IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 22, 2019, accepted October 8, 2019, date of publication October 21, 2019, date of current version October 30, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2947527

SQL Injection Detection for Web Applications Based on Elastic-Pooling CNN

XIN XIE, CHUNHUI REN, YUSHENG FU, JIE XU, AND JINHONG GUO
School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
Corresponding authors: Chunhui Ren (chuiren@uestc.edu.cn) and Jinhong Guo (guojinhong@uestc.edu.cn)

ABSTRACT An enterprise's data can be one of its most important assets and often critical to the firm's development and survival. SQL injection attack is ranked first in the top ten risks to network applications by the Open Web Application Security Project (OWASP). Its harmfulness, universality, and severe situation are self-evident. This paper presents a method of SQL injection detection based on Elastic-Pooling CNN (EP-CNN) and compares it with traditional detection methods. This method can output a fixed two-dimensional matrix without truncating data and effectively detects the SQL injection of web applications. Based on the irregular matching characteristics, it can identify new attacks and is harder to bypass.

INDEX TERMS Deep learning, neural network, CNN, network security, SQL injection.

I. INTRODUCTION

As an important asset of an enterprise, the value of data increases year by year. When enterprise data is leaked, not only economic benefits but also public prestige will be lost, which will lead to the distrust of employees and users, a large number of users will be lost, and some even face legal proceedings, triggering high-level shocks and so on. With the development of the Internet, more and more web applications have emerged. Many web applications collect users' personal information and interact with users. So they always connect to the database. Due to the large amount of valuable data stored in the database, it naturally becomes the target of attackers, so there are more and more SQL injection attacks.

SQL injection attack refers to the construction of special strings as parameters to be transmitted to web applications by submitting web forms or inputting query strings of domain names or page requests. These special strings often contain some executable statements in the SQL grammar, which make web applications mistake data as code to execute, and ultimately deceive servers to execute malicious SQL commands. Its main reason is the web application does not filter the users' input data accurately, so the database is invaded.

Existing methods for identifying SQL injection attacks include regular matching [1], Support Vector Machine (SVM) technology [2], [3], Decision Tree [4], [5], Naive Bayes [4], [6], etc. Among them, regular matching is widely used, with good effect, it have fast recognition speed and high accuracy rate.

Convolutional Neural Network (CNN) [7] is a kind of deep feedforward neural network, which imitates the formation mechanism of visual cognition of organisms. It can be used for supervised learning and unsupervised learning. In recent years, CNN has been widely used in computer vision and natural language processing because of its stable effect on the learning of pixels and audio and no additional feature engineering requirements for data. This paper presents a method of SQL injection detection based on CNN and massive web logs. In this paper, CNN is applied to the detection of SQL injection in Web applications, and SQL injection attacks are detected from massive web logs. Practical results show that this method has good effect and high recognition accuracy.

Distinguish from the drawbacks of CNN used in image recognition, such as putting the mouth on the forehead can also be recognized as a face, but the valid fields of SQL injection can exist anywhere in the string, thus avoiding the drawbacks of CNN. By preprocessing methods and improving the pooling layer, data information can be effectively retained.

II. RELATED WORK

SQL injection attack is ranked first in the top ten risks to network applications by the Open Web Application Security Project (OWASP) [8] and has been a consistent focus of network security research. In recent years, researchers have proposed a variety of methods for detecting SQL injection by

The associate editor coordinating the review of this manuscript and approving it for publication was Kim-Kwang Raymond Choo.

studying the threats, attack types, and attack modes [9]. The standardization of SQL query statements [10] is a common preprocessing method that unifies SQL keywords, symbols, and query parameter values. There are many ways to regulate, one of which is as Fig. 1.

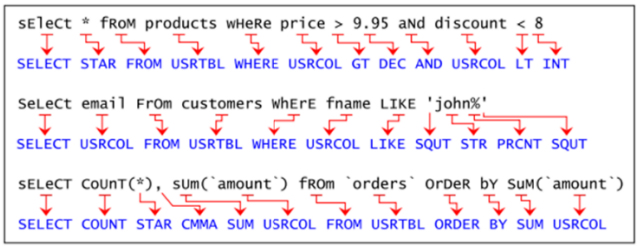

FIGURE 1. An example of normalization of SQL query statements [10].

