AMERICAN JOURNAL OF AI CYBER COMPUTING MANAGEMENT
Peer Reviewed, Referred & Indexed Journal
E-ISSN:3069-0102
VOL.6, NO. 2(2026)

Dynamic Pattern Intelligence for SQLi-Centric Anomaly Detection in Controlled Input Environments

Musham Swetha[1], K. Sharmila Reddy[2*], Guntipally Keerthana[3], Aruri Raghavendra[3], Banothu Lavanya[3], Chinthakindi Teja[3]

[1]Assistant Professor, [2]Associate Professor & Head, [3]UG Student, [1,2]Department of Computer Science and Engineering

[1,2]Vaagdevi Engineering College, Bollikunta, Warangal, 506005, Telangana, India.

*Correspondence: K. Sharmila Reddy (sharmilakreddy@gmail.com)

ABSTRACT

Web applications have become a critical component of modern digital infrastructure, supporting services such as banking, e-commerce, healthcare, and enterprise systems. With the increasing reliance on authentication mechanisms, these systems are exposed to significant security risks due to improper input handling and weak validation practices. Such vulnerabilities can lead to unauthorized access, data breaches, and compromise of sensitive information, making security a major concern in web-based environments. The primary challenge lies in the limitations of manual rule-based systems, which rely on predefined patterns and static logic to detect malicious inputs. Although effective for known attack signatures, these systems fail to identify complex, obfuscated, or newly emerging threats. Continuous manual updates are required to maintain their effectiveness, resulting in increased maintenance effort and reduced scalability. This leads to lower detection accuracy and higher chances of system exploitation in dynamic environments. To overcome these limitations, the proposed system introduces a dynamic exploit-and-defense framework for real-time detection in authentication systems. It utilizes models such as Logistic Regression (LRC), Random Forest (RF), Gaussian Naive Bayes (GNB), and Multinomial Naive Bayes (MNB) along with Term Frequency–Inverse Document Frequency (TF-IDF) based feature extraction to classify inputs. The system is implemented using Django for application handling, Structured Query Language (SQL) for data management, and bcrypt for secure password hashing. It identifies multiple input-based threat categories including SQL Injection (SQLi), Cross-Site Scripting (XSS), Command Injection (CMDI), Local File Inclusion (LFI), and Server-Side Template Injection (SSTI) as target outputs.

Key words: Input Validation, Vulnerability Detection, Dynamic Exploit-Defense Framework, Machine Learning, TF-IDF, Supervised Machine Learning, Bcrypt Password Hashing.

1. INTRODUCTION

SQL injection (SQLi) continues to be one of the most severe and widely exploited vulnerabilities in modern web applications, making its detection a critical research focus in the field of information security. With the rapid expansion of dynamic and database-driven systems, attackers increasingly target input validation weaknesses to manipulate backend queries. Among the various detection approaches, static analysis has gained prominence due to its ability to examine application source code without requiring execution. This technique relies on structural and semantic representations such as abstract syntax tree (AST), control flow graph (CFG), and call graph models to trace data flow and identify insecure coding patterns that may lead to injection vulnerabilities [1]. Although static analysis has matured significantly and enables automated large-

scale vulnerability detection, its effectiveness is often constrained in complex application architectures, especially those utilizing object-oriented database extension (OODBE), where abstraction layers and indirect data flows obscure the identification of vulnerable points [2].

A web application vulnerability (WAV) arises from defects introduced during the software development process, which can compromise the confidentiality, integrity, or availability of the system when exploited. To proactively identify such weaknesses, penetration testing, commonly referred to as ethical hacking, is widely employed. This approach simulates real-world attack scenarios to evaluate system defenses and uncover exploitable vulnerabilities. Standardized methodologies, such as those defined by the Open Web Application Security Project (OWASP), provide structured guidelines for conducting comprehensive security assessments and benchmarking the effectiveness of detection tools [3]. These methodologies emphasize systematic testing procedures, including input validation checks, authentication testing, and injection attack simulations, as illustrated in Figure 1.

