ISSN 2706-655X Vol.4, Issue 4: 21-32, DOI: 10.25236/IJFET.2022.040404

Source Code Vulnerability Mining Method Based on Graph Neural Network

Guo Li

Department of Journalism and Communication, Anhui Vocational College of Press and Publishing, Hefei, 230601, Anhui, China

Abstract: *In recent years, deep learning has completely changed many machine learning tasks, and the data in these tasks is usually expressed in Euclidean space. However, as more and more applications need to use non-Euclidean data, vulnerability mining is becoming more and more important. With the successful development of neural networks, many machine learning tasks, such as object detection, image classification, and speech recognition, once relied heavily on manual feature engineering to extract features, and can now be completed with various end-to-end deep learning models, such as Convolutional neural network, long and short-term memory networketc.Vulnerability mining is an important way to prevent and control system vulnerabilities. Traditional methods of vulnerability mining can no longer meet people's needs. In order to enable the vulnerability mining application to meet people's needs, we established a related source code vulnerability mining model based on graph neural networks. By investigating relevant literature, conducting interviews with professionals, etc., collected data from databases such as HowNet, Wanfang Database, SSCI, etc., and built a model of source code vulnerability mining based on graph neural networks using parallel algorithms. Through simulation, we found that the method of mining source code vulnerabilities based on graph neural networks is becoming more and more accepted by people, and the increase in 2016 reached 0.16. Moreover, the efficiency of source code vulnerability mining based on graph neural network is much higher than other vulnerability mining methods, and the mining speed is more than 20% ahead of other mining methods. This shows that source code vulnerability mining based on graph neural network can play an important role in preventing system vulnerabilities.*

Keywords: *Graph Neural Network, Machine Learning, Source Code, Research Method*

1. Introduction

Vulnerability discovery is the focus area of current computer security research. At present, most of the vulnerability mining methods require a lot of manual auditing work, but the code analysis is complicated and time-consuming, resulting in inefficient vulnerability mining. In view of the undecidability of the vulnerability mining problem itself, it is a basic tool design principle to assist security analysis instead of completely replacing security analysts [1]. Through the analysis of system attack events in recent years, security vulnerabilities are the main cause of hidden security risks in the system, and attackers can use security vulnerabilities to attack the system. With the help of graph neural network mining method, the use of self-contained information in source code to assist software analysis is a novel research idea [2].

The current general approach to system vulnerabilities is a fuzzing method based on network protocols. The discovery of system vulnerabilities improves the effectiveness of test cases, but these research results still have some problems[3-4]:

Li Yun believes that as time changes, software has become more and more complex, its scale and vulnerabilities are also increasing, and more and more diverse, traditional vulnerability mining methods can no longer adapt, there are high false positives, false negatives, etc. problem. It believes that to solvethe problem of system vulnerabilities, it is necessary to conduct vulnerabilities mining based on machine learning. It sorts out the process of machine learning, summarizes relevant experience, and makes relevant elaboration on the preparation work required for machine learning. The attention issues and the prospects of machine learning are related to the discussion [5]; Duan Bin believes that in today's frequent control system accidents, it is necessary to innovate vulnerability mining technology to ensure system security. The solution given is based on the method of dynamic taint analysis, explaining the theory of dynamic taint, designing related experiments, quantifying dangerous and sensitive words

ISSN 2706-655X Vol.4, Issue 4: 21-32, DOI: 10.25236/IJFET.2022.040404

that cause loopholes, making guiding information, importing it into the model, and comparing experiments, It is proved that the method based on dynamic taint can improve the efficiency of the system in vulnerability mining [6]. Lin Liangcheng believes that after vulnerability mining has become an important field, vulnerability mining can be carried out through fuzzing. In the article, he explained the meaning of the fuzzing theory and made a test model. Use relevant examples to run in the model, and use program instrumentation to collect system exceptions and discovered vulnerabilities, avoiding a large number of vulnerabilities in the system. In this experiment, the fuzzing method has achieved better than traditional vulnerabilities. Collect better results [7].

This article discusses vulnerability modeling and vulnerability discovery methods based on code attribute graphs. The generation strategy of abstract syntax tree, control flow graph, and program dependency graph is elaborated, and its constituent elements are analyzed in detail. It has opened up a new theoretical perspective for deep chemists' knowledge and understanding of vulnerability mining, standardized definitions of common vulnerability modes, focused on the polluting vulnerability query design with the strongest query capabilities, and conducted experiments on the description of the vulnerability query Verification and case analysis.

2. Source Code Vulnerability Mining Method

2.1 Graph Neural Network

Graph neural network is used to process data in the graph domain, for an undirected graph $G=(v,e,w)$, Here V refers to the nodes in the graph, e refers to the edges that exist between nodes, and w refers to the similarity parameter between nodes in the graph domain [8]. The graph neural network takes graph structure as input. For every node in G there is $v=[p,t]^T$, Where p is used to represent the location information corresponding to the node, and t is used to represent the texture information of the node.

In the graph neural network, each node has only the k nearest neighbors whose weight is closest to the edge e[9]. For each node, their local feature F is expressed as:

$$F=[\varphi_{i1}f_{i1},\varphi_{i2}f_{i2},...\varphi_{in}f_{ik}]^T \quad (1)$$

Their weights satisfy $w_{i1}<w_{i2}<...<w_{ik}$, φ in order to control the parameters contributed by the characteristics of the node, in order to highlight the contribution of the central node and weaken the contribution of the distant node，This paper defines the calculation method as:

$$\varphi_{ij}=1-\frac{i}{k}+\left(1-y_j^i\right)+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{l}\sum_{j=1}^{s_l+1}\left(\theta_{ji}^l\right)^2 \quad (2)$$

In order to improve the accuracy of the graph neural network, a model composed of multiple shallow convolutional neural networks is proposed, each of which is used to solve the same classification problem, and the final prediction result depends on multiple classifiers the result of the vote [10]. In terms of network topology, graph neural network uses edge network for communication. Geographically distributed fog nodes can infer their own location and track end-user equipment, and sense the information of nearby equipment, transmit messages in time, thereby improving real-time response. The distributed deployment characteristics of graph neural network at the edge of the network enable the edge of the network to directly calculate and store data and applications without other process steps [11]. Graph neural network can be understood as learning from other samples, so that it can absorb previous experience in vulnerability mining and continuously improve its efficiency [12].

