LETTER *Special Section on Log Data Usage Technology and Office Information Systems*

Character-Level Convolutional Neural Network for Predicting Severity of Software Vulnerability from Vulnerability Description

Shunta NAKAGAWA[†a)], **Tatsuya NAGAI**[†], **Hideaki KANEHARA**[††], ***Nonmembers***, **Keisuke FURUMOTO**[††], **Makoto TAKITA**[†], ***Members***, **Yoshiaki SHIRAISHI**[†,†††b)], ***Senior Member***, **Takeshi TAKAHASHI**[††], ***Member***, **Masami MOHRI**[††††], ***Senior Member***, **Yasuhiro TAKANO**[†], ***Member***, ***and*** **Masakatu MORII**[†], ***Fellow***

SUMMARY System administrators and security officials of an organization need to deal with vulnerable IT assets, especially those with severe vulnerabilities, to minimize the risk of these vulnerabilities being exploited. The Common Vulnerability Scoring System (CVSS) can be used as a means to calculate the severity score of vulnerabilities, but it currently requires human operators to choose input values. A word-level Convolutional Neural Network (CNN) has been proposed to estimate the input parameters of CVSS and derive the severity score of vulnerability notes, but its accuracy needs to be improved further. In this paper, we propose a character-level CNN for estimating the severity scores. Experiments show that the proposed scheme outperforms conventional one in terms of accuracy and how errors occur.
key words: *CVE, CVSS, Convolutional Neural Network*

1. Introduction

The vulnerability in software poses a risk of bringing serious losses to individuals and organizations. In order to minimize the risk of vulnerabilities being exploited by attackers, system administrators and security personnels in the organization need to address vulnerabilities of IT assets. Especially, those with severe vulnerabilities need to be addressed promptly.

The Common Vulnerability Scoring System (CVSS) [1] is a framework for scoring the severity of vulnerabilities. It can automatically produce the vulnerability severity score, i.e., CVSS base score, but a user needs to specify the values of several metrics for this by choosing one among predefined values. The uses are, in most cases, security operators within security organizations. Due to the severe shortage of security personnel in present days, it is usual that he needs to find vulnerabilities, analyze them, calculate their severity, prepare vulnerability reports, and register them, in addition to many other security tasks. To aid in their efficient and effective operations, it is desirable to automate the severity estimation operation. By automating the operation, the risk of causing human errors will be also minimized. For instance, the scores produced by human operators may be incorrect, but it is not always possible to thoroughly review the scores before their publications. By using the automation technique can be used as a tool for reviewing the scores produced by human operators. Likewise, the technique will also aid in the operations of insufficient experiences.

Several automation efforts have been reported until now, and one of such efforts was presented by Han *et al.* [2]. They proposed a word-level Convolutional Neural Network (CNN) for estimating the severity level, i.e., CVSS base score, based on Common Vulnerabilities and Exposures (CVE) description [3], which used Word2Vec [4] to represent words in CVE description as vectors. As a result, the prediction accuracy reached 81.6%, but it needs to be improved further.

We take a different approach in this paper. Because new technical terms and software are emerging day by day, we focus on letters rather than words and expect to learn new terms by reflecting the closeness of meaning to existing similar words. Many word-level CNN architectures have been proposed until now, but character-level CNN architectures have also been proposed by Zhang *et al.* [5] and Saxe *et al.* [6]. In this paper, we propose a character-level CNN architecture for estimating severity level from CVE description.

2. Vulnerability Database and Scoring Framework

CVE is a list of entries, each includes CVE identification number (CVE-ID), CVE description, and the reference [https://cve.mitre.org/]. In CVE Details [7], CVE description, CVSS score, and types of threats, etc. are posted. As in Ref. [2], we also extract and use CVE descriptions and CVSS base scores from CVE Details in this study.

