# Towards Automatically Matching Security Advisories to CPEs: String Similarity-based Vendor Matching

Kylie McClanahan
*University of Arkansas*
klmcclan@uark.edu

Qinghua Li
*University of Arkansas*
qinghual@uark.edu

***Abstract*—When a vulnerability is reported by the National Vulnerability Database (NVD), affected products are listed in the structured Common Platform Enumeration (CPE) format. Unfortunately, if the vulnerability is in a software library (e.g., Log4j), it will not include CPEs for each product containing that library. In these cases, security operators need to manually read the vendor's or third-party security advisories to see if their product is affected. However, these advisories do not report affected products in a structured format, which prevents automated processing.**

**This paper makes the first effort towards automatically constructing structured CPEs for the vulnerable products in a non-NVD security advisory from the unstructured data in the advisory. Since this is a very challenging problem, this paper specifically focuses on the initial but key step of matching the unstructured vendor names in security advisories to the structured vendor representations in the standard CPE format. We explore the feasibility of using string similarity to solve the problem. The basic idea is to compare a vendor name from the non-NVD advisory with each vendor in the official CPE dictionary. The CPE vendor with the highest similarity score to the advisory's vendor will be considered as the match. We first conduct an experimental, comparative study of multiple mainstream string similarity metrics for this matching problem. To improve the performance, we then design a new string similarity metric that is adapted from an existing metric by weighing different tokens in the advisory's vendor name differently.**

***Index Terms*—Vulnerability, CPE, Entity Matching**

## I. Introduction

Vulnerability and patch management is a difficult and multi-step process, lasting from the initial notification of a security issue to the verification of device functionality after a successful patch installation. One key step in the process is determining applicability: whether a device is affected by a certain vulnerability or not (or should apply a certain patch or not); indeed, this informs all other actions.

The National Vulnerability Database (NVD) uses the Common Platform Enumeration (CPE) to inform applicability. A vulnerability in the NVD, identified by a Common Vulnerabilities and Exposures (CVE) string, includes a listing of CPEs, or strings that represent vulnerable products and/or configurations. A security operator or security analyst could create a list of CPEs for their organization's devices and, in theory, automatically check that list against vulnerabilities released by the NVD.

One severe limitation with this approach is the case of a vulnerability in a commonly-used library or framework. The CVE for a framework will include a CPE for that framework by default. The MITRE CVE Program takes measures to prevent duplicate CVEs, and as a part of that, their program guidelines state that a product which contains a vulnerable component — when the vulnerability in the component has already been assigned a CVE identifier— is not enough for that product to be given its own CVE [1]. Instead, CPEs for products containing a vulnerable component can be added to the original CVE for that component. Thus, products that use or implement a vulnerable library may submit their CPEs to the NVD to be included in the CPE list for that library's CVE. However, in practice, the majority of vendors do not do this, so the list of vulnerable CPEs for that library is incomplete.

This can be seen most clearly in the Apache Log4j framework and the Log4Shell exploit seen in December of 2021. Log4Shell was defined by five CVEs. CVE-2021-44228 was the first released and remediated, while the following four detailed subsequent issues found in the fixed version. All five CVEs have the CPE for Log4j listed as the first vulnerable configuration. In addition to the CPE for Log4j itself, CVE-2021-44228 lists a number of vulnerable products from 10 unique vendors. Yet, a cursory web search for information on Log4j and Log4Shell yields nearly 100 vendors who have released security advisories, notices, or announcements that one or more of their products was vulnerable to the Log4Shell exploit. From this example, we can see that the CPEs in a CVE advisory of the NVD do not paint a complete picture of vulnerable products.

To address this issue, many security operators also monitor vulnerability information sources outside of the NVD, such as vendor security advisories or the Industrial Control Systems (ICS) Advisories published by the Cybersecurity & Infrastructure Security Agency (CISA) which cover many vendors. Security advisories often contain additional information about a vulnerable device, particularly that data which does not neatly fit into the standard CVE format, but they rarely, if ever, report affected products in a structured or standardized format, which means such data cannot been processed automatically.

