LSTM-Based SQL Injection Detection Method for Intelligent Transportation System

Qi Li, Fang Wang, Junfeng Wang, *Member, IEEE*, and Weishi Li

***Abstract*—Intelligent transportation is an emerging technology that integrates advanced sensors, network communication, data processing, and automatic control technologies to provide great convenience for our daily lives. With the increasing popularity of intelligent transportation, its security issues have also attracted much attention. SQL injection attack is one of the most common attacks in the intelligent transportation system. It has characteristics of various types, fast mutations, hidden attacks, etc., and leads to great harm. Most of the current SQL detection methods are based on manually defined features. The detection results are heavily dependent on the accuracy of feature extraction, so it cannot cope with the increasingly complex SQL injection attacks in the intelligent transportation system. In order to solve this problem, this paper proposes a long short-term memory based SQL injection attack detection method, which can automatically learn the effective representation of data, and has a strong advantage to confront with complex high-dimensional massive data. In addition, this paper proposes an injection sample generation method based on data transmission channel from the perspective of penetration. This method can formally model SQL injection attack and generate valid positive samples. It can effectively solve the over-fitting problem caused by insufficient positive samples. The experimental results show that the proposed method improves the accuracy of the SQL injection attack detection and reduces the false positive rate, which is better than several related classical machine learning algorithms and commonly used deep learning algorithms.**

***Index Terms*—SQL injection detection, intelligent transportation system, LSTM, word2vec.**

I. Introduction

THE intelligent transportation system establishes wireless interconnection between people, vehicles, roads and urban networks. It processes, calculates, shares and securely publishes information which is collected by multiple sources on the information network platform, which provides great convenience for people's lives [1], [2]. In the intelligent transportation system, the management and control system is responsible for the schedule, management and service of the entire ground transportation

Manuscript received August 31, 2018; revised November 16, 2018; accepted January 3, 2019. Date of publication January 17, 2019; date of current version May 28, 2019. This work was supported in part by the National Key Research and Development Program under Grants 2016QY06X1205 and 2018YFB0804503, and in part by the National Natural Science Foundation of China under Grants 61762086 and U1836103. The review of this paper was coordinated by the Guest Editors of the Special Section on Machine Learning-Based Internet of Vehicles. *(Corresponding author: Junfeng Wang.)*

Q. Li, F. Wang, and W. Li are with the University of Posts and Telecommunications, Beijing 100876, China (e-mail: liqi2001@bupt.edu.cn; 18829291986@163.com; weishili@bupt.edu.cn).

J. Wang is with Sichuan University, Chengdu 610017, China (e-mail: wangjf@scu.edu.cn).

Digital Object Identifier 10.1109/TVT.2019.2893675

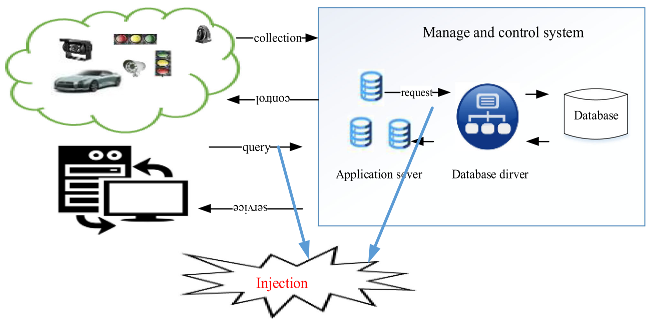

Fig. 1. SQL injection attacks in manage and control system.

[3]. By collecting, sharing and analyzing the information of road conditions, vehicles and people flow, we can obtain the real-time, dynamic and efficient traffic solutions. Currently, management and control system often provides services for users on websites [4]. It plays an important role in intelligent transportation system, and their security issues are facing increasingly severe challenges [5], [6]. SQL injection attacks are the most common type of attacks in intelligent transportation system. As is shown in Fig. 1, if the query validation is insufficient before sending to the database, the attacker can insert the aggressive SQL command into it, and execute the injection statement by spoofing server to for implement the SQL injection attack. Then, it can achieve access to sensitive data. In some cases, attackers can even control and destroy the system that hosts a Web application through SQL injection attack. When attackers gain full permission to its underlying database through SQL injection attack, they can control most of the system, even the entire system. Then these attack behaviors can cause serious problems such as privacy breaches, identity fraud and traffic safety. The SQL injection attack has great concealment. Attackers access from the normal web application port. It is similar to the normal request, and the firewall is difficult to detect. If the administrator is not accustomed to checking, the attack cannot be found even though the system has been in an attack for a long time. Then it will cause greater losses.

SQL injection attacks are among the top five in all web application security risks [7], that is why people are very concerned about SQL injection attacks. But as time goes by, SQL injection attack techniques are becoming more and more complex, especially in the field of intelligent transportation [8]. For example, when attackers want to achieve the vehicle information through a vehicle management system, they can insert an eternal condition into the position of the conditional statement. In this way, the query condition is always true and the attackers can achieve the purpose of injection. SQL injection detection method can protect the system from similar attacks and has been widely used in intelligent vehicle management and control system in recent years. Due to various types of SQL injection attacks and myriad variants, most proposed solutions can only detect a subset of possible SQL injection attacks. At present, methods for detecting SQL injection attacks mainly include instruction randomization [9], static analysis [10], dynamic analysis [11], and technologies which use shallow machine learning [12]–[14]. However, these methods cannot be used in the intelligent transportation system. Static analysis detection methods can only detect the type and syntax errors of SQL, but cannot detect attacks with the correct-input type. Dynamic detection can only detect vulnerabilities that are predefined by application developers. However, it cannot meet the detection requirements in the complex intelligent transportation system due to the diversity of SQL statement syntax. The existing shallow machine learning methods such as SVM and k-nearest neighbor train models based on features which are defined and selected artificially. The results mostly depend on selected features, but it is difficult to select representative features. So, it is easy to increase high false positive rate and false negative rate.

At present, SQL injection attack detection based on machine learning, especially deep neural network, has become a new research direction. But there are still many challenges and problems. For example, the machine learning method is easy to cause the over-fitting problem because of the insufficient dataset. In response to this problem, Zhang *et al.* have proposed an instance cloned extreme learning machine to augment the dataset [15]. However, the method of augmenting dataset based on features is not applicable in deep learning. Therefore, this paper first proposes a positive samples generation method. By formalizing the SQL injection samples, it guides the generation of positive samples to balance the distribution of input data and expand the training dataset which can alleviate the over-fitting problem. Secondly, the long short-term memory (LSTM) network is used to automatically extract features and train classifier to detect SQL injection attacks, which solves the problems better in the above methods. Finally, compared with the classical machine learning algorithms, it is proved that the proposed method can effectively detect SQL injection attacks. It not only improves the accuracy of detection, but also reduces the false positive rate and false negative rate.

