Scholar Publishing Group
International Journal of Multimedia Computing
https://doi.org/10.38007/IJMC.2025.060108
ISSN 2789-7168 Vol. 6, Issue 1: 81-92

Development and Function Extension Analysis of Computer Platform Based on JSP and JavaScript

Jiaji Zhou

College of Art and Design, Changchun Humanities and Sciences College, Changchun 130000, Jilin, China

Keywords: JavaScript platform development; Web application security detection; Vulnerability detection; Malicious code detection; Mixed attribute graph model

Abstract: With the development of big data, cloud computing, and artificial intelligence technologies, web applications have become important carriers of human-computer interaction. However, their security threats continue to escalate, with attack types expanding from traditional SQL injection and cross site scripting (XSS) to new types of file free attacks, deserialization vulnerability exploitation, etc. The security of software supply chains is out of control, attack methods are diversified, and attack targets are shifting towards data trafficking and malicious code implantation, posing serious challenges to the security of individuals, enterprises, and critical infrastructure. This article is based on the development and functional extension analysis of computer platforms using JSP and JavaScript, focusing on two main threats: web vulnerabilities and web malicious code. Combining traditional program analysis and machine learning techniques, innovative detection methods are proposed: for traditional web vulnerabilities (such as XSS), a gray box fuzzy testing method based on reinforcement learning is proposed. Through static analysis to locate injection points, structured partitioning of attack payloads, and optimization of reinforcement learning models, the experimental detection rate reaches 93.75%, which is better than mainstream scanners; To address the new Java deserialization vulnerability, a hybrid attribute graph model was constructed and the exploitation chain was searched, achieving a detection rate of 80%; Targeting traditional file based malicious code (JavaScript), combining semantic analysis with BiLSTM model, generating semantic slices through program dependency graph and optimizing input with optimal accuracy; A hybrid method of dynamic monitoring, security sensitivity, and lightweight taint analysis was used to detect malicious code on the Java file free web shell, achieving a detection rate of 81.33%. The research aims to improve detection efficiency and accuracy, and respond to the evolving threats of network attacks. Future work will focus on the transferability of detection models, improvement of program analysis accuracy, establishment of standard test sets, and research on code obfuscation mitigation techniques.

1. Introduction

With the development of technologies such as big data, cloud computing, and artificial intelligence, web applications have penetrated into various fields of social life, from information websites to complex information systems, providing users with convenient services but also becoming the main target of attackers. The number and types of attacks targeting web applications are rapidly increasing, from traditional SQL injection and XSS to file free attacks, deserialization vulnerability exploitation, etc., posing serious challenges to personal privacy, enterprise data, and national critical information infrastructure security. The current web application security detection faces three major challenges: firstly, the extensive use of open source software makes it difficult to control software supply chain security, and vulnerabilities and malicious code are easily migrated along the supply chain; Secondly, the attack methods are constantly updated. Traditional attacks (such as SQL injection and XSS) have not been fully defended, and new types of attacks (such as file free attacks and ReDoS/ROP attacks) have intensified the pressure of protection; Thirdly, the target of attacks has shifted from disrupting availability to data trafficking and malicious code extortion, with an increase in hacker attacks driven by profit.This article focuses on the development and functional extension of computer platforms based on JSP and JavaScript, focusing on web vulnerability detection and malicious code detection. Combining traditional program analysis with new machine learning technologies, a series of innovative methods are proposed: for traditional web vulnerabilities (such as cross site scripting vulnerabilities), a gray box fuzzy testing method based on reinforcement learning is proposed. Through static analysis to identify injection points and structured partitioning of attack payloads, reinforcement learning is used to generate/mutate payloads. The experimental detection rate reaches 93.75%, which is better than other well-known scanners; Build a hybrid attribute graph model for new web vulnerabilities (such as Java deserialization vulnerabilities), generate/merge graphs and search for exploitation chain detection, with an experimental detection rate of 80%; A detection method combining semantic analysis and BiLSTM is proposed for traditional file based web malicious code (such as JavaScript). The method generates semantic slices through program dependency graphs and converts them into model inputs, with higher accuracy than other machine learning models and tools; Design a hybrid method of dynamic monitoring and static feature analysis for file free web malicious code (such as Java file free web shell), using security sensitive methods screening, probe insertion, and lightweight taint analysis to determine whether suspicious classes are web shells. The experimental detection rate is 81.33%.In summary, the innovative method proposed in this article aims to improve the accuracy and efficiency of web application security detection, respond to evolving network attack threats, and provide support for ensuring web application security.

