Vulnerability Assessment Language)を用いた脆弱性対策に関わる処理の機械化を提案すると共に，開発したプロトタイプシステムについて述べる．

表 1：JVN で採用している共通基準 [2][3][4][5]

共通基準	概要
脆弱性の共通識別子 CVE (Common Vulnerabilities and Exposures)	プログラム自身に内在するプログラム上のセキュリティ問題に一意の番号(脆弱性識別子)を付与する仕様
共通脆弱性評価システム CVSS (Common Vulnerability Scoring System)	脆弱性自体の特性, パッチの提供状況, ユーザ環境での影響度などを考慮し脆弱性の影響度を評価する仕様
共通脆弱性タイプ一覧 CWE (Common Weakness Enumeration)	プログラム上のセキュリティ問題(脆弱性)の種類を識別するための仕様
共通プラットフォーム一覧 CPE (Common Platform Enumeration)	情報システム, プラットフォーム, ソフトウェアパッケージに一意の名称を付与する仕様

表 2：SCAP を構成する仕様群の概要

共通基準	概要
CCE (Common Configuration Enumeration)	プログラムが稼働するための設定上のセキュリティ問題に一意の番号を付与する仕様
XCCDF (EXtensible Checklist Configuration Description Format)	セキュリティチェックリストやベンチマークなどの文書を記述するための仕様
OVAL (Open Vulnerability Assessment Language)	プログラム上のセキュリティ問題や設定上のセキュリティ問題をチェックするための手続き仕様

2 関連技術

本章では，米国で推進している脆弱性対策関連の機械化処理の技術について整理する．

(1) 機械化処理のフレームワーク

SCAP (Security Content Automation Protocol)は，米 NIST (National Institute of Standards and Technology：国立標準技術研究所)が中心となって推進している機械化処理のフレームワークである．このフレームワークは米国政府を対象とした情報セキュリティ管理の技術面での自動化と標準化を規定した仕様群から構成されている[6]．表 1に示した 3 つの共通基準(除く CWE)の他に，CCE，XCCDF，OVAL の 3 つが規定されている．SCAP では，脆弱性管理，コンプライアンス管理の一部を自動化することにより，情報システムに対するセキュリティ対策の負荷軽減と情報セキュリティ施策の推進の両立を目的としている．

(2) 共通基準

表 1と表 2に示した 7 つの共通基準の他に，米 NIST では, プログラムが稼働するための設定上のセキュリティ問題に対して影響度を評価する仕様 CCSS (Common Configuration Scoring System)の検討を行なっている [7]．セキュリティに関連する共通基準を整備することにより，セキュリティ対策全般に関わる処理の機械化も促進されることになることから，CVE，CCE，OVAL などの仕様開発を担当している MITRE 社では，共通基準の拡充”Making Security Measurable”を推進している．進行している代表的な共通基準は次の通りである [8][9][10].

- 攻撃パターンの識別を共通化する CAPEC (Common Attack Pattern Enumeration and Classification)
- イベントの識別を共通化する CEE (Common Event Expression)
- 情報資産のセキュリティ評価結果を記述する CRF (Common Result Format)

3 MyJVN

本章では，脆弱性対策に関わる処理の機械化を目指すフレームワーク MyJVN について述べた後，解決したい課題について提示する．

3.1 JVN

JVN (Japan Vulnerability Notes)は，セキュリティに関わるシステム管理者ならびにシステムエンジニア向けに脆弱性対策情報を広く告知することを目的とした情報公開サイトである．2003 年 2 月に JPCERT/CC の試行サイトとして運用を開始した[11]. 2004 年 7 月には経済産業省告示「ソフトウェア等脆弱性関連情報取扱基準」[12]を受け，日本国内の製品開発者の脆弱性対応状況を公開するサイト (http://jvn.jp/)として情報発信を行なっている．2007 年 4 月からは，即時性と網羅性を備えた情報提供を実現するため，「JVN」と「JVN iPedia」の 2 つのシステムから構成されている．

- JVN：国内の情報セキュリティ早期警戒パートナーシップ，および海外の調整機関に届けられた脆弱性関連情報に関して製品開発者と調整した脆弱性対策情報をタイムリーに公開する．
- JVN iPedia：国内の製品開発者から公開された

対策情報，および海外の脆弱性対策データベースに登録された情報に基づき，国内で利用されている製品を対象にした脆弱性対策情報を網羅し蓄積する．

3.2 MyJVN

MyJVN は，処理の機械化や自動化を考慮した流通基盤上に，JVN の脆弱性対策情報を用いたサービスを構築するフレームワークである．2008 年 10 月時点で，脆弱性対策情報の流通基盤として，共通プラットフォーム一覧 CPE，Web サービス API (MyJVN Web API)と，次に示す 3 つの XML フォーマットを規定している．

