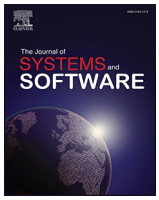
The Journal of Systems and Software 213 (2024) 112039

Contents lists available at ScienceDirect

The Journal of Systems & Software

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



A vulnerability severity prediction method based on bimodal data and multi-task learning

Xiaozhi Du *, Shiming Zhang, Yanrong Zhou, Hongyuan Du

School of Software Engineering, Xi'an Jiao Tong University, Xi'an Shaanxi, 710049, China

ARTICLE INFO

Editor: Professor Yan Cai

Keywords:
Vulnerability severity prediction
Bimodal data
Multi-task learning
GraphCodeBert

ABSTRACT

Facing the increasing number of software vulnerabilities, the automatic analysis of vulnerabilities has become an important task in the field of software security. However, the existing severity prediction methods are mainly based on vulnerability descriptions and ignore the relevant features of vulnerability code, which only includes unimodal information and result in low prediction accuracy. This paper proposes a vulnerability severity prediction method based on bimodal data and multi-task learning. First the bimodal data, which consists of the description and source code of each vulnerability, is preprocessed. Next the GraphCodeBert is used for the word embedding module to extract different vulnerability features from the bimodal data. Then the Bi-GRU with attention mechanism is adopted for further feature extraction of vulnerability severity. Considering the strong correlation between the two tasks of vulnerability severity prediction and exploitability prediction, this paper proposes a multi-task learning approach, which allows the model to learn the connection and shared information between different tasks through a hard parameter sharing strategy, so as to achieve more accurate and reliable prediction of vulnerability severity. Experimental results show that the severity prediction method proposed in this paper outperforms state-of-the-art methods, and can achieve an average F1 score of 93.83 % on the public vulnerability dataset.

1. Introduction

The continuous development of software has led to an increasingly prominent issue of software vulnerabilities. Each year, a large number of new vulnerabilities arise, and yet software vendors and security experts are unable to promptly address all of them (Soumyadeep et al., 2023). As a result, therefore, they need to pay more attention to the vulnerabilities with higher severity and prioritize addressing these vulnerabilities. In general, severity is an important characteristic of vulnerabilities, which indicates the harm degree of software vulnerabilities. Vulnerability severity level refers to the classification of the harm degree of vulnerabilities, which is usually divided into four levels: Low, Medium, High, and Critical (Napier et al., 2023). Each level corresponds to a different degree of vulnerability harm. For example, a low-level vulnerability may be only some minor technical problems, while a serious-level vulnerability may lead to a complete system crash or leak a lot of sensitive information. By assessing the severity of a vulnerability, software developers can allocate resources more effectively so that more serious vulnerabilities can be fixed as soon as possible. After a vulnerability is released, security experts will score the vulnerability based on a series of complex calculations in the Common Vulnerability Scoring System (CVSS) before assigning a severity rating to the vulnerability (Chen et al., 2023), which is inefficient for so many vulnerabilities exist in software and results that vulnerability severity assessments lag too far behind the release of the vulnerability, leading to the problem of "zero-day attacks" (Luo et al., 2022). Therefore, the use of deep learning techniques to predict vulnerability severity in a timely and accurate manner is important for vulnerability remediation efforts.

Vulnerability severity prediction is an estimation and classification of the level of vulnerability harm that helps security teams and developers to identify and address vulnerabilities quickly and accurately. By using deep learning techniques to deeply analyze and mine semantic information in vulnerability descriptions, and extracting features related to their severity, the severity of vulnerabilities can be effectively predicted. For example, the model based on convolutional neural network (CNN) can automatically extract keywords, entities, attributes and their relationships from vulnerability descriptions, and then construct the corresponding vulnerability feature vectors. The existing severity

* Corresponding author.
E-mail address: xzdu@xjtu.edu.cn (X. Du).

https://doi.org/10.1016/j.jss.2024.112039
Received 9 July 2023; Received in revised form 27 December 2023; Accepted 26 March 2024
Available online 27 March 2024

prediction methods are mainly based on the vulnerability description text (Han et al., 2017; Zeng et al., 2022; Jiang and Atif, 2022). However, vulnerability descriptions are generally relatively short and contain limited information, so information related to severity may not be sufficient and prediction results may not be accurate. Vulnerability source code is another manifestation of vulnerability that contains the severity characteristics of the vulnerability and can be used for severity prediction. Vulnerability description and code contain information about the severity of the vulnerability, therefore, combining these two types of data can improve the accuracy of vulnerability prediction.

In general, severity and exploitability are two key reference indicators of software vulnerabilities. Vulnerability exploitability prediction is to predict whether an attacker can successfully exploit a software vulnerability to compromise a system. It is another important task in the field of vulnerability analysis to help software security personnel better understand the likelihood of a vulnerability being exploited so that they can assess the level of danger of the vulnerability and the defensive measures that need to be taken. Compared with single-task learning, multi-task learning helps security professionals better secure their systems by combining multiple tasks that are closely related and sharing information learned from different tasks to obtain better generalization and further improve the accuracy of prediction. Through multi-task learning, the model can combine these two tasks and learn how to predict both vulnerability severity and exploitability at the same time.

In this paper, we propose a vulnerability severity prediction method based on bimodal data and multi-task learning. First, the bimodal data consisting of the description and source code of each vulnerability is preprocessed to improve the quality of the dataset. Next, different vulnerability features are extracted from the bimodal data using GraphCodeBert (Guo et al., 2020) in the word embedding module. Then, vulnerability severity features are further extracted using Bi-GRU (Chung et al., 2014) with attention mechanism. Finally, due to the strong correlation between vulnerability severity prediction and exploitability prediction, we propose a multi-task learning approach to achieve more accurate and reliable vulnerability severity prediction through a hard parameter sharing strategy. The main contributions of this paper are as follows:

- We propose a new vulnerability severity prediction method, which combines the description and source code of each vulnerability, to address the limitations of existing methods that use only vulnerability description.
- The GraphCodeBert (Guo et al., 2020) model is used for the first time in a word embedding module for severity prediction to improve the deep integration of bimodal features of vulnerability descriptions and code to obtain richer information on vulnerability severity.
- A multi-task learning strategy is introduced to learn a wider range of knowledge by combining vulnerability exploitability prediction tasks, which improves the generalization ability of the model and achieves more accurate prediction of vulnerability severity.
- Compared with the state-of-the-art methods, our proposed method has better prediction results and can achieve an average F1 score of 93.83 % on the public vulnerability dataset.

The rest of this paper is organized as follows. In Section 2, we review the related work on software severity studies in recent years. Section 3 provides a detailed description of the method proposed in this paper. In Section 4, we take some experiments to evaluate the effectiveness of our method. Section 5 summarizes some of the limitations of this method. Section 6 summarizes the conclusion.

2. Related works

Automated analysis of vulnerabilities is a hot topic in the field of software security, such as vulnerability detection and vulnerability severity prediction. The use of deep learning techniques has greatly improved the efficiency of vulnerability analysis.

