Software Vulnerability Management Techniques

Seminar: Selected Topics in IT Security, Winter Term 2016/17

Ehab Qadah

Abstract. On a daily basis, new security flaws are discovered in software applications. This makes the software vulnerabilities analysis one of the top concerns for organizations. The automatic identification of vulnerable software inside the organization is fundamental to avoid cyber-attacks. In this paper, we discuss two techniques to automatically monitor software vulnerabilities using open standards and public vulnerability information repositories, and alternative method to identify a vulnerable software using information obtained from social media platforms.

1 Introduction

Every year, new vulnerabilities are found in software products such as Internet Explorer, Flash Player, and Java. According to Symantec [1], in 2015, the number of new vulnerabilities reached 5,585, and 54 zero-day vulnerabilities were discovered. For cybercriminals unpatched vulnerabilities represent an opportunity to carry out their attacks successfully. Therefore, in order to prevent, for example, information theft, organizations require of effective and efficient methods to automatically detect vulnerability in a software that is installed in their networks. After detecting a software vulnerability, immediate actions (e.g., installing patches, or preventing users from employing the vulnerable software) must be taken.

To automatically detect software vulnerabilities, organizations need a reliable source of known vulnerabilities. Some solutions employ public repositories like the National Vulnerability Database (NVD)[1]. Then, organizations gather information on the software list that installed in their networks to match that information with the vulnerabilities.

In this paper, we beginning with a brief explanation of the software vulnerability management concept. Then, we describe standards used to unique identify software products and vulnerabilities. Thereafter, we present two approaches for automated software vulnerability management. Next, we discuss key points of these approaches. Finally, we give the conclusion.

[1] `https://nvd.nist.gov/`

2 Software Vulnerability Management

In this section, we shortly describe the concept of Software Vulnerability Management. Before doing this we define what a software vulnerability is. In the field of information technology, a vulnerability is a defect or weakness in a system that leads to a security incident or unauthorized access to information or services by attackers [2].

The Software Vulnerability Management concept can be defined as the continuous process of identifying vulnerabilities in a software product that is installed inside an organization. The process first requires managing the inventory of the organization's IT assets; gathering the IT assets information and store/update it. Then, search for the related vulnerabilities. In case of the discovery of a vulnerability, certain actions should be performed such as alerting the system's administrator or installing the corresponding patches to avoid possible threats based on those vulnerabilities. Figure 1 shows the basic process flow of the Software Vulnerability Management System.

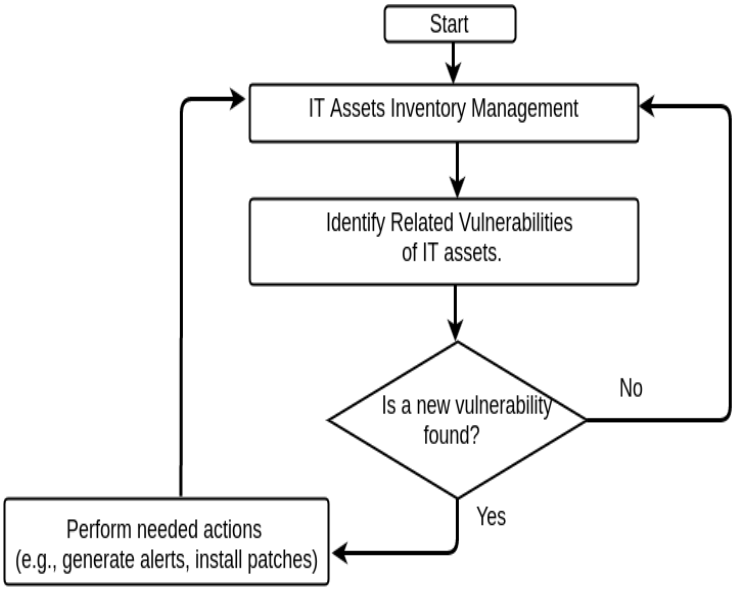

Fig. 1. Process flow of the Software Vulnerability Management System.

