Applied Intelligence (2024) 54:2881–2899
https://doi.org/10.1007/s10489-023-05262-4

Comprehensive vulnerability aspect extraction

Qindong Li[1,2] · Wenyi Tang[1,2] · Xingshu Chen[1,2,3] · Song Feng[1] · Lizhi Wang[1]

Accepted: 28 December 2023 / Published online: 8 February 2024

Abstract
Extracting valuable information from unstructured vulnerability reports constitutes a fundamental task in numerous cybersecurity applications. Existing approaches necessitate the creation of new extraction models and data labeling efforts, also inadvertently leading to duplicated information extraction. Therefore, we devote to extracting almost all valuable aspects at in a single sweep to benefit most downstream tasks. However, comprehensive extraction is challenging, which not only increases the boundaries of the aspects to be located but also reduces the number of learnable words in a vulnerability report. In this paper, we propose the **Vul**nerability **P**ortrait **A**utomatic **G**enerator (Vul-PAG), designed to facilitate comprehensive vulnerability aspect extraction by capturing and amalgamating word's multi-view information. It encompasses a split-reorganization mechanism based on the wordpiece mechanism to capture the internal writing feature of words alongside a MidConst task to grasp the syntactic feature of words. Further, we fuse them with the semantic feature output from the context-dependent language model to bolster the word's representation ability. Furthermore, we present the first-ever dataset crafted for the comprehensive extraction of vulnerability aspects, containing 2200 descriptions and encompassing eight distinct aspects. Extensive experimental results show that Vul-PAG outperforms state-of-the-art methods by 3.47, 2.68, and 3.07 in terms of precision, recall, and F1-score, respectively.

Keywords Cyber security · Vulnerability reports · Vulnerability aspect extraction · Multi-view information

1 Introduction

Maliciously exploited vulnerabilities would expose users, organizations, and countries to significant risk, with examples of WannaCry ransomware [36] crippling thousands of machines in hospitals and schools, and the Stuxnet worm [31] damaging the nation's critical infrastructure. Efforts to build a comprehensive vulnerability aspect information database based on vulnerability reports can play a pivotal role in mitigating the exploitation of vulnerability. However, some crucial vulnerability aspects, such as impact [33] and attack vector [21] reside within unstructured vulnerability description [30]. These information cannot be directly applied to downstream tasks like cybersecurity risk assessment [1, 15] (as shown in Fig. 1) and attack scenario reconstruction. Consequently, numerous researchers have designed schemes to extracting vulnerability aspect [6, 44].

Vulnerability aspect extraction refers to assigning correct aspect tag to each word in the description, which framing it as a sequence labeling task [9, 25]. As the description of CVE-2004-2368 shown in Fig. 2(a), the tags "product" and "version" are assigned to words "Opt-X" and "0.7.2", respectively. Compared to existing models extract limited aspects tailored to specific downstream tasks [2, 42], we aim to comprehensively extract vulnerability aspects to benefit a broader spectrum of downstream tasks. Comprehensive extraction presents challenges by increasing the number of aspect boundaries to be identified while reducing the number of remaining learnable words, it requires the extraction model to mine more information to support the localization. Conventional extraction models struggle to obtain sufficient information to support comprehensive extraction solely from the semantic information of the remaining words. As shown in Fig. 2(c), there are sufficient learnable words "for vim" and "which might allow" to locate the root cause "allows dangerous functions $\cdots$"(red). However, in the comprehensive extraction scenario shown in Fig. 2(b), there is no learnable

✉ Wenyi Tang
wtang@scu.edu.cn

1 School of Cyber Science and Engineering, Sichuan University, Chengdu 610065, China

2 Key Laboratory of Data Protection and Intelligent Management (Sichuan University), Ministry of Education, Chengdu 610065, China

3 Cyber Science Research Institute, Sichuan University, Chengdu 610065, China

word between the root cause(red) and the product(cyan) to provide localization information. Furthermore, the syntactic feature of root cause(red) include taking product(cyan) as subject, and its content includes a verb and subject-corresponding object. Conventional methods cannot capture such information, and fail to extract the aspect, as shown in Fig. 2(d).

To address these challenges, we propose a comprehensive vulnerability aspect extraction framework that fully leverages out-of-vocabulary(OOV) words and additionally integrates multi-view word information. These features mainly involve the word's internal and sentence-level, specifically writing feature and syntactic feature. Writing feature primarily pertain to OOV words. Based on the wordpiece mechanism, we search for the optimal subwords that can reflect the word's category and then weighted reorganize subwords to express OOV words, forming the writing feature. For syntactic feature, a syntactic parser abstracts real descriptions and vulnerability templates into MidConst sequences and constructs a specialized dataset using regular expressions. We utilize the output of different layers of the language model to train a classifier capable of recognizing these sequences, selecting the one with the best classification performance as the syntactic feature. As shown in Fig. 2(d), the addition of syntactic information significantly enhances tagging performance, Finally, we fuse the aforementioned features with regular semantic feature for encoding with Bi-LSTM and decoding via CRF. The model is trained and validated on a novel CVE aspects dataset, encompassing 2200 CVE descriptions and covering major aspects.

In summary, our contributions are as follows:

(1)We introduce Vul-PAG, a comprehensive vulnerability aspect extraction framework that aims to extract nearly all vulnerabilities to benefit a wide range of downstream security tasks. To the best of our knowledge, this is the first work in comprehensive vulnerability aspect extraction.

(2) A split-reorganization mechanism is devised to preserve writing information with respect to OOV words. It uses the wordpiece mechanism to voraciously search for subwords that can express the written information. The weighted reorganization eliminates the inconsistency of sequence length caused by splitting, and assigns higher weights to important subwords to retains the writing feature of words.

(3) We integrate multi-view information, including syntactic, semantic, and writing features, to enhance the model's ability to locate aspect boundaries. Specifically, using a context-dependent language model to obtain semantic feature, gain writing feature based on the proposed split-reorganization mechanism, and capture syntactic feature by a designed MidConst task.

(4) As far as we know, it is the first time to publist a CVE comprehensive aspects dataset, comprising 2200 CVE descriptions and up to 8 aspects. Extensive experiments demonstrate the superiority of Vul-PAG over state-of-the-art approaches. To foster future work in this area, we release our source code and dataset at https://github.com/dong-xs/Vul-PAG-new and https://github.com/dong-xs/Vul-PAG-dataset.

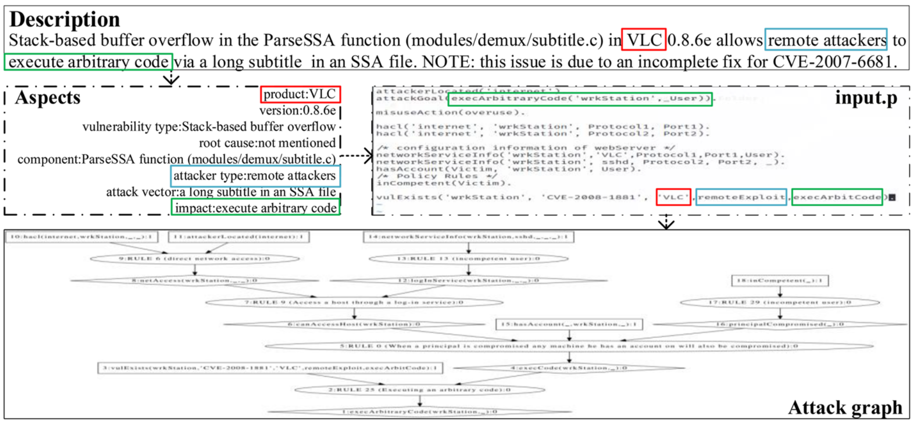
2 Related works

This section presents relevant research in two key domains: Natural Language Processing (NLP) for security and aspect extraction methods.

2.1 NLP for security

NLP techniques have been employed to extract threat entities and behaviors from various Open Source Cyber Threat Intelligence (OSCTI) sources such as technical blogs and vulnerability scan reports. This research aids in gaining into cyber threats, linking independent Indicators of Compromise (IOCs), and describing cybersecurity campaigns [11, 13, 19, 20, 43, 46]. This research also contributes to threat hunting [12, 38] and measuring intelligence inconsistency [5, 23]. These techniques are also been applied to domain-specific texts, including technical blogs [24, 29], academic papers for Android malware feature extraction [45], and social media for vulnerability exploit code extraction [7]. Researchers have also focused on API documentation and underground forums [28, 39].

A significant body of work concentrates on analyzing vulnerability reports and extracting various aspects using NLP. For example, [5] employ Named Entity Recognition (NER) and Relation Extraction (RE) models to extract vulnerability entities and relationships from multiple reports. Guo et al. [17] design an end-to-end NER system to extract key aspects from different vulnerability databases, merging aspect categories to leverage complementary information from various sources. [10] use NLP techniques to generate network signatures from unstructured vulnerability reports. Binyamini et al. [1] propose an end-to-end automated framework for modeling new attack techniques from textual vulnerability descriptions, enabling real-time system defense. Costa et al. [4], Sharma et al. [33] using only the vulnerability description as input to guide vulnerability metrics. Several studies aim to maximize the value of intelligence within vulnerability databases [26, 41]. These works collectively illustrate the wide-ranging applications of NLP in enhancing cybersecurity, with a specific focus on extracting valuable insights from unstructured security-related text data, including vulnerability reports.



Description

Stack-based buffer overflow in the ParseSSA function (modules/demux/subtitle.c) in VLC 0.8.6e allows remote attackers to execute arbitrary code via a long subtitle in an SSA file. NOTE: this issue is due to an incomplete fix for CVE-2007-6681.