The rest of the paper is organized as follows: Section II describes the background of SQL injection detection and the related work; Section III introduces the framework of our SQL injection detection; Section IV details the positive sample generation method and the architecture for detecting SQL injection attacks using LSTM; Section V presents the experiment and evaluates the results; Section VI concludes our work.

II. Background and Related Work

A. *SQL Injection Attack Detection Technology*

Traditional SQL injection detection methods mostly detect SQL injection attacks through static detection, dynamic detection and combination detection.

Static detection such as white box testing refers to the detection of errors and correctness through static source code analysis. Gould *et al.* [16] developed a tool called JDBC Checker for code analysis that can only detect a subset of SQLIA types but does not prevent them. Wassermann *et al.* [17] extended the white-box test to detect tautology attacks. However, these two methods can only detect some special kinds of attacks.

Dynamic detection refers to detecting errors and correctness by performing dynamic penetration tests or generating models at runtime. Yi *et al*. [18] designed an analysis model embedded in the Web application. They parsed SQL statements into SQL syntax trees through stain analysis, and then determined whether there was an SQL injection attack. Appiah *et al.* proposed an improved pattern matching method for signature-based SQL injection attack detection framework [19], which distinguishes between true SQL queries and malicious queries by integrating fingerprint recognition methods and pattern matching. Pandurang *et al.* proposed a mapping model for detecting and preventing SQL injection [20].

Static and dynamic combination detection is a hybrid use of pattern matching between valid requests and dynamic web requests to detect and block SQL injection attacks. An example is AMNESIA [21] proposed by William *et al.* It firstly analyzed the web application statically to generate a normal SQL query model. Then it monitored dynamical queries during the dynamic phase. Queries that do not conform to the model will be identified as SQLIA and blocked. Xiao *et al.* proposed a method for detecting and defending SQL injection based on URL-SQL mapping [22] by analyzing user behavior and SQL execution response. They extracted the predefined URL and the corresponding SQL query to establish a mapping model between the request and the SQL query. But there are many uncertain factors when the system executes SQL statements. If the extraction of invariants (the normal state of the web application) is not comprehensive, it will lead to false alarms and false negatives.

These methods have some common characteristics. Firstly, these researchers usually extracted some characteristics based on experience, and then detected the injection attacks by using string matching methods. However, this method cannot deal with the increasingly complex and changeable injection attacks in intelligent transportation system. Secondly, when detecting injection by machine learning method, researchers usually adopt crawling training samples from websites or generating training sets with some tools. However, in the intelligent transportation system, the number of positive and negative samples is extremely uneven, and this method can easily lead to the over-fitting problem.

B. *Application of Machine Learning Method in Network Security Detection*

In recent years, machine learning has developed rapidly and has achieved good results in the classification problem [23]–[25]. More and more researchers are trying to solve problems encountered in network security by using machine learning algorithms.

Kim *et al.* trained SVM classifiers by extracting valid n-gram features from malicious code [26], but various modes

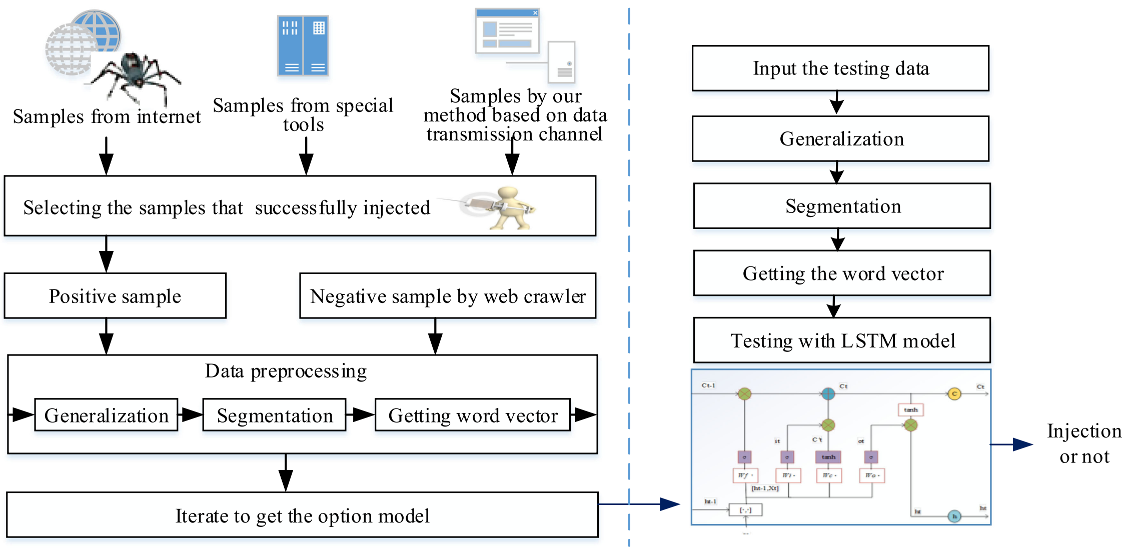

Fig. 2. The overall architecture of our proposed system.

were needed to improve the accuracy of the method. Wang and Li learned SQL statements through program tracking technology [27] to detect malicious behavior between database and application. The program tracking hashing technique, the tree structure of the SQL query and the query name similarity are used as features to distinguish between malicious queries and benign queries, in order to train support vector machine (SVM) classifier to detect malicious queries at runtime. Komiya *et al.* used the white space separation and token method to extract features [28], and used the three machine learning algorithms (SVM, naive Bayes and k-nearest neighbor algorithm) to classify malicious Web code to detect attacks. Buchanan *et al.* used TCSVM [29] as a classifier to detect SQL injection attacks, besides they combined hidden Markov model and similarity distance algorithms to detect SQL injection attacks.

In addition, all the shallow machine learning methods mentioned above use artificial definition features, therefore the detection results are also based on the representativeness of feature extraction. The variety and complexity of SQL injection attacks make this approach incapable of achieving high accuracy for detecting all types of attacks.

Recently, deep learning begins to be used in the field of network attack detection to achieve better detection accuracy. Deep learning methods such as LSTM networks does not require features by manual definition, it allows the model to learn abstract features for classification. Kim *et al.* proposed an attack detection method using long short-term memory (LSTM) network, and trained the LSTM network using the KDD Cup 1999 data set [30]. Roy *et al.* studied the ability of deep neural networks to classify different types of network attacks [31]. To predict user behavior in the Tor network, Ishitaki *et al.* applied deep recurrent neural networks [32]. Kang *et al.* established a DNN structure trained through probability-based feature vectors to improve the security of the in-vehicle network [33]. Wang *et al.* proposed an intrusion detection system based on hierarchical spatiotemporal features (HAST-IDS). They used deep convolutional neural networks (CNN) to learn low-level spatial features of network traffic. Then, they used Long Short Term Memory networks to learn advanced temporal features. The whole process of feature learning was automatically completed by the deep neural network. It did not require feature engineering technology, so that the false positive rate can reduce effectively [34].

