Deep Program Representation Learning Analysis for Program Security

Na Li¹,a, Haoyu Zhang¹,b, Zhihui Hu¹,c, Guang Kou¹,d and Huadong Dai*¹

¹Artificial Intelligence Research Center, National Innovation Institute of Defense Technology, Beijing, China
²email: nli20070917@163.com, ³email: zhanghaoyu10@nudt.edu.cn, ⁴email: zh_h769023502@163.com, ⁵email: kg5188@163.com
*Corresponding author’s e-mail: hddai@vip.163.com

Abstract As the scale of software and the complexity of programs continue to grow, it is hard to meet development of modern computer technology only by manually extracting program features. In recent years, deep learning has achieved rapid development in different fields. Program representation learning based on deep learning has been widely used in many works, such as software vulnerability analysis, program analysis and malware detection. And it has gradually become a hot research direction in information security. After the deep analysis of existing research work on automatic program security detection, we catalog deep program representation learning for program security based on data representation and provide a comprehensive overview of deep program representation learning for program security under different application scenes. Then, we propose a deep program representation learning framework for program security. Finally, we conduct comparative analysis and summarize the challenges in deep program representation learning for program security.

1. Introduction

Program code is a core part of software engineering, and its own bugs and vulnerabilities may cause losses to software systems. With the development of complex programs and the wide application of various kinds of software, the number of software vulnerabilities is increasing [1]. Therefore, it is urgent to carry out a comprehensive and systematic research on program security, which can provide security for information society and online life. However, traditional software vulnerability detection methods rely too much on rules defined by experts, which results in high false positive rates [2]. Researchers have begun to use automated methods in software vulnerability analysis [3] in recent years. Deep learning has shown great potential for solving problems in different fields. Meanwhile, deep learning algorithms have been introduced into malicious code detection and program vulnerability analysis. Deep learning can learn relevant information and get laws from a large amount of source code, so that it can judge and predict vulnerable codes. The core problem lies in the program representation method based on deep learning [4-5]. Deep program representation learning [6-9] (program representation learning based on deep learning) which uses deep neural networks to replace artificial feature extraction, analyzes and abstracts the program to extract feature representations which are effective for the program. Thus, it solves the problems of complex semantics, large-scale and sparse vocabulary in the expression of program data, to better serve the follow-up tasks.
Deep program representation learning captures relevant information between codes from a large amount of input program data in the way of neural network connection. While obtaining the efficient features of the program, it reduces the complexity and dimensionality of the input program data. Therefore, it has been widely used in analysis in large-scale programs. In this article, the deep learning-based program representation learning method is studied in the field of automatic detection of program security. Based on the comprehensive analysis of the existing research results, the current representative research methods are deeply summarized and compared, and then the deep program representation learning framework for program security is proposed. Finally, the deep program representation learning method for program security is analyzed.

2. Deep program representation learning methods for program security

2.1. Program representation learning and deep learning techniques

Program representation learning is the representation of the characteristics of the original input program data. Different program representation features can contain or omit the relevant information of the original data. The quality of program representation learning determines the effect of the later tasks. Early program feature acquisition [10-11] was mostly based on human prior knowledge, and human experts made rules and carried out feature analysis. There were great limitations in the extraction process, including over-reliance on expert rules, failure to try large-scale programs, poor generalization performance of manual definition, etc. With the continuous development and improvement of deep learning technology, detection engine program is characterized by constantly training and learning, and finally realize the inspection does not rely on predefined rules of malicious code detection work. The success of automated detection systems is largely attributed to effective program representation learning.

The program representation learning method based on deep learning is to mine the implicit feature relationships between programs by building deep neural networks, so as to overcome the problems of complex semantic relationships and sparsity among program sample vocabulary, so as to improve the efficiency and accuracy of later detection tasks. Deep learning and program representation methods are combined. Program representation, as the front end of deep representation learning, provides a data basis for program feature extraction based on deep neural networks; deep neural networks replace traditional manual methods to obtain program feature information. Deep learning techniques commonly used in deep program representation learning include Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Deep Belief Network (DBN), and attention mechanism [12] (attention), etc., which can be used in different program features extracting application.

2.2. Classification and application

This paper presents a classification system of deep program representation learning methods for program security, including deep program representation learning based on sequence, deep program representation learning based on abstract syntax tree and deep program representation learning based on graph. The following will describe the research work of the three methods in combination with the application scenarios of the automatic detection of program safety.