2.2 Machine Learning

To complete the source code vulnerability mining method based on graph neural network algorithm, we must first study the theoretical basis of neural network, which is also an indispensable and important part of this subject research [13]. Due to the huge amount of calculation required for machine learning, it is unrealistic and impractical to rely solely on people to calculate, so the help of computers is needed [14].

ISSN 2706-655X Vol.4, Issue 4: 21-32, DOI: 10.25236/IJFET.2022.040404

For a machine learning algorithm, in order to verify the pros and cons of the algorithm and whether the algorithm can successfully solve people's problems, it must pass the evaluation and test of the model [15]. The sample error rate is defined as:

$$E=\frac{1}{m}\left[\sum_{i=1}^{n}\sum_{j=1}^{j}x^{j}\log\left(h_{\vartheta}\left(x^{j}\right)\right)\right]_{j}+\left(1-y_{j}^{i}\right)+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{l}\sum_{j=1}^{s_l+1}\left(\theta_{ji}^{l}\right)^{2} \quad (3)$$

Sample correct rate:

$$P=1-E=1-\frac{1}{m}\left[\sum_{i=1}^{n}\sum_{j=1}^{j}x^{j}\log\left(h_{\vartheta}\left(x^{j}\right)\right)\right]_{j}+\left(1-y_{j}^{i}\right)+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{l}\sum_{j=1}^{s_l+1}\left(\theta_{ji}^{l}\right)^{2} \quad (4)$$

The backpropagation algorithm is adopted to solve the difference of the first batch of data at the beginning [16]. Then adjust the weight value of each layer through the error.

$$\delta=\left(\mu^{2}\right)^{t}\varphi^{3}*g'\left(z^{3}\right) \quad (5)$$

In general, the difference between the predicted result of the model learned by the machine learning algorithm and the original result of the overall sample is called error[17]. For a good machine learning algorithm[18]. So what we can do is to make the training error smaller.

For machine learning algorithms, the most widely used judgment methods mainly include the leave-out method, cross-validation method, and self-service method. The retention method is to divide the data set into two mutually exclusive subsets, which can be assembled into a whole set [19].

2.3 Source Code

With the increasing importance of software security, people expect developers to fix vulnerabilities in software systems as much as possible. However, in the face of massive code files, comprehensive detection and repair of potential vulnerabilities are very time-consuming and expensive [20]. Therefore, how to use data mining and other methods to detect code defects and guide software testers to detect and repair potential vulnerabilities has become a research hotspot that scholars at home and abroad pay attention to. Code defect detection usually adopts analysis and extraction of features in program source code, and uses machine learning algorithms to automatically learn modules with defects in the code. According to the different analysis methods, it can be roughly divided into conventional vulnerability code analysis, data flow and control flow analysis, code text mining, etc. [21].

Although conventional vulnerability code analysis methods can find code vulnerabilities more directly, they have limited ability to detect vulnerabilities that differ greatly from known vulnerability patterns. More importantly, it requires researchers to have a wealth of professional knowledge in the field of vulnerability mining and a certain scale of extremely high-quality training samples, which poses a huge challenge to researchers across fields.

The learning of programming logic will play a vital role in many fields. In the field of vulnerability mining, by learning the programming logic of known vulnerabilities, you can find parts with similar logic in other programs, and provide guidance for traditional vulnerability mining techniques; in the field of software engineering, analyze and find out that does not conform to logical specifications The code snippets can improve the robustness and reliability of the software; in the field of software repair and automatic programming, through statistical analysis of programming logic, a code model is established to predict the missing part of the code statement [22]. Usually, each line of code in a program does not exist independently, and several lines of code before it jointly determine the appearance and use of the code.

Compared with natural languages, programming languages have stricter requirements on grammatical structure and data types. There may be certain constraints between different objects, methods and even parameters. These constraint relationships reflect the rules of the language itself and the potential way of thinking of developers. Making full use of these constraints can make the statistical language model break through the limitation of relying solely on the occurrence probability of sentences/keywords in natural language, and make the programming logic model more in line with the developer's way of thinking [23]. In the source code construction process, the time complexity T of building the model can be expressed as:

ISSN 2706-655X Vol.4, Issue 4: 21-32, DOI: 10.25236/IJFET.2022.040404

$$T = \sum_{i=1}^{1}\left((m_i - 2)*\gamma_i * p_i\right) \quad (6)$$

$$T \approx \left(\bar{m} - 2\right)*\bar{\gamma}*\sum_{i=1}^{1}\left(p_i\right) \approx C*\sum_{i=1}^{1} p_i \quad (7)$$

The methods in the data structure class often implement some of the simplest data operations, with low mutual dependence, flexible invocation, and weak constraint relationships between methods [24].

2.4 Vulnerability Mining

Traditional vulnerability mining methods mainly include three technical methods, namely reverse analysis, penetration testing and fuzzing testing. Reverse analysis is to obtain the source code of the program through the means of disassembly and debugging, and monitor the errors that may occur during the operation of the program. However, this technology requires a lot of manual code review work. The reviewers need to have excellent disassembly debugging technology and secure code evaluation capabilities. They require too much operating system professional skills for the staff, and the operating environment of the system is closed, and the system components cannot Export, it is difficult to disassemble and debug the system. Penetration testing technology is to establish a known attack vector to simulate attacks on the tested object and test the robustness of the tested object. This technology tests from the perspective of known attacks and cannot cover all the abnormalities that may cause the tested object. Circumstances, and the requirements for the staff's information security professional skills are too high, there is also the risk of industrial data information leakage when the system is handed over to the testing organization. Fuzzing testing technology is to detect vulnerabilities in the tested object in the form of "black box" testing through random input. It does not need to perform too many operations on the tested object. This technology has a high degree of automation and wide adaptation. According to the characteristics of industrial control systems and industrial control network protocols, corresponding improvements can be made. Therefore, domestic and foreign researchers often use fuzzy testing technology to construct network packets as test cases to mine the network protocols in the system.