Aspects

product:VLC
version:0.8.6e
vulnerability type:Stack-based buffer overflow
root cause:not mentioned
component:ParseSSA function (modules/demux/subtitle.c)
attacker type:remote attackers
attack vector:a long subtitle in an SSA file
impact:execute arbitrary code

input.p

```
attackerLocated('internet').
attackGoal(execArbitraryCode('wrkStation',_User)).

misuseAction(overuse).

hacl('internet', 'wrkStation', Protocol1, Port1).
hacl('internet', 'wrkStation', Protocol2, Port2).

/* configuration information of webServer */
networkServiceInfo('wrkStation','VLC',Protocol1,Port1,User).
networkServiceInfo('wrkStation', sshd, Protocol2, Port2, _).
hasAccount(Victim, 'wrkStation', User).
/* Policy Rules */
inCompetent(Victim).

vulExists('wrkStation', 'CVE-2008-1881', 'VLC',remoteExploit,execArbitCode).
```

Attack graph
10:hacl(internet,wrkStation,_,_):1
11:attackerLocated(internet):1
14:networkServiceInfo(wrkStation,sshd,_,_,_):1
9:RULE 6 (direct network access):0
13:RULE 13 (incompetent user):0
8:netAccess(wrkStation,_,_):0
12:logInService(wrkStation,_,_):0
18:inCompetent(_):1
7:RULE 9 (Access a host through a log-in service):0
17:RULE 29 (incompetent user):0
6:canAccessHost(wrkStation):0
15:hasAccount(_,wrkStation,_):1
16:principalCompromised(_):0
5:RULE 0 (When a principal is compromised any machine he has an account on will also be compromised):0
3:vulExists(wrkStation,'CVE-2008-1881','VLC',remoteExploit,execArbitCode):1
4:execCode(wrkStation,_):0
2:RULE 25 (Executing an arbitrary code):0
1:execArbitraryCode(wrkStation,_):0

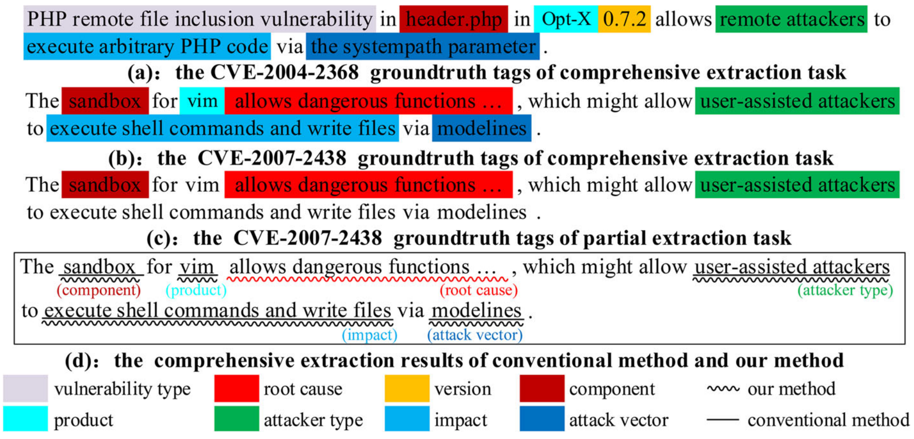
Fig. 1 An example of generating an attack graph from extracting vulnerability aspects from description. It uses a vulnerability description written in natural language as input: (1) extract vulnerability aspects such as "attacker type", and "impact"; (2) build the input file input.p of the MulVal platform based on these aspects; (3) combined with the inference rules of the platform to generate attack graph, which reveals the attack paths in the host

2.2 Aspect extraction methods

As previously discussed, extracting aspects from Cyber Threat Intelligence (CTI) reports is a prerequisite for various subsequent tasks. The specific methods for aspect extraction are broadly categorized into rule-based, statistical machine learning-based, and deep learning-based approaches.

Rule-based methods are used to achieve the extraction of IOCs (e.g., IP, domain names, HASH, etc.) with specific rules. This method requires a lot of prior knowledge to formulate rule templates, which can be achieved by entity comparison, regular expression, etc. Therefore, inefficiency and non-scalable are the main drawbacks of the method. The statistical machine learning-based method is to build models by fusing language models and statistical machine learning algorithms. However, these approaches heavily depend on expert knowledge for feature engineering, which limits their applicability. Examples of implementation include the use of Maximum Entropy models and Conditional Random Fields. Deep learning, a subset of machine learning, has made significant contributions [16] to aspect extraction and knowledge mapping in various domains, including text knowledge extraction, image classification, and disease diagnosis [14, 18]. In the knowledge extraction task, deep learning-based is often treated as a sequence tagging task, typically following the "Embedding+Encode+Decode" framework. Language models such as Word2vec [32], FastText [5], and Glove [33] are used to convert words into numerical vectors in embedding phase. Addressing the issue of embedding OOV words is a key challenge, and methods like char embedding [5] and the wordpiece mechanism [40] have been proposed as solutions. Recurrent Neural Networks (RNN), specifically Bidirectional Long Short-Term Memory (Bi-LSTM) [1] and Bidirectional Gated Recurrent Unit (Bi-GRU) [5], are utilized to capture dependencies between words and encoded as hidden states. Consideration for variable entity lengths has led to the proposal of multi-granularity-based methods for IOC identification [33]. Conditional Random Fields (CRF) are often employed to decode the hidden states generated by the neural network and produce the optimal tagging sequence.

Deep learning-based methods is the order of the day, but existing techniques mainly leverage rich information around aspect boundaries. However, this approach becomes less effective when comprehensive extraction introduces more boundary locations and reduces contextual information



PHP remote file inclusion vulnerability in header.php in Opt-X 0.7.2 allows remote attackers to execute arbitrary PHP code via the systempath parameter .

(a): the CVE-2004-2368 groundtruth tags of comprehensive extraction task

The sandbox for vim allows dangerous functions … , which might allow user-assisted attackers to execute shell commands and write files via modelines .

(b): the CVE-2007-2438 groundtruth tags of comprehensive extraction task

The sandbox for vim allows dangerous functions … , which might allow user-assisted attackers to execute shell commands and write files via modelines .

(c): the CVE-2007-2438 groundtruth tags of partial extraction task

The sandbox for vim allows dangerous functions … , which might allow user-assisted attackers to execute shell commands and write files via modelines .

(d): the comprehensive extraction results of conventional method and our method

Fig. 2 Groundtruth tag sequences of partial and comprehensive extraction task and comprehensive extraction results of conventional method and our method

around aspects. To address these challenges, the utilization of additional information, such as syntactic and writing, is necessary. OOV words are often overlooked, limiting the learning of valuable information.

To enhance the utilization of OOV words and leverage the wordpiece mechanism's effectiveness in handling OOV words, a wordpiece-based split-reorganization mechanism has been proposed to represent OOV words accurately. Additionally, multi-view information extraction, inspired by strategies for representing entities [34, 35], is employed to enhance the word's representation [37]. To address the scarcity of datasets containing comprehensive aspects and to enhance information quantity, a novel dataset is created with an expanded vulnerability aspect category and increased sample size. These developments aim to improve aspect extraction methods in the context of vulnerability reports.

3 Problem formulation

MITRE recommends vulnerability descriptions written in two specific templates [8]:

(1) [Vulnerability Type] in [Component] in [Vendor] [Product] [Version] allows [Attacker Type] to [Impact] via [Attack Vector];

(2) [Component] in [Vendor] [Product] [Version] [Root Cause], which allows [Attacker Type] to [Impact] via [Attack Vector].

A complete CVE description in these templates consists of nine aspects: vulnerability type, vendor, affected software, version, component, root cause, attacker type, impact, and attack vector. We take 8 aspects (excluding "vendor") as target. Reasons for the exclusion of "vendor" include: (1) it occurs close to "product", with no writing features within aspects and lacks locational boundaries between aspects; and (2) as a unique identifier of an organization, it is more efficiently extracted by constructing a dictionary from https://www.cvedetails.com/vendor.php for comparison.

The research assume the existence of a set of descriptions denoted as $S = [s^{(1)}, s^{(2)}, \cdots, s^{(m)}]$, where $s^{(i)}$ is the i-th description. Take $s^{(i)}$ as example, and using specific characters (e.g. space, period, comma) to slice it into a word sequence, denoted as $s^{(i)} = [s_1^{(i)}, s_2^{(i)}, \cdots, s_n^{(i)}]$, where n represents the number of words. Each word will be tagged with a tag, and the tag sequence donated as $y^{(i)} = [y_1^{(i)}, y_2^{(i)}, \cdots, y_n^{(i)}]$. The tag predictor reads description set S for model training, and giving description $s^{(i)}$ to the model to generates tag sequence $y'^{(i)} = [y'^{(i)}_1, y'^{(i)}_2, \cdots, y'^{(i)}_n]$. Finally, the difference between generated tag sequence $y'^{(i)}$ and ground truth tag sequence $y^{(i)}$ is used as the training signal to optimize model parameters.

4 Methodology

Extracting specific and accurate information from unstructured text presents formidable challenges in the intricate landscape of textual corpora. To effectively address the challenges in Section 1 and fulfill stringent accuracy requirements, we propose a framework called Vul-PAG(**Vul**nerability **P**ortrait **A**utomatic **G**enerator). This section describes its structure and how it works in detail.

4.1 Architecture

The architecture of Vul-PAG is shown in Fig. 3, which follows the pipeline including word indexing, word embedding, features construction, features fusing, and sequence tagging. Given a word sequence, word indexing is responsible for mapping it into an index sequence by a maintained dictionary; word embedding is responsible for transforming it into a matrix using the language model; features construction is responsible for constructing multi-view features of a word and features fusing would fuse these features to represent a word; sequence tagging is responsible for assigning correct tag for each word. We take the description of CVE-2002-0427 as input to demonstrate the framework comprehensively.

4.2 Framework implementation

Given a description $D = [Buffer, overflows, \cdots, privileges]$ contains n words, it must be converted into numeric format, a task facilitated using a language model. Given that context-independent language models (e.g., Word2Vec) require retraining prior to use, we have opted for BERT as our language model.

Word indexing BERT initially maps each word to an index value through a maintained dictionary. However, this dictionary cannot directly map all words, i.e., OOV words. For OOV words, there are two primary mapping schemes: the coarse-grained BERT replaces OOV words with "[UNK]" tag and assigns uniform indexes, the medium-grained BERT assigns corresponding indexes after splitting the OOV words into indexable subwords. The former captures word's feature at the sentence level by treating each word as a whole; the latter dissects OOV words to capture their internal features at the writing level, both subsequently serve our task.

The module translates D into a subword sequence $D^{med} = [[B, \#\#uff, \#\#er], [over, \#\#flow, \#\#s], \cdots, privileges]$ with a wordpiece-based tokenizer and indexes it as $IND^{med} = [[ind_1^{(0)}, ind_1^{(1)}, ind_1^{(2)}], [ind_2^{(0)}, ind_2^{(1)}, ind_2^{(2)}], \cdots, ind_n]$. Simultaneously, it uses a coarse-grained tokenizer to replace the OOV word in the origi-

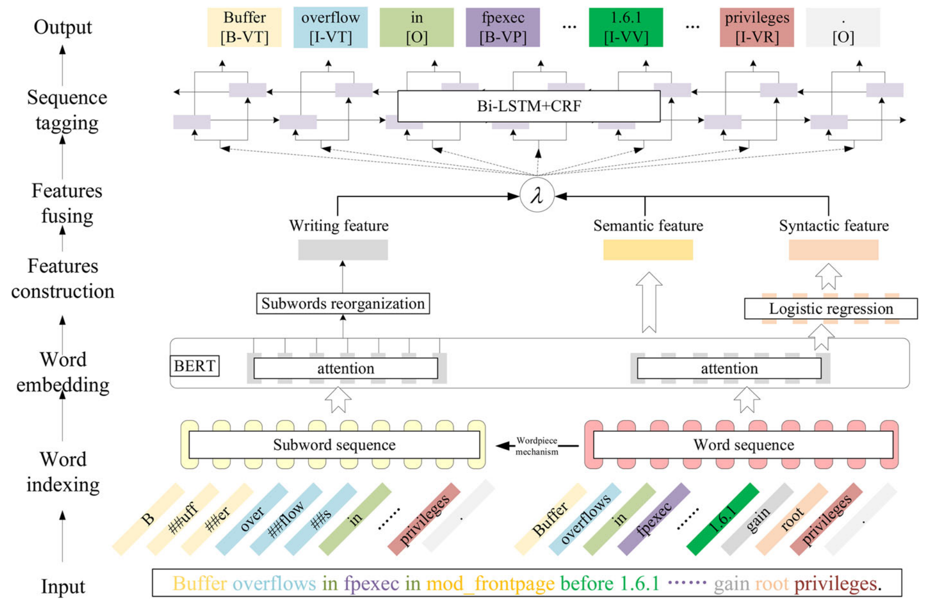


Fig. 3 Architecture of Vul-PAG

nal sequence with “[UNK]” and index it as $IND^{coa} = [ind_1^{[unk]}, ind_2^{[unk]}, \cdots, ind_n]$.

