21st Annual FIRST Conference on Computer Security Incident Handling
June 28–July 03, 2009 - Kyoto, Japan

Proposal of MyJVN (Web Service APIs) for Security Information Exchange infrastructure

Masato Terada[1)2)3)], **Ken Sugiyama**[1)], **Yoshiaki Saito**[1)]
Tadashi Yamagishi[1)], **Hideaki Kobayashi**[1)] **and Norihisa Doi**[2)]

Abstract
Unauthorized access intended to distribute malware has been widely spread across the Internet and causing a lot of damage worldwide. In order to eliminate vulnerabilities that can be exploited by those malware and prevent unauthorized access, it is necessary to improve the way to distribute security information about computer software and hardware. In this paper, we examine how we can provide a more efficient security information distribution service for the security administrators that helps them reduce their workload related in gathering information from various sources and take care of vulnerabilities and incidents.

We propose MyJVN that is the framework of Web service APIs with the common enumeration based as security information sharing and exchanging. Currently, JPCERT/CC and IPA (Information-technology Promotion Agency) are promoting a framework to handle vulnerability information in Japan. They offer JVN (Japan Vulnerability Notes), a portal site to provide security information about the domestic computer software and hardware manufactured by the vendors participating in the framework.

MyJVN is one of the methods JVN has been using to distribute security information. MyJVN has Web service APIs that are based on CPE (Common Platform Enumeration) as a structured naming scheme for products to correlate security information issued by various sources. In this paper, firstly we will explain the specification and applications of MyJVN. Secondly, we will introduce our feasibility study on MyJVN.

Keywords
Network Security, Vulnerability, Information Sharing, Web service API

1. Introduction

Recently, malware (Viruses, Worms, Trojan Horses and Bots etc.) propagation is widely seen and causing a lot of damage broadly in various ways. In order to prevent unauthorized access and eliminate the vulnerabilities in the information systems, it is necessary to improve the security information sharing to enable users to take appropriate actions.

In Japan, July 2004, the Ministry of Economy, Trade and Industry (METI) adopted the "Standard for Handling Software Vulnerability Information" as an official rule, and began promoting the "Information Security Early Warning Partnership" as an implementation framework [1]. The JVN has become the portal site to provide the security information about the domestic computer software and hardware manufactured by the vendors that are registered to the framework [2].

Currently, JVN has the two functions that are vulnerability handling coordination database (JVN, http://jvn.jp/en/index.html) and vulnerability archiving database (JVN iPedia, http://jvndb.jvn.jp/en/) (Figure 1). The former as the vulnerability handling coordination database stores the results of coordination with JPCERT/CC and product vendors. The latter as the vulnerability archiving stores the vulnerability countermeasure information that covers the international and Japan regional fields. And this security information is distributed in the form of a HTML-based web page. This means that fragmented information from various websites are collected and reassembled, and considerable time and efforts are required to reestablish the connections and relationships among the various bits and pieces of information. In short, it is necessary to improve the method of the security information exchange. From the information provider’s point of view, if information could be published in a form more easily processed by a machine, then information could be reused much more flexibly and extensively.

1) Security Center, IPA (Information-technology Promotion Agency, Japan)
2) Graduate School of Science and Engineering, Chuo University
3) JVN associate staff, JPCERT/CC
Hitachi Incident Response Team, Hitachi Ltd.

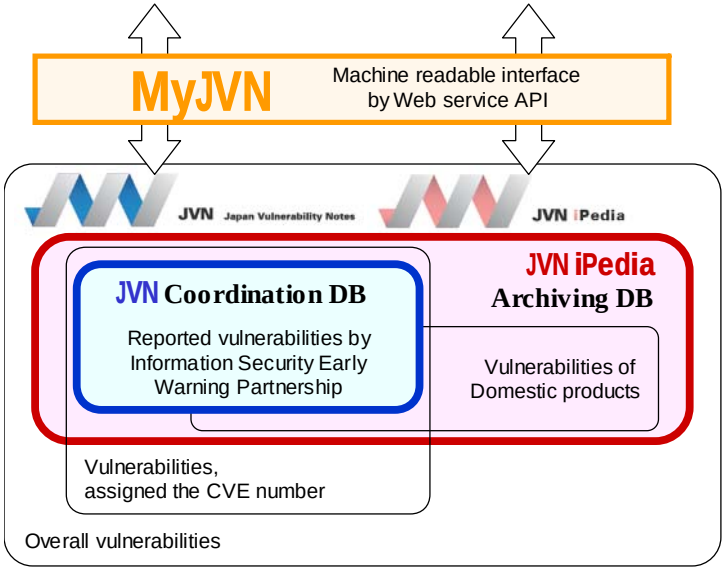

Figure 1 : Functional architecture of JVN.

We propose MyJVN[3] that is the framework of Web service APIs with the common enumeration based as a security information sharing and exchanging. This paper discusses the framework of MyJVN and its applications. At the end, we will introduce the feasibility study on MyJVN.