At present, most of the knowledge extraction and vulnerability locating research for software source code methods are focused on the use of some cloning and redundancy phenomena in the source code. The reused API has a relatively fixed usage pattern, and many functions in the software are similar. As a result, in large-scale software, many program fragments are very similar, with only minor differences. So programmers are likely to make the same mistakes in similar coding scenarios. The repetitiveness and redundancy of the software source code can help us dig and use most of the correct repetitive code segments and obtain standardized programming information to identify programming errors caused by imperfect modification and code omissions after copying and pasting or creating branches.

The economic root cause that loopholes are difficult to eliminate is the asymmetry of loophole trading information. The manpower and material resources that software vendors invest in software security are not immediate. Sellers of vulnerable products have more information on software vulnerability information, that is, the quality evaluation of software products than buyers. This formed the "lemon market" effect proposed by Akerlov in economics. Buyers do not understand the pros and cons of products as well as sellers, and will choose products with average prices. As a result, high-quality products are eliminated because no one buys them. Inferior products are gradually replaced by medium-quality products in the current market due to low cost, which eventually leads to inferior products Flood the market. The discovery and verification of vulnerabilities still requires a high technical threshold. In the face of a wide variety of vulnerabilities, how to design a fast and efficient vulnerability mining method is still an urgent problem for computer software security researchers. Nowadays, the rise of a new wave of computer science and technology research represented by big data and deep learning has caused computer researchers to focus on new database technologies, data mining and machine learning. There have been many attempts to use data mining in software engineering. The application of data mining technology to the discovery of software vulnerabilities is still a relatively new topic, but there have been some preliminary research results.

Vulnerability extraction at the source code level first uses compilation technology to parse the source code and extract API symbols. Simply put, API symbols refer to the name of the called function, parameter types, and the types of local and global variables. Use information retrieval methods to treat functions as articles, calculate the TFIDF value of each API symbol, and then map the function to a

ISSN 2706-655X Vol.4, Issue 4: 21-32, DOI: 10.25236/IJFET.2022.040404

vector; perform principal component analysis, find similar program bugs by searching for functions with similar semantics.

$$s(x) = f(x) - \frac{\sum_i (\lambda - \varphi(x)) - f(x)}{\sin \varphi(x) - 1} \quad (8)$$

$$f(x) = \|\mu - \gamma(x)\|_\infty = \max_{n \in E} (\gamma_n - I(x, n)) \quad (9)$$

$$\frac{At_{\max}(q) - At_{\min}(p)}{y - At\min(q)} = \frac{p_\mu(q) - p_l(q)}{x - p_l(q)} \quad (10)$$

$$P(q) = \int_{P_l(q)}^{P_u(q)} f(x)ds,\ f(x) = \begin{cases} 1, a < x < b \\ 0, others \end{cases} \quad (11)$$

3. Source code vulnerability mining experiment based on graph neural network

3.1 Experimental Analysis Object

This paper conducts data statistics on different vulnerability mining systems, and compares the differences between them. Through relevant literature, statistics of these types of vulnerability mining methods over time, study the impact of related variables on system security, and build relevant model.

3.2 Establish a Model Evaluation Index System

Among them, the first-level evaluation index and the second-level evaluation index are relatively abstract and cannot be used as a direct evaluation basis. The third-level evaluation indicators should be specific, measurable and behavior-oriented, and can be used as a direct basis for teaching evaluation.

3.3 Determine the Evaluation Weight

Therefore, this article uses a combination of analytic hierarchy process and entropy method to determine the weight coefficient of each evaluation index of regional higher education.

3.4 Comprehensive Evaluation Model

In the specific implementation process, the two methods can be implemented separately. Finally, the results of the two models are compared.

4. Source Code Vulnerability Mining Experiment Analysis

4.1 Vulnerability Mining Method

As time changes, people have made many attempts on vulnerability mining methods. Through questionnaire surveys and consulting a large amount of data, we have collected statistics on the most frequently used vulnerability mining methods in recent years, and converted them into specific values through model calculations. The data is shown in Table 1:

Table1: Vulnerability mining method

	2011	2012	2013	2014	2015	2016	2017	2018	2019
Missing semantics	1.94	2.38	2.13	2.21	2.29	2.34	1.97	2.06	2.2
Code attributes	2.35	1.8	2.46	1.97	2.36	2.03	1.94	2.2	1.96
Program dependency	1.99	2.08	1.98	1.89	2.09	2.49	1.8	2.19	1.94
Iterative Algorithms	2.15	2.45	2.25	2.1	2.24	2.2	2.15	2.46	1.81
Pollution properties	1.93	2.35	1.97	2.45	2.28	2.37	2.17	2.08	1.92
Subject sensitivity	2	2.26	2.33	2.3	1.86	2.35	1.95	2.08	2.12
Graph neural network	2.36	2.13	2.27	2.3	2.42	2.46	2.47	2.63	2.66

