Exploit-Aware Security Framework for Real-Time SQL Injection Defense in Login Systems

Pallavi Gudimilla[1*], Manugonda Kalyani[2], Koppula Bhargavi Sree[2], Manchala Sandeep[2], Palangi Ajay Winsent[2]

[1]Assistant Professor, [2]UG Student, [1,2]Department of Computer Science and Engineering (Data Science)

[1,2]Vaagdevi College of Engineering (UGC-Autonomous), Bollikunta, Warangal, 506005, Telangana

*Correspondence: Pallavi Guimilla (pallavi.gudimilla@gmail.com)

To Cite this Article

Pallavi Gudimilla, Manugonda Kalyani, Koppula Bhargavi Sree, Manchala Sandeep, Palangi Ajay Winsent, "Exploit-Aware Security Framework for Real-Time SQL Injection Defense in Login Systems", Journal of Science Engineering Technology and Management Science, Vol. 03, Issue 03, March 2026,pp: 211-221, DOI: http://doi.org/10.64771/jsetms.2026.v03.i03.pp211-221

Submitted: 06-02-2026 *Accepted: 13-03-2026* *Published: 20-03-2026*

Abstract

The rapid expansion of web-based applications has intensified the need for robust authentication systems to protect user data from escalating cyber threats, particularly SQL injection (SQLi) attacks. This research focuses on developing a Django-based web application that integrates both vulnerable and secured login mechanisms to demonstrate essential secure coding practices. The core problem addresses insecure authentication implementations where improper input handling allows attackers to manipulate database queries. Many traditional systems prioritize functionality over security, utilizing direct query concatenation and minimal validation, which enables unauthorized access and compromises data integrity. To address these vulnerabilities, the proposed system utilizes the Django framework paired with a MySQL database for user registration and authentication. A dual-module architecture is employed: a vulnerable module designed to illustrate SQLi risks, and a secured module that implements rigorous input validation and pattern-based attack detection to identify suspicious characters in real-time. This structure facilitates a comparative analysis between insecure and robust implementations. The significance of this project lies in its ability to bridge the gap between theoretical security and practical application. By showcasing real-world vulnerabilities alongside their respective protective mechanisms, the system enhances the understanding of secure software development lifecycles (SDLC). Ultimately, this research provides a functional foundation for building resilient authentication frameworks, improving system reliability, and promoting awareness of web security standards in modern software engineering.

Keywords: SQLi attack, secure authentication, cybersecurity awareness, Django web framework.

1. Introduction

SQLi vulnerability detection has become a research hot spot in information security. Currently, the mainstream approach is to use static analysis techniques to detect SQLi vulnerabilities. Static analysis technology refers to analysing web application code without executing it, mainly relying on syntax and semantic features, such as abstract syntax tree (AST), control flow graph (CFG), and call flow graph, matching according to the characteristics of vulnerabilities to discover them [1]. After a long development, static analysis techniques have become increasingly mature and have been used to automate the detection of SQLi vulnerabilities with certain results [2]. However, existing static analysis techniques cannot accurately detect SQLi vulnerabilities in applications that use object-oriented database extension (OODBE). A WAV is defined as "a flaw in the application that stems from coding defects and causes severe damage to the application upon exploitation". To identify and mitigate vulnerabilities that may be exploited by attackers, a penetration testing method or ethical hacking is used. The Open Web Application Security Project (OWASP) provides the standard for such penetration testing methodology to test web applications and could be used to evaluate the effectiveness of web vulnerability scanners.[3]

