# Toward Automated Vulnerability Monitoring using Open Information and Standardized Tools

Takeshi Takahashi*, Daisuke Miyamoto‡, Koji Nakao*
*National Institute of Information and Communications Technology, Tokyo, Japan
‡The University of Tokyo, Tokyo, Japan
E-mail: takeshi_takahashi@ieee.org

***Abstract*—To maintain acceptable levels of security, organizations must manage their IT assets and related vulnerabilities. However, this can be a considerable burden because their resources are often limited. This paper introduces a technique and system architecture that monitor the vulnerability of the IT assets on an organization's administrative networks. We use open information and standardized, non-proprietary tools in order to bolster cybersecurity capability for a wide range of organizations. In the proposed system, an agent module installed on each IT asset sends information to its server, while the server also probes the network to collect information on agentless IT assets. The server then converts the information into standard identifiers, which are used to query open repositories to obtain vulnerability information. The system provides an alert when vulnerability information pertaining to the IT asset is identified. This paper also introduces a prototype system, with which we analyze and discuss the proposed technique and system and clarify issues to be solved in our future work.**

## I. Introduction

Security is increasingly recognized as an important agenda for many organizations. To maintain acceptable levels of security, organizations must continually manage the vulnerabilities of their IT assets. To manage vulnerabilities, organizations must first compile list of IT assets on their administrative networks. Indeed, the importance of listing IT assets inside organizations is already specified by Information Security Management System (ISMS) [1] and other management practices. Many large organizations have already taken appropriate measures to achieve this goal, but there are still many organizations that have been unable to do so. After the current status of IT assets is obtained, it is necessary to keep track of their changes in real time.

Each organization should continually monitor assorted vulnerability information pertaining to its IT assets. Some commercial software provides automated patches, but it is not always desirable to allow their application. It is desirable to understand the vulnerability and determine needed actions. Nevertheless, finding and identifying needed vulnerability information takes a considerable amount of time, because organizations must find such information from the web, which contains vast amounts of usable and non-usable information.

To cope with these issues without increasing human resources, it is necessary to automate and streamline IT asset and vulnerability management operations. When addressing this problem, we will use open tools and data so that resource-constrained organizations can also take advantage of the output of this research. Indeed, more and more open data are becoming available online in these days, and we could get the most out of it by using standardized tools appropriately.

In this study, we contribute to the automation of IT asset vulnerability management on administrative networks, using open data and standardized tools. We propose a technique and system architecture that collect IT asset information, determine its identifiers, then identify relevant vulnerability information. First, the proposed technique collects information from IT assets by receiving information from agents installed on terminals and by actively probing the terminals. Second, the collected information is structured and accumulated, then converted into IT asset identifiers, so that we can easily locate and compare this information later on. Third, the proposed technique queries vulnerability repositories about the identifiers. If vulnerability information pertaining to the identifiers is found, an alert message is immediately generated, and sent to the administrator of the organization so that immediate corrective action can be taken[1]. Note that the system we propose does not replace traditional security systems and solutions. On the contrary, it is a complimentary system, as we believe our system leverages on those systems. The proposed technique is unique because it automatically determines standardized IT asset identifiers from the collected IT asset information, and uses these identifiers as keys to query vulnerability information.

We implemented a prototype of the proposed system, with which we show that the proposed technique can identify vulnerability information pertaining to an organization's IT asset. This information is then used to notify the system administrator. This leads to an efficient security operation inside an organization.

## II. Design Principles

This section describes the design principles of our system by listing requirements and describing the necessary tools.

### A. Requirements

**Req 1: Avoid manual operation.** This is the prerequisite of this research, and we follow it strictly.

[1]We are considering autonomous triage initiated by the alert message; for instance, the vulnerable terminal in question can be pruned from the network. This is outside the scope of this paper and will be studied in our future work

**Req 2: Use open information sources as much as possible.** Some information such as the system information of each organization should be kept confidential, but some information, such as vulnerability information, should be reused by multiple organizations and thus be public. Fortunately, there are public vulnerability information repositories. Although the use of private and commercial information may increase the quality of vulnerability management, we wish to reinforce the security level of a wide range of organizations. Thus, we use open information sources in this research.