2.2.1. Deep program representation learning based on sequence. Due to the weak expressing ability of program based on symbol sequence and lack of the expression of code syntax and semantic information, program safety automatic detection mostly uses program marker sequence as the research object. The core idea of deep program representation learning based on token sequence is to convert program code into token sequence for characterization through lexical analysis, and use deep neural network for modeling to obtain the feature vector of token sequence. [13] collects 9772 32-bit Linux program function call sequences and the dynamic values of their parameters as the static and dynamic characteristics of the binary program. The token is constructed through Keras, and the function call sequence is converted into a digital sequence for labeling. Different types of deep neural networks
extract features. [14] propose a data-driven method based on machine learning to mine software vulnerabilities for C/C++ programs. The code function is used as the detection granularity to generate a tag sequence from a custom lexical analyzer, and Word2vec [15] is used to vectorize and process the sequence. The features are extracted through the convolutional network. [16] marks each line of the program code segment, and each \( n \) marks are used as the basic unit for code feature learning. [17] takes the code segment containing the vulnerability information as the basic data unit to form the label sequence, and uses the convolutional network combined loop network to extract the features of the vectorized data. [18] uses data dependence and control dependence to label code sequences, which obtains global semantic information related to program vulnerabilities. It proposes and uses two sets of Bi-directional Long Short-Term Memory (BiLSTM) with the same settings but different sizes to extract global and local features. And add an attention mechanism to the network for specific code vulnerability attributes.

2.2.2. Deep program representation learning based on abstract syntax tree. Abstract Syntax Tree [19] (AST) is an abstract representation of source code using a tree structure. It is currently widely used in related work on program semantic understanding. It uses tree-shaped nodes to connect the variables, constants, and constants of the program. Representation of the relationship between methods and function calls is an effective expression of program syntax and structure. The deep program representation learning based on abstract syntax tree combines the tree representation of program code with the deep neural network model. The intermediate representation of feature extraction can effectively retain the grammatical rules of the program code. [20] uses the function in the code as the research unit to extract the abstract syntax tree structure, and analyzes the interaction relationship between the codes through the structured representation of the function. [21] converts the program code into an abstract syntax tree for vectorized embedding, uses Deep Belief Network (DBN) to map the feature vector to a low-dimensional space to extract the semantic features of the program, and form labeled data for classifier training. [22] tries to reveal the defects of the programming model by analyzing the structure of the abstract syntax tree representation of the function, which converts the marked function data into an AST sequence, and uses a two-way long and short-term memory network to learn the AST feature representation. [23] proposes a model for automatically learning the semantic and syntactic features of AST sequences. It constructs a look-up table method to map the marked AST sequences to fixed-length vectors, and uses a long- and short-term memory network combined with seq2seq technology to analyze vectorized features Detection. In order to fine-grain the vulnerability targets, [24] proposes a method to extract the sequential features of the abstract syntax tree. The AST sequence is embedded in the vector space through the Discrete Fourier Transform (DFT), and the feature representation of the deep AST is obtained based on the deep learning method.

2.2.3. Deep program representation learning based on graph. The graph representation of a program is a graphical representation of the data structure of the program. In the research of program safety automation detection, it is mainly manifested as the control flow graph (CFG) and program dependency graph of the program (PDG). In the graphical representation of the program, the data flow of the program and the dependency relationship between the codes are modeled. Nodes represent the elements of the code, such as variables, expressions and addresses. Edges describe the data flow and dependency relationships between nodes. This graphical program representation method can effectively analyze program syntax and semantics, function calls, and dependencies among variables. With the development of deep learning technology, graph-based deep program representation learning is used in program security detection, including malicious code copy detection, vulnerability detection, and defect detection. [25] uses extended code attribute diagrams to represent program functions to realize automated detection of pollution vulnerabilities. Code attribute diagrams are constructed from abstract syntax trees, program dependency diagrams, and program control flow diagrams, which are expanded by adding priority information of function statements. [26] proposes a binary code similarity detection algorithm based on graph neural network, which uses control flow chart to characterize the binary function, artificially
designs feature extraction and maps low-dimensional space. Meanwhile, it uses Structure2vec [27] for graph embedding. [28] initially classifies the vulnerability rules, uses program slicing technology to derive the control dependency graph, and expands the grammatical features through data dependency and control dependency. [29] proposes a vulnerability detection model based on graph hierarchical classification, which realizes feature extraction by adding code sequence edge attributes and fusing classic program analysis methods. [30] proposes an automated code vulnerability detection method based on minimum intermediate representation learning. In the pre-training stage, the extended corpus is embedded in the vector space through dependency analysis, slicing, and serialization. The convolutional network is used to obtain the vulnerability feature representation for the classification model.