Word embedding In this module, the index sequence is transformed into a high-dimensional space, where each index corresponds to a numerical vector representing the word’s embedding. BERT translates IND^{med} into $E^{med} = [e_1^{med}, e_2^{med}, \cdots, e_m^{med}]$ and translates IND^{coa} into $E^{coa} = [e_1^{coa}, e_2^{coa}, \cdots, e_n^{coa}]$ with (1):

$$
\begin{aligned}
E^{med} &= f(\Theta_{pre-BERT}, IND^{med}) \\
E^{coa} &= f(\Theta_{pre-BERT}, IND^{coa})
\end{aligned} \tag{1}
$$

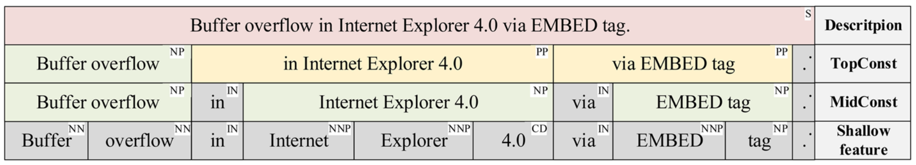
Here, $\Theta_{pre-BERT}$ represents the pre-trained BERT model. The dimension of E^{med} is $13 * m * 768$, where m is the sum of all subwords and non-OOV words. The dimension of E^{coa} is $13 * n * 768$ and n is the number of original words.

Generally speaking, the output of the language model primarily signifies the word’s semantic feature, which is also the only information used by other extraction methods. However, comprehensive extraction requires more information to assist locating aspects’ boundaries. On the one hand, based on the description templates in Section 3, some aspects exhibit syntactic structural features at the sentence level. For instance, “impact” functions as a complement of “attacker type”, “attack vector” behaves in an adverbial manner, “root cause” starts with a verb, etc. Capturing this feature can effectively enhance the determination of whether a word is an aspect boundary. On the other hand, some aspects are identifiable by conspicuous writing patterns within OOV words, e.g., OOV words containing subwords like “.php” and “.cpp” tend to be “component ”, consisting entirely of numbers and “.” and “-” are “version”, capturing such information can further refine our capability to assign OOV words to specific aspects. Hence, we aim to capture additional syntactic and writing features of word, and fuse them with semantic feature to further improve the model’s proficiency in locating the aspect’s boundary.

Syntactic feature The syntactic feature of word can be captured by the middle layer of BERT and verified by a TopConst task [22]. Using CVE-1999-0876 as an example (Fig. 4), the task initially abstracts the sentence as “NP-PP-PP-...” to form a TopConst sequence, and then selects BERT’s layer that makes the best classification of the sequence to represent the syntactic feature. However, the TopConst sequence slightly differs from the description templates, we further split the specific prepositional phrase “PP” to form MidConst sequence named “NP-IN(in)-NP-IN(via)-NP-.”. Based on this sequence, the optimal layer for BERT to express semantic feature is selected. It goes through steps of data selection, classifier training, and optimal layer selection.

(1) Data selection: Vulnerability descriptions with variable writing styles demonstrate flexible writing features, and we need to select descriptions that satisfy stable syntactic features as the baseline dataset, i.e., vulnerability descriptions that satisfy the template style. We abstract both vulnerability descriptions and templates into MidConst sequences (Fig. 5), employ regular expressions to match each description and template and assign labels accordingly. Descriptions that successfully match templates are compiled into a dataset, which is then divided into training and test sets.

(2) Classifier training: Using the constructed dataset, we train classifiers for MidConst sequences using each BERT layer. Syntactic feature is word-based to explore the relationship between words in the sentence structure, and coarse-grained BERT also takes word as a unit, we opt for it



Buffer overflow in Internet Explorer 4.0 via EMBED tag. S										Descritpion
Buffer overflow NP		in Internet Explorer 4.0 PP				via EMBED tag PP			. .	TopConst
Buffer overflow NP		in IN	Internet Explorer 4.0 NP			via IN	EMBED tag NP		. .	MidConst
Buffer NN	overflow NN	in IN	Internet NNP	Explorer NNP	4.0 CD	via IN	EMBED NNP	tag NP	. .	Shallow feature

Fig. 4 samples of constituency parsing at different levels

to explore word's syntactic feature. For training set, taking $D = [d_1, d_2, \cdots, d_n]$ with matrix E^{coa} as example, using the t-th layer's vector $E^{(coa,t)}$ and tag set Y to train a classifier $f^{(t)}_{REG}(E^{(coa,t)}, Y)$ with minimizing (2):

$$L = \frac{1}{m}\sum_{k=1}^{m} -y_k \log(y'^{(t)}_k) - (1 - y_k)\log(1 - y'^{(t)}_k) \tag{2}$$

where y_k is the k-th word's tag and m is the number of training samples, $y'^{(t)}_k$ refers to the k-th word's predicted tag with the t-th layer's vector.

(3) Optimal layer selecting: for test set, feeding the t-th layer's vector $e^{(t)}$ into the trained classifier $f^{(t)}_{REG}(.)$ to predict description's template tag $y^{(t)}$. With (3) to calculate this layer's total loss:

$$L^{(t)}_{pred} = \frac{1}{m}\sum_{k=1}^{m}(y^{(t)}_k - y'^{(t)}_k)^2 \tag{3}$$

the calculated losses range from m to n layer of BERT, and the softmax function is used to select the layer with the lowest loss as the optimal syntactic feature representation, expressed as $e^{(syn)}$.

Writing feature As mentioned earlier, the information indicating the aspect category of a word resides in multiple tokenized subwords, and it is more effective to unite them to indicate the word. Compared with the fine-grained approach like char embedding to splits word into individual character, and the coarse-grained approach ignores the internal structure of word directly, the medium-grained approach of the wordpiece mechanism, which can express the longest substring of textual features of a word with greedy search, is also opt for us to capture word's writing feature. However, the mechanism would lengthens the original sequence, making it inconsistent with the length of the original word sequence, which further severely affects the extraction performance. To preserve the writing information and eliminate the inconsistency issue, reorganizing the subwords is an effective method.

Analysis of the three vulnerability aspects with the most OOV words (versions, products, and components) reveals that the subwords used to indicate categories are highly repetitive and possess distinct characteristics. For example, subwords of numbers and special symbols (e.g., ".", "1", and "0") are most prevalent in "version", while ".", "/", "p", "##hp", and "c", which can form filenames, are frequent in "component". Assigning higher weights to these subwords during the reorganization process can more clearly represent the category of OOV words. Further examination of the subwords' position in OOV words reveals that such subwords tend to appear towards the latter part of OOV words. Therefore, we posit that the importance of indicative subwords is proportional to their indexing value within OOV words. For an OOV word $d^{(i)}$, a medium-grained BERT splits it into a subword list $d^{(i)} = [d^{(i)}_1, d^{(i)}_2, \cdots, d^{(i)}_p]$ and embeds it as $[e^{(med)}_{i,1}, e^{(med)}_{i,2}, \cdots, e^{(med)}_{i,p}]$, using (4) for reorganization:

$$e^{(mic)}_i = \frac{\sum_{j=1}^{p} \lambda_j \times e^{(med)}_{i,j}}{p}$$
$$\lambda_j = \frac{2 \times j}{p \times (p+1)} \tag{4}$$

Here, p represents the number of subwords, λ_j signifies the weight of the j-th subword, and $e^{(mic)}_i$ is the reorganized vector. $e^{(mic)}_i$ aggregates all subwords' information and is selected as a word's writing feature. Moreover, based on (4), the matrix of E^{med} transfers to $E^{(mic)} = [e^{(mic)}_1, e^{(mic)}_2, \cdots, e^{(mic)}_n]$ with dimension of $13 * n * 768$, matching the length of original tag sequence.

Semantic feature In most extraction tasks, the semantic feature is typically the sole focus. Our framework also incorporates it as an integral part. Due to medium-grained BERT's tendency to concentrate on the subword level, which can affect text comprehension and the expression of semantic information, especially for non-OOV words, we opt for coarse-grained BERT to capture word-level semantic information and select its last layer as the semantic feature,

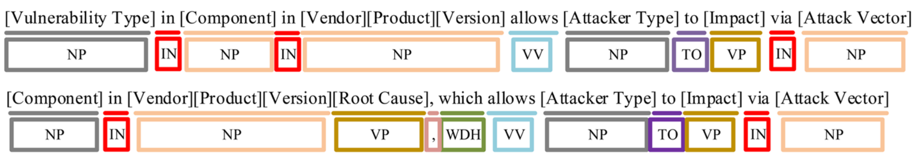

[Vulnerability Type] in [Component] in [Vendor][Product][Version] allows [Attacker Type] to [Impact] via [Attack Vector]
NP
IN
NP
IN
NP
VV
NP
TO
VP
IN
NP
[Component] in [Vendor][Product][Version][Root Cause], which allows [Attacker Type] to [Impact] via [Attack Vector]
NP
IN
NP
VP
,
WDH
VV
NP
TO
VP
IN
NP

Fig. 5 MidConst sequences of vulnerability description templates

denoted as $e_i^{(sem)}$, as shown in (5). This choice also follows the conclusion in [22].

$$e_i^{(sem)} = e_i^{(coa,-1)} \tag{5}$$

Features fusing The writing, semantic and syntactic features express words from the views of subword, word and sentence, and their contributions to the classification can differ. Therefore, we employ a linear combination to fuse the multi-view features, as depicted in (6):

$$f_i = \lambda_1 \times e_i^{(mic)} + \lambda_2 \times e_i^{(sem)} + (1 - \lambda_1 - \lambda_2) \times e_i^{(syn)} \tag{6}$$

Sequence tagging This section contains a Bi-LSTM network and a CRF model. The Bi-LSTM network takes the fused feature of a word as input for encoding and generates the word's hidden state, as illustrated in (7):

$$\begin{aligned} \vec{h}_i &= \vec{f}_{Bi-LSTM}(\theta_{Bi-LSTM}, f_i) \\ \overleftarrow{h}_i &= \overleftarrow{f}_{Bi-LSTM}(\theta_{Bi-LSTM}, f_i) \\ \overline{h} &= \vec{h}_i \bigoplus \overleftarrow{h}_i \end{aligned} \tag{7}$$

$\vec{h}_i$ and $\overleftarrow{h}_i$ is the forward and backword hidden state of the i-th word, the final result is their concatenation. $\theta_{Bi-LSTM}$ are parameters for LSTM layer, and $f_{Bi-LSTM}(.)$ is the mapping function for LSTM. The output of LSTM network is a probability matrix $H = \left[\overline{h}_1, \overline{h}_2, \cdots, \overline{h}_n\right]$ with dimension of $n * L$, where L donates the size of the tag set. Finally, the CRF model learns the logical relationships among tags from the perspective of the dataset to correct any incorrect logic within the sequence. Specifically, it calculates the transition probability between tags and the probability of the whole tagging sequence to obtain the global optimal sequence.