2. Correlation theory

In the field of computer platform development and feature extension based on JSP and JavaScript, multiple studies have been conducted from the dimensions of security, quality improvement, and functional innovation, forming a multidimensional technical optimization framework. In terms of security detection, JSHint effectively enhances its ability to detect malicious code by revealing API usage and analyzing malicious JavaScript behavior; Dora and others conducted a security centered evaluation of the web application code generated by LLM, revealing its hidden risks (such as potential vulnerabilities), but the sample size may affect the comprehensiveness of the conclusions. In terms of program quality optimization, Antal et al. proposed a call metric method based on mixed call graph to enhance the accuracy of bug prediction in JavaScript programs; Francesco's "JavaScript Design Patterns" summarizes design patterns from a methodological perspective, helping to improve code maintainability and extensibility, but its effectiveness depends on the actual application capabilities of developers. At the level of functional extension, Boostlet-js implements a web image processing plugin through JavaScript injection, which extends the image processing capabilities of the web, but may face security or compatibility challenges. In terms of system optimization, the JavaScript engine verification method proposed by the Korean patent aims to reduce detection errors in personal information monitoring systems and lower misjudgment rates through engine optimization, but the technical implementation complexity is relatively high. In addition, MetaPix, as a data centric AI development platform, focuses on efficient management and utilization of unstructured computer vision data, integrating data management and AI development processes to improve processing efficiency. However, its performance depends on the quality and diversity of input data. These studies have promoted platform development and feature extension based on JSP and JavaScript from different dimensions, balancing security, quality, and functionality requirements. However, they also have limitations in technology implementation, data dependency, or scenario adaptation.

3. Research method

3.1. Overview of Program Analysis and Web Security Detection Technologies

In the development and functional extension analysis of computer platforms based on JSP and JavaScript, program analysis technology, web vulnerability detection technology, and web malicious code detection technology are key components to ensure platform security. As a foundation, program analysis technology covers static analysis, dynamic analysis, and machine learning based analysis methods. Static analysis extracts program states through techniques such as data flow, while dynamic analysis records program execution paths and behaviors to obtain features. Machine learning based analysis technology can solve problems such as low path coverage, improve analysis accuracy and efficiency. Web vulnerability detection technology can be divided into static analysis, dynamic analysis, and machine learning based detection according to different analysis methods. Static analysis discovers vulnerabilities through lexical, syntactic, and other analyses during the development phase, while dynamic analysis detects vulnerabilities through runtime analysis during the testing phase. Machine learning detection improves detection accuracy and completeness through data preprocessing, feature extraction, and model selection.

Web malicious code detection technology mainly targets web shell and JavaScript malicious code, with feature extraction covering multiple dimensions such as lexical, syntactic, semantic, statistical, and abstract features. Static analysis detects malicious code by comparing static features with discriminative conditions, while dynamic analysis combines behavior judgment to enhance detection capabilities. Machine learning based detection methods automatically extract high-dimensional features and combine static and dynamic analysis to achieve effective identification of malicious code. These technologies together provide a solid security guarantee for the development and functional extension of computer platforms based on JSP and JavaScript.