- JVNRSS (JVN RDF Site Summary)[13]
- mod_sec (Qualified Security Advisory Reference)[14]
- VULDEF (The VULnerability Data publication and Exchange Format data model)[15]

また，JVN の脆弱性対策情報を用いたサービスとして，MyJVN Web API と組み合わせ，「製品にどのような脆弱性が存在するのか」という視点から対策情報を選別するフィルタリング型情報提供サービスを実現している．

3.3 解決したい課題

本節では，脆弱性対策情報の流通基盤と脆弱性対策情報を用いたサービスの視点から解決したい課題を示す．

- 脆弱性対策情報の流通基盤における課題
  流通基盤上で交換される情報は文書情報であり，脆弱性の有無をチェックして対策を促すなどの脆弱性対策に関わる処理の機械化については発展途上にある．
- 脆弱性対策情報を用いたサービスにおける課題
  フィルタリング型情報提供サービスでは，MyJVN Web API を利用した脆弱性対策情報チェックツールにより，定期的な脆弱性対策情報のチェック支援を実現している．しかしながら，ツール初期起動時に，普及している製品等をフィルタリング条件とした既定設定か，ユーザが個別に設定するカスタム設定のいずれかでベンダ名，製品名などのフィルタリング条件を設定する必要がある (図 1).

4 脆弱性対策情報の流通基盤の拡張

本章では，前述の課題を解決するため，脆弱性対策情報の流通基盤の拡張項目である，セキュリティ問題をチェックする手続き仕様 OVAL の導入と，OVAL を利用するための Web サービス API の拡張について述べる．

図 1：脆弱性対策情報チェックツールでのフィルタリング条件の設定

4.1 セキュリティ問題をチェックする手続き仕様 OVAL の導入

OVAL [16]は，プログラム上のセキュリティ問題や設定上のセキュリティ問題をチェックする手続き仕様である．2002 年 10 月に開催された SANS Network Security 2002 において，MITRE 社から，その仕様が開示された．XML を用いて汎用的に作られた仕様であり，セキュリティ修正プログラムの適用状況や，稼動しているプラットフォーム／ソフトウェアパッケージの判定にも利用できる．OVAL 開発を担当している MITRE 社では，チェック方法を XML に記述する仕様だけではなく，その XML に従い『プログラム上のセキュリティ問題』や『設定上のセキュリティ問題』をチェックするプログラム (OVAL インタプリタ)を参照実装として提供している．

MyJVN においても，このようなセキュリティ問題をチェックする手続き仕様を整備していくことで，脆弱性の有無をチェックして対策を促すなどの脆弱性対策に関わる処理の機械化が実現できる．将来的には，チェック項目が記載された XML ファイル(以降，OVAL 定義データ)を流通基盤上で交換することにより，脆弱性対策の支援や国際間での脆弱性対策情報の相互運用といった可能性が広がる．

4.2 Web サービス API の拡張

OVAL の導入に伴い，「システムにはどのような脆弱性が存在するのか」「システムにはどのような製品がインストールされているのか」という視点からチェック項目が記載された OVAL 定義データを提供することで，脆弱性の有無チェック処理の機械化だけではなく，脆弱性対策情報チェックツールのフィルタリング条件自動設定が可能となる．また，OVAL 定義データを提供するにあたり，フィルタリ

ング型情報提供サービスの Web サービス API を拡張することで対応できる(表 3).

表 3：MyJVN Web API の拡張メソッド一覧

名称	概要
OVAL 定義一覧取得 getOvalList	フィルタリング条件に該当する OVAL 定義一覧を XML 形式で取得する. http://jvndb.jvn.jp/myjvn?method=getOvalList&cpeName=cpe:/a:jvn*
OVAL 定義データ取得 getOvalData	該当する OVAL 定義データを XML 形式で取得する. http://jvndb.jvn.jp/myjvn/MyJVN?method=getOvalData&ovalid=oval:jp.ac.chuo-u.ise.jvnrss.oval:def:9009

5 フィルタリング型情報提供サービスの拡張

本章では，フィルタリング型情報提供サービスの拡張として，開発したプロトタイプシステムについて述べる．プロトタイプシステムは，脆弱性対策情報の流通基盤の拡張で導入した OVAL 定義データを取得する機能と，OVAL 定義データを解釈する簡易インタプリタ機能を実装している．

