Key Aspects Augmentation of Vulnerability Description based on Multiple Security Databases

Hao Guo[1], Zhenchang Xing[2], Sen Chen[1*], Xiaohong Li[1*], Yude Bai[1], Hu Zhang[3]
[1]*College of Intelligence and Computing, Tianjin University, Tianjin, China*
[2]*Research School of Computer Science, Australian National University and Data61 CSIRO, Australia*
[3]*State Grid Customer Service Center, Tianjin, China*
{*haoguo, senchen, xiaohongli, baiyude*}*@tju.edu.cn, zhenchang.xing@anu.edu.au*

***Abstract*—Common Vulnerabilities and Exposures (CVE) is one of the most influential security databases. With the continuous disclosure of security vulnerabilities, the characteristics of them are documented as vulnerability reports, and it is discovered that their impact on computer systems is increasing. However, as our research continues to deepen, we find that the current lack of key aspects of CVE description is more serious than before. In response to this situation, our research focuses on how to correctly and completely extract key aspect descriptions from various security vulnerability databases to supplement the CVE reports. First, we fetch almost all of semi-structured vulnerability reports from the CVE, Security Focus, and IBM X-Force Exchange databases before November 2020. We then propose a customized NER (Named entity recognition) method based on deep neural networks to extract six key aspects from unstructured descriptions. Finally, we use the corresponding security vulnerability reports in other vulnerability databases to complete the missing key aspects in the correlated CVE description. We conduct sampling surveys on various aspects of this information, and verify the accuracy of extracting key aspects, and find that our method can extract key information from vulnerability descriptions. To demonstrate the usefulness of key aspects augmentation, after completing the missing affected product, root cause, attacker type, attack vector, impact, and vulnerability type in the CVE description, we verify the effectiveness of completing the key aspects of the vulnerability in predicting the severity of the security vulnerability.**

***Index Terms*—CVE, Vulnerability information completeness, Vulnerability description, Neural network**

I. Introduction

Security vulnerabilities are constantly being discovered and disclosed [1]–[4]. With the development of computer systems and software, they continue to appear, discovered by security researchers and recorded in files. Once these vulnerabilities are exploited, they will bring huge security risks and damage to the systems. Common Vulnerabilities and Exposures (CVE) [5] is a list of entries, each entry contains an identification number, vulnerability description and reference list, and other information. Many security products and services use at least one known public vulnerability database (including the US National Vulnerability Database (NVD) [6], IBM X-Force Exchange [7], and SecurityFocus [8]) as a reference. Among these databases, CVE is the most widely-used in the current security research field. At present, researchers use CVE as a public database and accept security researchers to submit security vulnerabilities discovered by themselves.

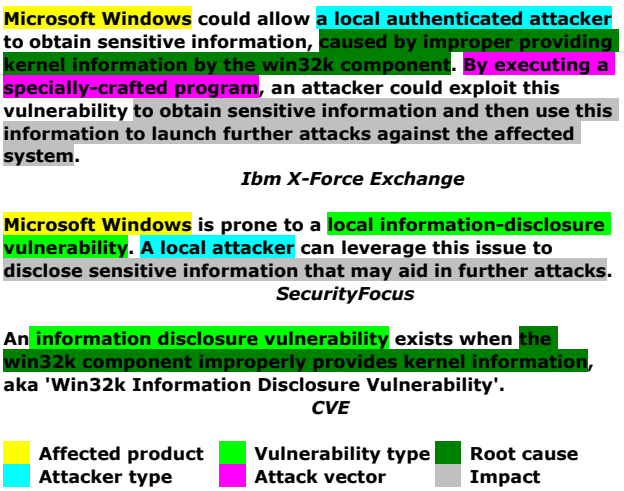

Fig. 1: Description of CVE-2019-0628 in different security vulnerability databases

In the CVE official vulnerability submission guidelines, the extended description should at least describe the following six key aspects in detail: vulnerability type or root cause, affected product (or vendor component, if applicable), attacker type, attack vector, and impact. These key aspects each describe different dimensions of information about security vulnerabilities, and each key aspect is important. These six key aspects describe the vulnerability from multiple perspectives such as the abstraction of the security vulnerability, the cause of the vulnerability, the product version and the part where the vulnerability occurs, the form of exploitation, the method of exploitation, and the consequences of the vulnerability.

