The 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications
22-25 September, 2021, Cracow, Poland

CPE and CVE based Technique for Software Security Risk Assessment

Roman Ushakov [1], Elena Doynikova [2], Evgenia Novikova [2], Igor Kotenko [2]
[1] Saint Petersburg Electrotechnical University "LETI", 5 Prof. Popov str, St. Petersburg, Russia, romik75263@gmail.com
[2] St. Petersburg Federal Research Center of the Russian Academy of Sciences (SPC RAS), 14ya Liniya V.O., St. Petersburg, Russia, {doynikova, novikova, ivkote}@comsec.spb.ru

***Abstract*—Currently, a lot of work has been done in the area of detection, scoring, and inventory of software and hardware vulnerabilities. Known vulnerabilities are listed in the open databases. It is essential to continuously monitor that information system doesn't contain severe vulnerabilities to ensure its information security. Applicability of open vulnerability databases is limited by the challenges occurring due to automated mapping the software product names in the analyzed system logs to their product names in the open sources (to extract relevant vulnerabilities from them). The paper proposes the technique incorporating an algorithm for mapping the software products names in the analyzed system logs to the relevant Common Platform Enumeration entries in open vulnerability databases based on the Ratcliff/Obershelp algorithm, identification of known vulnerabilities for the detected entries, and security risk assessment of the analysed system. The technique is implemented and tested using Windows computers software and has shown an accuracy of 79% on average.**

Keywords—cybersecurity; software; products; vulnerabilities; databases; CPE; CVE; risk assessment; mapping

I. INTRODUCTION

Currently in the information security area the investigation and detection of complex cyber attacks is highly relevant and researched [1, 2]. Though detection of cyber attacks exploiting earlier unknown vulnerabilities is essential, monitoring of the known vulnerabilities in the information system is still important. A lot of work has been done in the area of detection, scoring, and inventory of software and hardware vulnerabilities. Known vulnerabilities are listed in the open databases. Thus, one of the most comprehensive databases of known vulnerabilities is National Vulnerability Database (NVD)[1]. NVD stores vulnerabilities of the software and hardware represented in the Common Platform Enumeration (CPE)[2] format. The vulnerabilities are represented in the Common Vulnerability Enumeration (CVE)[3] format and are scored using Common Vulnerability Scoring System (CVSS) using the scale "None, Low, Medium, High, Critical" [3].

To ensure its information security it is essential to continuously monitor that the information system doesn't contain severe vulnerabilities. New vulnerabilities can be detected and added to the NVD. Besides, the configuration of the analysed information system can be modified resulting in new vulnerabilities. A manual search of the vulnerabilities in the vulnerability database (NVD) is resource intensive. Thus, the process of determining the information system's software products and vulnerabilities should be automated. The key challenge consists in the lack of unification of software naming when collecting it in the information system and searching possible errors in the names of the software in the information system and the NVD database.

Thus, in this research the following task was set: to develop a technique for the automated search of known vulnerabilities of information systems using CPE and CVE and its implementation in the tool for security risk assessment.

To implement the task the research problem of mapping software names in the information system into the CPE entries was considered. To solve this problem we proposed an algorithm based on Ratcliff/Obershelp algorithm.

The main contribution of this research is as follows:

- the technique for mapping software names in the information system into the CPE entries based on the Ratcliff/Obershelp algorithm;
- the proposed technique for vulnerability search;
- the developed security risk assessment tool.

The novelty of this research consists in the algorithm for mapping software names in the information system to the CPE entries based on the Ratcliff/Obershelp algorithm recursively implemented for the software product name's parts. The proposed an algorithm allows automation of searching the known vulnerabilities.

The proposed technique can be used to extend the functionality of vulnerability scanners, to automate

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

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

[3] https://cve.mitre.org/

known vulnerabilities search to construct cyber attack sequences against information systems for security risk assessment.

The developed tool can be enhanced in further research via interaction with the operator for the results verification and system learning on this base. Besides, an analysis of the impact of incorrect CPE mapping on the resulting security risk score should be conducted.

The paper is organized as follows. Section II analyses the related works. Section III describes the research methodology, the proposed technique, the developed security assessment tool, and the experimental results. The paper ends with a conclusion and a future research description.

