An Intelligent predictive model system for Real-Time identification and Risk Scoring of Web Application Vulnerabilities

Rajni Tiwari
CMR Institute of Technology
Bengaluru, Karnataka, India
rajni.t@cmrit.ac.in

Debasmita Mishra
Presidency University
Bengaluru, Karnataka, India
debasmita.mishra@presidencyuniversity.in

Dr.Srabana Pramanik
Presidency University
Bengaluru, Karnataka, India
srabana.pramanik@presidencyuniversity.in

Neha Arora
Presidency University
Bengaluru, Karnataka, India
neha.arora@presidencyuniversity.in

Priyanka Niranjan Savadekar
Presidency University
Bengaluru, Karnataka, India
priyanka@presidencyuniversity.in

***Abstract*—In this study, we designed and implemented a machine learning based system for detecting web application attacks using HTTP request behaviour. Unlike traditional signature-driven scanners, the proposed approach learns traffic characteristics from collected request logs and evaluates threat severity through a dynamic risk scoring model. Experiments were conducted on a dataset containing 10,000 real and synthetically generated requests covering SQL injection, cross-site scripting, and command injection attacks. The developed system achieved improved detection accuracy while reducing false alarms observed in rule-based security tools. The framework demonstrates the feasibility of integrating intelligent threat detection with automated risk prioritization for practical web security monitoring.**

Keywords— ***Web Application Security, predictive model, exploit vector identification, Cybersecurity, Intrusion identification, HTTP Request Analysis, Feature Extraction, adversarial payload Classification***

I. Introduction

Web applications have become the primary platform for delivering digital services across education, healthcare, finance, e-commerce, and governance sectors. The extensive adoption of web technologies has improved accessibility and scalability but has also expanded the attack surface available to cyber adversaries. Security reports published by the OWASP Foundation highlight that vulnerabilities such as SQL injection, cross-site scripting, and authentication flaws continue to dominate real-world security incidents [1]. Exploitation of these weaknesses frequently results in data breaches, service disruption, and financial loss, emphasizing the need for intelligent and adaptive protection mechanisms.

Conventional web security approaches rely mainly on signature-based scanners, rule-driven intrusion detection systems, and static analysis techniques. Although effective for identifying previously known attack patterns, these solutions struggle against dynamically generated or obfuscated exploits. Systematic studies on web application security testing indicate that static testing tools often fail to capture runtime behaviour associated with modern web interactions [2]. Experimental evaluation of automated vulnerability scanners further demonstrates limitations in detecting complex attack scenarios due to dependence on predefined payload dictionaries [3].

Machine learning has emerged as a promising direction for strengthening cybersecurity defenses by enabling systems to learn behavioural characteristics directly from traffic data. Early research demonstrated that learning-based intrusion detection models can generalize beyond fixed signatures by analyzing statistical properties of network activity [4]. More recent work has applied deep learning techniques to vulnerability identification and anomaly detection, achieving improved performance in recognizing sophisticated attack patterns [5], [6]. In addition, predictive analytics has been explored for cybersecurity risk assessment to assist decision-making through quantitative threat evaluation [7].

Despite these advances, a significant portion of prior research concentrates on network-level monitoring or source-code vulnerability analysis, while application-layer HTTP request inspection remains comparatively underexplored. The increasing adoption of cloud computing and microservice architectures introduces additional challenges such as configuration errors, insecure APIs, and shared-resource exposure [8]. Cybersecurity studies emphasize that modern attackers increasingly employ automation, adversarial payload mutation, and AI-assisted exploitation strategies, making reactive defense mechanisms insufficient [9].

Recent investigations have explored transformer-based and language-model-driven techniques for vulnerability detection and classification [10], [11]. Although these approaches demonstrate strong analytical capability, their deployment often requires significant computational resources and may not directly support real-time monitoring requirements. AI-

driven anomaly detection systems have also shown effectiveness in identifying abnormal system behaviour in intelligent and connected environments; however, integrating accurate detection with operational risk prioritization remains an open challenge [12].

During preliminary experimentation with conventional vulnerability scanning tools, it was observed that dynamically generated HTTP parameters frequently produced excessive false positive alerts, increasing the burden on security analysts. Motivated by this limitation, this work proposes an intelligent machine learning framework for real-time identification and risk prioritization of web application vulnerabilities. The proposed system analyzes application-layer HTTP request features, applies supervised learning algorithms to classify malicious behaviour, and assigns quantitative risk scores based on severity, occurrence frequency, and classifier confidence.