2. Related Work

The related researches on the security information exchange specification of vulnerability database are the followings.

NIST NVD [4]
National Vulnerability Database (NVD) is a comprehensive cyber security vulnerability database and refers to CVE(Common Vulnerabilities and Exposures)[5], CVSS(Common Vulnerability Scoring System)[6], CPE(Common Platform Enumeration)[7], CWE(Common Weakness Enumeration)[8]. It provides search capability and directs users to vulnerability and patch information. And also it provides the XML export function as machine-readable format [9].

OSVDB [10]
Open Source Vulnerability Database (OSVDB) is a vendor-neutral vulnerability database for the information security community. The goals of the project are to promote a greater and more open collaboration between companies and individuals, eliminate redundant works and reduce expense inherent in the system and product development. OSVDB supports the APIs that will return a list of OSVDB ID's, or regular results, which are in XML (See Table 1).

Table 1 : Web Service APIs provided by OSVDB.

Class	Description
Mapping API	These queries map a given value to an OSVDB_ID, and return either XML in the case of regular queries, or text (csv) for simple queries.
Core Queries	These functions map a vendor, for example, to their vulnerabilities, and return XML regarding the vulnerabilities.
Supporting Queries	These functions map ID values returned from other queries to supporting information within OSVDB, like vendors, products, classifications, etc.

Making use of MyJVN is an essential point in the security information exchange, for this handily resolves the following three basic issues:

- **Distribution designed to encourage reuse of information**
  Our primary objective is to aggregate security information from the product vendors and provide it through the JVN website. But in order to reuse published information, it must be offered in a machine-readable format. This is where RSS comes in. By using RSS, JVN data can be distributed in the same way just as the news feeds provided by the news websites. And

because the content is described by RSS, it can be easily verified if information has been added to an item or an item has been updated.

- **Easy confirmation procedure to vulnerability countermeasure**
  Second objective is easier to check the vulnerability countermeasure information announced.
- **Establishment of security information exchange infrastructure**
  There is a limit of HTML provider based approach for security information exchange. We should provide the possibility of the security information exchange customized to users.

3. Proposal of MyJVN as security information exchange infrastructure

The purpose of framework MyJVN is to provide the security information exchange infrastructure in consideration of the vulnerability management and the countermeasure automation. In first step, we establish a platform identifier, XML formats and Web service APIs to achieve the framework MyJVN. Next, we implement a filtered security information service as the point of view of "What kind of vulnerability exists in the product?".

3.1 Components of framework MyJVN

In this section, we describe a platform identifier, XML formats and a Web service APIs in the framework MyJVN.

(1) Platform identifier

It is important to distinguish the software package being used, because the impacts of the vulnerability are different in the version even if they are the distinction of the software package being installed and the same software for the vulnerability countermeasure. So, MyJVN promotes the CPE for the software package distinction. CPE is a structured naming scheme for information technology systems, platforms and packages (Figure 2). Based upon the generic syntax for Uniform Resource Identifiers, CPE includes a formal name format, a language for describing complex platforms, a method for checking names against a system, and a description format for binding text and tests to a name.

```
cpe:/{part}:{vendor}:{product}:{version}:{update}:{edition}:{language}
```

Figure 2 : CPE Name Basic Structure.

(2) XML formats

In order to gather the information and perform the relationship between the gathered information, it is necessary to improve the method of the security information sharing. If the security information is machine readable, many Internet sites can reduce the cost of information gathering. Our security information sharing proposes the XML formats as to approach solving these problems. JVNRSS is the overview XML format based on RSS and VULDEF is the detail XML format (Figure 3). Also, mod_sec is RSS Extension of security information distribution, and definition of the tags for RSS 1.0, 2.0 and Atom [11].

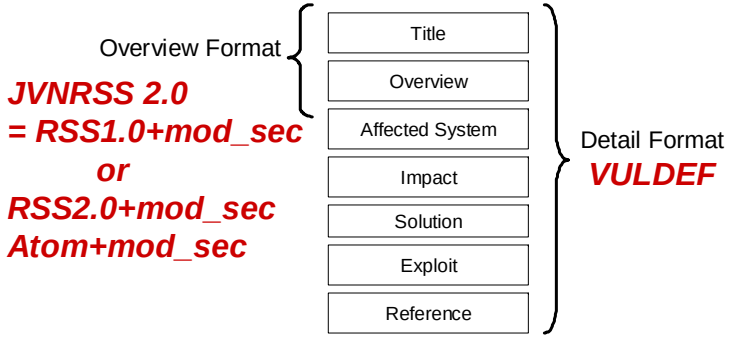

Figure 3 : XML format for Overview(JVNRSS) and Detail(VULDEF) of vulnerability countermeasure information.

(a) JVNRSS(JVN RDF Site Summary)[12]
JVNRSS 2.0 is based on RSS 1.0 and uses the field <sec:references> of the mod_sec as a primary key to group security information. Also it uses the field <dcterms:issued> <dcterms:modified> of the Qualified Dublin Core and <sec:identifier> of the mod_sec. RSS contains a list of items, each of which is identified by a link. Each item can have any amount of metadata associated with it. The most basic metadata supported by RSS includes a title for the link and a description of it. We use JVNRSS as overview XML format to distribute the vulnerability countermeasure information in MyJVN.