Among the deep learning-based vulnerability detection methods, they can be mainly divided into code sequence, text, abstract syntax tree and graph-based methods according to the way the code is embedded (Zeng et al., 2022). Soumyadeep et al. (2023) proposed a novel framework, consisting of a deep reinforcement learning agent and an integer programming method to determine the optimal amount of resources and the optimal set of prioritized vulnerability instances. Napier et al. (2023) constructed a method based on extracted source code functions and compared the effects of different machine learning models, natural language processing (NLP) techniques, and data processing methods. Chen et al. (2023) proposed a vulnerability model based on Petri nets, modeling both the subject and environment of the vulnerability. Jiang and Atif (2022) proposed an artificial intelligence-based approach to simplify the computation of vulnerability severity, reducing the error rate caused by the manual computation process traditionally used for cybersecurity analysis. Li et al. (2021) proposed a hybrid neural network framework for source code vulnerability detection, which learns local and global features of vulnerabilities through convolutional and recurrent neural networks, respectively. Zou et al. (2021) designed and implemented a new vulnerability detection system, uVulDeePecker, which has improved detection results compared to VulDeePecker (Li et al., 2018). uVulDeePecker extracts code fragments based on their control dependencies and data dependencies, and defines them as code gadget, using building-block BLSTM network to achieve vulnerability detection. Zhou et al. (2019) proposed a vulnerability detection framework called Devign. It is based on the joint graph of abstract syntax tree of code, data flow graph and control flow graph for code tableau evidence, and uses gated graph neural network (Liu et al., 2023) to deeply mine the features of code graph. FUNDED is a vulnerability detection framework proposed by Wang et al. (Wang et al., 2021), which uses AST for intermediate representation of code, effectively extracts features of the code graph, and enables cross-language vulnerability detection. Chakraborty et al. (2022) proposed a new vulnerability dataset 'Reveal' with an intermediate representation of the code using code property graph. Cheng et al. (2021) proposed DeepWukong vulnerability detection model, which constructs program dependency graph (PDG) for source code and extracts code fragments related to API function call vulnerabilities, and finally uses graph neural networks k-GNNs for vulnerability detection, which improves the accuracy of vulnerability detection.

Vulnerability severity prediction is a hot topic in vulnerability prediction tasks, and deep learning techniques are used to mine features in vulnerability data to achieve automated prediction of vulnerability severity. Luo et al. (2022) proposed Compact Abstract Graphs (CAGs) of source code in different programming languages for predicting a broad range of code vulnerabilities with Graph Neural Network (GNN) models. Han et al. (2017) used word embedding and convolutional neural networks to feature extract vulnerability descriptions, which enabled the model to better understand the semantic information in vulnerability descriptions and improve the prediction effect. Wang et al. (2019) used Text mining to extract keywords, feature extraction of vulnerability descriptions using principal component analysis, and then severity prediction of vulnerabilities related to cross-site scripting attacks by XGBoost algorithm. Liu et al. (2019) used CNN, LSTM (Graves, 2012), TextRCNN (Lu et al., 2023) and XGBoost to predict the severity of vulnerabilities related to cross-site scripting attacks, respectively, with the best prediction on TextRCNN (Lu et al., 2023). Sahin and Sahin (2019) explored the impact of different classification models on the performance of severity prediction models, and showed that LSTM (Graves, 2012) has better performance through comparative experiments. Kudjo et al. (2020) proposed a Bellwether method for severity prediction by using Bellvul algorithm to sample and train the vulnerability data on different neural network models to obtain higher prediction accuracy. Chen et al. (2020) improved the existing word frequency statistics method and proposed a new statistical method TF-IGM to predict

vulnerability severity by vulnerability reports into two categories, severe and not severe. Sharma et al. (2021) used Glove word embedding model for vector representation of vulnerability text and classify it using convolutional neural network. Nakagawa et al. (2019) used Character-level CNN for severity prediction. Compared to CNN, charCNN can extract more features using character-level information and can be more efficient in processing short texts like vulnerability descriptions. Malhotra (2021) explored the impact of various word embedding methods on the performance of severity prediction models by comparing experiments on five different vulnerability datasets of Mozilla products and analyzed the classification performance of different word embedding methods. Jabeen et al. (2021) proposed a software vulnerability prediction method based on Markov chain model, which provides a more comprehensive description model with the potential to accurately predict the severity of vulnerabilities. Ni et al. (2022) added vulnerability-related features to the processing of vulnerability descriptions, fine-tuned on the Bert pre-training model, and used CNN to further extract textual features. Aivatoglou et al. (2022) used a multi-label classification algorithm Random k-Labelsets to achieve the prediction of vulnerability features and scores, by scoring the vulnerability the severity features of the vulnerability can be further obtained. Malhotra and Vidushi (2023) used vulnerability descriptions for severity prediction and explored the effect of different word embedding models on the prediction effect. Hao et al. (2023) use vulnerability code for severity prediction. The prediction is performed by extracting the severity features of the vulnerability property graph and function call graph.

However, the current severity prediction methods are only based on vulnerability description texts or code. These methods have some limitations. The vulnerability description text is usually short and has limited information, which may not fully reflect the severity information of the vulnerability, thus limiting the prediction effect. The vulnerability code, on the other hand, is another manifestation of the vulnerability and contains the severity characteristics of the vulnerability, which can be used for severity prediction. Dong et al. (2023) proposed a multi-type vulnerability classifier that combines vulnerability descriptions and source codes, which uses TextRCNN (Lu et al., 2023) and RGAT to extract vulnerability descriptions and code features respectively. However, it does not retain the positional information between codes well, which can lead to the loss of contextual relationships.

Therefore, we propose a severity prediction method that combines vulnerability descriptions and codes to extract vulnerability severity features for bimodal data using GraphCodeBert (Guo et al., 2020) and Bi-GRU (Chung et al., 2014) with added attention mechanism. And multi-task learning based on hard parameter sharing is used so that our model can predict vulnerabilities more accurately.

3. Proposed method

In this paper, we use vulnerability bimodal data for severity prediction and use a multi-task learning strategy to further improve the prediction accuracy.

3.1. Overview of method

The general framework of the method is shown in Fig. 1, which is divided into four modules: vulnerable bimodal data pre-processing, bimodal data word embedding, bimodal data feature extraction and multi-task learning. The first module is the pre-processing of vulnerability data. For the vulnerability description, the text is denoised by tag replacement; for the vulnerability source code, it is modeled to form Abstract Syntax Trees (AST), and the data flow graph is finally formed by extracting the variable relationships in the AST. The data flow diagram can clearly show the interdependencies between variables, which provides important information for a deeper understanding of the program code. Then we use the GraphCodeBert (Guo et al., 2020) model in the word embedding module to improve the deep fusion of the bimodal features of vulnerability description and code, and the Bi-GRU (Chung et al., 2014) with attention mechanism (Att-BiGRU (Li et al., 2022)) model is used for the fused features for further feature extraction. The downstream task based on vulnerability description also has vulnerability exploitability prediction, and these two tasks are strongly correlated, and there may be learnable hidden knowledge between them. In order to make full use of the connection and shared information between these two downstream tasks and improve the model performance, this paper uses a hard parameter sharing multi-task learning technique to learn the features of vulnerability data.

3.2. Vulnerable bimodal data pre-processing module

The textual information in vulnerability reports includes information such as vulnerability descriptions, CVSS features, etc. Currently, existing methods use vulnerability descriptions to predict vulnerability severity, while usually vulnerability descriptions are short and contain limited information, resulting in limited features for model learning. In this section, we aim to break the limitations of severity prediction by vulnerability descriptions alone by combining vulnerability descriptions with code to predict the severity of vulnerabilities more

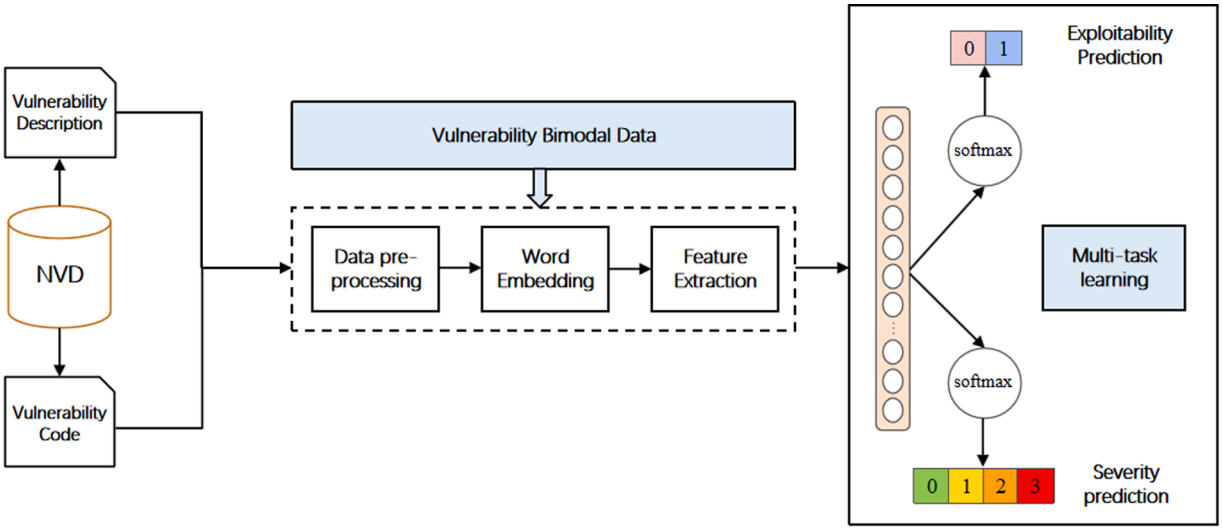


Fig. 1. Vulnerability severity prediction framework.

comprehensively and accurately. The method we use to extract useful information from the text of vulnerability descriptions and code is derived from NLP (Wu, 2021; Ziems and Wu, 2021), which converts natural language into something computers can understand.