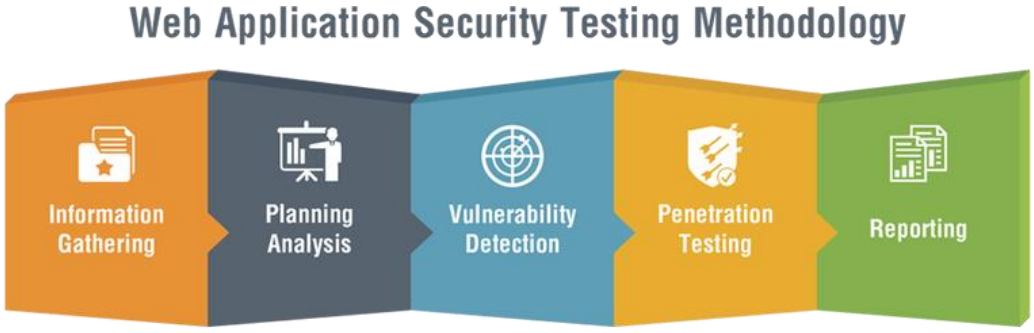

Fig. 1: Web application VAPT services.

Web Application Vulnerability Scanners (WAVS) are tools used by penetration testers. It is used to conduct web application evaluations with the primary goal of identifying and mitigating potential vulnerabilities to prevent security breaches. Security measures should be incorporated throughout the development life cycle rather than being added and tested only at the final stages of the development life cycle [4]. The challenge here is that when using different WAVS, they will not offer the same vulnerability results for the same target. These differences arise from the different levels of precision, speed, and coverage in terms of finding various vulnerabilities, for example, with respect to XSS and SQLi attacks. These differing results mean that penetration testers need to utilise multiple WAVS and thus, the accuracy and detection coverage of the penetration testing report depend on the testers' knowledge and experience of the most effective WAVS in particular situations. This manual approach to Web Application Penetration Testing (WAPT) can be time-consuming, costly, and subject to human error [5].

2. Literature Survey

Abdulghaffar, et al. [6] presented a novel framework designed to automate the operation of multiple Web Application Vulnerability Scanners (WAVS) within a single platform. The framework generates a combined vulnerabilities report using two algorithms: an automation algorithm and a novel combination algorithm that produced comprehensive lists of detected vulnerabilities. Goutam, et al. [7] focused on web application security. In this proposed research work, a framework has been built to test the vulnerabilities. This framework has the same working module as that of a financial institution's website. After penetration testing, based on the vulnerability further, a framework will be designed which will provide more security to such web sites. The developed framework can be used in several institutions, companies or organizations to test the vulnerability. Nagpure, et al. [8]

proposed Vulnerability Assessment and Penetration Testing are two different vulnerability testing. These tests have different strengths and are frequently combined to get a more complete vulnerability analysis. Penetration Testing and Vulnerability Assessments execute two different tasks, usually with distinctive outcomes, within the same area of application. For any organization, proper working of security arrangement is checked by Vulnerability Assessment and Penetration Testing. Web applications vulnerable to attacks like Session exploitation, Cross-Site Scripting, SQLi, Cross Site Request Forgery, Buffer overflows, and Security Misconfiguration etc. Alhamed, et al. [9] designed to provide prevention and detection controls against attacks in the network. A tester looks for security issues in the network operation, design, or implementation of the particular company or organization. Thus, it is important to identify the vulnerabilities and identify the threats that may exploit them in order to find ways to reduce their dangers. The ports at risk are named and discussed in this study. Udosi, et al. [10] processed of an audit includes several steps, such as penetration testing, vulnerability scans, and network assessments. After the audit is conducted, a report that contains the vulnerabilities is generated to help the organization to understand the current situation from this perspective. Risk exposure should be as low as possible because in cases of an attack, the entire business is damaged.

3. Proposed Methodology

The research focuses on understanding how web-based authentication mechanisms behave under normal usage as well as malicious conditions. It examines how user credentials are collected, processed, and validated against a backend database, highlighting the interaction between the application layer and the database layer. Special attention is given to how improper handling of user inputs can expose systems to serious security threats. By observing both normal user behaviour and crafted malicious inputs, the study emphasizes the importance of secure coding practices in real-world web applications.