ISSN 2706-655X Vol.4, Issue 4: 21-32, DOI: 10.25236/IJFET.2022.040404

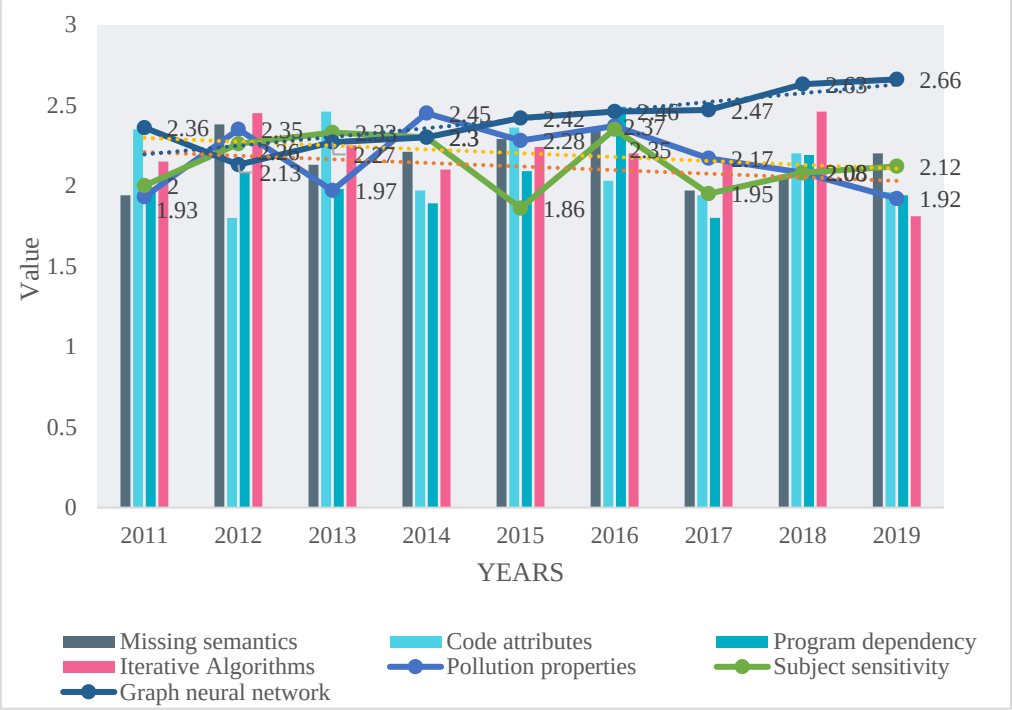

Figure 1: Methods of vulnerability mining

From Figure 1 we can see that in the method of vulnerability mining, the people used in various methods are basically similar, and there is no uniformity of which types of methods. As time changes, the people using different methods are also constantly changing. Changes, among which the method used in this article, the source code vulnerability mining method based on graph neural network shows a trend of first decline and then rise, from 2014 to 2019, from 2.3 to 2.66, and from 2015, it has become people The most commonly used vulnerability mining method. We show the different growth trends from 2014 to 2019, as shown in Figure 2:

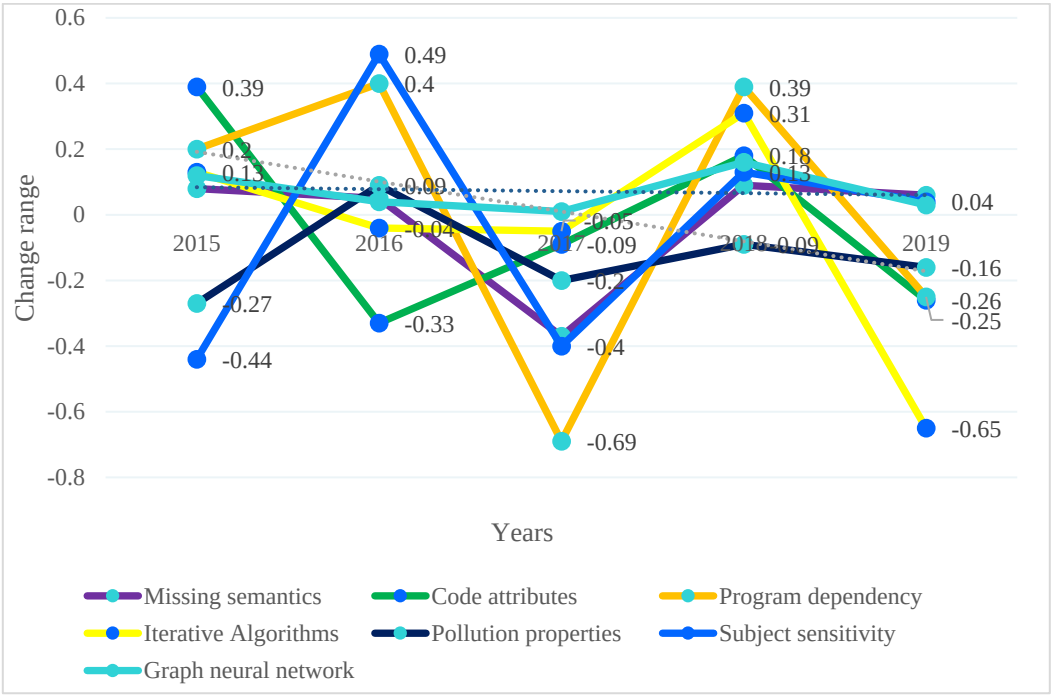

Figure2: Growth trend of mining methods

From Figure 2, we can see that the variation of the vulnerability mining methods used in the past five years is relatively large, but the graph neural network-based source code vulnerability mining method used in this article has been different in the past five years. It shows an upward trend. Among

ISSN 2706-655X Vol.4, Issue 4: 21-32, DOI: 10.25236/IJFET.2022.040404

them, the largest increase was in 2018, with an amplitude of 0.16, which also shows that the method of source code vulnerability mining is increasingly accepted by people.

4.2 Vulnerability Mining Efficiency of Different Methods

In order to study the difference between the efficiency of vulnerability mining by different methods, we have made statistics on the computing time, response time, number of vulnerabilities and mining speed of these types of mining methods. The specific data is shown in Table 2:

Table2: Vulnerability mining efficiency

	Start speed(s)	Running speed(s)	responding speed(s)	Speed of finding vulnerabilities(s)	Number of vulnerabilities found	Average number of discoveries
Missing semantics	19.1	20.8	24.8	19.7	86	15.6
Code attributes	18.9	23.2	23.4	23.5	77	14.7
Program dependency	18.1	18.9	20.5	21.7	82	14.9
Iterative Algorithms	18.8	23.6	21.3	24.5	69	13.5
Pollution properties	22.6	19.2	21.6	18.9	71	13.9
Subject sensitivity	22.1	21.7	19.7	24.4	62	12.7
Graph neural network	18.2	21.3	19.3	22.4	94	16.4