SecurityFocus [8] and IBM X-Force Exchange [7] are professional vulnerability security databases, which have similar functions to CVE. These two vulnerability databases contain most of the vulnerabilities that appear in CVE. These databases have their own evaluation standards for security vulnerabilities, and the key aspects of the vulnerabilities described are similar to CVE. After investigated SecurityFocus [8] and IBM X-Force Exchange [7], we found that there are 99,377 Bugtraq-ids in SecurtiyFocus and 120,879 X-Force vulnerabilities in IBM X-Force Exchange. The same as CVE,

*Sen Chen and Xiaohong Li are the corresponding authors.

the information contained in these reports is composed of the six aspects mentioned above but the focus is different. Among these incomplete CVE reports, over 78% of the vulnerability reports have at least one related vulnerability description in the IBM X-Force Exchange and SecurityFocus databases. The key aspects of vulnerabilities in these different security databases are described differently, and a large amount of information is complementary, which shows that our work is highly feasible. Fig. 1 shows the description of the CVE-2019-0628 vulnerability in each of the three vulnerability databases. From the figure, we can see that the key aspects of the vulnerability described in the three databases are not the same completely. Each of the three vulnerability databases does not fully describe the six key aspects of the vulnerability, but the key aspects in these databases are complementary.

The inconsistencies in the key aspects of these security databases have created prerequisites for our augmentation of the key aspects. In this paper, we propose a customized NER (Named entity recognition) method based on deep neural networks to extract descriptions of CVE and other vulnerability databases and supplement the CVE data with key aspects of vulnerabilities extracted from corresponding vulnerability descriptions in other vulnerability databases. For the description of CVE, it is helpful to describe the details of CVE as much as possible for the research and use of vulnerabilities. After that, we studied the lack of key aspects of each vulnerability database, explored the characteristics and shortcomings of each vulnerability database. We calculated how comprehensive the three vulnerability databases can augment CVE information. Finally, we conducted experiments using artificially concealed CVEs in one of the six key areas, using augmented CVEs and using CVE source data to predict the performance of CVE severity levels (CVSS). Experiments show that our method can help the vulnerability severity prediction task well. These key aspects in the database are valuable resources. How to use them to supplement the deficiencies in CVE is one of the focuses of our research on CVE description.

In summary, this paper make the following contributions:

- We are the first to focus on the vulnerability description augmentation based on the relation of security databases, and explore the similarities and differences between multiple security vulnerabilities.
- We design a neural network-based model to automatically extract the key aspects of multiple vulnerability databases and use them to complete the missing information in the CVE. We also conduct experiments to compare the effectiveness of different model design variants.
- We further conduct experiments to explore the improvement of our method on the task of vulnerability severity prediction and the impact of the lack of key aspects on this task. Experiments show that this method can improve the effectiveness of vulnerability severity prediction task.

II. Preliminary

1) Vulnerability type is an abstraction of vulnerabilities, usually identified as one of the "Common Weakness Enumeration" (CWE) [9]. When submitting security vulnerabilities, the submitter needs to choose to indicate the vulnerability type, but if the vulnerability type of the vulnerability is not in the selection list or the submitter does not know, you can choose "Other" or "Unknown". *2) Root cause* indicates the cause of the security vulnerabilities, which can be program design, environment configuration, value or condition verification, and errors in the system operation process, or other aspects. *3) Affected product* identifies certain versions of software products affected by vulnerabilities and software product components that have problems. Product components can be environment variables, certain modules, files, functions or executable files in the software. When submitting a new CVE, the reporter must provide the affected product and version as well as the product supplier, so there will be no lack of such information. *4) Attacker type* is an abstraction of an attacker who exploits a security vulnerability. As an optional field, the CVE request site provides 5 selection mechanisms: Authenticated, local, remote, physical, and context-sensitive. Although the vulnerability discoverer can leave this field unspecified or select other fields when submitting the vulnerability report, they may mention the type of attacker in the vulnerability description. *5) Impact* represents what an attacker can achieve by exploiting this vulnerability and the impact on software products. The CVE request site provides 4 common choices: Code execution, information leakage, denial of service, and privilege escalation. This key aspect is a phenomenon that is an external manifestation of security vulnerabilities and is easy to be observed, so Impact is not missing much in the vulnerability report. *6) Attack vector* describes the method and medium of exploiting the vulnerability. For example, to exploit the vulnerability, a carefully crafted JPEG file must be opened, or a malicious query path must be used.