The disadvantage of this method is also obvious: it cannot be standardized directly in the URL. The characters and codes of SQL injection statements in the production environment are confusing and complex, and it is difficult to standardize them directly. Some typical SQL injection statements in the production environment are as Fig. 2.

FIGURE 2. Examples of some typical SQL injection query strings in production environments.

According to a certain way of query sentence segmentation, statistics of the frequency of each word [4] is also a way of data preprocessing. This method splits the relationship between words, resulting in a large scale of loss of sentence information, affecting the subsequent classification effect. Based on this, a better feature selection is to count special characters and special statements [11]. After vectorizing the extracted features, the traditional machine learning method is used for classification and recognition. Naive Bayes [4], [6], Decision Tree [4], [5], and Support Vector Machine (SVM) [2], [3] are commonly used. There are many detection methods that do not use machine learning and that include classification technology based on user behavior [12], [13] and expectation criteria [14]. Regular expression [1] is the most convenient, effective, and fast detection method used by the traditional WAF and by most internet security companies. This method requires predefined regular expressions. However, due to the diversity of SQL grammar and user input, regular expressions cannot meet the detection requirements and are unable to identify new attacks. Detecting the SQL injection attack by removing the parameter value of the SQL query [15] is fast, but it is easily bypassed. Hidden Markov Model (HMM) [16] detects SQL injection by establishing the browsing behavior model of attackers and legitimate users, and has common shortcomings with the detection method based on user behavior [12], [13], which is more suitable for the detection of a single website. The database table is expanded [17] or redesigned to monitor the query behavior of the database to determine whether it belongs to abnormal behavior. This method can handle any type of query and the algorithm is platform-independent. Other SQL injection detection methods include dynamic query matching technology [18], mapping model from URL to SQL features [19]. For the detection method of second-order SQL injection, a new set of SQL instructions [20] is constructed dynamically and then detected with a database and proxy server.

Traditional detection methods for SQL injection have different shortcomings and their classification performance is not good enough. Convolutional Neural Network (CNN) [7] is a kind of deep feedforward neural network that imitates the formation mechanism of visual cognition in organisms. CNN has a very good performance in computer vision and voice processing. This paper presents a method of SQL injection detection based on Elastic-Pooling CNN (EP-CNN) and massive web logs, and CNN is improved and applied to the detection of SQL injection in Web applications. Practical results show that this method has good effect and high recognition accuracy.

III. PROPOSED MODEL

There are various forms of SQL injection and new attack methods inevitably appear to bypass traditional detection. Aiming at the problem of universality, this paper proposes a method of SQL injection detection based on Elastic-Pooling CNN (EP-CNN). This method mainly solves the problem of identifying SQL injection attacks that cannot be recognized by traditional methods. It can be used as an auxiliary method of traditional SQL detection methods.

A. CORE IDEAS

Query statements of SQL injection often have a lot of redundant information, so it matches the characteristics of CNN that can remove redundant information. It is theoretically feasible to select CNN for training and recognition.

There is no obvious distinction between normal query statements and query statements of SQL injection, that is to say, it is impossible to artificially provide good classification features, and bad features will lead to a large number of errors in recognition. The differences between them are often not obvious features, which can not be observed. CNN can automatically extract features that can not be recognized by