The methods mentioned above used deep neural network to detect network attacks, but they cannot effectively and specially solve complex SQL injection attack detection. In this paper, we use the long short-term memory network (LSTM) to complete feature extraction automatically and train classifier specifically to detect SQL injection attacks.

III. The Framework of Our Proposed System

As shown in Fig. 2, the overall architecture of our proposed system consists of two important phases: off-line training and on-line testing.

In the off-line training phase, we collect a large number of samples in various ways and label them as positive and negative samples. Then we preprocess these samples, including generalization, word segmentation and word vector generation. Finally, the preprocessed data is used to train the model. In the on-line testing phase, we preprocess test data exactly as we did in the off-line training phase. Then, the preprocessed data is input into the trained classifier to determine whether it is an SQL injection or not.

The collection and selection of training data is a key problem in artificial intelligence-based anomaly detection. The collection and selection of positive samples is very difficult during the detection process of SQL injection. On the one hand, there are fewer public data sets for SQL injection, and the amount of data that can be collected by various security websites (such as GitHub, exploit-db, etc.) is insufficient. It is easy to cause overfitting in the model training process, especially in the training process based on deep learning. On the other hand, many SQL injection statements are actually intercepted by the protective measures taken by the website, so the intercepted statements cannot be input into the model as positive samples in principle. However, due to the different protection mechanisms adopted

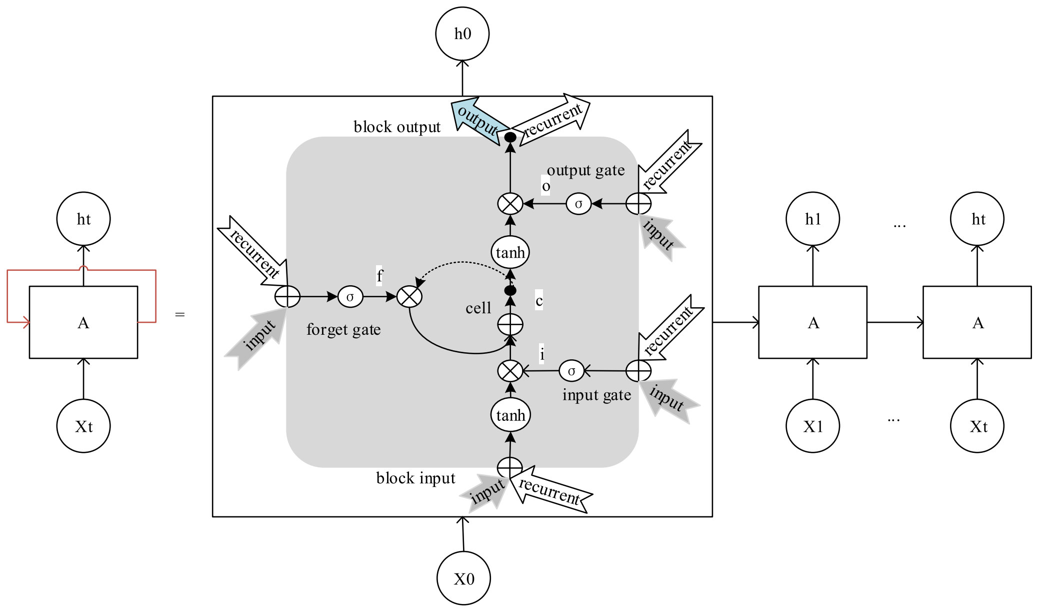

Fig. 3. An unrolled LSTM layer.

by different websites, the selection of positive samples needs to be targeted. Therefore, in this paper, in order to guarantee the number of positive samples, we propose a method of generating SQL injection samples based on communication attack behaviors analysis, which can perform a comprehensive and regular expression of SQL injection, then formally model the attack behaviors of SQL injection. Using formal modeling to describe SQL injection attacks is beneficial to the generalization of the inherent law of the attack behaviors. By abstracting the specific attack request into a formal description of the logical form, it can better reflect the inherent rules of its behavior. And it is effective to resolve the over-fitting problem due to too few positive samples.

In the traditional SQL injection detection process based on shallow machine learning, the researchers extracted various features, such as "percentage of uppercase characters", "the typical SQL injection keywords", but these features are easy to cause false negatives, etc. It is difficult to effectively detect the mutated SQL injection attacks. To solve this problem, this paper proposes an LSTM-based SQL injection detection method, which allows the model to automatically learn abstract features to classify and detect diverse SQL injection attacks.

IV. The Details of SQL Injection Detection Method Based on LSTM

In this section, we firstly describe the process to detect diversity of SQL injection attacks with LSTM in detail. Then we propose a positive samples generation method to enrich the training data set, so as to alleviate the over-fitting problem that often occurs in experiments.

A. The Process of SQL Injection Detection Based on LSTM

LSTM (Long Short Term Memory) is widely used in deep learning. Compared to RNN (Recurrent Neural Networks), LSTM network has been shown to learn long-term dependencies more easily than the simple recurrent architectures and it also can reduce the problem of gradient disappearance and gradient explosion. Therefore, after embedding the word vector, LSTM is selected to learn the long-term dependence and timing in the higher-level feature sequence in our experiments. LSTM consists of state boxes receiving the inputs through time. In each time step, an input vector is fed into LSTM, the output is computed refer to (1).

$$h_t = f_w\left(h_{t-1}, x_t\right) \tag{1}$$

Where x_t is the input vector, h_t and h_{t-1} are the state vectors at time t and $(t-1)$, f_w is a nonlinear activation function where w are the weights matrix.

An unrolled LSTM layer and the LSTM block diagram is often depicted as in Fig. 3. The LSTM uses three gates to control the contents of the unit state. The forget gate determines how much the state of the cell at the previous moment is retained to the current state. The input gate determines how much of the input to the network at the current time is saved to the unit state. The output gate controls how much the state of the cell is output to the current output.

The forget gate, input gate and output gate at time t are given as follows:

$$f_t = \sigma\left(W_{hf}h_{t-1} + W_{xf}x_t + b_f\right) \tag{2}$$

$$i_t = \sigma\left(W_{hi}h_{t-1} + W_{xi}x_t + b_i\right) \tag{3}$$

$$o_t = \sigma\left(W_{ho}h_{t-1} + W_{xo}x_t + b_o\right) \tag{4}$$

Where x_t is the current input vector and W_{hi}, W_{xi}, W_{hf}, W_{xf}, W_{ho}, W_{xo} are respectively input weights, recurrent weights for the forget gates and output weights. b_i, b_f and b_o are biases. σ is sigmoid function.