**Req 3: Use standardized tools and techniques.** We use standardized tools and techniques so that a wide range of organizations can implement and further develop our work.

**Req 4: Each organization interprets its vulnerability information and judges its security situation.** We only deliver the necessary information for vulnerability management. The interpretation of the information and the judgment of security situations are left to the information consumer, i.e., each organization. At most, we could automate the triage operation that is followed prior to each organization's judgment of a security situation, but this is outside the scope of this paper.

### B. Necessary Tools

*1) Standardized Tools for Information Identification:* Our study uses security identifiers. Common Platform Enumeration (CPE) [2] and Common Vulnerabilities and Exposures (CVE) [3] provide identifiers, i.e., CPE identifiers (CPE-ID) and CVE identifiers (CVE-ID). They facilitate identification and retrieval of security information. A **CPE-ID** identifies classes of applications, operating systems (OSs), and hardware devices present among an enterprise's computing assets. The CPE specification provides a well-formed CPE data model, URI binding, and formatted string binding, but we use URI binding for describing a CPE-ID because it concisely creates identifiers and because the National Vulnerability Database (NVD) [2], one of the largest online vulnerability repositories, uses it to list "vulnerable software and versions." Along with the CPE specifications, a list of CPE identifiers is published online as an official CPE dictionary[3]. We could easily locate CPE-IDs using the dictionary. A **CVE-ID** identifies publicly known cybersecurity vulnerabilities. The CVE-ID syntax is variable length and includes the CVE prefix, year, and arbitrary digits.

*2) Open Vulnerability Information:* Several organizations have published online vulnerability information repositories. Well-known examples include the NVD and Japan Vulnerability Notes (JVN) [4]. They use differing XML schemata to formulate the description of their vulnerability information. In contrast, we have created a cybersecurity knowledge base that links assorted online security information sources including NVD and JVN [5]. The knowledge base can link any type of security information described in well-formed XML and provide means to locate information across repositories on the web; users do not need to know the exact locations of the repositories.

[2]https://nvd.nist.gov/

[3]https://nvd.nist.gov/cpe.cfm

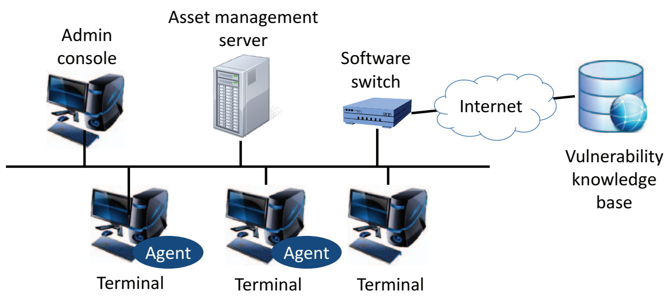


Fig. 1. Typical set up of the proposed system

## III. Automated Vulnerability Monitoring System

This section describes the fundamental schemes and algorithms needed to monitor vulnerability within our system.

### A. Role model

Figure 1 shows the typical set up of the proposed system. We define four types of roles: terminal, asset management server, vulnerability knowledge base, and administration console. A **terminal** is any terminal used by employees to carry out their duties inside an organization. It is assumed that in general, a software module called an "agent" is installed in the terminal, but there may be cases where a terminal may not have an agent installed. An **asset management server** is a server that stores asset information collected by the agents or collected by the server itself. This role also encompasses assigning an identifier for each piece of asset information and storing the identifier along with the asset information itself. Furthermore, this role communicates with the vulnerability knowledge base and decides if a vulnerability exists; if one does exist, it creates and sends an alert to the system administrator. A **vulnerability knowledge base** is a database that stores information related to vulnerability. It can be situated either inside or outside the administrative network, provided it can communicate with the asset management server. An **administration console** is a terminal used by the system administrator. It receives an alert when a vulnerability is discovered. In the proposed system, we have situated it on the same network as the other terminals, but we may situate it on a separate network, for example, on a mobile phone network. In addition, Figure 1 shows a software switch, which does not play any significant role in our current proposed scheme. We intend to use it for triaging and controlling network devices using software-defined networking (SDN) in our future work.