3.2. Research on Java deserialization vulnerability detection method

The Java deserialization vulnerability, as a serious threat to web application security, has received much attention in recent years. The Java serialization mechanism can convert objects into byte sequences to optimize inter process communication, improve system performance, or save models (such as machine learning models), while deserialization is its inverse process. However, when dealing with untrusted data, the deserialization mechanism may be maliciously exploited, leading to serious threats such as denial of service attacks, remote code execution, or malicious code implantation, with a huge potential impact range.The severity of this vulnerability was widely recognized after the Fox Glove security team first exploited the Apache Commons Collections deserialization vulnerability to launch a remote code execution attack in 2015. It was listed as one of the top 10 most serious web security risks in 2017 due to its impact on popular web containers such as WebLogic and DB2. According to statistics, the National Vulnerability Database (NVD) recorded a total of 126 related vulnerabilities from 2015 to 2022, and the number has been on the rise in the past four years. Most of them exist in third-party open source software such as fastjson, and may spread to downstream web applications through the software supply chain.The existing detection methods have made some progress: in 2018, Haken proposed the first detection tool, Gadget Inspector, which assists security personnel in detecting vulnerabilities by searching and exploiting chains. However, due to the rough analysis of method calls, there are many false positives; In 2020, Rasheed et al. improved their ability to analyze large programs by combining pointer analysis and fuzz testing; In 2021, Lai et al. proposed a detection method that combines static and dynamic analysis to further optimize detection efficiency. However, existing methods still have limitations in terms of semantic analysis depth and precise mining using chains.Detecting Java deserialization vulnerabilities faces two core challenges: firstly, the lack of appropriate code abstraction representation, such as the inheritance, rewriting, and calling relationships of traditional abstract syntax trees (AST), control flow graphs (CFG), and other non integrated classes, making it difficult to achieve accurate analysis; The second challenge is the difficulty of building a utilization chain, which requires quickly locating potential paths among tens of thousands of classes in the target program and Java Runtime Environment (JRE).In response to the above challenges, this article proposes a hybrid attribute graph model that integrates features such as class inheritance and method invocation related to deserialization. By using a bidirectional construction algorithm to merge and search the hybrid attribute graph, it effectively detects Java deserialization vulnerabilities. This method aims to improve the accuracy and efficiency of detection, providing technical support for addressing such high-risk vulnerabilities.

3.3. Analysis of Cross site Script Vulnerability Principle

Cross site scripting vulnerability is a common security threat in web applications, which essentially involves attackers injecting malicious code into trusted websites. When a user visits the website, the embedded malicious code automatically executes (difficult for the user to detect), thereby stealing sensitive information (such as user cookies) or spreading malicious content. According to the triggering method of malicious code, XSS vulnerabilities can be divided into three categories: reflective XSS induces users to click by constructing links containing malicious scripts, and the browser triggers the attack after executing the script; Storage based XSS stores malicious data on the server side (such as a database), and when other users browse related content, the stored malicious data is rendered onto the page and executed; DOM type XSS utilizes the dynamic modification feature of the Document Object Model (DOM), allowing malicious code to modify page content or style on the client side and execute it. Reinforcement learning is a machine learning method that studies agents maximizing long-term rewards through interaction in complex environments. Its core consists of agents and the environment: when the environment is in a certain state, the agent selects an action based on the current state, and after the action is executed, the environment transitions to the next state and returns a reward. The goal of an intelligent agent is to learn the optimal strategy through continuous interaction, which can be formalized as a Markov Decision Process (MDP) described by a quadruple of state space, action space, reward function, and state transition function. The decision effect is quantified through the state value function and state action value function, and the optimal strategy is ultimately solved to maximize cumulative rewards.

4. Results and discussion

4.1. The evolution and challenges of Java deserialization vulnerability detection technology

The Java deserialization vulnerability, as a serious threat to web application security, has received widespread attention in recent years. The serialization mechanism of Java can convert objects into byte sequences to optimize inter process communication, improve system performance, or save models (such as machine learning models), while deserialization is its inverse process. However, when handling untrusted data, the deserialization mechanism may be maliciously exploited, leading to serious threats such as denial of service attacks, remote code execution, or malicious code implantation. The potential impact of such vulnerabilities is enormous. As early as 2006, researchers raised concerns about potential Java deserialization vulnerabilities, but it wasn't until 2015 when the Fox Glove security team first exploited the Apache Commons Collections deserialization vulnerability to launch a remote code execution attack that its severity was widely recognized. This vulnerability was listed as one of the most serious web security risks in the 2017 OWASP Top 10 due to its impact on popular web containers such as WebLogic and DB2. According to statistics, the National Vulnerability Database (NVD) recorded a total of 126 Java deserialization vulnerabilities from 2015 to 2022, with the number showing an upward trend in the past four years, and most of them exist in third-party open source software (such as fastjson), which may spread to downstream web applications through the software supply chain. The existing detection methods have made some progress. In 2018, Haken proposed the first detection tool, Gadget Inspector, which assists security personnel in detecting vulnerabilities by searching and exploiting chains. However, due to the rough method call analysis, there are many false positives; In 2020, Rasheed et al. improved their ability to analyze large programs by combining pointer analysis and fuzz testing; In the same year, Du Xiaoyu and others proposed a detection method based on bytecode search, using taint analysis and symbolic execution techniques to mine call chains; In 2021, Lai et al. proposed a detection method that combines static and dynamic analysis; In 2022, Wu Yongxing et al. proposed a chain mining method based on mixed information flow analysis. However, existing methods still have limitations in terms of semantic analysis depth and precise mining using chains. There are two major challenges in detecting Java deserialization vulnerabilities: firstly, the lack of appropriate code abstraction representation. Traditional abstract syntax trees, control flow graphs, and other non integrated class inheritance, rewriting, and call relationships make it difficult to achieve accurate analysis; The second challenge is the difficulty of building a utilization chain, which requires quickly locating potential paths among tens of thousands of classes in the target program and Java Runtime Environment (JRE). In response to the above challenges, this article proposes a hybrid attribute graph model that integrates features such as class inheritance and method invocation related to deserialization. By using a bidirectional construction algorithm to merge and search the hybrid attribute graph, it effectively detects Java deserialization vulnerabilities.