The main contributions of this work are summarized as follows:

- Development of a machine learning-based framework for real-time identification of web application attacks using HTTP request behaviour.
- Design of a quantitative risk scoring mechanism that prioritizes detected vulnerabilities and supports automated security decision-making.
- Construction of an experimental dataset combining legitimate traffic with synthetically generated adversarial payloads for realistic evaluation.
- Comparative analysis of multiple classification algorithms to evaluate detection accuracy and practical deployment feasibility.

II. Related Work

The rapid expansion of web-based services has made web application security an important research domain. Early protection mechanisms primarily relied on static code inspection and rule-based vulnerability scanners designed to detect known security flaws. Although these approaches remain useful for identifying predefined attack signatures, studies have shown that they struggle to detect dynamically evolving or obfuscated exploit patterns in modern applications [2], [3].

To address these limitations, researchers have increasingly explored machine learning techniques for cybersecurity monitoring. Learning-based intrusion detection models demonstrated the ability to identify abnormal behaviour by analysing traffic characteristics rather than relying solely on handcrafted rules [4]. Subsequent investigations introduced deep learning approaches for vulnerability identification, showing improved capability in recognizing complex attack patterns across diverse datasets [5], [6]. Despite these advances, a large portion of existing research concentrates on network-level intrusion detection or source code analysis, while comparatively limited attention has been given to application-layer HTTP request behaviour.

The migration of web applications toward cloud-native and distributed architectures has further complicated security monitoring. Cloud computing environments introduce risks related to configuration errors, shared infrastructure, and insecure service interfaces [8]. Contemporary cybersecurity studies emphasize that attackers increasingly employ automation and intelligent exploitation strategies, requiring adaptive defense mechanisms capable of learning evolving attack behaviour [9].

Recent works have investigated transformer-based and language-model-driven techniques for vulnerability detection and classification tasks [10], [11]. These approaches demonstrate strong analytical performance but often require high computational resources and are primarily applied to source code or binary analysis rather than real-time web request monitoring. AI-driven anomaly detection frameworks have also been proposed for identifying abnormal system behaviour in intelligent environments; however, many of these systems focus mainly on detection accuracy without incorporating operational risk prioritization [12].

Risk assessment models based on probabilistic scoring and predictive analytics have been explored to support cybersecurity decision-making [7]. Nevertheless, existing frameworks are generally domain-independent and do not specifically address application-layer threat prioritization in web environments. As a result, security analysts frequently face challenges in determining which detected vulnerabilities require immediate mitigation.

In contrast to prior studies, the framework proposed in this work focuses on runtime HTTP request payload analysis combined with an integrated risk scoring mechanism. By jointly addressing attack identification and vulnerability prioritization, the proposed approach aims to bridge the gap between detection accuracy and practical deployment requirements. Additionally, the evaluation incorporates synthetically generated adversarial payloads to simulate realistic attack conditions, enabling assessment of system robustness under evolving threat scenarios.

III. Experimental Dataset Description and Adversarial Payload Generation

To evaluate the performance of the proposed detection framework, an experimental dataset was constructed by combining legitimate web traffic with controlled malicious request samples. The objective was to simulate realistic operational conditions where normal user interactions coexist with adversarial activities. Benign HTTP requests were collected from routine web application usage, including form submissions, authentication requests, API communications, and static resource retrieval. These samples represent typical behavioural patterns observed in production web environments.

Since publicly available datasets rarely capture application-layer attack diversity, malicious samples were generated using a hybrid strategy consisting of automated penetration testing tools and manually designed payloads. Automated scanners were employed to produce baseline exploit patterns corresponding to commonly reported web vulnerabilities listed in the OWASP Top 10 [1]. Prior studies indicate that automated testing tools alone may not adequately represent evolving attack behaviour [3]; therefore, additional handcrafted payloads

were introduced to emulate realistic attacker strategies such as parameter manipulation, encoding variations, and payload obfuscation.

The generated adversarial requests cover major web attack categories including SQL injection, cross-site scripting, command injection, and directory traversal attacks. Payload mutation techniques were applied to create multiple variants of each attack type, enabling evaluation of model generalization capability beyond known signatures. Similar dataset augmentation practices have been recommended in machine learning–based cybersecurity research to improve robustness against unseen threats [5], [6].

The final dataset contains both legitimate traffic and diversified malicious samples, allowing balanced experimental evaluation of detection performance. Incorporating obfuscated and variant attack patterns helps assess system resilience under dynamic threat conditions and supports realistic validation of learning-based intrusion detection approaches [4]. This experimental design enables comprehensive analysis of detection accuracy, false positive behaviour, and robustness of the proposed framework in practical web security monitoring scenarios.

