IEICE TRANS. INF. & SYST., VOL.E107–D, NO.5 MAY 2024

PAPER Special Section on Data Engineering and Information Management

Automated Labeling of Entities in CVE Vulnerability Descriptions with Natural Language Processing*

Kensuke SUMOTO[†a)], **Kenta KANAKOGI**[†b)], ***Nonmembers***, **Hironori WASHIZAKI**[†c)], ***Member***, **Naohiko TSUDA**[†d)], ***Nonmember***, **Nobukazu YOSHIOKA**[†e)], **Yoshiaki FUKAZAWA**[†f)], ***and*** **Hideyuki KANUKA**[††g)], ***Members***

SUMMARY Security-related issues have become more significant due to the proliferation of IT. Collating security-related information in a database improves security. For example, Common Vulnerabilities and Exposures (CVE) is a security knowledge repository containing descriptions of vulnerabilities about software or source code. Although the descriptions include various entities, there is not a uniform entity structure, making security analysis difficult using individual entities. Developing a consistent entity structure will enhance the security field. Herein we propose a method to automatically label select entities from CVE descriptions by applying the Named Entity Recognition (NER) technique. We manually labeled 3287 CVE descriptions and conducted experiments using a machine learning model called BERT to compare the proposed method to labeling with regular expressions. Machine learning using the proposed method significantly improves the labeling accuracy. It has an f1 score of about 0.93, precision of about 0.91, and recall of about 0.95, demonstrating that our method has potential to automatically label select entities from CVE descriptions.

key words: *technology, security knowledge repository, CVE, BERT, natural language processing, named entity recognition*

1. Introduction

Common Vulnerabilities and Exposures (CVE) [1] is a widely recognized security knowledge repository containing descriptions of vulnerabilities about software or source code. Each vulnerability entry in CVE is assigned a unique identifier, and the repository compiles the information, including a description of the vulnerability written in natural language. It is recommended that CVE descriptions are written using templates [2] (Fig. 1). Templates should include specific labels such as Attacker and Impact. However, the database is not structured since each entity is grouped as a single description. The absence of structure poses challenges when searching for specific entities. For example, CVE Details [3] compiles CVE information but it does not include Attacker and RootCause data. Search ease would greatly be enhanced if structured data were available for each entity.

- [VulnType] in [Component] in [Vendor] [Product] [Version] allows [Attacker] to [Impact] via [Vector].
- [Component] in [Vendor] [Product] [Version] [RootCause], which allows [Attacker] to [Impact] via [Vector].

Fig. 1 CVE templates

Herein we propose a method to automatically label select entities of CVE descriptions. We applied the Named Entity Recognition (NER) technique by BERT, a transformer-based model for natural language processing [4]. Then we experimentally validated our method by comparing it to labeling using regular expressions.

Additionally, we manually measured the percentage of descriptions that do and do not follow the CVE templates. We fine-tuned BERT model with template descriptions only or no template descriptions only, and compared the differences in accuracy.

This study aims to answer four research questions:

RQ1 How many descriptions conform to the CVE templates?
RQ2 Can each entity be automatically labeled in a CVE description?
RQ3 Does the accuracy differ between machine learning and regular expressions?
RQ4 Does the accuracy differ between descriptions that conform to the CVE templates and those that do not?

The proportion of CVE descriptions that do not follow templates has increased in recent years. Here, we labeled CVE descriptions with regular expressions with a precision of 0.79 and a recall as low as 0.62. In contrast, our proposed method using machine learning realized labeling with a precision of about 0.91 and a recall of about 0.95.

Our contributions are:

- Trend analysis: We analyzed trends in CVE descriptions, distinguishing between those that adhere to templates and those that do not.
- Automatic labeling methods: We propose methods to automatically label each entity in a CVE description.
- Creation of labeling data: We create labeling data for select labels: Attacker, RootCause, VulnType, Impact,

Manuscript received June 27, 2023.
Manuscript revised November 15, 2023.
Manuscript publicized February 9, 2024.

[†]The authors are with the Waseda University, Tokyo, 169–8555 Japan.
[††]The author is with the Hitachi, Ltd., Yokohama-shi, 244–0817 Japan.

*We build on the work of a poster presented at IEEE 23rd International Conference on Information Reuse and Integration for Data Science [5]. We extended this poster by increasing the amount of data and adding more experiments.

a) E-mail: sumoken.159@fuji.waseda.jp
b) E-mail: kanakogi-soft@fuji.waseda.jp
c) E-mail: washizaki@waseda.jp
d) E-mail: 821821@toki.waseda.jp
e) E-mail: nobukazuy@acm.org
f) E-mail: fukazawa@waseda.jp
g) E-mail: hideyuki.kanuka.dv@hitachi.com