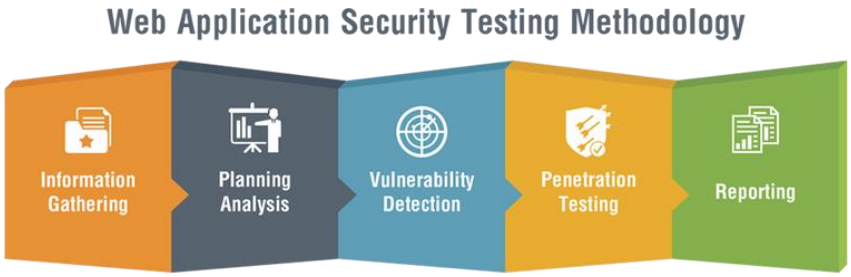

Figure 1: Web Application VAPT Services.

Web Application Vulnerability Scanners (WAVS) are integral tools in the penetration testing process, designed to automate the detection of common vulnerabilities such as SQLi and cross-site scripting (XSS). These scanners analyze web applications by crawling pages, injecting payloads, and evaluating responses to identify potential security flaws. However, a significant challenge lies in the inconsistency of results produced by different scanners when evaluating the same application. Variations in detection algorithms, scanning depth, payload generation strategies, and coverage capabilities lead to discrepancies in identifying vulnerabilities. Consequently, security professionals often rely on multiple WAVS tools to achieve broader coverage and improve detection reliability. This multi-tool dependency increases operational complexity and places a strong reliance on the expertise of penetration testers to interpret and consolidate results effectively.

Furthermore, the traditional approach to Web Application Penetration Testing (WAPT) is largely manual and resource-intensive, involving tool configuration, result validation, and report generation. This process can be time-consuming, costly, and prone to human error, particularly in large-scale or complex systems [4], [5]. The limitations of existing methods highlight the need for more intelligent, unified, and automated solutions that can enhance detection accuracy, reduce redundancy, and minimize reliance on human intervention. Advancements in hybrid analysis techniques and AI-driven security models are increasingly being explored to address these challenges and improve the overall efficiency of web application vulnerability detection frameworks.

2. RELATED WORK

2.1 Automated Vulnerability Assessment Frameworks

Abdulghaffar, et al. [6] proposed an integrated and automated framework that orchestrates the execution of multiple WAVS within a unified environment. Their system incorporates an automation algorithm to manage scanner operations along with a combination algorithm to aggregate and normalize outputs from different tools. This approach enhances vulnerability coverage, reduces redundancy, and generates a consolidated report highlighting critical security issues more effectively than individual scanners. Additionally, the framework improves operational efficiency by minimizing manual intervention and enabling centralized vulnerability management.

Udosi, et al. [10] outlined a systematic cybersecurity audit process integrating vulnerability scanning, penetration testing, and network evaluation. Their approach emphasizes generating detailed post-assessment reports that provide actionable insights into security gaps. The study highlights that minimizing risk exposure is critical, as cyberattacks can disrupt business operations and cause financial and reputational damage, thereby reinforcing the importance of continuous security auditing.

2.2 Web Application Security Testing and Penetration Frameworks

Goutam, et al. [7] developed a structured web application security testing framework modeled on real-world financial systems to ensure practical relevance. Their approach combines penetration testing with enhanced defensive mechanisms, making the framework reusable across multiple domains such as enterprises, institutions, and organizations. This improves security validation practices and supports scalable deployment.

Nagpure, et al. [8] presented a comparative analysis of vulnerability assessment and penetration testing, emphasizing their complementary roles. While vulnerability assessment identifies and classifies weaknesses, penetration testing evaluates exploitability through simulated attacks. Their study demonstrates that integrating both approaches provides a more comprehensive understanding of risks, particularly for threats such as SQLi, XSS, CSRF, session hijacking, buffer overflows, and configuration flaws.

Altulaihan, et al. [13] conducted an extensive study on penetration testing methodologies, discussing various web-based threats and defense mechanisms. Their work includes a comparison of widely used penetration testing tools, outlining their strengths, limitations, and applicability across different scenarios, thereby assisting practitioners in selecting appropriate tools based on system requirements.

2.3 Network Security and Risk Assessment Approaches

Alhamed, et al. [9] introduced a framework focused on strengthening network security through proactive detection and prevention strategies. Their study emphasizes identifying vulnerable network ports, analyzing configurations, and understanding potential attack vectors. By mapping threats to system weaknesses, the framework supports prioritization of mitigation strategies and enhances overall network resilience.

