The Journal of Systems & Software 204 (2023) 111790

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Automatic software vulnerability assessment by extracting vulnerability elements☆

Xiaobing Sun [a,b], Zhenlei Ye [a], Lili Bo [a,b,c,*], Xiaoxue Wu [a,b], Ying Wei [a], Tao Zhang [d], Bin Li [a,b]

[a] School of Information Engineering, Yangzhou University, Yangzhou, China
[b] Jiangsu Province Engineering Research Center of Knowledge Management and Intelligent Service, Yangzhou University, Yangzhou, China
[c] Key Laboratory of Safety–Critical Software Ministry of Industry and Information Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
[d] School of Computer Science and Engineering, Macau University of Science and Technology (MUST), Macao Special Administrative Region of China

ARTICLE INFO

Article history:
Received 1 October 2022
Received in revised form 8 May 2023
Accepted 15 June 2023
Available online 20 June 2023

Keywords:
Vulnerability assessment
Deep learning
Multi-class classification
Mining software repositories

ABSTRACT

Software vulnerabilities take threats to software security. When faced with multiple software vulnerabilities, the most urgent ones need to be fixed first. Therefore, it is critical to assess the severity of vulnerabilities in advance. However, increasing number of vulnerability descriptions do not use templates, which reduces the performance of the existing software vulnerability assessment approaches. In this paper, we propose an automated vulnerability assessment approach that using vulnerability elements for predicting the severity of six vulnerability metrics (i.e., *Access Vector*, *Access Complexity*, *Authentication*, *Confidentiality Impact*, *Integrity Impact* and *Availability Impact*). First, we use BERT-MRC to extract vulnerability elements from vulnerability descriptions. Second, we assess six metrics using vulnerability elements instead of full descriptions. We conducted experiments on our manually labeled dataset. The experimental results show that our approach has an improvement of 12.03%, 14.37%, and 38.65% on Accuracy over three baselines.

1. Introduction

Software vulnerabilities can expose software systems to risks that might be exploited to cause harm or loss (Lomio et al., 2022; Aota et al., 2020; Yitagesu et al., 2021b). The risk of different vulnerabilities to software systems is variable. Apache Log4j2 Remote Code Execution Vulnerability (CVE-2021-44228[1]) is a high-risk vulnerability that emerged recent years and affects over 90% of java-based application platforms (Srinivasa et al., 2022). Compared with medium-risk vulnerabilities such as Divide By Zero,[2] these high-risk vulnerabilities require fixing as soon as possible. It is therefore important to accurately assess the severity of the vulnerabilities in advance.

Software vulnerability assessment is a process that takes the vulnerability data as input, the vulnerability Severity Score as output, and provides vulnerabilities to the developers in descending order of vulnerability Severity Score (Chen et al., 2019). Security companies have defined a series of robust evaluation metrics. Common Vulnerability Scoring System (CVSS)[3] is an open framework that uses the same scoring criteria for each vulnerability. CVSS contains assessment metrics (i.e., *Access Vector*, *Access Complexity*, *Authentication*, *Confidentiality Impact*, *Integrity Impact* and *Availability Impact*) which represents the principal characteristics of a vulnerability and the Severity Score which represents a composite score of assessment metrics. The assessment metrics and the Severity Score are used to help developers select high-risk vulnerabilities for priority fixing.

Existing approaches (e.g., Han et al., 2017; Le et al., 2019) use the full vulnerability descriptions from software repositories to assess their severity level. However, there are three challenges with these approaches. First, the vulnerability descriptions contain many useless elements for severity assessment (such as *Product*, *Version*). Compared with useful elements (e.g., *Attacker*, *Impact*, *Vector*), useless elements make up a large percentage of the description, which increases difficulties in capturing key elements from descriptions. Second, existing approaches use one vulnerability description to predict different assessment metrics (Le et al., 2019; Shahid and Debar, 2021). Different elements in descriptions have different impacts on the assessment of each

☆ Editor: Laurence Duchien.

* Corresponding author at: School of Information Engineering, Yangzhou University, Yangzhou, China.
E-mail addresses: xbsun@yzu.edu.cn (X. Sun), 2361784228@qq.com (Z. Ye), lilibo@yzu.edu.cn (L. Bo), xiaoxuewu@yzu.edu.cn (X. Wu), 252764195@qq.com (Y. Wei), tazhang@must.edu.mo (T. Zhang), lb@yzu.edu.cn (B. Li).

1 https://nvd.nist.gov/vuln/detail/CVE-2021-44228

2 https://cwe.mitre.org/data/definitions/369.html

3 https://www.first.org/cvss/

https://doi.org/10.1016/j.jss.2023.111790
0164-1212/

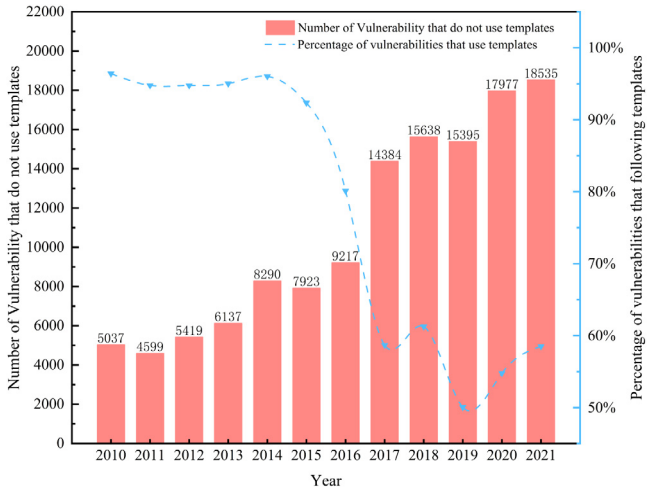


Fig. 1. Number of vulnerability descriptions without using templates.

assessment metric, which makes that the approaches using the full descriptions can not focus on the key points. Especially for some assessment metrics (e.g., *Authentication*), its corresponding key elements consist of only a few words (e.g., "remote authenticated users"), which makes it difficult for the model to learn the information and leads to a decrease in model performance. Third, the format of the vulnerability descriptions has changed a lot in the last decade. CVE[4] discloses the suggested templates for vulnerability descriptions. According to our statistics using regular expressions, the number of vulnerability descriptions that do not use templates has increased each year from 2010 to 2021, shown in Fig. 1. The percentage of vulnerability descriptions using templates decreases from 99.6% in 2010 to 58.5% in 2021. The fact that CVE-ID applicants construct descriptions based on their own habits is thought to be the main reason. However, different applicants focus on different aspects of the description, with some spotlighting on the impact, some on the causes, and others on the method of attack. Based on this finding, the direct use of full vulnerability descriptions may be affected by changes in format. In addition, for most vulnerability descriptions, in the case of not following the templates, the vulnerability descriptions still contains the vulnerability elements. For example, in CVE-2021-29565,[5] where "missing validation that was covered under a 'TODO'" describes the cause of the vulnerability and "the implementation of 'tf.raw_ops. SparseFillEmptyRows'" describes the product of the vulnerability, "the 'dense_shape' tensor is empty" describes the situation of the vulnerability, and " TensorFlow 2.4.2, TensorFlow 2.3.3, TensorFlow 2.2.3 and TensorFlow 2.1.4" describes the version in which the vulnerability occurs. "a null pointer dereference" describes the impact of the vulnerability. Therefore, these elements describing the specific characteristics of the vulnerability (vulnerability elements) are still available, even though it does not follow the templates.

To solve these problems, we propose a novel approach using the vulnerability elements (i.e., *Attacker*, *Impact*, *Vector*, *Situation*, *Cause*, *Product*, *Version*). First, we train a Named Entity Recognition (NER) model to extract vulnerability elements from vulnerability descriptions. Second, we train several classifiers using the extracted elements to predict six assessment metrics. Third, we calculate *Severity Score* using six assessment metrics. In order to demonstrate the effectiveness of our approach, we constructed a vulnerability dataset with 4371 vulnerability elements and selected three baseline approaches for comparison, one using only vulnerability descriptions (Han et al., 2017), one with concept drift (Le et al., 2019), and the other with supplementary texts (Kuehn et al., 2022). The approach using only vulnerability descriptions adopts random undersampling to obtain a balanced dataset and trains the CNN classifier directly using the full vulnerability description. The approach with concept drift uses time-based k-fold cross-validation to filter the most suitable classifier and text representation configuration, then trains the filtered classifier directly using the full vulnerability descriptions as input. The approach with supplementary texts uses a combination of textual information from all NVD reference links with vulnerability descriptions, and the DistilBERT (Sanh et al., 2019) model as a classifier to predict assessment metrics. The dataset is available online.[6]

This paper makes the following three contributions:

- We propose a severity assessment approach with vulnerability elements. Our approach allows for the efficient extraction of vulnerability elements and the effective classification of assessment metrics.
- We manually constructed a vulnerability elements dataset with 4371 data, containing seven elements including *Attacker*, *Impact*, *Vector*, *Situation*, *Cause*, *Product*, *Version*.
- Our approach showed a 1.53%–30.39% improvement over three baseline approaches, and got 14.37%, 12.03%, and 38.65% improvement in recent three years data compared with three baseline approaches (Le's (Le et al., 2019), Kuehn's (Kuehn et al., 2022), and Han's (Han et al., 2017)), respectively.

2. Background & motivation

In this section, we present the definitions related to the vulnerability elements in our approach, the pre-trained model we use, and the motivation of our work.