II. Related Works

A. Used Sources and Formats

This research uses NVD as an open database of known vulnerabilities. NVD stores vulnerabilities of the software and hardware represented in the CPE format.

The CPE standard was developed to unify product naming. The CPE entry has the following format [4]:

CPE = cpe:/ {part} : {vendor} : {product} : {version} : {update} : {edition} : {language}.

For example, CPE entry for the Microsoft Internet Explorer 8.0.6001 Beta is as follows:

cpe: 2.3: a: microsoft: internet_explorer: 8.0.6001: beta: *: *: *: *: *: *: *,

where '*' denotes that an appropriate field can take any value.

The vulnerabilities in NVD are represented in the CVE format. The CVE standard was developed to unify vulnerability description. CVE entry should contain CVE ID, name of an affected product, the affected or fixed versions, vulnerability type/root cause and/or impact, public reference, description, and information if the related product is no more supported [5].

Finally, the vulnerabilities in NVD are scored using CVSS [3].

B. Research Works

The challenge of CPE mapping was considered in multiple research works. Thus in [6], a regular expression for name and version is applied.

In [7-9] the Levenshtein algorithm is used. In [7, 8] it is applied to vendor and name fields of the software information in the inventory database and CPE entries in the CPE dictionary to select several CPE candidates using Levenshtein distance less than 2. Then the selected CPE candidates are prioritized using versions. An administrator should manually select the correct CPE entry from the CPE candidates. This entry is then used to search CVE entries for the analysed software product.

However, the authors provide testing results just for 12 products. In [9] the software products are compared with the CPE list. Three lists are consequently generated: (1) the list containing candidates from the CPE list with Levenshtein distance = 0; (2) the list containing candidates with Levenshtein distance < 2 and (3) the list containing candidates with Levenshtein distance = 2. Then the length of the lists is consequently checked. If it equals 0 then the next list is analysed; if 1 – the appropriate candidate is selected as matching CPE entry; if > 1 – the matching is unsuccessful. The selected CPE entry is then used to search CVE entries for the analysed software product.

In [10] a hierarchical tree is used generated based on the CPE entry fields. The field values are consequently compared. This approach doesn't consider errors or a lack of unification. Besides, validation of the approach is not provided. In [11] multiple hierarchical trees are used.

The results of the related work analysis are provided in Table I. Though the CPE mapping challenge is considered in multiple papers the proposed methods are still error-prone.

TABLE I. Results of Related Work Analysis

Research work	Method	CPE fields	Disadvantages
Gawron et al., 2017 [6]	Regular expression.	Name and version	Not unified entries or entries with errors are not mapped
Sanguino and Uetz [7, 8]	Levenshtein algorithm to select CPE candidates based on vendor and name, distance should be less than 2. Versions are used to prioritize the candidates.	Vendor, name, and version	Requires manual selection from several CPE candidates to avoid errors. Evaluation on 12 products.
O'Hare and Macfarlane, 2018 [9]	Levenshtein algorithm applied to the CPE list. Candidates with the minimum distance are selected (consequently, 0, 1, or 2).	Full entry	Error-prone.
Na et al., 2017 [10]	Hierarchical tree based on entry fields, full words comparison	Full entry	Error-prone, no evaluation.
Bela and Enachescu, 2015 [11]	Multiple hierarchical trees	Full entry	Error-prone

In this research we analysed different string comparison algorithms and selected the Ratcliff/Obershelp algorithm to get better accuracy while mapping software products to CPE, besides we used word-by-word comparison. We further used the proposed mapping in the scope of the technique for vulnerability searching and security assessment.

III. TECHNIQUE FOR SOFTWARE SECURITY RISK ASSESSMENT

A. Research Methodology

We performed the following steps to develop the CVE and CPE based technique for software security risk assessment.