3 Standards Used in Software Vulnerability Management

In this section, we describe three components which are part of the Security Content Automation Protocol (SCAP)[2]. This protocol is a collection of open

[2] `https://scap.nist.gov/`

standards and enumerations for the security software flaws and configuration issues, the exchange of these standards offers the ability to automate vulnerability management [3]. SCAP is provided and maintained by the National Institute of Standards and Technology (NIST). The SCAP content is publicly accessible via NVD, which is also managed by NIST.

3.1 Common Vulnerabilities and Exposures (CVE)

CVE[3] is a list of known security software vulnerabilities (available in XML format). For each vulnerability, a unique standard identifier (e.g., CVE-2016-7892) is assigned. The CVE standard identifiers allow the data exchange between security solutions and vulnerability repositories (e.g., NVD, Japan Vulnerability Notes (JVN)[4]). The CVE list is managed by The MITRE Corporation.

The CVE list is not a vulnerability repository, but a list of identifiers (i.e., CVE names) for known vulnerabilities with basic information such as standard identifier and description. The CVE common identifiers of security vulnerabilities allow the linking between different vulnerability repositories (e.g., NVD, JVN, and Symantec Security Response Web site [5]). The linking between these databases allows to aggregate information about a vulnerability form all of them.

NVD provides an XML vulnerability feed that is built based on the CVE list. According to the XML feed schema [6], each vulnerability entry contains CVE id, summary, vulnerable software list, etc. Figure 2 provides an example of venerability entry (CVE-2014-3299) with subset of information obtained from the NVD XML feed.

```
<entry id="CVE-2014-3299">
<vuln:vulnerable-software-list><vuln:product>cpe:/o:
cisco:ios:</vuln:product></vuln:vulnerable-software-list>
<vuln:cve-id>CVE-2014-3299</vuln:cve-id>
<vuln:references xml:lang="en" reference_type=
"VENDOR_ADVISORY"><vuln:source>BID</vuln:source>
<vuln:reference href="www.securityfocus.com/bid/68177"
xml:lang="en">68177</vuln:reference></vuln:references>
<vuln:summary>Cisco IOS allows remote authenticated users
tocause a denial of service (device reload) via malformed
 IPsec packets, aka Bug ID CSCui79745.
 </vuln:summary>
</entry>
```

Fig. 2. Basic content of CVE-2014-3299 entry (other attributes are omitted).

[3] https://cve.mitre.org/index.html
[4] https://jvn.jp/en/
[5] https://www.symantec.com/security_response/landing/vulnerabilities.jsp
[6] https://scap.nist.gov/schema/nvd/vulnerability_0.4.xsd

3.2 Common Platform Enumeration (CPE)

CPE[7] is the identifiers dictionary for the software products and applications, operating systems and hardware devices. The official CPE dictionary (current version is 2.3) [8] is provided in XML format by NVD and it is updated frequently.

The CPE naming specifications document [4] defines two binding representations of the CPE-IDs. The first one, is formating the CPE-IDs based on the generic syntax of Uniform Resource Identifiers (URI), it follows the structure of **cpe:/{part}:{vendor}:{product}: {version}: {update}:{edition}:{lang}** (e.g., cpe:/a:autodesk:autocad_lt:2002:sp1::de). The possible values of the **{part}** component are h as hardware, o as operating system, and a as application. While the other attributes based on the product specification like the **{vendor}** component is filled with the name of the product's vendor. The unspecified attribute represents any value of the component as in the above example [edition=ANY]. The second binding representation, is the formatted string binding which is similar to the URI binding, but it has additional attributes, namely "target_sw/hw" to represent target software and hardware, "sw_edition" as software edition, "other" for extra attributes. It follows the syntax of **cpe:2.3: "part":"vendor": "product":"version": "update":"edition":"lang":"sw_edition":"target_sw":" target_hw":"other"** (e.g., cpe:2.3:a:autodesk: autocad_lt:2002:sp1:*: de:*:*:*:*). The formatted string binding is just a simple formatted string starts with the "cpe:2.3" prefix and uses the asterisks * to represent the any value of attribute.