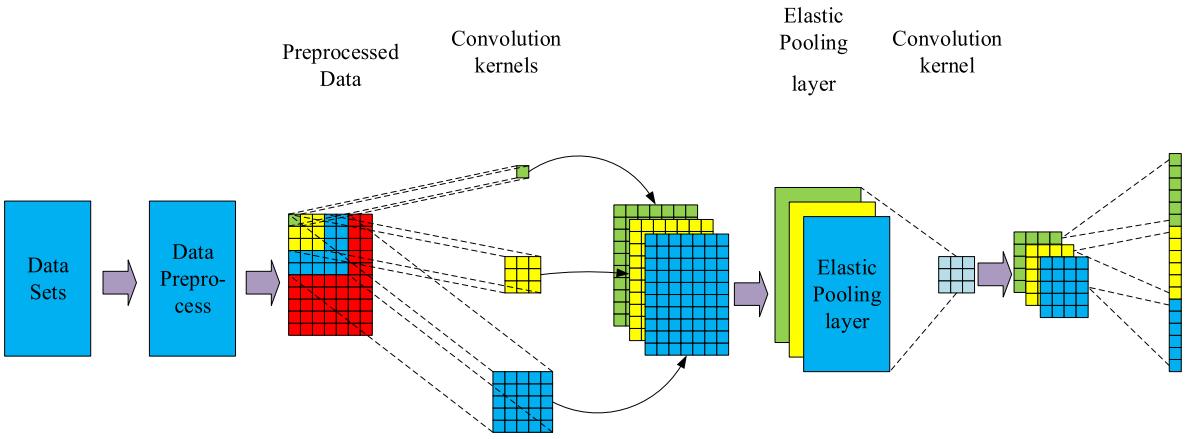

FIGURE 3. Model schematic.

human. Therefore, CNN is used to extract common features that are not obvious.

The training of CNN needs a lot of data, so the premise of using CNN is massive log, which can avoid over-fitting and enhance robustness.

Maximum pooling has translation invariance and is robust to deformation. The injection point of the SQL injection statement can also exist anywhere in the string and has translation invariance.

Based on the above four ideas and with data preprocessing techniques, we can accurately identify the behavior of SQL injection through query statements.

B. OVERALL MODEL

The overall structure of Elastic-Pooling CNN is similar to that of traditional CNN. The overall model is shown in Fig. 3.

After data preprocessing, three convolution kernels of different sizes are used for padding convolution for preprocessed data with different sizes. The convolution kernels' sizes are 1×1, 3×3 and 5×5, respectively. Padding convolution keeps the number of rows and columns of a matrix remains unchanged; then, input to the elastic pooling layer and output the same size matrix. After that, convoluted with a 3×3 convolution kernel. Finally, the convoluted output dimensionality reduction is spliced into a one-dimensional vector.

C. ELASTIC-POOLING CNN

TEXT-CNN [21] and SPP-NET [22] can output fixed size of one-dimensional vectors when non-fixed size matrices input to them. These methods can realize text classification of different lengths and image recognition of different sizes, but the output is a one-dimensional vector. Inspired by TEXT-CNN [21] and SPP-NET [22], in order to create a matrix with a different number of rows but fixed columns that have the same dimension output, we improve the max pooling layer. And with data preprocessing, the same dimension output can be obtained without truncating the query strings. In this way, the convolution layer can be added later.

1) DATA PREPROCESSING

Word2vec method is used to vectorize each original query string and the single character is used as the minimum training unit as shown in Fig. 4. Then the query string is converted to a matrix. Then add 0-vector rows to the end of the matrix that make the number of rows of the matrix is an integer multiple of the gradient value and it is at least twice as much as the gradient value. For example, if the number of rows is less than two times the gradient value, then add 0-vector rows to the end of the matrix until the rows is two times the gradient value; if the number of rows is greater than two times and less than three times the gradient value, then add 0-vector rows until the number of rows is three times the gradient value, and so on. The number of rows corresponding to each query string is as formula (1).

$$h = \left\lfloor \frac{l}{p} \right\rfloor \bullet p \tag{1}$$

where h represents the number of rows of the matrix corresponding to the data, l represents the length of the data, p represents the gradient value, and $\lfloor\ \rfloor$ represents the upward integration. Each data point needs to fill in $(h - l)$ 0-vector.

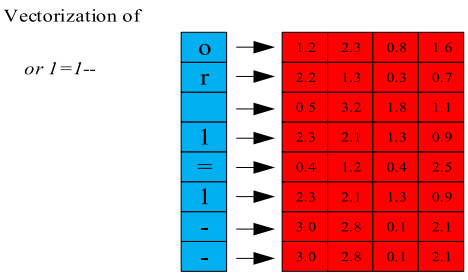

FIGURE 4. Word vectorization schematic.

2) ELASTIC POOLING LAYER

The schematic diagram of the EP layer is as shown in Fig. 5.