Sarker, et al. [15] highlighted the importance of continuous security assessment through regular penetration testing. Their work demonstrates that structured testing based on standardized frameworks enables the identification of vulnerabilities across operating systems, applications, and network infrastructures, facilitating timely remediation and improving overall organizational security posture.

2.4 Advanced and Intelligent Penetration Testing Systems

Xiao, et al. [11] proposed SatGuard, a framework designed for satellite system security that integrates a three-dimensional penetration testing methodology with a nonlinear risk assessment model. The system leverages LLMs such as GPT-4 and DeepSeek-R1 to enable semi-automated vulnerability discovery and exploitation. This significantly enhances efficiency, scalability, and adaptability in complex and dynamic environments.

Shahid, et al. [12] evaluated automated penetration testing tools aimed at reducing human effort, cost, and time. Their findings indicate that while automation improves efficiency, existing tools face challenges such as incomplete scanning coverage and false positives or negatives. The study also emphasizes the need for quantitative evaluation methods to assess tool effectiveness and guide improvements.

2.5 Machine Learning in Vulnerability Detection

Moreno, et al. [14] explored the application of ML techniques in vulnerability detection and penetration testing. Their study shows that ML models can identify hidden attack patterns and anomalies by learning from large datasets, improving detection capabilities. However, they also highlight that real-world penetration testing environments are complex and dynamic, requiring human expertise to interpret results and execute advanced exploitation strategies alongside automated methods.

3. PROPOSED SYSTEM

The system is designed to provide a multi-layered security framework for detecting and preventing web-based attacks in real time, as illustrated in Figure 2. It integrates both rule-based and machine learning approaches to analyze user inputs such as usernames and passwords within a unified architecture. The process begins with user interaction through login interfaces, where inputs may be either legitimate or malicious, and these inputs are captured and forwarded for further analysis. The system processes requests through dual login paths consisting of a secured login and a vulnerable login, where the secured path enforces strict validation while the vulnerable path simulates attack scenarios for analysis and understanding of system weaknesses. In the first layer, regex-based detection is applied to identify known attack patterns such as SQLi, XSS, CMDI, LFI, and SSTI by scanning for predefined signatures including SQL keywords, script tags, command separators, and directory traversal patterns.

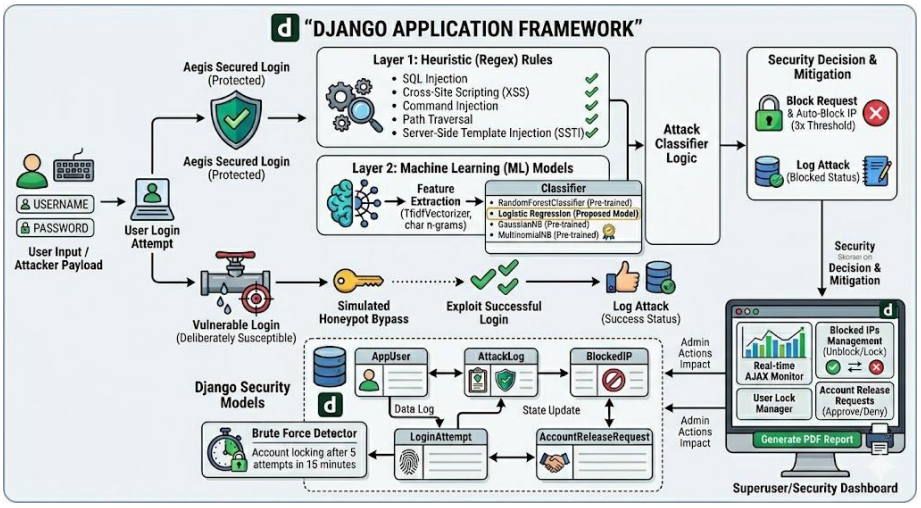

Figure 2: Proposed system architecture

This layer acts as the initial line of defense by providing fast and efficient detection of common threats. In the second layer, inputs are transformed into TF-IDF feature representations and analyzed using ML models such as LRC, RF, GNB, and MNB to detect complex, hidden, and previously unseen attack patterns. This layer enhances detection accuracy and enables identification of obfuscated attacks that bypass traditional rule-based systems. The outputs from both layers are then integrated into an attack classification module, where final decisions are made based on combined results to determine whether the input is normal or malicious. Based on this classification, the system either grants access or blocks the request and flags it as an attack. Detected threats are logged in the database along with relevant details such as IP address and payload for further analysis and auditing. The system continuously monitors repeated attack attempts and enforces threshold-based blocking mechanisms to prevent abuse. Additionally, the architecture includes administrative controls that support real-time monitoring, account management, and report generation. These features ensure continuous system supervision, improved traceability, and effective enforcement of security policies. The integrated design provides a robust, scalable, and adaptive solution for real-time web application security.