3. Deep program representation learning framework for program security

Deep program representation learning for program security is the application of program representation learning methods in software vulnerability analysis and prediction. As shown in Figure 1, the framework uses program text as the input data sample space, and finally generates feature vectors as output results. The core is a breakthrough in key technologies such as program representation, feature extraction based on deep learning, and data mining.

![Figure 1. The deep program representation learning framework for program security](image)

This paper starts from the research framework proposed above, and studies the existing research results of deep program representation learning for program security from three dimensions: task analysis, data preprocessing methods, program semantic representation selection, and deep neural network-based feature extraction methods. The specific description is as follows:

- **Data preprocessing method for task analysis**, that is, how to obtain data suitable for program representation. Use the program code as the sample space for input, select different methods to operate on the original input data according to the requirements of later tasks, and obtain the preprocessed data that meets the analysis requirements. The operations include deleting duplicate data, adding/removing related program information, and unifying data sample length, etc. This dimension can improve the versatility of data in different program representation methods to a certain extent, thereby comprehensively improving the research efficiency in the direction of program safety automation detection.

- **The choice of the semantic representation of the program**, that is, the semantic representation of the program is incorporated into the prior knowledge of the code so that the subsequent feature extraction model can better understand the program semantics. The program can be expressed in many forms, and different data structures can be obtained by choosing different
representation methods. According to the presentation form of data, this paper divides the program representation methods into sequence representation, abstract syntax tree representation and graph representation. Compared with the program representation method based on sequence, the program representation method based on abstract syntax tree and graph can express the structural characteristics of source code more intuitively and efficiently.

- The feature extraction method based on deep neural network, that is, the use of deep learning technology to mine the implicit correlation between program codes. The program representation data is used as input. The program features are combined and extracted through neural network connection. The final result is output in the form of vectorization. The feature extraction method is based on the program representation. The results of the program representation serve feature extraction while restricting. On the one hand, different algorithms for feature learning have different requirements for the input of the representation data. On the other hand, the quality of the data directly affects the quality of the extraction.

4. Comparison and analysis

A comparative analysis of the deep program representation learning methods for program security is carried out. Table 1 summarizes the technical characteristics of the above-mentioned different research results. As shown in Table 1, based on the combing of the prior art methods, the following analysis can be made:

- At present, the research on automated software vulnerabilities detection based on deep learning takes the program source code as the main research object. The program source code is composed of text with grammatical rules and has the ability to further express high-level semantics. Compared with binary code, source code is closer to natural language, and advanced technology of natural language processing can be introduced in the problem of program understanding based on deep learning technology. Meanwhile, the problem of binary code program understanding should face many challenges, and the analysis methods based on deep learning are still being explored.

- Program representation is the basis for the establishment of deep program representation learning. While drawing on natural language processing technology, it is different from natural language processing. Sequence program representation converts the program code into a linear sequence for representation without considering the program structure. The representation process is relatively easy to operate and can be used for large-scale software, but the flat representation does not consider the grammar and syntax of the program. Semantic analysis can easily cause the loss of program information. The program representation based on the abstract syntax tree effectively retains the grammatical structure of the program and has a good detection effect. However, due to the high technical complexity and long dependence, it is difficult to promote large-scale detection tasks. The graph representation method of the p-program takes full consideration of program syntax and semantic information, and effectively improves the accuracy of feature representation. But, the establishment of graph network model is complicated, and there are problems such as high algorithm difficulty, high time and space overhead. Regarding the automated detection of program security, it is meaningless to compare the effectiveness of the three methods on the premise of setting aside the data scale, detection granularity and task background. Therefore, in the field of program security, the three methods can be further discussed based on the establishment of benchmark data sets.