Figure 3 provides an example of CPE entry that contains the URI and formatted string bindings representations, and the title and related references of the referenced product:

```
<cpe-item name =
"cpe:/o:canonical:ubuntu_linux:10.04::~~lts~~">
<title xml:lang="en-US">
Canonical Ubuntu Linux 10.04 LTS
</title>
<references><reference
href="http://www.canonical.com/">Vendor
</reference>
</references>
<cpe-23:cpe23-item name =
"cpe:2.3:o:canonical:ubuntu_linux:10.04:*:*:*:lts:*:*:*"/>
</cpe-item>
```

Fig. 3. An example of CPE Dictionary entry - "Ubuntu Linux 10.04 LTS" operating system.

[7] https://cpe.mitre.org/cpe/
[8] The Official CPE Dictionary available at https://nvd.nist.gov/cpe.cfm

3.3 Common Vulnerability Scoring System (CVSS)

CVSS[9] is a scoring system of the software vulnerabilities, which provides the characteristics and relative severity of the vulnerability. CVSS is managed by the Forum of Incident Response and Security Teams (FIRST).

CVSS entry consists set of different metrics. For example, access complexity, integrity impact, and access vector. The overall CVSS score value in the range from 0 to 10 is calculated using a formula that depends on the metrics values. Figure 4 shows an example of CVSS (version 2) score and base metrics of "CVE-2014-3299" (presented in Section 3.1) vulnerability .

```
<vuln:cvss>
<cvss:base_metrics>
<cvss:score>6.8</cvss:score>
<cvss:access-vector>NETWORK</cvss:access-vector>
<cvss:access-complexity>LOW</cvss:access-complexity>
<cvss:authentication>SINGLE_INSTANCE</cvss:authentication>
<cvss:confidentiality-impact>NONE</cvss:confidentiality-impact>
<cvss:integrity-impact>NONE</cvss:integrity-impact>
<cvss:availability-impact>COMPLETE</cvss:availability-impact>
<cvss:source>http://nvd.nist.gov</cvss:source>
<cvss:generated-on-datetime>2017-01-11T14:37:58.247-05:00
</cvss:generated-on-datetime>
</cvss:base_metrics>
</vuln:cvss>
```

Fig. 4. CVSS score and metrics of the "CVE-2014-3299" vulnerability provided by NVD XML feed.

4 Software Vulnerability Management Approaches

In this section, we present two approaches to implement software vulnerability management systems based on open standards (e.g., CVE) and open repositories (e.g., NVD), or based on security information extracted from social media.

4.1 Software Vulnerability Management Using Open Standards

This section presents a system proposed by Takahashi et al. in [5] to automatically monitor vulnerabilities of computing assets inside the IT infrastructure of an organization. Their main contribution is to automate the process of vulnerability management using open standards and repositories. The idea behind using

[9] `https://www.first.org/cvss/specification-document`

open standards and information sources is to develop a system that can be used by a wide range of organizations.

First, the system collects information about the list of IT assets inside an organization. Afterward, the system stores the collected information and uses it to determine a CPE-ID for each IT asset. Then, the system utilizes these identifiers to check the existence of related vulnerabilities, by querying the system's vulnerability knowledge base using the computed identifiers as keys. Finally, an alert about the identified vulnerable software will be sent to the system administrator.