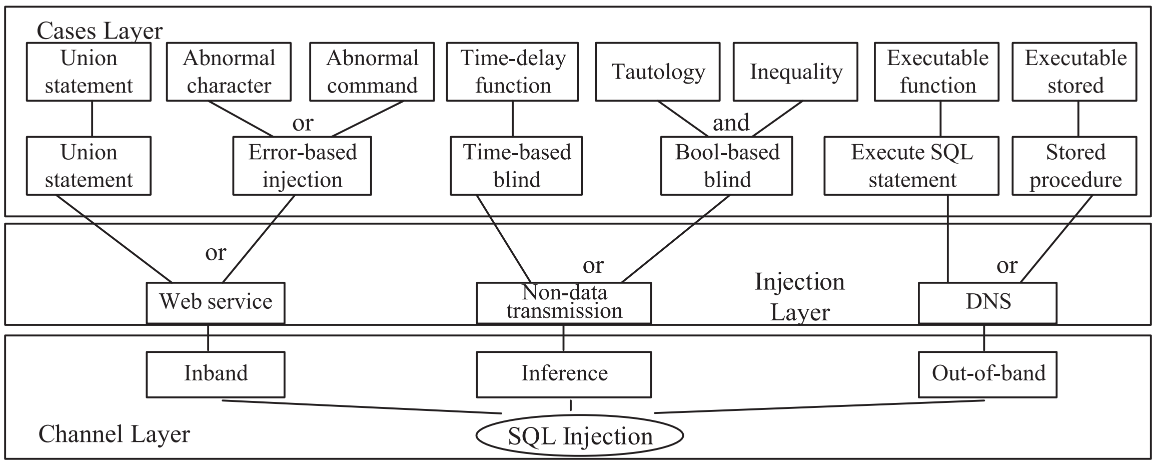

Fig. 4. An SQL injection attack model based on data transmission channel.

The current cell state and the final output of the LSTM cell are as follows:

$$c_t = f_t c_{t-1} + i_t \tanh\left(W_{hc} \cdot h_{t-1} + W_{xc} x_t + b_c\right) \quad (5)$$

$$h_t = o_t \tanh\left(c_t\right) \quad (6)$$

Where c_{t-1} are the previous state of cell, W_{hc} and W_{xc} are weight metrics. b_c is bias. h_t is the current hidden layer vector, containing the outputs of all the LSTM cells.

The procedure of SQL injection detection method based on LSTM is as follows: Data preprocessing includes label, normalization, word segmentation and embedded word vector. According to the classification of the data, we label the SQL injection samples as 1 and the negative sample as 0. Then all English letters in samples are changed to lowercase and detect the encoding as well as decode the different encodings, which are mainly url encoding and base64 encoding. Build a word segmentation model by knowledge, which are mainly using symbols, including "(",")", "", "%". Word Embedding is performed on the word segmentation result and each word is represented as a numeric word vector. In this process, the sentences of different lengths the LSTM layer, pass through a time series to obtain the vectors of *t* hidden LSTM neural units. After these vectors pass through the mean pooling layer, a vector *h* can be obtained. Finally, there is a SoftMax layer. We can get a class distribution vector and select the category output with the highest probability as the final prediction category. In process, we use the training data to train the model, adjust the hyper-parameters and continue to train it. The final epoch set is 30,000 times. After each training 3000 times, the model is output. The accuracy and F1-score are selected as the evaluation criteria. After the training, the best model which is selected from output models acts as a classifier to detect SQL injection attacks. In the testing stage, user input is normalized, word segmentation, embedded word vector and then input into the classifier for judgment. If the classifier's output is 0, it indicates that the user input is normal, otherwise it indicates that the user input is SQL injection attack statement.

B. The Method of Generating Positive Samples

The collection of training samples has always been one of the most important issues in the field of machine learning. The number of samples needs to be sufficient, the positive and negative samples should be as balanced as possible. At present, the proportion of positive and negative samples collected from the network is generally around 1:5, the accuracy of the trained model is lower. Currently, in order to enrich the sample of SQL injection detection, the researchers designed more efficient methods to crawl injection samples, and generated attack samples with tools such as SQLmap. However, in the intelligent transportation system, the attackers always mutate and camouflage the attack code constantly to bypass the defense measures of the management and control system. Although the existing method increases the number of attack samples, the obtained samples are highly random and lack regularity. On one hand, it is easy to generate a large number of redundant samples. On the other hand, the type of attacks is not widespread enough to achieve comprehensive protection. Therefore, a more complete samples generation method needs to be proposed. It should be able to describe the attack behaviors and law better, and can be filtrated combined with system defense measures. In this way, more effective training samples can be obtained and the quality of the samples can be improved. Therefore, the generalization ability of the trained classifier can also be enhanced to improve the accuracy of detection.

To this end, this paper proposes a SQL injection attack model based on data transmission channel from the perspective of penetration. It combines the characteristics of intelligent traffic management and control system. As shown in Fig. 4, an SQL injection can be expressed as the root node. From the bottom, according to the transmission channel, SQL injection attacks are divided into three categories: in-band, inference, out-of-band channels.

According to the SQL injection attack model established in Fig. 4, we propose a formal description of the SQL injection samples. The definition of the attack payload is given as F(x): F(x) denotes a set of attack input. Table I gives the classification

TABLE I
CLASSIFICATION BASIS OF SQL INJECTION ATTACK PAYLOAD AND EXAMPLE

Payload	Classification basis	Example
F(DS)	Select common exception characters or strings	'/'''/''/;and 1=1#/--/ @@version;--
F(CON)	Comparison operators and conditional categories	=/</>/!=/<=/>=/in/like/between
F(IE)	Comparison operators and tautology categories	2=2/'b'<'c'/2>1/ name like '%admin%'
F(NE)	Comparison operators and inequality categories	2=3/'a'>'b'/2>3/3<2/'c' not in ('b', 'c')
F(SC)	SQL command verb keyword	; select * from ... / union all select from/;insert into ... Values...
F(OC)	SQL function name	Select load_file('\\\\attacker.com\\foo');
F(OP)	Stored procedure	select dbms_ldap.init ('||.attacker.com',80) from dual;
F(TI)	Time function	Sleep(1)/benchmark(1,md5(1))/waitfor delay '0:0:5'
F(LG)	Select common logical conjunctions	And/ &&/|| /or
F(CN)	Conditional function name	If(1,b,c)/if null(b, c)
F(DC)	Select common exception commands	Select @@version/having 1=1--/union select user()
F(WAF)	According to the coding rules	SeLEct/%73%45%6c%65%43%74/Union /*!40001 select*/1

TABLE II
SQL INJECTION SAMPLE EXPRESSION

Injection method	SQL Injection sample expression
Error injection	F(DS) || F(DC) || F(WAF) ∗ (F(DS) || F(DC))
Joint query	F(SC) || F(WAF) ∗ F(SC)
Time-based blind	(F(SC) ∗ F(TI) || F(TI)) ∗ (F(CON) || F(CN))
Boolean blind	F(CON) ∗ (F(LG) ∗ F(IE) && F(LG) ∗ F(NE))
Execute SQL/Stored	(F(SC) || F(LG) ∗ F(IE)) ∗ (F(OC) || F(OP))

basis of injection attack payload and examples. When there are multiple examples, they are split with the symbol "/".