3.1 LRC model

LRC is a supervised machine learning algorithm used for classification tasks by estimating probabilities using a logistic (sigmoid) function, as illustrated in Figure 3. In the system, LRC is used as the final model to classify input payloads into different attack categories based on TF-IDF feature representations. It computes a weighted sum of input features and transforms it into a probability value between 0 and 1. The model learns optimal weights during training by minimizing a loss function using optimization techniques. LRC is efficient, interpretable, and performs well on high-dimensional data. It provides stable and consistent results compared to complex models. The model generalizes well to unseen data and avoids overfitting.

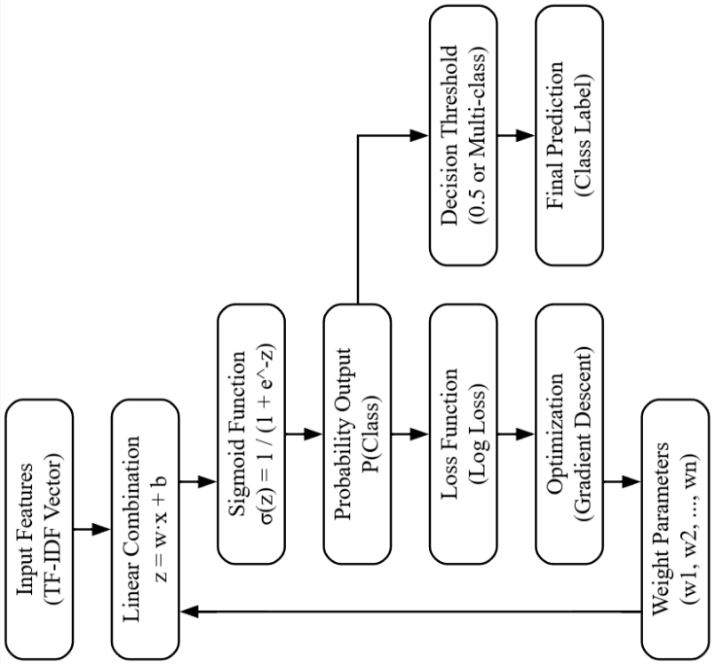

Figure 3: Internal workflow of LRC

Input Feature Preparation: The system receives processed input in the form of TF-IDF feature vectors. These numerical features represent textual patterns extracted from input payloads. This transformation ensures that the input is structured and compatible with the LRC model. It provides a suitable representation for learning feature relationships.

Linear Combination Calculation: The model computes a linear combination of input features and corresponding weights. This is represented as $z = w \cdot x + b$, where weights are learned during training. This step determines how strongly each feature contributes to the prediction. It forms the foundation of the classification process.

Sigmoid Transformation: The linear output is passed through a sigmoid function to convert it into a probability value. This function maps any real number into a range between 0 and 1. It allows the model to interpret outputs as probabilities. This step is essential for binary and multi-class classification.

Probability Estimation: The model generates probability scores for each class based on the sigmoid output. These probabilities represent the likelihood of the input belonging to each class. This step enables comparison between different class outcomes. It provides a probabilistic interpretation of predictions.

Loss Computation and Optimization: During training, the model calculates loss using a function such as log loss. The optimization process adjusts weights using gradient descent to minimize this loss. This iterative process improves model accuracy over time. It ensures that the model learns optimal parameters.

Decision and Final Prediction: The model applies a decision threshold to convert probabilities into class labels. The class with the highest probability is selected as the final output. This step completes the classification process. It provides accurate and consistent predictions for input data.