In some vulnerability descriptions, some redundant information such as URLs and related code may be included. The meaning of this information may need to be interpreted from third-party sources or additional knowledge. Therefore, if this information is used to predict the severity of a vulnerability, it is not possible to achieve the desired effect of noise. In order to effectively mine the information related to vulnerabilities, this chapter takes a series of measures to preprocess the vulnerability description text and use the corresponding tags such as {code} and {url} to replace the redundant content. For example, the vulnerability description 'The following code results in a compile time error: public void Test(){code}. The method geColumnCount() is undefined for the type.javax.swing.table.AbstractTableModel', after preprocessing it becomes 'the following code result compile time error build public void test code the method gecolumncount undefined type javax swing table', where the word 'code' refers to the replacement of the code part of the vulnerability description. In this way, the density of textual information can be increased, the noise in the vulnerability description can be eliminated, and the information that cannot be recognized by the model can be significantly reduced, thus increasing the density of information related to severity. When dealing with vulnerability code, we find that their structured information is different from the syntax and semantics of traditional natural language. They have some structured information, usually presented as trees or graphs, so that the code can be graphically represented. Data Flow Graph (DFG) is a widely used tool for program analysis, which can clearly show the association between variables. In a Data Flow Graph, nodes are used to describe the variables in the code, while edges are used to indicate the interrelationships between these variables. This code graph structure provides key code semantic information for code understanding and also allows the model to learn the remote dependencies induced between variables or functions. The flow of data flow extraction is shown in Fig. 2.

For the given source code $C = \{c_1, c_2, \ldots, c_n\}$, it is first converted into an abstract syntax tree (AST). The AST contains the syntax information of the code, and the leaf nodes in the AST are used to identify sequences of variables, represented as $V = \{v_1, v_2, \ldots, v_m\}$.

Identify the sequence of variants in it and extract the variable relationships from it if there is a transfer of values between v_i, v_j, then create an edge $\varepsilon = \langle v_i, v_j \rangle$. The specific steps are as follows.

1) If the value of a variant node v_1 comes from its ancestor node v_2, create an edge $v_2 \rightarrow v_1$ that represents the data flow relationship from v_2 to v_1;
2) If the value of a variant node v_1 comes from its children node v_2 and v_3, the relations pointing from v_2 to v_1 and from v_3 to v_1 are obtained, respectively;
3) If variant v_1 and v_2 are two child nodes of the operator node, two independent directed graphs are obtained and therefore no relationship is established;
4) If variant v_1 and v_2 are the left and right children of an assignment operator node or assignment function node, the relationship from to v_1 is obtained in the middle and reverse order, respectively;

The above steps lead to the construction of a function-based data flow graph, in which the nodes and directed edges are represented by sets and collections, respectively, and the elements in the collections are the data flow lexical elements.

3.3. Bimodal data word embedding module

In order to fully learn the severity information in the vulnerability bimodal data, this section conducts word embedding processing on the bimodal data.

In this section, we take bimodal data such as CVE vulnerability descriptions and their associated source code to perform vulnerability severity prediction tasks. For the processing of bimodal data, we use GraphCodeBert (Guo et al., 2020), a pre-trained model based on natural language and code data streams, to perform word embedding. In natural language processing tasks, pre-trained models facilitate a variety of software engineering tasks (Hassan et al., 2023; Suzuki et al., 2023; Han

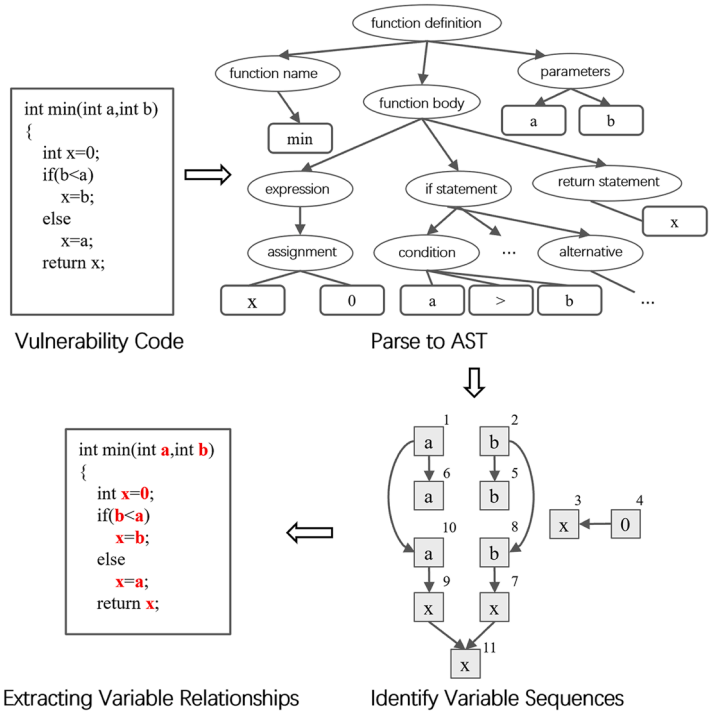


Fig. 2. The code data flow extraction process.

et al., 2023; Hadi and Fard, 2023). The pre-trained model can mine the potential vulnerability severity features of both and fuse them in a deeper way.

The input of the GraphCodeBert (Guo et al., 2020) model consists of three parts, which are vulnerability description text, vulnerability code token sequence and code data stream sequence. The vulnerability description text in the input part of the model can be defined as $T = \{t_1, t_2,...,t_i\}$, the vulnerability code is defined as $C = \{c_1,c_2,...,c_n\}$. Firstly, get the data flow diagram of the code by using the method described in the previous subsection $G(C) = (V,E)$. $V = \{v_1, v_2, ..., v_m\}$ represents the set of code variant sequence, $E = \{e_1, e_2, ..., e_n\}$ represents the set of data flow edge. Next, concatenate the preprocessed vulnerability description text, code, and variable sequences as the input of the model $X = \{[CLS], T, [SEP], C, [SEP], V\}$. $[CLS]$ is a special marker in front of three inputs, $[SEP]$ is a separator used to separate different types of input data. GraphCodeBert (Guo et al., 2020) converts the input sequence X into an input vector H^0. For each input token, a combination of token embedding and position embedding is used for encoding, and the position embedding also reflects the position information of the variable within the variable sequence, which also corresponds to different nodes in the data stream. model uses 12 identical transformer (Vaswani et al., 2017) layers to build a core network architecture. Let the output of transformer layer n be H^n, which is represented as shown in Eq. (1).

$$H^n = transformer\left(H^{n-1}\right),\ n \in [1, 12] \tag{1}$$

where the output of the transformer network for each layer is shown in Eq. (2).

$$\begin{aligned} G^n &= LN\left(MultiAtn\left(H^{n-1}\right) + H^{n-1}\right) \\ H^n &= LN(FFN(G^n) + G^n) \end{aligned} \tag{2}$$

whereLN is normalization operation, $MultiAtn$ is multi-headed self-attentive mechanism, FFN is feedforward network. The output G^n of the multi-headed self-concentration can be calculated by Eq. (3).

$$\begin{aligned} G^n &= W_n \cdot [head_1; \ldots; head_u] \\ head_i &= softmax\left(\frac{Q_i \cdot K_i^T}{\sqrt{d_k}} + M\right) \cdot V_i \\ Q_i = H^{n-1}W_i^Q, K_i &= H^{n-1}W_i^K, V_i = H^{n-1}W_i^V \end{aligned} \tag{3}$$

where W_n, W_i^Q, W_i^K, W_i^V are model parameters, $head$ is attention head, u is the number of attention heads, d_k is dimension of attentional head, M is graph-guided masked attention matrix. The structure of Bimodal data word embedding module is shown in Fig. 3.