5 Dataset

This section introduces issues about the dataset, including dataset construction and dataset analysis.

5.1 Dataset construction

We collect over 16W descriptions from the NVD from 1999 to 2020. After filtering out invalid CVE entries that tagged by "**RESERVED**" and "**REJECTED**", there remains 146,302 entries.

Undersampling We employ undersampling to construct our dataset from the filtered CVE entries. We draw 100 samples each year to ensure an even distribution over time. Additionally, we maintain comprehensiveness in the dataset by selecting an equal proportion of samples from descriptions that adhere to template 1, template 2, and free-writing styles, which output a dataset with 2,200 samples.

Data annotation Data annotation is a crucial step in creating ground-truth datasets for our framework. Given the prevalence of proper nouns and the flexible format of text writing in vulnerability descriptions, there may be biases in annotation results when using existing automatic annotation models and tools (e.g., Doccano, tagtog). To mitigate this bias, we employed an odd number of researchers for this work and set specific rules:

(1) If vendor and product are co-occurring and the vendor is in front, both are annotated as "VN".

(2) If the product's version is limited to a specific platform/series/system, these additional conditions are also annotated as "VN".

(3) The affected product and version should be consistent with the value recorded in the corresponding CPE, the rest of the product-related components, interfaces, etc. are annotated as "VP".

(4) Vulnerability type, attack vector, and impact with "unspecified" are annotated as "O", such as "unspecified vulnerability", "unspecified vectors", and "unknown impacts".

(5) During annotation, we employ a voting mechanism to determine the final label when at least two researchers have inconsistent results for the same word.

For annotation methods, BIO mode [3] is used to annotate eight aspects, which mainly come from previous templates. The correspondence between aspects and tags is shown in Table 1.

5.2 Dataset analysis

The vulnerability descriptions in our dataset exhibit various formats, broadly categorized into three groups: template 1 and its variations, template 2 and its variations, and free writing. By analyzing the dataset, 685 entries satisfy template 1 and its variants, 156 follow template 2 and its variations, and 1,359 entries are in free writing format, which achieves the coverage of all writing formats of vulnerability descriptions. It is important to note that a description may contain all eight aspects or only a subset, leading to an inherent imbalance in aspect distribution. To gauge the degree of this imbalance, we conducted a count of each aspect across 2,200 samples, and the results are summarized in Table 2.

It is evident that aspects such as "vendor","version" and "impact" are highly prevalent in the dataset, reflecting their

Table 1 Correspondence between aspects and tagging labels

Aspect	Label	Aspect example	Label sequence
Vulnerability type	B-VT, I-VT	"Buffer overflow"	B-VT, I-VT
Product	B-VN, I-VN	"Cilem Hiber"	B-VN, I-VN
Version	B-VV, I-VV	"2.700 through 2.802"	B-VV, I-VV, I-VV
Root cause	B-VRC, I-VRC	"Does not log invalid logins"	B-VRC, I-VRC, I-VRC, I-VRC, I-VRC
Component	B-VP, I-VP	"db2www CGI interpreter"	B-VP, I-VP, I-VP
Impact	B-VR, I-VR	"Obtain sensitive information"	B-VR, I-VR, I-VR
Attacker type	B-VAT, I-VAT	"Remote attackers"	B-VAT, I-VAT
Attack vector	B-VAV, I-VAV	"A crafted web site"	B-VAV, I-VAV, I-VAV, I-VAV

common presence in vulnerability descriptions. Conversely, "root cause" is notably absent, likely due to the focus of vulnerability disclosures on visible results rather than an in-depth analysis of root causes. The absence of "vulnerability type" can be attributed to separate recordings in the Common Weakness Enumeration (CWE) system. This analysis provides insights into the aspect distribution within our dataset, which is essential for understanding the challenges associated with the imbalance of aspects in real-world vulnerability descriptions.

6 Experimental setups

Our experimental setups consider both baseline models and specific settings for hyper-parameters.

6.1 Baseline models

We base our work on existing models that treat vulnerability aspect extraction as a sequence labeling task. These models have been chosen as our baselines:

* Dong et al. [5] proposed VIEM to extract vulnerability aspects from multi-source reports and measure inconsistencies between aspects. It combines FastText and char embedding model to embed words and uses Bi-GRU for feature encoding and CRF for decoding.
* Binyamini et al. [1] proposed a framework to extract aspects for automated construction of vulnerability explbreak oitation rules. It uses the Word2Vec model to embed words and Bi-LSTM to generate tag sequence.
* Li et al. [27] designed architecture with a self-attention mechanism for recognizing cyber entities. It uses the Word2Vec and char embedding model for embedding, and leverages Bi-LSTM for word's contextual feature and self-attention mechanism for global information within a sentence.

6.2 Settings

Several crucial hyper-parameters must be set before training the model:

* Max Sequence Length: We use the wordpiece mechanism as a tokenizer to elongate the word sequence. After analyzing the length distribution of both word and token in a single description, we find that 99.86% of descriptions are shorter than 250 words, and 98.39% are shorter than 250 tokens with the wordpiece tokenizer (Fig. 6). Considering the additional "[CLS]" and "[SEP]" tags in the BERT model [40], we set the "max sequence length" to 256.
* Pre-trained Model: The extracted aspects include "product", which comes with a distinct writing feature, so we choose "bert-base-cased" as the pre-trained model. It consists of 12 layers with a hidden size of 768.
* Other Hyper-Parameters: We set the following hyper-parameters: the batch size is 32, the optimizer is Adam, the number of epochs is 100, and the learning rate is 0.01.

Table 2 Count distribution statistics of each aspect in the dataset with 2200 samples

Aspect	VN	VV	VT	VRC	VP	VAT	VAV	VR
Count	2196	1919	1065	645	1182	1716	1559	1979
Percentage	99.82%	87.23%	48.41%	29.32%	53.73%	78%	70.86%	89.95%

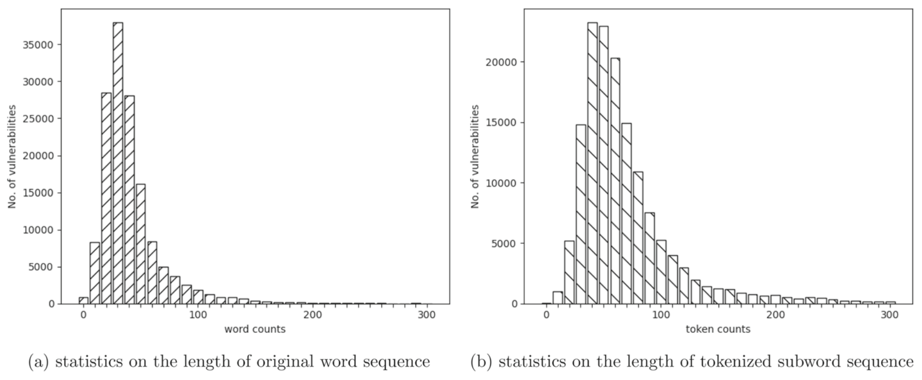


Fig. 6 The length distribution statistics of original word sequence and tokenized subword sequence of each description

(a) statistics on the length of original word sequence (b) statistics on the length of tokenized subword sequence

7 Experimental results

All experiments in this work are implemented on a Windows 10 operating system with an NVIDIA GeForce RTX 2080 Ti GPU. The deep learning framework is used in Pytorch.

7.1 Enhancements explanation

We introduce two enhancements to improve the performance of our model. These enhancements involve different factors and possible values:

(1) In embedding OOV word, char embedding, coarse-grained BERT, and the wordpiece mechanism are selected to compare the accuracy of embedding OOV word.

(2)The wordpiece mechanism lengthens the original sequence due to tokenization. We investigate the effect of concatenating subwords versus not concatenating them on the final extraction performance.

(3) We design a MidConst task to determine which layer of BERT can best capture a word's syntactic feature. Candidate layers considered range from layer 6 to layer 10.

(4) In the features fusion phase, we explore the linear combination of features and examine how hyper-parameter settings influence the final extraction performance.

7.1.1 Effectiveness of the wordpiece mechanism

OOV words are prevalen in vulnerability aspects. According to statistics, the percentages of OOV words in each aspect are summarized in Table 3.

Based on the statistics, aspects like "version", "product" and "component":(1) contain technical terms ("Mozilla", "Wireshark"), or (2) words do not follow conventional word formation methods ("0.9.1alpha", "Joomla!") show high percentage. Whereas, aspects like "impact" and "attacker type" that described in common language or composed of fixed words, the percentage significantly reduced.

To assess the effectiveness of different OOV word embedding methods, we select samples from aspects with high OOV word percentages and compare their semantic similarities. The samples of "component" include ["home/secretqtn.php", "objects.php","cosine.c","html_r.c"], and samples of "version" contain ["0.8.6e","7.3.0.29","2.4.1-test1","7.3-2"]. The results of this comparison are visualized in Fig. 7.

1. **Char embedding** Char embedding performs well in capturing the semantic relations of words with similar character type distribution, as the characters are densely mapped to a small number of ranges after indexing. For example, it effectively captures the similarity between "0.8.6e" and "7.3.0.29" (Fig. 7(a)) with high similarity of 0.62. However, it struggles with words with variable or completely different character type distributions, such as "7.3.0.29" and "html_r.c" with similarity of 0.

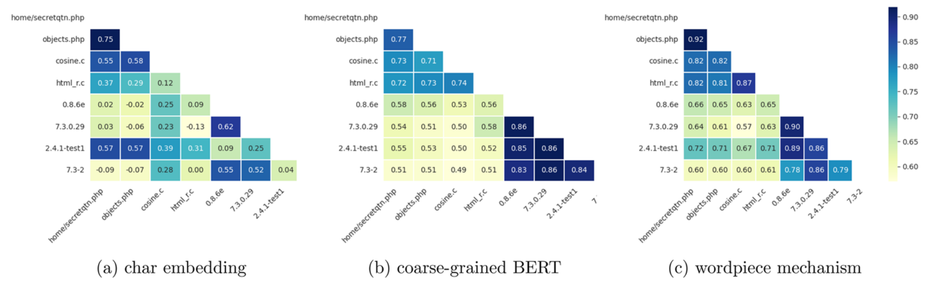
2. **Coarse-grained BERT** Coarse-grained BERT treats OOV words equally and expresses them through contextual information, and its extraction performance is closely related to the number of available samples. Among aspects with similar proportions of OOV words, the missing rate in "component" is significantly higher than that in "version" (Fig. 7(b)), which affects the learning ability of the model and leads to an average performance difference of nearly 12 percentage point.