5.1 SWF バージョンチェックツール

SWF バージョンチェックツールは，Web ブラウザ上で稼動する SWF ベースの GUI ツールである．OVAL 定義データの<file_state>に記載された Flash Player バージョン情報(図 2)と，自身のバージョンとの比較を行なうことで Flash Player のバージョンチェックを実施する．

バージョン情報が一致しない場合には，最新版のインストールが必要であると判断し，SWF ベースの GUI ツールからダウンロードサイトへのリンクを有効にする (図 3).

```
<states>
  <file_state id="oval:jp.ac.chuo-u.ise.jvnrss.oval:ste:1001" version="1"
  xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">
  <version datatype="version" operation="equal">10.0.12.36</version>
  </file_state>
</states>
```

図 2：Flash バージョン用 OVAL 定義データ (抜粋)

図 3：SWF バージョンチェックツールの結果表示
(バージョン情報不一致の場合)

5.2 脆弱性影響有無チェックツール

脆弱性影響有無チェックツールは，Web ブラウザ上で稼動する Java ベースの GUI ツールである．OVAL 定義データに記載された Windows システムのレジストリ情報を利用して，レジストリ設定値のチェックを実施する．例えば，Windows システムにおいて，すべての種類のドライブが自動再生無効となっている場合には，レジストリキー "NoDriveTypeAutoRun"の値に 0xFF が設定されている．OVAL 定義データでは，これらの情報を <registry_object>と<registry_state>に格納する (図 4).

チェック結果は，図 5に示すようにレジストリ設定値の一致性を提示すると共に，Java ベースの GUI ツールから関連サイトへのリンクを有効にする．

```
<objects>
 <registry_object id="oval:jp.ac.chuo-u.ise.jvnrss.oval:obj:3002" version="1"
 comment="NoDriveTypeAutoRun value is not xFF"
 xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">
  <hive>HKEY_LOCAL_MACHINE</hive>
  <key>SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥policies¥Explorer</key>
  <name>NoDriveTypeAutoRun</name>
 </registry_object>
</objects>
<states>
 <registry_state id="oval:jp.ac.chuo-u.ise.jvnrss.oval:ste:3002" version="1"
 comment="NoDriveTypeAutoRun value is not xFF"
 xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">
 <value>255</value>
 </registry_state>
</states>
```

図 4：レジストリ"NoDriveTypeAutoRun"用
OVAL 定義データ (抜粋)

図 5：脆弱性影響有無チェックツールの結果表示

5.3 脆弱性対策情報チェックツール

脆弱性対策情報チェックツールは，Web ブラウザ上で稼動する Java ベースの GUI ツールである．開発したプロトタイプは，フィルタリング条件設定機能，脆弱性対策概要情報表示機能の 2 つの機能から構成している．

(1) フィルタリング条件設定機能 (図 8の左上部パネル)

MyJVN Web API の拡張メソッドである getOvalList と getOvalData を介して取得した OVAL 定義データを用いて，レジストリ設定値のチェックを実施する．レジストリ設定値からインストールされていると判定した場合には，インストールされている製品の一覧を表示する．

```
<?xml version="1.0" encoding="UTF-8" ?>
 <Result xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://jvndb.jvn.jp/myjvn/Productlist"
 xmlns:status="http://jvndb.jvn.jp/myjvn/Status"
 xsi:schemaLocation="http://jvndb.jvn.jp/myjvn/Productlist
 http://jvnrss.ise.chuo-u.ac.jp/jtg/apis/productlist_2.0.xsd
 http://jvndb.jvn.jp/myjvn/Status
 http://jvnrss.ise.chuo-u.ac.jp/jtg/apis/status_3.0.xsd">
 <VendorInfo xml:lang="en">
  <Vendor vname="Sun Microsystems, Inc." cpe="cpe:/:sun" vid="1">
   <Product pname="JRE" cpe="cpe:/a:sun:jre" pid="266">
    <definition-item oid="oval:jp.ac.chuo-u.ise.jvnrss.oval:def:1200900002003" />
   </Product>
  </Vendor>
  <Vendor vname="mozilla.org contributors" cpe="cpe:/:mozilla" vid="27">
   <Product pname="Mozilla Firefox"
    cpe="cpe:/a:mozilla:firefox" pid="170">
    <definition-item oid="oval:jp.ac.chuo-u.ise.jvnrss.oval:def:1200900002007" />
   </Product>
   <Product pname="Mozilla Thunderbird"
    cpe="cpe:/a:mozilla:thunderbird" pid="504">
    <definition-item oid="oval:jp.ac.chuo-u.ise.jvnrss.oval:def:1200900002008" />
   </Product>
  </Vendor>
  <Vendor vname="OpenOffice.org" cpe="cpe:/:openoffice" vid="214">
   <Product pname="OpenOffice.org"
    cpe="cpe:/a:openoffice:openoffice.org" pid="602">
    <definition-item oid="oval:jp.ac.chuo-u.ise.jvnrss.oval:def:1200900002009" />
   </Product>
  </Vendor>
 </VendorInfo>
 <status:Status xmlns:status="http://jvndb.jvn.jp/myjvn/Status" version="3.0"
 method="getOvalList" lang="en" retCd="0" retMax="10000" errCd="" errMsg=""
 totalRes="7" totalResRet="7" firstRes="1" />
</Result>
```