We make the first effort towards automatically constructing structured CPEs for the vulnerable products in a non-NVD se-

979-8-3503-7099-7/24/$31.00 ©2024 IEEE

TABLE I
EXAMPLES OF VENDOR NAMES AND THEIR CPE DICTIONARY ENTRIES

| | Advisory Vendor Name | CPE Vendor Name |
|---|---|---|
| 1 | Automated Logic | automatedlogic |
| 2 | B&R Industrial Automation | br-automation |
| 3 | B+B SmartWorx | advantech |
| 4 | Environmental Systems Corporation | envirosys |
| 5 | Intellicom | intellicom |
| 6 | Propump and Controls, Inc | propumpservice |
| 7 | Schweitzer Engineering Laboratories | selinc |

curity advisory (e.g., those from CISA) from the unstructured data in the advisory, in order to supplement CPEs listed in the NVD. With these additional CPEs in the machine-readable format, security operators would obtain, in an automated way, a more complete picture of the vulnerabilities of their system. Since constructing CPEs from the unstructured advisory data is a very challenging problem, this paper specifically focuses on the initial but key step of matching the vendor names in security advisories (which are in unstructured formats) to vendor representations in the standard, structured CPE formats. Table I shows examples of vendor names in security advisories and their corresponding vendor representations in CPE standards.

This problem is not trivial due to the variety of vendor names in advisories and vendor representations in CPEs. In Table I, rows 2 and 4 require some shortening of the advisory vendor name, and the CPE for row 7 uses an acronym not present in the advisory vendor name (SEL). For rows 3 and 6, the CPE vendor name contains some additional piece of context, while rows 1 and 5 are a straightforward match. Another challenge is that corporate identifiers may change over time through mergers, acquisitions, or rebranding. When a change occurs, existing CPEs that reflect the old company name are, in most cases, not edited to retroactively reflect the change. CPEs published after a change occurs will typically reflect the new name, but this is not a requirement. Some vendors may choose to not change the vendor name even for newly published CPEs for the sake of consistency. Additionally, larger companies may have multiple disparate teams who report CPEs, where each team may handle security issues for a single product family. For example, Team A may submit CPEs with a vendor name of `vendor_inc`, while Team B might submit under the vendor name `vendorinc`.

In this paper, we explore the feasibility of using string similarity to solve the problem of matching vendors in security advisories to CPE vendors. The basic idea is to compare a vendor name from the CISA advisory with each vendor in the official CPE dictionary (as well as other valid CPEs in the NVD). The CPE vendor with the highest similarity score to the advisory's vendor will be considered as the match. We first conduct an experimental, comparative study of multiple mainstream string similarity metrics for this matching problem. To improve the matching performance, we also design a new string similarity metric that is adapted from an existing metric by weighing tokens in the advisory's vendor name differently.

```
cpe:2.3:part:vendor:product:version
```

Fig. 1. Commonly-populated attributes of a CPE identifier

The paper is structured as follows. Section II covers background and related work. Section III introduces the mainstream string similarity metrics used in this study, and also our new metric. Section IV details our dataset and presents the results of the comparative study. Section V presents future research directions, and Section VI concludes the paper.

## II. BACKGROUND

### A. Vulnerability Landscape

As mentioned in Section I, the National Vulnerability Database (NVD) is the central repository for vulnerability information and is maintained through the National Institute of Standards and Technology (NIST). A reported vulnerability is given a Common Vulnerabilities and Exposures (CVE) identifier. A CVE record contains a listing of vulnerable CPEs, along with other structured data that provides a severity score, references, weakness types, and others. CVEs are reported by CVE Numbering Authorities (CNAs); each CNA has a defined scope in which it can report vulnerabilities. If the CNA is a software manufacturer, for example, its scope might be limited to its own products.