DOI: 10.1587/transinf.2023DAP0013

Table 1 Comparison of our research to related works

Research	Labeling database	Method	f1 score (Reference value)	Target labels
Sun [6]	ExploitDB	NER (BERT) +QA-base (BERT) +Gazetteers	0.76 (NER) 0.82 (QA-base)	VulnType, Component, Vendor, Product, Version, RootCause, Attacker, Impact, Vector
Gao [7]	CVE	NER (Original)	0.88	Version, OS, Vendor etc.
Wang [8]	multiple security companies websites, government agencies, GitHub	NER (LSTM)	0.71	hacker organization, attack, sample file, security team, tool, time, purpose, area, industry, organization, way, loophole, features
This work	CVE	NER (BERT)	0.93	VulnType, RootCause, Attacker, Impact, Vector

and Vector.
- Accuracy of each label: We compare the accuracy of each label and confirm which entities are easily labeled.

The rest of this paper is structured as follows. Section 2 describes related works and the techniques used in our research. Section 3 discusses the terms related to our research and motivation. Section 4 details our method and experiment. Section 5 discusses the results. Section 6 provides a conclusion and future work.

2. Related Work

Yang *et al.* [9] proposed a method to label software names and versions based on CVE descriptions. Specifically, they used BERT, RoBERTa, and Electra for NER. Although their method substantially reduced the amount of training data compared with previous methods, the labeling accuracy was not markedly improved. They labeled the software name and version but did not label the security words. Because their study had different objectives than ours, a direct comparison is not feasible.

Sun *et al.* [6] covered nine labels with NER from ExploitDB to provide vulnerability information and create a CVE description automatically using these entities. They labeled each entity using three methods: NER-based machine learning, QA-based machine learning, and gazetteers. Although their study targeted some of the same labels (e.g., Attacker and Impact), how entities were labeled differed. They used ExploitDB, whereas we used the CVE descriptions.

Gao *et al.* [7] proposed a data and knowledge-driven cyber security NER method. They covered named labels such as version, OS, and vendor. Our study targeted other labels such as Impact. Additionally, Wang *et al.* [8] labeled threat intelligence reports to create a dataset called DNRTI and trained it with LSTM. Their study differs from ours in terms of labeling database and target labels.

Wei *et al.* [10] proposed an automatic labeling method for CVE descriptions called VE-Extractor. They covered six labels with high accuracy: cause, location, version, attacker, consequence, and operation. Consequence and operations are similar to vector and impact in our paper, respectively. They used the BERT-BiLSTM-CRF model to extract cause and used BERT Q&A to extract other entities. Although their RootCause (cause) labeling f1 score is higher, our study generated higher f1 scores for the other labels. In addition, they excluded CVE descriptions that were too short or did not describe sufficient vulnerability information from the dataset.

Guo *et al.* [11] investigated missing aspects of CVE descriptions and proposed a predictive method. Similar to our research, their analysis focused on the entities contained in the CVE descriptions.

Table 1 compares our study with the related previous works. The f1 scores in that table are for reference only because the labeling database, target labels, and other factors differ.

3. Background and Problems

3.1 CVE

CVE is a program established by the MITRE Corporation to identify, define, and catalog vulnerabilities in information security [1]. It contains detailed information about a vulnerability such as the CVE-ID and description. CVE descriptions recommend using templates [2]. Templates are composed of nine labels: VulnType, Component, Vendor, Product, Version, RootCause, Attacker, Impact, and Vector (Fig. 1). Table 2 lists the descriptions of the labels. Figure 2 shows an example of the description structure using CVE-2013-6449.

CVE descriptions combine these entities in a single description without a structure. Data must initially be labeled by entity to search or analyze by CVE entity. Consequently, our motivation is to create structured datasets of CVE entities to assist in searching or analysis.

Many CVE descriptions are written following templates. However, not all descriptions conform to the templates. Additionally, some CVE descriptions are missing entities or are written uniquely. For these reasons, labeling with regular expressions is difficult. Machine learning is a more suitable method.

3.2 NER

NER is a technique used to label specific character strings. Examples include the names of people or places in a text. BERT [4] is a model for natural language processing based on transformers and can also be used for NER. Each transformer

layer in BERT's model for NER has a fully connected layer. The fully connected layer consists of two parts: an input layer and an output layer. The default value for the number of units in the input layer is 768, while that for the output layer is the number of labels.

Yang *et al.* [9], Sun *et al.* [6], and Bridges *et al.* [12] proposed methods for NER related to cyber security. Liu *et al.* [13] introduced CrossNER, a manually labeled NER dataset with multiple entities defined for each of the five domains (not software security). In addition, large language models (LLMs) such as ChatGPT have emerged in the field of natural language processing. Tang *et al.* [14] suggested a method for NER from clinic texts using ChatGPT. According to their experiments, simply using ChatGPT significantly reduced accuracy compared to SOTA. Gonzalez *et al.* [15] proposed a method for extracting entities from historical documents. They concluded that there are some difficulties in performing NER with ChatGPT. For complex tasks such as NER, methods such as BERT may be more appropriate than simply using LLMs.