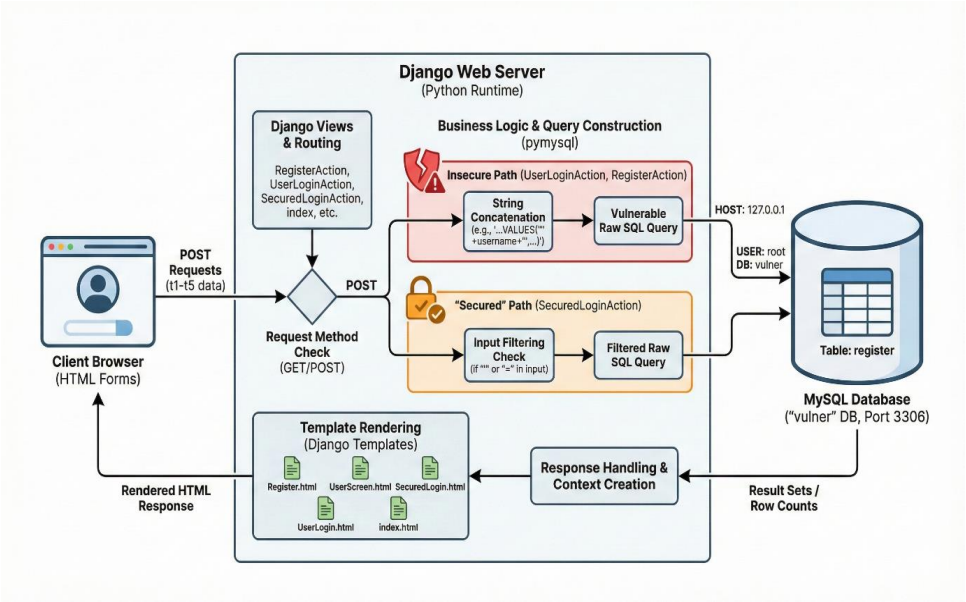

Fig. 2: Proposed system architecture of real-time SQLi defense.

The study further analyzes different authentication approaches to compare vulnerable and protected access mechanisms. One approach demonstrates how direct query construction using user input can allow unauthorized access through manipulation techniques, while another approach introduces basic validation checks to identify suspicious patterns. This comparative analysis helps in understanding how attackers exploit weak input handling and how even simple defensive checks can reduce risk,

though not eliminate it entirely. The behavior of the system under attack scenarios provides clear insights into common weaknesses present in beginner-level authentication implementations.

3.1 SQLi Detection Workflow

The SQLi Detection Workflow is designed to safeguard the authentication system by identifying input patterns that indicate malicious intent. This module monitors user-submitted credentials and analyses them for harmful characters or SQL keywords commonly used in exploits, such as ', --, OR, or conditional expressions like 1=1. By performing server-side validation before constructing or executing database queries, the system prevents attackers from manipulating SQL logic or bypassing authentication. This workflow adds an essential defensive layer, ensuring that only clean and trusted input is processed while potential injection attempts are immediately blocked. Through this proactive detection mechanism, the system enhances overall application security and protects sensitive user data from unauthorized access.

User Enters Username and Password: The detection workflow begins when the user inputs their login credentials into the login form. Both fields username and password are directly submitted to the server without any client-side validation. Since the system allows free-form text, users may unknowingly or intentionally enter special characters or SQL fragments, creating a potential risk for SQLi. This step marks the starting point where malicious input can enter the system.

Server Receives User Input: Once the form is submitted, Django processes the POST request and retrieves the entered values using request.POST.get (). At this stage, the system treats these values as plain text without performing sanitization. Whatever is typed by the user, including harmful SQL operators, reaches the backend unchanged. This step highlights the importance of server-side filtering because client-side validation cannot be trusted in security-sensitive workflows.

System Checks for Malicious Characters or Patterns: The system performs a defensive check by scanning the input for dangerous characters such as ', ", =, --, OR, or patterns like 1=1. These symbols are red flags for SQLi attempts. The detection logic evaluates if the input contains anything that could manipulate the structure of the SQL query. This step serves as the primary layer of security to identify abnormal or suspicious login attempts before processing them further.