2.1. Problem definition

Vulnerability Assessment: The vulnerability assessment task refers to assessing the urgency of vulnerabilities to be fixed so that developers can fix the most damaging ones as quickly as possible. In this paper, we focus on *Vulnerability Severity Assessment*, which is intended to predict the *Severity Level* of a vulnerability. Severity is often a function/combination of Exploitation and Impact (Le et al., 2022). According to CVSS v2,[7] there are six assessment metrics for the *Severity Level*, including *Access Vector*, *Access Complexity*, *Authentication*, *Confidentiality Impact*, *Integrity Impact*, *Availability Impact*. *Severity Score* is calculated by six assessment metrics according to the different weights given by CVSS and the calculation formula. *Severity level* is obtained by dividing the *Severity Score* in different ranges. The definitions of assessment metrics are given below:

- **Access Vector (Network/Adjacent Network/Local)**: This metric reflects how the vulnerability is exploited.
- **Access Complexity (Low/ Medium/ High)**: This metric measures the complexity of the attack needed for an attacker to exploit the vulnerability.
- **Authentication (None/ Single/ Multiple)**: This metric measures the number of times an attacker must authenticate a target to exploit a vulnerability.

4 https://cve.mitre.org/

5 https://nvd.nist.gov/vuln/detail/CVE-2021-29565

6 https://github.com/Fino2020/Automatic-Software-Vulnerability-Assessment-by-Extracting-Vulnerability-Elements

7 https://www.first.org/cvss/v2/

- **Confidentiality Impact (None/ Partial/ Complete)**: This metric measures the impact on confidentiality of a successfully exploited vulnerability. Confidentiality refers to limiting information access and disclosure to only authorized users, as well as preventing access or disclosure by unauthorized users.
- **Integrity Impact (None/ Partial/ Complete)**: This metric measures the impact on the integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and guaranteed veracity of information.
- **Availability Impact (None/ Partial/ Complete)**: This metric measures the impact on the availability of a successfully exploited vulnerability. Availability refers to the accessibility of information resources.

Vulnerability Elements: We consider the information in vulnerability descriptions with different meanings as vulnerability elements, because they play a distinctly different role in the vulnerability description, as defined below:

- **Product:** It is the specific product affected by the vulnerability (including systems, software, open source libraries, etc.). In Fig. 2, "The terminal dispatcher" in CVE-2012-2385, and "The Bluetooth Classic implementation" in CVE-2021-28136 are both *Product*.
- **Version:** It is the specific version of the Product affected by the vulnerability. In Fig. 2, "mosh before 1.2.1" in CVE-2012-2385, "Ushahidi Platform 2.5.x through 2.6.1" in CVE-2013-2025, and "Espressif ESP-IDF 4.4 and earlier" in CVE-2021-28136 all belong to *Versio*.
- **Vector:** It refers to a specific way to exploit this vulnerability (e.g., to exploit this vulnerability, someone must open a crafted JPEG file). In Fig. 2, "an escape sequence with a large repeat count value" in CVE-2012-2385, and "a replayed (duplicated) LMP packet" in CVE-2021-28136 are both *Vector*.
- **Attacker:** It is the identity of the initiator of the attack. In Fig. 2, "remote authenticated users" in CVE-2012-2385, "remote attackers" in CVE-2013-2025, and "attackers in radio range" in CVE-2021-28136 all belong to *Attacker*.
- **Impact:** It indicates what the attacker gains by exploiting this vulnerability. In Fig. 2, "a denial of service (long loop and CPU consumption)" in CVE-2012-2385, "inject arbitrary web script or HTML" in CVE-2021-28136, "a use-after-free condition may potentially occur" in CVE-2017-2613 and "trigger memory corruption (and consequently a crash) in ESP32" in CVE-2021-28136 are *Impact*.
- **Cause:** It indicates why the vulnerability occurred. In Fig. 2, "Cross-site scripting (XSS) vulnerability" in CVE-2013-2025, "does not properly handle the reception of multiple LMP IO Capability Request packets" in CVE-2021-28136 are both *Cause*.
- **Situation:** It is meant to be what is the background of the vulnerability occurrence. In Fig. 2, "While calling the IPA IOCTL handler for IPA_IOC_ADD_HDR _PROC_CTX in Android for MSM, Firefox OS for MSM, and QRD Android before 2017-10-13 " in CVE-2017-2613, "the pairing process" in CVE-2021-28136 are *Situation*.

Among the seven vulnerability elements, *Product*, *Version*, *Vector*, *Attacker*, *Impact*, *Cause* are defined by CVE. In our investigation, there exist 7.16% of vulnerability descriptions contain elements that describe the background of the vulnerability occurrence. This information can be helpful in determining the complexity of access and provides necessary information for severity assessment. The formats of this element are generally under While..... (in some background), or if..... (under some condition), and this element is defined as *Situation*. For example, in CVE-2019-15163,[8] "rpcapd/daemon.c in libpcap before 1.9.1 allows attackers to cause a denial of service (NULL pointer dereference and daemon crash) if [a crypt() call fails]"., "a crypt() call fails" describes the specific situation required to exploit the vulnerability.

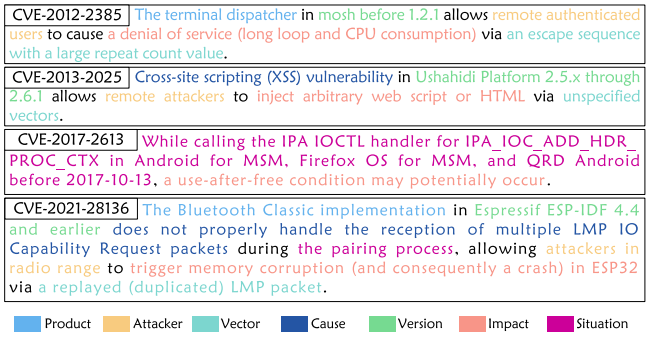


Fig. 2. Example of vulnerability description labeling.

2.2. Pre-trained model

RNN-based models view a text as a sequence of words and are designed to capture word dependencies and text structures for text classification (Karpathy, 2015). Among many variants of RNNs, Long Short-Term Memory (LSTM) (Olah, 2015) is the most popular architecture, which is designed to better capture long-term dependencies.

Unlike RNNs, which are trained to recognize patterns over time, TextCNNs learn to recognize patterns over space. TextCNN (He et al., 2016) uses only one layer of convolution on top of the word vectors obtained from an unsupervised neural language model (Gong and Ji, 2018) and works well where detecting local and position-invariant patterns is important (Chen, 2015).

Classifiers based on attention mechanisms (Vaswani et al., 2017) are becoming an increasingly useful tool (Bahdanau et al., 2015; Luong et al., 2015). The use of attention mechanisms allows for better characterization of text, learning contextual information about words, and assigning weights to words that are contextually relevant (Yang et al., 2016).

2.3. Motivation

Observation 1. **Increasing number of vulnerability descriptions do not follow the specified templates, and for a vulnerability description, 75% of new terms appear in *Product* and *Version*, which are not useful for vulnerability assessment.** The changes in description format and the emergence of new terms can lead to a decrease in model prediction performance.

Fig. 3 shows the number of new terms appearing each year from 2002 to 2021, with an average of 13,125. The number has increased from 2011 to 2017, subsequently stabilized from 2018 to 2021, and new terms consistently occupy around half of the total corpus. Fig. 4 reflects the attribution of new terms appearing each year since 2003. It can be seen that almost 55% of the new terms appear in *Version*, and 20% of the new terms appear in *Product*. Only 25% of new terms appear in other elements, and this proportion is continuously decreasing. Furthermore, *Product* and *Version* only reflect that the vulnerability occurred in a specific version of a product, with no explanation of the attacker, the impact, or the authentication and how the vulnerability was exploited. Using vulnerability elements instead of vulnerability

[8] https://nvd.nist.gov/vuln/detail/CVE-2019-15163

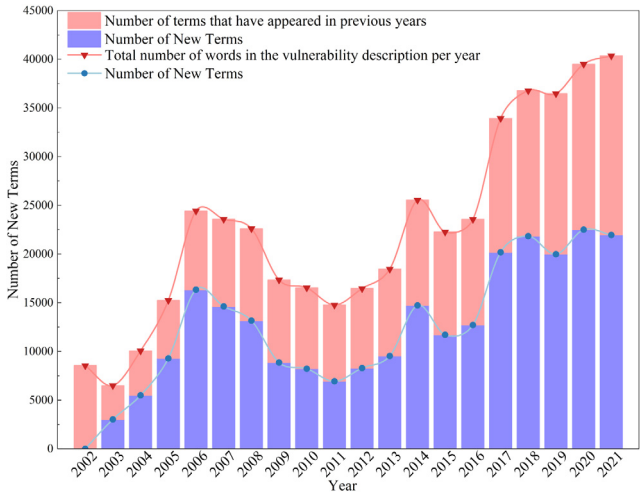


Fig. 3. Emerging terms per year from 2002 to 2021.

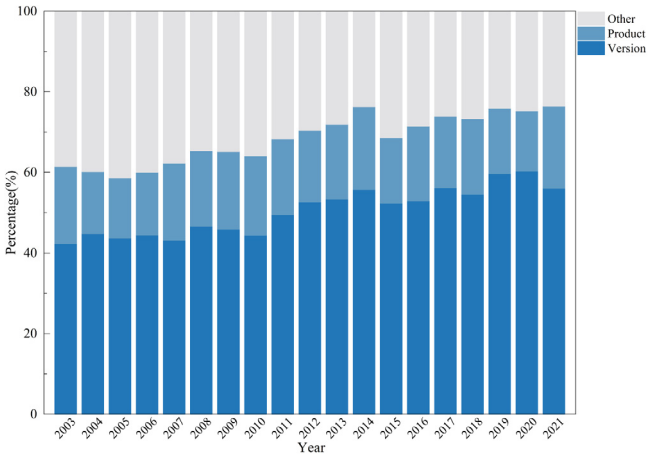


Fig. 4. Proportion of new terms present in different vulnerability elements from 2003 to 2021.

descriptions avoids the impact of useless terms (e.g., *Product* and *Version*) while avoiding the interference of other elements. It is possible to avoid the interference of 75% of the new terms and thus alleviate the OOV (Out Of Vocabulary) problem.