Because the number of rows is difference between the input matrices of the pooling layer, and it is an integer multiple

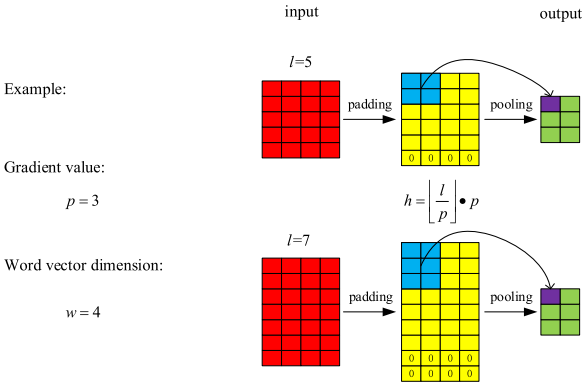

FIGURE 5. Schematic diagram of elastic pooling layer.

of the gradient value and the number of columns is fixed, the number of rows of the pooling kernel is equals the number of rows of the input matrix divided by the gradient value, and the number of columns of the pooling kernel is fixed as 2; thus, the matrix dimension of the output is fixed.

This layer can be added behind the convolution layer. If multiple convolution cores of different sizes are used for convolution, then padding convolution must be used to keep the input and output dimensions unchanged. This method retains all the information of the original query strings, but this layer greatly reduces the huge computational overhead.

3) TRAINING MODEL

Step 1: Disturb the marked positive and negative samples. Since all samples are imported into the training in batches, disturbing the positive and negative samples can prevent the whole training batch from being full of positive or negative samples.

Step 2: Randomly select some labeled positive and negative samples as training set and the rest as test set.

Step 3: Set up CNN network for training. Generally, the more training data, the better, but the training cost will rise.

Step 4: Test the trained model through the test set. If the error (correctness, loss function value, etc.) is within acceptable range, the training will stop. Otherwise, the training will continue by adjusting the parameters of the neural network, including CNN structure, network layers, training times, convolution kernel size, pooling function and so on. Generally, the classification accuracy of the training model is more than 99%.

Then deploying the model into the server environment can detect the SQL injection attacks based on query statements in real time, and can also detect historical web logs.

IV. EXPERIMENT

The code we used to train and evaluate our models is available at *https://github.com/uestcer-xx/EP-CNN*. Since the data is provided by a commercial company, it involves the company's business secrets. Therefore, these data should not be made public. But we uploaded 100 positive and 100 negative samples for reference.

A. DATA SETS

About 4.48 million real web logs in the production environment are used, of which about 1.28 million are SQL injection attack logs and 3 million normal logs. All logs are URL access records marked with query statements. Including normal and SQL injection logs, such as "http://www.example.com/path1/path2/?query#fragment" URL with "query" parameters; that is, the string after "?" and before "#" (in general, the format is key1 = value1 & key2 = value2...). 200,000 samples were randomly selected as training sets (about 100,000 positive and negative samples each) and the rest were used as testing sets. There was no intersection between the test and training sets.

The experimental equipment is a notebook computer and the programming language is Python3. The keras framework based on tensorflow-gpu is used and the GPU is NVIDIA GTX1050Ti.

B. EXPERIMENTAL RESULT

1) EP-CNN

The loss curve is plotted in Fig. 6. The value of the loss tends to be stable with the increase of training times and converges at about 0.005.

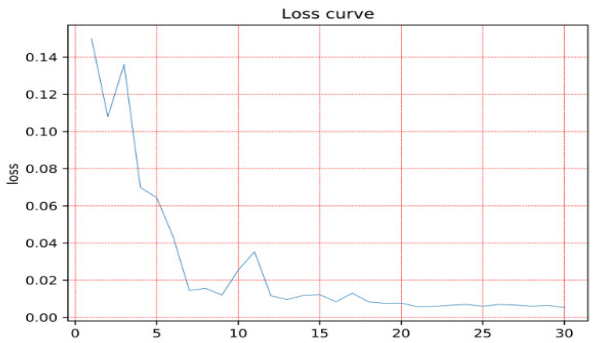

FIGURE 6. Loss curve.

As shown in Table 1, the confusion matrix shows that the TP and TN values are both much higher than FP and FN.

TABLE 1. Confusion matrix.