By using the GraphCodeBert (Guo et al., 2020) word embedding model, we obtained a more accurate word embedding vector representation of bimodal data, which helps to analyze the vulnerability description and code effectively, and in the subsequent feature extraction stage, we can mine the semantic association between the two and extract deeper features to provide strong support for the next vulnerability severity work.

3.4. Bimodal data feature extraction module

The previous subsection passed to use GraphCodeBert (Guo et al., 2020) for effective word embedding of bimodal data and deep fusion of features of both. Further feature extraction is then required to capture more vulnerability information. To ensure the integrity of the vulnerability description and the internal structure of the code, a Bi-GRU (Chung et al., 2014) model is used for feature extraction, which extracts outgoing sequence features and statement-level features from the word embedding vector of the bimodal data, and adds an attention mechanism to optimize the Bi-GRU (Chung et al., 2014) model, enabling us to predict the severity of the vulnerability more accurately. Based on the Bi-GRU (Chung et al., 2014) feature extraction model, we can combine the forward and backward states of bimodal data to form the final vector representation, and make the model run more effectively by introducing the sub-attention mechanism. By combining different sequence states, a probability distribution, i.e., attention weights, can be computed on the input tokens, which indicates the degree of contribution of each token to the final prediction of the model, and then weighted and summed separately to obtain a final vector representation. In the Att-BiGRU (Li et al., 2022) feature extraction model, Bi-GRU (Chung et al., 2014) can not only effectively handle sequential data like vulnerability descriptions and code, but also learn more comprehensive features like contextual information of bimodal data. With the addition of the attention mechanism in it, the key features of the current prediction task can be better focused, allowing the model to give more attention to the key information useful for the current prediction task, retaining the useful key information and avoiding the interference of redundant information in order to complete the classification task more effectively. The whole Att-BiGRU (Li et al., 2022) model consists of an input layer, a forward hidden layer, a backward hidden layer and an Attention layer. The structure of bimodal data feature extraction module is shown in Fig. 4.

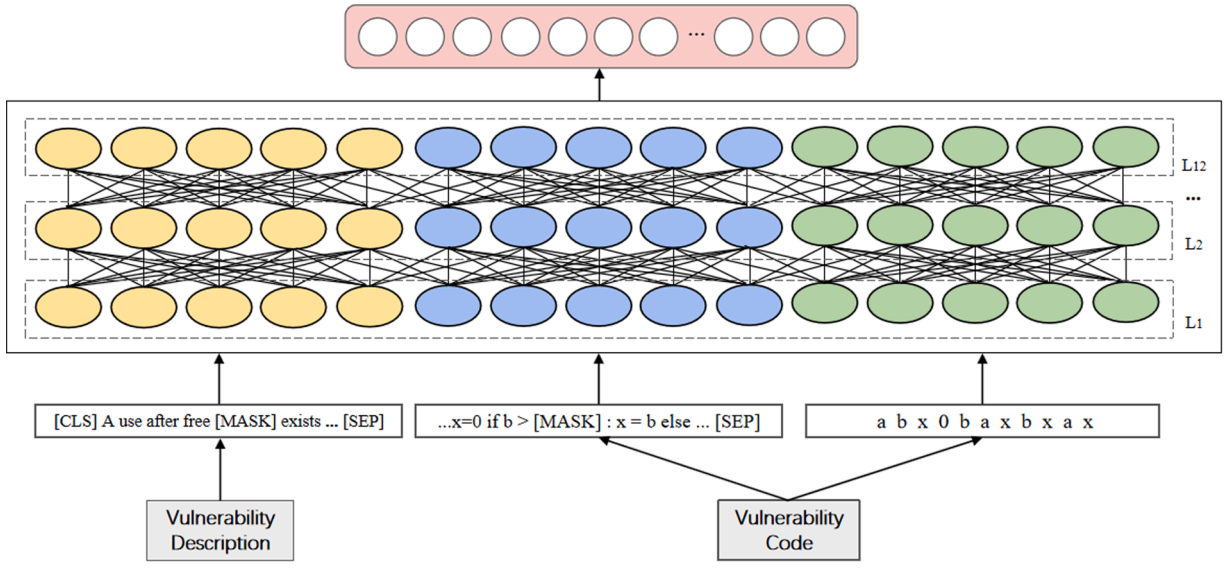


Fig. 3. Bimodal data word embedding module.

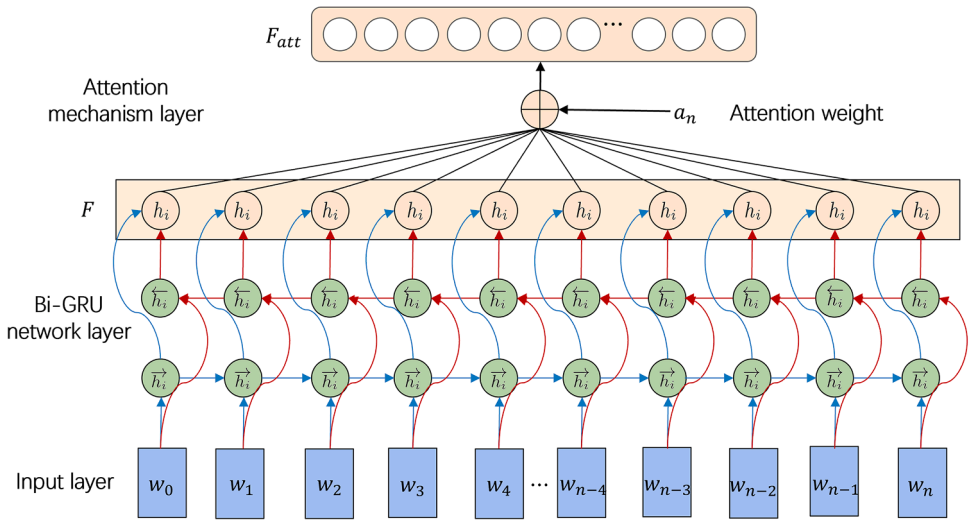


Fig. 4. Bimodal data feature extraction module.

In Fig. 4, $w_0, w_1, w_2, \ldots, w_n$ is the input after GraphCodeBert (Guo et al., 2020) to the quantized representation, F is the final state of Bi-GRU (Chung et al., 2014), a_n is the attention weight, and its calculation process is shown in Eq. (4). After F is weighted by the attention mechanism, the feature vector F_{att} of the final output of the Att-BiGRU (Li et al., 2022) model is obtained.

$$a_n = \frac{\exp(h_n')}{\sum_{i=0}^{N}\exp(h_i')} \qquad h_n' = h_n{}^T W F \tag{4}$$

where N is the number of model input vectors, W is weighting matrix, F is model final state, obtained by combining the forward and reverse final states of the hidden layers of the model, h_n is the splicing of forward state $\overrightarrow{h_n}$ and reverse state $\overleftarrow{h_n}$ at moment n. $h_n$is calculated as shown in Eq. (5).

$$h_n = \overrightarrow{h_n} \oplus \overleftarrow{h_n} \tag{5}$$

The eigenvector F_{att} of the final output of the Att-BiGRU (Li et al., 2022) model is calculated as shown in Eq. (6).

$$F_{att} = \sum_{n=0}^{N} a_n h_n \tag{6}$$

3.5. Multi-task learning module

After the word embedding processing of the bimodal data by GraphCodeBert (Guo et al., 2020) in the previous subsection and the feature extraction by Att-BiGRU (Li et al., 2022), we extracted a large number of vulnerability features, and then by using multi-task learning, the model can better understand and identify vulnerabilities.