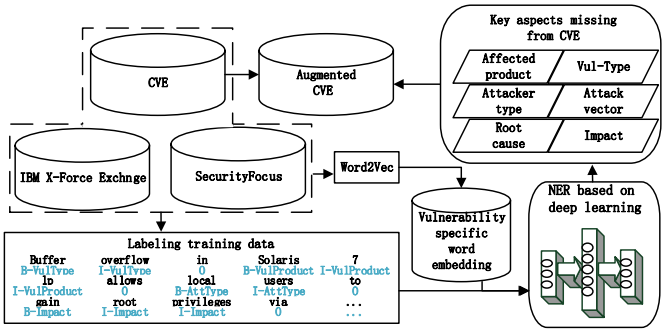

Fig. 2: Overview of our approach

III. Approach

A. Approach Overview

To complement the key aspects missing in CVE, we chose SecurityFocus and IBM X-Force Exchange, which are widely recognized by the industry. We crawled the relevant data from the official websites of these databases. As shown in Fig. 2, we convert these vulnerability descriptions into continuous tokens and label these tokens. After extracting the vulnerability description, we use the Skip-gram model to train the word

embedding on the crawled vulnerability description corpus. The output of the training word embedding is the word vector dictionary of each word in the vocabulary of the crawled vulnerability descriptions. After that, we used a customized NER (Named entity recognition) method based on deep neural networks to extract six key aspects of these databases: affected product, vulnerability type, root cause, attacker type, attack vector, and impact. Finally, we complete the CVE description using key aspects missing from the CVE included in SecurityFocus and IBM X-Force Exchange.

Security Vulnerability Dataset. The CVE list can be downloaded from the CVE official website [5]. The data of SecurityFocus and IBM X-Force Exchange can also be obtained from their official websites. Then we need to convert these descriptions into continuous tokens and label them. The CRF (Conditional Random Fields) layer can add some constraints to the last predicted label to ensure that the predicted label is legal. In the training process of training data, these constraints can be automatically learned through the CRF layer. These constraints can be: *1)* The first word in the six key aspects always starts with the label "B-", and "I-" indicates the middle word. *2)* For labels "B-label1 I-label2 I-label3 I-...", label1, label2, label3 should belong to the same type of entity. For example, "B-Impact I-Impact" is a legal sequence, but "B-Impact I-Attack_vector" is an illegal tag sequence. *3)* The token label that does not belong to the six key aspects is always "O". In this work, we randomly selected 3,000 vulnerability entries from the vulnerability database released from January 1999 to November 2020. We convert those data into a continuous sequence and then label them.

B. Key Aspects Extraction

1) Input and Representation: From the perspective of the model, the named entity recognition (NER) [10] problem is actually a sequence labeling problem. The sequence labeling problem means that the input of the model is a sequence, including text, time, etc., and the output is also a sequence. For each unit of the input sequence, a specific label is an output. Tokens in vulnerability description are discrete symbols that need to be represented as vectors in NLP tasks. Inspired by many successful applications of word embedding in general and domain text [11]–[13], word embedding is a real-valued, low-dimensional word vector that captures lexical rich syntactic and semantic characteristics, so we decided to use word embedding to represent tokens in vulnerability descriptions.

Our corpus has the vocabulary size of 62,245. Continuous Skip-gram model learns word embeddings that are good at predicting the surrounding words with a center word. The objective function of the model is to maximize the sum of log probabilities of the surrounding words w_{i+j} in a context window of size $2k+1$ ($k = 5$ in this work) conditioned on the center word w_i in the window.

The output of the continuous Skip-gram model is a dictionary of words. Each word is associated with a vector representation $w \in \mathcal{R}^d$ where d is the word embedding dimension. Following the experiments on word embedding dimension in Han et al. [11], we set d at 100 in this work. The words in the dictionary may not cover all words in the vulnerability description. We follow the common practice to deal with those out-of-vocabulary words [12], i.e., randomly initialize the corresponding word vectors.