Actual Class	Predicted Class	
	Positive	Negative
Positive	TP = 2998374	FN = 1626
Negative	FP = 1115	TN = 1282200

As shown in Table 2 and Fig. 7, the accuracy, precision, recall, F1, and AUC are higher than 0.999, demonstrating that

TABLE 2. Performance index.

Performance Index	Value
Accuracy=(TP+TN)/(TP+FP+FN+TN)	0.999360
Precision=TP/(TP+FP)	0.999628
Recall=TP/(TP+FN)	0.999458
F1=2*(Recall * Precision) / (Recall + Precision)	0.999543
AUC	0.999998

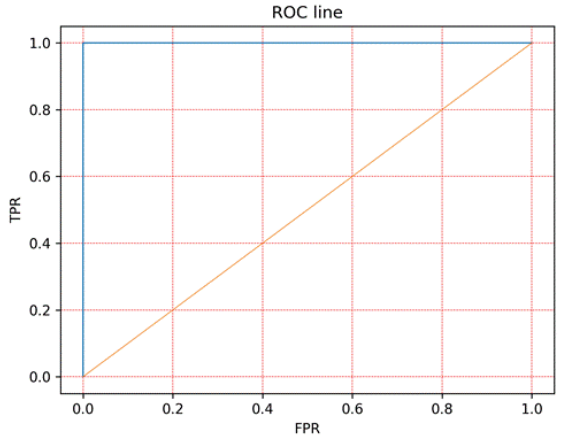

FIGURE 7. ROC line.

the classification performance of this model is very good. The ROC line is even close to a perfect rectangle.

Examples of FP and FN query strings are shown in Table 3 and Table 4.

TABLE 3. Examples of FP.

id=966092	*id=950860*
id=5588	*id=890406*
id=4876	*id=5211*
id=949486	*id=1216*
id=942516	*id=5802*
id=30506	*page=23*
id=1810	*id=69722*
id=74136	*id=1914*
id=58	*id=922156*

TABLE 4. Examples of FN.

dh=-1'%20OR%203%2b393-393-1=0%2b0%2b0%2b1%20--%20
dicttblid=%25%27%20anD%202189%3D2189%20%2F%2A
cid=-1_2&navtypeid=6&page=6&priceFrom=&priceTo=&sort=sm_price&sorttype=asc&status=-1&typeid=-1'"%20gt8m=Ef37([!%2b!])%20rdi="
v=245513055ea63aa7879db0e2e0c7eede%29%20AND%20SLEEP%285%29%20AND%20%285715%3D5715
*code=C823E8B8B80A6438653E354BB565B4A8au2&type=%5e(%23%24!%40%23%24)(()))*******
desc=ad&nt=-1'%20OR%202%2b816-816-1=0%2b0%2b0%2b1%20--%20&q=s&wd=%EF%BF%BD%EF%BF%BD

From the examples of FP and FN, it is obvious that these data are mislabeled data, but they are recognized by the model. Thus, the model has good generalization performance.

2) OTHER METHODS

For comparison, traditional machine learning methods are used to classify and compare the accuracy and F1 values, including SVM [2], [3], Naive Bayes [4], [6], Decision Tree [4], [5], and Random Forest [23], all of the training sets for these traditional machine learning methods is randomly selected from the training sets of EP-CNN. After many tests, we use 10000 query strings to train SVM and Naive Bayes model, and 100000 query strings to train Random Forest and Decision Tree model. The input data of these methods are one-dimensional vectors, so we trained another word vector model and then convert the query string to a one-dimensional vector. The experimental equipment for these methods is a notebook computer and the programming language is Python3. As can be seen from Table 5, the accuracy and F1 value of EP-CNN are higher than those of other methods. But the training time and testing time is higher than others too, that is because EP-CNN and CNN have more computational load in the testing process. In the traditional machine learning methods of SVM [2], [3], Naive Bayes [4], [6], Decision Tree [4], [5], and Random Forest [23], the effect of Naive Bayes [6], [4], is much worse than that of the other methods because it is based on the conditional independence hypothesis; that is, the position correlation information between characters is split. However, SQL injection is closely related to location correlation between characters. The performance of traditional CNN is also very good, second only to EP-CNN. Compared with CNN, EP-CNN does not truncate query statements and retains all information; hence, the classification effect is better than CNN. For the training time and testing time per query string,

TABLE 5. Comparison of different machine learning methods.