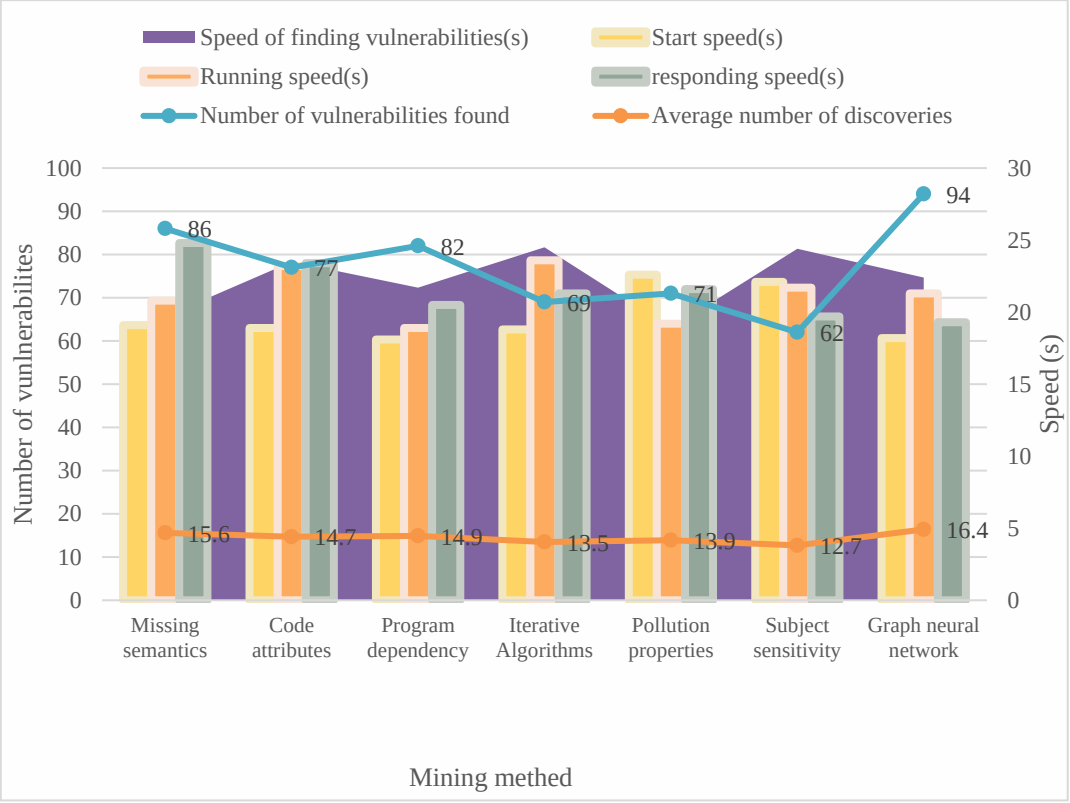

Figure3: Difference in efficiency of mining methods

It can be seen from Figure 3 that these methods have their own advantages in different fields. For example, program dependency mining is the fastest in terms of response speed, which only takes 19 seconds, while other methods generally take more than 20 seconds. On the whole, source code vulnerability mining methods based on graph neural networks are only inferior to other methods in some areas, but comprehensive evaluation of the number and efficiency of mining vulnerabilities can

ISSN 2706-655X Vol.4, Issue 4: 21-32, DOI: 10.25236/IJFET.2022.040404

lead other methods. We use the model to digitize it for easy comparison, as shown in Table 3:

Table3: Scores of different mining methods

	Start speed(s)	Running speed(s)	responding speed(s)	Speed of finding vulnerabilities(s)	Number of vulnerabilities found	Average number of discoveries
Missing semantics	5.16	4.93	4.85	4.9	4.58	5.12
Code attributes	5.37	4.98	4.84	4.92	5.25	5.19
Program dependency	5.2	5.9	5.18	5.62	5.22	5.53
Iterative Algorithms	6.27	6.02	5.95	6.03	6.22	6.21
Pollution properties	6.62	4.72	6.44	6.91	6.72	6.74
Subject sensitivity	6.27	5.21	7.84	7.94	7.62	6.25
Graph neural network	7.87	8.08	8.46	7.91	8.23	7.83

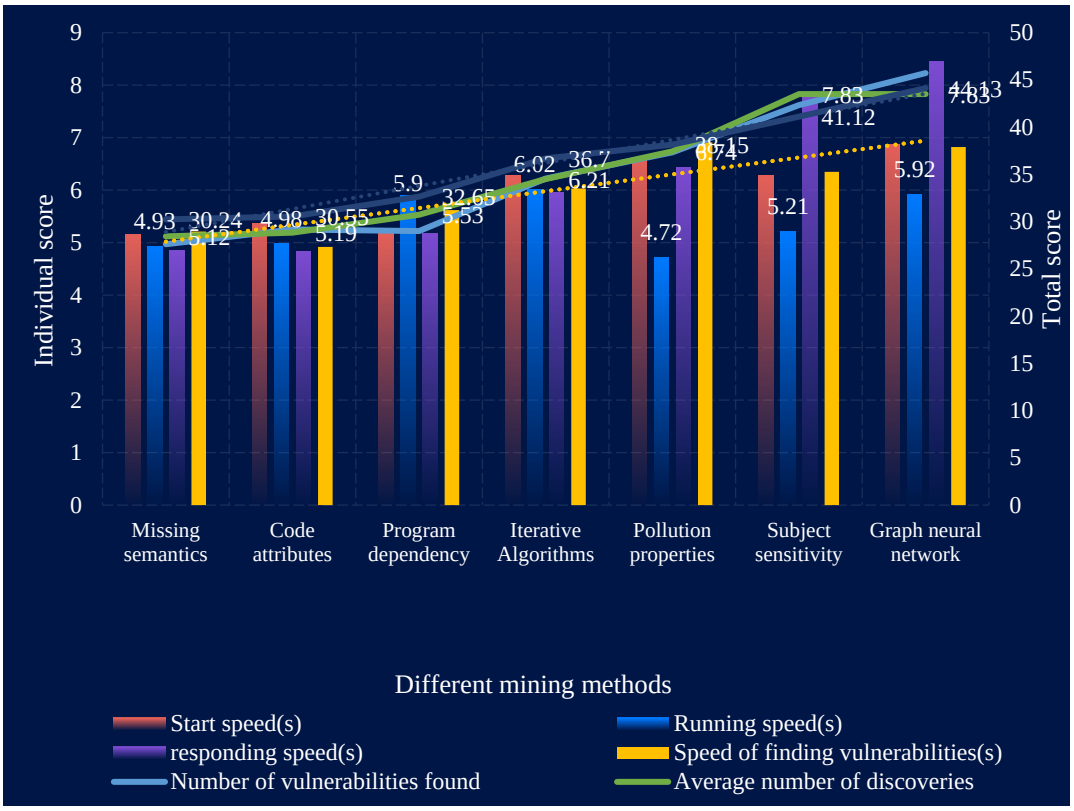

Figure 4: Difference in efficiency of mining methods

From Figure 4, we can clearly see that in terms of the overall score, the source code vulnerability mining method based on graph neural network used in this article leads other mining methods with a total score of 44.13 points. Among them, the semantic mining method The lowest score was 30.24 points. This shows that the use of source code vulnerability mining methods based on graph neural networks can play an important role in vulnerability mining.