4. RESULTS AND DISSCUSSION

Figure 4 depicts the detailed record of login attempts along with detected cyberattacks within the system. It illustrates how the system captures and categorizes different types of malicious activities based on input analysis. The figure highlights the distinction between vulnerable and secured login attempts. It also reflects how repeated attacks are tracked and stored for further evaluation. Additionally, it demonstrates the system's capability to maintain comprehensive logs for monitoring suspicious behavior.

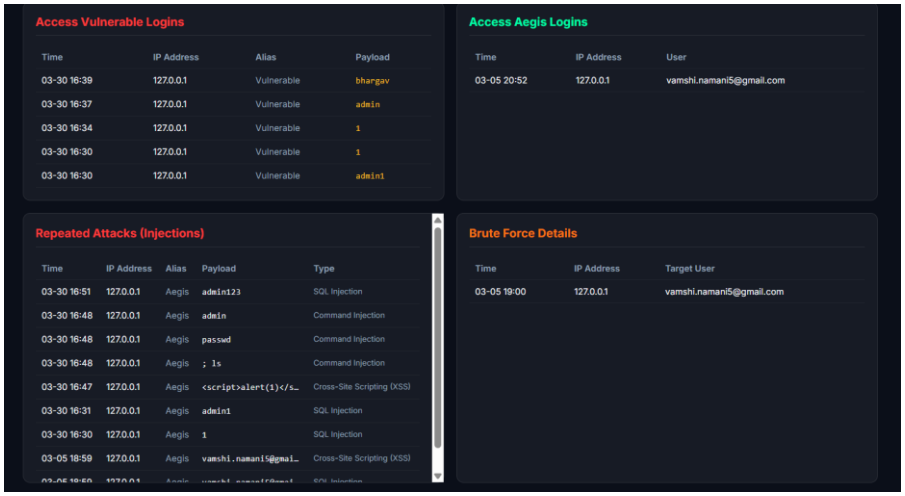

Figure 4: login attempts and detected cyberattacks.

Figure. 5 illustrates the confusion matrix obtained for the LRC model, representing classification performance across six attack categories. The model correctly classifies Command Injection with 8 instances, while 2 instances are misclassified as SSTI. Cross-Site Scripting (XSS) achieves perfect classification with 11 correctly predicted instances and no misclassifications. The Normal class records 9 correct predictions with 2 instances incorrectly classified as SSTI. Path Traversal shows 9 correct classifications, with 2 instances misclassified as Command Injection. SQL Injection achieves 10 correct predictions with no observed misclassification. SSTI demonstrates perfect classification with 11 correctly identified instances. Overall, the matrix indicates strong performance of the LRC model, with most values concentrated along the diagonal and only a few misclassifications occurring between related attack classes

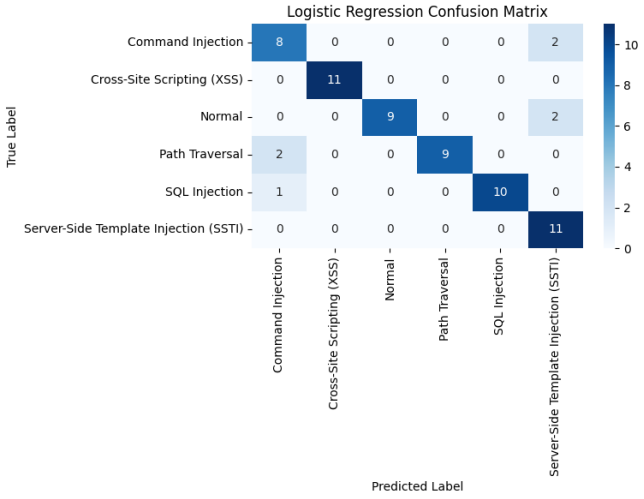

Figure 5: Confusion Matrix obtained for Models LRC

Figure. 6 illustrates the security enforcement mechanism implemented in the system, where access control is strengthened through a threshold-based protection strategy. It depicts how the system continuously monitors login attempts by tracking invalid credentials and suspicious inputs across sessions. Each unsuccessful attempt is recorded and incremented in the system log, enabling accurate counting of repeated failures. Once the number of consecutive failed attempts reaches the defined threshold of 3, the system automatically restricts further login access and flags the activity as potentially malicious. The figure also shows the generation of alert notifications to inform the user about the restriction and provides an option to request account reactivation through administrative approval.