Table 2 Labels and their descriptions

Label	Description
VulnType	Vulnerability type
Component	Component including vulnerability
Vendor	Vendor name
Product	Product name
Version	Product version
RootCause	Main cause
Attacker	Attacker type (e.g., remote attacker)
Impact	Impact caused by vulnerability exploits
Vector	Inputs or processes to cause vulnerability

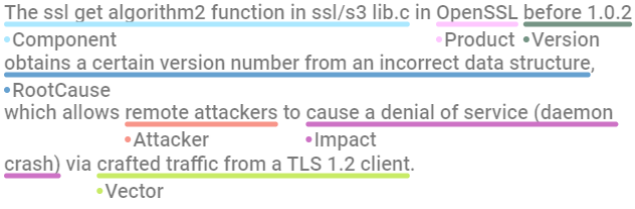

Fig. 2 Example of a CVE description structure (CVE-2013-6449)

4. Automatic Labeling

We propose a framework to label each entity of a CVE description by applying machine learning-based NER.

4.1 Proposed Method

Figure 3 overviews our proposed method. We customized a pre-training model, BERT, by fine-tuning it to accurately label select entities in CVE descriptions. By adapting this model, we can automatically assign entities to CVEs. We can also automatically label CVE descriptions using the model for our task. Our method has the following flow:

1. *Create training data*
   The training data requires tokens and labeling data about CVE descriptions. The training data was prepared in three steps.
   First, we divided the CVE descriptions into tokens using a tokenizer. Tokenization is the process of breaking up a description individual units, such as words or characters. Tokenization is done to convert CVE descriptions into a format that is easier for the model to process.
   Second, we labeled entities manually (Sect. 4.2). In this step, we labeled meaningful parts of the CVE descriptions, such as Attacker and Impact. The details of labeling procedure are described in Sect. 4.2.
   Third, we held labeling information and tokens as an array. This array is used as input data for fine-tuning the model.
2. *Train using the BERT model*
   BERT serves as the token-classification model, which is fine-tuned using the array of tokens and labeling data. BERT is a high-performing method in natural language processing and is suitable for our token-classification task.
   The name of the BERT model for the token-classification we used is "bert-base-uncased". It is the most popular

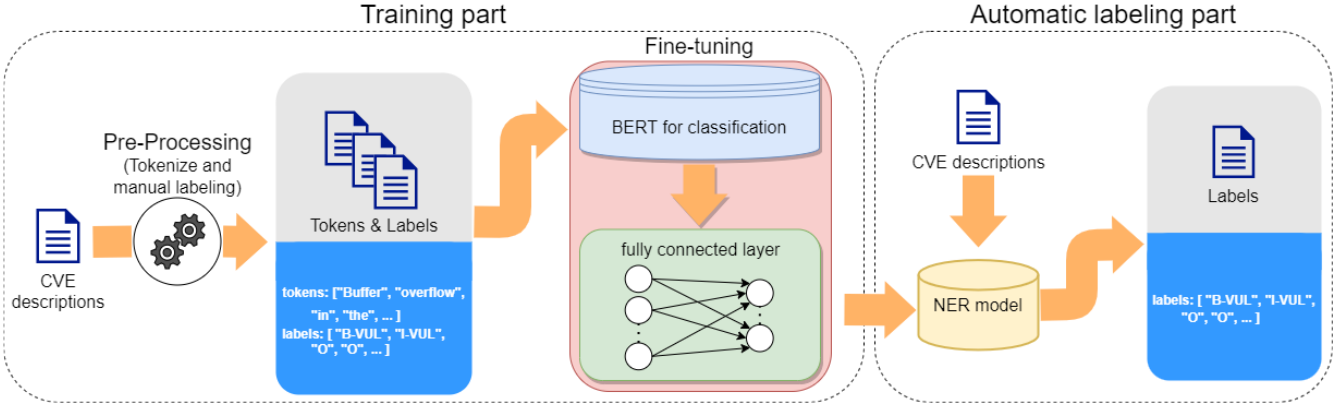

Fig. 3 Overview of our labeling method

BERT model, in which tokens are lowercased.
To optimize this BERT model for our research, we fine-tuned it using the array of tokens and labeling information.

3. *Automatic labeling*
Then CVE descriptions are loaded into the fine-tuned model to generate entity information. This information is used in the experiment to evaluate and determine the accuracy of automatic labeling.

4.2 Labeling Procedure

For our experiment, we prepared descriptions from Big-Vul, a CVE dataset created by Fan *et al.* [16]. We manually targeted five important labels of the CVE template: Attacker, RootCause, VulnType, Impact, and Vector. We chose these because only a few studies have attempted to label these five labels. In addition, these five labels are more useful for security analysis. Consequently, we needed data that included labeling information for these five labels as well as information about whether the CVE descriptions followed the template. We manually labeled these descriptions and verified whether they are templated.