Observation 2. **Different vulnerability elements are correlated with different assessment metrics.** *Attacker* indicates the identity of the initiator of the attack (e.g., "remote authenticated users" in CVE-2018-9250[9] means that the attackers are remote and authenticated); *Authentication* measures the number of times an attacker must authenticate a target to exploit a vulnerability; *Access Vector* reflects how the vulnerability is exploited (the attacker exploited the vulnerability from network or local). All of them focus on the initiator of the attack. *Vector* refers to a specific way to exploit this vulnerability (e.g., "an escape sequence with a large repeat count" in CVE-2012-2385[10] means that the attacker starts the attack by repeating an escape sequence a large number of times); *Situation* meant to be what is the background of the vulnerability occurrence (e.g., "While calling the IPA IOCTL handler for IPA_IOC_ADD_HDR_PROC_CTX in Android for MSM, Firefox OS for MSM, and QRD Android before 2017-10-13" in CVE-2017-2613[11] means that the attack needs to occur when the IPA IOCTL processor is called.); *Access Complexity* measures the complexity of the attacker required to exploit the vulnerability. Each of them focuses on the process of the attack. *Impact* indicates what the attacker could gain by exploiting this vulnerability (e.g., "a denial of service (long loop and CPU consumption)" in CVE-2012-2385 means that the attack will result in a denial of service (long loops and CPU consumption).); *Confidentiality Impact*, *Integrity Impact*, *Availability Impact* describe the degree of impact of the attack on confidentiality, integrity, and availability. All of them described the impact of the vulnerability.

Table 1
The number and percentage of different vulnerability elements in our manually labeled 4371 descriptions.

Vulnerability element	Number	Percentage
Product	3415	78.13%
Version	4207	96.25%
Attacker	2627	60.10%
Vector	2918	66.76%
Impact	3642	83.32%
Situation	313	7.16%
Cause	3445	78.81%

Observation 3. **The number of relevant vulnerability elements is sufficient to train the NER model.** We investigate the number of occurrences of all vulnerability elements in our 4371 manually labeled dataset mentioned in Section 4.2. The number and percentage of occurrences of each element are shown in Table 1. *Version* and *Impact* show up most often, with 96.25% and 83.32% respectively. *Attacker*, *Vector*, *Cause* and *Impact* are all over 60%, only *Situation* appeared in 7.16% of the vulnerability descriptions. Therefore, the volume of the vulnerability element dataset is sufficient to train the extraction model.

3. Our approach

Fig. 5 shows an overview of our approach. The process includes two steps, (1) Vulnerability Elements Extraction, and (2) Vulnerability Elements Classification. The input of our approach is vulnerability descriptions collected from the NVD database, and the output is the score of six assessment metrics (i.e., *Access Vector*, *Access Complexity*, *Authentication*, *Confidentiality Impact*, *Integrity Impact* and *Availability Impact*). *Severity Score* is then calculated by six assessment metrics. In Step(1), we label the vulnerability elements (i.e., *Product*, *Vector*, *Version*, *Attacker*, *Impact*, *Situation*, *Cause*) from vulnerability descriptions and construct the labeled dataset using the BIO labeling method. The labeled dataset is then used to train the NER model. In Step(2), the vulnerability elements extracted by the NER model are provided as input to predict six assessment metrics. At last, we use assessment metrics to calculate *Severity Score* which represents how urgent the vulnerability needs to be tackled.

3.1. Vulnerability elements extraction

We label the vulnerability elements in the vulnerability descriptions collected from NVD. There are two annotators involved in our annotation process, they first follow the criteria and elements defined in Section 2.1 before labeling. For example, "The terminal dispatcher in mosh before 1.2.1 allows remote authenticated users to cause a denial of service (long loop and CPU consumption) via an escape sequence with a large repeat count value". in Fig. 2 CVE-2012-2385[12] can be labeled as "Product: The

9 https://nvd.nist.gov/vuln/detail/CVE-2018-9250
10 https://nvd.nist.gov/vuln/detail/CVE-2012-2385
11 https://nvd.nist.gov/vuln/detail/CVE-2017-2613
12 https://nvd.nist.gov/vuln/detail/CVE-2012-2385

terminal dispatcher", "Version: mosh before 1.2.1", "Attacker: remote authenticated users", "Impact: a denial of service (long loop and CPU consumption)", "Vector: an escape sequence with a large repeat count value". Based on our observations, there is no particular pattern in the ending position of vulnerability elements, nor are there any vulnerability elements with only one word in our annotations. Therefore, we choose the BIO labeling method to label each word as "B-X", "I- X" or "O" (Reimers and Gurevych, 2017). "B-X" means that the word is the beginning word of element X, "I- X " means that the word is the middle word of element X, and "O" means that the word does not belong to any element. For example, "The", "terminal", and "dispatcher" in Fig. 2 CVE-2012-2385 are labeled as "B-Product", "I-Product", "I-Product". "mosh", "before", "1.2.1" are labeled as "B-Version", "I-Version", "I-Version", respectively.

The labeled data will be used to train and validate Named Entity Recognition (NER) model. In order to preserve the integral semantics of the sentence, we use the full descriptions for vulnerability elements extraction. NER refers to the task of detecting the span and the semantic category of entities from a chunk of text (Huang et al., 2015). To extract vulnerability elements from vulnerability descriptions, we choose BERT-MRC (Li et al., 2020) for named entity recognition, because MRC (Machine Reading Comprehension) can make good use of semantic prior information about entity categories by answering predefined questions, the specific demonstration of the problem construction is shown in Table 2. In addition, in vulnerability assessment task, the meaning of vulnerability elements varies widely, and the model should learn the knowledge in advance to better understand the differences between vulnerability element classes. For example, when using the BERT-MRC model to extract Impact from vulnerability description, it is converted to answering "Find the impact in vulnerability description, meaning what the attacker gains by exploiting this vulnerability". Compared with other approaches that extract elements from vulnerability descriptions, Guo et al. (2022) introduced a set of regular expression patterns to extract the six key aspects from CVE descriptions. However, regular expressions are difficult to adapt to the current scenario where vulnerability descriptions increasingly do not follow the specific templates. Yitagesu et al. (2021a) proposed an unsupervised approach to label and extract important vulnerability concepts in textural vulnerability descriptions (TVDs). They build absolute and relative paths for different phrases by constructing syntactic trees and learning patterns of paths unsupervised. However, in recent years, vulnerability descriptions have become less and less templated and the absolute and relative paths of different tokens in the syntactic tree have changed, making their approach difficult to apply to vulnerability descriptions that do not follow the templates. There are also some other NER models: BERT-BiLSTM-CRF (Li et al., 2022), IDCNN-CRF (Cao and Yusup, 2022)), etc, and they are all sequence labeling models. This type of approach lets the model learn the label of each token specifically (Begin, In, Outside) by giving each sequence a label. However, they cannot use semantic prior information about entity categories.

3.2. Vulnerability elements classification

Despite the fact that we filter out some of the stop words by extracting, there exist some stop words in vulnerability elements (e.g., a, the, of). Removing stopwords can reduce the vector space as well as improve the performance by increasing the execution speed, calculations, and accuracy (Kaur and Buttar, 2018). Furthermore, lemmatization reduces the corpus size, and tokenization on word granularity maximizes the integrity of complete semantic information (Bahdanau et al., 2015; Lee et al., 2017; Martinez et al., 2021). Vulnerability elements and vulnerability descriptions are preprocessed using removing stopwords, tokenization, and lemmatization respectively.

Table 2
Questions for different entity settings.

Entity	Natural language question
Product	Find the product in the vulnerability description, including Files, Functions, Projects.
Vector	Find the vector in the vulnerability description, including the specific way to exploit this vulnerability.
Version	Find version in vulnerability description, including systems, software, open source libraries, and specific version number.
Attacker	Find the attacker in the vulnerability description, including Attack initiation channels such as local, network, and adjacent networks.
Impact	Find the impact in the vulnerability description, meaning what the attacker gains by exploiting this vulnerability.
Situation	Find the situation in the vulnerability description, indicating what is the background of the vulnerability occurrence.
Cause	Find cause in the vulnerability description, that is the problem with the Product that causes the vulnerability to be exploited.

According to the observation 1 in Section 2.3, an increasing number of vulnerability descriptions do not follow the specified templates, 75% of new terms appear in *Product* and *Version*. Our insight is that using vulnerability elements instead of the full vulnerability description can avoid interference from irrelevant elements and useless elements. It is possible to prevent around 75% of the new terms and thus alleviate the impact of the OOV problem. Therefore, we train the classifier using vulnerability elements instead of vulnerability descriptions. *Attacker* is used to train the classifiers for *Access Vector* and *Authentication* because all of these focus on the initiator of the attack. *Vector* and *Situation* are used to train the classifier for *Access Complexity* because each of these focuses on the process of the attack. *Impact*, *Cause* and *Situation* are used to train the classifiers for *Confidentiality Impact*, *Integrity Impact*, *Availability Impact*. because they describe the consequences and the conditions for vulnerability to occur. TextCNNs are chosen as classifiers in our approach because they are better at extracting key features than RNNs which use sequential structure. In addition, the lengths of vulnerability elements are generally short, and CNN-based models handle short sentences better than RNN-based models.