1. Gather names of applications from several Windows workstations. For this goal we used the winapps Python library.
2. Select fields for comparison. We used the same CPE fields as in the related works: vendor, name, and version.
3. Compare vendors, names and versions in the obtained on step 1 applications list with entries in the CPE dictionary using direct comparison automatically by a Python script. Manual verification of the results shown that accuracy is low. Thus, we decided to test other string matching algorithms.
4. Compare string matching algorithms, including the Levenshtein distance and the Ratcliff/Obershelp algorithm. While these algorithms shown similar values for the correct CPE entries, the Ratcliff/Obershelp algorithm shown greater distance with incorrect entries. The comparison results for the Ratcliff/Obershelp algorithm for 7 software products are provided in Fig. 1. The y-axis of the bar chart in Fig. 1 represents the Ratcliff/Obershelp similarity metric (1 means the complete match of the two strings, while 0 means there is no match). The x-axis of the bar chart in Fig. 1 represents the software product names gathered by the winapps Python library. The software product named on the x-axis is compared to the CPE product names listed in the legend and represented by the bars. Thus "7-Zip 19.00 (x64)" software product name is most similar to the "7-Zip 19.00" CPE software product name and less similar to the "Skype" CPE software product name.
5. Develop the novel algorithm for comparison based on the Ratcliff/Obershelp algorithm and name field partitioning.
6. Develop the Python tool for software products and CPE matching and test the results.
7. Develop the technique for security risk assessment.
8. Develop the Python tool for the information system's security assessment.

B. Technique for Software Security Risk Assessment

The developed technique for software security risk assessment is represented in Fig. 2 and includes the following stages:

1. Gather information on the software products from the analyzed system, remove duplicates. At this stage access to the information system can be provided or system logs.
2. Define CPE entries for the gathered software products.
3. Use defined CPE entries to find known vulnerabilities in the NVD database.
4. Define CVSS scores for the software product vulnerabilities.
5. Define security risk for the software product as the maximum CVSS value for its detected vulnerabilities.
6. Define security risk for the host as the maximum risk of software products.
7. Define security risk for the information system as the maximum risk of its hosts.

The proposed technique allows automatically monitor if known vulnerabilities appeared in the system under analysis and lead to unacceptable security risks.

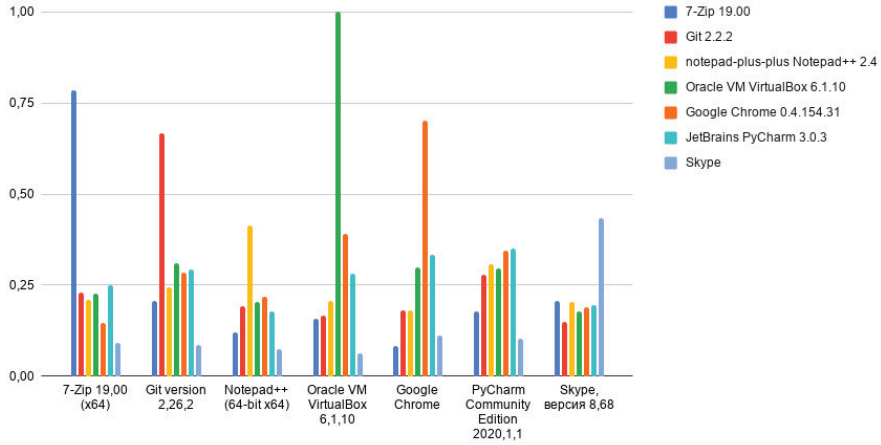

Figure 1. The comparison results for the Ratcliff/Obershelp algorithm for 7 software products

The core of the proposed technique is the CPE mapping algorithm used in the second stage. It supposes construction of the CPE entry based on the list returned by the winapps Python library in current implementation but any other source of information on system software can be used as an input. The algorithm is as follows.

1. Compare constructed CPE entry for the software product and entry in the CPE list using Ratcliff/Obershelp algorithm.
2. If the similarity is more than 0.95, return the CPE entry. The threshold was set experimentally.
3. Else outline three strings from the constructed CPE entry: string for the vendor, string for the name, and string for the version.
4. Compare names from the constructed CPE entry and entry from the CPE list using the Ratcliff/Obershelp algorithm. If the similarity is more than 0.95, go to step 6.
5. Else divide name strings by words and compare separate words. If the similarity of separate words exceeds the given experimentally set thresholds, go to step 6, otherwise mapping is not successful.
6. Compare vendors using the Ratcliff/Obershelp algorithm. If the similarity is more than 0.95, go to step 7.
7. Compare versions using regular expression. If versions are similar, return the CPE entry. Otherwise, mapping is not successful.