Method	Training Set acc	Testing Set acc	Training Time	Testing Time per query string
SVM	0.97840	0.94490	14.0 s	1.13 ms
Naive Bayes	0.67470	0.67628	**4.9 s**	0.48 ms
Decision Tree	0.98642	0.97013	52.3 s	0.48 ms
Random Forest	0.99932	0.98700	60.3 s	**0.47 ms**
CNN	0.99950	0.99413	44320 s	4.56 ms
EP-CNN	**0.99980**	**0.99936**	39780 s	4.41 ms

SVM model has two very important parameters c and γ. c is the penalty coefficient, that represents the tolerance of errors. The higher the c is, the more intolerable the error and easy to over-fit. The smaller c, the less fitting. The c is too large or too small, and its generalization ability becomes worse.

The γ is a parameter of RBF function (Gauss kernel) which is selected as the kernel function. Implicitly determines the distribution of data after mapping to a new feature space. The larger the γ value is, the fewer the support vectors, the smaller the γ value is and the more the support vectors. The number of support vectors affects the speed of training and prediction.

The smaller the γ value is, the more continuous the classification interface is, the bigger the γ value is, the more scattered the classification interface is, the better the classification effect is, but it may be over-fitting. When $c = 0.8$ and $\gamma = 0.3$, the maximum of accuracy is 94.49%. At this time, the accuracy of training data is 97.84%.

The training time of SVM is short, the amount of data needed is small, and the model is simple. It is very sensitive to kernels and parameters, and there is no general solution for non-linear problems. Classification is less effective than in-depth learning when the amount of data is large.

Naive Bayes model is as formula (2).

$$p\left(c|x\right) = \frac{p(c)p(x|c)}{p(x)} = \frac{p(x,c)}{p(x)} \tag{2}$$

Naive means the assumption of conditional independence of attributes the assumption of conditional independence of attributes as formula (3).

$$p\left(c|x\right) = \prod_{i=1}^{d} p(x_i|c) \tag{3}$$

According to the prior probability distribution (i.e. input data distribution), it can be generally divided into: Gauss Naive Bayes, Naive Bayesian polynomial distribution and Bernoulli Naive Bayes.

Based on the assumptions of independence between independent variables (conditional feature independence) and normality of continuous variables, the algorithm accuracy will be affected to some extent, and the classification accuracy will be directly affected.

Decision Tree has three choices of partition: information entropy, gain rate and Gini index. Information entropy means reduction of entropy after partitioning by an attribute. Gain rate is similar to information entropy, it only becomes a percentage of the reduction of information entropy. Gini index is the probability of inconsistent labeling of two samples randomly selected. Using information entropy as partition selection, the maximum depth of the tree is 15. The accuracy of training set is 98.6420%. Test accuracy of testing set is 97.0130%.

For Random Forest, Decision Tree is the based learners, the simple understanding of Random Forest is to train several almost unrelated weak learners and to determine classification by voting. The traditional decision tree chooses an optimal attribute in the attribute set of the current node when choosing partitioning attributes, in the random forest, for each node of the base decision tree, a subset containing k attributes is randomly selected from the attribute set of the node, and then an optimal attribute is selected from the subset for partitioning. Commonly $k = \log_2 d$, d denotes the number of attributes. Using 101 base decision trees, maximum depth 20. Accuracy rate of training set is 99.9320%. Test set accuracy: 98.7000%

The forms of SQL injection are various, and new methods will inevitably appear to bypass the regular matching technology. At the same time, it is difficult to identify new attacks based on existing rules. Traditional machine learning techniques (SVM, Bayesian, decision tree and other statistical learning methods) need artificial features. It is not easy to get the best features and to judge whether they are the best features and whether there are better features, which results in low recognition accuracy and a large number of false positives. The quality of feature engineering greatly affects the final recognition accuracy.

Traditional machine learning methods need to select features artificially, but good features are not easy to obtain. The quality of features directly affects the recognition accuracy. Some deep learning recognition is very troublesome for URL preprocessing and will lose some important information, such as decoding url. Chinese characters may appear. If not removed, it will lead to a large increase in the vocabulary. If not removed, some of the important feature information injected by SQL will be reduced.