Compared with the CVE templates, some CVE descriptions are distorted. Distortions include missing or additional entities. In this study, distorted CVE descriptions are counted as following the templates.

Common distortions of the CVE description templates include missing template entities. These are two main types of description omissions: either missing Vector or missing both RootCause and VulnType. Other distortions include adding more descriptions. Often these descriptions not only contain additional information but also other labels such as Impact or RootCause. Some descriptions used different writing styles. For example, some descriptions used "by [Vector]" instead of "via [Vector]".

5. Experimental Analysis

5.1 Overview

We used Big-Vul [16] as a dataset of CVE descriptions. Big-Vul contains various columns with vulnerable and modified code, which facilitated our research in the future. Among these, we used the CVE description. Although CVE templates have nine labels (Fig. 1), the experiment targeted five labels: Attacker, RootCause, VulnType, Impact, and Vector.

First, we created training data for fine-tuning by extracting CVE descriptions from Big-Vul. Then we performed tokenization and manual labeling. BIO tagging was adopted for the labeling. In addition, we assigned information on whether each CVE description followed the template. We created training data for 3287 CVE descriptions in BigVul. Big-Vul had a total of 3288 CVE descriptions, but only CVE-2019-5797 was "** RESERVED **" and a correct description had not yet been registered. Hence, it was the only one excluded in this study.

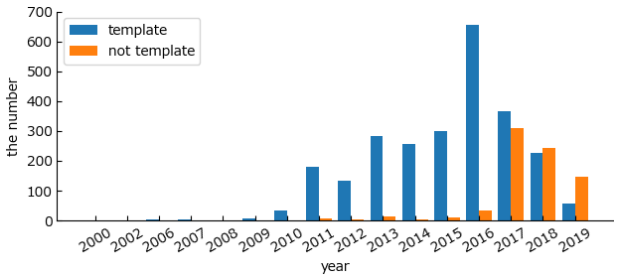

Fig. 4 Number of template descriptions per year

Then this training data was used in our experiments. We counted the number of CVE descriptions that followed the templates by year. Additionally, we fine-tuned BERT for token-classification. We used the "bert-base-uncased" pre-training model and performed fine-tuning using the source code provided by hugging face [17]. For comparison, we also performed regular expression labeling.

5.2 Results

5.2.1 EX1: Number of CVE Description According to the Templates

Among the 3287 labeled CVE descriptions, 2512 (76.4%) of the CVE descriptions followed the templates. The templates differed significantly by year (Fig. 4). Since 2017, the percentage of CVE descriptions that do not follow the template has increased.

5.2.2 EX2: Fine-Tuning

In this experiment, we prepared tokens and labeling information for 3287 CVE descriptions. Then 2629 descriptions were used as training data, and the remaining 658 were used as the evaluation data. The evaluation data were randomly selected, and the experiment was repeated five times. Figure 5 shows examples of automatic labeling results for the four potential outcomes: correctly labeled with template description, incorrectly labeled with template description, correctly labeled with no template description, and incorrectly labeled with no template description. In CVE-2018-6033, "Insufficient data validation" hits VulnType, but it was actually automatically labeled as RootCause. In CVE-2017-18187, "an integer overflow" was determined to be RootCause and Vector by automatic labeling, but its correct label is VulnType.

Table 3 shows the results of fine-tuning, where the accuracy is the average of five trials using seqeval [18] as the evaluation method. Our method labeled entities with an f1 score of **0.93**, a precision of **0.91**, and a recall of **0.95**. For the labels, Attacker had the highest precision, recall, and f1 score, while RootCause had the lowest values.

Table 4 shows the number of descriptions that our method labeled perfectly. The evaluation used 658 CVE descriptions because the remaining 2629 CVE descriptions

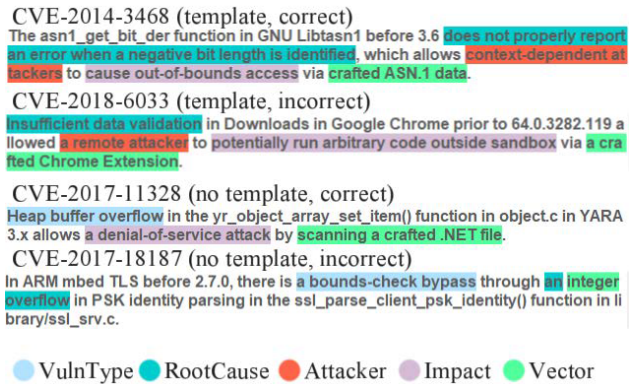

Fig. 5 Examples of automatic labeling results

Table 3 Automatic labeling results

Label	f1	Precision	Recall
Attacker	0.98	0.98	0.99
RootCause	0.82	0.78	0.88
VulnType	0.94	0.92	0.95
Impact	0.95	0.94	0.95
Vector	0.93	0.91	0.95
Overall	0.93	0.91	0.95

Table 4 Cross table of labeling results (BERT)

Dataset	Correct	Wrong	Percentage
Template	445	58	88.47%
No template	90	65	58.06%

were used for fine-tuning. It should be noted that this cross table shows the accuracy at the sentence level not sequence level. We performed this experiment only once. Our method perfectly labeled 88.47% of entities with template descriptions but only 58.06% of entities with no template descriptions.