3. **Wordpiece mechanism** The wordpiece mechanism greedy searches for the longest subwords in a word, forming a structural representation of the word. It is effective in embedding OOV words (Fig. 7(c)), as data imbalance does not

Table 3 Number and percentage distribution statistics of OOV words in each aspect

Aspect	VN	VV	VT	VRC	VP	VAT	VAV	VR
OOV words	3653	3938	1562	1279	1824	321	3877	2310
Orginal words	6934	7180	3827	6442	3695	4129	13557	13073
Percentage	52.68%	54.85%	40.82%	19.85%	49.36%	7.77%	28.60%	17.67%

Fig. 7 Semantic similarity of OOV words with different granularity methods



affect it. Words containing the same subword sequence show higher semantic similarity, such as "home/secretqtn.php" and "objects.php" both contain [".","p","##hp"], its similarity is as high as 0.92; "html_r.c" and "cosine.c" also contain [".","c"] with similarity of 0.87.

7.1.2 Necessity of subwords reorganization

According to statistics, the wordpiece mechanism expands the length of sentence to an average of 1.668 times. Whether the splited subwords are concatenated will impact the final result, [BWC] and [BWN] denote concatenated and not respectively. Accuracy, precision, recall, and F1-score were used to evaluate the overall extraction performance of both schemes, and F1-score was chosen to evaluate the detailed extraction performance, which are shown in Fig. 8.

Here are the key findings from the comparison. In Fig. 8(a), [BWC] shows a significant advantage over [BWN] in terms of overall extraction performance, with improvements of 0.40, 0.55, 0.62, and 0.59 in terms of accuracy, precision, recall, and F1-score respectively. This highlights the need for subwords reorganization. The wordpiece mechanism causes misalignment between the predicted tag sequence and the original tags by splitting words into subwords. Subwords reorganization helps solve this problem.

When examining the detailed extraction performance of the two schemes (Fig. 8(b)), [BWC] also outperforms [BWN] in all aspects. Take "VT" as an example, which located in the first half of the sentence and has a few OOV words in front of it, [BWN] perform well with F1-score of 0.77. However, the proportion of OOV words is relatively high, and do not reorganize subwords also affect the alignment of the predicted sequence with the original sequence. In contrast, its extraction performance is 0.9 after reorganization, and the performance is improved by **17%**. Taking "VAV" as another example, which is located in the last half of the sentence and has vast OOV words in front of it. [BWN] cause a serious asymmetry between the predicted sequence and the sequence, showing poor extraction performance (F1 -score is 0.03). The effect after reorganizing the subwords is significantly improved, and the F1-score in [BWC] is 0.81, an improvement of up to **260%**. In summary, [BWC] reorganizes multiple subwords into single words, eliminating the length inconsistency between predicted and original labels. This results in better extraction performance for all aspects, regardless of their position in the description.

To further validate the effectiveness of the split-reorganization mechanism compared to the traditional sequence tagging methods (BERT+CRF, BERT+BiLSTM+CRF) in a real scenario with a large number of OOV wordswe, we integrate the split-reorganization mechanism into the Vul-PAG's backbone framework(BERT+BiLSTM+CRF). The extraction performance comparisons are shown in Table 4.

The results show that Vul-PAG with writing outperforms BERT+CRF and BERT+BiLSTM+CRF. For BERT+CRF +CRF and BERT+BiLSTM, the framework of BiLSTM network is added to capture the contextual information of word neighborhoods, which is reliable for samples with structural

Fig. 8 Overall and detailed extraction performance for reorganized and non-reorganized subwords based on F1-score

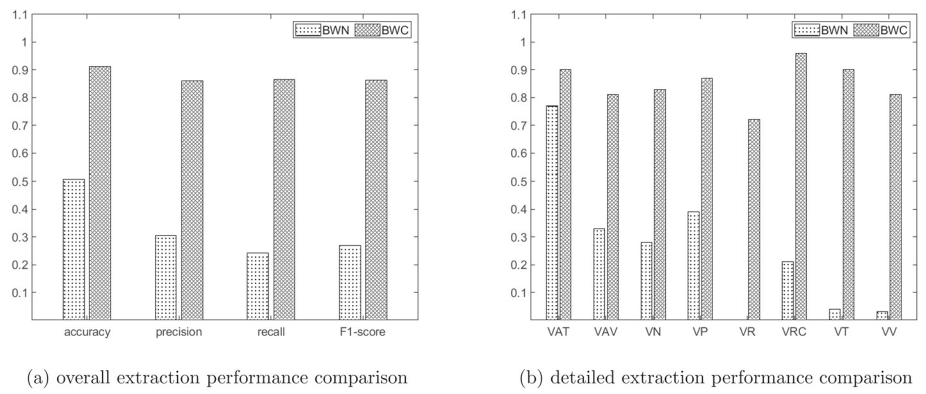


Table 4 Extraction performance comparisons of split-organization and other methods

Model	Precision	Recall	F1-score
BERT+CRF	80.08%	77.58%	78.81%
BERT+BiLSTM+CRF	83.88%	83.87%	83.88%
Vul-PAG with writing	86.04%	86.48%	86.26%

features, and thus the extraction performance is improved by **5.07**. However, while BERT+BiLSTM+CRF treats all OOV words uniformly and relies on global and contextual information for expression, Vul-PAG with writing utilizes a splitting and reorganization mechanism to extract the internal writing features of OOV words and enhance their expressiveness, which leads to a further improvement in the extraction performance (an improvement of **2.38**), especially for the aspect with OOV words.

7.1.3 Syntactic feature representation analysis

This section delve into the best layer of BERT for classifying the description's Minconst sequence. We abstract templates' Midconst sequences and construct regular expressions to match the real description's Midconst sequence, as shown in Table 5. Assigning label "1" if the description matches the first template, label "2" for the second one, and label "0" for mismatching.

After matching, there are 447 descriptions with label "1" and 89 descriptions with label "2". We select 100 descriptions with label "0" and "1", and all descriptions with label "2" as our dataset. Since the length of each sentence is variable, and all original words' information will summarize into a fixed token through a self-attention mechanism, we choose the "[CLS]" token to represent the sentence. A Logistic regression model is selected as the classifier, and the classification result is shown in Table 6:

Table 6 The Midconst sequence classification results of different layers in BERT

Layer of BERT	6	7	8	9	10
Accuracy	0.71	0.72	0.72	0.69	0.67
Macro avg	0.70	0.72	0.72	0.71	0.70

According to the result, the 7-th and 8-th layers get better performance, which is similar to the conclusion in [22]. Further analysis of the detailed performance of the two layers, as shown in Table 7.

The results indicate that the 8-th layer classifies sentences with labels "1" and "2" slightly better, improving by 0.04 and 0.02 respectively. Both labels correspond to vulnerability descriptions that satisfy the template, so we choose it to represent syntactic features. Further to verify the effectiveness of syntactic information, we compare the extraction performance of fusing semantic and syntactic features with semantic only. Equation (8) is used to combine the two features:

$$f'_i = \alpha * e_i^{(sem)} + (1 - \alpha) * e_i^{(syn)} \tag{8}$$

A grid search is performed to determine the optimal parameter α, and the results are shown in Fig. 9(a). The optimal performance is achieved when $\alpha = 0.7$, indicating that the semantic feature primarily drives the extraction process, with the syntactic feature playing a complementary role. This finding aligns with the fact that syntactic features are designed based on samples adhering to specific rules, making them better suited for capturing characteristics found in template-compliant samples. However, in real-world data, free-writing samples are more prevalent, and such data's flexible and variable nature diminishes the guidance provided by syntactic features on word labels compared to semantic features.

Table 5 Description template's Midconst sequence and corresponding regular expressions and labels

Midconst sequence	Regular expression	Label
$NP\_IN\_NP\_IN\_NP$ $\_VV\_NP\_TO\_VP\_IN\_PP$	$((NN\vert NNS\vert NNP\vert NNPS)(.*)?IN(.*)?(NN\vert NNS\vert$ $NNP\vert NNPS)(.*)?IN(.*)?(NN\vert NNS\vert NNP\vert NNPS)$ $(.*)?(VB\vert VBZ\vert VBP)(.*)?(NN\vert NNS)(.*)?TO\ VB(.*)?$ $(NN\vert NNS)(.*)?IN\ NN(.*)?)$	1
$NP\_IN\_NP\_VP\_,\_WDH$ $\_VV\_NP\_TO\_VP\_IN\_NP$	$(NN\vert NNS\vert NNP\vert NNPS)(.*)?IN(.*)?((NN\vert NNS\vert NNP\vert$ $NNPS)(.*)?(VB\vert VBZ\vert VBP)(.*)?(NN\vert NNS\vert NNP\vert$ $NNPS)(.*)?, WDTVB(.*)?(NN\vert NNS)(.*)?TOVB(.*)?$ $(NN\vert NNS)(.*)?INNN(.*)?)$	2
Other	_	0

Table 7 Detailed classification results of the 7-th and 8-th layer of BERT

Layer of BERT	Tag	Metric		
		Precision	Recall	F1-score
7-th layer	0	0.73	0.89	0.8
	1	0.62	0.56	0.59
	2	0.8	0.73	0.76
8-th layer	0	0.68	0.83	0.75
	1	0.65	0.62	0.63
	2	0.84	0.73	0.78

We further compared the detailed extraction performance of various aspects of models with and without syntactic features, as shown in Fig. 9(b). The results show that after adding syntactic information, the extraction performance of other aspects except “component” is improved. For “attack vector”, “product”, “version”, “root cause”, and “attacker type”, which exhibit distinct structural characteristics such as conditional clauses and specific starting verbs, the extraction performance is notably improved with the inclusion of syntactic features, with 0.03, 0.04, 0.03, 0.02, and 0.02 improvements, respectively. These characteristics, when captured, effectively enhance the extraction performance. On the other hand, “impact” and “vulnerability type”, which have fixed locations but lack clear structural features, show limited improvement with syntactic features. These aspects capture limited structural information at the sentence level, resulting in modest performance gains, both by only 0.01 respectively. For “component,” a flexible aspect composed of common noun phrases, capturing its syntactic features is challenging, resulting in minimal improvement in extraction performance.

7.1.4 Features fusion analysis

This section explores the fusion of three features to achieve the best extraction performance. We utilize (6) to combine features and select optimal hyperparameters, $\lambda_1 and \lambda_2$, to maximize extraction performance. F1-score is used as the metric, and the results are displayed in Fig. 10.

According to the result, the extraction performance shows an increasing trend as λ_1 increasing, which indicates the writing feature contributes significantly to the extraction with the condition of numerous OOV words. Longitudinally, the extraction performance shows an up-and-down trend as λ_2 increasing, suggesting that syntactic feature is more beneficial to extraction performance when it carries the main weight compared to semantic feature. The best extraction performance achieves at $\lambda_1 = 0.9$, $\lambda_2 = 0.05$.