(b) mod_sec(Qualified Security Advisory Reference)[13]
The mod_sec is RSS Extension of security information distribution, and definition of the tags for RSS 1.0, 2.0 and Atom. We propose new tags <sec:cvss> and <sec:cpe-item> such as common enumeration to easier establish security information exchange infrastructure (Figure 4).

```
xmlns:sec="http://jvn.jp/rss/mod_sec/"
xsi:schemaLocation="http://jvn.jp/rss/mod_sec/
                    http://jvndb.jvn.jp/schema/mod_sec_2.0.xsd">

<sec:identifier>Unique identifier assigned by vendor</sec:identifier>
<sec:references>Best reference to a related security information</sec:references>
<sec:cvss score="Overall score"
          severity="Severity level (High - Medium - Low)"
          vector="Value of each vector in CVSS"
          version="CVSS version" />
<sec:cpe-item name="CPE Name">
  <sec:vname>Vendor Name</sec:vname>
  <sec:title>Product Name</sec:title>
</sec:cpe-item>
```

Figure 4 : RSS Extension of security information distribution (mod_sec).

(c) VULDEF(The VULnerability Data publication and Exchange Format data model)[14]
The purpose of “The VULnerability Data publication and Exchange Format data model” is to define data formats for the information related to security advisory typically published by the product vendors and Computer Security Incident Response Teams. VULDEF has some elements to describe the vulnerability, affected item and solution etc.

(3) Web service APIs
With the Web Service APIs, many user sites can build applications to perform tasks. We would like to establish the Web Service APIs for Security Information Exchange infrastructure. In the first step, we propose the following Web Service APIs. The four basic request methods and the JVNRSS response format of “method=getVulnOverviewList” are shown in Table 2 and Figure 5.

Table 2 : The four basic methods in Web Service APIs of MyJVN.

Class	Description
getVendorList	The vendor list that is filtered by the CPE is acquired in XML format [15]. http://jvndb.jvn.jp/myjvn?method=getVendorList&cpeName=cpe:/*:j*&lang=en
getProductList	The product list that is filtered by the CPE is acquired in XML format [15]. http://jvndb.jvn.jp/myjvn?method=getProductList&cpeName=cpe:/*:sony:*&lang=en
getVulnOverviewList	The vulnerability overview list that is filtered by the CPE is acquired in JVNRSS(RSS + mod_sec) format. http://jvndb.jvn.jp/myjvn?method=getVulnOverviewList&cpeName=cpe:/*:fujitsu:*&rangeDatePublic=n&rangeDatePublished=n&lang=en
getVulnDetailInfo	The vulnerability detail information is acquired in VULDEF format. http://jvndb.jvn.jp/myjvn?method=getVulnDetailInfo&vulnId=JVNDB-2009-000004&lang=en

```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns="http://purl.org/rss/1.0/"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:dcterms="http://purl.org/dc/terms/"
   xmlns:sec="http://jvn.jp/rss/mod_sec/"
   xsi:schemaLocation="http://purl.org/rss/1.0/ http://jvndb.jvn.jp/schema/jvnrss_2.0.xsd">
   <channel rdf:about=" http://jvndb.jvn.jp/myjvn?method=getVulnOverviewList">
      <title>JVN vulnerability OVERVIEW list </title>
      <link>http://jvndb.jvn.jp/en/apis/myjvn</link>
      <description>JVN vulnerability OVERVIEW list</description>
      <items>
         <rdf:Seq>
            <rdf:li rdf:resource
                  ="http://jvndb.jvn.jp/en/contents/2009/JVNDB-2009-000004.html"/>
         </rdf:Seq>
      </items>
   </channel>
   <item rdf:about="http://jvndb.jvn.jp/en/contents/2009/JVNDB-2009-000004.html">
      <title>About Web Service APIs of MyJVN</title>
      <link>http://jvndb.jvn.jp/en/contents/2009/JVNDB-2009-000004.html </link>
      <sec:identifier>JVNDB-2009-000004</sec:identifier>
      <sec:references>http://jvn.jp/en/jp/JVN66828183/index.html </sec:references>
      <sec:references>http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-5941
         </sec:references>
      <sec:cvss score="2.6" severity="Low" vector="" version="2.0" />
      <sec:cpe-item name="cpe:/a:jvn:jvndb">
         <sec:vname>JVN</sec:vname>
         <sec:title>Japan Vulnerability Notes</sec:title>
      </sec:cpe-item>                                                    mod_sec
      <dc:publisher>IPA</dc:publisher>
      <dc:date>2008-10-10T10:10+00:00</dc:date>
      <dcterms:issued>2008-10-10T10:10+00:00</dcterms:issued>          Qualified
      <dcterms:modified>2008-11-11T11:11+00:00</dcterms:modified>      Dublin Core
   </item>
</rdf:RDF>
```

Figure 5 : JVNRSS response format of method=getVulnOverviewList.