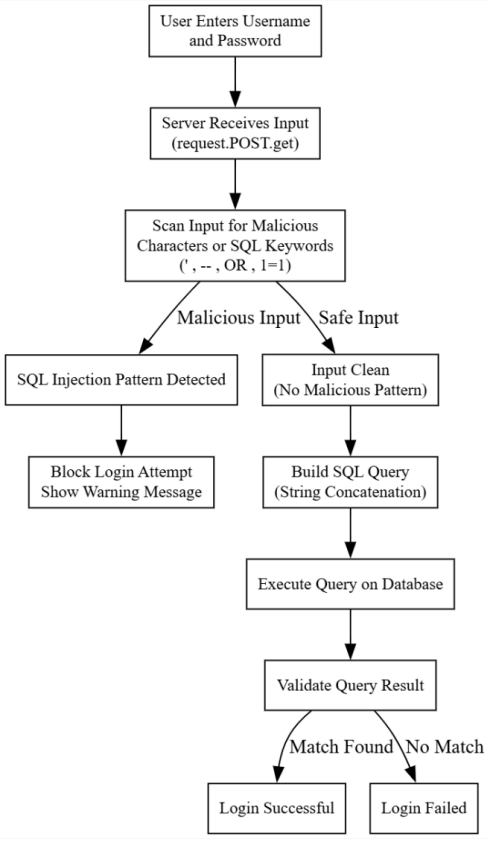

Fig. 3: SQLi detection workflow.

Detection of SQLi Attempt: If the system identifies any suspicious character or keyword in the input, it immediately classifies the request as a potential SQLi attack. This is done before any database query execution, ensuring that harmful SQL does not reach the database layer. The detection mechanism effectively prevents attackers from crafting queries that bypass authentication or manipulate data. At this point, the workflow branches into the defensive response sequence.

Block the Login Attempt and Display Warning: Upon detecting malicious input, the system halts the login process entirely and avoids executing any database query. Instead, it displays a warning message such as "SQLi Attack Detected" to the user. This prevents unauthorized access and helps administrators monitor security events. This step ensures that harmful payloads cannot exploit vulnerabilities in the authentication logic.

Proceed with Normal Login if No Malicious Pattern Found: If the input does not contain any harmful characters or suspicious patterns, it is classified as clean. The system then proceeds to build the SQL query for verifying the credentials. This step ensures legitimate users can authenticate normally while keeping attackers out. Clean input represents a safe pathway that allows the workflow to continue into the database query stage.

Build SQL Query with Clean Input: Once input is validated, the backend constructs a SQL SELECT query to match the username and password in the database. Although the query still uses string concatenation, the risk is reduced because malicious characters were filtered earlier. The query is structured to retrieve only the matching user record. This step acts as the bridge between detection and actual authentication processing.

3.2 Secured Login Workflow

The secured login workflow operates by first passing user input through validation filters that remove or block suspicious characters. Only after inputs are verified, the SQL query is executed in a controlled manner using clean parameters within the query. Because the detection layer cancels harmful input upfront, the final SQL statement cannot be manipulated or injected with malicious patterns. As a result, the database returns accurate and safe results allowing login only when credentials match exactly and redirects the user accordingly while maintaining protection despite using string concatenation.