Of the 133,639 reported vulnerability descriptions disclosed by CVE over the past 20 years, CVEs of missing vulnerability type, root cause, attack vector, and attacker type achieve 56%, 85%, 38%, and 28%, respectively (Guo et al., 2022). Therefore, there exists a certain percentage of elements extracted by the NER model will be empty. If the required combinations of vulnerability elements are all empty, we need to input the preprocessed descriptions into the classifier instead of the vulnerability elements. According to our statistics, around 10% of combinations require the use of vulnerability descriptions as an alternative, because it is described too briefly. Preprocessed vulnerability descriptions from 2002 to 2016 are used to pre-train the classifiers of six assessment metrics. Then we utilize vulnerability elements to fine-tune these classifiers. The output of classifiers is the category of each assessment metric. Finally, we calculate the *Severity Score* according to CVSS v2 framework.

4. Experimental design

In this section, the dataset for our approach is presented, along with research questions, the classifier model we chose, the experimental setup, and evaluation metrics.

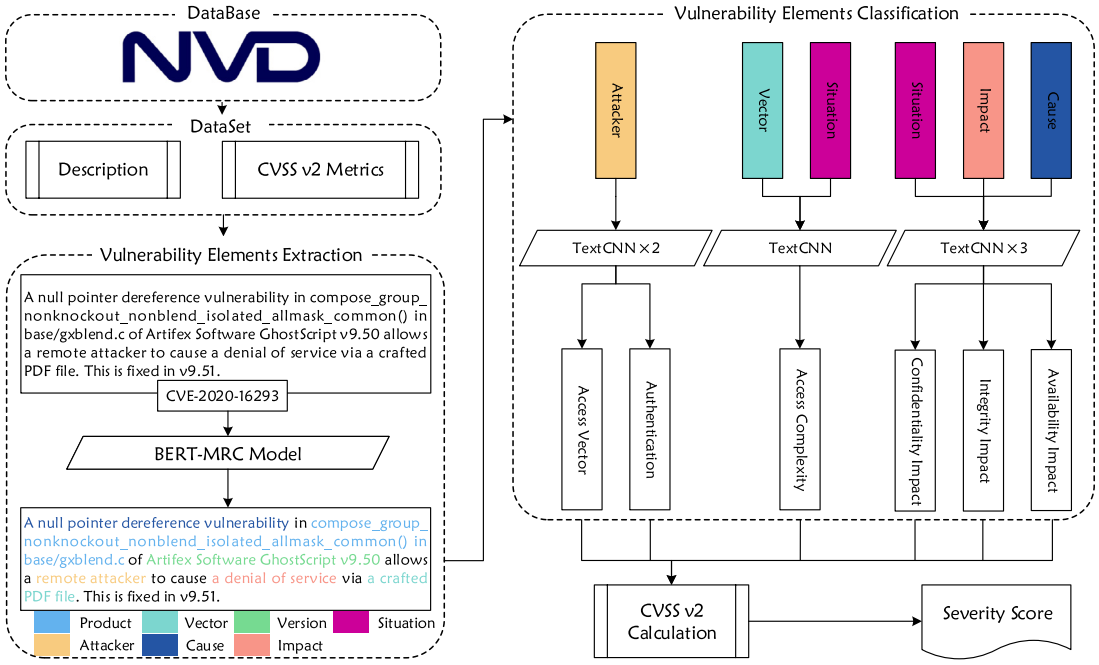


Fig. 5. Overview of our approach.

4.1. Dataset

The dataset construction process is divided into three steps. First, we collect all vulnerability descriptions on NVD that contain fix links from 2002 to January 2022, as fix patches could effectively help us to distinguish *Situations* (what is the background of the vulnerability occurrence), *Vector* (by what means the attacker caused the vulnerability) and *Cause* (why an attacker could exploit the vulnerability). Second, we manually label all the vulnerability elements in the vulnerability descriptions (i.e., *Product*, *Vector*, *Version*, *Attacker*, *Impact*, *Situation*, *Cause*). This dataset was labeled by two annotators individually. When 5%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, and 100% of the dataset were labeled, we cross-verified the two labeled results. Cohen's Kappa coefficient rating aggregation was used to calculate the labeling difference between the two callers (Landis and Koch, 1977). In total 11 rounds of labeling, each round contains the dataset from the previous round. The Kappa coefficient ranges from 0.0% to 100.0%, and the larger the value, the better the consistency. In general, a Kappa coefficient greater than 80% is considered to be nearly identical between the two evaluations. In our labeling process, the overall kappa coefficient is 90.62% ($>$ 90%). Third, we collect the vulnerability information from NVD corresponding to the data items we labeled. The information we collect includes the vulnerability descriptions, Severity Score, the status of the vulnerability analysis (Sun et al., 2022), whether the vulnerability is disputed, and assessment metrics (e.g., AV:L/AC:M/Au:N/C:N/I:N/A:C means that the attacker initiates the attack locally, the complexity of the attacker to exploit the vulnerability is medium, the attacker does not need to be authorized to exploit the vulnerability, the exploitation of the vulnerability will not have an impact on confidentiality (such as restricting access and disclosure of information only to authorized users), will not have an impact on integrity (such as the trustworthiness of information and assurance of authenticity), will completely change the availability (availability of information resources)). We filter out those vulnerability items whose analysis status is RECEIVED and AWAITING ANALYSIS because NVD has not yet analyzed them. We also filter out items with controversy (i.e., vulnerability descriptions that begin with ** DISPUTE **, ** REJECT **, or **UNSUPPORTED WHEN ASSIGNED **), and ended up with 4371 data items.

4.2. Research questions

RQ1: *Can vulnerability elements improve the performance of classification compared with that using vulnerability descriptions?*

To answer RQ1, we quantitatively analyze the advantages of using vulnerability elements compared with other approaches (using the vulnerability descriptions directly and removing irrelevant elements from the vulnerability description).

RQ2: *Is our vulnerability assessment approach more effective than the existing approaches using vulnerability descriptions?*

To answer RQ2, we reappear three baseline approaches and evaluated their approaches and our approach on all datasets in order to validate the effectiveness of our entire approach.

RQ3: *How do the different vulnerability elements (combinations) affect different assessment metrics?*

To answer RQ3, we compare the impact of different combinations of vulnerability elements on the classification results of assessment metrics to determine how different vulnerability elements contribute to different assessment metrics. Specifically, we analyze the impact of six elements and their combinations on different assessment metrics.

RQ4: *How effective is our approach on recent three years datasets?*

To answer RQ4, we evaluate our approach on a total of 38,541 data (removing common parts with our dataset) over the recent three years, while comparing the validity of other baseline approaches.

4.3. Experimental setup

We implemented our approach using Python 3.7.12, with PyTorch-1.11.0+cu113, and the hyper-parameters for the deep learning models are set as follows: learning rate is 5e-4 (5e-5 for fine-tuning), the number of channels is 256, the batch size is 128, training epochs is 100 and the maximum length of a sequence

Table 3
The Impact of Training Classifiers Using Dataset with *Product* and *Version* Removed on The Test Set.

Preprocess	Classifier	Access complexity	Access vector	Authentication	Confidentiality impact	Integrity impact	Availability impact
Full description	TextCNN	76.19%	93.70%	93.24%	80.34%	80.65%	79.72%
	LSTM	64.36%	78.65%	89.55%	47.31%	49.62%	73.58%
	TextRNN-Att	71.12%	94.01%	94.01%	80.95%	78.96%	79.42%
Delete Ver&Pro	TextCNN	77.73%	95.70%	94.32%	79.72%	81.11%	82.95%
	LSTM	68.36%	94.01%	89.55%	75.12%	76.19%	78.65%
	TextRNN-Att	71.74%	95.85%	94.16%	77.42%	80.03%	82.03%
Using Vulnerability Elements	TextCNN	82.33%	96.47%	88.79%	79.72%	90.63%	88.79%
	LSTM	76.50%	78.65%	78.65%	76.65%	76.65%	79.57%
	TextRNN-Att	77.27%	95.08%	96.01%	81.87%	78.96%	83.87%

is limited to 50. These optimal hyper-parameters were chosen to train a robust yet lightweight model. Our experiments were conducted on a Windows 10 machine, with Intel Core i7-12700KF CPU(3.6GHZ), 32 GB memory, and NVIDIA RTX3070Ti with 8 GB RAM.

4.4. Evaluation metrics

We choosed *Accuracy*, *Weighted F1-score*, and *Marco F1-score* for classification evaluation. Because *Micro F1-score*, *Accuracy*, *Micro-Precision*, *Micro-Recall* are equivalent in multi-classification scenarios. *Precision*, *Recall* and *F1-score* are used to evaluation for BERT-MRC. There are four possibilities for prediction:

- **TP**: The model predicts as a positive sample, the actual sample is positive.
- **TN**: The model predicts as a negative sample, but the actual sample is positive.
- **FP**: The model predicts as a positive sample, but the actual sample is negative.
- **FN**: The model predicts as a negative sample, the actual sample is negative.

Accuracy, *Precision*, *Recall* and *F1-score* are calculated as follows:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

$$F1\text{-}score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{4}$$

5. Experimental results

In this section, we analyze and discuss the experiment results and answer four RQs.

5.1. RQ1: Can vulnerability elements improve the performance of classification compared with using vulnerability descriptions?