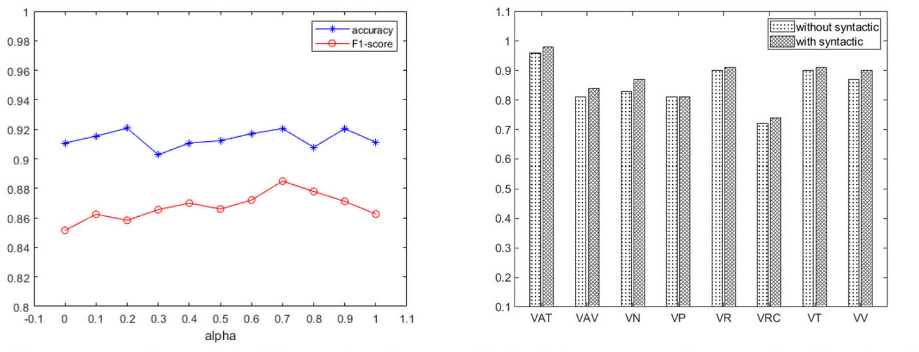
7.1.5 Stability and generalizability validation

In this section, we verify the effect of undersampling on the stability of the model and the generalization of the model to samples with different writing styles.

To verify the possible effect of undersampling on the comprehensiveness of the results, we validate both the internal data from the dataset and external data. On the one hand, we validate the stability of our model using 5-fold cross-validation, whose F1-score per fold is 0.8871, 0.8892, 0.8873, 0.8876, 0.8882, respectively; at the same time, we also re-selected 10 data from each year for the validation of the model’s performance under real environmental data, whose extracted performances are: precision is 0.8797, recall is 0.8734 and F1-score is 0.8765. From the cross-validation results within the dataset, the model’s extraction performance for each fold is within a slight fluctuation range, showing good stability; from the extraction results on data in the wild, its extraction performance is slightly affected, but still shows good generalization ability. Hence, the effect of undersampling on the model’s stability is slight.

To validate the generalizability of the model to samples with different writing styles, we selected 50 samples each from those satisfying Template 1, Template 2, and free-writing, respectively, as the validation sets. We evaluate them based on precision, recall, and F1-score.

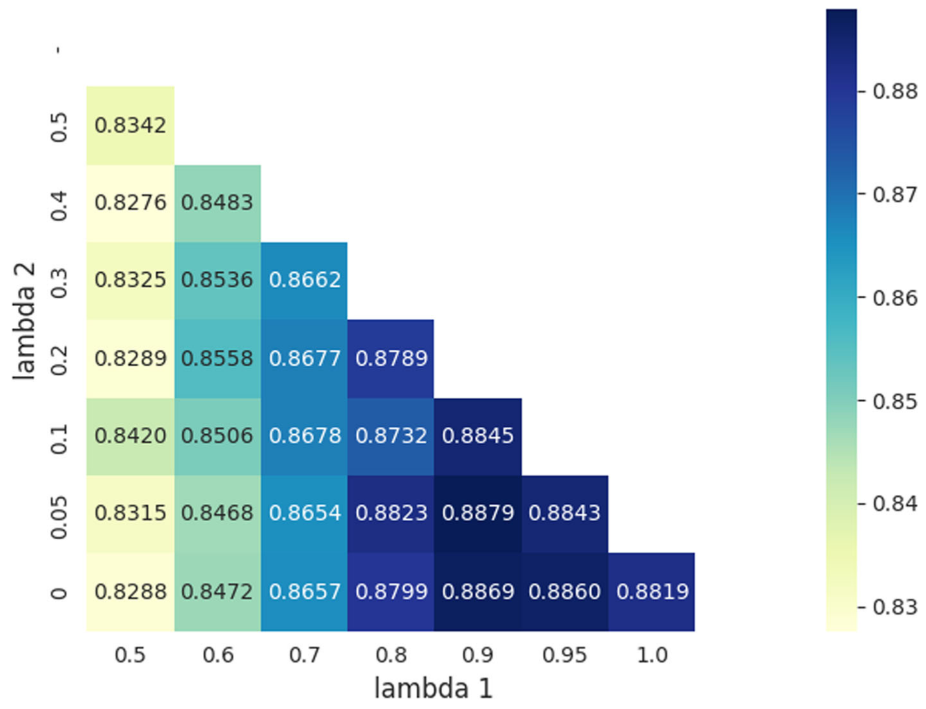
Fig. 9 Extraction performance with different values of α and detailed extraction performance of models with and without syntactic feature



(a) extraction performance for different values of α

(b) detailed extraction performance of models with and without syntactic feature

Fig. 10 Extraction performance with different values of hyper-parameters λ_1, λ_2



The experimental results are shown in Table 8. As observed, Vul-PAG performs best in extracting samples of Template 1 while performing less effectively on free-writing samples. The model's enhancement in extraction ability via the split-reorganization mechanism is slightly weaker than the impact of syntactic features when OOV words are generally distributed across the samples. Additionally, free-writing samples lack fixed sentence-level structural features, while the model's syntactic features are constructed based on templated samples, making it perform better in extracting samples with fixed structures. The relatively better performance of Template 1 over Template 2 can be attributed to differences in template complexity and training samples. As can be seen in Section 3, Template 2 tends to have more ambiguous aspect boundaries and complex sentence structures, while Template 1 benefits from a larger training dataset, which helps it capture the characteristics of various information types in the samples more effectively.

Table 8 Extraction performance comparisons of Vul-PAG on different writing style samples

Sample type	Precision	Recall	F1-score
Template 1	92.75%	91.67%	92.21%
Template 2	91.53%	91.21%	91.37%
Free-writing	88.01%	87.84%	87.92%

7.2 Ablation study

We conducted an ablation study to further validate Vul-PAG's superiority in comprehensive vulnerability extraction and identify which part of Vul-PAG has the greatest impact on improving the model's boundary localization ability. As previously mentioned, Vul-PAG is built upon BERT+BiLSTM+CRF, referred to as Vul-PAG-ORG. We then introduced two enhancements: one with the split-reorganization mechanism-based writing feature, and the other with the syntactic feature. Finally, we fused both enhancements to form the complete Vul-PAG. The three fusion cases are evaluated alongside Vul-PAG-ORG for extraction performance using precision, recall, and F1-score metrics.

The experimental results are summarized in Table 8. Compared with Vul-PAG with syntactic which improves 1.27 in Vul-PAG-ORG, Vul-PAG with writing improves 2.38, exhibiting a more significant improvement. On the one hand, the split-recombination mechanism effectively preserves written information, categorizing OOV words and providing directional cues for label prediction. On the other hand, the dataset contains a relatively small proportion of samples that conform to fixed syntactic features. Additionally, the exploration results for optimal syntactic features are relatively better, and the model's ability to capture syntactic features when handling many free-writing samples is somewhat limited. However, it demonstrates a better per-

Table 9 Overall extraction performance comparisons of Vul-PAG with different features

Fusion mode	Precision	Recall	F1-score
Vul-PAG-ORG	83.88%	83.87%	83.88%
Vul-PAG with writing	86.03%	86.48%	86.26%
Vul-PAG with syntactic	85.24%	85.06%	85.15%
Vul-PAG	88.94%	88.65%	88.79%

formance on samples and aspects with distinct syntactic features, aligning with our earlier conclusion. In summary, these two enhancements extract features from the internal level of words and the sentence structure level, and their integration equips Vul-PAG with the advantage of information boundary localization ability in comprehensive extraction, and the most significant improvement in its extraction performance, which reaches 4.91. The results are shown in Table 9.

7.3 Comparison with other works

In this section, we compare our Vul-PAG method with baseline models, each of which takes vulnerability descriptions as input and aims to extract eight predefined aspects (as defined in Section 6.1). The results for the baseline models are obtained by replicating the original papers.

The results presented in Table 10 show that Vul-PAG achieves the best extraction performance, while Binyamini's method performs the worst. An in-depth analysis reveals several key factors:

OOV word handling Vul-PAG leverage the split-reorganization mechanism, a more accurate approach for embedding OOV words instead of character embeddings. This mechanism plays a significant role in accurately embedding and predicting the labels of OOV words.

Contextual information Vul-PAG utilizes the context-dependent language model BERT, enabling it to capture richer semantic information compared to Word2vec and FastText, which are context-independent. Context-independent models require pre-training on a corpus, limiting their scalability.

Multi-view features Vul-PAG fuses multi-view features as input in the aspect extraction phase, whereas other methods often consider only monotonous features. This broader feature set allows Vul-PAG to learn more information and improve aspect tag prediction.

Table 10 Overall extraction performance comparisons of Vul-PAG and baseline models

Method	Precision	Recall	F1-score
(Li et al. [27])	84.85%	84.35%	84.60%
(Binyamini et al. [1])	70.05%	82.04%	75.57%
(Dong et al. [5])	85.47%	85.97%	85.72%
Vul-PAG	88.94%	88.65%	88.79%

Regarding Binyamini's method [1], it relies on the Word2vec model as a language model and lacks additional means to handle OOV words. This approach can result in high-dimensional sparse vectors during pre-training, leading to feature encoding and decoding errors for tag prediction.

Figure 11 illustrates the extraction performance comparisons for each aspect of both baseline models and Vul-PAG to provide a more detailed perspective.

The extraction performance trends across all methods follow a similar pattern. For aspects such as "product" and "version" that contain a large number of OOV words but no internal indicative characters, existing models also use additional means (e.g., char embedding) to process OOV words, and the improvement in extraction performance is limited. Compared with the latest approach, Vul-PAG only improves by 0.01. Meanwhile, the split-reorganization mechanism is advantageous for the "component" aspect, which has a high proportion of OOV words and distinctive writing features, and Vul-PAG improves the extraction performance by 0.09. Aspects with clear syntactic feature, such as "impact", "attack vector" and "root cause", benefit greatly from the addition of syntactic information, improving their performance by 0.05, 0.06, and 0.11, respectively. In addition, the fusion of additional features enhances the extraction performance of other OOV words, especially when the writing features are less obvious (performance of "attacker type" improves by 0.02) or the position of the word in the sentence is highly fixed (performance of "vulnerability type" performance improves by 0.01). In conclusion, the superior performance of Vul-PAG can be attributed to its effective processing of OOV words, the utilization of contextual information, and the combination of multi-view features.

7.4 Statistical test

To analyze how different features impact extraction, we evaluate the extraction performance across all mentioned schemes, acrossing overall performance and detail performance of eight specific aspects, illustrated by the 9 raindrops in each scheme in Fig. 12.