4.1.1 System Components

The system is composed of 4 elements, which are described in the following:

1. **Terminals**: this element includes all electronic devices used by the organization's employees to perform their activities. In most cases, an agent is installed on a terminal to collect information about the installed IT assets. The collected information is then sent to the asset management server.
2. **Asset management server**: this component is responsible to communicate with the installed agents on the terminals to collect information of the IT assets, and also it monitors and analyzes the organization's network traffic in order to collect information about the terminal's IT assets that does not have an installed agent. The collected information is then organized and stored using the system's schema. Furthermore, the asset management server determines the IT asset standard identifier (i.e., CPE-ID) using the collected information and appends this identifier to the IT asset information. This element also uses these identifiers to check the presence of a vulnerability by querying the vulnerability knowledge base.
3. **Vulnerability knowledge base**: is the local system's database for the vulnerability information contains data from the following repositories: NVD, JVN.
4. **Administrator terminal**: is the console used by the system administrators which receives the notification alert about the discovered security vulnerabilities from the asset management server.

4.1.2 System Workflow

In this section the workflow of the proposed system in [5] is described.

1. **Compile IT assets information**: the system starts with the process of collecting the information of the organization's IT assets. This is achieved by sending requests to the agents installed on the terminals, or by monitoring the network. The agents gather the information of the installed software, and then, an XML document is generated. This document contains, for instance, the software name, version, publisher, and installation date, as shown in Figure 5. [10]

[10] Based on figure 4 in [5] with omitting some details

```
<assetInfo version="1">
 <installedSoftwareInfo version="1">
 <softwareInfo>
 <name>Adobe Flash Player 23.0.0.205</name>
 <version>23.0.0.205</version>
 <publisher> Adobe Systems Software </publisher>
 <size>0x24e23</size>
 <installationDate>20161122</installationDate>
 </softwareInfo>
 </installedSoftwareInfo>
</assetInfo>
```

Fig. 5. An example of collected information by the proposed system for an installed software on some terminal.

2. **Resolving of IT asset identifier**: the system determines the CPE-IDs for the IT assets using the information collected in the first step to uniquely identify them. The Official CPE dictionary is obtained from NVD repository and used as reference CPE-IDs for the proposed system. The basic algorithm builds a query from the collected software information like name, version and owner, then a matching rate (percentage of the similar characters between the query and CPE-ID) for all CPE-IDs in the CPE dictionary is calculated. The CPE-ID with the highest matching rate is selected and added to the XML document of the asset. Figure 6 shows a CPE-ID which was assigned to an asset. In this example the matching rate of CPE-ID is 7.65.

```
<cpe id="cpe:/a:adobe:flash_player:23.0.0.205"
  matchingRate="7.65"/>
```

Fig. 6. Example of a calculated CPE-ID with its corresponding matching rate.

3. **Finding vulnerability information**: the system uses the determined CPE-IDs to query the vulnerability knowledge base for related vulnerabilities and in case of discovering a vulnerability the system administrator is notified by an alert containing the vulnerability details.

4.2 Software Vulnerability Management Using Social Networks Information

This section describes an alternative vulnerability data source to the traditional vulnerability repositories such as NVD for the software vulnerability monitoring task. Due to the delay to publish a security flaw by the typical vulnerability information sources, detection of zero-day vulnerabilities is difficult. In order

to solve this issue, Trabelsi, Slim, et al. in [6] propose an approach based on security information collected from social media (Twitter) to detect the new software vulnerabilities.

The proposed system takes the advantage of getting informed about vulnerable software earlier than the normal software vulnerabilities repositories. That is because software developers use the social media platforms and technical blogs to discuss the security software flaws and issues. On the contrary, the classical information sources (e.g., NVD) wait for the availability of patches from the software vendor before publishing the vulnerability information.

The introduced system is called SMASH (Social Media Analysis for security on HANA), consists two subsystems data collection and processing. The data collection subsystem is responsible to gather the security information from the social media (Twitter [11]), by searching the Twitter's stream content for related security information (by using security associated terms e.g., exploit) and store it in the local database to be utilized by the system later. The system also keeps a copy of the software vulnerabilities form NVD (XML vulnerability feed) to distinguish the new vulnerabilities from already published ones.