For the approach that directly uses the full description, we trained three types of classifiers (TextCNN, LSTM, TextRNN-Att) separately, using the assessment metrics (*Access Vector*, *Access Complexity*, *Authentication*, *Confidentiality Impact*, *Integrity Impact*, *Availability Impact*) as labels and vulnerability descriptions as input. For the approach of removing irrelevant elements, we first removed *Product* and *Version* from the vulnerability descriptions, as these two elements only describe the version and specific location where the vulnerability occurred. Next, the three types of classifiers were trained using altered vulnerability descriptions as input and assessment metrics as labels. For the approach of extracting vulnerability elements, we first trained the BERT-MRC model and then trained the three classifiers using vulnerability elements (*Cause*, *Vector*, *Impact*, *Version*, *Attacker*, *Product*, *Situation*) as input and assessment metrics as labels. The relationship between the inputs (vulnerability elements) and the labels (assessment metrics) is shown in Table 5. The performance of the BERT-MRC model is shown in Table 4. The Precision of *Cause*, *Vector*, *Impact*, *Version*, *Attacker*, *Product*, *Situation* obtain 85.22%, 81.93%, 84.11%, 86.03%, 93.90%, 82.28%, 60.42%, respectively.

Table 4
Performance of BERT-MRC model with different element.

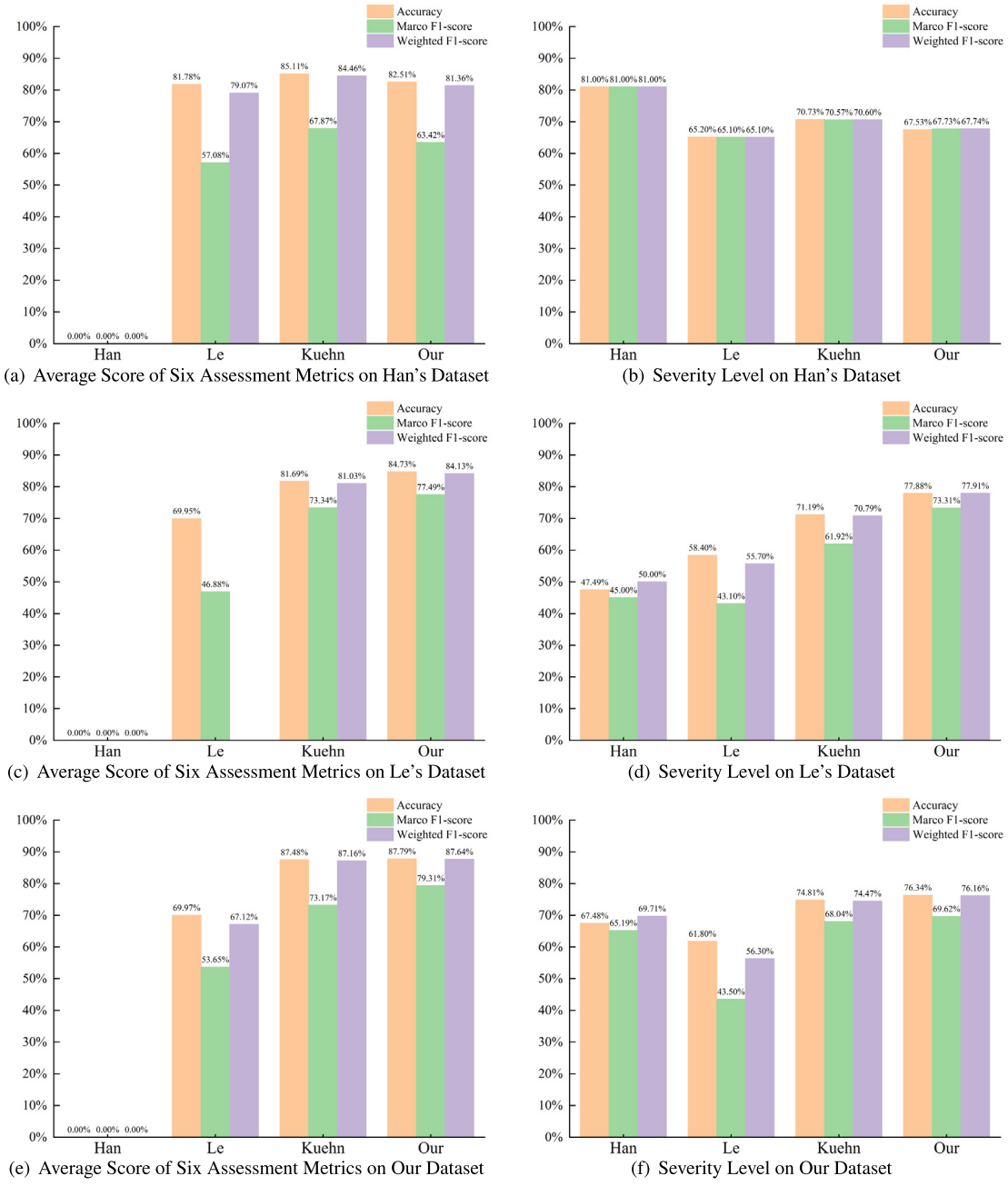
Element	Precision	Recall	F1-score
Cause	85.22%	89.64%	87.37%
Vector	81.93%	86.69%	84.24%
Impact	84.11%	89.03%	86.50%
Version	86.03%	89.14%	87.56%
Attacker	93.90%	95.47%	94.67%
Product	82.28%	86.69%	84.24%
Situation	60.42%	74.36%	66.67%

We evaluated the three approaches on our labeled dataset, and the results are shown in Table 3. Removing *Product* and *Version* can obtain better results than using full vulnerability descriptions, achieving a 4% improvement in almost all assessment metrics and classifiers. We concluded that by removing *Product* and *Version*, which are highly affected by the OOV problem, the accuracy of the classifier can be improved. However, removing some elements in the text also leads to the destruction of the semantic information of vulnerability description, and compared with preprocessing operations e.g., Lemmatization (Plisson et al., 2004), Stemming (Lovins, 1968), removing some elements from the text had a greater impact on the semantic information (Stavrianou et al., 2007). Furthermore, compared with RNN with sequential structure, CNN is better at extracting key features. Only if the full sequence needs to be understood at the semantic level, RNN outperforms CNN (Yin et al., 2017; Zhang and Wang, 2016). The approach of extracting vulnerability elements had a 5% to 10% improvement over the other two scenarios. Using elements rather than full descriptions could reduce the interference caused by other elements. And compared with the scenario that deletes *Version* and *Product*, our approach can avoid interference from missing semantics.

Summary of RQ1: Using vulnerability elements for classification reduces the interference of useless and irrelevant elements as well as the semantic deficiencies caused by the deletion of elements. Therefore, the vulnerability elements can improve the performance of classification.

Table 5
Correspondence of vulnerability elements and assessment metrics.

Assessment metrics	Vulnerability elements						
	Product	Vector	Version	Attacker	Impact	Situation	Cause
Access vector	×	×	×	✓	×	×	×
Access complexity	×	✓	×	×	×	×	×
Authentication	×	×	×	✓	×	×	×
Confidentiality impact	×	×	×	×	✓	✓	✓
Integrity impact	×	×	×	×	✓	✓	✓
Availability impact	×	×	×	×	✓	✓	✓



(a) Average Score of Six Assessment Metrics on Han's Dataset
(b) Severity Level on Han's Dataset
(c) Average Score of Six Assessment Metrics on Le's Dataset
(d) Severity Level on Le's Dataset
(e) Average Score of Six Assessment Metrics on Our Dataset
(f) Severity Level on Our Dataset

Fig. 6. Performance of our approach and three baselines on three test sets.

5.2. RQ2: Is our vulnerability assessment approach more effective than the existing approaches using vulnerability descriptions?

We compared our approach with three baseline approaches (Han's (Han et al., 2017), Le's (Le et al., 2019), and Kuehn's (Kuehn et al., 2022)). Han's approach trains the CNN model directly using the full vulnerability descriptions and the *Severity Levels*. They first count the number of minimum classes and apply random undersampling to the other classes until the number of all classes is balanced. Le's approach uses time-based k-fold cross-validation, dividing the dataset into k equal parts by year, then iteratively adding each validation set to the training set and using a new test set, the model and NLP representation with the least loss is selected with minimum loss of the test set in those k times. Kuehn's approach uses a combination of textual information from all NVD reference links with vulnerability descriptions, then trains the DistilBERT model as a classifier to predict assessment metrics. We reappeared their work along with the original ideas in their papers. For Han's approach, we collected all NVD descriptions from 2002 to 2018 and counted the number of minimal classes, and then undersampled other classes to balance all classes, which obtained a total of 17,802 data. We provided the *Severity Levels* as labels and directly used these vulnerability descriptions as input for model training. For Le's approach, we selected all NVD data from 2002 to 2018, for a total of 107,365 data. We provided the six assessment metrics as labels and train six classifier models directly using these vulnerability descriptions as inputs (we used the models from the original paper, e.g., Extreme Gradient Boosting (XGB) (Chen and Guestrin, 2016), Random Forest (Ho, 1995), Support Vector Machine (SVM) (Cortes and Vapnik, 1995), etc.). For Kuehn's approach, we first crawled all the text information from all the NVD-related links (including commit messages on the GitHub link, etc.), to ensure fairness, we trained with all the NVD vulnerability descriptions from 2002 to 2018 same as Le's and our approaches, and added text combinations as additional input. we selected DistilBERT, the best-performing classifier mentioned in their paper, as the classifier. Six assessment metrics were first predicted and then used to calculate *Severity Score* and *Severity Level*. For our approach, we first pre-trained using 2002–2018 NVD data and then fine-tuned using 4371 manually labeled data mentioned in Section 4.1. All three datasets are randomly divided into the training set, validation set, and test set in the ratio of 8:1:1. We evaluated three approaches on three test sets because these three test sets are different in terms of the time dimension, data distribution, and volume.