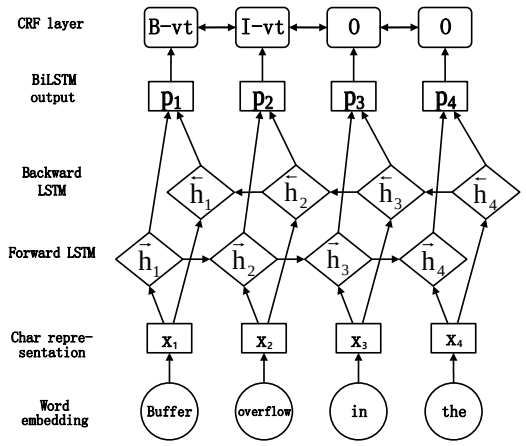

Fig. 3: Neural network model structure

2) Customized NER based on Deep Learning: The vulnerability description is concise, but the scope contains rich and diverse vulnerability semantic information. In addition, considering that some special symbols also have certain semantics, it will increase the complexity of feature extraction. Our task requires a complete sequence function. Since the vulnerability description usually contains 1-2 sentences with rich information, we leverage BiLSTM (Bidirectional LSTM)+CharCNN+CRF (Conditional Random Field) to capture the structure and characteristics of the various information we need. The vulnerability description token sequence is given. First, our method finds the vector describing each tag in the sentence in the word embedding dictionary, and then connects it to the vulnerability description vector.

Neural network has become a model that can effectively handle many NLP tasks. We convert the security vulnerability description information extraction into a sequence labeling task. The label is mapped from a discrete one-hot representation to a low-dimensional space and then the embedded sentence sequence is input to BiLSTM, and then the neural network automatically extracts the features and uses Softmax to predict each token label.

As shown in Fig. 3, we trained a BiLSTM+CharCNN+CRF model to solve the sequence labeling problem in the key aspects of CVE description. We perform batch training. The length of the word vector is 100, the BiLSTM layer is 1, the learning rate is set to 0.012, and the subnet parameters are updated by using the Adam optimization method [14]. We convert the labeled CVE description into a word sequence (including words, hollow Brackets, parentheses, dots, etc.), and then mark each unit in the sequence according to the extracted CVE information. We conducted several experiments to study the performance of the proposed BiLSTM+CharCNN+CRF architecture and compared it with other baseline methods.

We highlight that CharCNN can capture character-level features well. In this work, CharCNN includes a character representation layer, a maximum pooling layer, a convolutional layer and a Char Embedding layer. Taking as input the word embedding representation of a vulnerability description $D = (w_1, w_2, \cdots, w_n)$, where w_i is the word embedding of the ith word and n is the total number of words in the description, a standard LSTM recursively computes a hidden vector sequence $h = (h_1, h_2, \cdots, h_n)$ in one direction. A Bi-directional LSTM processes the input sequence in both forward and backward direction with two standard LSTMs, respectively. In this work, we develop a BiLSTM with c ($c = 100$) LSTM cells which contain a forward LSTM $\overrightarrow{lstm}_f$ network that reads the input from w_1 to w_n, and a backward LSTM $\overleftarrow{lstm}_b$ that reads from w_n to w_1: $\overrightarrow{h}_i = \overrightarrow{lstm}_f(w_i), i \in [1, n]$, $\overrightarrow{h}_i \in \mathcal{R}^c$, $\overleftarrow{h}_i = \overleftarrow{lstm}_b(w_i), i \in [n, 1]$, $\overleftarrow{h}_i \in \mathcal{R}^c$. The parameters of $\overrightarrow{lstm}_f$ and $\overleftarrow{lstm}_b$ will be learned during model training. We obtain the hidden vector h_i for a given word w_i by concatenating the forward hidden vector $\overrightarrow{h}_i$ and the backward hidden vector $\overleftarrow{h}_i$, i.e., $h_i = \overrightarrow{h}_i \oplus \overleftarrow{h}_i$ ($h_i \in \mathcal{R}^{2c}$). h_i is the output vector of the BiLSTM for the input word w_i, which encodes both the preceding and succeeding sentence context centered around w_i. The CRF layer can add constraints to the last predicted label to ensure that the predicted label is legal. In the training process of training data, these constraints can be automatically learned through the CRF layer.

Finally, we can automatically mark a continuous sequence belonging to each key aspect in a vulnerability description, thereby extracting key aspects from the vulnerability.

IV. Experiments

We conduct a series of experiments to answer to the following three research questions:

- **RQ1: (Efficiency of neural network)**: How do different model architectures and neural network designs affect the extraction performance of key aspects?
- **RQ2: (Key aspects missing from the security vulnerability databases)**: How many key aspects are missing from CVE, SecurityFocus, and IBM X-Force Exchange? How many key aspects are missing in the augmented CVE?
- **RQ3: (Advantages of CVE augmentation)**: Does the augmentation in key aspects enhance the prediction accuracy of the severity of security vulnerabilities?