### B. Process Overview

Figure 2 shows the process flow of the proposed system. The process begins by collecting information on IT assets inside an organization's administrative networks. The system collects information by querying agents installed inside the IT assets, or by proactively probing the IT assets and the networks. It then generates IT asset identifiers from the IT asset information based on the techniques described in Section III-D. It then queries the vulnerability knowledge base about the identifiers, so that it can find vulnerability information pertaining to the IT assets. If any such information is found, the system evaluates the severity of the information and sends alerts to the administrator of the IT assets when needed. The system then waits for a next trigger to run this process again; the trigger could be a time out or any evincive instructions by administrators. This operation will continue so that the system can continuously monitor the vulnerabilities of the IT assets. Note that the detailed message exchange procedure and message formats are excluded from this paper and will be published in the future.

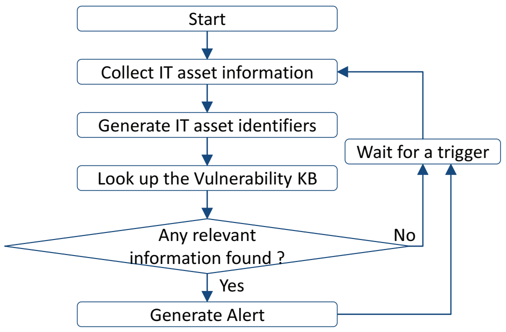

Start
Collect IT asset information
Generate IT asset identifiers
Look up the Vulnerability KB
Any relevant information found ?
No
Yes
Wait for a trigger
Generate Alert

Fig. 2. Process flow of the proposed system

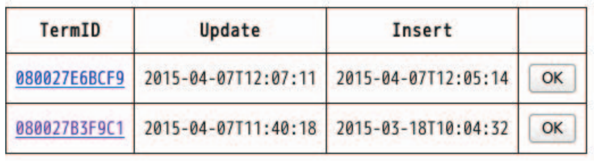
## C. IT asset information collection

The system collects information by communicating with agent applications installed inside terminals and/or by probing the network. The types of information the system collects include software name, version, and vendor name of the OS and installed applications, as well as IP and MAC addresses. The system structures and accumulates the collected asset information, using our own schema.

## D. IT Asset identifier determination

The algorithm mentioned in Section III-B generates IT asset identifiers to uniquely identify the IT assets. CPE-IDs are used for this. We use the information collected in Section III-C, i.e., software names, versions, and vendor names of OS and installed applications to build the query to locate information in the CPE dictionary. Because we typically find only a few exact matches, we use a numerical method to determine a CPE-ID. We introduce a parameter, called matching rate, which is calculated as the percentage of the characters that match the query. We choose the CPE-ID that shows the highest matching rate, although we judge that no CPE-ID is found if we do not find a CPE-ID with matching rate higher than a certain threshold value. Upon finding the CPE-ID, we add it to the asset information collected in Section III-C.