Specifically, by using multi-task learning techniques, we divide the vulnerability severity prediction process into two steps: task assignment and model training. In the task assignment phase, the vulnerability severity prediction task and the vulnerability exploitability prediction task need to be assigned to different network layers. In this section, our focus is on how to predict the severity of software vulnerabilities, so we use it as the main task and vulnerability availability prediction as a secondary task. In the model training phase, a multi-task learning approach is required to train the network model. Vulnerability severity feature extraction is performed by using a hard shared parameter strategy, and then softmax is constructed for each task for classification. The hard parameter sharing for multi-task learning is shown in Fig. 5. In this way, we can use the shared features to improve the generalization ability and efficiency of the model, while using task-specific features to improve the vulnerability severity prediction.

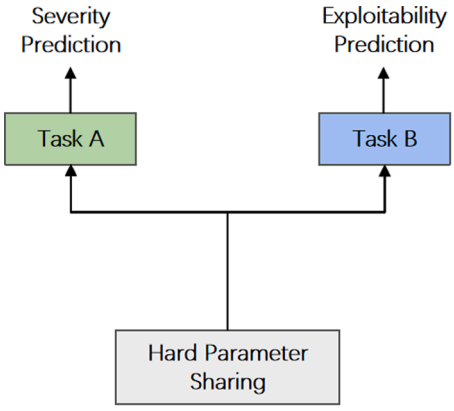


Fig. 5. Hard parameter sharing for multi-task learning.

This method can effectively train the model by using the strategy of hard parameter sharing in multi-task learning, where vulnerability feature learning parameters are shared between two subtasks, and then train the classifier for two subtasks of vulnerability severity prediction and vulnerability exploitability prediction, so as to achieve effective evaluation of vulnerabilities, whose prediction process is shown in Equation (7).

$$L_{mtl} = \sum_{k=1}^{K} L^k(y^k, \hat{y}^k) \qquad L^k(y^k, \hat{y}^k) = -\sum_{i=1}^{N}\sum_{j=1}^{m} y_{ij}^k log\left(\hat{y}_{ij}^k\right) \tag{7}$$

where K is number of classifiers, $L^k(y^k, \hat{y}^k)$ is cross-entropy function, N is number of samples, m is number of labels, y_{ij}^k is true label of the $i-th$ sample on the $j-th$ class, $\hat{y}_{ij}^k$ is the probability that $i-th$ training sample is predicted to be $j-th$ class.

In addition, in multi-task learning, vulnerability severity prediction and availability prediction both have different complexity, and their training schedules can be different. If left unchecked, this may lead to problems such as training bias and overfitting as one task is trained while the other is not yet fully trained. In this case, the setting of weights

is very important because it can affect the performance of the model on different tasks. A proper weight setting can also improve the performance and effectiveness of the model, making it perform more balanced and better on different tasks. Therefore, in this paper, different weights are assigned to the losses of different tasks to control the magnitude of the losses in the process of joint multi-task training. In this paper, the static weight setting method of grid search is used. For the main task of vulnerability prediction, the loss weight is set to 1, and for the auxiliary task, the loss weight of exploitability prediction is searched in steps of 0.1 between 0 and 1 to select the appropriate weight value. The final weight of main task and auxiliary task are set to 1 and 0.4, respectively.

4. Experiment

All experiments were run on Google Colab for Linux with a NVIDIA Tesla K80 GPU. And the method proposed in this paper was written in python 2.7, and the models were trained and tested using Tensorflow 1.5 and CuDNN 7.1.4, and the implementation code is available in our Github project (Du et al., 2023).

4.1. Data set

For vulnerability description and vulnerability code data, this paper uses the Big-Vul dataset, which is a large dataset collected by Fan et al. (2020) from the open source Github project that contains a set of vulnerability information, such as vulnerability identifier (CVE-ID), severity level, and vulnerability description. CVE is the identifier of the vulnerability, and for each vulnerability, the vendor provides a unique ID and a concise report describing the information about the vulnerability. These vulnerabilities are typically stored in the National Vulnerability Database (NVD) or other databases. In addition, the Big-Vul dataset contains code associated with the vulnerabilities, which are extracted from 348 Github projects covering 91 different vulnerability types. We extract vulnerability descriptions and code from Big-Vul as the Bimodal-vul dataset for this paper. For vulnerability exploitability data, we crawl from NVD and ExploitDB (ExploitDB[EB/OL], 2022) to collect all vulnerability exploitability data released between 1999 and 2022. On ExploitDB, exploitable vulnerabilities are identified by CVE-ID/EDB-ID accordingly. The multi-task learning is performed with the Big-Vul dataset. Therefore, to mark the exploitable vulnerabilities in the Big-Vul dataset, they are linked with the ExploitDB vulnerability exploitability data by CVE-ID to find out the exploitable vulnerabilities in Big-Vul. The total dataset size for vulnerability descriptions is 158, 383 pieces of data, and the total dataset size for vulnerability code is 24, 138 sets of data, each of which contains multiple function codes, and the specific datasets can be found in our Github project (Du et al., 2023).

4.2. Experimental setup

1) Evaluation metrics

This paper adopts four evaluation indicators, which are accuracy, precision, recall, and F1 score. The formula for calculating the metrics is shown in Eq. (8).

$$\begin{aligned} ACC &= \frac{TP+TN}{TP+TN+FP+FN} \\ P &= \frac{TP}{TP+FP} \\ R &= \frac{TP}{TP+FN} \\ F1 &= \frac{2\times P\times R}{P+R} \end{aligned} \tag{8}$$

where *TP* is number of samples that belong to a certain severity level and are correctly predicted, *TN* is number of samples that do not fall into a certain severity class and are correctly predicted, *FP* is number of samples that do not belong to a severity level but are incorrectly predicted to be at that level, *FN* is number of samples that belong to a severity level but are incorrectly predicted as other levels.

2) Parameter settings

In this paper, the samples are randomly disrupted and then partitioned into a training set, a validation set and a test set according to the ratio of 8:1:1. The relevant parameters of the whole network are shown in Table 1.

4.3. Comparative experiments and analysis

To verify the effectiveness of the method in this paper, the following sets of experiments were conducted.

1) Comparison experiments with different methods

First, in order to investigate the improvement of the vulnerability severity prediction effect of this paper, we first compare it with other methods for testing. Current severity prediction methods are based on vulnerability descriptions alone, and ignore the vulnerability features that can be exploited in the vulnerability code. In this paper, we set up five sets of experiments, four of which are baseline models of current vulnerability severity prediction. In order to investigate the improvement of bimodal data for vulnerability severity prediction, we set up a set of experiments on the data set with only vulnerability code. The specific experimental setup is as follows.

(1) Unimodal data: Five sets of baseline models are used to compare with the methods in this paper, namely Sharma et al. (2021), Malhotra (2021), Ni et al. (2022), Malhotra and Vidushi (2023) and Hao et al. (2023). These methods use only vulnerability unimodal data for severity prediction.
(2) Bimodal data: The DeKeDVer proposed by Dong et al. (2023) is used to compare with the method of this paper. This paper using the combined vulnerability description and code severity prediction approach based on multi-task learning, the dataset uses the combined vulnerability description and code Bimodal-vul dataset. The results of the above groups of experiments are shown in Table 2.

From the above table of experimental results, it can be seen that among the methods for severity prediction by vulnerability description only, Bert-CNN has the highest F1 score and achieves the best results compared to the other three methods. This is due to the fact that the Bert model used in this method learns the widely used linguistic expressions through pre-training. The Bert model can learn the contextual information of the text in depth by pre-training a large amount of unlabeled data, so that it can better understand the meaning and semantic relationships in the vulnerability descriptions. The model is also used for downstream vulnerability severity prediction tasks by fine-tuning the

Table 1
Parameter settings.

Parameter	Value
max_len	512
learning rate	2e-5
Epochs	50
batch_size	64
Dropout	0.4
hidden_size	768
hidden_layer	2
Optimizer	Adam
Activation Function	RELU

Table 2
Results of different approaches.