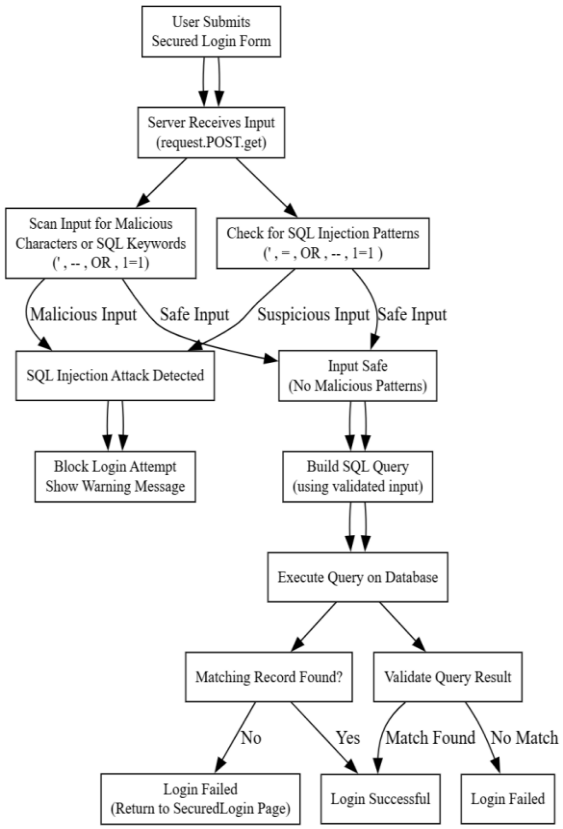

Fig. 4: Secured login workflow.

User Submits Secured Login Form: The secured login workflow begins when the user enters their username and password in the secured login interface and submits the form. These credentials are sent to the server via a POST request. Unlike the vulnerable login, this workflow is specifically designed to detect suspicious input patterns before interacting with the database.

Server Receives User Credentials: Once the form data reaches the Django backend, the server extracts the username and password using request.POST.get (). These values are temporarily stored in variables for validation. At this stage, the system prepares to analyse the input to detect any signs of SQLi attempts before further processing.

Input Validation and Malicious Pattern Detection: The server checks the username and password for common SQLi indicators such as single quotes ('), double quotes, equal signs (=), or logical operators. If any of these characters or patterns are detected, the system immediately identifies the request as a potential SQLi attack. This preventive mechanism ensures that malicious payloads never reach the query-building phase.

Block Attack and Display Warning Message: If suspicious characters are found, the server blocks the login process right away. Instead of sending a query to the database, the system generates a

security alert message such as "SQLi Attack Detected." This response helps protect the backend by preventing harmful code from being executed and keeps the application safe from unauthorized access attempts.

Proceed with Clean Input: If no malicious patterns are detected, the workflow continues with the validated input. The server now considers the username and password safe enough to use in the SQL query. This step forms the basis of the secured login process, ensuring that only clean, human-intended input is forwarded for authentication.

4. Results description

Fig. 5 depicts the sign-up page screen, representing the process of new user registration within the secure login framework. It illustrates how user information is collected and prepared for secure storage while maintaining validation procedures. The figure reflects the importance of secure onboarding as part of preventing unauthorized system usage. It demonstrates how the registration process contributes to identity verification and controlled account creation.

Fig. 5: Sign Up Page.

Fig. 6 illustrates the login and welcome page screen, showing the stage where authenticated users successfully access the system after credential verification. It depicts the transition from authentication to authorized system interaction, indicating successful login validation. The figure represents how user sessions are established securely within the framework. It highlights the role of secure login confirmation in ensuring proper access control and system integrity.

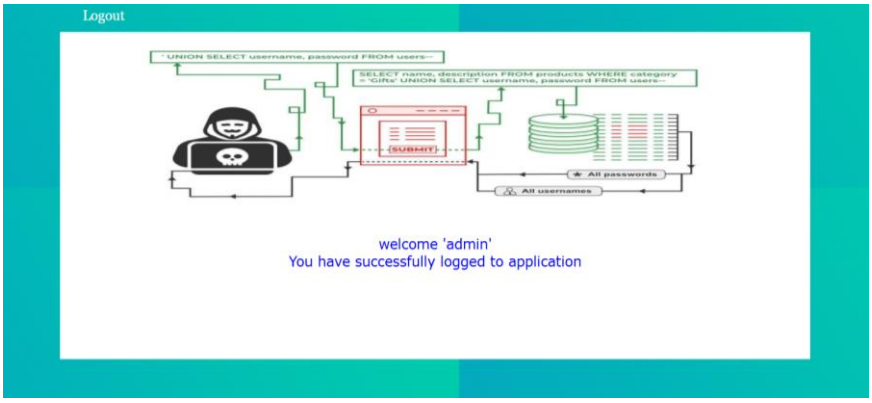

Fig. 6: Login and Welcome Page.