Table 1. Summarization of relative researches on deep program representation learning for program security

| Paper | Data           | Program Representations | Embedding              | Representation Learning |
|-------|----------------|-------------------------|------------------------|------------------------|
| [13]  | Binary code    | Sequence                | tokenization tool by Keras | LSTM, CNN, FCN         |
| [14]  | C/C++ code     | Sequence                | Word2vec               | CNN-LSTM               |
| [16]  | C/C++ code     | Sequence                | Word2vec               | CNN                    |
### Table 1: Code Implementation of Deep Learning Techniques

| No. | Language | Source | Feature Extraction | Learning Method |
|-----|----------|--------|--------------------|-----------------|
| 17  | C/C++ code | Sequence | a fixed $k$-dimensional | CNN, RNN |
| 18  | C/C++ code | Sequence | NNLM | NNLM |
| 20  | C/C++ code | AST | SVD | SVD |
| 21  | Java code | AST | $k$-nearest neighbor algorithm | DBN |
| 22  | C/C++ code | AST | Word2vec | RNN |
| 23  | Java code | AST | Bag-of-Word | seq2seq RNN |
| 24  | C/C++ code | AST | DFT Word2vec-CBOW | RNN |
| 25  | C/C++ code | Graph | Bag-of-words K-means | K-means |
| 26  | Binary code | Graph | Structure2vec | Structure2vec |
| 28  | C/C++ code | Graph | Word2vec | Word2vec |
| 29  | C/C++ code | Graph | Word2vec | GNN |
| 30  | C/C++ code | Graph | Word2vec | CNN |

- Target entity vectorization is a basic work based on deep learning analysis methods, and it is the data source of deep neural network input. As a commonly used input format, vectorized data has different implementation methods, mainly relying on natural language processing technology. Starting from the goal of improving the feature extraction ability and reducing the complexity of mapping, embedding the representation data into the vector space is the link between the program representation and deep learning. Choosing a suitable vectorization method will help to improve the accuracy of program feature extraction based on deep learning methods, improve generalization performance and save time and space overhead.

- Feature extraction is the core of deep program representation learning. The extraction method is based on the understanding of program features, combined with the representation method to build the deep neural network model algorithm. The choice of the model is closely related to the detection target, granularity and research ideas. In the field of deep program representation learning for program security, deep representation learning technology is used in program safety automated inspection tasks to replace manual feature extraction. The method is mainly to extend the application of the classic neural network model.

### 5. Conclusions

The close integration of program representation and deep learning methods has promoted the reform of program vulnerability detection methods and realized the improvement of software security assurance capabilities. At present, the research of program vulnerability methods based on deep learning is still in the early stage and still needs to continue to analyze and explore:

- At present, most developers are researching on self-built data sets, and the lack of high-quality data sets severely restricts the rapid progress of related fields;
- The generalization performance between the existing program representation methods is poor, and large-scale migration applications cannot be carried out;
- How to better combine the rich semantic and syntactic features of code is still the focus of future research in the field of deep program representation learning for program security.

Deep program representation learning is an important research field of automated detection technology for program security. The effective analysis and extraction of program code features based on deep learning methods provide technical support for the automatic mining of software vulnerabilities. This paper sorts out the representative working principles and techniques of deep program representation learning for program security in recent years. First, a research framework for deep program representation learning is proposed. Second, the current research methods are explained from multiple angles, and the comparisons are made respectively. Finally, the problems and challenges in the research field are discussed. Program representation learning based on deep learning is a research hotspot in the direction of program security. It is of great significance to discover software program defects and maintain the stability of system engineering.
References

[1] National Vulnerability Database. (2021). https://nvd.nist.gov.

[2] Brooks, T.N. (2018) Survey of automated vulnerability detection and exploit generation techniques in cyber reasoning systems[C]//Proceedings of the Science and Information Conference, Springer, UK. 10–12.

[3] Lin, G.J., Wen, S., Han, Q.L., et al. (2020) Software Vulnerability Detection Using Deep Neural Networks: A Survey[J]. Proceedings of the IEEE, PP(99): 1-24.

[4] Lin, G.J., Zhang, J., Wei, L., et al. (2017) POSTER: Vulnerability discovery with function representation learning from unlabeled projects[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, ACM, USA, 2539–2541.

[5] Li, Z., Zou D.Q., Xu, S.H., et al. (2020) VulDeeLocator: A Deep Learning-based Fine-grained Vulnerability Detector[J]. arXiv preprint arXiv:2001.02350.

[6] Bengio, Y., Courville, A., Vincent, P., (2013) Representation Learning: A Review and New Perspectives[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence. 35(8):1798-1828.

[7] Allamanis, M., Brockschmidt, M., Khademi, M., (2017) Learning to represent programs with graphs [J]. arXiv preprint arXiv: 1711.00740.

[8] Alon, U., Zilberstein, M., Levy, O., et al. (2018) A general path-based Representation for predicting program properties[C]//Proc of the ACM SIGPLAN Notices. ACM, New York, 404-419.