5.2.3 EX3: Regular Expressions

The CVE descriptions were labeled with regular expressions. We used two patterns of regular expressions (Fig. 6). We used all the data in the evaluation because regular expressions do not require training data. Because a description may match both patterns, we verified whether the CVE descriptions matched pattern A and B separately. We generated the regular expressions with reference to Fig. 1. In pattern B, we expected Version and RootCause to be separated by a number. Thus, we separated them by '\d'. Table 5 shows the results of regular expressions. Table 6 shows how many descriptions are labeled perfectly at the sentence level. Named entity recognition using regular expressions could only perfectly extract 35.67% of the template descriptions. However, none of the no template descriptions were perfectly extracted.

5.2.4 EX4: Template Model vs. No Template Model

Next, we compared the training results on whether the CVE descriptions conformed to the templates. Of the 3287 CVE

A) [VulnType] in [Component] in [Vendor] [Product] [Version] allows [Attacker] to [Impact] via [Vector].

B) [Component] in [Vendor] [Product] [Version] [RootCause], which allows [Attacker] to [Impact] via [Vector].

```
patternA = r'(.*?) in (.*) (allow|allows|allowed|
    allowing) (.*) to (.*) (via|by) (.*)\.'
patternB = r'(.*?) in (.*)\d (.*), which (allow|
    allows|allowed|might allow) (.*) to (.*) (via|by
    ) (.*)\.'
```

Fig. 6 Regular expressions patterns of CVE templates

Table 5 Regular expression results

Label	f1	Precision	Recall
Attacker	0.88	0.98	0.81
RootCause	0.50	0.83	0.36
VulnType	0.49	0.44	0.55
Impact	0.75	0.91	0.63
Vector	0.65	0.71	0.59
Overall	0.69	0.79	0.62

Table 6 Cross table of labeling results (regular expressions)

Dataset	Correct	Wrong	Percentage
Template	896	1616	35.67%
No template	0	775	0.00%

Table 7 Number of training sets (CVE template or not)

Model	Number of data	Training data	Evaluation data
Template	2512	2009	503
No template	775	620	155
Tow-step fine-tuned no template model	775	620	155

Table 8 Training results (CVE templates or not)

Model	Accuracy	f1	Precision	Recall
Template	0.98	0.96	0.94	0.97
No template	0.92	0.64	0.57	0.75
Tow-step fine-tuned no template model	0.93	0.70	0.63	0.79

descriptions, 2512 followed templates, including those with distortions. We fine-tuned bert-base-uncased model with 2512 CVE descriptions that followed templates (template model) and 775 that did not follow templates (no template model). We also created a "two-step fine-tuned no template model" because the no template data was insufficient (Table 8). The two-step fine-tuned no template model was generated by fine-tuning BERT with the template descriptions to create the template model. Then this template model was fine-tuned with the no template descriptions. Figure 7 overviews the procedure to create the two-step fine-tuned no-template model. Table 7 details the dataset.

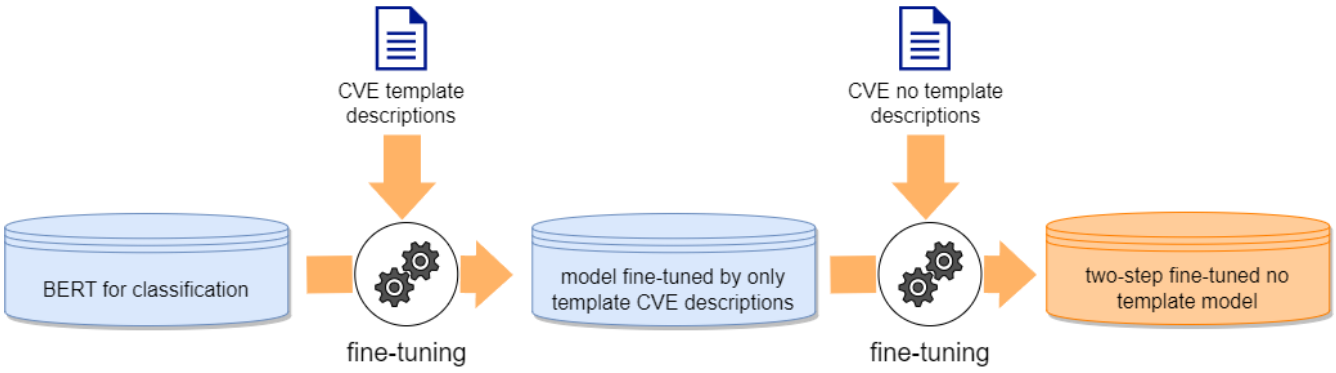

Fig. 7 Overview of creating the two-step fine-tuned no template model

5.3 Discussion

5.3.1 RQ1: How Many Descriptions Conform to the CVE Templates?

RQ1 is evaluated based on EX1. Of the 3287 CVE descriptions, 76.4% followed the templates, 23.6% did not. By year, the number of template descriptions overwhelmingly outnumbered the number of no template descriptions. However, the share of no template descriptions has increased since 2017. This increase may indicate that more people are participating in vulnerability reporting but are writing in a format that does not follow the templates.