3.2 Filtered Security Information Service

In this section, we describe a Filtered Security Information Service that combined with the front end GUI and Web Service APIs of MyJVN (http://jvndb.jvn.jp/en/apis/myjvn/). Filtered Security Information Service has four components (JVN DB, CPE DB, Web Service APIs and SWF) that are shown in Figure 6.

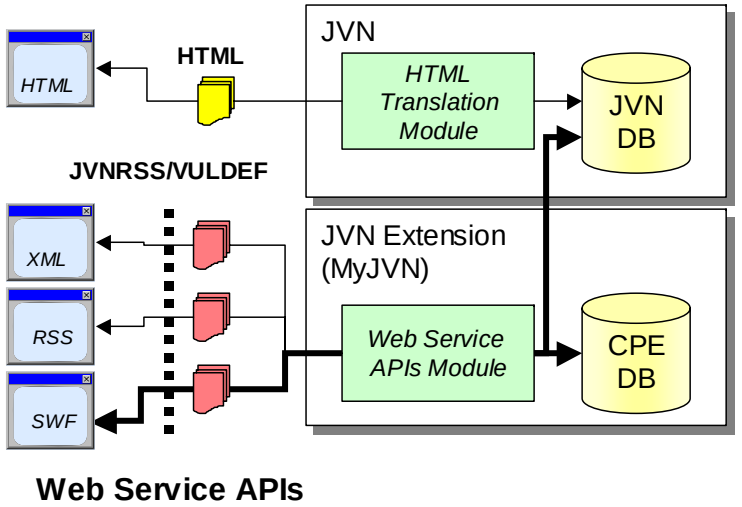

Figure 6 : System Overview of the Filtered Security Information Service.

(1) JVN DB

JVN DB is the vulnerability archiving database that stores the vulnerability countermeasure information which cover the international and Japan regional fields.

(2) CPE DB

CPE DB is mapping database that stores the mapping of the product entries of JVN DB and CPE names.

(3) Web Service APIs Module

We implemented the four functions (getVendorList, getProductList,

getVulnOverviewList and getVulnDetailInfo) as basic Web Service APIs.

(4) Flash GUI (SWF) as the front end GUI

In shown Figure 7, Flash GUI is the front end GUI as reference implementation of Web Service APIs. This Flash GUI has the grouping module that finds <sec:cpe-item> with the same name in the response of getVulnOverviewList method, it makes these <items> into a same group. When "VEND" is selected, the displayed listing will have the following priority: vendor name (ascending order) > product name (ascending order) > last updated (descending order) > vulnerability countermeasure information ID (descending order).

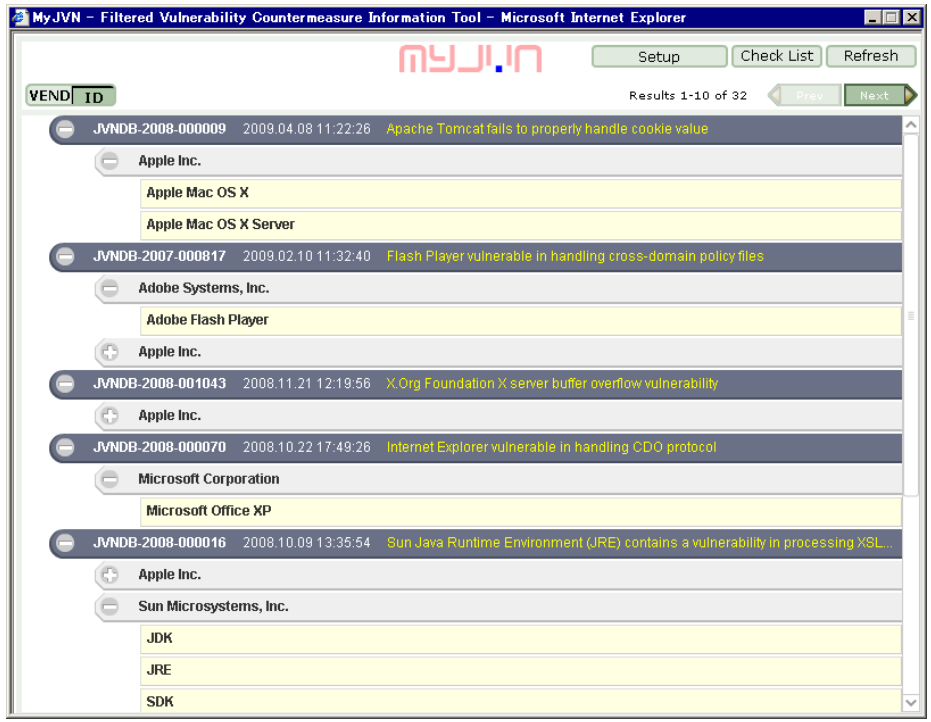

Figure 7 : Filtering Result Window
by getVulnOverviewList Web Services API.