4.2. JavaScript Malicious Code Detection Methods

JavaScript, as a widely used lightweight scripting language in web development, has become an important target for malicious code attacks due to its dynamic nature. It can launch various threats such as cross site request forgery, driver download attacks, and distributed denial of service attacks. Traditional detection methods face challenges due to attackers using techniques such as random obfuscation, encoding obfuscation, data obfuscation, and logic obfuscation - these techniques hide malicious intent by modifying variable names, inserting irrelevant logic, recombining strings, etc., leading to the failure of tools based on keywords or static analysis. To this end, researchers propose a deep learning based detection method that treats programs as special natural language and utilizes models such as LSTM, Graph Convolutional Networks (GCN), and attention mechanisms to capture code semantics; However, the flexibility of program syntax is high, and the dependency relationships between statements do not depend on distance. Simple traversal of abstract syntax trees (AST) or control flow graphs (CFG) is difficult to effectively extract semantics. This article proposes a program slicing method based on semantic analysis: combining control flow and data flow dependencies to construct a program dependency graph (PDG), restoring code semantics through obfuscation processing (such as JSDetox tool), and then generating slices that preserve rich semantics based on key functions (string operations, encoding conversion, URL redirection, and special behaviors, etc.), and converting them into vector input bidirectional long short-term memory networks (BiLSTM). This model captures long-term dependencies between sentences through forward and backward propagation. Experiments have shown that its detection performance is superior to other machine learning models (such as Naive Bayes, SVM, Random Forest) and traditional tools (such as JaSt, ClamAV). The accuracy on the de obfuscation dataset is 97.71%, and the F1 score is 98.29%, effectively addressing the interference of obfuscation techniques on detection

4.3. Comparative analysis of evaluation effects

In the development and functional extension analysis of computer platforms based on JSP and JavaScript, in order to systematically evaluate the effectiveness of JShellDetector, this article records in detail the results of two key steps: suspicious class dynamic recognition and webshell detection. The number of suspicious class recognitions is used to evaluate the performance of the method probe and determine whether it can accurately capture the triggering of sensitive features by the webshell; The number of detected webshells is used to evaluate the detection performance of webshells based on taint analysis, and is also the evaluation criterion for the overall performance of JShellDetector. Table 1 shows the specific performance of JShellDetector on various test cases, where step 1 identifies suspicious classes and step 2 detects webshells. As shown in Table 1

Table 1. Fileless webshell detection results of JShellDetector

ShellDetector Classification Type	Step 1 (Identify Suspicious Classes)	Step 2 (Webshell Detection)
Component Memory Shell	60	54(42 successful, 12 failed)
Web y/t Listener-based Type	60	55 (41 successful, 14 failed)
Servlet Type	60	56 (48 successful, 8 failed)
Interceptor Type Based on Spring Controller Framework	60	51 (35 successful, 16 failed)
Total	300	244(162 successful, 78 failed)