The attack payloads defined above classifies the attack input of SQL injection. F(DS), F(CON), F(IE), F(NE), F(TI) are basic attack function sets, F(SC), F(OC), F(LG) are basic sets of SQL operations. In order to combine the attack payloads into attack inputs with certain rules, the operators between attack payloads need to be defined. Therefore, this paper defines the following operator: "||" is defined as the attack load or operation, which indicates that F(a) and F(b) can be used as one of the two attack loads; "&&" is the attack load and operation, which shows that both F(a) and F(b) attack payloads need to be used simultaneously; "∗" is the compound operation between attack payloads. The operation sequence is from right to left, such as. The parentheses in the operator have the highest priority and the other priorities from high to low are ∗, &&, ||.

According to the defined attack payloads, operators and arithmetic rules, the injection samples expression used in each SQL injection attack mode are shown in Table II. It describes how basic attack payloads composite SQL injection samples.

V. EXPERIMENTS AND RESULT DISCUSSION

A. Dataset

In the experiments, we used $DS_i (i = 1, 2, 3, 4, 5, 6)$ to represent the i-th dataset. The details of each dataset are shown in Table III. Part of the collected SQL injection positive samples in each dataset were obtained from the vulnerability submission platform exploit-db and WooYun vulnerability submission platform. The other part was acquired by running SQLmap and its tamper scripts on certain websites within one month. The total number of SQL injection samples we collected was 6052. We generated injection samples with different quantities by the positive sample generation method described in Section IV and superimposed the collected positive samples as positive samples in dataset DS_i. The negative samples in the datasets were mainly from enterprises and institutions, school websites and various social platforms.

TABLE III
THE NUMBER OF SAMPLES IN DATASET

Datasets	negative samples	positive samples		Ratio of positive and negative samples
		Samples generated	total	
DS_1	30370	0	6052	1:5
DS_2	30370	6000	12052	2:5
DS_3	30370	12000	18052	3:5
DS_4	30370	18000	24052	4:5
DS_5	30370	24000	30052	5:5
DS_6	30370	30000	36052	6:5

TABLE IV
THE RESULT OF USING DIFFERENT FEATURE VECTOR

the feature vector transformation method	Acc	P	R	F1
Word2vec	93.47%	93.56%	92.43%	92.99%
BoW	91.93%	90.67%	92.91%	91.78%

B. Experimental Setup and Evaluation Criteria

All experiments carried out in the Ubuntu 14.04 LTS environment, using python 3.5.4 and Keras2.1.2 neural network library to build networks. We use Tensorflow1.4.1 as a backend computing framework. CPU server is Inter (R) Xeon (R) CPU E5-2637 v4 @ 3.50GHz, GPU is TITAN (X) (Pascal).

In the experiments, the selected evaluation criteria of SQL injection attack detection method are the detection accuracy (Acc), the precision (P), the recall rate(R). Besides, we use the F1-score as a comprehensive assessment of detection performance. The formulas of the metric are as follows:

$$\text{Acc} = \left(1 - \frac{errors_{num}}{sum}\right) \times 100\% \tag{7}$$

$$\text{P} = \frac{TP}{TP + FP} \times 100\% \tag{8}$$

$$\text{R} = \frac{TP}{TP + FN} \times 100\% \tag{9}$$

$$\text{F1} = \frac{2 \times P \times R}{P + R} \times 100\% \tag{10}$$

Acc represents the accuracy, which is the ratio of the samples that are classified correctly to all samples. $errors_{num}$ indicates the number of samples misclassified and *sum* indicates the total number of samples. P refers to the proportion that is classified

TABLE V
DETECTION ACCURACY FOR A SMALL NUMBER OF TRAINING SAMPLES

%	LSTM	SVM	KNN	NB	DT	RF	RNN	CNN	MLP
Acc	91.53	90.79	89.28	91.51	93.57	93.21	90.24	88.83	87.84
P	91.32	90.77	90.30	90.62	92.02	92.88	90.71	90.63	88.32
R	90.89	91.38	88.26	91.83	91.31	94.43	89.40	84.01	85.78
F1	91.10	91.07	89.27	91.22	91.66	93.64	90.34	87.19	87.03

correctly in the SQL injection statement that has been predicted to be true. TP is true positive, it represents the number of SQL injection samples. FP and FN are the false positive and false negative respectively. R is the proportion of individuals who are predicted correctly in all true SQL injection statements. F1-score is defined based on the harmonic mean of the precision and recall ratio. The smaller value is considered more important than arithmetic average and geometric average.

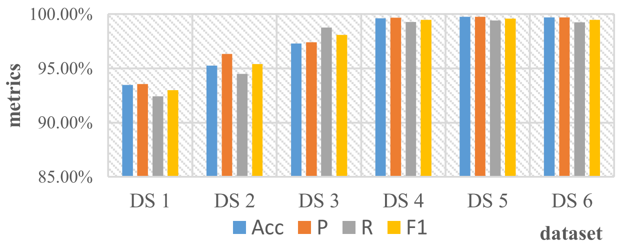

Fig. 5. Results on different datasets.

C. Experimental Results

The experimental results were based on 10-fold cross-validation that divides the dataset into 10 non-overlapping subsets and takes 9 of them as training data in turn, the remaining 1 subset as the test data. The test error was estimated by taking the average test error across 10 trials.

1) The Influence of Different Feature Vectors: In experiment, the feature vectors transformation used the word2vec and Bow methods. We trained the model using LSTM on the dataset DS_1. The results are shown in Table IV.

- Feature vector based on word2vec

Word2Vec uses the word vector to represent the semantic information of the word by learning the text. It maps the word from the original space to the new multi-dimensional space and makes the semantically similar words close in the space through an embedded space. The dimension of our word vector was 128.

- Feature vectors based on BoW

The Bag-of-words model is a method of counting each word and its number of occurrences in each document. In the experiments, we selected the 200 most common words in SQL injection based on experience and built the Bag-of-words model.

It can be seen from Table IV that the accuracy is 93.47% in our method that use feature vector based on word2vec and the F1-score is 92.99%, which is better than the detection results using BoW for word embedding. Therefore, our model uses the feature vector based on Word2vec.