The accumulated IT asset information contains not only alphabetic characters, but also Japanese characters including Kanji, Hiragana, and Katakana. CPE supports multiple languages, and the CPE dictionary already contains Japanese entries. Therefore, even for asset information containing Japanese characters, it is possible to retrieve a corresponding CPE-ID.



| TermID | Update | Insert | |
|---|---|---|---|
| 080027E6BCF9 | 2015-04-07T12:07:11 | 2015-04-07T12:05:14 | OK |
| 080027B3F9C1 | 2015-04-07T11:40:18 | 2015-03-18T10:04:32 | OK |

Fig. 3. List of terminals

## E. Vulnerability information identification

Using the CPE-IDs determined in Section III-D, we query the vulnerability repository and locate vulnerability information. To facilitate locating information across assorted repositories, we use our cybersecurity knowledge base introduced in Section II-B2, with which we can access assorted vulnerability repositories, including the NVD. In our prototype, we retrieve only NVD information through the cybersecurity knowledge base for simplicity. We search for a match in the NVD file, inside the <vuln:product>element, and extract the CVE-IDs of the matched entries. If a matching NVD is found, it is determined that a vulnerability exists in that particular asset.

## IV. Discussion and Analysis

This section analyzes our system and discusses the challenges for the future research.

## A. Proof-of-concept Implementation

To demonstrate the feasibility of the proposed scheme and system architecture, we built a proof-of-concept implementation as a prototype. The prototype was designed for Windows 7 terminals, but it can function with terminals that use other platforms because it can collect information from them by probing the network and finding their fingerprints.

Figure 3 shows the administrators' web interface for managing IT assets. In this view, administrators can see the list of terminals on their networks. By clicking on any item in the list, detailed asset information is displayed for that particular terminal. Figure 4 shows the detailed asset information in XML. The XML contains the <cpe>tag, where the CPE-ID we determined using the technique mentioned in Section III-D is embedded. An IT asset reported as vulnerable lists the CVE-IDs we identified using the technique mentioned inSection III-E. We may send an alert message to an administrator by embedding the <cpe>and <cve>fields, though we omit the figure to conserver page space.

As can be seen, the prototype demonstrates that the proposed technique and system architecture are feasible.

## B. Consideration on Determining asset Identifier

One issue we anticipated was the language problem; we expected that it would be difficult to determine a CPE-ID for non-English IT asset information. Contrary to our expectations, this was not a significant issue because the CPE specifications and official dictionary already cope with multilingualism. Indeed, we were able to determine the CPE-ID of "Hidemaru", which is a popular Japanese shareware

```
<?xml version="1.0" encoding="UTF-8"
 standalone="yes"?>
<assetInfo version="1">
<!-- SNIP -->
 <installedSoftwareInfo version="1">
  <softwareInfo>
   <name>Microsoft .NET Framework 4.5.1</name>
   <version>4.5.50938</version>
   <publisher>Microsoft Corporation</publisher>
   <size>0x2ce058</size>
   <installationDate>20141113</installationDate>
   <cpe id="cpe:/a:microsoft:.net_framework:4.5.1"
    matchingRate="6.0152926"/>
  </softwareInfo>
  <softwareInfo>
   <name>Oracle VM VirtualBox 4.3.6</name>
   <version>4.3.6</version>
   <publisher>Oracle Corporation</publisher>
   <size>0x24f7c</size>
   <installationDate>20131220</installationDate>
   <cpe id="cpe:/a:oracle:vm_virtualbox:4.3.6"
    matchingRate="7.654244"/>
   <cve id="CVE-2014-6540"/>
   <cve id="CVE-2014-4261"/>
   <cve id="CVE-2014-4228"/>
   <cve id="CVE-2014-2489"/>
   <cve id="CVE-2014-2488"/>
   <cve id="CVE-2014-2487"/>
   <cve id="CVE-2014-2486"/>
   <cve id="CVE-2014-2477"/>
   <cve id="CVE-2014-0983"/>
   <cve id="CVE-2014-0981"/>
  </softwareInfo>
<!-- SNIP -->
 </installedSoftwareInfo>
 <networkInfo version="1">
  <hostName>SimulationTerminal20150521</hostName>
  <openPorts>
   <port>135</port>
<!-- SNIP -->
  </openPorts>
  <nicInfoList>
   <gateWay>133.243.119.241</gateWay>
   <ipAddress>133.243.119.37</ipAddress>
   <macAddress>84:2B:2B:BA:AB:02</macAddress>
   <subnetMask>255.255.252.0</subnetMask>
   <nicName>Intel(R) 82578DM Gigabit Network
    Connection</nicName>
  </nicInfoList>
<!-- SNIP -->
 </networkInfo>
</assetInfo>
```