The experimental results showed that JShellDetector successfully captured all suspicious classes in 300 test cases, with a detection rate of 100%, proving the effectiveness of the method probe based screening method and significantly reducing the detection range. In the webshell detection phase, a total of 244 test cases were detected, with a detection rate of 81.33%. To explore the reasons for detection failure, this article conducted manual analysis on failed cases and found that the main reasons for failure were string encryption and dynamic code generation technology. A test case fragment showing detection failure due to string encryption was presented, in which the attacker encoded the code fragment implementing malicious logic in Base64 and stored it in a string. The webshell was remotely connected and re decoded the string into a Java class, and the malicious command was executed in the parsed Java class, thus bypassing detection. In summary, JShellDetector successfully captured all test cases that triggered security sensitive methods, extracted trigger classes from the JVM as bytecode files, and detected 81.33% of malicious behavior through taint analysis, demonstrating its effectiveness in detecting Java fileless webshells. At present, research on Java fileless webshells is still in its infancy, and Copagent and CacheShell are two open-source detection tools. Copagent filters suspicious classes through package name, class name, interface name, and comments, using three built-in rules; Memory Shell enhances its protection against file free webshell attacks and method call analysis based on this foundation.

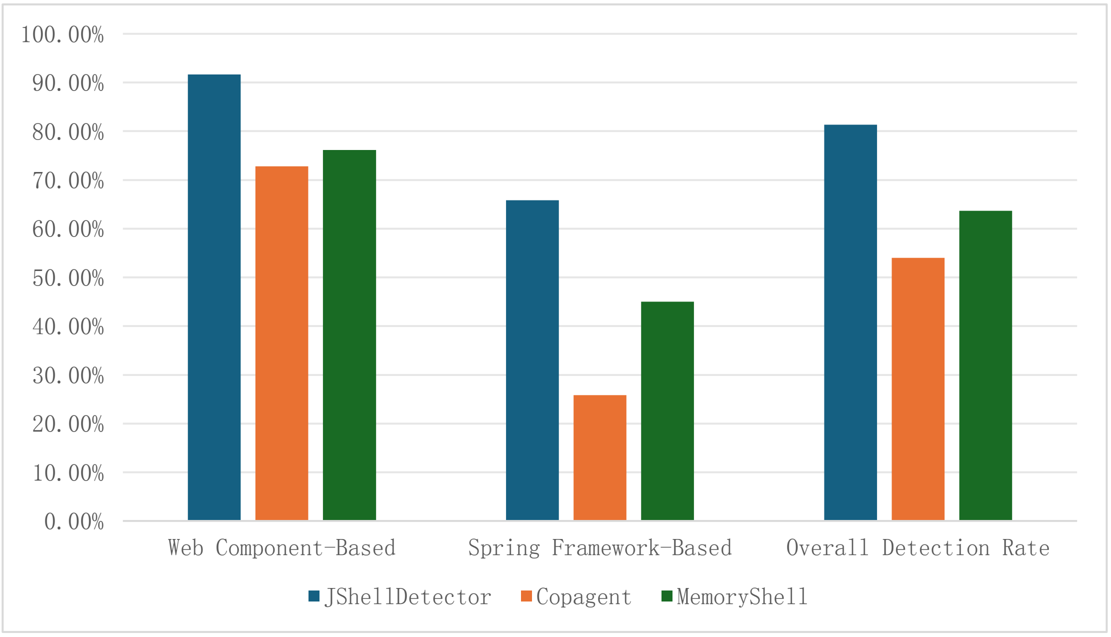

Figure 1. Detection Rates of Different Types of Java Fileless Webshells

However, as shown in Table 2, the performance of both Memory Shell and Copagent is inferior to JShellDetector, with detection rates of 63.67% and 54%, respectively. JShellDetector has a detection rate of 81.33%, especially in Spring based webshells, where its detection rates are 40% and 20.83% higher, respectively. Copagent is also unable to detect interceptor type webshells. The superiority of JShellDetector lies in its behavior based detection method, which, unlike blacklist based mechanisms, can detect malicious behavior that does not inherit dangerous parent classes or implement specific interfaces. In terms of system resource consumption, JShellDetector adopts offline analysis and deployment methods to detect and unload analysis tasks. It communicates through message queues, and the additional memory consumption does not exceed 5% of the web server capacity. This study proposes a lightweight, hybrid analysis based detection method to address the issue of maintaining the persistence of web attacks using Java fileless webshells. By monitoring security sensitive method triggers to narrow down the detection range, the trigger class is extracted and converted into a bytecode file, and then lightweight taint analysis is applied to determine whether the suspicious class is a webshell. The experimental results show that this method detected 81.33% of malicious samples, which is about 18% higher than existing tools.