V. CONCLUSION

Existing SQL injection detection mainly uses regular matching. Regular technology has high recognition accuracy and speed, but it cannot identify new attacks. It is inevitable that new bypassing methods will emerge to avoid rules such as URL multiple encoding. The recognition of SQL injection based on EP-CNN automatically extracts the hidden common features of SQL injection and identifies the attack traffic, bypassing the regular SQL injection, which is fast. It is vectorized based on a single character and the vocabulary is small, but all the credentials of the query statements can be retained. The training difficulty and cost can be reduced if the vocabulary is small. A good model can completely replace the original regular recognition method and update the model in real time. Next we will study the subdivisions of attack types and implement a multi-classification model that is not limited to the identification of SQL injection attacks.

REFERENCES

[1] M. Qbea'h, M. Alshraideh, and K. E. Sabri, "Detecting and preventing SQL injection attacks: A formal approach," in *Proc. Cybersecur. Cyberforensics Conf. (CCC)*, Aug. 2016, pp. 123–129.

[2] S. O. Uwagbole, W. J. Buchanan, and L. Fan, "Applied machine learning predictive analytics to SQL injection attack detection and prevention," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, May 2017, pp. 1087–1090.

[3] P. R. McWhirter, K. Kifayat, Q. Shi, and B. Askwith, "SQL injection attack classification through the feature extraction of SQL query strings using a gap-weighted string subsequence kernel," *J. Inf. Secur. Appl.*, vol. 40, pp. 199–216, Jun. 2018.

[4] M. Lodeiro-Santiago, C. Caballero-Gil, and P. Caballero-Gil, "Collaborative SQL-injections detection system with machine learning," in *Proc. 1st Int. Conf. Internet Things Mach. Learn.*, 2017, Art. no. 45.

[5] B. Hanmanthu, B. R. Ram, and P. Niranjan, "SQL injection attack prevention based on decision tree classification," in *Proc. IEEE 9th Int. Conf. Intell. Syst. Control (ISCO)*, Jan. 2015, pp. 1–5.

[6] J. Santoso, E. M. Yuniarno, and M. Hariadi, "Large scale text classification using map reduce and Naive Bayes algorithm for domain specified ontology building," in *Proc. 7th Int. Conf. Intell. Hum.-Mach. Syst. Cybern.*, vol. 1, Aug. 2015, pp. 428–432.
[7] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," 2019, *arXiv:1901.06032*. [Online]. Available: https://arxiv.org/abs/1901.06032
[8] *The Ten Most Critical Web Application Security Risks*, Top OWASP 10, Toronto, ON, Canada, 2013.
[9] N. Singh, M. Dayal, R. S. Raw, and S. Kumar, "SQL injection: Types, methodology, attack queries and prevention," in *Proc. 3rd Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, Mar. 2016, pp. 2872–2876.
[10] D. Kar, S. Panigrahi, and S. Sundararajan, "SQLiGoT: Detecting SQL injection attacks using graph of tokens and SVM," *Comput. Secur.*, vol. 60, pp. 206–225, Jul. 2016.
[11] K. Kamtuo and C. Soomlek, "Machine Learning for SQL injection prevention on server-side scripting," in *Proc. Int. Comput. Sci. Eng. Conf. (ICSEC)*, Dec. 2016, pp. 1–6.
[12] S. M. Darwish, "Machine learning approach to detect intruders in database based on hexplet data structure," *J. Elect. Syst. Inf. Technol.*, vol. 3, no. 2, pp. 261–269, 2016.
[13] Z. Xiao, Z. Zhou, W. Yang, and C. Deng, "An approach for SQL injection detection based on behavior and response analysis," in *Proc. IEEE 9th Int. Conf. Commun. Softw. Netw. (ICCSN)*, May 2017, pp. 1437–1442.
[14] L. Xiao, S. Matsumoto, T. Ishikawa, and K. Sakurai, "SQL injection attack detection method using expectation criterion," in *Proc. 4th Int. Symp. Comput. Netw. (CANDAR)*, Nov. 2016, pp. 649–654.
[15] R. A. Katole, S. S. Sherekar, and V. M. Thakare, "Detection of SQL injection attacks by removing the parameter values of SQL query," in *Proc. 2nd Int. Conf. Inventive Syst. Control (ICISC)*, Jan. 2018, pp. 736–741.
[16] P. Li, L. Liu, J. Xu, H. Yang, L. Yuan, C. Guo, and X. Ji, "Application of hidden Markov model in SQL injection detection," in *Proc. IEEE 41st Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 2, Jul. 2017, pp. 578–583.
[17] A. Ghafarian, "A hybrid method for detection and prevention of SQL injection attacks," in *Proc. IEEE Comput. Conf.*, Jul. 2017, pp. 833–838.
[18] R. P. Karuparthi and B. Zhou, "Enhanced approach to detection of SQL injection attack," in *Proc. 15th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, vol. 2016, pp. 466–469.
[19] P. A. Sonewar and N. A. Mhetre, "A novel approach for detection of SQL injection and cross site scripting attacks," in *Proc. Int. Conf. Pervasive Comput. (ICPC)*, Jan. 2015, pp. 1–4.
[20] C. Ping, "A second-order SQL injection detection method," in *Proc. IEEE 2nd Inf. Technol., Netw., Electron. Automat. Control Conf. (ITNEC)*, Dec. 2017, pp. 1792–1796.
[21] Y. Zhang and B. Wallace, "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification," 2015, *arXiv:1510.03820*. [Online]. Available: https://arxiv.org/abs/1510.03820
[22] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.
[23] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

