Research on Deep Learning Vulnerability Detection Method Based on Fusion Features

Shuai Liu(✉) and Guan Wang

School of Information Security, University of Technology of Beijing, Beijing, China
netomark@126.com, wangguan@bjut.edu.cn

Abstract. Software security flaw is one of the most important security problems nowadays. It may cause incalculable loss. However, today's vulnerability detection technologies mostly rely on a single method, such as expert rules or deep learning, which has low scalability and fails to achieve better detection effect in the face of complex situations. In order to achieve better results of vulnerability detection, this paper proposes a vulnerability detection method named MF-TD based on the combination of neural network and expert rules and fusion of two characteristics. This method uses the combination of expert rules to highlight the semantic relation information, deeply understand the code logic structure based on the expert rules, and use the operation diagram to capture the statistical form and internal relation between the codes, and finally fuse the features for detection. The effectiveness of MF-TD was demonstrated in two different data sets.

Keywords: vulnerability detection · Neural network · Expert rules · Fusion Features

1 Introduction

Software vulnerability detection technology is used to find software vulnerabilities, but also an important means to ensure system security. However, since there is no uniform attribute of software vulnerability, most cases in detecting software vulnerability are undeterminable [1]. And through research, it is found that most software vulnerability detection methods can only play a certain role in some cases and have certain limitations.

The most common vulnerability detection method is static vulnerability detection based on expert rules. But this approach relies too heavily on human experts to define characteristics. Recent studies have used deep learning to infer program structure to identify potential software vulnerabilities in source code [2]. For example, [3] designed a deep learning vulnerability detection system named VulDeePecker, and used the bidirectional Long Short-Term Memory (LSTM) neural network for vulnerability detection, proving that the timing model has a significant effect on code semantic analysis and extraction. Based on VulDeePecker, [4] proposed μVulDeePecker, which improves VulDeePecker to capture only data dependent defects, and achieves higher efficiency in multi-classification vulnerability detection by increasing control dependency. However, for the vulnerability detection method based on deep learning, whether it can accurately

P. Qi and Z. Chen (Eds.): ICBDIE 2023, AISR 178, pp. 909–914, 2024.
https://doi.org/10.2991/978-94-6463-238-5_117

identify the logic and semantics of complex code cannot be verified. For the detection of complex and changeable vulnerabilities, more information should be combined to reduce the poor detection effect caused by information loss.

Therefore, in order to improve the performance of vulnerability detection and deal with complex and changeable vulnerabilities, a new vulnerability detection scheme named MF-TD is proposed, which includes automatic code processing and segmentation based on expert rules, automatic extraction of tree logical features and morphological features dual-dimensional features, and joint modeling detection by integrating sequential model and spatial model. It can solve the problem of low detection accuracy caused by information loss and improve the detection and recognition of covert threats.

2 Proposed Method

This paper proposes a vulnerability detection method based on a combination of neural networks and expert rules, called MF-TD (Fusion Feature Vulnerability Detection), which can automatically construct detection rule features by deeply understanding the logical structure and statistical morphology of the code. This method utilizes expert rules to construct code slice segmentation methods and build features to capture logical relationships between codes, improve the accuracy of identifying known vulnerabilities, and further use a high-precision deep network architecture to automatically obtain fusible features. Extract control operation and data operation semantics from code slices, and construct operation graph features to capture statistical patterns and internal relationships between codes, improving the accuracy of identifying hidden vulnerabilities. MF-TD consists of four modules, namely, Data Processing Module, Logic Feature Construction Module, Shape Feature Learning Module, and Feature Fusion Detection Module. The overall structure of this method is shown in Fig. 1.

2.1 Data Processing Module Based on Expert Rules

According to expert rules, attacks due to vulnerabilities often occur in the misuse of library functions. Therefore, this paper uses a pattern extraction and data preprocessing