CVSS is a mechanism for quantitatively evaluating vulnerabilities. CVSS base score ranges from 0.0 to 10.0. According to Atlassian's security advisory, vulnerabilities are classified into four classes — Critical (CVSS 9.0 - 10.0), High (CVSS 7.0 - 8.9), Medium (CVSS 4.0 - 6.9), or Low (CVSS 0.1 - 3.9) — based on CVSS base score [8]. Likewise, we classify vulnerabilities into four classes.

Manuscript received November 12, 2018.
Manuscript revised April 2, 2019.
Manuscript publicized June 21, 2019.
[†]The authors are with the Department of Electrical and Electronic Engineering, Kobe University, Kobe-shi, 657–8501 Japan.
[††]The authors are with National Institute of Information and Communications Technology, Koganei-shi, 184–8795 Japan.
[†††]The author is with Center for Mathematical and Data Sciences, Kobe University, Kobe-shi, 657–8501 Japan.
[††††]The author is with the Department of Electrical, Electronic and Computer Engineering, Gifu University, Gifu-shi, 501–1193 Japan.
a) E-mail: nakagawa@stu.kobe-u.ac.jp
b) E-mail: zenmei@port.kobe-u.ac.jp
DOI: 10.1587/transinf.2018OFL0006

3. Proposed Character-Level CNN Architecture for Estimating Severity Level

3.1 Overview

Figure 1 shows the procedure of our severity estimation. CVE descriptions and CVSS base scores are used from CVE Details for severity estimation in this paper. CVE descriptions are classified into four classes (Critical, High, Medium, Low) based on CVSS scores. Note that we call these classes severity level. A one-hot encoded vector of each character of CVE description is as an input vector and severity level is as a teacher signal for training CNN. Trained model estimates the severity level from one-hot encoded CVE descriptions.

3.2 One-Hot Encoding

Based on Zhang *et al.*'s method [5], we encode each character using one-hot encoding. We deal with 1014 characters as input and ignore any characters exceeding 1014. If the number of characters in the sentence is less than 1014 characters, fill the blank with zero vector. 69 kinds of target characters are as follows.

abcdefghijklmnopqrstuvwxyz0123456789
-,;.!?:'"/¥|_@#$%^&*~'+-=<>()[]{}

We deal with characters that are not included in this list, including whitespace, as zero vector. In this research, we deal with 69 kinds of characters and zero vector. 70-dimensional one-hot vectors represent the characters in CVE description.

3.3 CNN Architecture

Figure 2 shows the proposed CNN architecture. The architecture has one dense layer to prevent over-fitting. In the dropout layer, the fraction of neurons we keep is set to 0.5.

In the embedding layer, vectors of CVE description are input to learn the 32-dimensional distributed representation. The vector is given to the convolutional layer.

The convolutional layer has four types of filters different in the length (11, 13, 15, 17) to extract the features of the sequence of characters with different lengths. The width of the filter is 32, which is the dimensionality of the character. We prepare 256 filters per type. Threshold ReLU is used as the activation function.

The output of the convolutional layer is input to the 1-max-pooling layer. 1-max-pooling extracts the largest feature in the vector. The output of the 1-max-pooling layer is input to the dropout layer.

All the outputs of the 1-max-pooling layer are concatenated to form a 1024-dimensional vector, and this vector is input to the dropout layer. Then, the output of the dropout layer is input to the dense layer with 1024 units. Threshold ReLU is used as the activation function. The output of the dense layer is input to the dropout layer.

The output layer is the dense layer with 4 units. We

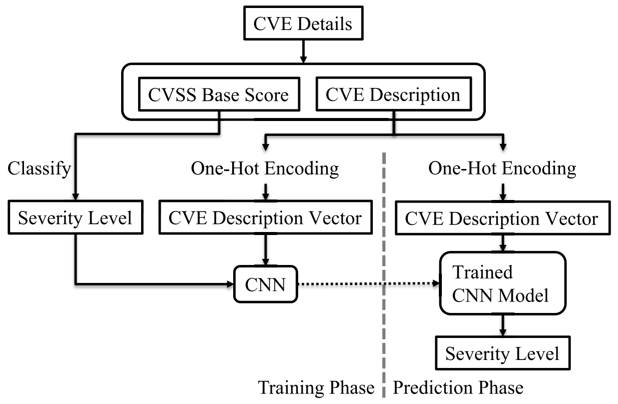

Fig. 1 The procedure of severity estimation.