And Figure 8 is the initial setup wizard of the filtering condition. First, from the vendor listing, you select the vendor of the product for which you wish to obtain information. After the vendors have been selected, at the product selection window, you select the product name from the product listing. In this setup wizard uses getVendorList method for the acquirement of vendor information and getProductList method for the acquirement of product information.

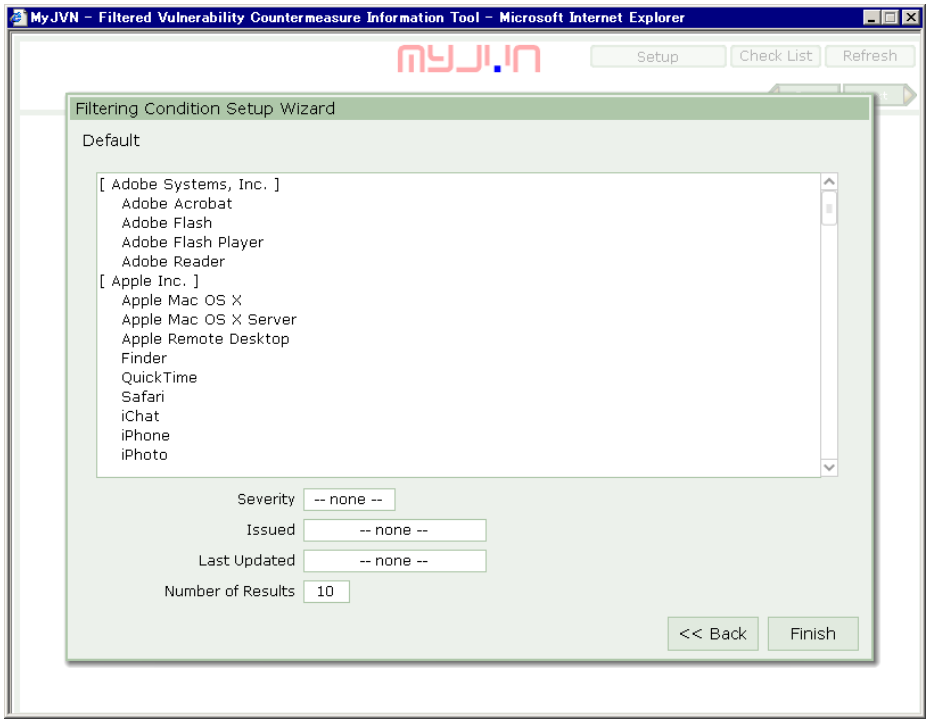

Figure 8 : Initial setup wizard of Flash GUI
for Filtered Security Information Service.

In shown Figure 9, this window is displayed when one of the vulnerability countermeasure information with VULDEF format listed in the filtering result window is clicked.

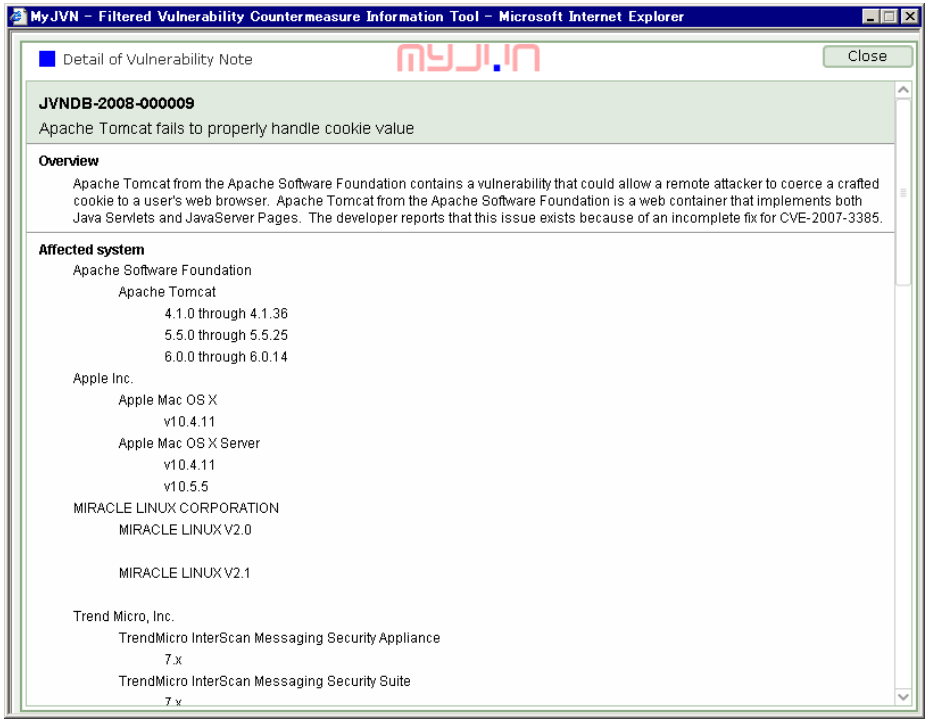

Figure 9 : Detailed Vulnerability Countermeasure Information Window by getVulnDetailInfo Web Services API.