We evaluated the four approaches on three test sets, and the results are shown in Fig. 6. Han's approach directly predicts *Severity Level*, so the column in average assessment metrics score is empty. Their approach performs well in their test set but is not optimistic in other test sets. We believe that it is limited by the size of the dataset, which makes it difficult to generalize to new descriptions, and their approach is less effective in reappearing the results due to the random undersampling approach, which is uncontrollable. At the same time, the other three approaches use assessment metrics to calculate *Severity Score* and *Severity Level*, which leads to further propagation of errors in the prediction step. We argue that this is the reason for the unsatisfactory results of the three approaches on *Severity Level*. Le's approach did not perform well in all the experiments. We consider this because their approach uses time-based K-fold cross-validation to select the most appropriate classifier and text representation. Although it can reduce the influence of new terms to a certain extent, it still cannot avoid the influence of irrelevant elements in the text. Elements such as *Product* and *Version* where the new terms occur most frequently are retained. Kuehn's approach presents sub-optimal results in all experiments. Their approach uses text information from NVD-related links as supplementary input, although they do not explicitly demonstrate in the paper that these inputs necessarily lead to performance improvements (Kuehn et al., 2022). We consider that the supplementary input used in their approach comes from various sources, some from commit messages, some from IBM X-Force Exchange,[13] and some from different enterprise websites. These supplementary text messages are written in a very different style from the vulnerability descriptions in NVD. At the same time, only about 15% of the vulnerability items are able to crawl to relevant links, which also reduces the effectiveness of supplementary input.

Compared with Han's approach, which only performs well on its own dataset, our approach has better generalization ability. Our approach outperforms both Le's test set and our test set. Compared with Le's approach, our approach is more effective on all three test sets, and the intermediate results (vulnerability elements) of our approach could also help developers to evaluate vulnerabilities. We believe this is due to the fact that the information in the vulnerability elements is more concise and does not contain irrelevant information (e.g., *Product* and *Version*), as well as the fact that there is a correlation between the vulnerability elements and the assessment metrics, with some vulnerability elements and assessment metrics both describing the attacker and others both describing the impact. Therefore, it is difficult to judge whether the model has learned the correct knowledge by training six classifiers using the same description. Compared with Kuehn's approach, our approach can achieve better performance without using the general corpus pretraining model.

> **Summary of RQ2:** Our approach is able to have more effective performance on different test sets and has a 1.53%-17.82% improvement in the average score of assessment metrics. And our intermediate results (i.e., vulnerability elements) are a good help for developers. Therefore, vulnerability assessment using our approach is more effective than existing approaches using full vulnerability descriptions.

5.3. RQ3: How do the different vulnerability elements (combinations) affect different assessment metrics?

We conducted experiments on our dataset. First, we divided the six elements into four categories, *Vector* and *Attacker* characterizing the attacker, *Impact* characterizing the attack consequence, *Cause* and *Situation* characterizing the conditions under which the attack occurred. Second, we trained six assessment metrics classifiers using each vulnerability element and the three categories, separately. A combination of *Impact*, *Cause*, and *Situation* is added because they all describe the vulnerability rather than the attacker. We did not train the classifier for *Situation* and *Cause&Situation* because *Situation* occurs infrequently. For combinations in which all vulnerability elements are empty, we prepared the preprocessed vulnerability descriptions as replacements. In our experiments, only about 10% of the dataset used the entire vulnerability description.

We evaluated the influence of these eight elements (or combinations) as inputs on our dataset, and the results are shown in Table 6. For *Access Complexity*, the best results are achieved by using *Vector* as input only. For *Authentication*, training with *Attacker* achieves 93.39% in Accuracy and 92.35% in Weighted F1-score, which is comparable to the control group using *Vector & Cause & Attacker*, while considering that the more vulnerability elements

[13] https://exchange.xforce.ibmcloud.com/

Table 6
Performance of different combinations of vulnerability elements for different assessment metrics.

Vulnerability elements	Access vector			Access complexity			Authentication			Confidentiality impact			Integrity impact			Availability impact		
	A%	MF%	WF%	A%	MF%	WF%	A%	MF%	WF%	A%	MF%	WF%	A%	MF%	WF%	A%	MF%	WF%
Product	61.50	56.50	61.30	64.67	54.93	63.13	63.30	56.10	63.00	53.92	42.76	51.32	62.21	50.06	59.85	60.22	52.84	57.58
Version	91.24	57.98	91.18	65.28	54.92	64.93	90.78	61.82	88.13	56.53	50.64	55.44	67.43	58.37	66.22	68.20	67.22	67.36
Vector	88.94	54.50	88.22	**74.04**	**66.23**	**73.85**	91.09	63.10	88.52	64.52	56.78	63.50	67.43	55.09	65.70	69.28	66.18	68.87
Attacker	**95.85**	**62.53**	**95.76**	71.74	65.45	71.63	**93.39**	**76.92**	**92.35**	62.06	58.15	61.21	65.59	55.05	63.81	66.21	62.87	64.45
Impact	85.87	52.60	85.80	71.74	59.19	71.31	90.17	58.35	87.15	79.42	73.37	79.19	81.87	74.71	81.42	82.49	79.77	82.15
Cause	85.87	49.80	84.29	71.12	61.72	70.89	90.94	63.66	88.57	68.36	57.63	66.79	70.05	56.96	68.22	74.50	69.03	73.15
Vector&Attacker	94.47	61.00	94.30	72.96	64.10	72.81	93.55	76.83	92.40	61.14	54.37	60.44	65.75	55.59	64.69	69.89	67.76	69.59
Impact&Cause&Situation	84.95	50.62	84.33	73.43	62.99	72.87	90.63	57.90	87.27	**81.41**	**75.54**	**81.27**	**82.95**	**77.97**	**82.77**	**84.64**	**82.21**	**84.38**

are used, the more the error increases in the NER step, so we consider it appropriate to use only *Attacker*. For *Confidentiality Impact*, *Integrity Impact*, and *Availability Impact*, the combination of *Impact&Situation&Cause* can achieve better results than other combinations. The correspondence between vulnerability elements and assessment metrics is shown in Table 5.

Summary of RQ3: Our experiments show that *Attacker* has a direct impact on *Access Vector* and *Authentication*; *Vector* has a direct impact on *Access Complexity*; *Impact*, *Cause* and *Situation* have a strong impact on *Confidentiality Impact*, *Integrity Impact* and *Availability Impact*.

5.4. RQ4: How effective is our approach on recent three years datasets?

We collected all NVD descriptions from 2020 to January 2022, removing ** DISPUTE **, ** REJECT **,** UNSUPPORTED WHEN ASSIGNED **, finally getting 38,541 vulnerability descriptions, and then compared the performance of Han's, Le's, Kuehn's, and our approach under this dataset and the results are shown in Table 7.

Han's approach achieves the lowest results, 34.12% in Accuracy (A), 25.96% in Marco F1-score (MF), and 43.97% in Weighted F1-score (WF), which has a huge difference from the results on their own test set (81.00% in Accuracy, 81.00% in Marco F1-score and 81.00% in Weighted F1-score). We consider that this is due to the insufficient number and unpredictable distribution of their dataset which is caused by the random undersampling method. Furthermore, Han's approach does not work well for data outside of the training set attributed to the new terms and the increasing number of vulnerability descriptions not using templates.

Our approach outperforms Le's approach in all assessment metrics and *Severity Level*, improving by 7.8% in *Access Complexity*, 11.8% in *Access Vector*, 11.1% in *Authentication*, 9.6% in *Confidentiality Impact*, 11.7% in *Integrity Impact*, 16.9% in *Availability Impact*, and 17% in *Severity Level*. We believe that it depends on the use of vulnerability elements instead of full descriptions, using vulnerability elements instead of full descriptions reduces the impact of new terms present in *Product*, *Version* while avoiding the interference of other vulnerability elements.

Our approach outperforms Kuehn's approach in *Access Vector* (10.27% in Accuracy), *Confidentiality Impact* (0.64% in Accuracy), *Integrity Impact* (3.59% in Accuracy), *Availability Impact* (6.40% in Accuracy), and *Severity Level* (12.03% in Accuracy). Although generic pre-trained corpus is nor used, our approach is still more effective than Kuehn's approach in the recent three years of data. We consider that this is due to the fact that our approach uses the vulnerability elements for classification, which effectively avoids the interference of useless and irrelevant elements and makes it easier to accurately predict assessment metrics (e.g., *Access Vector*, *Confidentiality Impact*, *Integrity Impact*, and *Availability Impact*) that correspond strongly to the vulnerability elements. Furthermore, different assessment metrics contributed differently to the *Severity Score*, with *Access Vector*, *Confidentiality Impact*, *Integrity Impact*, and *Availability Impact* having a greater impact on the *Severity Score* compared with the other metrics. Errors that occur at prediction will be further propagated in the *Severity Score* calculation, which leads to sub-optimal results for Kuehn's approach.

Summary of RQ4: Our approach showed significant improvements over Le's, Han's, and Kuehn's approach in almost all assessment metrics and *Severity Level*. Therefore, our approach is effective in the recent three years of data.

6. Threats to validity

In this section, we discuss the threats to our approach, including external threats and internal threats.

Internal Validity. (1) In the annotation phase, our annotation results rely on expert knowledge and our subjective judgment, some errors can occur and we alleviate this threat by annotating with several people. (2) We choose the *spaCy* library for POS tagging and lemmatization when preprocessing, which does not achieve 100% accuracy for POS tagging. (3) *Situation* occurs less frequently and the model may not learn the feature when classifying *Confidentiality Impact*, *Integrity Impact*, and *Availability Impact*. (4) Despite our approach mitigating the interference of new terms by filtering out irrelevant vulnerability elements, there are still around 25% new terms in the vulnerability elements that are input to the classifiers, thus our approach is still somewhat affected by the OOV problem. It could be further alleviated by expanding the size of the pre-trained corpus.