A CPE is a collection of key-value pairs, called attributes, that together form a representation of a computer system, whether software, firmware, or hardware. Attributes are loosely ordered from least specific to most specific. The most commonly used attributes are VENDOR, PRODUCT, and VERSION, as shown in Figure 1. The prefix `cpe:2.3` indicates a CPE version 2.3 string. The `part` attribute can have a value of "a", for application, "h", for hardware, "o", for operating system, or "*" for any. All other fields have string values. For this work, we focus on the `vendor` attribute.

The NVD maintains an official CPE dictionary. Organizations can submit potential CPEs to the NVD ad hoc, or a CNA can include new CPEs not present in the official dictionary when submitting a new CVE to the NVD. Per the NVD website, any CPE in a CVE submission that is not present in the official dictionary will be reviewed and added [2]. In practice, however, there are many CPEs present in CVEs that have no corresponding entry in the official CPE dictionary. If these CPEs were only found in recently released CVEs, that might simply point to a "backlog" of work; however, they appear in CVEs that are now many years old.

For this work, we consider vendors from both CPEs present in the official CPE dictionary *and* those included in published CVEs to be valid options.

### B. Related Work

*1) CPE Matching:* There is not much work in this area. [3] proposes a method to construct CPEs for affected products from the description of CVEs in the NVD. However, their

method is customized for CVE summaries in the NVD, and might not work well for other third-party advisories that have different content and styles of describing vulnerabilities. More importantly, their method does not address the focus of this paper: the case where a CVE in a software library does not list CPEs for products using the vulnerable library or even mention the products. [4] creates a set of potential CPEs for a software product and allows a user to select the best option. However, their solution is not fully automated since users still need to manually make a selection. [5] chooses CPEs using the list of Installed Programs on a Windows machine and compares them against the CPE dictionary. However, this work can only handle structured software product information obtained from the Windows installer, and their approach cannot apply in our case where the vendor names in advisories are unstructured data. [6] proposes a system to recommend CPEs for an asset inventory and performs well. Like those above, though, the NVD is still assumed to be the ground truth of affected products for a vulnerability.

*2) Entity Matching:* Entity matching (also called fuzzy matching, deduplication, record linkage, etc.) is the process of deciding whether two descriptions or representations refer to the same entity. String similarity is a well-established method for entity matching [7, 8, 9]. [10] evaluates string similarity methods for toponym (place name) matching; [11] for historical spelling variants. There is some work to use machine learning techniques for entity matching, like [12], but in cases with low context and/or limited training data, like CPEs, string similarity will often still outperform.

*3) Vulnerability Management:* There is a growing body of work around automated vulnerability management. [13] uses Twitter (now X) data for open-source intelligence (OSINT). [14] identifies mitigation information for vulnerabilities. [15] processes vulnerability descriptions to determine which network service or protocol is being exploited, which informs actions to fix the vulnerability. [16, 17] use vulnerability features to predict remediation actions and scheduling. [18] studies the use of large language models in vulnerability management. However, we address a different problem from them.

## III. String Comparison Methods

In the following algorithms, $|s_i|$ is the length of a string $s_i$, and $s_{i,j}$ represents the character of the string $s_i$ at index $j$. $s_{i,j:}$ represents the substring of $s_i$ starting at index $j$ to the end of $s_i$.

Finally, similarity is the inversion of distance.

$$sim = (1 - dist) \tag{1}$$

### A. Normalized Levenshtein

The Levenshtein Distance between two strings is the minimum number of edits (addition, deletion, and substitution) required to turn a string $s_1$ into another string $s_2$ [19]. Equation 2 shows a naive recursive definition.

$$L(s_1, s_2) = \begin{cases} L(s_{1,1:}, s_{2,1:}) & s_{1,0} = s_{2,0} \\ 1 + min \begin{cases} L(s_{1,1:}, s_2) \\ L(s_1, s_{2,1:}) \\ L(s_{1,1:}, s_{2,1:}) \end{cases} & else \end{cases} \tag{2}$$

Because the other metrics used in this work are measures of similarity, a normalized Levenshtein distance was needed. To normalize, the Levenshtein distance calculated by Equation 2 is divided by the length of the longer of the two strings.

$$L_{norm} = \frac{L(s_1, s_2)}{max(|s_1|, |s_2|)} \tag{3}$$

At that point, the Levenshtein similarity can be obtained using Equation 1 and compared to other similarity metrics.