Categories	Approaches	Acc (%)	P(%)	R(%)	F1 (%)
Based on vulnerability description	(Sharma et al., 2021)	84.56	86.82	81.50	84.07
	(Malhotra, 2021)	86.34	87.93	84.25	86.05
	(Ni et al., 2022)	90.69	89.84	91.75	90.79
	(Malhotra and Vidushi, 2023)	87.22	88.44	85.63	87.01
Based on vulnerability code	(Hao et al., 2023)	82.69	83.57	81.38	82.46
Based on vulnerability description and code	(Dong et al., 2023)	–	85.29	84.83	84.49
	Our approach	**93.88**	**94.54**	**93.13**	**93.83**

model to further improve its generalization capability and accuracy. Compared with the prediction by vulnerability code, the prediction by vulnerability description is more effective because the complexity of vulnerability description is much smaller than that of vulnerability code, which usually consists of only a few dozen words and contains a large number of nouns and degree words related to vulnerabilities, and the text is concise and clear, which makes them much less complex compared with vulnerability code, thus providing vulnerability severity prediction provides a great help. Compared with the method of Hao et al. (2023) which is a severity prediction model based on vulnerability code, the method in this paper improves the accuracy, precision, recall and F1 score by 13.53 %, 13.13 %, 14.44 % and 13.79 %, respectively. This is due to the fact that the vulnerability code itself has a special syntactic structure, and the experimental data are all from the code of some open source projects, whose complexity is relatively high, and the model does not easily extract vulnerability severity features. So the effect of using only vulnerability code for prediction is not very satisfactory, and its F1 score of prediction is the lowest.

Compared with existing severity prediction methods, the method in this paper has a better prediction effect, and the values of the four evaluation indexes are improved. Its prediction accuracy can reach 93.88 %, precision can reach 94.54 %, recall can reach 93.13 %, and F1 score can reach 93.83 %. Compared with the methods of Sharma et al. (2021), Malhotra (2021) and Malhotra and Vidushi (2023), three baseline models for severity prediction based on vulnerability description, the method in this paper improves 9.76 %, 7.78 % and 6.82 % in F1 score, respectively. Compared with the method of Dong et al. (2023), the method in this paper improves 11.05 % in F1 score. This is due to the fact that the method used in this paper allows better extraction of vulnerability descriptions and features of the code, and uses multi-task learning to improve the learning ability of the model. Since accuracy was not calculated for Dong et al. (Dong et al., 2023), it is not compared in this paper. Compared with the top-performing Bert-CNN method (Ni et al., 2022), this paper's method improves the F1 score by 3.04 %. This is due to the fact that the GraphCodeBert (Guo et al., 2020) used in this paper adopts the same Transformer-based core architecture as Bert, in which the multi-headed attention mechanism can capture the interconnections between different positions in the sequence and learn the global and contextual information of the vulnerability description. In addition, GraphCodeBert (Guo et al., 2020) adds a dataflow graph embedding of the code in the pre-training phase for better representation learning of the code structure. The word embedding of bimodal data by GraphCodeBert (Guo et al., 2020) allows the model to not only capture the severity features in the vulnerability descriptions in the downstream task, but also to learn the syntactic semantic information and structured features related to the severity in the vulnerability code. In the subsequent feature extraction stage, the Att-BiGRU (Li et al., 2022) model used can effectively handle not only sequential data such as vulnerability descriptions and code, but also learn more comprehensive features such as contextual information of bimodal data compared to CNNs used in other methods. By introducing an attention mechanism, the important information related to vulnerability severity is given more weight and considered in the model output during the learning process to mitigate the influence of irrelevant information, thus improving the robustness of the model. In addition, the method in this paper makes full use of the connection and shared information between two downstream tasks by incorporating a multi-task learning strategy to improve the model generalization ability and improve the model performance. Therefore, the method in this paper achieves better prediction results compared with existing methods.

The above comparison can show that although the prediction by vulnerability code has a lower F1 score, the severity features contained in the vulnerability code can be used as a supplement to the vulnerability description and provide more information for the model prediction. In this paper, by combining vulnerability description and code to predict vulnerability severity, and adding the strategy of multi-task learning, richer features can be extracted from vulnerability description and code, which further improves the prediction effect.

2) Comparison experiments of different feature extraction models

To explore the effects of different feature extractors on model performance, we use LSTM (Graves, 2012), GRU (Chen et al., 2019), Bi-LSTM (Graves and Schmidhuber, 2005), Bi-GRU (Chung et al., 2014), Att-BiLSTM (Zhou et al., 2022), and Att-BiGRU (Li et al., 2022) for feature extraction, respectively, with these conditions of data set, word embedding model, and multi-task learning held constant, where Att-BiLSTM (Zhou et al., 2022) is the addition of attention to the Bi-LSTM (Graves and Schmidhuber, 2005) mechanism. The experimental results are compared to observe the effects of different feature extractors on the model performance. The experimental results are shown in Table 3.

After comparison, we found that Att-BiGRU (Li et al., 2022) and Att-BiLSTM (Zhou et al., 2022) performed better in the bimodal dataset compared to other feature extractors, and they both performed not much differently, but Att-BiGRU (Li et al., 2022) performed slightly better, so we chose it as the feature extraction model for this task. The detection effectiveness of these two models then decreases slightly after removing the attention mechanism, because the models cannot give more attention to the vulnerability features. Comparing the Bi-LSTM (Graves and Schmidhuber, 2005) and LSTM (Graves, 2012) models, the Bi-LSTM (Graves and Schmidhuber, 2005) model has the advantage of learning bidirectional information, which can learn the potential information of sequence elements like code more accurately. However, the prediction effect of LSTM (Graves, 2012) model is degraded due to its lack of back-to-back information in encoding, which leads to inaccurate results. For a sequence element like a code, both its forward and backward information are meaningful, so better results can be obtained by using a bi-directional model. Therefore, we choose Att-BiGRU (Li et al., 2022) as our feature extractor.

3) Comparison experiments of different weights for multi-task learning

The vulnerability severity prediction method based on multi-task learning proposed in this paper makes the model's performance on

Table 3
Results of different feature extraction models.

Model	Acc(%)	P(%)	R(%)	F1(%)
LSTM (Graves, 2012)	91.88	91.77	92.00	91.89
GRU (Chen et al., 2019)	92.06	92.01	92.13	92.07
Bi-LSTM (Graves and Schmidhuber, 2005)	92.75	93.07	92.38	92.72
Bi-GRU (Chung et al., 2014)	92.88	93.20	92.50	92.85
Att-BiLSTM (Zhou et al., 2022)	93.63	94.29	92.88	93.58
Att-BiGRU (Li et al., 2022)	**93.88**	**94.54**	**93.13**	**93.83**

different tasks more balanced and excellent by jointly training the vulnerability severity prediction and exploitability prediction tasks together, and assigning different loss function weights to the two tasks during the training process by using the static weight setting method of grid search. The experiments in this section aim to investigate the effects of different weight settings on the model severity prediction performance and to find out the optimal weight settings by comparing the experimental results. The loss weight w_1 for the vulnerability severity prediction for the main task is set to 1 and the loss weight w_2 for the exploitability prediction for the auxiliary task is searched between 0.1 and 1 in steps of 0.1. The results of the severity prediction experiments with different weight settings are shown in Table 4.

From the above experimental results, it can be seen that the model achieves optimal severity prediction when the weights of the loss functions for the main and auxiliary tasks are set to 1 and 0.4, respectively. In addition, the F1 score predicted by the model decreases as the weights of the auxiliary tasks continue to increase, and the model's prediction is worst when the weights of the loss function are all set to 1. This is because when the weight of the auxiliary task is set too large, the model pays too much attention to the auxiliary task and ignores the main task, resulting in a decrease in the learning effect. And when the weight of the auxiliary task is set too small, the model cannot fully utilize the information of the auxiliary task, and the predicted F1 score of the model also appears to decline. In addition, by comparing the experimental results derived from Tables 2 and 4, the accuracy, precision, recall, and F1 score of multi-task learning with different weights are better than the existing defect prediction methods, which proves the advantages of multi-task learning.