External Validity. (1) Our experiments require fine-tuning on a certain number of manually labeled datasets and may require adding some new annotations when faced with some special cases, however, annotating the datasets in the real world may require more human effort, and incorrect annotations may lead to less effectiveness. (2) When none of the vulnerability elements or combinations are available, we provide the full vulnerability description as a replacement, so for vulnerability descriptions with extremely missing vulnerability elements, our approach only achieves the same result as the baseline. Furthermore, we need to periodically retrain our pre-train model when the format of vulnerability descriptions changes significantly.

7. Related work

There are two existing mainstream approaches to vulnerability assessment, using descriptions and using code. A vulnerability description is a piece of natural language written by the CVE-ID applicant during the application process according to given templates or compiled by the applicant. After the CVE has accepted the CVE-ID application, NVD will periodically obtain CVE data and manually analyze the type, and severity of each CVE. Existing

Table 7
Evaluate performance of Le's, Han's, and our approaches on 38541 descriptions from 2020 to 2022.

Approach	Access vector			Access complexity			Authentication			Confidentiality impact			Integrity impact			Availability impact			Severity level		
	A	MF	WF	A	MF	WF	A	MF	WF	A	MF	WF	A	MF	WF	A	MF	WF	A	MF	WF
Han	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	34.12%	25.96%	43.97%
Le	82.40%	42.00%	77.50%	68.40%	39.80%	61.90%	75.40%	28.90%	65.20%	63.30%	56.10%	63.00%	68.70%	58.00%	67.60%	61.50%	56.50%	61.30%	58.40%	43.10%	55.70%
Kuehn	79.94%	55.42%	79.07%	92.43%	73.07%	92.50%	89.35%	78.57%	88.26%	72.32%	61.01%	71.16%	76.80%	66.16%	79.03%	71.96%	67.97%	72.87%	60.74%	51.83%	60.23%
Our	90.21%	69.90%	89.30%	83.12%	59.47%	82.68%	86.50%	79.83%	85.77%	72.96%	66.06%	71.94%	80.39%	69.76%	78.86%	78.36%	69.09%	77.09%	72.77%	67.37%	72.66%

work suggests that vulnerability descriptions can support the severity assessment task (Han et al., 2017). Han et al. (2017) concluded that the severity assessment task could be accomplished by directly using the full vulnerability descriptions as input, they filtered the non-minimal categories by random undersampling until the dataset reached equilibrium and subsequently trained a CNN classifier, in their work the vulnerability descriptions are directly used to predict *Severity Level* (Low/Medium/High). Sahin et al. (Sahin and Tosun, 2019) follow Han's approach on feature extraction using word embeddings, and on prediction model using Convolutional Neural Networks (CNNs). In addition, Long Short Term Memory (LSTM) and Extreme Gradient Boosting (XGBoost) models are used. They also extend their work by aiming to predict *Severity Scores* rather than levels. Le et al. (2019) discuss the impact of Concept Drift on the dataset and accordingly devise a time-based k-fold cross-validation method to address the problem. They use the full vulnerability descriptions as input to predict six assessment metrics and *Severity Level*, then they filter the most suitable classifier and text representation configurations for each assessment metric in the cross-validation stage and use those corresponding configurations for training in the training stage. Kuehn et al. (2022) crawl all the text information from all the NVD-related links and add text combinations as additional input. However, their work does not give a definite statement on whether supplementary inputs can improve the predictive performance of the model. Kudjo et al. (2020) investigated the feasibility of using Bellwethers to improve vulnerability severity prediction. They improved the accuracy of the severity prediction by filtering a sample subset (Bellwether) of the dataset. Furthermore, Le et al. (2022) investigated data-driven severity assessment and prioritization approaches prior to 2022, categorizing by task type (Exploit Likelihood, Exploit Time, Exploit Characteristics, Impact Prediction, Severity Prediction).

In addition to the use of vulnerability descriptions for severity assessment, the use of code for severity assessment is also emerging. Le et al. (2021) are proposing to assess vulnerabilities immediately after they are detected to provide information about the exploitability, impact, and severity of software vulnerabilities. They focus on the automatic evaluation of software vulnerabilities early in the software system. The approach uses Vulnerability-Contributing Commit to train a classifier to obtain assessment metrics. In another paper by Le et al. (Le and Babar, 2022), their research investigated the possibility of using vulnerable statements as input for evaluation and found that the number of vulnerable statements was 5.8 times smaller than that of non-vulnerable statements, but the performance of models trained with vulnerable statements was 7.5%–114.5% higher than models trained with non-vulnerable statements. Their approach also provides a function-level severity assessment model.

8. Conclusion

In this paper, we analyze the changing trends of vulnerability descriptions. We propose a novel approach to assess vulnerability severity, which uses vulnerability elements instead of descriptions. Experiments demonstrate that our approach allows for higher Accuracy in the assessment of vulnerability severity.

We find that using *Situation* for combined classification training can effectively improve classification performance. In the future, we plan to use the code for automatic *Situation* generation. Existing work does not consider the correlation between different vulnerabilities, which is also a direction worth investigating. Furthermore, when a vulnerability is discovered but not yet assessed for severity, some attributes (including Vulnerability Type, Situation, and Cause) may be temporarily unavailable, automatic generation of these elements using vulnerability codes and patch codes can improve the performance of vulnerability assessments.

CRediT authorship contribution statement

Xiaobing Sun: Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Zhenlei Ye:** Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Lili Bo:** Conceptualization, Methodology, Supervision, Project administration. **Xiaoxue Wu:** Conceptualization, Supervision. **Ying Wei:** Supervision, Writing – review & editing. **Tao Zhang:** Software, Validation. **Bin Li:** Software, Validation.

Declaration of competing interest

The authors declare that they have no conflict of interest.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61872312, No. 61972335, No. 62002309); the Six Talent Peaks Project in Jiangsu Province (No. RJFW-053), the Jiangsu "333" Project; the Natural Science Foundation of the Jiangsu Higher Education Institutions of China (No. 20KJB520016); the Open Funds of State Key Laboratory for Novel Software Technology of Nanjing University (No. KFKT2022B17), the Innovation (Science and technology) project of Scientific Research Base of Nanjing University of Aeronautics and Astronautics (No. NJ2020022), the Yangzhou University Interdisciplinary Research Foundation for Animal Husbandry Discipline of Targeted Support (yzuxk202015), the Yangzhou city-Yangzhou University Science and Technology Cooperation Fund Project (No. YZ2021157) and Yangzhou University Top-level Talents Support Program (2019).

References

Aota, M., Kanehara, H., Kubo, M., Murata, N., Sun, B., Takahashi, T., 2020. Automation of vulnerability classification from its description using machine learning. In: IEEE Symposium on Computers and Communications. ISCC 2020, Rennes, France, July 7-10, 2020, IEEE, pp. 1–7. http://dx.doi.org/10.1109/ISCC50000.2020.9219568.

Bahdanau, D., Cho, K., Bengio, Y., 2015. Neural machine translation by jointly learning to align and translate. In: Bengio, Y., LeCun, Y. (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. URL: http://arxiv.org/abs/1409.0473.

Cao, Y., Yusup, A., 2022. Chinese electronic medical record named entity recognition based on BERT-WWM-IDCNN-CRF. In: 9th International Conference on Dependable Systems and their Applications. DSA 2022, Wulumuqi, China, August 4-5, 2022, IEEE, pp. 582–589. http://dx.doi.org/10.1109/DSA56465.2022.00084.

Chen, Y., 2015. Convolutional Neural Network for Sentence Classification (Masters thesis). University of Waterloo.

Chen, T., Guestrin, C., 2016. Xgboost: A scalable tree boosting system. In: Krishnapuram, B., Shah, M., Smola, A.J., Aggarwal, C.C., Shen, D., Rastogi, R. (Eds.), Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016. ACM, pp. 785–794. http://dx.doi.org/10.1145/2939672.2939785.

Chen, H., Liu, J., Liu, R., Park, N., Subrahmanian, V.S., 2019. VEST: a system for vulnerability exploit scoring & timing. In: Kraus, S. (Ed.), Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence. IJCAI 2019, Macao, China, August 10-16, 2019, ijcai.org, pp. 6503–6505. http://dx.doi.org/10.24963/ijcai.2019/937.

Cortes, C., Vapnik, V., 1995. Support-vector networks. Mach. Learn. 20 (3), 273–297. http://dx.doi.org/10.1007/BF00994018.

Gong, L., Ji, R., 2018. What does a TextCNN learn? arXiv preprint arXiv:1801.06287.

Guo, H., Chen, S., Xing, Z., Li, X., Bai, Y., Sun, J., 2022. Detecting and augmenting missing key aspects in vulnerability descriptions. ACM Trans. Softw. Eng. Methodol. 31 (3), 49:1–49:27. http://dx.doi.org/10.1145/3498537.

Han, Z., Li, X., Xing, Z., Liu, H., Feng, Z., 2017. Learning to predict severity of software vulnerability using only vulnerability description. In: 2017 IEEE International Conference on Software Maintenance and Evolution. ICSME 2017, Shanghai, China, September 17-22, 2017, IEEE Computer Society, pp. 125–136. http://dx.doi.org/10.1109/ICSME.2017.52.

He, T., Huang, W., Qiao, Y., Yao, J., 2016. Text-attentional convolutional neural network for scene text detection. IEEE Trans. Image Process. 25 (6), 2529–2541. http://dx.doi.org/10.1109/TIP.2016.2547588.

Ho, T.K., 1995. Random decision forests. In: Third International Conference on Document Analysis and Recognition, Vol. 1. ICDAR 1995, August 14 - 15, 1995, Montreal, Canada, IEEE Computer Society, pp. 278–282. http://dx.doi.org/10.1109/ICDAR.1995.598994.

Huang, Z., Xu, W., Yu, K., 2015. Bidirectional LSTM-CRF models for sequence tagging, CoRR abs/1508.01991. URL: http://arxiv.org/abs/1508.01991.