### B. Discounted Levenshtein

Discounted Levenshtein is a variant of Levenshtein where edits later in a string are penalized less than edits at the beginning; it was proposed in [20]. In the implementation of Discounted Levenshtein used in this paper, the decrease in cost is applied from the second character of the string. The value per-edit can be calculated according to the log function in Equation 4.

$$d_i = \frac{1}{log(1 + \frac{max(0, i-1)}{5}) + 1} \tag{4}$$

This changes Equation 2 into:

$$L(s_1, s_2) = \begin{cases} L(s_{1,1:}, s_{2,1:}) - d_i & s_{1,0} = s_{2,0} \\ 1 - d_i + min \begin{cases} L(s_{1,1:}, s_2) \\ L(s_1, s_{2,1:}) \\ L(s_{1,1:}, s_{2,1:}) \end{cases} & else \end{cases} \tag{5}$$

### C. Jaro

The Jaro similarity between two strings [21] considers the number of "matching" characters $m$ and the fraction of $m$ that are "transposed". Two characters $s_{1,i}$ and $s_{2,j}$ are considered matching if $s_{1,i} = s_{2,j}$ and if $i$ and $j$ are within a certain margin defined by Equation 6.

$$|i - j| \leq \frac{max(|s_1|, |s_2|)}{2} - 1 \tag{6}$$

The number of transpositions $t$ is determined by counting the number of matches where $s_{1,i} \neq s_{2,j}$ and dividing by 2.

Once $m$ and $t$ have been calculated, the Jaro similarity can be calculated using Equation 7.

$$sim_j = \begin{cases} 0 & m = 0 \\ \frac{1}{3}(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}) & m \neq 0 \end{cases} \tag{7}$$

### D. Jaro-Winkler

Jaro-Winkler is a variant of the Jaro similarity where two strings which match at the beginning are judged to be more similar than two strings which match at the end [22]. In Equation 8, $sim_{jw}$ is the Jaro-Winkler similarity, $sim_j$ is the Jaro similarity calculated by Equation 7, $\ell$ is the number of prefix characters which should be scaled if matching, and $p$ is a scaling factor. In Winkler's original work [22] defines $\ell = 4$ and $p = 0.1$ as standard; this paper adopts those values as well.

$$sim_{jw} = sim_j + \ell p(1 - sim_j) \tag{8}$$

### E. Ratcliff-Obershelp

The Ratcliff-Obershelp is a method of calculating string similarity by recursively locating common substrings and summing their lengths. To calculate the total number of matching characters $m$, the largest common substring (LCS) is found, and its length is added to $m$. Substrings are selected to the left and right of the LCS: one beginning with the first character of the string and ending at the first character of the LCS (non-inclusive), and one beginning at the character after the end of the LCS and ending at the end of the string. Either of these can be the empty string. Then, the LCS is found for each of these substrings. This continues until no further common substrings exist. At this point, $m$ can be substituted into Equation 9.

$$sim_{ro} = \frac{2m}{|s_1| + |s_2|} \tag{9}$$

### F. Our Proposed Metric: Modified Jaro-Winkler

When considering the vendors that Jaro-Winkler, which has the highest performance, does not correctly match to the NVD, a pattern emerges.