4.4. Ablation experiments and analysis

We validate the impact of the GraphCodeBert (Guo et al., 2020) bimodal data word embedding module, the Att-BiGRU (Li et al., 2022) feature extraction module, and the multi-task learning assistance task on the model through ablation experiments. The experiments were all conducted on the Bimodal-vul dataset. For the network after removing the GraphCodeBert (Guo et al., 2020) word embedding module, keeping all other variables unchanged, the treatment of the vulnerability descriptions is still performed as described in Section III-A subsection. For the vulnerability code we process it as a token sequence. Finally, the vulnerability description and the code token sequence are stitched together and used as the input of the network. After removing the Att-BiGRU (Li et al., 2022) feature extraction module from the network, we extract a semantic feature vector [*CLS*] from GraphCodeBert (Guo et al., 2020), which represents the features after bimodal data fusion and can be used for the subsequent classification prediction task. In addition, we remove the auxiliary task of vulnerability availability prediction to explore the impact of multi-task learning on the overall network prediction effectiveness. The experimental results are shown in Table 5.

The GraphCodeBert (Guo et al., 2020) module can better embed the vulnerability description and code bimodal data to learn their potential features, and when the GraphCodeBert (Guo et al., 2020) word embedding module is removed from the network, the F1 score is reduced by 8.92 %. The GraphCodeBert (Guo et al., 2020) word embedding module is able to deeply fuse the features of vulnerability descriptions and code, which makes up for the lack of severity prediction using only vulnerability descriptions. When the Att-BiGRU (Li et al., 2022) feature extraction module is removed from the original structure, the F1 score of its prediction results is reduced by 1.38 %, which shows that the further feature extraction of Att-BiGRU (Li et al., 2022) has a certain effect on the improvement of the model prediction effect. The F1 score of the prediction results of the network model decreased by 1.18 % when the vulnerability availability prediction auxiliary task module was removed, which shows that the multi-task learning can learn richer shared information factors compared with the single task, tap the relationship between tasks, get to learn additional useful information between different subtasks, and the model learns better and generalizes more.

Table 4
Results of different weights for multi-task learning.

Weights(w_1, w_2)	A(%)	P(%)	R(%)	F1(%)
(1, 0.1)	92.81	93.30	92.25	92.77
(1, 0.2)	93.00	93.54	92.38	92.96
(1, 0.3)	93.38	93.92	92.75	93.33
(1, 0.4)	**93.88**	**94.54**	**93.13**	**93.83**
(1, 0.5)	93.56	94.17	92.88	93.52
(1, 0.6)	93.02	93.52	92.45	92.98
(1, 0.7)	93.19	93.68	92.63	93.15
(1, 0.8)	92.88	93.75	91.88	92.80
(1, 0.9)	92.00	91.90	92.13	92.01
(1, 1)	91.81	91.76	91.88	91.82

Table 5
Results of ablation experiments.

Removal module	A(%)	P(%)	R(%)	F1(%)
GraphCodeBert (Guo et al., 2020)	85.00	85.44	84.38	84.91
Att-BiGRU (Li et al., 2022)	92.50	93.04	91.88	92.45
Auxiliary task	92.69	93.17	92.13	92.65
Our approach	**93.88**	**94.54**	**93.13**	**93.83**

5. Limitations

The method proposed in this paper has achieved good results, but there are some limitations.

GraphCodeBert (Guo et al., 2020) combines graph-based structures and BERT, increasing model complexity and requiring significant computational resources. The iterations required for its learning from the graph representation may increase the computational overhead, leading to longer training times. And dealing with larger code bases or more extensive code repositories may introduce significant overhead due to increased resource requirements. Att-BiGRU (Li et al., 2022) integrates the self-attention mechanism into the BIGRU (Chung et al., 2014) model, which makes the decision to interpret the model more complex, increasing the complexity of the model and requiring higher computational resources. It may be more sensitive to hyperparameters, making fine-tuning challenging and risking overfitting. Multi-task learning merging multiple tasks in a single model increases its complexity, and the need to learn multiple objectives simultaneously increases training time and requires more memory and computational power. In addition to this, obtaining sufficiently large and varied datasets for multiple tasks can be difficult, especially if data availability is limited for some tasks. In this paper, only two tasks were used for the research component and each task had sufficient datasets so that the above problems do not exist. However, applying multi-task learning to other directions may have these drawbacks.

In future work, we will take effective measures to expand the scope of the dataset to minimize the negative impact of factors such as noise on severity prediction models. The scope of the dataset can be effectively expanded by performing a detailed analysis of vulnerability reports, including vulnerability characteristics, source code, and other relevant severity information, to collect relevant data from a wider range of open source projects. And by training on a large amount of labeled data, the performance of our model can be improved.

6. Conclusion

This paper proposes a severity prediction method based on bimodal data of vulnerabilities, which can effectively solve the limitation of using only unimodal data in vulnerability severity prediction, and the method effectively fuses vulnerability description with code to obtain richer vulnerability severity information. Feature extraction of bimodal data

using GraphCodeBert model and Att-BiGRU model for better extraction of different vulnerability severity features. In addition, this paper introduces a multi-task learning strategy to learn a wider range of knowledge by combining vulnerability exploitability prediction tasks, which improves the generalization ability of the model and achieves more accurate prediction of vulnerability severity. The experimental results of this method achieve a ground prediction F1 score of 93.83 %, indicating that the use of bimodal data and multi-task learning can significantly improve the severity prediction of the model.

CRediT authorship contribution statement

Xiaozhi Du: Conceptualization, Investigation, Methodology, Writing – original draft. **Shiming Zhang:** Conceptualization, Methodology, Validation, Writing – original draft. **Yanrong Zhou:** Data curation, Software, Validation, Writing – review & editing. **Hongyuan Du:** Data curation, Software, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data and code are publicly available on Github and referenced in the article and a link is provided in the submission as well.

Acknowledgements

This work was partially supported by the Chinese National Natural Science Foundation under Grant 12275208.