4.3 Comparison of Indicators of Vulnerability Mining Methods

For different mining methods, in addition to comparing their mining efficiency, we also need to compare their related indicators, such as machine learning, precision rate, recall rate, false positive rate and false negative rate, etc., especially false positive rate and false negative rate. Rate is extremely important for vulnerability mining, and represents the excellence of the mining method. We collected

ISSN 2706-655X Vol.4, Issue 4: 21-32, DOI: 10.25236/IJFET.2022.040404

these data through the model, as shown in Table 4:

Table4: Scores of different mining methods

	Machine learning	Accuracy	Recall rate	False alarm rate	False negative rate	Comprehensive Evaluation
Missing semantics	2.72	3.58	2.84	0.97%	1.23%	16.9
Code attributes	3.49	2.89	3.15	0.89%	1.15%	18.2
Program dependency	3.56	3.33	2.96	0.93%	0.99%	21.7
Iterative Algorithms	3.8	3.15	4.09	0.78%	0.83%	17.5
Pollution properties	4.69	4.53	4.35	0.72%	0.81%	19.3
Subject sensitivity	5.04	5	4.85	0.83%	0.94%	20.8
Graph neural network	5.16	5.77	5.68	0.22%	0.28%	28.5

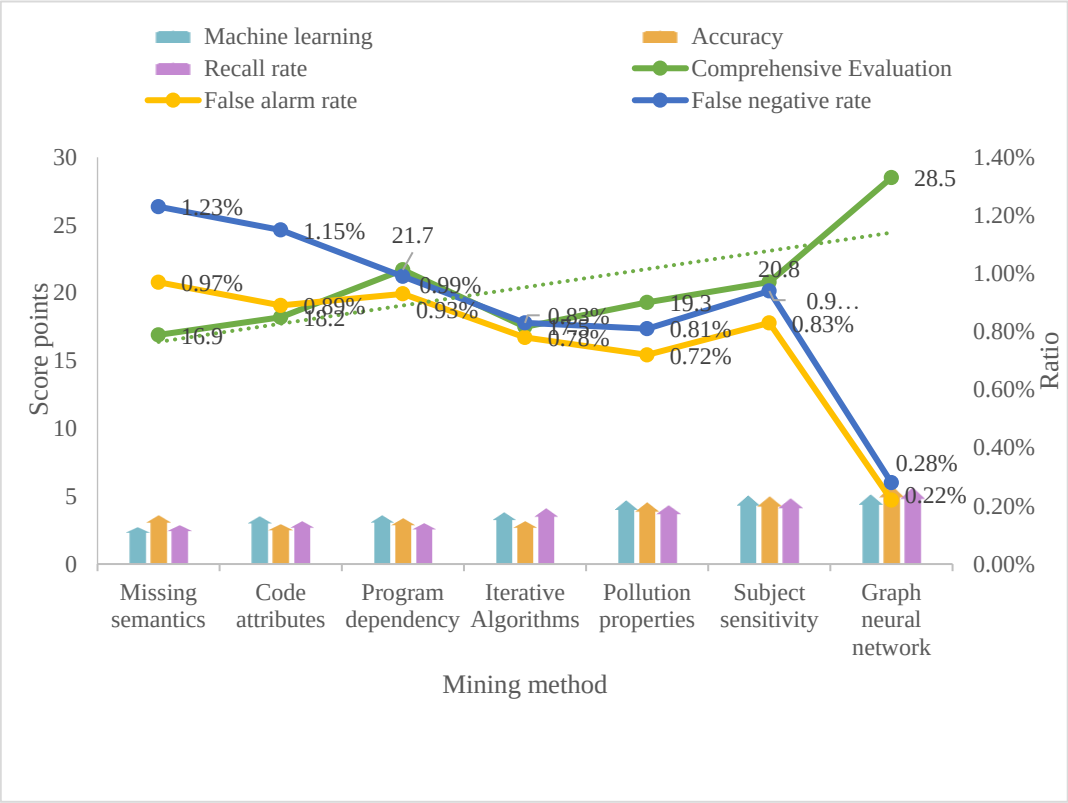

Figure 5: Vulnerability mining indicators

From Figure 5, we can find that among the various indicators of vulnerability mining, the traditional vulnerability mining has the lowest score, and its false positives and underreports are also the most serious. This is because the traditional vulnerability mining methods are based on manual work. In the process, it is inevitable that there will be false negatives and false positives. For the graph neural network system, due to the use of computers and machine learning reasons, it will continue to correct itself, and the false positives and false negatives will occur. The probability of problems is smaller, so the source code vulnerability mining score based on graph neural networks is much higher than other methods. In order to verify the correctness of the results, we used the model for a 24-hour test run, and the results obtained are shown in Table 5:

ISSN 2706-655X Vol.4, Issue 4: 21-32, DOI: 10.25236/IJFET.2022.040404

Table 5: Model test run results

	Machine learning	Accuracy	Recall rate	False alarm rate	False negative rate	Comprehensive Evaluation
Missing semantics	2.32	3.78	2.84	0.827%	0.927%	17.1
Code attributes	3.24	2.91	3.15	0.815%	1.082%	18.9
Program dependency	3.67	3.53	2.96	0.952%	0.86%	22.1
Iterative Algorithms	3.79	3.05	4.09	0.692%	1.13%	16.8
Pollution properties	5.13	4.23	4.35	0.762%	0.795%	17.8
Subject sensitivity	4.74	4.86	4.85	0.793%	0.892%	23.6
Graph neural network	5.66	6.17	6.28	0.235%	0.267%	31.1