Fig. 7 depicts the secure login module screen, representing the authentication interface enhanced with SQLi protection mechanisms. It illustrates how login inputs are processed under security-aware validation to prevent malicious query execution. The figure reflects the system's approach to

safeguarding database interactions through structured input handling. It demonstrates the integration of real-time security monitoring within the authentication process.

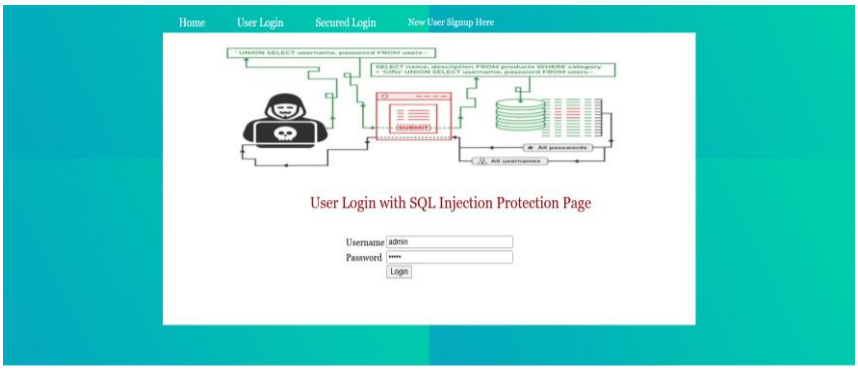

Fig. 7: Secure Login Module.

Fig. 8 illustrates the secure login authentication screen, representing the stage where validated credentials allow access under enforced security policies. It depicts the confirmation of successful authentication while maintaining protection against potential vulnerabilities. The figure reflects how security checks are embedded throughout the login lifecycle to ensure safe system usage. It highlights the continuity of protection even after authentication is completed.

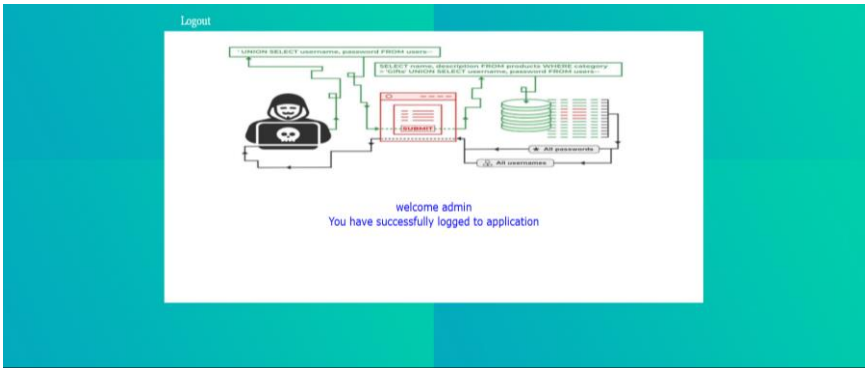

Fig. 8: Secure Login Authentication.

Fig. 9 depicts the SQLi detection screen, illustrating the system's capability to identify and respond to malicious input attempts in real time. It represents the detection and prevention mechanism that safeguards database queries from unauthorized manipulation. The figure reflects the integration of exploit-aware monitoring that ensures system resilience against common injection attacks. It demonstrates how alerts or detection outcomes are communicated within the interface.

Fig. 9: SQLi detection.