4. Feasibility Study of OVAL light-weight interpreter

In above section, we described the Filtered Security Information Service with the manual configuration. In this section, we consider the filtered security information service with the automatic configuration possibilities using OVAL definition files.

(1) OVAL

OVAL (Open Vulnerability and Assessment Language) is an international, information security, community standard to promote open and publicly available security content, and to standardize the transfer of this information across the entire spectrum of security tools and services [16].

In feasibility study, we implement OVAL light-weight interpreters to check the version number of installed products on your PC. Also, we combine the filtered security information service with OVAL light-weight interpreter to achieve the automatic configuration possibilities of the filtering condition.

(2) SWF-based version checker

We introduce SWF-based OVAL light-weight interpreter (SWF interpreter). This SWF interpreter is Flash application program and provides the function of version number check of Flash Player using OVAL definition file. Figure 10 is a <states> part of OVAL definition file using by SWF interpreter which extracts a version number of itself and compares it with a value of <file_state id="oval:jp.ac.chuo-u.ise.jvnrss.oval:ste:1001">.

```
<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>
```

Figure 10 : OVAL definition for Flash Player version check.

In shown Figure 11, we show the results of version number check by SWF interpreter. The Figure of left is to same value of version number of OVAL definition. The Figure of right is to different value that means old Flash Player version. These results lead the automatic configuration possibilities in the Filtered Security Information Service by OVAL.

The latest Flash Player version installed.

The old Flash Player version installed.

Figure 11 : The results of version check by SWF interpreter.

(3) JAR-based version checker

JAR-based OVAL light-weight interpreter (JAR interpreter) provides the function of version number check of some installed programs using OVAL definition files. Figure 12 is a part of OVAL definition file for Flash Player by JAR interpreter. JAR interpreter acquires a registry data (CurrentVersion value) from the registry repository by <registry_object id="oval:jp.ac.chuo-u.ise.jvnrss.oval:obj:1002"> and compares it with a value in <registry_state id="oval:jp.ac.chuo-u.ise.jvnrss.oval:ste:1002">.

In shown Figure 13, we show the results of version check by JAR interpreter. The character (O:) of the upper side in box is a value of <registry_state> of OVAL definition. The character (R:) of the bottom side in box is a registry data of the registry repository of Microsoft Windows system. Also, the character circle (O) of DirectX and Windows XP show a registry data is same version number of OVAL definition. The character cross(X) is to different value that means old Flash Player and JRE version. The character minus (-) of Netscape shows that application program is not installed. These results lead the automatic configuration possibilities in the Filtered Security Information Service, too.

```
<oval_definitions>
 <generator>
  <oval:schema_version>5.4</oval:schema_version>
  <oval:timestamp>2008-06-14T12:18:00+09:00</oval:timestamp>
 </generator>
 <definitions>
  <definition id="oval:jp.ac.chuo-u.ise.jvnrss.oval:def:1200800001002"
   class="vulnerability" version="1">
   <metadata>
    <title>Flash Player Latest Version Check</title>
    <affected family="windows">
     <product>Flash Player</product>
    </affected>
    <reference source="CPE" ref_id="cpe:/a:adobe:flash_player"
     ref_url="http://www.adobe.com/jp/support/flashplayer/ts/documents/tn_15507.htm" />
    <description>Adobe (Macromedia) Flash Player Latest Version Check</description>
   </metadata>
   <criteria comment="FlashPlayer" operator="AND">
    <criterion comment="FlashPlayer (Latest version 9.x) is not installed"
     test_ref="oval:jp.ac.chuo-u.ise.jvnrss.oval:tst:1002" />
   </criteria>
  </definition>
 </definitions>
 <tests>
  <registry_test id="oval:jp.ac.chuo-u.ise.jvnrss.oval:tst:1002" version="1"
   comment="a version of FlashPlayer 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:1002" />
   <state state_ref="oval:jp.ac.chuo-u.ise.jvnrss.oval:ste:1002"/>
  </registry_test>
 </tests>
 <objects>
  <registry_object id="oval:jp.ac.chuo-u.ise.jvnrss.oval:obj:1002" version="1"
   comment="This registry key identifies the FlashPlayer current version."
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">
   <hive>HKEY_LOCAL_MACHINE</hive>
   <key>SOFTWARE¥Macromedia¥FlashPlayer</key>
   <name>CurrentVersion</name>
  </registry_object>
 </objects>
 <states>
  <registry_state id="oval:jp.ac.chuo-u.ise.jvnrss.oval:ste:1002" version="1"
   comment="The registry key has a value of FlashPlayer"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">
   <value>9,0,124,0</value>
  </registry_state>
 </states>
</oval_definitions>
```

Figure 12 : OVAL definition for Flash Player version check.