[9] Alon, U., Zilberstein, M., Levy, O., et al. (2019) code2vec: Learning distributed representations of code[J]. Proceedings of the ACM on Programming Languages. ACM, New York,3(POPL):1-29.

[10] Balzarotti, D., Cova, M., Felmetser, V., et al. (2008) Saner: Composing Static and Dynamic Analysis to Validate Sanitization in Web Applications[C]// Proc of IEEE Symposium on Security and Privacy. IEEE, USA, 387-401.

[11] Nielson, F., Nielson, H.R., Hankin, C., (2015) Principles of Program Analysis[M]. Springer, Berlin.

[12] Bahdanau, D., Cho, K., Bengio, Y., (2014) Neural machine translation by jointly learning to align and translate[J]. arXiv preprint arXiv:1409.0473.

[13] Wu, F., Wang J.G., Liu J.Q., et al. (2017) Vulnerability detection with deep learning[C]//IEEE International Conference on Computer and Communications. IEEE, Chengdu , 1298-1302.

[14] Harer, J.A., Kim L.Y., Russell, R.L., et al. (2018) Automated software vulnerability detection with machine learning[J]. arXiv preprint arXiv: 1803.04497.

[15] word2vec. [OL]. (2021) http://radimrehurek.com/gensim/models/word2vec.html.

[16] Li, Z., Zou, D.Q., Xu S.H., et al. (2018) VulDeePecker: A Deep Learning-Based System for Vulnerability Detection[C]//Network and Distributed System Security Symposium. The Internet Society, USA.

[17] Russell, R.L., Kim, L., Hamilton, L.H., et al. (2018) Automated Vulnerability Detection in Source Code Using Deep Representation Learning[J]. arXiv preprint arXiv: 1807.04320.

[18] Zou, D.Q., Wang, S.J, Xu, S.H., et al. (2020) µVulDeePecker: A deep learning-based system for multiclass vulnerability detection[J]. arXiv preprint arXiv: 2001.02334.

[19] Rabinovich, M., Stern, M., Klein, D., (2017) Abstract syntax networks for code generation and semantic parsing[J]. arXiv preprint arXiv:1704.0753.

[20] Yamaguchi, F., Lottmann, M., Rieck, K., et al. (2012) Generalized vulnerability extrapolation using abstract syntax trees[C]//The 28th Annual Computer Security Applications Conference. ACM, U-SA, 359-368.

[21] Wang, S., Liu, T.Y., Tan, L., (2016) Automatically Learning Semantic Features for Defect Prediction[C]//IEEE/ACM 38th International Conference on Software Engineering., IEEE/ACM, USA, 297-308.

[22] Liu, S.G., Lin, G.J., Han, Q.L., et al. (2020) DeepBalance: Deep-Learning and Fuzzy Oversampling for Vulnerability Detection[J]. IEEE Transactions on Fuzzy Systems, 28(7): 1329-1343.
[23] Dam, H.K., Tran, T., Ng, T.P., et al. (2017) Automatic feature learning for vulnerability prediction[J]. arXiv preprint arXiv: 1708.02368.
[24] Lin, G.J., Zhang, J., Lou, W., et al. (2018) Cross-Project Transfer Representation Learning for Vulnerable Function Discovery[M]// IEEE Transactions on Industrial Informatics. IEEE, USA, 3289-3297.
[25] Yamaguchi, F., Maier, A., Gascon, H., et al. (2015) Automatic inference of search patterns for taint-style vulnerabilities[C]//Proc of 2015 IEEE Symposium on Security and Privacy. IEEE, USA, 797-812.
[26] Xu, X.J., Liu, C., Feng, Q., et al. (2017) Neural network-based graph embedding for cross-platform binary code similarity detection[C]//The 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, USA, 363-376.
[27] Dai, H.J., Dai, B., Song Le. (2016) Discriminative Embeddings of Latent Variable Models for Structured Data[J]. arXiv preprint arXiv: 1603.05629.
[28] Li, Z., Zou, D.Q., Xu, S.H, et al. (2018) SySeVR: A framework for using deep learning to detect software vulnerabilities[J]. arXiv preprint arXiv: 1807.06756.
[29] Zhou, Y.Q., Liu, S.Q., Siow, J.K., et al. (2019) Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks[C]//Proceedings of the Advances in Neural Information Processing Systems. NeurIPS, Canada, 10197-10207.
[30] Li, X., Wang, L., Xin, Y., et al. (2020) Automated Vulnerability Detection in Source Code Using Minimum Intermediate Representation Learning[J]. Applied Sciences, 10(5): 1692-1708.