5. Conclusion

In recent years, with the development of technologies such as cloud computing, big data, and artificial intelligence, web application functions have become increasingly complex. However, security risks have intensified, especially with the prosperity of open source communities leading to vulnerabilities and malicious code spreading through software supply chains, causing serious losses. In response to the expansion of web application attack types from traditional SQL injection and XSS to file free attacks, deserialization vulnerability exploitation, etc., traditional defenses are insufficient and new types of attacks exacerbate the pressure of protection. The attack targets have shifted to data trafficking, extortion, etc. This article focuses on web vulnerability and malicious code detection, combining traditional program analysis and machine learning methods to propose four innovative methods: the grey box fuzzy testing method based on reinforcement learning is applied to cross site script vulnerability detection, identifying potential injection points and structured partition attack payloads through static analysis, and using reinforcement learning models to generate and mutate attack payloads. The experiment shows that the vulnerability detection rate in the test samples reaches 93.75%, significantly better than other well-known vulnerability scanners; A mixed attribute graph model is proposed for detecting Java deserialization vulnerabilities. By constructing a mixed attribute graph with features such as deserialization related class inheritance and method invocation, and searching for exploitation chains on the graph, the experimental detection rate reaches 80%; A method based on semantic analysis and BiLSTM model is proposed for traditional file based web malicious code detection. By analyzing the program dependency graph to generate semantic slices and converting the slices into deep learning model inputs, the experimental accuracy is superior to other machine learning models and detection tools; A hybrid method of dynamic monitoring and static feature analysis is designed for detecting malicious code on the fileless web. Through screening security sensitive methods, inserting method probes, and lightweight taint analysis techniques, the suspicious class is determined to be a web shell, with an experimental detection rate of 81.33%. In the future, it is necessary to improve the transferability of detection models, the universality of vulnerability methods (such as unified intermediate representation), enhance the accuracy of program analysis (develop tools for specific problems), establish standard test datasets, and study anti obfuscation techniques to strengthen the robustness of malicious code detection.

References

[1] Chen, X. (2025). Research on the Application of Multilingual Natural Language Processing Technology in Smart Home Systems. Journal of Computer, Signal, and System Research, 2(5), 8-14.

[2] Hui, X. (2025). Research on Improving the Matching Efficiency between Cancer Patients and Clinical Trials Based on Machine Learning Algorithms. Journal of Medicine and Life Sciences, 1(3), 74-80.

[3] Pan Y. Research on Cloud Storage Data Access Control Based on the CP-ABE Algorithm[J]. Pinnacle Academic Press Proceedings Series, 2025, 2: 122-129.

[4] Shen, D. (2025). AI-Driven Clinical Decision Support Optimizes Treatment Accuracy for Mental Illness. Journal of Medicine and Life Sciences, 1(3), 81-87.

[5] Yan J. Analysis and Application of Spark Fast Data Recommendation Algorithm Based on Hadoop Platform[C]//2025 Asia-Europe Conference on Cybersecurity, Internet of Things and Soft Computing (CITSC). IEEE, 2025: 872-876.

[6] Huang J. Digital Technologies Enabling Rural Revitalization: The Practice of AI and BIM in the Adaptive Reuse of Historic Buildings[J]. International Journal of Architectural Engineering and Design, 2025, 2(1): 1-8.

[7] Jiang Y. Research on the Optimization of Digital Object System by Integrating Metadata Standard and Machine Learning Algorithm[J]. Procedia Computer Science, 2025, 262: 849-858.

[8] An C. Research on High Frequency Financial Transaction Data Modeling and Cloud Computing Implementation Based on SSA-GA-BP Model[J]. Procedia Computer Science, 2025, 262: 859-867.

[9] Zhang M. Design of Object Segmentation and Feature Extraction Based On Deep Learning for AFM Image Processing and Analysis System[J]. Procedia Computer Science, 2025, 262: 982-991.

[10] Lai L. Research and Design of Data Security Risk Assessment Model Based on Fusion of Deep Learning and Analytic Hierarchy Process (AHP)[J]. Procedia Computer Science, 2025, 262: 747-756.

[11] Cai Y. Design and Implementation of a Cross Platform i0s Application Development Framework Based on YAI Configuration Files[J]. Procedia Computer Science, 2025, 262: 939-947.