The data processing subsystem is performing the extraction of security information by analyzing the stored Twitter content using various data mining techniques (e.g., K-mean algorithm). The extracted security information includes zero-day vulnerabilities which are not published yet, and requests to create new CVEs or update old entries.

The proposed system offers the functionality of monitoring certain software components selected by the system's users, then the discovered vulnerabilities by the system are displayed to the users.

5 Discussion

The system proposed in Section 4.1 mainly focuses on the automation of software vulnerabilities management using open standards and public vulnerability repositories. The proposed system does not fully automate the process of software vulnerability monitoring, since in case of a new vulnerability is found, the system just sends an alert for the system's administrator who is responsible to perform the needed action manually. To deal with this drawback, the system can execute fast and immediate actions such as blocking the corresponding terminal from accessing to the company's network, to prevent the attackers from exploiting the vulnerable software on that terminal. Furthermore, the accuracy of the system's outcome mainly depends on the precision of the determination of the IT assets identifiers (i.e, CPE-IDs). The proposed system finds the CPE-ID for an IT asset based on a numerical method of calculating a matching rate. According to the authors, this approach does not always deliver accurate results. In addition, the determined CPE-IDs must be validated by the system before

[11] Twitter has been used in the proposed system prototype, but the technique is also applicable for other social media platforms.

moving to the further steps to avoid unnecessary processing or the generation of invalid alerts.

The system outlined in Section 4.2 proposes a method to find vulnerable software based on information obtained from Twitter. The system analyzes the social media content to extract the security vulnerability information. This approach allows detecting zero-day vulnerabilities which is not feasible using classical channels such as NVD. On the other hand, this approach may utilize non-validated or wrong information to identify security software vulnerabilities, this imposes the proposed system to contain a trust module of the extracted information from the social media data. Consequently, the accuracy of the system's output is fundamentally affected by the strength of the trust model.

6 Conclusion

In this paper, we explore SCAP standards that related to software venerability management. And we present two approaches to implement software vulnerability management systems. First approach based on open standards (e.g., CPE, CVE) and open repositories (e.g., NVD). And the second one based on security information obtained from social media (Twitter).

The importance of the automatic detection and monitoring of vulnerable softwares inside organizations to avoid cyber-attacks introduces the need for software vulnerability management systems. The two approaches described in Section 4 allow the enterprises to get informed about software vulnerabilities of their IT assets. Nonetheless, they do not replace the normal security solutions and policies (e.g., firewalls) but they just support their goal to prevent attacks on the organization's IT infrastructure. The vulnerability repositories (e.g., NVD) and the open standards of the SCAP protocol (e.g., CPE, CVE) play the main role in developing software vulnerability management systems.

References

1. Symantec. Internet Security Threat Report, Vol. 21 https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf, 2016.
2. G Stoneburner, A Goguen, and A Feringa, Risk Management Guide for Information Technology Systems, NIST Special Publication 800- 30, July 2002 http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf
3. The Technical Specification for the Security Content Automation Protocol (SCAP) NIST Special Publication 800-126 Revision 3.
4. Common Platform Enumeration: Naming Specification Version 2.3 Brant A. Cheikes, David Waltermire, Karen Scarfone, Brant A. Cheikes, David Waltermire, Karen Scarfone, Rebecca M. Blank, Acting Secretary 2011 NIST Interagency Report 7695, NIST-IR-7695
5. Takahashi, Takeshi, Daisuke Miyamoto, and Koji Nakao. "Toward automated vulnerability monitoring using open information and standardized tools." 2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops). IEEE, 2016.

6. Trabelsi, Slim, et al. "Mining social networks for software vulnerabilities monitoring." 2015 7th International Conference on New Technologies, Mobility and Security (NTMS). IEEE, 2015.