2) The Effect of Sample Expansion Method on Detection Results: In Table V, we compared the effectiveness of various machine learning methods for a small number of training samples. It can be seen that when the number of training data is relatively small, none of the selected algorithm can obtain satisfactory detection results. Therefore, we propose an injection samples generation method based on communication attack behaviors analysis

In order to reduce the over-fitting problem, this paper designs a method of generating positive samples to generate SQL injection statements to expand the training set in the datasets. In the experiments, the training process was performed on different datasets $DS_i (i = 1, 2, 3, 4, 5, 6)$ respectively. The detection results are shown in Fig. 5, in which we used the feature vector based on word2vec.

As can be seen in Fig. 5, the accuracy, precision and F1-score of the classifier trained by our method are all above 92% on the dataset DS_1. After the expansion by the injection samples generation method described in IV, the same test samples are used to test the classifier. The detection accuracy, recall and precision are improved as the number of positive samples increases. The F1-score reaches 99.76% and the false positive rate is less than 0.5% on the dataset DS_5. The experimental results show that using the method proposed in Section IV to expand the sample effectively solves the over-fitting problem caused by the imbalance of positive and negative samples, and reduces the randomness of the sample distribution, so that the trained classifier has stronger generalization ability to improve the accuracy of detection.

3) Comparison With Other Detection Methods

Comparison with classic machine learning methods: Currently several classic machine learning methods are SVM, KNN, Decision Tree, NB and RF. In this section, we compared our method with these state-of-arts methods. Feature extraction refers to reference [35]. Five classic machine learning methods were tested on the datasets $DS_i (i = 1, 2, 3, 4, 5, 6)$ respectively. For the SVM model, three kernel functions were used to conduct experiments, and the best one was selected for comparison. Fig. 6 show the experimental results.

It can be seen from Fig. 6 that the detection method using random forest has the best effect in the shallow machine learning method and its F1-Score reaches 95.64% on the dataset DS_1. The nearest neighbor algorithm has the worst detection effect, only 90.86%. With the increase of training samples, the F1-Score of detecting SQL injection using various shallow machine learning algorithms is also increasing gradually. The increase of detection effect is largest when the ratio of positive and

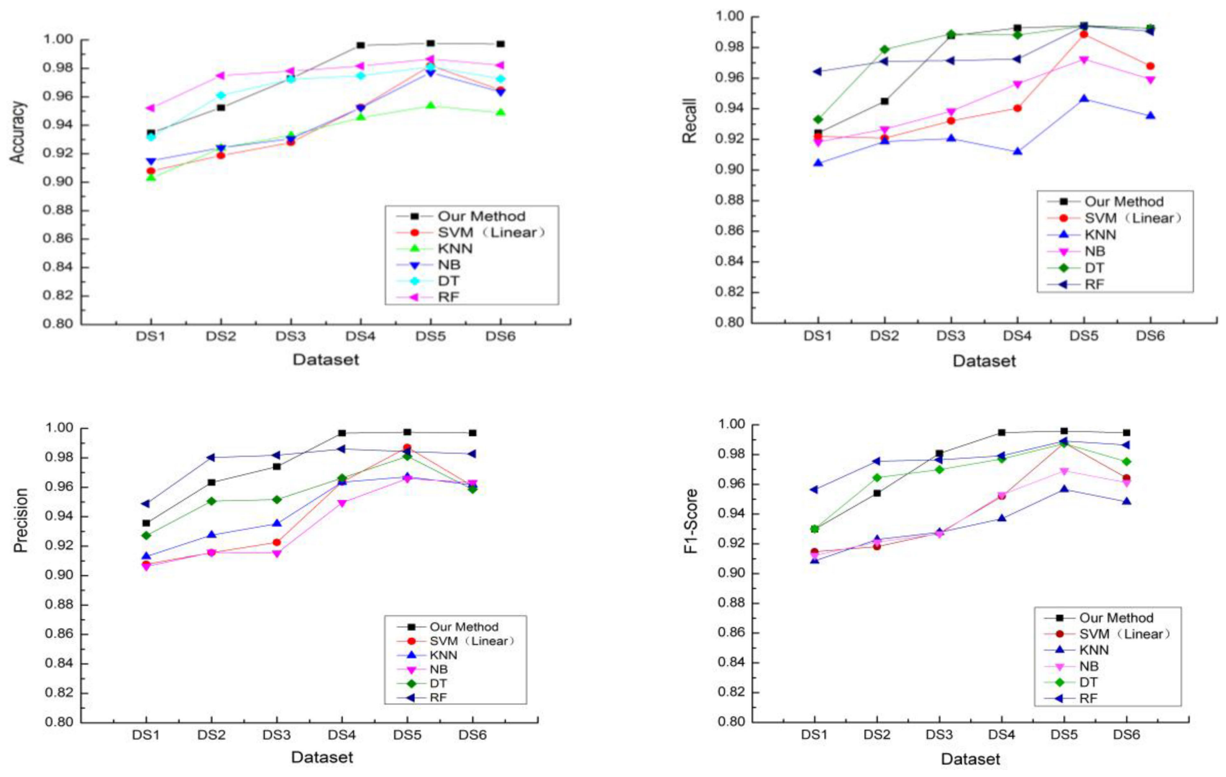

Fig. 6. Results comparing with classic machine learning methods.