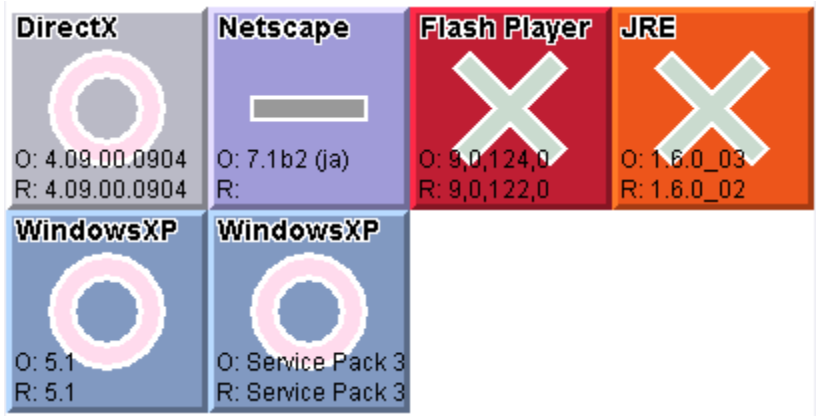

Figure 13 : The results of version check by JAR interpreter.

(4) JAR-based automatic configuration & filtering tool

We show the proof of concept prototype for the Filtered Security Information Service with the automatic configuration. This system has five components (JVN DB, CPE DB, OVAL DB, Web Service APIs extension and JAR) that are shown in Figure 14. JVN DB and CPE DB are same Dada bases in Figure 6.

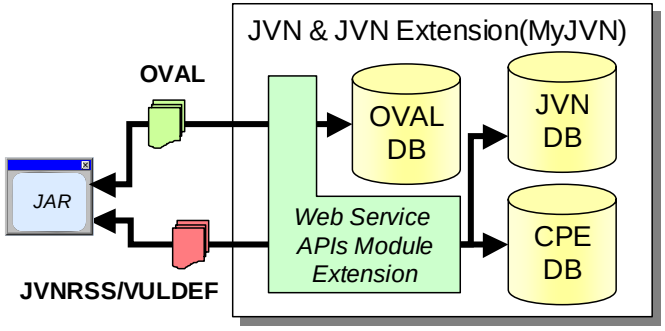

Figure 14 : Prototype overview of the Filtered Security Information Service with automatic configuration.

Web Service APIs module extension supported the two following APIs newly (Table 3). OVAL DB has the OVAL definition files of some application programs and provides that files by the Web Service APIs module extension to JAR-based automatic configuration tool.

Table 3 : The new two methods of Web Service APIs in extension.

Class	Description
getOvalList	The OVAL definition list is acquired in XML format [15]. http://jvndb.jvn.jp/myjvn?method=getOvalList&platform=cpe:/o:microsoft:windows-nt:vista
getOvalData	The OVAL definition file is acquired in OVAL format. http://jvndb.jvn.jp/myjvn?method=getOVALdata&ovalid=oval:jp.ac.chuo-u.ise.jvnrss.oval:def:9009

In shown Figure 15, JAR-based automatic configuration & filtering tool (JAR filtering tool) is the front end GUI that includes the OVAL light-weight interpreter for the version check function and the implementation of Web Service APIs extension. JAR filtering tool of this prototype downloads some OVAL files from OVAL DB via getOvalList and getOvalData methods, and checks whether the targeted application programs are being installed. "Start" button is clicked in Figure 15, then JAR filtering tool begins to check the installed application programs by OVAL definition files. And JAR filtering tool displays the installed application programs in left side window.

Also, JAR filtering tool extracts a CPE name from <reference source="CPE" ref_id="cpe:/a:sun:jre" ref_url="http://java.com/ja/" /> in OVAL definition file and issues the getVulnOverviewList method to display the vulnerability countermeasure information related with JRE (Sun Java Runtime Environment) in the right side window. And this window invokes the your browser window that is displayed when one of the vulnerability countermeasure information with VULDEF format listed in the filtering result window is double clicked.

5. Conclusion

We propose MyJVN that is proof of concept of Web service APIs with the common enumeration based as security information sharing and exchanging. This paper has discussed the framework of MyJVN and the Filtered Security Information Service as its applications.

Furthermore, we introduced our feasibility study of the automatic configuration possibilities in the Filtered Security Information Service by OVAL definition.