Consider the vendor string `pdq manufacturing, inc.` The correct CPE for this vendor is `pdqinc`. The Jaro-Winkler similarity between `pdq manufacturing, inc.` and `pdqinc` is $0.6836$. The CPE vendor with the highest Jaro-Winkler similarity to the vendor string `pdq manufacturing, inc.` is `phpmanufaktur`, with a similarity of $0.7612$. Even if we consider the 10 CPE vendors with the highest similarity to the vendor string, `pdqinc` does not appear among them.

In this example, the substring with the most information content ("pdq") comprises only a small fraction of the characters of the string. The discrepancy in length between the vendor string and the correct CPE depresses the similarity between them. Likewise, the commonality of the longer words "manufacturing" and "manufaktur" leads to the higher similarity of the incorrect suggestion.

To handle this, we propose a modified Jaro-Winkler metric which considers the information value of each token when the advisory vendor string is tokenized. We define the *information value* of a token as the inverse of its frequency in a corpus.

For this work, we defined our corpus as the full text of every published CISA ICS advisory. The corpus was tokenized and stemmed using the tokenizer and Snowball Stemmer from the Natural Language Toolkit (NLTK) for Python. From the corpus, we created a term frequency dataset by counting the number of appearances of each token in the corpus to support our proposed metric. We chose to create our own corpus because of the specialized vocabulary which appears in vendor names.

In the above example, the advisory vendor string `pdq manufacturing, inc.` is comprised of the tokens `pdq`, `manufacturing`, and `inc`. Of these tokens, `pdq` has the lowest frequency and thus the highest information value. The Jaro-Winkler similarity between `pdq` and `pdqinc` is $0.8833$, significantly higher than the previous highest similarity.

In our approach, each advisory vendor is tokenized using the NLTK `word_tokenize` method. If an advisory vendor string contains only one token, then the Jaro-Winkler similarity between the full advisory vendor string and the candidate CPE vendor string is calculated and returned. If the advisory vendor string consists of many tokens, then each token is stemmed using the NLTK Snowball Stemmer, and its information value is computed using the term frequency dataset described above. The token with the highest information value (i.e., the rarest token) is chosen. Finally, we compute and return the Jaro-Winkler similarity between that token and the candidate CPE vendor string.

## IV. Evaluation and Results

### A. Dataset

For this work, our dataset consisted of the ICS Advisories published by CISA on or before July 25, 2023, totaling 2364 advisories [23]. While security advisories from Acronis, for example, could be reasonably expected to always address Acronis products, advisories from CISA cover many vendors; this allows us to compare string similarity metrics on a large list of vendor names. Vendor names were pulled from the HTML page source of each advisory. Once extracted from the page source, all vendor names were converted to lowercase.

The vendor names present in CISA advisories were not standardized, particularly in regards to punctuation. One extreme example can be seen below, where the same entity was present in seven different advisory, with minor differences in punctuation each time.

> sensormatic electronics, a subsidiary of johnson controls inc.
> sensormatic electronics, llc, a subsidiary of johnson controls
> sensormatic electronics, llc, a subsidiary of johnson controls inc
> sensormatic electronics, llc, a subsidiary of johnson controls inc.
> sensormatic electronics, llc, a subsidiary of johnson controls, inc.
> sensormatic electronics, llc., a subsidiary of johnson controls, inc.
> sensormatic electronics, llc; a subsidiary of johnson controls

However, removing punctuation introduces more complexity, as which characters to include or exclude must be determined in the system design. Additionally, a CPE attribute can contain any properly-escaped punctuation character alongside alphanumeric characters. Some exploratory tests were performed to evaluate this; we found that removing all or even a subset of punctuation characters via substitution or regular

TABLE II
VENDORS AND ADVISORIES CORRECTLY IDENTIFIED PER-METRIC

| String Similarity Metric | # Vendors | % Vendors | % Advisories |
|---|---|---|---|
| Ratcliff-Obershelp | 337 | 74.23 | 88.92 |
| Jaro | 353 | 77.75 | 90.78 |
| Jaro-Winkler | 372 | 81.94 | 91.96 |
| Levenshtein | 326 | 71.81 | 88.37 |
| Discounted Levenshtein | 342 | 75.33 | 89.04 |
| Modified Jaro-Winkler | 380 | 83.70 | 92.39 |

expression resulted in markedly worse performance. For this reason, punctuation was not removed for the results presented in this paper.