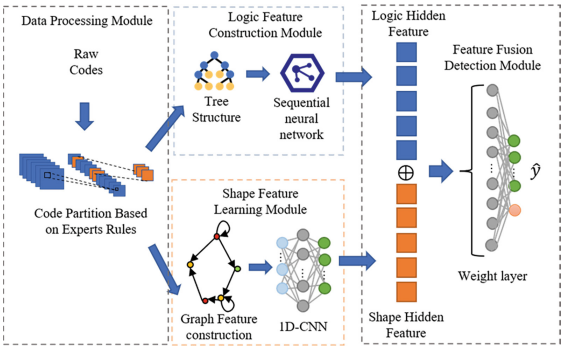

Fig. 1. Overall structure of MF-TD

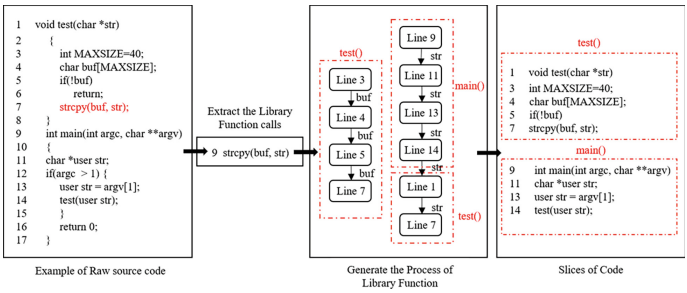

Fig. 2. Process example of data processing module based on expert rules

method based on library functions to extract code slices and track suspicious operation flows. Figure 2 illustrates the basic flow of this module. First, scan the input source code for library function calls. Secondly, match the calling parameters or variables of the library function, find the corresponding statements, and build the flow process. Finally, relevant statements are cut and pieced together to form a code slice for subsequent input.

2.2 Logic Feature Construction Module

The logic feature construction module takes code slice samples as input, and the specific implementation process is as follows: First, extract the execution tree of the code slice. In order to obtain execution logic that can understand code slices and extract the jump order of execution, the code AST tool is used to construct an execution tree for the code slices during the process, with operators in the root node of the execution tree. For example: function identification, variable definition, invocation, etc. Secondly, use a language library to define an operation command dictionary and assign command values. It is necessary to use a language manipulation library to establish a corresponding numerical dictionary in the form of neural network operability analysis. After that, traverse the root node of the tree to obtain the execution logic, path, and corresponding label. Traverse the root nodes of the tree according to the construction method of the execution tree, without traversing the leaf nodes to remove the noise caused by different variable definitions in different programs, and use the operation command dictionary to obtain sequence features that have execution logic and paths and are operable by the neural network.

Finally, a sequence based neural network is constructed. This paper constructs a BiLSTM neural network based on sequence characteristics. The sequential neural network constructed in this paper consists of multiple BiLSTM layers, a dense layer, and a softmax layer. The softmax layer takes the low-dimensional vectors received from the dense layer as input, and is responsible for the representation and formatting of classification results, providing feedback for the update of neural network parameters during the learning phase. The output of the learning stage is a BLSTM neural network with finely tuned model parameters. The output of this stage is the output of softmax which is called execution logic features L_f.

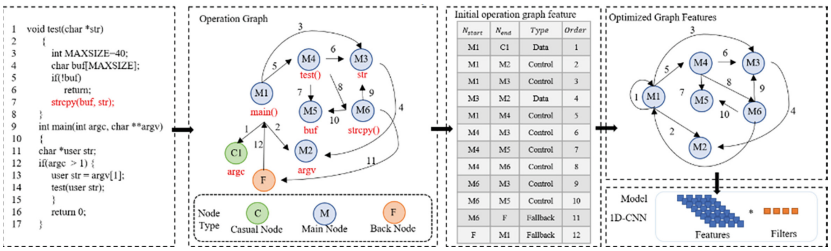

Fig. 3. Structure of shape feature learning module

2.3 Shape Feature Learning Module

Figure 3 shows the structure of the shape feature learning module based on the code operation diagram. This module takes code slice samples as input, extracts the semantics of control operations and data operations, constructs the features of operation graph, and, based on the characteristics of statistical information, highlights the structural features of key node information screening, builds the neural network model, and outputs the fusing result features. To build the sufficient neural network, CNN can well identify simple patterns in data and then use these patterns to form more complex patterns at higher layers. The 1D-CNN model is proven to be effective when potential features are obtained from shorter segments of the overall dataset and the positional correlation of the features in the segments is not high. The present invention uses 1D-CNN in the Keras library to establish a model, which consists of three layers of 1D-CNN, pooling layer, activation layer, and softmax. Input the classification labels of the features and samples predicted by the model, and output the neural network connection layer results which is called operation diagram feature R_f.

2.4 Fusion Feature Detection Module

Combining expert rule execution logic features and operation diagram features: After obtaining expert rule execution logic features L_f and operation diagram feature R_f, the invention combines two dimensional features:

$$X_f = L_f \oplus R_f \tag{1}$$

Transfer to the feedforward neural network to obtain the detection results: According to the combined characteristics of step 31, transfer to the feedforward network composed of two fully connected layers and one sigmoid layer, and the process is expressed as follows:

$$\hat{y} = sigmoid(FC(X_f)) \tag{2}$$

3 Experimental Result

This section conducts experiments on MF-TD in the context of conventional software security vulnerability scenarios and firstly compares the experimental results with the Flawfinder and Checkmarx, the currently popular static code vulnerability detection

tools. The method in this paper is further compared with other advanced deep learning related works Lin [5] and Vuldeepecker [3]. NVD and SARD, two data set sources commonly used in the field of vulnerability detection, are used for experiments on MF-TD. Two common vulnerability types buffer error (CWE-119) and Resource management error (CWE-399) were extracted from two sources, a total of 3330 samples. In this paper, four metrics are commonly used in the field of vulnerability detection to evaluate the detection performance of MF-TD. Model test performance evaluation metrics including: accuracy rate, accuracy rate, recall rate and F1-Measure. The comparison results are shown in Fig. 4.