References

Aivatoglou, G., Anastasiadis, M., Spanos, G., et al., 2022. A RAkEL-based methodology to estimate software vulnerability characteristics & score. Multimed. Tools Appl. 81, 9459–9479.
Chakraborty, S., Krishna, R., Ding, Y., et al., 2022. Deep learning based vulnerability detection: are we there yet. IEEE Trans. Softw. Eng. 48 (9), 3280–3296.
Chen, J., Jing, H., Chang, Y., et al., 2019. Gated recurrent unit based recurrent neural network for remaining useful life prediction of nonlinear deterioration process. Reliab. Eng. Syst. Saf. 185, 372–382.
Chen, J.F., Kudjo, P.K., Mensah, S., et al., 2020. An automatic software vulnerability classification framework using term frequency-inverse gravity moment and feature selection. J. Syst. Softw. 167, 110616.
Chen, J.F., Zhang, C., Cai, S.H., et al., 2023. A memory-related vulnerability detection approach based on vulnerability model with petri net. J. Log. Algebr. Methods Programm. 132 https://doi.org/10.1016/j.jlamp.2023.100859.
Cheng, X., Wang, H.Y., Hua, J.Y., et al., 2021. DeepWukong: statically detecting software vulnerabilities using deep graph neural network. ACM Trans. Softw. Eng. Methodol. (TOSEM) 30 (3), 1–33.
Chung, J., Gulcehre, C., Cho, K., et al., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In: NIPS 2014 Workshop on Deep Learning, December 2014.
Dong, Y., Tang, Y., Cheng, X., et al., 2023. DeKeDVer: a deep learning-based multi-type software vulnerability classification framework using vulnerability description and source code. Inf. Softw. Technol., 107290.
Du X., Zhang S., Zhou Y., Du H., (2023), Github repository, https://github.com/NoCaiTnT/A-Vulnerability-Severity-Prediction-Method-Based-on-Bimodal-Data-and-Multi-task-Learning.
ExploitDB[EB/OL]. 2022 https://www.exploit-db.com/.
Fan, J.H., Li, Y., Wang, S.H., et al., 2020. A C/C++ Code vulnerability dataset with code changes and CVE summaries. In: Proceedings of the 17th International Conference on Mining Software Repositories (MSR), pp. 508–512. Virtual, Online.
Graves, A., Schmidhuber, J., 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks 18 (5–6), 602–610.
Graves, A., 2012. Supervised Sequence Labelling With Recurrent Neural Networks. Long short-term memory, pp. 37–45.
Guo, D., Ren, S., Lu, S., et al., 2020. GraphCodeBERT: pre-training code representations with data flow. In: International Conference on Learning Representations.
Hadi, M.A., Fard, F.H., 2023. Evaluating pre-trained models for user feedback analysis in software engineering: a study on classification of app-reviews. Empir. Softw. Eng. 28, 88. https://doi.org/10.1007/s10664-023-10314-x.
Han, Z.B., Li, X.H., Xing, Z.C., et al., 2017. Learning to predict severity of software vulnerability using only vulnerability description. In: 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, Shanghai, China, pp. 125–136.
Han, X., Wang, Y.T., Feng, J.L., et al., 2023. A survey of transformer-based multimodal pre-trained modals. Neurocomputing 515, 89–106.
Hao, J.W., Luo, S.L., Pan, L.M., 2023. A novel vulnerability severity assessment method for source code based on a graph neural network. Inf. Softw.Technol. 161 https://doi.org/10.1016/j.infsof.2023.107247.
Hassan, S., Fahim, D., Nadir, D., et al., 2023. On the effect of dropping layers of pre-trained transformer models. Comput. Speech Lang. https://doi.org/10.1016/j.csl.2022.101429.
Jabeen, G., Yang, X., Luo, P, 2021. Vulnerability severity prediction model for software based on Markov chain. Int. J. Inf. Comput. Secur. 15 (2–3), 109–140.
Jiang, Y.N., Atif, Y., 2022. Towards automatic discovery and assessment of vulnerability severity in cyber–physical systems. Array 15. https://doi.org/10.1016/j.array.2022.100209.
Kudjo, P.K., Chen, J.F., Mensah, S., et al., 2020. The effect of Bellwether analysis on software vulnerability severity prediction models. Softw. Qual. J. 28 (4), 1413–1446.
Li, Z., Zou, D.Q., Xu, S.H., et al., 2018. VulDeePecker: a deep learning-based system for vulnerability detection. In: Proceedings of the 25th Annual Network and Distributed System Security Symposium. Internet Society, San Diego, USA, pp. 1–10.
Li, X., Wang, L., Xin, Y., et al., 2021. Automated software vulnerability detection based on hybrid neural network. Appl. Sci. 11 (7), 3201.
Li, H., He, E., Kuang, C., et al., 2022. An abnormal traffic detection based on attention-guided bidirectional GRU. In: 2022 IEEE 22nd International Conference on Communication Technology (ICCT). IEEE, pp. 1300–1305.
Liu, K., Zhou, Y., Wang, Q., et al., 2019. Vulnerability severity prediction with deep neural network. In: 2019 5th International Conference on Big Data and Information Analytics (BigDIA). Kunming, China, pp. 114–119.
Liu, C., Li, Y., Lin, H., et al., 2023. GNNRec: gated graph neural network for session-based social recommendation model. J. Intell. Inf. Syst. 60, 137–156.
Lu, S., Jiang, Y., Xie, S., et al., 2023. Sensitive word recognition scheme based on Text-RCNN model. In: 2023 International Conference on Data Science and Network Security (ICDSNS). IEEE, pp. 1–6.
Luo, Y., Xu, W.F., Xu, D.X., 2022. Compact abstract graphs for detecting code vulnerability with GNN models. In: Proceedings of the 38th Annual Computer Security Applications Conference (ACSAC). Austin, USA, pp. 497–507.
Malhotra, R., Vidushi, V., 2023. Impact of word embedding methods on software vulnerability severity prediction models. In: 2023 13th International Conference on Cloud Computing, Data Science & Engineering (Confluence), pp. 293–297.
Malhotra, R., 2021. Severity prediction of software vulnerabilities using textual data. In: Proceedings of the International Conference on Recent Trends in Machine Learning, pp. 453–464.
Nakagawa, S., Nagai, T., Kanehara, H., et al., 2019. Character-level convolutional neural network for predicting severity of software vulnerability from vulnerability description. IEICE Trans. Inf. Syst. 102 (9), 1679–1682.
Napier, K., Bhowmik, T., Wang, S., 2023. An empirical study of text-based machine learning models for vulnerability detection. Empir. Softw. Eng. 28, 38. https://doi.org/10.1007/s10664-022-10276-6.
Ni, X.M., Zheng, J.X., Guo, Y., et al., 2022. Predicting severity of software vulnerability based on BERT-CNN. In: International Conference on Computer Engineering and Artificial Intelligence(ICCEAI). IEEE, Shijiazhuang, China, pp. 711–715.
Sahin, S.E., Sahin, 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.
Sharma, R., Sibal, R., Sabharwal, S., 2021. Software vulnerability prioritization using vulnerability description. Int. J. Syst. Assur. Eng. Manag. 12 (1), 58–64.
Soumyadeep, H., Ankit, S., Nathaniel, D., 2023. Deep VULMAN: a deep reinforcement learning-enabled cyber vulnerability management framework. Expert. Syst. Appl. 221 https://doi.org/10.1016/j.eswa.2023.119734.
Suzuki, J., Zen, H., Kazawa, H., 2023. Extracting representative subset from extensive text data for training pre-trained language models. Inf. Process. Manage. 60 (3) https://doi.org/10.1016/j.ipm.2022.103249.
Vaswani, A., Shazeer, N., Parmar, N., et al., 2017. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach, USA, pp. 6000–6010.
Wang, P., Zhou, Y., Sun, B., et al., 2019. Intelligent prediction of vulnerability severity level based on text mining and XGBboost. In: Eleventh International Conference on Advanced Computational Intelligence, pp. 72–77.
Wang, H.T., Ye, G.X., Tang, Z.Y., et al., 2021. Combining graph-based learning with automated data collection for code vulnerability detection. IEEE Trans. Inf. Forens. Secur. 16, 1943–1958.
Wu J. Literature review on vulnerability detection using NLP technology. arXiv preprint arXiv:2104.11230, 2021.
Zeng, P., Lin, G.J., Pan, L., et al., 2022. Software vulnerability analysis and discovery using deep learning techniques: a survey. IEEE Access 8, 197158–197172.
Zhou, Y.Q., Liu, S.Q., Siow, J.K., et al., 2019. Devign:effective vulnerability identification by learning comprehensive program semantics via graph neural networks. Adv. Neural Inf. Process. Syst. 32, 12–18.

Zhou, D., Zhuang, X., Zuo, H., et al., 2022. A model fusion strategy for identifying aircraft risk using CNN and Att-BiLSTM. Reliab. Eng. Syst. Saf. 228, 108750.
Ziems, N., Wu, S., 2021. Security vulnerability detection using deep learning natural language processing. In: IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, pp. 1–6.
Zou, D.Q., Wang, S.J., Xu, S.H., et al., 2021. μVulDeePecker: a deep learning-based system for multiclass vulnerability detection. IEEE Trans. Dependable Secure Comput. 18 (5), 2224–2236.

Xiaozhi Du received the Ph.D. in Computer Science and Technology from Xi'an Jiaotong University of China in 2010. He is now an Associate Professor in the school of Software Engineering of the same university. His research interests mainly include software security, software engineering and machine learning.

Shiming Zhang is a master student of Software Engineering at Xi'an Jiaotong University. His research interests include software security and machine learning.

Yanrong Zhou is a master student of Software Engineering at Xi'an Jiaotong University. Her research interests include software security and machine learning.

Hongyuan Du is a master student of Software Engineering at Xi'an Jiaotong University. Her research interests include software security and machine learning.