The Python library `cleanco` was used to remove terms which indicate company type, like "inc.", "co.", or "LLC" [24]. Using `cleanco` to process the list of examples earlier in this section leaves just one string: "sensormatic electronics, a subsidiary of johnson controls". After processing, the list of vendors was deduplicated, leaving 454 unique vendors. Ground-truth data was manually collected by searching the NVD and reading CISA advisories to locate the correct CPE vendor value.

As mentioned previously, we considered as potential CPE vendor matches both those vendors present in the official CPE dictionary and those reported in CVEs.

### B. Results

For each metric, similarity measures were calculated between each vendor from the CISA ICS advisories and each vendor from the list of candidate CPE vendors. The vendor from the CPE dictionary with the highest similarity score to the advisory vendor was checked against the correct CPE vendor from our ground-truth list.

Table II shows the number of vendors correctly identified, the percentage of vendors correctly identified, and the percentage of advisories for which the vendor was correctly identified. The percentage of vendors and advisories differ because of the one-to-many relationship between vendors and advisories. One vendor may appear in multiple advisories; some vendors publish advisories more frequently than other vendors. Consider the example shown in Figure 2. If the CPE vendor for Vendor A was found and the CPE vendor for B was not, one-half of the vendors would have been correctly identified (i.e., % Vendor = 50.00%), but two-thirds of the advisories would have been correctly identified (i.e., % Advisories = 66.67%). This becomes an important distinction, particularly with the CISA ICS Advisories used, where, for example, Siemens is the vendor for 620 of the advisories in the dataset, much more than most vendors.

As it can be seen from Table II, among the existing metrics, Jaro-Winkler performs best. Our proposed metric, Modified Jaro-Winkler, performs even better than the basic Jaro-Winkler metric, showing the effectiveness of our design.

### C. Discussions

*1) Acronyms and Corporate Context:* Some companies use acronyms in their CPE names. A few examples can be seen in Table III. The exact string compared against the CPE dictionary is in the "CISA Vendor" column, and the "CPE Vendor" column lists the correct CPE vendor entry. The "Matched?" column shows whether the modified Jaro-Winkler metric described in Section III-F returns the correct CPE vendor. Consider Schweitzer Engineering Laboratories (SEL) with a CPE vendor of `selinc` and the National Renewable Energy Laboratory (NREL), whose CPE vendor name is `nrel`. The CISA vendor string for SEL does not include the acronym, while the CISA vendor string for NREL does. Accordingly, our hybrid similarity metric can correctly match `nrel` to the National Renewable Energy Laboratory, but not `selinc` to Schweitzer Engineering Laboratories.

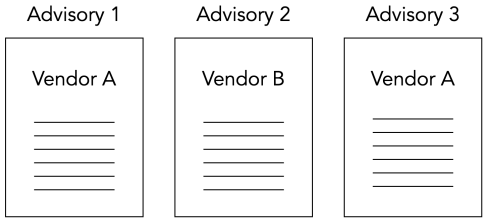


Fig. 2. An example to demonstrate the difference between the percentage of correctly-identified vendors vs. the percentage of correctly-identified of advisories. Correctly identifying Vendor A would give a % Vendor = 50% but a % Advisory = 66.67%.

TABLE III
EXAMPLES OF VENDOR CPES WITH ACRONYMS OR SHORTENINGS

| CISA Vendor | CPE Vendor | Matched? |
|---|---|---|
| national renewable energy laboratory (nrel) | nrel | Yes |
| 7-technologies | 7t | No |
| texas instruments | ti | No |
| schweitzer engineering laboratories | selinc | No |

*2) No Match:* Sixteen vendors from our dataset have no CPE at all. Despite this, the first suggestion returned for each of these when using Jaro-Winkler to compare against possible CPE vendors has a similarity score greater than 0.75. The strings "multiple" and "other", which appeared 4 and 18 times respectively, both returned matches with a similarity greater than 0.9. This highlights the difficulty of using a similarity threshold to select the correct CPE recommendation.