Karpathy, A., 2015. The unreasonable effectiveness of recurrent neural networks. Andrej Karpathy Blog 21, 23.

Kaur, J., Buttar, P.K., 2018. A systematic review on stopword removal algorithms. Int. J. Future Revolut. Comput. Sci. Commun. Eng. 4 (4), 207–210.

Kudjo, P.K., Chen, J., Mensah, S., Amankwah, R., Kudjo, C., 2020. The effect of Bellwether analysis on software vulnerability severity prediction models. Softw. Qual. J. 28, 1413–1446.

Kuehn, P., Relke, D.N., Reuter, C., 2022. Common vulnerability scoring system prediction based on open source intelligence information sources. arXiv preprint arXiv:2210.02143.

Landis, J.R., Koch, G.G., 1977. The measurement of observer agreement for categorical data. Biometrics 159–174.

Le, T.H.M., Babar, M.A., 2022. On the use of fine-grained vulnerable code statements for software vulnerability assessment models. In: IEEE/ACM 19th International Conference on Mining Software Repositories. MSR 2022, Pittsburgh, PA, USA, May 23-24, 2022, IEEE, pp. 621–633. http://dx.doi.org/10.1145/3524842.3528433.

Le, T.H., Chen, H., Babar, M.A., 2022. A survey on data-driven software vulnerability assessment and prioritization. ACM Comput. Surv. 55 (5), 1–39.

Le, T.H.M., Hin, D., Croft, R., Babar, M.A., 2021. DeepCVA: Automated commit-level vulnerability assessment with deep multi-task learning. In: 36th IEEE/ACM International Conference on Automated Software Engineering. ASE 2021, Melbourne, Australia, November 15-19, 2021, IEEE, pp. 717–729. http://dx.doi.org/10.1109/ASE51524.2021.9678622.

Le, T.H.M., Sabir, B., Babar, M.A., 2019. Automated software vulnerability assessment with concept drift. In: Storey, M.D., Adams, B., Haiduc, S. (Eds.), Proceedings of the 16th International Conference on Mining Software Repositories. MSR 2019, 26-27 May 2019, Montreal, Canada, IEEE / ACM, pp. 371–382. http://dx.doi.org/10.1109/MSR.2019.00063.

Lee, J., Cho, K., Hofmann, T., 2017. Fully character-level neural machine translation without explicit segmentation. Trans. Assoc. Comput. Linguist. 5, 365–378. http://dx.doi.org/10.1162/tacl_a_00067.

Li, W., Du, Y., Li, X., Chen, X., Xie, C., Li, H., Li, X., 2022. UD_BBC: Named entity recognition in social network combined BERT-BiLSTM-CRF with active learning. Eng. Appl. Artif. Intell. 116, 105460. http://dx.doi.org/10.1016/j.engappai.2022.105460.

Li, X., Feng, J., Meng, Y., Han, Q., Wu, F., Li, J., 2020. A unified MRC framework for named entity recognition. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J.R. (Eds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. ACL 2020, Online, July 5-10, 2020, Association for Computational Linguistics, pp. 5849–5859. http://dx.doi.org/10.18653/v1/2020.acl-main.519.

Lomio, F., Iannone, E., Lucia, A.D., Palomba, F., Lenarduzzi, V., 2022. Just-in-time software vulnerability detection: Are we there yet? J. Syst. Softw. 188, 111283. http://dx.doi.org/10.1016/j.jss.2022.111283.

Lovins, J.B., 1968. Development of a stemming algorithm.. Mech. Transl. Comput. Linguist. 11 (1–2), 22–31.

Luong, T., Pham, H., Manning, C.D., 2015. Effective approaches to attention-based neural machine translation. In: Màrquez, L., Callison-Burch, C., Su, J., Pighin, D., Marton, Y. (Eds.), Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. EMNLP 2015, Lisbon, Portugal, September 17-21, 2015, The Association for Computational Linguistics, pp. 1412–1421. http://dx.doi.org/10.18653/v1/d15-1166.

Martinez, A., Sudoh, K., Matsumoto, Y., 2021. Sub-subword N-gram features for subword-level neural machine translation. J. Nat. Lang. Process. 28 (1), 82–103.

Olah, C., 2015. Understanding lstm networks.

Plisson, J., Lavrac, N., Mladenic, D., et al., 2004. A rule based approach to word lemmatization. In: Proceedings of IS, Vol. 3. pp. 83–86.

Reimers, N., Gurevych, I., 2017. Optimal hyperparameters for deep LSTM-networks for sequence labeling tasks, CoRR abs/1707.06799. URL: http://arxiv.org/abs/1707.06799.

Sahin, S.E., Tosun, A., 2019. A conceptual replication on predicting the severity of software vulnerabilities. In: Proceedings of the Evaluation and Assessment on Software Engineering. pp. 244–250.

Sanh, V., Debut, L., Chaumond, J., Wolf, T., 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108.

Shahid, M.R., Debar, H., 2021. CVSS-BERT: Explainable natural language processing to determine the severity of a computer security vulnerability from its description. In: Wani, M.A., Sethi, I.K., Shi, W., Qu, G., Raicu, D.S., Jin, R. (Eds.), 20th IEEE International Conference on Machine Learning and Applications. ICMLA 2021, Pasadena, CA, USA, December 13-16, 2021, IEEE, pp. 1600–1607. http://dx.doi.org/10.1109/ICMLA52953.2021.00256.

Srinivasa, S., Pedersen, J.M., Vasilomanolakis, E., 2022. Deceptive directories and "vulnerable" logs: a honeypot study of the LDAP and log4j attack landscape. In: IEEE European Symposium on Security and Privacy, EuroS&P 2022 - Workshops. Genoa, Italy, June 6-10, 2022, IEEE, pp. 442–447. http://dx.doi.org/10.1109/EuroSPW55150.2022.00052.

Stavrianou, A., Andritsos, P., Nicoloyannis, N., 2007. Overview and semantic issues of text mining. SIGMOD Rec. 36 (3), 23–34. http://dx.doi.org/10.1145/1324185.1324190.

Sun, X., Li, L., Bo, L., Wu, X., Wei, Y., Li, B., 2022. Automatic software vulnerability classification by extracting vulnerability triggers. Journal of Software: Evolution and Process e2508.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2017. Attention is all you need. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017. December 4-9, 2017, Long Beach, CA, USA, pp. 5998–6008, URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E., 2016. Hierarchical attention networks for document classification. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 1480–1489.

Yin, W., Kann, K., Yu, M., Schütze, H., 2017. Comparative study of CNN and RNN for natural language processing, CoRR abs/1702.01923. URL: http://arxiv.org/abs/1702.01923.

Yitagesu, S., Xing, Z., Zhang, X., Feng, Z., Li, X., Han, L., 2021a. Unsupervised labeling and extraction of phrase-based concepts in vulnerability descriptions. In: 36th IEEE/ACM International Conference on Automated Software Engineering. ASE 2021, Melbourne, Australia, November 15-19, 2021, IEEE, pp. 943–954. http://dx.doi.org/10.1109/ASE51524.2021.9678638.

Yitagesu, S., Zhang, X., Feng, Z., Li, X., Xing, Z., 2021b. Automatic part-of-speech tagging for security vulnerability descriptions. In: 18th IEEE/ACM International Conference on Mining Software Repositories. MSR 2021, Madrid, Spain, May 17-19, 2021, IEEE, pp. 29–40. http://dx.doi.org/10.1109/MSR52588.2021.00016.

Zhang, D., Wang, D., 2016. Relation classification: CNN or RNN? In: Lin, C., Xue, N., Zhao, D., Huang, X., Feng, Y. (Eds.), Natural Language Understanding and Intelligent Applications - 5th CCF Conference on Natural Language Processing and Chinese Computing, NLPCC 2016, and 24th International Conference on Computer Processing of Oriental Languages, ICCPOL 2016, Kunming, China, December 2-6, 2016, Proceedings. In: Lecture Notes in Computer Science, vol. 10102, Springer, pp. 665–675. http://dx.doi.org/10.1007/978-3-319-50496-4_60.

Xiaobing Sun is currently a professor in School of Information Engineering, Yangzhou University, China. He received the Ph.D. degree in School of computer science and engineering, Southeast University in 2012. His research interests include intelligent software engineering and software data analytics.

Zhenlei Ye is working towards the master degree in the School of Information Engineering, Yangzhou University, China. His research interests include vulnerability assessment based on deep learning.

Lili Bo is currently an associate professor in School of Information Engineering, Yangzhou University, China. She received the Ph.D. degree in School of Computer Science and Technology, China University of Mining and Technology in 2019. Her research interests include software testing, software security.

Xiaoxue Wu is currently a lecturer in School of Information Engineering, Yangzhou University, China. She received the Ph.D. degree in Northwestern Polytechnical University in 2021. Her research interests include software testing, software security.

Ying Wei is working towards the Ph.D. degree in the School of Information Engineering, Yangzhou University, China. Her research interests include vulnerability diagnose based on deep learning.

Tao Zhang is currently an associate professor in School of Computer Science and Engineering, Macau University of Science and Technology (MUST), Macao SAR, China. He received the BS degree in automation, the M.Eng. degree in software engineering from Northeastern University, China, and the Ph.D. degree in computer science from the University of Seoul, South Korea. After that, he spent one year with the Hong Kong Polytechnic University as a postdoctoral research fellow. Before joining MUST, he was the faculty member of Harbin Engineering University and Nanjing University of Posts and Telecommunications, China. His research interests include AI for software engineering and mobile software security.

Bin Li is a professor in School of Information Engineering, Yangzhou University, China. His current research interests include software engineering, artificial intelligence.