XIN XIE received the bachelor's degree in electronic information engineering and finance from the University of Electronic Science and Technology of China, Chengdu, China, in 2017, where he is currently pursuing the degree in signal and information processing, machine learning, and network security with the School of Information and Communication Engineering.

CHUNHUI REN received the bachelor's, master's, and Ph.D. degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 1992, 1998, and 2006 respectively.

She has published more than ten academic articles, including several papers published in SCI (including SCIE) source journals, and several EI-retrieved journals, and conference papers. In recent years, her research has mainly focused on electronic countermeasures, statistical signal processing, non-cooperative signal processing, and other fields.

YUSHENG FU received the bachelor's degree in avionics engineering from Air Force Engineering University, Xi'an, China, in 1995, and the master's and Ph.D. degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 2000 and 2004, respectively.

He has been spent a total of ten million Yuan on scientific research, with an average annual expenditure of more than one million Yuan over the past five years. His research in the past five years has mainly focused on signal processing, aeroelectronics, and biomedical electronics engineering. His recent research on network science and technology has been carried out. He is a Reviewer of *Circuit System and Signal Processing* and other academic journals.

JIE XU received the bachelor's degree from Chongqing University, Chongqing, China, in 2003, and the master's degree in computer and communication engineering from the University of Besancon, in 2004.

His research results have published more than 30 articles in domestic and foreign academic journals and international academic conferences. In recent years, his researches mainly focus on the fields of network and information security, video recognition and security, data mining and analysis, and chaotic dynamics and its applications. He has applied for more than 20 national patents for technological inventions. He serves as a Reviewer for journals at home and abroad and a member of the Procedure Committee (TPC) of academic conferences.

JINHONG GUO received the bachelor's degree in electronic engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2010, and the Ph.D. degree in biomedical engineering from Nanyang Technological University, in 2014.

He is currently a Full Professor with the School of Communication and Information Engineering, University of Electronic Science and Technology of China. After the Ph.D. degree, he was a Postdoctoral Fellow of the pillar of engineering design with MIT-SUTD, Singapore, from 2014 to 2015. He then worked as a Visiting Professor with the School of Mechanical Engineering, University of Michigan, Ann Arbor, from January 2016 to July 2016. He has published over 100 research articles in journals such as the IEEE TII, TBME, TBioCAS, *Analytical Chemistry*, *Biosensor and Bioelectronics*, SNB, and Angew. His current research interests include sensor design and machine learning for network security. He was a recipient of the China Sichuan Thousand Talents Plan for Scholars Award, in 2015, and the Chengdu Expert in Science and Technology Award, in 2015.