The results unveil several phenomena: (1) in vulnerability reports with a wide distribution of OOV words, reorganizing splitted subwords is essential (compare columns 8 and 9); (2) the integration of writing features of OOV words, captured through the split-reorganization mechanism(columns 8), improves extraction performance compared to coarse-grained OOV expressions (columns 5); (3) the inclusion of syntactic features (column 7) enhances extraction performance compared to (column 5); (4) comparing syntactic feature with writing feature (columns 7 and 8), the improve-

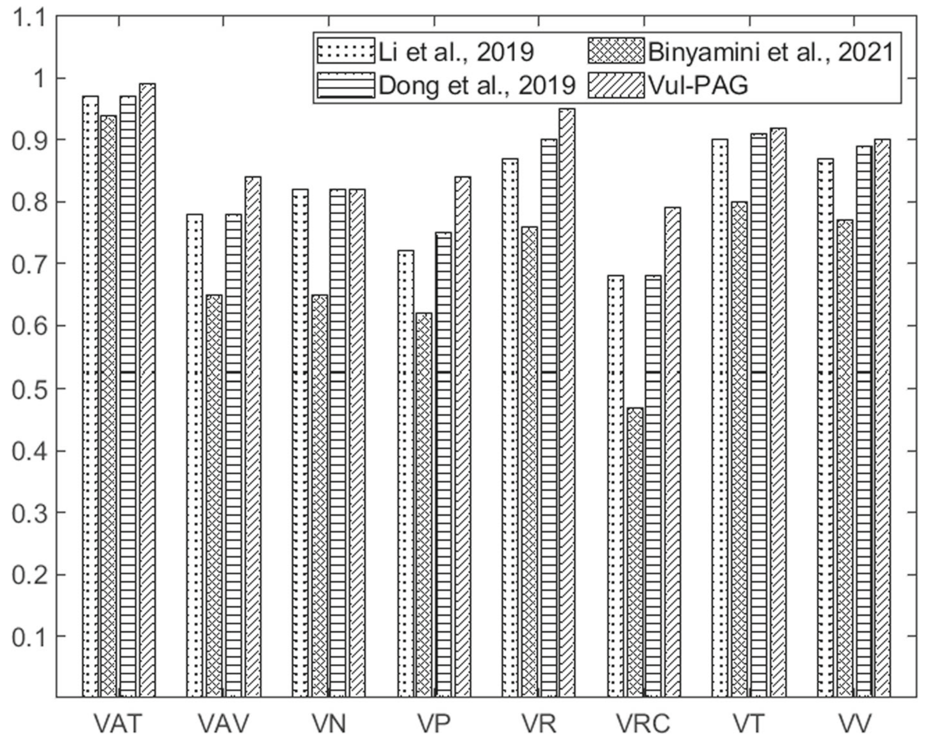


Fig. 11 Setailed extraction performance comparisons on each aspect of baseline models and Vul-PAG

ment of writing features is broad-based due to the wide distribution of OOV words in nearly all reports, while syntactic features are focused on specific aspects as syntactic features are not explicitly reflected in certain reports that not conform to the vulnerability template; (5) Syntactic features and writing features complement each other, and their fusion can comprehensively enhance extraction performance (e.g., columns 6, 7, and 8).

Comparing Vul-PAG with existing schemes(column 1-5), it is found that extraction schemes with a complete structure tend to have better extraction performance. Schemes that lack context-dependent language models (column 2) and schemes

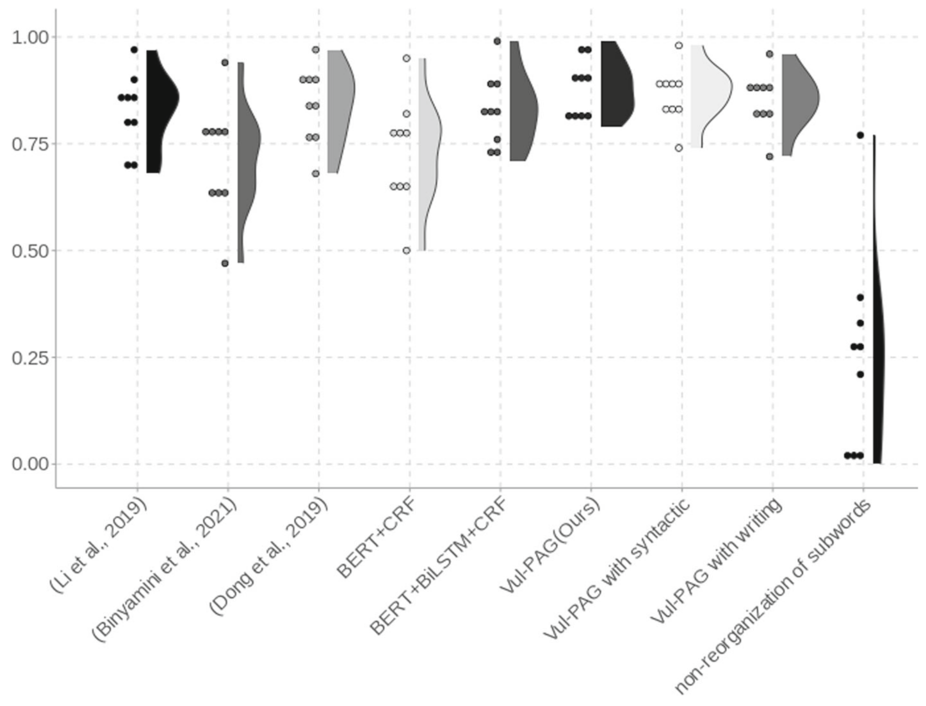


Fig. 12 Statistical tests of the performance of each extraction scheme

CVE-2007-2438	[⋯, allows, dangerous, functions, ⋯, might, allow, user-assisted, attackers, to, execute, ⋯]
orginal	[⋯,O,B-VRC,I-VRC, ⋯ ,O,O,B-VAT,I-VAT,O,B-VR, ⋯]
Plain approach	[⋯,O,O,O, ⋯ ,O,O,B-VAT,I-VAT,O,B-VR, ⋯]
Vul-PAG	[⋯,O,B-VRC,I-VRC, ⋯ ,O,O,B-VAT,I-VAT,O,B-VR, ⋯]

Fig. 13 Tag sequence comparison of "root cause"

that lack the ability to capture contextual information (column 4) perform poorly due to the former's limitation in language modeling, hindering its ability to cope with numerous OOV words, and the latter's inability to learn contextual information about aspect boundaries, hindering its use in aiding boundary localization. The other schemes improve the extraction performance by adding various other ways, including the supplementation of OOV word processing with fine-grained methods (column 3), and incorporating an attention mechanism for capturing global information (column 1), which fail to fully exploit the comprehensive features of words (e.g., writing, syntax), resulting in a performance gap with Vul-PAG.

7.5 Case study

To validate the effectiveness of our approach, we perform case studies. We take aspects with a distinct syntactic feature, and whose extraction performance is improved with Vul-PAG as examples.

(1) Root Cause: we take the description of "CVE-2007-2438" as case, as shown in Fig. 13.

According to the sample, there are two semantically similar verbs "allow(s)". The object of the latter is "user-assisted attackers", which satisfy the form of "allow [attacker type] to [impact]" in templates, and shows more accurate tag prediction. The object of the former is "dangerous functions", which is semantically unrelated to the express aspect. However, it can be abstracted to "VB+NP" with syntactic information added, which is similar to the syntactic feature of "root cause" (i.e. "starting with a verb"). This result suggests that Vul-PAG can accurately label aspects with an inner syntactic feature.

(2) Attack vector: we take the description of "CVE-2018-16849" as case, as shown in Fig. 14.

Generally speaking, "attack vector" comes after repositions like "via" and plays the role of a modal gerund. In this example, it's hard to predict the tag sequence for the phrase "manipulating the ..." alone with semantic information. However, the phrase is used as a modal gerund of the preposition "By" with the addition of syntactic information, which is consistent with the syntactic feature of "attack vector". This result suggests that Vul-PAG ignores semantic information to some extent, and performs accurate tag prediction based on the syntactic feature.

The above cases underscore Vul-PAG's sophistication compared to general methods in vulnerability extraction, especially after capturing text syntactic features. However, there is still room for improvement. Vul-PAG performs slightly weaker on free-writing samples compared to template-specific samples, and addressing this gap through more effective feature construction is an avenue for future work. Additionally,Vul-PAG improves extraction performance on aspects with distinct syntactic features but fewer training samples, but still falls short of aspects with sufficient samples, and exploring more efficient feature construction to weaken this gap is another aspect of subsequent work. Finally, another direction for future research is exploring vulnerability aspect extraction methods with small or zero samples, considering the labeling cost of training samples and scalability limitations.

8 Discussion

In this section, we discuss the implications and limitations of our research:

CVE-2018-16849	[A, flaw, was, ⋯. By, manipulating, the, SSH, ⋯, disclose, the, presence, of, ⋯]
orginal	[O,O,O,⋯,O,B-VAV,I-VAV,I-VAV,I-VAV,⋯,O,I-VR,I-VR,I-VR,I-VR⋯]
Plain approach	[O,O,O,⋯,O,O,O,O,O,⋯,O,I-VR,I-VR,I-VR,I-VR⋯]
Vul-PAG	[O,O,O,⋯,O,B-VAV,I-VAV,I-VAV,I-VAV,⋯,O,I-VR,I-VR,I-VR,I-VR⋯]

Fig. 14 Tag sequence comparison of "attack vector"

Implications

* Handling OOV Words: Some vulnerability aspects contain numerous OOV words with distinct writing features. Our split-reorganization mechanism effectively tokenize words into subwords while preserving their indication information. By reorganizing all subwords with indicative information, we improve the accuracy of embedding OOV words. This has significant implications for enhancing the handling of OOV words in the area of vulnerability aspect extraction.
* Boundary Localization: Comprehensive aspect extraction often results in limited words available for assisting in boundary localization. Our work addresses this issue by enhancing the model's boundary localization capability through the learning and fusion of multi-view features of words.

Limitations

* Data Source and Text Structure: Our approach is to improve aspect boundary localization by taking advantage of template-specific samples and fusing multi-view features when aspects are densely distributed in text. In future work, we will further explore the problem of aspect boundary localization in the case of aspect discrete distributed and variable text structure.
* Supervised Approach and Labeling Costs: Our supervised approach relies on manual labeling of data, which can be labor-intensive and prone to inconsistent understanding during labeling. To mitigate this, future research could explore methods for mining intrinsic structural features of text to develop unsupervised aspect extraction techniques.

Overall, our research presents promising advances in vulnerability aspect extraction, but further work is needed to address these limitations and broaden the applicability of the proposed methods.

9 Conclusion

Addressing the problems of customized tasks leading to frequent duplicated extraction contents, and existing extraction models limited by scalability and high annotation cost, this paper proposes a framework for comprehensive extracting vulnerability aspects. To solve the problem of diminishing the available localization information of aspect boundaries caused by comprehensive extraction, the framework combines wordpiece mechanism, regular expressions, and machine learning for extracting multi-perspective information from words to strengthen the expressive capacity of words. It encompasses writing, syntactic, and semantic information, and then enhance the model's capability to locate aspect boundaries. We found that in a textual environment in environments with widespread distribution of OOV words and obvious syntactic structures, capturing the writing feature within OOV words and integrating sentence-level syntactic feature effectively improve the framework's ability to locate vulnerability aspect boundaries and achieve optimal extraction performance. In this context, the advantage of writing feature surpasses that of syntactic feature. This is because OOV words are found throughout various aspects in all reports, and thus it can contribute to all aspects. In contrast, syntactic feature is learned based on a dataset characterized by distinct syntactic features and are more effective for aspects with distinct syntactic features, fall short in benefiting all aspects, as demonstrated by the weighting applied when merging the two feature. However, the framework's dependency on specific vulnerability description templates and its reliance on manually labeled data are still challenges that require attention in the framework. Therefore, in our future work, we would like to explore unsupervised extraction methods to eliminate dependence on human labor. Meanwhile, we also hope to investigate more efficient methods for syntactic feature extraction in texts with ambiguous syntactic structures.