Figure 7 illustrates the detection and prevention of Cross-Site Scripting (XSS) attacks within the system by analyzing user input for malicious script patterns. It depicts how payloads such as <script>alert(1)</script> are identified using pattern recognition techniques that detect script tags and embedded JavaScript execution. The system evaluates input fields for potentially harmful content and classifies such inputs as XSS attacks when suspicious patterns are detected. The figure shows that repeated injection attempts from the same source lead to automatic IP blocking, preventing further interaction with the system. Additionally, the detection module labels the request as malicious and ensures that it is blocked before execution, thereby protecting the application from client-side script exploitation.

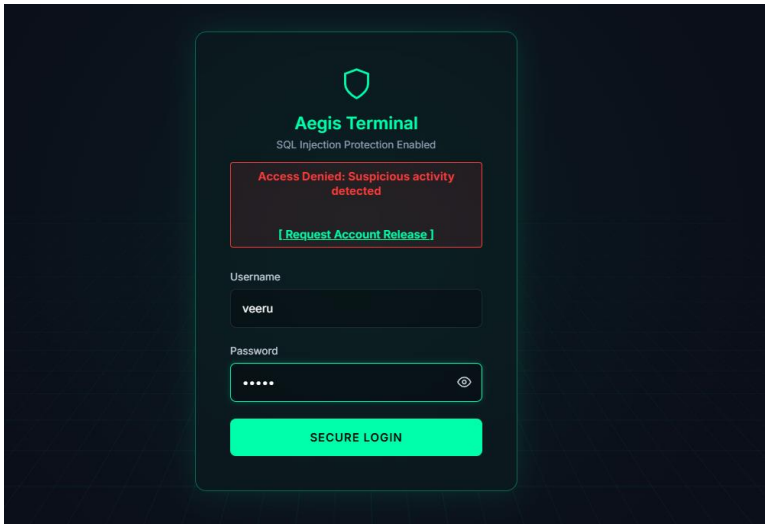

Figure 6: Access Denying after 3 Unsuccessful Logins

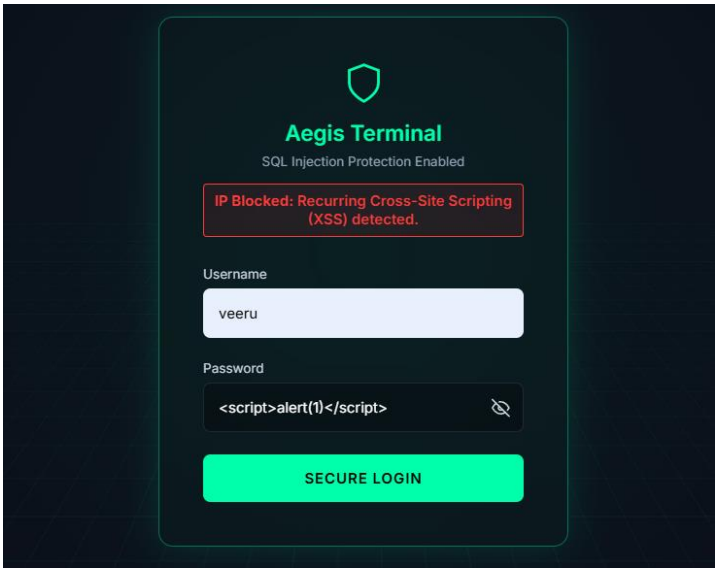

Figure 7: Detecting XSS Attack

Figure 8 illustrates the detection and prevention of Command Injection (CMDI) attacks within the system, where inputs such as ls, whoami, and command separators are identified as malicious patterns attempting to execute system-level operations. It depicts how the input validation module analyzes user inputs for command execution signatures and shell-related syntax that could compromise the system. The detection mechanism applies both rule-based pattern matching and learning-based classification to accurately identify such threats. The figure shows that each detected CMDI attempt is logged and associated with the originating IP address for monitoring purposes. When multiple command injection attempts are observed from the same source, the system enforces threshold-based blocking, resulting in denial of further access. Additionally, the classification module labels the detected input explicitly as a command injection attack and prevents its execution at the server level.