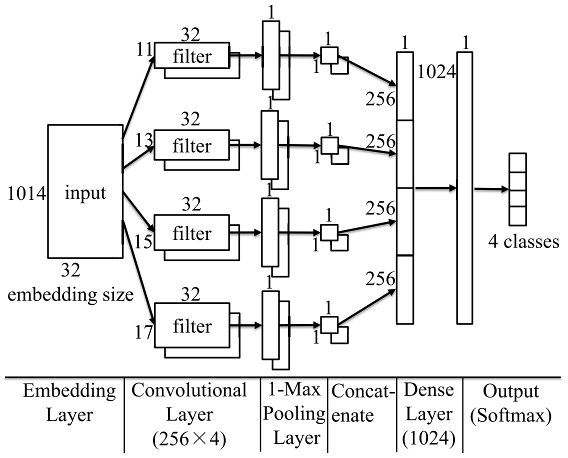

Fig. 2 Proposed CNN architecture.

use L2 regularization. Softmax is generally used in multi-class single-label classification, so we also use Softmax as the activation function. The output represents severity level.

This model uses Adam update algorithm as the optimizer and categorical cross entropy as the loss function.

4. Experiment

We compare our character-level CNN architecture with a word-level CNN architecture and other two character-level CNN architectures. We use CVSS base scores and CVE descriptions extracted from CVE Details.

4.1 Preliminary Experiment

We carried out preliminary experiment to test which is better, 1-max-pooling or average-pooling in our CNN. It was to compare proposed character-level CNN using 1-max-pooling with proposed one using average-pooling in before Oct. 16. 2018 dataset. We set the batch size to 32. We build a model for each epoch and calculate validation accuracy. We stop learning if the validation accuracy does not improve any further. And we adopt the model whose validation accuracy was the best among the built models. The highest val-

Table 1 The number of vulnerabilities as of 31st March 2017.

Severity Level	CVSS	Number
Critical	9.0 - 10.0	12,286
High	7.0 - 8.9	20,722
Medium	4.0 - 6.9	43,503
Low	0.1 - 3.9	6,213
Total	0.1 - 10.0	82,724

idation accuracies of proposed character-level CNN using 1-max-pooling and average pooling were same 74.76%. At that time, the learning time of proposed character-level CNN using 1-max-pooling is smaller than when using average-pooling. Therefore, in the experiment in Sect. 4.3, we adopt 1-max-pooling.

4.2 Used Dataset and Architectures in Experiment

As of 31st March 2017, CVE Details has 82,772 vulnerabilities. Among them, we discard the vulnerabilities with CVSS base score 0, which have not been evaluated yet.

We use three architectures to compare our character-level CNN with word-level CNN and other character-level CNN except for our CNN architecture.

The first architecture is word-level CNN, based on Han *et al.*'s architecture [2]. We use Softmax as the activation function in the output layer and categorical cross entropy as the loss function. After training CNN, we extract the input to the last layer as the feature vector. We use this as input to SVM and make predictions.

The other two architectures are character-level CNN. Like the proposed, we represent characters in distributed expression in 32 dimensions.

The one is based on the architecture of Zhang *et al.* (Zhang's character-level CNN [5]) and the other one is based on the architecture of Saxe *et al.* (Saxe's character-level CNN [6]). In Saxe's character-level CNN, we set the size of the filters of the convolution layer to (5, 7, 9, 11) and use Softmax as the activation function in output layer and categorical cross entropy as the loss function.