We chose not to remove these "NULL" matches from our dataset for two reasons. First, we wanted to more fully consider the edge cases that arise in this problem. More importantly, though, the "NULL" cases create a problem that cannot be easily solved even by a naive technical implementation.

The issues caused by acronyms and corporate structure that are discussed in the previous section can be addressed by creating a mapping table which contains acronyms, related company names, or other contextual data for a given vendor name. Then, string similarity methods could be used with this extra data.

By contrast, though, consider the difficulty of creating and maintaining a list of all companies who do *not* have an associated CPE vendor name. We find that using a threshold

on the similarity value is not a reliable method of detecting when an advisory vendor has no CPE vendor, and so more work is needed.

## V. Future Research Directions

It is not as straightforward as it may appear to identify an entity as the vendor, as there is little specification and no standardization about how to handle parent companies, subsidiaries, product families, or extended corporate hierarchies.

For instance, CISA ICS Advisory ICSA-17-234-04 reports the vendors as General Motors and Shanghai OnStar. In fact, OnStar is a subsidiary of General Motors and SIAC-GM (Shanghai General Motors). Nevertheless, the CPE vendor is reported as `gm` with the product `shanghai_onstar`.

The CPE dictionary is not an accurate representation of the current state of corporate mergers, acquisitions, or subsidiary structures. Detcon, a company which manufactures gas detection sensors, analyzers, and controllers, was previously owned by 3M but was sold to Teledyne in 2019 [25]. Detcon is listed in the NVD, but the vendor is still listed as `3m` with an product name of `detcon_sitewatch_gateway`.

CISA ICS Advisory ICSA-23-037-01 reports a vulnerability in the EnOcean SmartServer product, but the webpage for the associated CVE does not mention EnOcean at all and lists the CPE vendor as `echelon` and the product as `smartserver`. Searching for information on Echelon shows that it was acquired by Avesta in 2018, but there is no easily-seen link to EnOcean.

To address this, one future research direction is to combine CPE vendor and product searches. Rather than limiting the search to existing CPE vendors, there is merit in searching both existing vendors and existing product strings. This would significantly increase computation cost but would have a better chance of correctly identifying CPEs for these edge cases. Another future research direction is to construct full CPEs from the unstructured information in CISA advisories.

## VI. Conclusion

Identifying the affected assets in security advisories and mapping them onto a structured representation like the CPE is a difficult and time-consuming task. To automate this task, we explored the performance of several mainstream string similarity methods on matching vendors reported in the CISA ICS Security Advisories to vendors in the NVD's CPE dictionary. Additionally, we proposed a modified Jaro-Winkler metric that prioritizes tokens with a higher information value, and experiments showed that it out-performs all other metrics.

## Acknowledgement

This material is based upon work supported by the Department of Energy under Award Number DE-CR0000003.

## References

[1] *CVE Numbering Authority (CNA) Rules*. URL: https://www.cve.org/ResourcesSupport/AllResources/CNARules#section_7-3_cna_scope.

[2] *Product Identification*. URL: https://nvd.nist.gov/products.

[3] Emil Wåreus and Martin Hell. "Automated CPE Labeling of CVE Summaries with Machine Learning". In: *Detection of Intrusions and Malware, and Vulnerability Assessment*. Ed. by Clémentine Maurice et al. Cham: Springer International Publishing, 2020, pp. 3–22. ISBN: 978-3-030-52683-2.

[4] Luis Alberto Benthin Sanguino and Rafael Uetz. "Software vulnerability analysis using CPE and CVE". In: *arXiv preprint arXiv:1705.05347* (2017).