Figure 9 illustrates the detection of Path Traversal (LFI) attacks within the system, where malicious input such as "../" is identified as an attempt to access restricted directories. It depicts how the system analyzes user input fields, particularly the password or query parameters, to detect

traversal patterns that aim to bypass directory restrictions and access sensitive files. The detection mechanism applies pattern matching techniques to identify sequences commonly associated with LFI attacks, such as relative path indicators and unauthorized file access attempts. The figure shows that upon detection of such input, the system immediately generates a security alert indicating "Path Traversal Detected," thereby notifying the presence of a malicious request. The request is blocked before being processed further, ensuring that no unauthorized file access occurs on the server. Additionally, the system classifies the input under LFI attack type and prevents execution at the backend level. This process also contributes to logging and monitoring, enabling the system to track repeated attempts and enforce further security measures such as IP blocking.

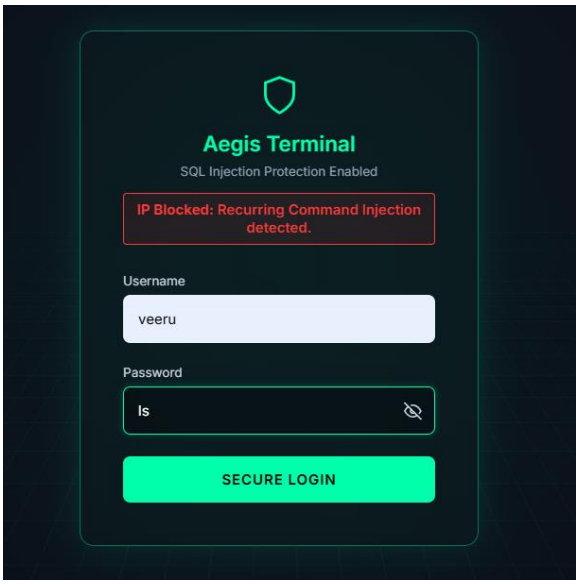

Figure 8: Detecting CMDI Attack

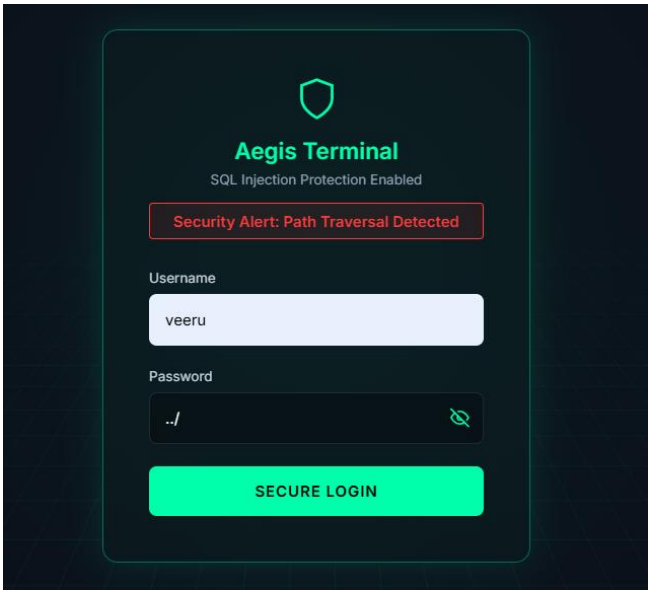

Figure 9: Detecting Path Traversal (LFI) Attack

Figure 10 illustrates the detection and mitigation of Server-Side Template Injection (SSTI) attacks, where inputs such as {{77}}* are used to exploit template engines by executing embedded expressions. It depicts how the system identifies template-specific syntax patterns, including

expression delimiters and evaluation constructs, and flags them as malicious inputs. The detection process involves analyzing input structure and identifying abnormal patterns that deviate from expected user data formats. The figure shows that such payloads are intercepted and blocked before being processed by the server, preventing any execution of injected template code. Each detected SSTI attempt is recorded in the system logs along with relevant details for auditing and analysis. The system also classifies the input explicitly as an SSTI attack and generates a security alert to notify administrators or monitoring modules.