A. Experiment Setup

1) Evaluation Metrics: We use Precision, Recall, and F1-score that are commonly used to evaluate the effectiveness of multi-class classification in the literature. In the exact matching process, the entity category and boundary range need to be predicted. Only when these two parts are matched successfully, the prediction is correct, otherwise, it is a prediction error. Since F1-score conveys the balance between the precision and the recall, we use F1-score as the main evaluation metric.

TABLE I: Performance on CVE

		Affected product	Impact	Vulnerability type	Root cause	Attack vector	Attacker type
Pre	1-L BiLSTM+CRF	0.9239	0.8177	0.9598	0.9291	0.7218	0.9010
	2-L BiLSTM+CRF	0.9230	0.8242	0.9592	0.9311	0.7576	0.9026
	1-L BiLSTM	0.8101	0.5230	0.8328	0.7205	0.5085	0.8042
	1-L BiLSTM+CNN	0.8278	0.5490	0.8203	0.8611	0.5612	0.8102
	1-L BiLSTM+CNN+CRF	0.9317	0.8533	0.9549	0.9151	0.7250	0.8923
Rec	1-L BiLSTM+CRF	0.9724	0.8632	0.9593	0.9221	0.9385	0.9690
	2-L BiLSTM+CRF	0.9771	0.8411	0.9674	0.9210	0.9221	0.9662
	1-L BiLSTM	0.9209	0.6886	0.8232	0.8312	0.8607	0.9669
	1-L BiLSTM+CNN	0.9316	0.7215	0.8483	0.9123	0.8648	0.9655
	1-L BiLSTM+CNN+CRF	0.9766	0.8862	0.9774	0.9221	0.9508	0.9711
F1	1-L BiLSTM+CRF	0.9495	0.8445	0.9596	0.9259	0.8165	0.9338
	2-L BiLSTM+CRF	0.9493	0.8326	0.9632	0.9260	0.8218	0.9333
	1-L BiLSTM	0.8620	0.6147	0.8280	0.7719	0.6393	0.8781
	1-L BiLSTM+CNN	0.8767	0.6239	0.8341	0.8860	0.6806	0.8810
	1-L BiLSTM+CNN+CRF	0.9536	0.8694	0.9660	0.9186	0.8227	0.9324

TABLE II: Performance on SecurityFocus

		Affected product	Impact	Vulnerability type	Root cause	Attack vector	Attacker type
Pre	1-L BiLSTM+CRF	0.9365	0.9410	0.9372	0.8500	0.7500	0.9728
	2-L BiLSTM+CRF	0.9346	0.9399	0.9269	0.8395	0.5116	0.9790
	1-L BiLSTM	0.9184	0.8221	0.9111	0.6739	0.4043	0.9284
	1-L BiLSTM+CNN	0.9309	0.8329	0.8856	0.6865	0.6000	0.9496
	1-L BiLSTM+CNN+CRF	0.9245	0.9067	0.9220	0.8831	0.6757	0.9647
Rec	1-L BiLSTM+CRF	0.9819	0.9381	0.9586	0.9067	0.7500	0.9699
	2-L BiLSTM+CRF	0.9839	0.9428	0.9710	0.9067	0.7857	0.9849
	1-L BiLSTM	0.9758	0.9074	0.9545	0.8267	0.6786	0.9759
	1-L BiLSTM+CNN	0.9778	0.9257	0.9572	0.8633	0.7500	0.9639
	1-L BiLSTM+CNN+CRF	0.9879	0.9598	0.9793	0.9067	0.8930	0.9880
F1	1-L BiLSTM+CRF	0.9587	0.9395	0.9478	0.8774	0.7500	0.9713
	2-L BiLSTM+CRF	0.9566	0.9414	0.9484	0.8718	0.6197	0.9820
	1-L BiLSTM	0.9462	0.8589	0.9323	0.7425	0.5067	0.9515
	1-L BiLSTM+CNN	0.9538	0.8768	0.9321	0.7556	0.6667	0.9567
	1-L BiLSTM+CNN+CRF	0.9552	0.9323	0.9498	0.8947	0.7692	0.9762

2) Setting of Model Training: We implement the proposed neural network in TensorFlow. All security vulnerability databases are trained by using the proposed model. Specifically, we train each model for 256 iterations with a batch size of 128, set learning rate at 0.0012, and use Adam [14] as the optimizer. All experiments run on an NVIDIA Tesla M40 GPU machine.