76.4% of CVE descriptions follows the templates. Since 2017, the percentage of descriptions that follow the templates has decreased.

5.3.2 RQ2: Can Each Entity be Automatically Labeled in a CVE Description?

RQ2 discussion is based on EX2. Our method accurately labeled various entities of CVE descriptions. The f1, precision, and recall of entity labeling were approximately 0.93, 0.91, and 0.95, respectively (Table 3). Moreover, our method perfectly labeled about 88.47% of the template descriptions and 58.06% of the no template descriptions (Table 4). Labeling no template descriptions is more challenging because there are no fixed rules and they vary by description.

We considered the accuracy by label. Attacker was the most accurate and the easiest to label. This was attributed to two factors. First, Attacker was located in more fixed locations than the other ones. Second, it used similar words. VulnType, Impact, and Vector showed similar accuracies. VulnType was often used for specific words, making it well suited for the NER technique. Impact and Vector were often used in sets with certain prepositions, making them easier to locate.

RootCause had the lowest accuracy among the labels. This was attributed to two reasons. First, the pattern for RootCause was more complex than the other labels. Second, there was less data for RootCause since it only appeared in one of the templates.

Our method allows labeling with a high degree of accuracy as it labeled CVE descriptions with an f1 score of about 0.93, a precision of about 0.91, and a recall of about 0.95.

5.3.3 RQ3: Does the Accuracy Differ between Machine Learning and Regular Expressions?

RQ3 discussion is based on EX3. Overall, machine learning was more accurate than regular expressions (Tables 3 and 5). Some labels in regular expressions had a higher precision than machine learning, but the recall tended to be low for all labels, indicating that regular expressions often missed entities. Consequently, machine learning yielded higher f1 scores for all labels than those for regular expressions.

As shown in Table 6, NER with regular expressions perfectly labeled only about 35.67% of the template descriptions and none of the no template descriptions. In contrast, machine learning perfectly labeled about 88.47% of the template descriptions and 58.06% of the no template descriptions. Although increasing the number of regular expression patterns may increase the accuracy, it is easier to add more detailed rules in machine learning. Hence, machine learning is simpler and more accurate than regular expressions.

In addition, we considered the four remaining labels: Component, Vendor, Product, and Version. However, entities of these labels are difficult to extract with regular expressions for two reasons. First, clear rules to separate these labels do not exist. Unlike the other labels, there are not prepositions between Vendor, Product, and Version (Fig. 1). As for the component, it is difficult to determine where to separate “in” since it is very common. Second, entities may be omitted. For these reasons, we think that it is difficult to extract these entities using regular expressions compared to machine learning. However, some labels such as Version using numbers can be extracted with regular expressions. In the future, an extraction method using regular expressions

should be evaluated.

The method using regular expressions labeled CVE descriptions with an f1 score of about 0.69, a precision of about 0.79, and a recall of about 0.62, while our method shows an f1 score of 0.93, a precision of 0.91 and a recall of 0.95. These results show that machine learning is more accurate than regular expressions.

5.3.4 RQ4: Does the Accuracy Differ between Descriptions that Conform to the CVE Templates and those that Do Not?

RQ4 discussion is based on EX4. The model with descriptions following the templates had a high f1 score of about 0.96, precision of about 0.94, and recall of about 0.97 (Table 8). In contrast, the no template model gave a very low f1 score of about 0.64, a precision of about 0.57, and a recall of about 0.75. This suggests that descriptions following the templates have a certain degree of sentence formatting, making it easier to distinguish the characteristics and extract entities. Another reason for the low accuracy of the no template model may be the lack of training data because adequate training is not possible with only 775 data points.

To compensate for the lack of training data, we used the two-step fine-tuned no template model. This gave an f1 score of about 0.70, precision of about 0.63, and recall of about 0.79. Although the two-step fined-tuned no template model improved f1, precision, recall because the characteristics of the terminology related to vulnerability are discernible, f1, precision, recall remained low. One reason is that the description format varied widely, making it challenging to distinguish the description features.

The template model is more accurate, while no template model is less accurate. The two-step fine-tuned no template model is slightly more accurate than no template models.

5.4 Threats to Validity

Big-Vul is a dataset of CVEs related to only C/C++ vulnerabilities. If a vulnerability has a lower likelihood of occurrence in C/C++, training may be insufficient. Thus, it is preferable to prepare datasets in other programming languages.