4.3 Methods and Results

In the following experiments (A) and (B), we used the model built by the method described in Sect. 4.2.

(A) Experiment with randomly sampled data

In order to match the number of severity classes, we randomly select 6,213 vulnerabilities from each "Critical", "High", and "Medium" vulnerabilities. We use 80% vulnerabilities of each severity level as training data, 10% as validation data, and 10% as test data.

The architectures are compared with the view of accuracy and how errors occur. Accuracy is defined as correct predictions among the total number of vulnerabilities in the dataset. Table 2 shows the results of comparative experiments with other architectures in accuracy. Saxe's character-level CNN is the best and proposed character-level CNN is second in terms of accuracy, but it seems that the difference among all architectures is small. Because it is not still clear which architecture is better, we compare the top three architectures by using confusion matrix.

Table 2 Accuracy of all architectures.

Architecture	Accuracy
word-level CNN [2]	0.6961
word-level CNN+SVM [2]	0.7081
Zhang's character-level CNN [5]	0.6771
Saxe's character-level CNN [6]	0.7310
Proposed character-level CNN	0.7250

Table 3 Confusion matrix of word-level CNN+SVM [2].

		True Label			
		Critical	*High*	*Medium*	*Low*
Predicted Label	*Critical*	**504**	130	47	13
	High	63	**360**	89	16
	Medium	44	102	**377**	74
	Low	10	29	108	**518**

Table 4 Confusion matrix of Saxe's character-level CNN [6].

		True Label			
		Critical	*High*	*Medium*	*Low*
Predicted Label	*Critical*	**490**	98	38	8
	High	81	**400**	83	15
	Medium	41	102	**422**	94
	Low	9	21	78	**504**

Table 5 Confusion matrix of proposed character-level CNN.

		True Label			
		Critical	*High*	*Medium*	*Low*
Predicted Label	*Critical*	**505**	108	43	9
	High	80	**401**	105	18
	Medium	27	88	**385**	84
	Low	9	24	88	**510**

Tables 3, 4 and 5 are confusion matrices of word-level CNN+SVM, Saxe's character-level CNN, and proposed character-level CNN, respectively. Each number of one level error is 566 (22.79%) in word-level CNN, 536 (21.58%) in Saxe's character-level CNN and 553 (22.26%) in proposed character-level CNN. Each number of two level error is 136 (5.48%) in word-level CNN, 115 (4.63%) in Saxe's character-level CNN and 112 (4.51%) in proposed character-level CNN. Each number of three level error is 23 (0.93%) in word-level CNN, 17 (0.68%) in Saxe's character-level CNN and 18 (0.72%) in proposed character-level CNN. Each number of three level error of Saxe's character-level CNN and proposed character-level CNN are much the same and smaller than word-level CNN. Note that mistaking "Critical" as lower level is not preferable, for example, "Critical"→"High" and "Critical"→"Medium", especially "Critical"→"Low". In the case where "Critical" is mistaken as lower level, the number of Saxe's character-level CNN and proposed character-level CNN are as below: Although the numbers of mistaking "Critical" as "Low" is same as 9 (1.45%) in both architectures, each number of mistaking "Critical" as "Medium" is 41 (6.60%) and 27 (4.35%), and each number of mistaking "Critical" as "High" is 81 (13.04%) and 80 (12.88%), in Saxe's character-level CNN

Table 6 Results of word-level CNN+SVM [2] with the dataset separated by periods.

Testing Dataset	Accuracy	Overall Precision	Overall Recall	Overall F1-measure
Before-2016	0.695	0.692	0.695	0.693
Jan-Mar, 2016	0.632	0.624	0.632	0.620
Apr-Jun, 2016	0.614	0.595	0.614	0.596
Jul-Sep, 2016	0.640	0.626	0.640	0.620

Table 7 Results of proposed character-level CNN with the dataset separated by periods.