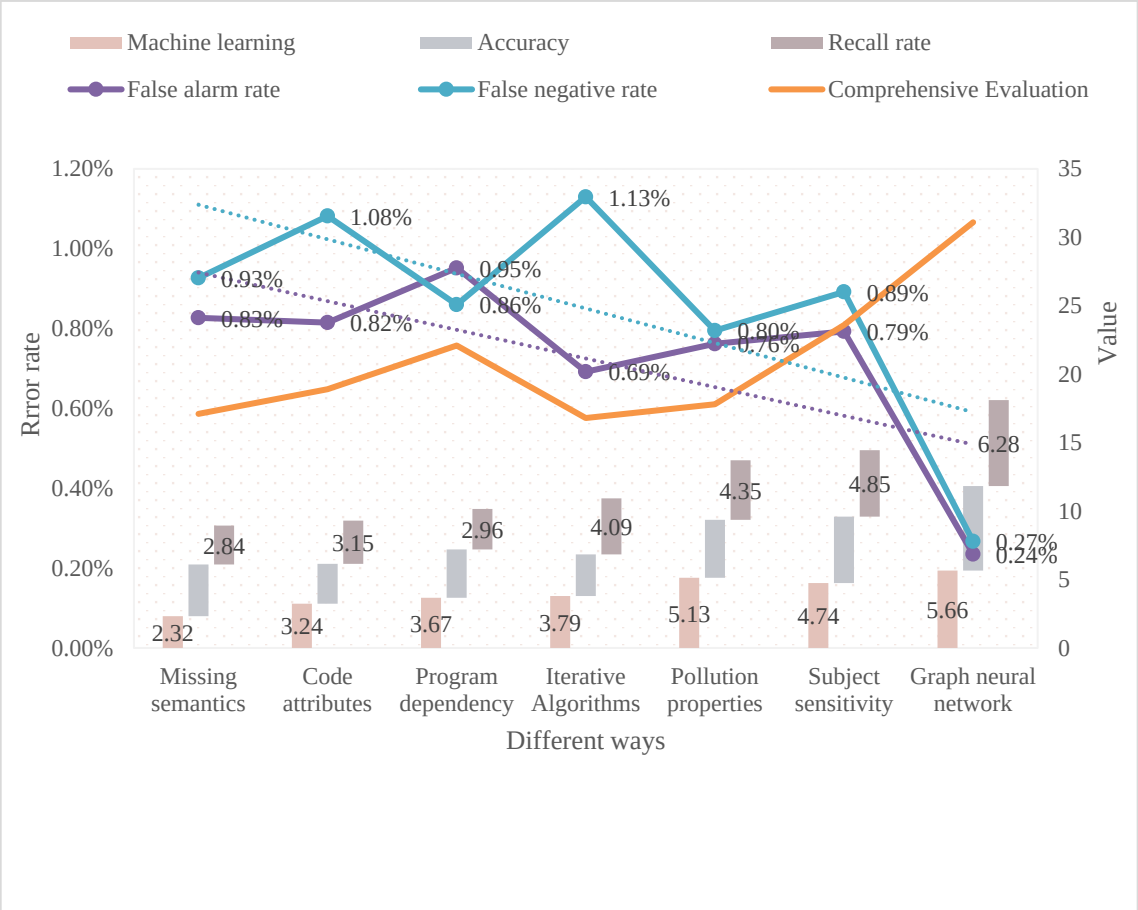

Figure 6: Mining method under trial operation

From Figure 6 we can see that after the model has been tested for one day, the data is statistically relevant. In this statistics, there is a certain deviation between the final test result and the prediction, as shown in Figure 7, but the difference is within the allowable range, which shows that the test results of the model can still be used as the final result, and the source code vulnerability mining method based on graph neural network is feasible.

ISSN 2706-655X Vol.4, Issue 4: 21-32, DOI: 10.25236/IJFET.2022.040404

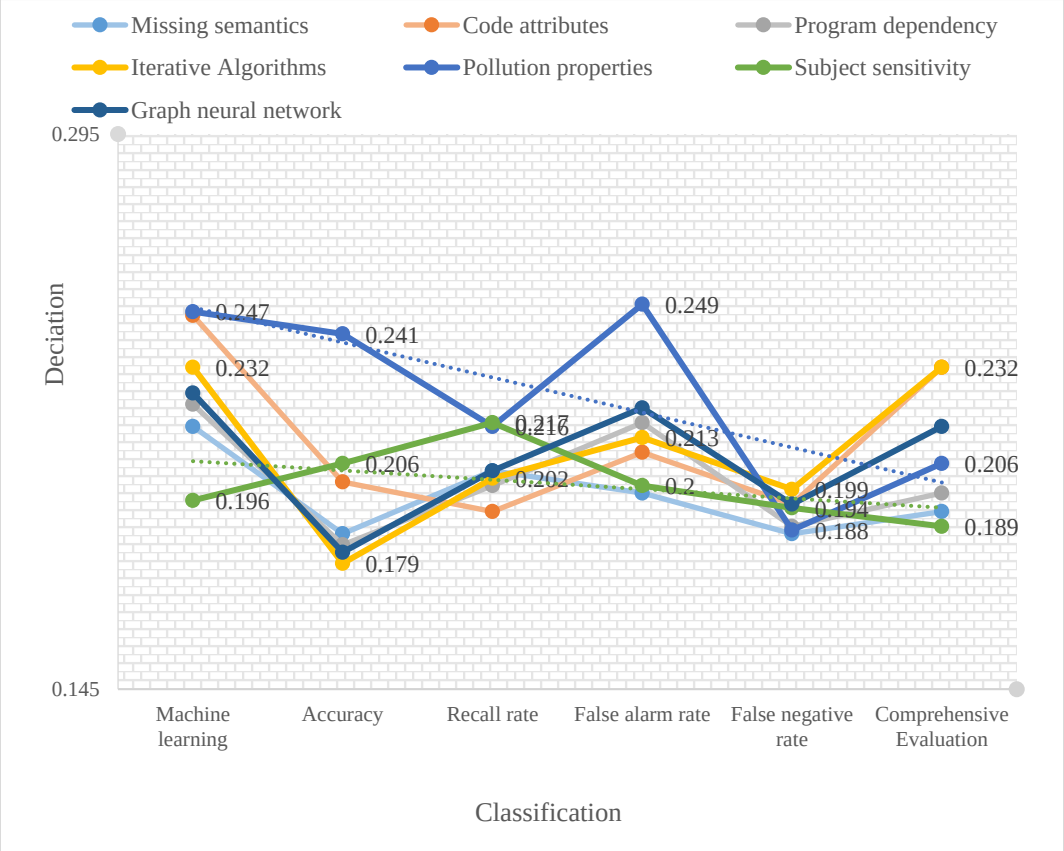

Figure 7: Test result deviation

5.Conclusions

The granularity and the applicability of cross-program testing still need to be improved. In the process of feature analysis, the geometric similarity and texture similarity between nodes are considered at the same time, which effectively eliminates the influence of background or noise and enhances the ability of features to express local information. By training a sub-neural network for each node in the graph structure, the original complex problem is decomposed into multiple local sub-problems, and further parameters are transferred in machine learning according to the graph neural network, which effectively improves the efficiency of algorithm training.