B. Efficiency of Neural Network

Motivation. Extracting the key aspects of vulnerability descriptions is the first step in our work, which is very important for the next experiments. We want to explore the influence of different neural network models on the extraction of key aspects of vulnerability in order to find the model that is most suitable for our work. We also want to know the impact of CharCNN, CRF, and BiLSTM layers on our NER based network. For the three vulnerability databases, the impact of the model may also be different. We want to study the impact of these design schemes and combine their performance on the three vulnerability databases to determine the most effective design of the neural network.

Approach. For each key aspect of the three security vulnerability databases, we use five different models to explore the impact of models on our work. We conduct a series of comparative experiments (see in Table I-III) to explore the number of BiLSTM layers, whether there is a CharCNN layer, whether there is a CRF layer, and the effect of layer combination. We apply these neural network models to the key

TABLE III: Performance on IBM X-Force Exchange

		Affected product	Impact	Vulnerability type	Root cause	Attack vector	Attacker type
Pre	1-L BiLSTM+CRF	0.9242	0.8904	0.9135	0.7392	0.8613	0.8098
	2-L BiLSTM+CRF	0.9211	0.8895	0.9133	0.7639	0.8615	0.8009
	1-L BiLSTM	0.8490	0.8206	0.8419	0.6019	0.8013	0.7398
	1-L BiLSTM+CNN	0.8996	0.8804	0.8711	0.7025	0.8214	0.7816
	1-L BiLSTM+CNN+CRF	0.9351	0.8911	0.9103	0.7401	0.8496	0.8178
Rec	1-L BiLSTM+CRF	0.9778	0.9197	0.9483	0.8648	0.9394	0.8697
	2-L BiLSTM+CRF	0.9770	0.9267	0.9459	0.8673	0.9403	0.8529
	1-L BiLSTM	0.9150	0.8665	0.8568	0.7673	0.8411	0.7726
	1-L BiLSTM+CNN	0.9450	0.8999	0.9059	0.8197	0.8703	0.8236
	1-L BiLSTM+CNN+CRF	0.9650	0.9199	0.9446	0.8471	0.9303	0.8726
F1	1-L BiLSTM+CRF	0.9550	0.9016	0.9263	0.7863	0.9003	0.8326
	2-L BiLSTM +CRF	0.9515	0.8976	0.9233	0.7773	0.9007	0.8301
	1-L BiLSTM	0.8680	0.8371	0.8462	0.6671	0.8295	0.7519
	1-L BiLSTM+CNN	0.9330	0.8899	0.8863	0.7254	0.8602	0.8042
	1-L BiLSTM+CNN+CRF	0.9511	0.9023	0.9259	0.7773	0.9003	0.8376

aspects extraction work of the three vulnerability databases. CharCNN is an effective method to extract information (such as the prefix or suffix of a word) from the characters of the word and encode it into a neural representation.

Results. As shown in Tables I-III, we can see that our NER model achieves great performance, and the model with the best comprehensive effect is 1-layer BiLSTM+CharCNN+CRF. The output of BiLSTM is connected to the input of the CRF layer. Unlike models that can consider long-term context information (such as LSTM), CRF pays more attention to the linear weighted combination of local features of the entire sentence (scanning the entire sentence through a feature template). In order to train the neural network model to extract the features of related sequences from the vulnerability description sequence and accurately label the sentence sequence, we input a large number of crawling vulnerability description sequences and the corresponding label of each tag in the sequence into the model. The CRF model has obviously improved the effectiveness of predicting relatively inaccurate aspects, and the CharCNN model also has a good effect on feature extraction. The addition of the BiLSTM model has a counterproductive effect. Our model is relatively weak in predicting the Attack vector in CVE and SecurityFocus, Impact in CVE, and Root cause in SecurityFocus. In other key aspects, our model achieves significant performance.

C. Key Aspects Missing from Security Vulnerability Databases

Motivation. Understanding the key aspects of security vulnerability descriptions is a prerequisite for our other security vulnerability research work. The vulnerability description in the security vulnerability database is mainly composed of six key aspects: affected product, vulnerability type, root cause, attacker type, attack vector, and impact. The key aspect missing from the vulnerability description is the premise to help us understand the advantages and disadvantages of the security vulnerability database. We want to explore how many key aspects are missing from CVE, SecurityFocus, and IBM X-Force Exchange, and how many key aspects of CVE after augmentation are missing. In this RQ, we want to understand the degree of missing in each security vulnerability database and the augmented CVE.