5. Conclusion

The research successfully demonstrates the importance of secure authentication in web applications by comparing a vulnerable login system with a secured, SQLi–protected login module. Through this implementation, it becomes clear how easily malicious users can exploit unsensitized inputs to bypass authentication and manipulate databases. The project highlights the weaknesses of traditional query-building methods and emphasizes the need for secure coding techniques such as input validation and structured database communication. By using Django and PyMySQL, the system provides a realistic environment for understanding how security flaws occur and how they can be prevented. The secured login module effectively blocks suspicious patterns, ensuring that only legitimate users gain access. The project not only strengthens knowledge about SQLi attacks but also encourages developers to adopt safe practices, such as filtering inputs, validating credentials, and avoiding direct string concatenation in SQL queries.

Reference

[1] Altulaihan, E.A.; Alismail, A.; Frikha, M. A Survey on Web Application Penetration Testing. Electronics 2023, 12, 1229.

[2] Sadqi, Y.; Maleh, Y. A systematic review and taxonomy of web applications threats. Inf. Secur. J. Glob. Perspect. 2022, 31, 1–27.

[3] Trickel, E.; Pagani, F.; Zhu, C.; Dresel, L.; Vigna, G.; Kruegel, C.; Wang, R.; Bao, T.; Shoshitaishvili, Y.; Doupé, A. Toss a Fault to Your Witcher: Applying Grey-box Coverage-Guided Mutational Fuzzing to Detect SQL and Command Injection Vulnerabilities. In Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 21–25 May 2023; pp. 2658–2675.

[4] Deepa, G.; Thilagam, P.S. Securing web applications from injection and logic vulnerabilities: Approaches and challenges. Inf. Softw. Technol. 2016, 74, 160–180.

[5] Alhamed, M.; Rahman, M.M.H. A Systematic Literature Review on Penetration Testing in Networks: Future Research Directions. Appl. Sci. 2023, 13, 6986.

[6] Abdulghaffar K, Elmrabit N, Yousefi M. Enhancing Web Application Security through Automated Penetration Testing with Multiple Vulnerability Scanners. Computers. 2023; 12(11):235. https://doi.org/10.3390/computers12110235

[7] A. Goutam and V. Tiwari, "Vulnerability Assessment and Penetration Testing to Enhance the Security of Web Application," 2019 4th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 2019, pp. 601-605

[8] S. Nagpure and S. Kurkure, "Vulnerability Assessment and Penetration Testing of Web Application," 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, India, 2017, pp. 1-6,

[9] Alhamed M, Rahman MMH. A Systematic Literature Review on Penetration Testing in Networks: Future Research Directions. Applied Sciences. 2023; 13(12):6986. https://doi.org/10.3390/app13126986

[10] Tudosi A-D, Graur A, Balan DG, Potorac AD. Research on Security Weakness Using Penetration Testing in a Distributed Firewall. Sensors. 2023; 23(5):2683. https://doi.org/10.3390/s23052683

[11] Mahesh Ganji. (2025). Enhancing Oracle Cloud HR Reporting Through AI-Driven Automation. Journal of Science & Technology, 10(6), 28–36. https://doi.org/10.46243/jst.2025.v10.i06.pp28-36

[12] Todupunuri, A. (2025). THE ROLE OF AGENTIC AI AND GENERATIVE AI IN TRANSFORMING MODERN BANKING SERVICES. American Journal of AI Cyber Computing Management, 5(3), 85–93. https://doi.org/10.64751/ajaccm.2025.v5.n3.pp85-93

[13] Todupunuri, A. . (2024). Artificial Intelligence Ethics: Investigating Ethical Frameworks, Bias Mitigation, and Transparency in AI Systems to Ensure Responsible Deployment and Use of AI Technologies. International Journal of Innovative Research in Science,Engineering and Technology, 13(09), 1–14. https://doi.org/10.15680/ijirset.2024.1309002

[14] Sushma Babburi. (2025). Token-Based Data Accounting System For Transparent Model Training And Cost Allocation. American Journal of AI Cyber Computing Management, 5(4), 463–474. https://doi.org/10.64751/ajaccm.2025.v5.n4.pp463-474

[15] Snigdha Gaddam. (2025). SOFTWARE STACK PREPARED FOR AI TRANSITIONING FROM MODULES TO MODELS. American Journal of AI Cyber Computing Management, 5(4), 451–462. https://doi.org/10.64751/ajaccm.2025.v5.n4.pp451-462

[16] Gaddam, S. INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING.