We covered five of the nine labels in the CVE templates. However, these five labels may not be sufficient. The applicable label is ambiguous for some tokens in no templated CVEs. Therefore, a label classification beyond the templates may be necessary.

One person completed the labeling. It is possible that labeling errors are not thoroughly examined or labels for ambiguous tokens are not fully considered. This may be overcome by having multiple people check the label assignments.

The number of data is biased by the label. Compared to other labels, RootCause and VulnType tend to have smaller datasets because they are only found in one template, which may affect the training.

6. Conclusion and Future Work

Although CVE descriptions are composed of entities of several labels such as Attacker, Impact, and RootCause, this information is unstructured. This lack of structure makes it difficult to handle individual entities for searching, grouping, or analyzing. To overcome this issue, this study proposes a method to automatically label CVE descriptions by applying NER. We used BERT for token-classification to automatically label entities. We also experimented with regular expressions to compare with BERT. In addition, we verified whether CVE descriptions follow the templates by comparing the training results with CVE descriptions that follow templates to those that do not.

The experiment found that 76.4% of the CVE descriptions of the 3287 cases prepared followed the templates. The proposed method successfully labeled CVE descriptions with a precision of about 0.91 and a recall of about 0.95. In addition, descriptions that followed the templates are easier to label than those that do not. The two-step fine-tuned no template model shows an improved accuracy relative compared with the no template model. A comparison of labeling accuracy by label revealed that Attacker has the highest f1 score, followed in order by Impact, VulnType, Vector, and RootCause.

In the future, the accuracy should be improved. The labeling results indicate that precision and recall can be enhanced sufficiently for some entities. Another option may be to use our labeling model. For example, a study could be conducted to propose the missing entities using our labeling model. We also intend on increasing the number of CVE descriptions used in our experiments. Currently, labeling is performed only for CVE descriptions included in Big-Vul. In the future, we plan to consider CVE descriptions not included in Big-Vul to increase the data and evaluate the accuracy.

Our method facilitates the labeling of data where the entity is structured from CVE descriptions. Such data may be useful in research. For example, it is possible to search for individual entities by specific content. Grouping such data may be beneficial for further analysis. In addition, these data could be used to automatically generate CVE descriptions.

References

[1] The MITRE, “Common Vulnerabilities and Exposures (CVE),” [no date info]. https://cve.mitre.org/ (accessed: 2022-1-12).
[2] J. Evans, “Key details phrasing,” [no date info]. http://cveproject.github.io/docs/content/key-details-phrasing.pdf (accessed: 2022-1-9).

[3] CVE Details, "CVE Details," [no date info]. https://www.cvedetails.com/ (accessed: 2022-3-15).
[4] J. Devlin, M.W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.
[5] K. Sumoto, K. Kanakogi, H. Washizaki, N. Tsuda, N. Yoshioka, Y. Fukazawa, and H. Kanuka, "Automatic labeling of the elements of a vulnerability report cve with nlp," 2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI), pp.164–165, IEEE, 2022.
[6] J. Sun, Z. Xing, H. Guo, D. Ye, X. Li, X. Xu, and L. Zhu, "Generating Informative CVE Description From ExploitDB Posts by Extractive Summarization," arXiv preprint arXiv:2101.01431, 2021.
[7] C. Gao, X. Zhang, and H. Liu, "Data and knowledge-driven named entity recognition for cyber security," Cybersecurity, vol.4, no.1, pp.1–13, 2021.
[8] X. Wang, X. Liu, S. Ao, N. Li, Z. Jiang, Z. Xu, Z. Xiong, M. Xiong, and X. Zhang, "DNRTI: A Large-Scale Dataset for Named Entity Recognition in Threat Intelligence," 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp.1842–1848, IEEE, 2020.
[9] G. Yang, S. Dineen, Z. Lin, and X. Liu, "Few-sample named entity recognition for security vulnerability reports by fine-tuning pre-trained language models," International Workshop on Deployable Machine Learning for Security Defense, pp.55–78, Springer, 2021.
[10] Y. Wei, L. Bo, X. Sun, B. Li, T. Zhang, and C. Tao, "Automated event extraction of cve descriptions," Information and Software Technology, vol.158, p.107178, 2023.
[11] H. Guo, S. Chen, Z. Xing, X. Li, Y. Bai, and J. Sun, "Detecting and augmenting missing key aspects in vulnerability descriptions," ACM Transactions on Software Engineering and Methodology (TOSEM), vol.31, no.3, pp.1–27, 2022.
[12] R.A. Bridges, C.L. Jones, M.D. Iannacone, K.M. Testa, and J.R. Goodall, "Automatic labeling for entity extraction in cyber security," arXiv preprint arXiv:1308.4941, 2013.
[13] Z. Liu, Y. Xu, T. Yu, W. Dai, Z. Ji, S. Cahyawijaya, A. Madotto, and P. Fung, "Crossner: Evaluating cross-domain named entity recognition," Proc. AAAI Conference on Artificial Intelligence, vol.35, no.15, pp.13452–13460, 2021.
[14] R. Tang, X. Han, X. Jiang, and X. Hu, "Does synthetic data generation of llms help clinical text mining?," arXiv preprint arXiv:2303.04360, 2023.
[15] C.-E. González-Gallardo, E. Boros, N. Girdhar, A. Hamdi, J.G. Moreno, and A. Doucet, "Yes but.. can chatgpt identify entities in historical documents?," arXiv preprint arXiv:2303.17322, 2023.
[16] J. Fan, Y. Li, S. Wang, and T.N. Nguyen, "A C/C++ Code Vulnerability Dataset with Code Changes and CVE Summaries," Proc. 17th International Conference on Mining Software Repositories, pp.508–512, 2020.
[17] Hugging Face, "huggingface/transformers/examples/pytorch/token-classification/," [no date info]. https://github.com/huggingface/transformers/tree/master/examples/pytorch/token-classification (accessed: 2022-1-16).
[18] Hugging Face, "seqeval," [no date info]. https://huggingface.co/spaces/evaluate-metric/seqeval (accessed: 2022-6-6).