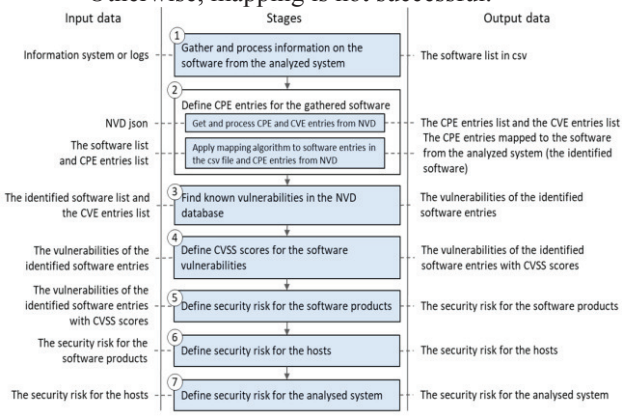

Figure 2. The technique for software security risk assessment

C. Tool for Software Security Risk Assessment

The developed technique and CPE mapping algorithm are implemented in the scope of the tool developed in Python. It incorporates the database with software products, their CPE entries, and the related CVEs and security risk scores; the component that implements CPE matching, and the component that implements the security risks assessment technique.

The tool was used to analyse the security of the tested infrastructure. The first experiment was conducted using one Windows host with 26 applications. 16 from them present in the CPE dictionary. 12 from them were successfully matched and 1186 vulnerabilities were detected correctly. The second experiment was conducted using the test infrastructure incorporating 6 Windows hosts with 160 applications: 68 from them present in the CPE dictionary; 54 - were successfully matched, moreover 8384 vulnerabilities were detected and 8370 from them were detected correctly. Thus, the system successfully matched 79% of applications on average, at the same time performance has significantly increased.

IV. Conclusion and Future Work

The paper proposed: the technique incorporating algorithm for mapping software products' names to the relevant Common Platform Enumeration entries in known databases based on the Ratcliff/Obershelp algorithm; identification of vulnerabilities for the detected entries; and tool for risk assessment implementing the proposed technique and algorithm. The experiments were conducted using the test environment and shown 79% accuracy on average. Future work will include extended analysis of the results accuracy, analysis of the impact of incorrect mapping on the security risk score, and enhancement of the analysis via machine learning based on the incorrect mappings.

Acknowledgment

This research is being supported by the grant of RSF #21-71-20078 in SPC RAS.

References

[1] V. Kharchenko, Y. Ponochovnyi, A.-S. M. Q. Abdulmunem, A. Boyarchuk, "Security and availability models for smart building automation systems," *International Journal of Computing*, 16(4), 2017, pp. 194–202.

[2] J. Dalou', B. Al-Duwairi, M. Al-Jarrah, "Adaptive entropy-based detection and mitigation of ddos attacks in software defined networks," *International Journal of Computing*, 19(3), 2020, pp. 399–410.

[3] FIRST, "CVSS. Common Vulnerability Scoring System version 3.1. Specification Document. Revision 1," 24 p.

[4] B.A. Cheikes, D. Waltermire, K. Scarfone, "Common P;atform Enumeration: Naming Specification. Version 2.3," NIST Interagency Report 7695, 2011, 49 p.

[5] MITRE, "CVE Numbering Authority (CNA) Rules," version 3.0, 2020, 32 p.

[6] M. Gawron, F. Cheng and C. Meinel, "PVD: Passive vulnerability detection," *2017 8th International Conference on Information and Communication Systems (ICICS)*, 2017, pp. 322–327.

[7] L. Sanguino, R. Uetz, "Software Vulnerability Analysis Using CPE and CVE," 2017.

[8] Github website. Inventory Vulnerability Analysis (IVA), URL: https://github.com/fkie-cad/iva.

[9] J. O'Hare, R. Macfarlane, "Honours Project - Scout: A Contactless 'Active' Reconnaissance Known Vulnerability Assessment Tool," 2018.

[10] S. Na, T. Kim, H. Kim, "A Study on the Service Identification of Internet-Connected Devices Using Common Platform Enumeration," in: Lecture Notes in Electrical Engineering, vol 448. Springer, Singapore, 2017,

[11] G. Bela, C. Enachescu, "ShoVAT: Shodan-based vulnerability assessment tool for Internet-facing services," *Security and Communication Networks*, 9, 2015, 10.1002/sec.1262.