Testing Dataset	Accuracy	Overall Precision	Overall Recall	Overall F1-measure
Before-2016	0.697	0.699	0.697	0.697
Jan-Mar, 2016	0.662	0.655	0.662	0.656
Apr-Jun, 2016	0.648	0.636	0.648	0.632
Jul-Sep, 2016	0.662	0.653	0.662	0.652

and proposed character-level CNN, respectively. From the above experimental results, proposed character-level CNN is considered to be able to predict "Critical" more accurately than other methods.

(B) Experiment with the dataset separated by time

We used before-2016 dataset and 2016 dataset to evaluate the estimation of new vulnerability description's severity level. First, as dataset before-2016, we build training dataset-2015, validation dataset-2015 and test dataset-2015 by using the data discovered before Dec 31, 2015. We choose 10,000 (2,500 for each severity level) vulnerabilities as before-2016-training dataset, 4,000 (1,000 for each severity level) vulnerabilities as before-2016-validation dataset, 600 (150 for each severity level) vulnerabilities as before-2016-test dataset by using the data discovered before Dec 31, 2015. Second, we create test datasets for the three periods of 2016. We made three datasets by choosing 500 (125 for each severity level) vulnerabilities from the data in Jan 1-Mar 31, Apr 1-Jun 30, and Jul 1-Sep 30, 2016. These three datasets of 2016 are used as test datasets to evaluate model learned using before-2016-training dataset and before-2016-validation dataset.

Tables 6 and 7 show the result of the test accuracy, when using word-level CNN+SVM [2] and proposed character-level CNN in this experiment. When using before-2016-test dataset, each test accuracy of the two architectures is almost the same. However, when using three test datasets of 2016, the test accuracy of the proposed character-level CNN exceeds the accuracy on the word-level CNN+SVM in all three test datasets. It is considered that proposed character-level CNN can estimate the severity level of unknown vulnerability reflecting similarity of known character string.

5. Conclusion

In this paper, we studied to find CNN architecture to estimate severity level from the description of vulnerabilities. System administrators and security officials of an organization need to deal with vulnerable IT assets, especially those with severe vulnerabilities. To aid in their efficient and effective operations, it is desirable to automate the severity estimation operation. Furthermore, estimating new vulnerability description's severity level is particularly important because new technical terms and software are emerging day by day. So, we focused on letters rather than words and proposed character-level CNN for estimating severity level.

In order to evaluate the proposed architecture, we constructed a word-level CNN and two other character-level CNN architectures for comparison. As the results of experiment (A) and (B), the proposed character-level CNN architecture is the best in these architectures for estimation of severity level and it is considered that proposed character-level CNN can estimate the severity level of unknown vulnerability reflecting similarity of known character string.

References

[1] FIRST, "A Complete Guide to the Common Vulnerability Scoring System," https://www.first.org/cvss/v2/guide, accessed Nov. 11. 2018.

[2] Z. Han, X. Li, Z. Xing, H. Liu, and Z. Feng, "Learning to Predict Severity of Software Vulnerability Using Only Vulnerability Description," 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), Shanghai, China, pp.125–136, Sept. 2017.

[3] MITRE Corporation, "Common vulnerabilities and exposures (cve)," https://cve.mitre.org/, accessed Oct. 16. 2018.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv preprint, arXiv: 1301.3781, Sept. 2013.

[5] X. Zhang, J. Zhao, and Y. LeCun, "Character-level Convolutional Networks for Text Classification," arXiv preprint, arXiv: 1509.01626, Sept. 2015.

[6] J. Saxe and K. Berlin, "eXpose: A Character-Level Convolutional Neural Network with Embeddings For Detecting Malicious URLs, File Paths and Registry Keys," arXiv preprint, arXiv: 1702.08568, Feb. 2017.

[7] CVE Details, "CVE details," http://www.cvedetails.com/, accessed Oct. 16. 2018.

[8] A.S. Advisories, "Severity levels for security issues," https://www.atlassian.com/trust/security/security-severity-levels, accessed Oct. 28. 2018.