Perform threat correlation analysis and mining on the vulnerabilities that have already erupted, starting from the sentence where the vulnerability occurs, and then looking for information such as identifiers, keywords, key sentences and statements that are strongly related to software vulnerabilities. This can be combined with technologies such as vulnerability information collection and vulnerability code keyword extraction to do the establishment of an automated vulnerability knowledge base.

Source code vulnerabilities mining based on graph neural network, build statistical models from the source code that can reflect the programming thinking logic of developers, and use programming logic models to detect code defects, providing a foundation for intelligent mining of vulnerabilities, automated programming and other cutting-edge technologies Sexual research expounds the concept of programming logic, analyzes the related problems of research programming logic, and clarifies the research content and interrelationship of each part of programming logic research, which can better solve the problem of system security vulnerabilities.

Acknowledgments

This work was supported by Academic support project for top-notch talents in disciplines in universities of Anhui Provincial Department of Education (gxbjZD2021040) and the Natural science research project of the Anhui Education Department (KJ2020A1134).

ISSN 2706-655X Vol.4, Issue 4: 21-32, DOI: 10.25236/IJFET.2022.040404

References

[1] Deng Qigui, Wei Bingui. Research on Vulnerability Mining Technology of Industrial Control System Based on Stain Analysis. Popular Science and Technology, 2019, 021(004):5-7,4.
[2] Wenping Li, Yu Liu, Wei Qiao. An Improved Vulnerability Assessment Model for Floor Water Bursting fr. Mine Water and the Environment, 2017, 37(1):1-9.
[3] Ghaffarian S M, Shahriari H R. Software Vulnerability Analysis and Discovery Using Machine-Learning and Data-Mining Techniques. ACM Computing Surveys (CSUR), 2017, 50(4):1-36.
[4] Tiwari A K , Singh P K , De Maio M . Evaluation of aquifer vulnerability in a coal mining of India by using GIS-based DRASTIC model. Arabian Journal of Geosciences, 2016, 000(6):1-15.
[5] Yang Y , Ren X , Zhang S , et al. Incorporating ecological vulnerability assessment into rehabilitation planning for a post-mining area. Environmental Earth Sciences, 2017, 76(6):245.
[6] Li J , Chen J , Huang M , et al. An Integration Testing Framework and Evaluation Metric for Vulnerability Mining Methods. Wireless Communication over ZigBee for Automotive Inclination Measurement. China Communications, 2018, 15(002):190-208.
[7] Wang C , Ren T , Li Q , et al. Network computer security hidden dangers and vulnerability mining technology. IOP Conference Series: Materials Science and Engineering, 2020, 750(1):121-155.
[8] Lai Y , Gao H , Liu J . Vulnerability Mining Method for the Modbus TCP Using an Anti-Sample Fuzzer. Sensors, 2020, 20(7):20-40.
[9] Ren T , Wang X , Li Q , et al. Vulnerability Mining Technology Based on Genetic Algorithm and Model Constraint. IOP Conference Series: Materials ence and Engineering, 2020, 750(1):122-128 .
[10] PawlakR ,Monperrus M , Petitprez N , et al. Spoon: A Library for Implementing Analyses and Transformations of Java Source Code. Software Practice and Experience, 2016, 46(9):1155-1179.
[11] Vamsi K G . Prediction of Source Code Quality Using Cyclomatic Complexity and Machine Learning. International Journal of Advanced Trends in Computer Science and Engineering, 2020, 9(4):4409-4413.
[12] Lee G , Yu J , Kim I , et al. Implementation of Software Source Code Obfuscation Tool for Weapon System Anti-Tampering. Journal of KIISE, 2019, 46(5):448-456.
[13] ManahiM, Sulaiman S, Bakar N S A A , et al. Source Code Plagarism Detection Approaches: A Systematic Literature Review. Journal of Advanced Research in Dynamical and Control Systems, 2020, 12(4-Special Issue):1575-1587.
[14] Jang Y S . Source Code Instrumentation Technique for Buffer Overflow Vulnerability Detection. The Journal of Korean Institute of Information Technology, 2019, 17(9):133-144.
[15] Wang W , Li G , Shen S , et al. Modular Tree Network for Source Code Representation Learning. ACM Transactions on Software Engineering and Methodology, 2020, 29(4):1-23.
[16] FranclintonR, Karnalim O . A Language-Independent Library for Observing Source Code Plagiarism. Journal of Information Systems Engineering and Business Intelligence, 2019, 5(2):110-119.
[17] Li Hang, Dong Wei, Zhu Guangyu. Research on Industrial Control Protocol Vulnerability Mining Technology Based on Fuzzing Test. Application of Electronic Technology, 2016, 42(7):79-82.
[18] Sha Letian, Xiao Fu, Yang Hongke, et al. IaaS layer vulnerability mining method based on adaptive fuzzing. Journal of Software, 2018, v.29(05):1303-1317.
[19] Sha Letian, Xiao Fu, Yang Hongke, et al. IaaS layer vulnerability mining method based on adaptive fuzzing. Journal of Software, 2018, 029(005):1303-1317.
[20] Zhou Min, Zhou Anmin, Liu Liang, et al. Mining denial of service vulnerability in Android applications automatically. Computer Applications, 2017, 037(011):3288-3293,3329.
[21] Liu Jinhui, Ge Lina, Zhang Jing, et al. Research on XSS Vulnerability Mining Technology Based on Fuzzy Testing. Network New Media Technology, 2016, v.5;No.25(01):13-20.
[22] QiuZhiqing, HuanFei. Vulnerability mining and detection tool based on web crawler and Fuzzing. Microcomputer Applications, 2016, v.32;No.275(03):73-76.
[23] Li Jiali, Chen Yongle, Li Zhi, et al. RTSP protocol vulnerability mining based on protocol state diagram traversal. Computer Science, 2018, 45(09):178-183.
[24] Fu Menglin, Wu Lifa, Hong Zheng, et al. Research on mining technology of smart contract security vulnerabilities. Journal of Computer Applications, 2019, 039(007):1959-1966.