[5] Roman Ushakov et al. "CPE and CVE based Technique for Software Security Risk Assessment". In: *IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*. Vol. 1. 2021, pp. 353–356.

[6] Philip Huff et al. "A Recommender System for Tracking Vulnerabilities". In: *International Conference on Availability, Reliability and Security (ARES)*. 2021.

[7] L. Karl Branting. "A Comparative Evaluation of Name-Matching Algorithms". In: *Proceedings of the 9th International Conference on Artificial Intelligence and Law*. ICAIL '03. Scotland, United Kingdom: Association for Computing Machinery, 2003, pp. 224–232.

[8] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. "A Comparison of String Distance Metrics for Name-Matching Tasks". In: *International Conference on Information Integration on the Web (IIWEB)*. 2003, pp. 73–78.

[9] Peter Christen. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer Publishing Company, Incorporated, 2012. ISBN: 3642311636.

[10] Gabriel Recchia and Max Louwerse. "A Comparison of String Similarity Measures for Toponym Matching". In: *Proceedings of The First ACM SIGSPATIAL International Workshop on Computational Models of Place (COMP)*. 2013, pp. 54–61.

[11] S. Kempken, W. Luther, and T. Pilz. "Comparison of distance measures for historical spelling variants". In: *Artificial Intelligence in Theory and Practice*. Springer US, 2006, pp. 295–304.

[12] Nils Barlaug and Jon Atle Gulla. "Neural Networks for Entity Matching: A Survey". In: *ACM Trans. Knowl. Discov. Data* 15.3 (Apr. 2021).

[13] Dakota Dale, Kylie McClanahan, and Qinghua Li. "AI-based Cyber Event OSINT via Twitter Data". In: *International Conference on Computing, Networking and Communications (ICNC)*. 2023, pp. 436–442.

[14] Kylie McClanahan and Qinghua Li. "Automatically Locating Mitigation Information for Security Vulnerabilities". In: *IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. 2020, pp. 1–7.

[15] Philip Huff and Qinghua Li. "Towards Automated Assessment of Vulnerability Exposures in Security Operations". In: *EAI International Conference on Security and Privacy in Communication Networks (SecureComm)*. 2021, pp. 62–81.

[16] Fengli Zhang and Qinghua Li. "Dynamic Risk-Aware Patch Scheduling". In: *IEEE Conference on Communications and Network Security (CNS)*. 2020, pp. 1–9.

[17] Fengli Zhang et al. "A Machine Learning-based Approach for Automated Vulnerability Remediation Analysis". In: *IEEE Conference on Communications and Network Security (CNS)*. 2020, pp. 1–9.

[18] Kylie McClanahan et al. "When ChatGPT Meets Vulnerability Management: the Good, the Bad, and the Ugly". In: *IEEE Int'l Conf. on Computing, Networking and Communications (ICNC)*. 2024.

[19] Vladimir I Levenshtein. "Binary codes capable of correcting deletions, insertions, and reversals". In: *Soviet physics doklady*. Vol. 10. 8. 1966, pp. 707–710.

[20] Mathilde Boltenhagen, Thomas Chatain, and Josep Carmona. "A Discounted Cost Function for Fast Alignments of Business Processes". In: *Business Process Management*. Springer International Publishing, 2021, pp. 252–269.

[21] Matthew A. Jaro. "Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida". In: *Journal of the American Statistical Association* 84.406 (1989), pp. 414–420.

[22] William E Winkler. "String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage". In: (1990).

[23] *Cybersecurity Alerts & Advisories*. URL: https://www.cisa.gov/news-events/cybersecurity-advisories.

[24] *cleanco - PyPI*. URL: https://pypi.org/project/cleanco/.

[25] *3M Completes Sale of Gas and Flame Detection Business*. URL: https://news.3m.com/2019-08-01-3M-Completes-Sale-of-Gas-and-Flame-Detection-Business.