図 6：OVAL 定義一覧のレスポンス例

```
<?xml version="1.0" encoding="UTF-8" ?>
<Result>
 <oval_definitions>
  <definitions>
   <definition id="oval:jp.ac.chuo-u.ise.jvnrss.oval:def:1200900002007"
   class="vulnerability" version="1">
    <metadata>
     <title>Mozilla Firefox is installed</title>
    </metadata>
    <criteria comment="Mozilla Firefox">
     <criterion comment="Mozilla Firefox is installed"
     test_ref="oval:jp.ac.chuo-u.ise.jvnrss.oval:tst:2007" />
    </criteria>
   </definition>
  </definitions>
  <tests>
   <registry_test id="oval:jp.ac.chuo-u.ise.jvnrss.oval:tst:2007"
   version="1" comment="Mozilla Firefox is installed"
   check_existence="at_least_one_exists" check="at least one"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">
    <object object_ref="oval:jp.ac.chuo-u.ise.jvnrss.oval:obj:2007" />
   </registry_test>
  </tests>

 <objects>
  <registry_object id="oval:jp.ac.chuo-u.ise.jvnrss.oval:obj:2007"
  version="1" comment="Mozilla Firefox is installed"
  xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">
   <hive>HKEY_LOCAL_MACHINE</hive>
   <key>SOFTWARE¥Mozilla¥Mozilla Firefox</key>
   <name xsi:nil="true" />
  </registry_object>
 </objects>

 </oval_definitions>
 <status:Status xmlns:status="http://jvndb.jvn.jp/myjvn/Status" version="3.0"
 method="getOvalData" lang="en" retCd="0" retMax="10000" errCd="" errMsg=""
 totalRes="1" totalResRet="1" firstRes="1" />
</Result>
```

図 7：OVAL 定義データ取得のレスポンス例

getOvalList では，図 6に示すようなチェックの対象となる製品の OVAL 定義一覧を取得する．また，getOvalData では，図 7に示すように製品インストールを判定するレジストリ設定値データを取得する．図 7の場合，レジストリキーの存在を，<registry_object>を用いて確認する．

(2) 脆弱性対策概要情報表示 (図 8の右パネル)
一覧から選択した製品の CPE 名をフィルタリング条件として設定し，脆弱性対策概要情報を取得した RSS+mod_sec で記載された XML 形式の結果を表示する．脆弱性対策概要情報の一覧から項目を選択すると，ブラウザ上に当該項目の脆弱性対策詳細情報を表示する．