TABLE IV: Missing aspects in the security vulnerability databases

	Affected product	Impact	Vulnerability type	Root cause	Attack vector	Attacker type
CVE	-	0.06	0.56	0.85	0.38	0.36
SecurityFocus	-	0.02	0.23	0.58	0.83	0.55
IBM X-Force Exchange	-	0.02	0.21	0.51	0.31	0.33
Augmented CVE	-	-	0.17	0.31	0.23	0.19

Approach. We categorize the description data in the three vulnerability databases according to the corresponding vulnerabilities to facilitate our completion of CVE and statistics-related information. After extracting the key aspects from the three vulnerability databases, the number of key aspects in the three vulnerability databases and the percentage of missing key aspects are calculated. We use the data extracted from the other two databases to complete the missing key aspects of CVE, and in this case, calculate the missing rate and number of key aspects again on the completed CVE data set.

Results. As shown in Table IV, in CVE, SecurityFocus, IBM X-Force Exchange, there is almost no missing of affected product information, and the missing rate of Impact is relatively small. The difference is that for the vulnerability types, there are more missing rate in the CVE database, reaching 56%, while relatively few in IBM X-Force Exchange and SecurityFocus, about 23% and 21%, respectively. For attack vectors, SecurityFocus loses the most, reaching 83%, while the attacker type is also as high as 55%. In general, the lack of CVE data is obviously alleviated after completion. After the completion, there are basically no missing cases in the impact. Compared with the previous 56% and 85%, the vulnerability type and root cause missing rate after the completion are reduced to 17% and 31%.

D. Advantages of CVE Augmentation

Motivation. Han et al. [11] proposed a neural network-based model to predict the severity of vulnerabilities. The input of the prediction model is the description of the vulnerability, and the output is the corresponding severity of the vulnerability. For a long time, we have been exploring the impact of augmentation of key aspects on other security vulnerability research work. Does the augmentation in key aspects enhance the prediction of the severity of security vulnerabilities? Combining the comprehensive factors of all aspects, if we want to further study this aspect, we need to conduct a series of experiments to solve this problem. In this RQ, we want to know how the lack of vulnerability information affects the prediction of vulnerability severity.

Approach. In order to verify the impact of the lack of key aspects on predicting the severity of CVE, we designed 8 experiments, which were used in ablation studies. In this RQ, a certain aspect of ablation means that we completely ignore the ablation aspect in the input stage of the model, even if the CVE describes the ablation aspect, as shown in Fig. 4. We use the method proposed by Han et al. [11] to predict the severity level of security vulnerabilities. We use 1 layer CNN as the neural network model for this prediction work and divide the vulnerability severity into four levels: low, medium, high, and

TABLE V: Performance on vulnerability severity prediction

	Augmented CVE	Original CVE	Ablation of affected product	Ablation of impact	Ablation of Vul-Type	Ablation of root cause	Ablation of attack vector	Ablation of attacker type
Pre	0.796	0.747	0.681	0.711	0.729	0.735	0.715	0.710
Re	0.795	0.749	0.688	0.720	0.728	0.729	0.715	0.709
F1	0.792	0.745	0.682	0.713	0.723	0.726	0.712	0.703

Information Disclosure Vulnerability in Opera 9.64 caused by the improper parsing of XML documents allows remote attackers to cause a denial of service (application crash) via an XML document containing a long series of start-tags with no corresponding end-tags.

Augmented CVE-2009-1234

Opera 9.64 allows remote attackers to cause a denial of service (application crash) via an XML document containing a long series of start-tags with no corresponding end-tags.

Original CVE-2009-1234

Opera 9.64 allows to cause a denial of service (application crash) via an XML document containing a long series of start-tags with no corresponding end-tags.

Ablating attacker type in CVE-2009-1234

Fig. 4: Data comparison

critical. The input of the prediction model is the description of the CVE vulnerability, and the output is the severity level of the vulnerability. We cut out six key aspects in the CVE data set one by one to predict the severity of CVE and observe the impact of the lack of each key aspect on the prediction work. In addition to those six data sets, there are the original CVE data set and the expanded CVE data set. We use the control variable method to compare the experiments on the other seven data sets with those on the original CVE data set. We use the augmented CVE data set to predict the severity of the vulnerability to verify the practicability of our method. We perform 10-fold cross-validation in all experiments.