Acknowledgements This research work was partly supported by the Young Scientists Fund of the National Natural Science Foundation of China under Grant (No. 62302322), the Fundamental Research Funds for the Central Universities (No. SCU2023D008), the Fundamental Research Funds for the Central Universities (No. 2022SCU12116), Sichuan University Postdoctoral Interdisciplinary Innovation Startup Fund Project (No. 10822041A2076), the Science and Engineering Connotation Development Project of Sichuan University (No. 2020SCUNG129) and the Key Laboratory of Data Protection and Intelligent Management (Sichuan University), Ministry of Education.

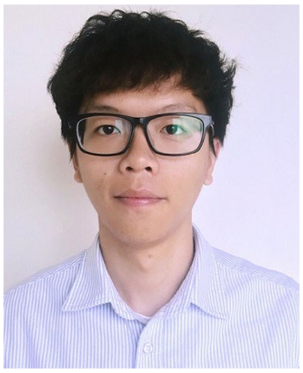
References

1. Binyamini H, Bitton R, Inokuchi M, Yagyu T, Elovici Y, Shabtai A (2021) A framework for modeling cyber attack techniques from security vulnerability descriptions. Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining
2. Cheng Y, Yang S, Lang Z, Shi Z, Sun L (2023) VERI: a large-scale open-source components vulnerability detection in iot firmware. Comput Secur 126:103068
3. Cnblogs (2022) Several major sequence annotation methods? https://www.cnblogs.com/zjuhaohaoxuexi/p/15506307.html

4. Costa JC, Roxo T, Sequeiros JBF, Proenca H, Inacio PRM (2022) Predicting cvss metric via description interpretation. IEEE Access 10:59125–59134
5. Dong Y, Guo W, Chen Y, Xing X, Zhang Y, Wang G (2019) Towards the detection of inconsistencies in public security vulnerability reports. In USENIX Security symposium
6. Dong Y, Tang Y, Cheng X, Yang Y (2023) Dekedver: a deep learning-based multi-type software vulnerability classification framework using vulnerability description and source code. Inf Softw Technol 163:107290
7. Du Y, Huang C, Liang G, Fu Z, Li D, Ding Y (2022) Expseeker: extract public exploit code information from social media. Appl Intell 53:15772–15786
8. Evans MJ (2020) key details phrasing. http://cveproject.github.io/docs/content/key-details-phrasing.pdf
9. Fang Z, Cao Y, Li T, Jia R, Fang F, Shang Y, Lu Y (2021) Tebner: domain specific named entity recognition with type expanded boundary-aware network. In EMNLP
10. Feng X, Liao X, Wang X, Wang H, Li Q, Yang K-T, Zhu H, Sun L (2019) Understanding and securing device vulnerabilities through automated bug report analysis. In USENIX Security symposium
11. Gao P, Liu X, Choi E, Soman B, Mishra C, Farris K, Song DX (2021) A system for automated open-source threat intelligence gathering and management. Proceedings of the 2021 International conference on management of data
12. Gao P, Shao F, Liu X, Xiao X, Qin Z, Xu F, Mittal P, Kulkarni SR, Song DX (2021) Enabling efficient cyber threat hunting with cyber threat intelligence. 2021 IEEE 37th International conference on data engineering (ICDE) pp 193–204
13. Gao Y, Li X, Peng H, Fang BX, Yu PS (2022) Hincti: a cyber threat intelligence modeling and identification system based on heterogeneous information network. IEEE Trans Knowl Data Eng 34:708–722
14. Garavand A, Behmanesh A, Aslani N, Sadeghsalehi H, Ghaderzadeh M (2023) Towards diagnostic aided systems in coronary artery disease detection: a comprehensive multiview survey of the state of the art. International Journal of Intelligent Systems
15. Ghazo ATA, Ibrahim M, Ren H, Kumar R (2020) A2G2V: automatic attack graph generation and visualization and its applications to computer and SCADA networks. IEEE Trans Syst Man Cybern Syst 50:3488–3498
16. Gheisari M, Ebrahimzadeh F, Rahimi M, Moazzamigodarzi M, Liu Y, Pramanik PKD, Heravi MA, Mehbodniya A, Ghaderzadeh M, Feylizadeh MR, Kosari S (2023) Deep learning: applications, architectures, models, tools, and frameworks: a comprehensive survey. CAAI Transactions on intelligence technology
17. Guo H, Xing Z, Chen S, Li X, Bai Y, Zhang H (2021) Key aspects augmentation of vulnerability description based on multiple security databases. 2021 IEEE 45th Annual computers, software, and applications conference (COMPSAC), pp 1020–1025
18. Hosseini A, Eshraghi MA, Taami T, Sadeghsalehi H, Hoseinzadeh Z, Ghaderzadeh M, Rafiee M (2023) A mobile application based on efficient lightweight cnn model for classification of b-all cancer from non-cancerous cells: a design and implementation study. Informat Med Unlocked 39
19. Husari G, Al-Shaer E, Ahmed M, Chu B, Niu X (2017) Ttpdrill: automatic and accurate extraction of threat actions from unstructured text of cti sources. Proceedings of the 33rd annual computer security applications conference
20. Husari G, Al-Shaer E, Chu B, Rahman RF (2019) Learning apt chains from cyber threat intelligence. Proceedings of the 6th annual symposium on hot topics in the science of security
21. IBM (2022) Ibm x-force exchange. https://exchange.xforce.ibmcloud.com/
22. Jawahar G, Sagot B, Seddah D (2019) What does bert learn about the structure of language? In ACL
23. Jo H, Kim J, Porras PA, Yegneswaran V, Shin S (2021) Gapfinder: finding inconsistency of security information from unstructured text. IEEE Trans Inf For Secur 16:86–99
24. Kim D, Kim HK (2019) Automated dataset generation system for collaborative research of cyber threat intelligence analysis. Secur Commun Netw 6268476(1–6268476):10
25. Li J, Sun A, Han J, Li C (2018) A survey on deep learning for named entity recognition. IEEE Trans Knowl Data Eng 34:50–70
26. Li R-Y, Tan S, Wu C, Cao X, He H, Wang W (2020) Ifvd: Design of intelligent fusion framework for vulnerability data based on text measures. 2020 29th International conference on computer communications and networks (ICCCN), pp 1–6
27. Li T, Guo Y, Ju A (2019) A self-attention-based approach for named entity recognition in cybersecurity. 2019 15th International conference on computational intelligence and security (CIS), pp 147–150
28. Li Y, Cheng J, Huang C, Chen Z, Niu W (2021) Nedetector: automatically extracting cybersecurity neologisms from hacker forums. J Inf Secur Appl 58:102784
29. Liao X, Yuan K, Wang X, Li Z, Xing L, Beyah RA (2016) Acing the ioc game: toward automatic discovery and analysis of open-source cyber threat intelligence. Proceedings of the 2016 ACM SIGSAC conference on computer and communications security
30. MITRE (2022) How are the cve entry descriptions created or compiled? https://cve.mitre.org/about/faqs.html#cve_entry_descriptions_created
31. News B (2010) Stuxnet worm 'targeted high-value iranian assets'. https://www.bbc.com/news/technology-11388018
32. Pan Q, Dong H, Wang Y, Cai Z, Zhang L (2019) Recommendation of crowdsourcing tasks based on word2vec semantic tags. Wirel Commun Mob Comput 2121850(1–2121850):10
33. Sharma R, Sibal R, Sabharwal S (2021) Software vulnerability prioritization using vulnerability description. International Journal of System Assurance Engineering and Management 12:58–64
34. Tang W, Hui B, Tian L, Luo G, He Z, Cai Z (2021) Learning disentangled user representation with multi-view information fusion on social networks. Inf Fus 74:77–86
35. Tang W, Tian L, Zheng X, Yan K (2022) Analyzing topics in social media for improving digital twinning based product development. Digital Communications and Networks
36. Times TNY (2017) A cyberattack the world isn't ready for. https://www.nytimes.com/2017/06/22/technology/ransomware-attack-nsa-cyberweapons.html
37. Wang H, Qin K, Lu G, Yin J, Zakari RY, Owusu JW (2021) Document-level relation extraction using evidence reasoning on rst-graph. Knowl Based Syst 228:107274
38. Wei R, Cai L, Yu A, Meng D (2021) Deephunter: a graph neural network based approach for robust cyber threat hunting. In SecureComm
39. Yang L, Chen X, Luo Y, Lan X, Chen L (2021) Purext: Automated extraction of the purpose-aware rule from the natural language privacy policy in iot. Secur Commun Netw 5552501(1–5552501):11
40. Yin J, Tang M, Cao J, Wang H (2020) Apply transfer learning to cybersecurity: predicting exploitability of vulnerabilities by description. Knowl Based Syst 210:106529
41. Yitagesu S, Xing Z, Zhang X, Feng Z, Li X, Han L (2021) Unsupervised labeling and extraction of phrase-based concepts in vulnerability descriptions. 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp 943–954

42. Yitagesu S, Xing Z, Zhang X, Feng Z, Li X, Han L (2023) Extraction of phrase-based concepts in vulnerability descriptions through unsupervised labeling. ACM Trans Softw Eng Methodol 32
43. You Y, Jiang J, Jiang Z, Yang P, Liu B, Feng H, Wang X, Li N (2022) Tim: threat context-enhanced ttp intelligence mining on unstructured threat data. Cybersecurity 5:1–17
44. Zhou Z, Bo L, Wu X, Sun X, Zhang T, Li B, Zhang J, Cao S (2022) SPVF: security property assisted vulnerability fixing via attention-based models. Empir Softw Eng 27:171
45. Zhu Z, Dumitras T (2016) Featuresmith: automatically engineering features for malware detection by mining the security literature. Proceedings of the 2016 ACM SIGSAC conference on computer and communications security
46. Zhu Z, Dumitras T (2018) Chainsmith: automatically learning the semantics of malicious campaigns by mining threat intelligence reports. 2018 IEEE European symposium on security and privacy (EuroS&P), pp 458–472

Qindong Li is currently pursuing the Ph.D. degree at the School of Cyber Science and Engineering, Sichuan University, Sichuan, China. His research interests include big data analysis and threat intelligence.



Wenyi Tang received the Ph.D. degree from the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China, in 2021. He is currently an Assistant Professor in the School of Cyber Science and Engineering at Sichuan University. His research interests are both theoretical and experimental, including Weakly Supervised Learning, Data Privacy and Security.

Xingshu Chen is full professor at the School of Cyber Science and Engineering, Chengdu, China and she is also with the Cyber Science Research Institute, Sichuan University and with the Key Laboratory of Data Protection and Intelligent Management (Sichuan University), Ministry of Education. She received the Master's degree at Sichuan University in 1999, the Doctor's degree at the same University in 2004. Her main research focus is related to Cloud Computing Security, Data Security, Network Threat Detection, Intelligence Analysis and AI Security.

Song Feng received the Master's degree at the School of Cyber Science and Engineering, Sichuan University, Sichuan, China, in 2023. His research interests include threat intelligence and vulnerability management.

Lizhi Wang received the Master's degree at the School of Cyber Science and Engineering, Sichuan University, Sichuan, China, in 2022. His research interests include threat intelligence and social network.