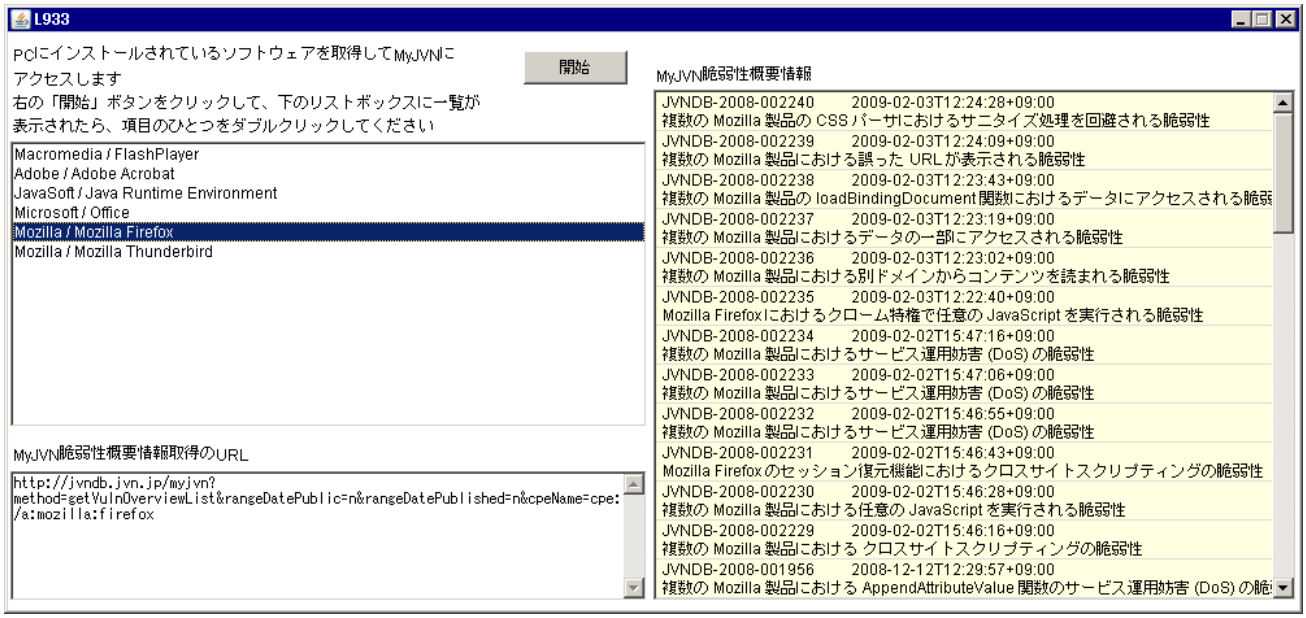

図 8：フィルタリング条件設定機能付き製品の脆弱性対策情報チェックツール

左図：MyJVN Web API のメソッド getVulnOverviewList を用いた脆弱性対策概要情報一覧取得
右図：MyJVN Web API のメソッド getProductList を用いた製品一覧取得
図 9：Web サービス API 活用促進のための参照実装例

6 おわりに

本稿では，脆弱性対策に関わる処理の機械化を目指すフレームワーク MyJVN の課題について，脆弱性対策情報の流通基盤と脆弱性対策情報を用いたサービスの視点から示した．また，課題を解決するために，セキュリティ問題をチェックする手続き仕様 OVAL を用いた脆弱性対策に関わる処理の機械化と，Web サービス API の拡張について提案すると共に，開発したプロトタイプシステムについて報告した．

脆弱性対策に関わる処理の機械化を推進するにあたり，脆弱性対策情報の流通基盤の活用，特に，提供している Web サービス API の活用が重要であると考えている．

今後は，図 9に示すような Web サービス API 活用促進のための参照実装を示しながら，脆弱性対策に関わる処理の機械化を目指すフレームワーク MyJVN の普及推進を進めていく予定である．

謝辞

本研究を進めるにあたって有益な助言と協力を頂いた，高崎仁氏ならびに，IPA の関係者各位に深く感謝致します．

参考文献

1) 寺田，杉山他, "脆弱性対策情報の利活用基盤 MyJVN の提案", CSS2008 (2008)
2) 共通脆弱性識別子 CVE 概説
http://www.ipa.go.jp/security/vuln/CVE.html
3) 共通脆弱性評価システム CVSS v2 概説
http://www.ipa.go.jp/security/vuln/SeverityCVSS2.html
4) 共通脆弱性タイプ一覧 CWE 概説
http://www.ipa.go.jp/security/vuln/CWE.html
5) 共通プラットフォーム一覧 CPE 概説
http://www.ipa.go.jp/security/vuln/CPE.html
6) NIST: The Information Security Automation Program and The Security Content Automation Protocol,
http://nvd.nist.gov/scap.cfm
7) Karen et.al, “The Common Configuration Scoring System (CCSS)”, NIST Interagency Report 7502 (2008),
http://csrc.nist.gov/publications/drafts/nistir-7502/Draft-NISTIR-7502.pdf
8) CAPEC: Common Attack Pattern Enumeration and Classification, http://capec.mitre.org/
9) CEE: Common Event Expression,
http://cee.mitre.org/
10) CRF: Common Result Format,
http://makingsecuritymeasurable.mitre.org/crf/
11) 寺田，高田，土居, “脆弱性対策情報データベース JVN の提案”, 情処論文誌 Vol.46 No.5 (2005)
12) 経済産業省,「情報セキュリティ早期警戒パートナーシップ」の運用開始について
http://www.meti.go.jp/policy/it_policy/press/0005399/
13) JVNRSS: JVN RDF Site Summary,
http://jvnrss.ise.chuo-u.ac.jp/jtg/jvnrss/
14) mod_sec: Qualified Security Advisory Reference,
http://jvnrss.ise.chuo-u.ac.jp/jtg/mod_sec/
15) VULDEF: The VULnerability Data publication and Exchange Format data model,
http://jvnrss.ise.chuo-u.ac.jp/jtg/vuldef/
16) OVAL: Open Vulnerability Assessment Language
http://oval.mitre.org/