Kensuke Sumoto received the B.E. degree in Computer Science and Engineering from Waseda University, Tokyo, Japan, in 2022. He is now a master course student of Department of Computer Science and Communications Engineering, Waseda University. His research interests include analysis of vulnerability reports with natural language processing.

Kenta Kanakogi received the B.E. degree in Communications and Computer Engineering and the M.E. degree in Computer Science and Communications Engineering from Waseda University, Tokyo, Japan, in 2020 and 2022, respectively. His research interests include analysis of vulnerability reports with natural language processing.

Naohiko Tsuda received the B.E. and M.E. degrees in Computer Science and Engineering and the D.E. degree in Computer Science and Communications Engineering from Waseda University, Tokyo, Japan, in 2013, 2014, and 2020, respectively. He is now an adjunct researcher of Global Software Engineering Laboratory of Green Computing Systems Research Organization at Waseda University. His research interests include software engineering especially quantitative measurement and evaluation for software quality.

Hironori Washizaki received his Doctoral degree in information and computer science from Waseda University in 2003. Currently, he is a Professor and the Associate Dean of the Research Promotion Division at Waseda University in Tokyo, a Visiting Professor at the National Institute of Informatics, and an Advisor at the University of Human Environments. He also works in industry as an Outside Director of eXmotion. His research interests include reliable and intelligent software engineering, machine learning engineering, and ICT education. He is leading a professional IoT/AI/DX education project called SmartSE. He has served as Chair of IPSJ SIGSE and Convenor of ISO/IEC/JTC1 SC7/WG20. He has been elected IEEE Computer Society 2025 President.

Nobukazu Yoshioka is a senior researcher/Professor at the Research Institute for Science and Engineering at Waseda University, Japan. Dr. Nobukazu Yoshioka received his B.E. degree in Electronic and Information Engineering from Toyama University in 1993. He received his M.E. and Ph.D. degrees in School of Information Science from Japan Advanced Institute of Science and Technology in 1995 and 1998, respectively. From 1998 to 2002, he was with Toshiba Corporation, Japan. From 2002 to 2004 he was a researcher, and from 2004 to 2021, he had been an associate professor of National Institute of Informatics, Japan. Since 2021, he has been a Senior Researcher of Waseda Research Institute for Science and Engineering, Waseda University, Japan. His research interests include Security and Privacy Software Engineering and Software Engineering for Machine Learning-based Systems. He is a member of the Information Processing Society of Japan (IPSJ), the Institute of Electronics, Information and Communication Engineers (IEICE) and Japan Society for Software Science and Technology (JSSST), the Japanese Society for Artificial Intelligence (JSAI) and IEEE CS. He had been a board member of a SIG of Machine Learning Systems Engineering since 2018, a board member of JSSST from 2011 to 2015 and an auditor of JSSST from 2017 to 2021. He had been a chair of IEEE CS Japan Chapter in 2017, 2018 and 2020.

Yoshiaki Fukazawa received the B.E., M.E. and D.E. degrees in electrical engineering from Waseda University, Tokyo, Japan in 1976, 1978 and 1986, respectively. He is now a professor at the Department of Information and Computer Science, Waseda University. His research interests include software engineering, especially the reuse of object-oriented software and agent-based software. He is a member of IPSJ, IEICE, JSSST, ACM and IEEE.

Hideyuki Kanuka is a Chief Researcher at Hitachi, Ltd., Research and Development Group, Japan. He received his B.E. from Musashi Institute of Technology in 2001 and M.E. from Tokyo Institute of Technology in 2003. His research interests include software engineering, especially software architecture and testing.