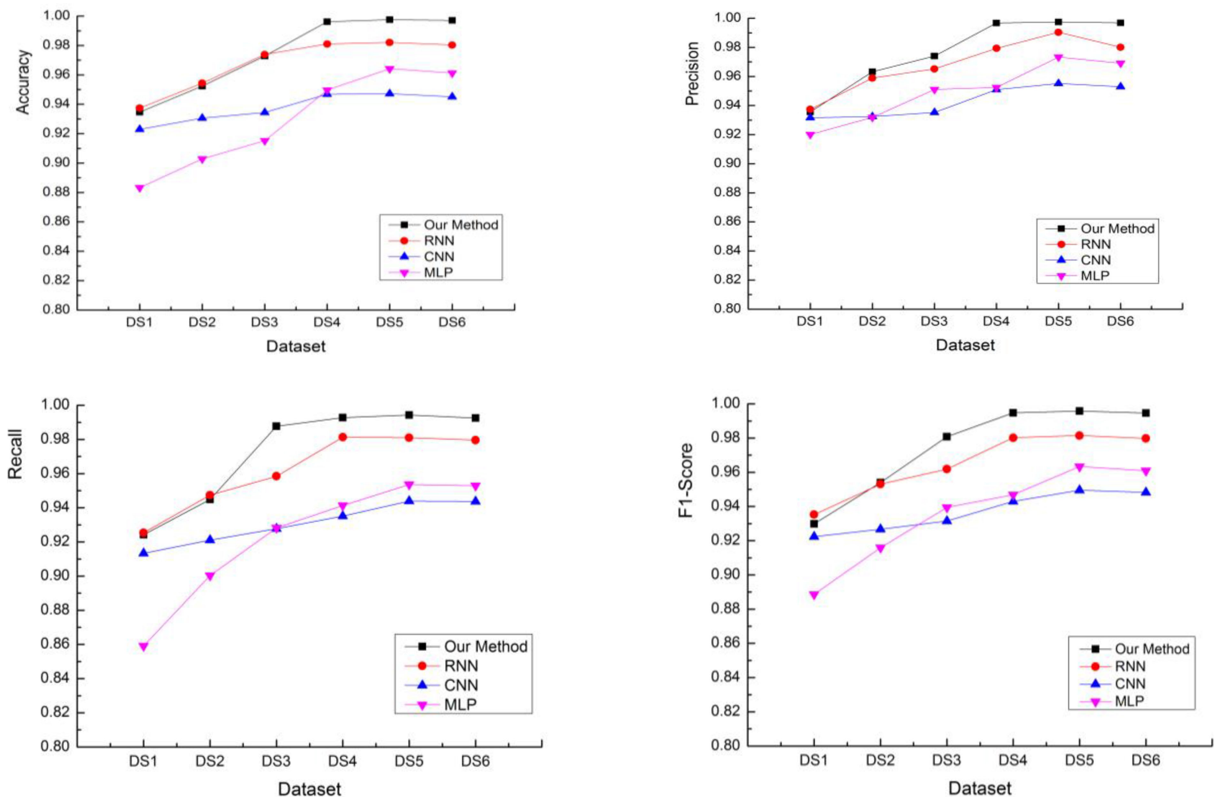

Fig. 7. Results comparing with common deep neural networks.

negative samples is between 4:5 and 5:5. When the ratio of positive and negative samples is 1:1, the detection effect reaches a peak value, and then the detection effect gradually decreases with the increase of the positive samples. It can be found that after the expansion of the training dataset, the detection accuracy, precision, recall rate and F1-score of the shallow machine learning algorithm are greatly improved, indicating that the proposed dataset expansion method can also mitigate the over-fitting problem that occurs in the shallow machine learning method. The accuracy of our classifier using LSTM network to automatically learn the characteristics of various SQL injection statements is 93.47% on the dataset DS_1. As the training samples expand, the F1-score reaches 99.58% on the dataset DS_5, which is significantly higher than using shallow machine learning. This is mainly because the deep learning method does not need to use prior knowledge to extract features, and avoids the problem that the loss of accuracy is caused by the lack of key features.

Comparison with common deep neural networks: It is compared with several commonly used deep neural networks CNN, RNN and MLP in the field of deep learning. For fair comparison, the above networks also used the feature vectors based on Word2vec to train the classifier. Three hidden layers were set up in the same way as the method proposed in this paper. CNN used a one-dimensional convolution kernel for detection, using Input-> Conv->Conv->Maxpool-> Conv-> Maxpool-> Dense structure, RNN and LSTM used the structure of Input->RNN1(LSTM1)-> RNN2(LSTM2)-> RNN3(LSTM3)-> Dense.

As is shown in Fig. 7, the detection effect of MLP is the worst, and our method is the best on the dataset DS_1. As the number of positive samples in the training samples increases, the detection effect of the classifiers trained is gradually increasing. The detection effect reaches a peak on dataset DS_5, and the false negative rate is less than 1%. On the dataset DS_5, our method is the best, RNN is second, and CNN is the worst. CNN mainly obtains the local relevance of data, while RNN and LSTM mainly obtain the sequence of data. The results show that the sequence of SQL injection data can reflect the characteristics of SQL injection data more than local correlation. In addition, the F1-score of our method is higher than that of RNN. That is, the comprehensive detection effect of our proposed method is better than that of RNN. This is mainly because LSTM can capture long-range dependence, so this method is more suitable for SQL injection attack detection than other deep neural networks. The experimental results also show that our proposed data expansion method is very effective in deep learning.

VI. Conclusion

This paper analyzes the existing SQL injection detection methods in intelligent transportation system and finds that the traditional detection methods have the disadvantages of detecting unknown SQL injection attacks and high false positive rate. The method based on shallow machine learning has difficulty in manually extracting features, and it is easy to cause over-fitting problem. Therefore, we designed a SQL injection samples generation method to augment the dataset, and utilized LSTM to capture the advantages of data sequence and long-range dependence to detect SQL injection attacks. At the same time, the specific model training process was given. Finally, we programed and implemented it. A large number of experiments were performed on the dataset. We compared the efficiency of detecting SQL injection statements using classical machine learning methods (SVM, KNN, Decision Tree, NB, RF) and commonly used deep neural networks (CNN, RNN, MLP). The experimental results show that using our positive sample generation method to augment the dataset can alleviate the overfitting problem, which is helpful for subsequent SQL injection detection. Our method for SQL injection attacks detection in intelligent transportation system obtains excellent results, it improves accuracy, reduces false negative rate and false positive rate.

References

[1] L. F. Herrera-Quintero, J. Vega-Alfonso, K. Banse, and E. C. Zambrano, "Smart ITS sensor for the transportation planning based on IoT approaches using serverless and microservices architecture," *IEEE Intell. Transp. Syst. Mag.*, vol. 10, no. 2, pp. 17–27, Summer 2018.

[2] L. Li, J. Liu, L. Cheng, and S. Qiu, "CreditCoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 7, pp. 2204–2220, Jul. 2018.

[3] M. Chaturvedi and S. Srivastava, "Multi-modal design of an intelligent transportation system," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 8, pp. 2017–2027, Aug. 2017.

[4] G. Xiong, D. Shen, X. Dong, B. Hu, and D. Fan, "Parallel transportation management and control system for subways," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 7, pp. 1974–1979, Jul. 2017.

[5] D. Fernandez-Llorca, R. Q. Minguez, I. P. Alonso, and C. F. Lopez, "Assistive intelligent transportation systems: The need for user localization and anonymous disability identification," *IEEE Intell. Transp. Syst. Mag.*, vol. 9, no. 2, pp. 25–40, Summer 2017.

[6] P. Sui, X. Li, and Y. Bai, "A study of enhancing privacy for intelligent transportation systems: K-correlation privacy model against moving preference attacks for location trajectory data," *IEEE Access*, vol. 55, pp. 24555–24567, 2017.

[7] B. Appiah, "OWASP Top 10, 2017," 2017. [Online]. Available: https://www.owasp.org/index.php/Top_10-2017_Top_10.

[8] 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. Int. Conf. Inventive Syst. Control*, 2018, pp. 736–741.

[9] L. K. Shar and H. B. K. Tan, "Defeating SQL injection," *Computer*, vol. 46, no. 3, pp. 69–77, 2013.

[10] X.-X. Yan, Q.-X. Wang, and H.-T. Ma, "Path sensitive static analysis of taint-style vulnerabilities in PHP code," in *Proc. Int. Conf. Commun. Technol.*, 2017, pp. 1382–1386.