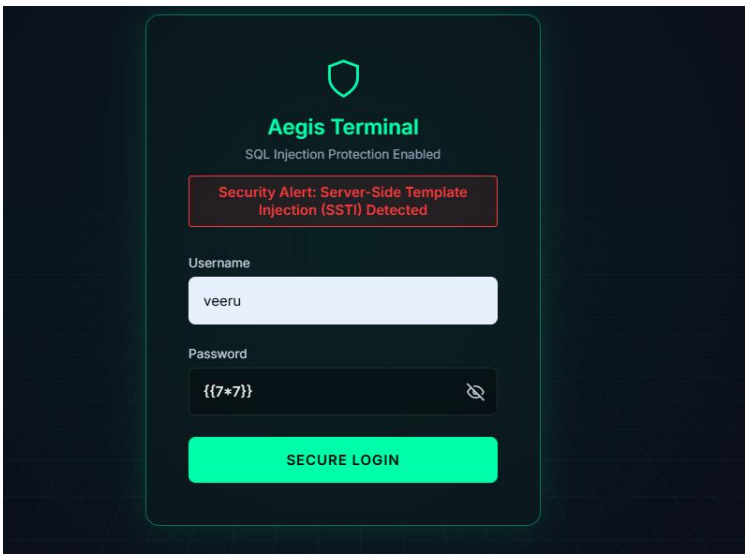

Figure 10: Detecting SSTI Attack

Table 2 presents the overall performance comparison of the four models using accuracy, precision, recall, and F1-score. LRC and RF achieve the highest accuracy of 89.23%, indicating strong classification capability. Logistic Regression records a slightly higher F1-score of 89.54% compared to 89.26% for RF, showing better balance between precision and recall. MNB achieves an accuracy of 86.15% with an F1-score of 85.60%, reflecting moderate performance. GNB shows the lowest accuracy of 84.62% and F1-score of 83.85%, indicating comparatively weaker performance. Precision values are highest for RF at 91.98%, followed by LRC at 91.29%. Recall remains consistent with accuracy across all models, indicating stable predictions.

Table 1: Overall Model Performance

Model	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
GNB	84.62	87.96	84.62	83.85
MNB	86.15	90.30	86.15	85.60
LRC	89.23	91.29	89.23	89.54
RF	89.23	91.98	89.23	89.26

5. CONCLUSION

The research presents a comprehensive framework for real-time detection of web-based attacks in authentication systems by integrating multiple detection mechanisms. The system effectively identifies various attack types such as SQLi, XSS, CMDI, LFI, and SSTI through structured input analysis. The evaluation results demonstrate improved classification performance across models including LRC, RF, GNB, and MNB, with LRC achieving higher consistency and accuracy compared to other models. Confusion matrix analysis shows strong diagonal values, indicating correct classification rates reaching up to 10–11 instances in categories such as SQLi and SSTI, with minimal misclassifications. The system is implemented using Django for handling request-response flow and application logic, SQL for efficient data storage and retrieval, and bcrypt for secure password hashing and authentication. Additionally, the integration of threshold-based blocking and real-time logging enhances system responsiveness and security enforcement. Compared to traditional manual approaches, the system provides better accuracy, faster detection, improved scalability, and stronger security mechanisms.

REFERENCES

[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] Kumara, S. (2026, February). A Lightweight Deep Learning Based Classification Models for Non-Human Identity Threat Detection. In 2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC) (pp. 1-6). IEEE.

[12] Shahid J, Hameed MK, Javed IT, Qureshi KN, Ali M, Crespi N. A Comparative Study of Web Application Security Parameters: Current Trends and Future Directions. Applied Sciences. 2022; 12(8):4077. https://doi.org/10.3390/app12084077

[13] Altulaihan EA, Alismail A, Frikha M. A Survey on Web Application Penetration Testing. Electronics. 2023; 12(5):1229. https://doi.org/10.3390/electronics12051229

[14] Moreno AC, Hernandez-Suarez A, Sanchez-Perez G, Toscano-Medina LK, Perez-Meana H, Portillo-Portillo J, Olivares-Mercado J, García Villalba LJ. Analysis of Autonomous Penetration Testing Through Reinforcement Learning and Recommender Systems. Sensors. 2025; 25(1):211. https://doi.org/10.3390/s25010211

[15] Sarker KU, Yunus F, Deraman A. Penetration Taxonomy: A Systematic Review on the Penetration Process, Framework, Standards, Tools, and Scoring Methods. Sustainability. 2023; 15(13):10471. https://doi.org/10.3390/su151310471.