[12] Wei X. Research on Preprocessing Techniques for Software Defect Prediction Dataset Based on Hybrid Category Balance and Synthetic Sampling Algorithm[J]. Procedia Computer Science, 2025, 262: 840-848.

[13] Wang C. Research on Modeling and Forecasting High-Frequency Financial Data Based on Histogram Time Series[J]. Procedia Computer Science, 2025, 262: 894-900.

[14] Pan H. Design and Implementation of a Cloud Computing Privacy-Preserving Machine Learning Model for Multi-Key Fully Homomorphic Encryption[J]. Procedia Computer Science, 2025, 262: 887-893.

[15] Zhang Y. Research on the Application and Optimization of Multi Dimensional Data Model Based on Kylin in Enterprise Technology Management[C]//International Conference on Innovative Computing. Springer, Singapore, 2025: 234-241.

[16] Hao, Linfeng. "Research on Automatic Driving Road Object Detection Algorithm Integrating Multi Scale Detection and Boundary Box Regression Optimization." In 2025 4th International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE), pp. 1-6. IEEE, 2025.

[17] Tu, Xinran. "Feature Selection and Classification of Electronic Product Fault Short Text by Integrating TF-IDF and Wor D2vec." In 2025 4th International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE), pp. 1-6. IEEE, 2025.

[18] Jiang, Yixian. "Research on Random Sampling Data Diffusion Technique in the Construction of Digital Object System Test Dataset." In 2025 4th International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE), pp. 1-6. IEEE, 2025.

[19] Fu, Yilin. "Design and Empirical Analysis of Financial Quantitative Trading Model based on VMD-DCNN-SGRU Architecture and Integrated System." In 2025 4th International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE), pp. 1-7. IEEE, 2025.

[20] Zhou, Yixin. "Design and Implementation of Online Log Anomaly Detection Model based on Text CM and Hierarchical Attention Mechanism." In 2025 4th International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE), pp. 1-6. IEEE, 2025.

[21] Yuan S. Design and Optimization of Network Security Situation Awareness Algorithm for Generative Adversarial Networks Targeting Attack Data and Traffic [C]//2025 4th International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE). IEEE, 2025: 1-6.

[22] *Guo Y. Research on Investment Bank Risk Monitoring and Early Warning Model Combining Factor Analysis and Artificial Neural Network[J]. Procedia Computer Science, 2025, 262: 878-886.*

[23] *Cui N. Research and Application of Traffic Simulation Optimization Algorithm Based on Improved Road Network Topology Structure[C]//The International Conference on Cyber Security Intelligence and Analytics. Springer, Cham, 2025: 156-163.*

[24] *Yang D, Liu X. Collaborative Algorithm for User Trust and Data Security Based on Blockchain and Machine Learning[J]. Procedia Computer Science, 2025, 262: 757-765.*

[25] *Zhang X. Optimization and Implementation of Time Series Dimensionality Reduction Anti-fraud Model Integrating PCA and LSTM under the Federated Learning Framework[J]. Procedia Computer Science, 2025, 262: 992-1001.*

[26] *Huang, Jiangnan. "Online Platform user Behavior Prediction and Decision Optimization based on Deep Reinforcement Learning." In 2025 4th International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE), pp. 1-6. IEEE, 2025.*

[27] *Chen, H., Yang, Y., & Shao, C. (2021). Multi-task learning for data-efficient spatiotemporal modeling of tool surface progression in ultrasonic metal welding. Journal of Manufacturing Systems, 58, 306-315.*

[28] *Chen, H., Wang, Z., & Han, A. (2024). Guiding Ultrasound Breast Tumor Classification with Human-Specified Regions of Interest: A Differentiable Class Activation Map Approach. In 2024 IEEE Ultrasonics, Ferroelectrics, and Frequency Control Joint Symposium (UFFC-JS) (pp. 1-4). IEEE.*

[29] *Varatharajah, Y., Chen, H., Trotter, A., & Iyer, R. K. (2020). A Dynamic Human-in-the-loop Recommender System for Evidence-based Clinical Staging of COVID-19. In HealthRecSys@ RecSys (pp. 21-22).*