Results. Table V shows our experimental results. We can see that the vulnerability information completion can improve the prediction of vulnerability severity level, up about 4.7%. Ablating root cause results in a relatively smaller drop (about 1.9%) in F1. For predicting vulnerability severity, ablating affected product have the most significant impact, resulting in a 6.3% drop in F1. Other aspects of ablation decreased by approximately 2.2%-4.2%. It can be seen from the results that our work does have an impact on the prediction of CVE severity grade, and has a good promotion effect.

V. Related Work

Neural networks have been widely-used in natural language processing [15]. A lot of research have been done in predicting vulnerable or error-prone components [16], or assessing how the system is more vulnerable [17]. They used various features, including software indicators, developer activity indicators, and code structure [18]. The difference in our work is that we analyze the vulnerability text and understand the relationship between different aspects in the vulnerability description.

VI. Conclusion

This paper studies the information integrity issues in the vulnerability reports. We separately checked the severity of six key aspects missing from the description of CVE, SecurityFocus, and IBM X-Force Exchange: statistical affected product, root cause, vulnerability type, attacker type, attack vector, and impact. We proposed a customized NER method based on deep neural networks to extract key aspects of vulnerability descriptions and mitigate the missing information in the CVE vulnerability reports. This method uses a neural network model to extract important features from the aspect description and captures the differences in various aspects of the vulnerability descriptions. Our experiments determined the most effective model design for the prediction tasks. We conclude that 1 layer BiLSTM+CharCNN+CRF achieves a better performance in the alternative scheme. We believe that our method has the ability to effectively reduce human efforts and time cost during the updating iterations among different security databases and facilities the vulnerability description-based works in the future.

Acknowledgments

This work has partially been sponsored by the National Science Foundation of China (No. 61872262).

References

[1] S. Chen, L. Fan, G. Meng, T. Su, M. Xue, Y. Xue, Y. Liu, and L. Xu, "An empirical assessment of security risks of global Android banking apps," in *ICSE*. IEEE Press, 2020.

[2] S. Chen, T. Su, L. Fan, G. Meng, M. Xue, Y. Liu, and L. Xu, "Are mobile banking apps secure? what can be improved?" in *ESEC/FSE*. ACM, 2018.

[3] X. Zhan, L. Fan, S. Chen, F. Wu, T. Liu, X. Luo, and Y. Liu, "Atvhunter: Reliable version detection of third-party libraries for vulnerability identification in android applications," in *ICSE*. IEEE Press, 2021.

[4] Z. Tang, K. Tang, M. Xue, Y. Tian, S. Chen, M. Ikram, T. Wang, and H. Zhu, "iOS, your OS, everybody's OS: Vetting and analyzing network services of iOS applications," in *USENIX Security*, 2020.

[5] C. MITRE, "Common vulnerabilities and exposures (cve)," https://cve.mitre.org/, 2019, [Online; accessed 30-June-2019].

[6] ——, "National vulnerability database (nvd)," https://nvd.nist.gov/, 2017, [Online; accessed 21-January-2017].

[7] IBM, "Ibm x-force exchange," https://exchange.xforce.ibmcloud.com/, 2019, [Online; accessed 30-June-2019].

[8] Symantec, "securityfocus," https://www.securityfocus.com/, 2019, [Online; accessed 30-June-2019].

[9] CWE, "Common weakness enumeration (cwe)," http://cwe.mitre.org/, 2019, [Online; accessed 30-June-2019].

[10] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

[11] Z. Han, X. Li, Z. Xing, H. Liu, and Z. Feng, "Learning to predict severity of software vulnerability using only vulnerability description," in *ICSME*, 2017.

[12] X. Gong, Z. Xing, X. Li, Z. Feng, and Z. Han, "Joint prediction of multiple vulnerability characteristics through multi-task learning," in *ICECCS*, 2019.

[13] L. Yuan, Y. Bai, Z. Xing, S. Chen, X. Li, and Z. Deng, "Predicting entity relations across different security databases by using graph attention network," in *In COMPSAC*, 2021.

[14] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2014.

[15] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *CoRR*, vol. abs/1802.05365, 2018.

[16] S. Neuhaus, T. Zimmermann, C. Holler, and A. Zeller, "Predicting vulnerable software components," in *CCS*, 2007.

[17] L. Wang, T. Islam, L. Tao, A. Singhal, and S. Jajodia, "An attack graph based probabilistic security metric," in *Lecture Notes in Computer Science*, 2008.

[18] Y. Shin, A. Meneely, L. Williams, and J. A. Osborne, "Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities," in *TSE*, 2011.