As shown in Fig. 4, Fig. 4(a) and (b) respectively show the results of MF-TD and two static code vulnerability detection tools on four detection metrics of vulnerability CWE-199 and vulnerability CWE-399. Figure 4(c) shows the comparison of detection effects on total samples. In the experiment of the code data set of conventional software security vulnerabilities, the method Flawfinder and Checkmarx, a static code identification tool based on expert rules, are also compared in this paper. The method MF-TD proposed in this paper Flawfinder is a comparison of four metrics of the identification of key vulnerability buffers and resource management errors. Or comprehensive software security vulnerability data inspection, have reflected the great effect of improvement.

Figure 4(d), (e) and (f) respectively show the comparison of two kinds of vulnerabilities and comprehensive detection effects of MF-TD and two advanced related works. In the experiment of conventional software security vulnerability code data set, the method MF-TD proposed by Lin showed 3.08%, 5.51%, 2.57% and 2.28% improvement in accuracy, precision, recall and F1-Measure, respectively. Compared with Vuldeepecker baseline method, there are 1.74%, 2.04, 1.72 and 1.95% improvements in the four detection indexes, respectively. Thanks to the extraction form of fusion features in this paper, the detection effect of MF-TD has been improved in comparison with related work.

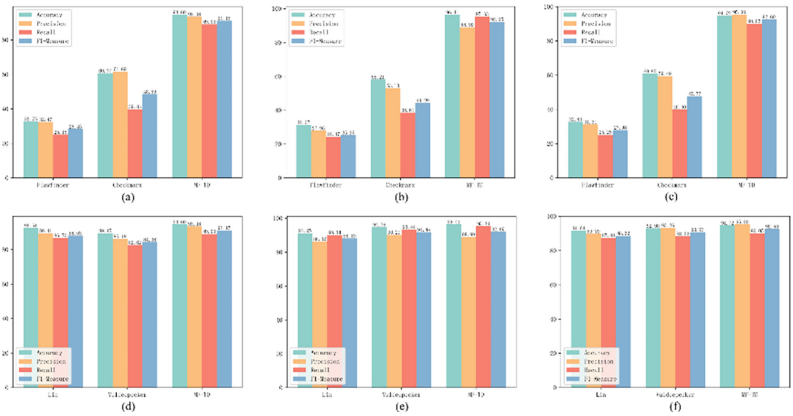

Fig. 4. Comparison results of MF-TD

4 Conclusion

Compared to existing intrusion detection systems, MF-TD (Fusion Feature Vulnerability Detection) proposed in this paper has the following two advantages. First, it has a code slice segmentation mode that combines expert rules, which can accurately locate vulnerabilities and reduce false positives. Secondly, it has a dual dimensional feature extraction method that can automatically capture the logical relationships of code execution and the morphological and endogenous relationships of code semantics, solving the problem of low detection accuracy caused by information loss. Thirdly, it can jointly model two dimensional features, in-depth analyze the knowledge learned from deep network structures, and utilize the representation capabilities of neural networks to improve the detection and recognition of hidden threats.

And the experimental results show that the method proposed in this paper has better accuracy and effectiveness compared to traditional vulnerability detection methods.

References

1. H. G. Rice. Classes of recursively enumerable sets and their decision problems[J]. Journal of Symbolic Logic, 1954, 19(2): 358–122.
2. N. Jovanovic, C. Kruegel, E. Kirda. Pixy: A static analysis tool for detecting web application vulnerabilities[C]//In Proceeding of IEEE Symposium on Security and Privacy. 2006
3. Z. Li, D. Zou, S. Xu, et al. VulDeePecker: A Deep Learning-Based System for Vulnerability Detection[J]. arXiv preprint arXiv: 1801.01681, 2018.
4. D. Zou, S. Wang, S. Xu, et al. μVulDeePecker: A Deep Learning-Based System for Multiclass Vulnerability Detection[J]. IEEE Transactions on Dependable and Secure Computing, 2021, 18(5): 2224–2236.
5. G. Lin, J. Zhang, W. Luo, et al. Cross-project transfer representation learning for vulnerable function discovery[J]. IEEE Transactions on Industrial Informatics, 2018, 14(7): 3289–3297.