[30] *Chen, H., Zuo, J., Zhu, Y., Kabir, M. R., & Han, A. (2024). Polar-Space Frequency-Domain Filtering for Improved Pulse-echo Speed of Sound Imaging with Convex Probes. In 2024 IEEE Ultrasonics, Ferroelectrics, and Frequency Control Joint Symposium (UFFC-JS) (pp. 1-4). IEEE.*

[31] *Wei Z. Construction of Supply Chain Finance Game Model Based on Blockchain Technology and Nash Equilibrium Analysis[J]. Procedia Computer Science, 2025, 262: 901-908.*

[32] *Yang, D., & Liu, X. (2025). Research on Large-Scale Data Processing and Dynamic Content Optimization Algorithm Based On Reinforcement Learning. Procedia Computer Science, 261, 458-466.*

[33] *Ma, Zhuoer. "Research and Development of Financial Contract Text Information Extraction System based on M-BiLSTM and Our-M Models." In 2025 4th International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE), pp. 1-7. IEEE, 2025.*

[34] *Chen, H., Zhu, Y., Zuo, J., Kabir, M. R., & Han, A. (2024). TranSpeed: Transformer-based Generative Adversarial Network for Speed-of-sound Reconstruction in Pulse-echo Mode. In 2024 IEEE Ultrasonics, Ferroelectrics, and Frequency Control Joint Symposium (UFFC-JS) (pp. 1-4). IEEE.*

[35] *Chen A. Research on Intelligent Code Search Technology Based on Deep Learning[J]. Pinnacle Academic Press Proceedings Series, 2025, 2: 137-143.*

[36] *Pan Y. Research on the Design of a Real-Time E-Commerce Recommendation System Based on Spark in the Context of Big Data[C]//2025 IEEE International*

Conference on Electronics, Energy Systems and Power Engineering (EESPE). IEEE, 2025: 1028-1033.

[37] Xiu L. Research on the Design of Modern Distance Education System Based on Agent Technology[J]. Pinnacle Academic Press Proceedings Series, 2025, 2: 160-169.

[38] Yan J. Research on Application of Big Data Mining and Analysis in Image Processing[J]. Pinnacle Academic Press Proceedings Series, 2025, 2: 130-136.

[39] Xiu L. Analyses of Online Learning Behaviour Based on Linear Regression Algorithm[C]//2025 IEEE International Conference on Electronics, Energy Systems and Power Engineering (EESPE). IEEE, 2025: 1333-1338.

[40] Ren B. Research Progress of Content Generation Model Based on EEG Signals[J]. Journal of Computer, Signal, and System Research, 2025, 2(4): 97-103.

[41] Liu Y. The Impact of Financial Data Automation on the Improvement of Internal Control Quality in Enterprises[J]. European Journal of Business, Economics & Management, 2025, 1(2): 25-31.

[42] Hua X. Optimizing Game Conversion Rates and Market Response Strategies Based on Data Analysis[J]. European Journal of AI, Computing & Informatics, 2025, 1(2): 37-43.

[43] Zhou Y. Research on the Innovative Application of Fintech and AI in Energy Investment[J]. European Journal of Business, Economics & Management, 2025, 1(2): 76-82.

[44] Huang J. Resource Demand Prediction and Optimization Based on Time Series Analysis in Cloud Computing Platform[J]. Journal of Computer, Signal, and System Research, 2025, 2(5): 1-7.

[45] Sheng C. Research on AI-Driven Financial Audit Efficiency Improvement and Financial Report Accuracy[J]. European Journal of Business, Economics & Management, 2025, 1(2): 55-61.

[46] Zhang Q. Research on AI-Driven Advertising Optimization and Automated Decision System[J]. European Journal of Business, Economics & Management, 2025, 1(2): 62-68.

[47] Xu D. Design and Implementation of AI-Based Multi-Modal Video Content Processing[J]. European Journal of AI, Computing & Informatics, 2025, 1(2): 44-50.

[48] Li W. Audit Automation Process and Realization Path Analysis Based on Financial Technology[J]. European Journal of Business, Economics & Management, 2025, 1(2): 69-75.

[49] Liu X. The Role of Generative AI in the Evolution of Digital Advertising Products[J]. Journal of Media, Journalism & Communication Studies, 2025, 1(1): 48-55.

[50] Liu F. Research on Supply Chain Integration and Cost Optimization Strategies for Cross-Border E-Commerce Platforms[J]. European Journal of Business, Economics & Management, 2025, 1(2): 83-89.