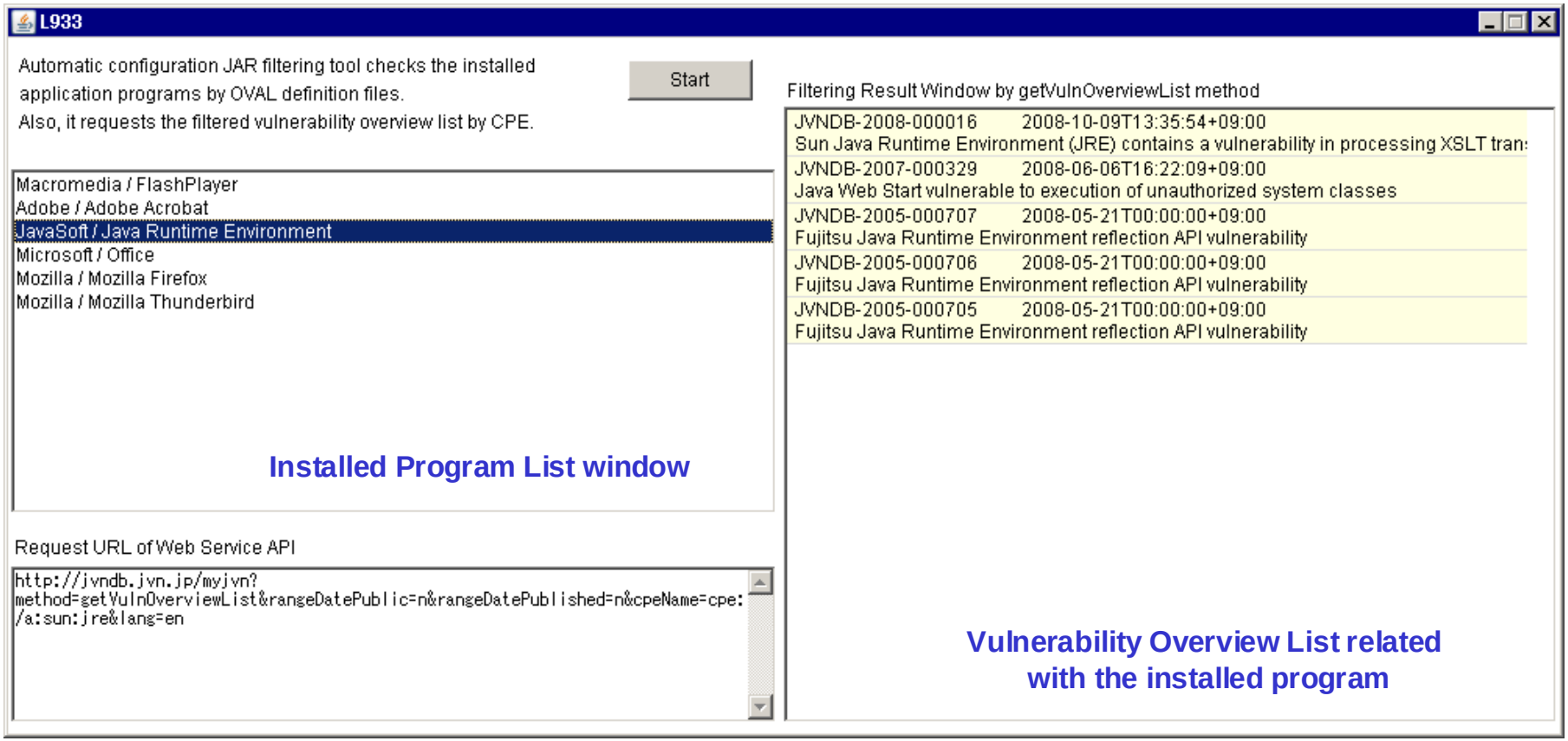

Figure 15 : JAR interpreter with automatic configuration combined with the Filtered Security Information Service.

Acknowledgements

The authors are grateful to Hiroshi Takasaki and Chika Nukui for their assistance, and to their colleagues in the IPA and JPCERT/CC for their insightful comments.

References

1) Information Security Early Warning Partnership, http://www.ipa.go.jp/english/security/third.html
2) JVN: Japan Vulnerability Notes, http://jvn.jp/en/
3) MyJVN, http://jvndb.jvn.jp/en/apis/myjvn/
4) NVD: National Vulnerability Database, http://nvd.nist.gov/
5) CVE: Common Vulnerabilities and Exposures, http://cve.mitre.org/
6) CVSS: Common Vulnerability Scoring System, http://www.first.org/cvss/
7) CPE: Common Platform Enumeration, http://cpe.mitre.org/
8) CWE: Common Weakness Enumeration, http://cwe.mitre.org/
9) NVD Data Feed and Product Integration, http://nvd.nist.gov/download.cfm
10) Open Source Vulnerability Database, http://www.osvdb.org/
11) Masato Terada, Shingo Takada, Junji Fukuzawa, Norihisa Doi: “Proposal of RSS Extension for Security Information Exchange”, 18th Annual FIRST Conference (Baltimore, Maryland, United States, June 25 - 30, 2006), http://www.first.org/conference/2006/papers/terada-masato-papers.pdf
12) JVNRSS: JVN RDF Site Summary, http://jvndb.jvn.jp/en/schema/jvnrss.html
13) mod_sec: Qualified Security Advisory Reference, http://jvndb.jvn.jp/en/schema/mod_sec.html
14) VULDEF: The VULnerability Data publication and Exchange Format data model, http://jvndb.jvn.jp/en/schema/vuldef.html
15) XML Schema for the product list of MyJVN, http://jvndb.jvn.jp/schema/productlist_2.0.xsd
16) OVAL: Open Vulnerability and Assessment Language, http://oval.mitre.org/