[17] Bajarang Bhagwat, V. (2023). Optimizing Payroll to General Ledger Reconciliation: Identifying Discrepancies and Enhancing Financial Accuracy. JOURNAL OF ADVANCE AND FUTURE RESEARCH, 1(4). https://doi.org/10.56975/jaafr.v1i4.501636

[18] Srinivasa Kalyan Immadi. (2025). Harnessing Artificial Intelligence In Oracle Hcm: Revolutionising Workforce Management With Automation And Predictive Analytics. International Journal of Data Science and IoT Management System, 4(4), 7–13. https://doi.org/10.64751/ijdim.2025.v4.n4.pp7-13

[19] S. M. K. P. (2025). Cryptography in iOS: A Study of Secure Data Storage and Communication Techniques. International Journal on Science and Technology, 16(1). https://doi.org/10.71097/ijsat.v16.i1.1403

[20] Suhasnadh Reddy Veluru, Sai Teja Erukude, and Viswa Chaitanya Marella. 2025. Multimodal Detection of Fake Reviews using BERT and ResNet-50. In 2025 4th International Conference on Innovative Mechanisms for Industry Applications (ICIMIA). IEEE, 877–882.

[21] Cyril, H. P. (2025). Event-Driven Provisioning Architectures For Modern Telecom Networks: Overcoming Legacy Limitations And Enabling Autonomous 6g Operations. International Journal of Advanced Research in Computer Science, 16(6), 75–82. https://doi.org/10.26483/ijarcs.v16i6.7389

[22] Jay Bharat Mehta. (2025). AUTONOMOUS PATCH VALIDATION FOR ZERO-DAY EXPLOITS IN ENTERPRISE CLOUDS. International Journal of Applied Mathematics, 38(4s), 1270–1285. https://doi.org/10.12732/ijam.v38i4s.685

[23] Reddy, S. K. (2025). Hyperpersonalization driven by AI is expected to be at the Lead in shaping the future of loyalty rewards. Journal of Emerging Technologies and Innovative Research.

[24] Reddy, S. K. R. (2021). Strengthening the Security of Loyalty Reward Systems: An In-Depth Analysis of Emerging Cyber Threats and Protection Mechanisms. Journal of Computational Analysis and Applications, 29(6).

[25] Poojari, R. (2026). Privacy-Preserving Generative AI in Healthcare Systems Using Federated Learning Approaches. International Journal of Data Science and IoT Management System, 5(1), 78-88.

[26] Uday Kumar Kalae. (2025). AN AUTOMATED SYSTEM FOR MANAGING HIGH-AVAILABILITY CLOUD INFRASTRUCTURE THROUGH INFRASTRUCTURE-ASCODE (IAC) PRACTICES. American Journal of AI Cyber Computing Management, 5(2), 42–50. https://doi.org/10.64751/ajaccm.2025.v5.n2.pp42-50

[27] Saikumar, B. (2024). Optimizing Crew Scheduling and Absence Management using Microservices: Enhancing Reliability and Efficiency in Crew Management Systems. International Journal of Enhanced Research in Management & Computer Applications, 13(11), 50–55. https://doi.org/10.55948/ijermca.2024.0116

[28] Saikumar, B. (2023). Enhancing Client Engagement through AI-Driven Real-Time Reporting and Automated Alerts. International Journal of Enhanced Research in Science, Technology & Engineering, 12(11), 111–117. https://doi.org/10.55948/ijerste.2023.1115