Fig. 4. Collected IT asset information for an terminal

product, using the collected IT asset information written in Japanese. The problem we found was the accuracy of the determined CPE-ID. Our current system calculates a value, called matching rate, which is used as a threshold value for determining an apprpriate CPE-ID; however, this does not always produce an accurate CPE-ID. One major reason for this is that even the latest version of the CPE dictionary contains outdated information. Indeed, NVD uses CPE-IDs that are not listed in the official dictionary.

We will revise the algorithm for determining CPE-ID. In principle, if the vendor name, software name, and version number are known, one can theoretically deduce the CPE-ID. Thus, one may argue that determining the CPE-ID based on a publicly available CPE dictionary is unnecessary. However, some CPE-IDs registered in the dictionary may not follow the nomenclature rules of CPE specification. For instance, Hidemaru's CPE-ID violated the rules at the time of this writing. Conventionally, it is believed that CPE-IDs listed in the CPE dictionary are used in most cases. Hence we believe that the CPE dictionary should not be ignored when matching the CPE-ID. Therefore, for future research, we consider determining a CPE-ID by locating information in the CPE dictionary and fine-tuning the derived CPE-ID.

### C. *Consideration on Locating Vulnerability Information*

Vulnerability repositories and their description techniques are always evolving; indeed, CPE-ID binding schemes are evolving. Our current prototype supports only URI bindings, but formatted string bindings are becoming popular. Although backward compatibility is usually maintained, it is beneficial to utilize the latest evolution of description techniques used in the repositories. Likewise, the prototype only work with NVD and JVN, but it should work with assorted online repositories via our cybersecurity knowledge base that links any type of XML information regardless of schemata gaps [5].

Furthermore, while retrieving vulnerability information, it is desirable to consider the severity of the vulnerability. The current prototype identifies the existence of vulnerabilities, but it cannot determine their severity. The severity may be determined based on the Common Vulnerability Scoring System (CVSS) scores [6] available in NVD, but these scores are not designed for this purpose. It is necessary to consider this aspect in implementing an efficient triaging system.

## V. Conclusion

Our proposed technique monitors the vulnerabilities of IT assets on networks using open information and standardized tools. It is unique because it automatically determines standardized IT asset identifiers from the collected IT asset information, and uses these identifiers as keys to query vulnerability information. Nevertheless, the proposed technique should be improved further. In particular, the identifier determination algorithm should be improved.

## References

[1] ISO/IEC. *Information technology – Security techniques – Information security management systems – Requirements*. International Organization for Standardization/International Electrotechnical Commission, Geneva, Switzerland, 2013. ISO/IEC 13335-1:1996.

[2] B. A. Cheikes, D. Waltermire, and K. Scarfone. *Common Platform Enumeration: Naming Specification Version 2.3*. National Institute of Standards and Technology, Maryland, USA, 2011. NIST Interagency Report 7695.

[3] ITU-T. *Common vulnerabilities and exposures*. International Telecommunications Union, Geneva, Switzerland, 2014. ITU-T Recommendation X.1520.

[4] JPCERT/CC and IPA. Japan Vulnerability Notes. http://jvn.jp/, 2014.

[5] T. Takahashi, and Y. Kadobayashi. Mechanism for linking and discovering structured cybersecurity information over networks. In *IEEE International Conference on Semantic Computing*. 2014.

[6] Forum of Incident Response and Security Teams. *Common Vulnerability Scoring System v3.0: Specification Document*, 2007. https://www.first.org/cvss/cvss-v30-specification-v1.7.pdf.