[11] P. A. Sonewar and S. D. Thosar, "Detection of SQL injection and XSS attacks in three tier web applications," in *Proc. Int. Conf. Comput. Commun. Control Automat.*, 2016, pp. 1–4.

[12] R. Komiya, I. Paik, and M. Hisada, "Classification of malicious web code by machine learning," in *Proc. Int. Conf. Awareness Sci. Technol.*, 2012, pp. 406–411.

[13] 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. Serv. Manage.*, 2017, pp. 1087–1090.

[14] K. Kamtuo and C. Soomlek, "Machine learning for SQL injection prevention on server-side scripting," in *Proc. Int. Comput. Sci. Eng. Conf.*, 2016, pp. 1–6.

[15] Y. Zhang, J. Wu, and C. Zhou, "Instance cloned extreme learning machine," *Pattern Recognit.*, vol. 68, pp. 52–65, 2017.

[16] C. Gould, Z. Su, and P. Devanbu, "JDBC checker: A static analysis tool for SQL/JDBC applications," in *Proc. Int. Conf. Softw. Eng.*, 2004, pp. 697–698.

[17] G. Wassermann, C. Gould, Z. Su, and P. Devanbu, "Static checking of dynamically generated queries in database applications," *ACM Trans. Softw. Eng. Methodol.*, vol. 16, no. 4, 2007, Art. no. 14.

[18] W. Yi, L. Zhoujun, and G. Tao, “Literal tainting method for preventing code injection attack in web application,” *J. Comput. Res. Develop.*, vol. 49, no. 11, pp. 2414–2423, 2012.

[19] B. Appiah, E. Opoku-Mensah, and Z. Qin, “SQL injection attack detection using fingerprints and pattern matching technique,” in *Proc. 8th IEEE Int. Conf. Softw. Eng. Serv. Sci.*, 2017, pp. 583–587.

[20] R. M. Pandurang and D. C. Karia, “A mapping-based podel for preventing cross site scripting and sql injection attacks on web application and its impact analysis,” in *Proc. Int. Conf. Next Gener. Comput. Technol.*, 2016, pp. 414–418.

[21] W. G. J. Halfond and A. Orso, “AMNESIA: Analysis and monitoring for neutralizing SQL-injection attacks,” in *Proc. 20th IEEE/ACM Int. Conf. Autom. Softw. Eng.*, 2005, pp. 174–183.

[22] Z. Xiao, Z. Zhou, and W. Yang, “An approach for SQL injection detection based on behavior and response analysis,” in *Proc. IEEE Int. Conf. Commun. Softw. Netw.*, 2017, pp. 1437–1442.

[23] Z. Ma, J. Xue, A. Leijon, Z. Y. Z. Tan, and J. Guo, “Decorrelation of neutral vector variables: Theory and applications,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 129–143, Jan. 2018.

[24] Z. Ma, Y. Lai, W. B. Kleijn, L. Wang, and J. Guo, “Variational Bayesian learning for Dirichlet process mixture of inverted Dirichlet distributions in non-Gaussian image feature modeling,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 2, pp. 449–463, 2019.

[25] Z. Ma, A. E. Teschendorff, A. Leijon, Y. Qiao, H. Zhang, and J. Guo, “Variational Bayesian matrix factorization for bounded support data,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 4, pp. 876–889, Apr. 2015.

[26] J. Choi, H. Kim, C. Choi, and P. Kim, “Efficient malicious code detection using N-gram analysis and SVM,” in *Proc. 14th Int. Conf. Netw.-Based Inf. Syst.*, 2011, pp. 618–621.

[27] Y. Wang and Z. Li, “SQL injection detection via program tracing and machine learning,” in *Internet and Distributed Computing Systems* (Lecture Notes in Computer Science), vol. 7646. Berlin, Germany: Springer, 2012, pp. 264–274.

[28] R. Komiya, I. Paik, and M. Hisada, “Classification of malicious web code by machine learning,” in *Proc. Int. Conf. Awareness Sci. Technol.*, 2012, pp. 406–411.

[29] 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. Serv. Manage.*, 2017, pp. 1087–1090.

[30] J. Kim, J. Kim, and H. Thu, “Long short-term memory recurrent neural network classifier for intrusion detection,” in *Proc. Int. Conf. Platform Technol. Serv.*, 2016, pp. 1–5.

[31] S. S. Roy, A. Mallik, and R. Gulati, “A deep learning based artificial neural network approach for intrusion detection,” in *Proc. Int. Conf. Math. Comput.*, 2017, pp. 44–53.

[32] T. Ishitaki, R. Obukata, and T. Oda, “Application of deep recurrent neural networks for prediction of user behavior in tor networks,” in *Proc. 31st Int. Conf. Adv. Inf. Netw. Appl. Workshops*, 2017, pp. 238–243.

[33] M. J. Kang and J. W. Kang, “Intrusion detection system using deep neural network for in-vehicle network security,” *PLoS ONE*, vol. 11, no. 6, 2016, Art. no. e0155781.

[34] W. Wang, Y. Sheng, and J. Wang, “HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection,” *IEEE Access*, vol. 6, no. 99, pp. 1792–1806, 2018.

[35] S. Smusz, S. Mordalski, and J. Witek, “Multi-source data analysis and evaluation of machine learning techniques for SQL injection detection,” *J. Chem. Inf. Model.*, vol. 55, no. 4, pp. 823–832, 2015.

Qi Li received the Ph.D. degree in computer science and technology from the Beijing University of Posts and Telecommunications, Beijing, China, in 2010. She is currently an Associate Professor with the Information Security Center, State Key Laboratory of Networking and Switching Technology, School of Computer Science, Beijing University of Posts and Telecommunications. Her current research focuses on information systems and software.

Fang Wang received the B.S. degree in information security from the Xi’an University of Posts and Telecommunications, Xi’an, China, in 2017. She is currently working toward the M.S. degree in information security at the Beijing University of Posts and Telecommunications, Beijing, China. Her research interests include the areas of software security and data analysis.

Junfeng Wang received the M.S. degree in computer application technology from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2001 and the Ph.D. degree in computer Science from the University of Electronic Science and Technology of China, Chengdu, China, in 2004. From July 2004 to August 2006, he held a postdoctoral position with the Institute of Software, Chinese Academy of Sciences. Since August 2006, he has been with the College of Computer Science and the School of Aeronautics & Astronautics, Sichuan University, as a Professor. His recent research interests include network and information security, spatial information networks, and data mining.

Weishi Li received the B.Eng. degree in telecommunication engineering and management from the Beijing University of Posts and Telecommunications, Beijing, China, in 2018. He is currently working toward the M.Eng. degree in cyberspace security at the Beijing University of Posts and Telecommunications. His research interests include the areas of software security and data analysis.