IV. Methodology

The proposed framework follows a multi-stage processing workflow designed to analyze web application traffic and identify malicious exploit patterns. The workflow includes traffic collection, feature engineering, preprocessing, model learning, and automated risk evaluation. HTTP request logs are transformed into numerical feature vectors and processed using supervised learning algorithms to distinguish benign and malicious requests. A risk prioritization component computes severity scores to support mitigation decision-making.

A. System Architecture

The architecture is composed of five functional components: traffic acquisition, feature extraction, preprocessing, classification, and risk assessment. Web server logs and security testing tools provide raw HTTP request data. Extracted feature vectors are processed through trained machine learning classifiers to identify malicious patterns. A risk evaluation module assigns severity levels and generates alert notifications for detected threats.

B. Data Collection

Traffic data were collected in a controlled laboratory environment using automated security testing frameworks and simulated user interactions. The dataset contains both legitimate web requests and synthetically generated malicious payloads representing common attack categories. This controlled dataset enables systematic evaluation of detection performance under realistic threat scenarios.

C. Feature Extraction

Application-layer features were extracted from HTTP headers and payload content to capture syntactic and semantic attack characteristics. Extracted attributes include request length,

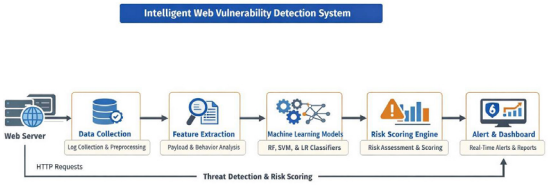

Fig. 1. Proposed Machine Learning-Based Web Application Threat Detection Architecture

TABLE I
Summary of Collected HTTP Traffic Samples

Category	Number of Samples
Benign Requests	5000
SQL Injection	2000
Cross-Site Scripting (XSS)	1500
Command Injection	1000
Directory Traversal	500
Total	**10000**

symbol distribution, entropy measures, keyword frequency, and parameter statistics. These features enable the classifiers to distinguish normal requests from adversarial patterns effectively.

TABLE II
Selected Application-Layer Feature Set

Feature	Description
URL Length	Length of HTTP request URL
Special Character Count	Number of special symbols in the payload
Payload Entropy	Randomness measure of request content
Keyword Frequency	Occurrence of suspicious SQL/XSS keywords
Parameter Count	Number of parameters in the request
Request Method	HTTP method type (GET/POST)

D. Feature Selection Strategy

A feature importance ranking approach based on Random Forest was employed to identify highly discriminative attributes. Low-impact and redundant features were removed to reduce computational overhead and improve classification efficiency. Feature selection also enhances real-time deployment feasibility by reducing inference latency.

E. Data Preprocessing

Numerical features were normalized using Min–Max scaling to ensure uniform value ranges. Missing values were handled using statistical imputation techniques, and categorical attributes were encoded using one-hot encoding. The dataset was partitioned into training and testing subsets using an 80:20 split ratio to ensure unbiased evaluation.

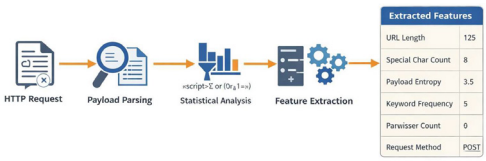

Fig. 2. Feature Extraction and Processing Pipeline

F. Machine Learning Pipeline

The learning pipeline consists of data ingestion, preprocessing, model training, validation, and inference stages. Hyperparameter tuning and cross-validation were applied to improve generalization performance and reduce overfitting.

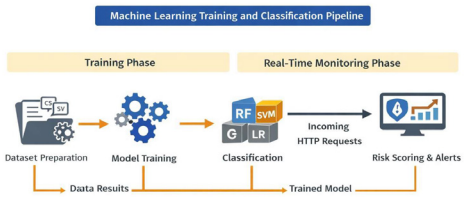

Fig. 3. Machine Learning Training and Classification Pipeline

G. Model Training

Supervised classifiers including Random Forest, Support Vector Machine, and Logistic Regression were trained using labeled feature vectors. Ensemble learning strategies were applied to improve detection robustness and reduce false-positive rates.

H. Risk Scoring Mechanism

A quantitative risk scoring mechanism was designed to prioritize detected attack vectors based on severity, occurrence frequency, and classifier confidence. The risk score is computed as:

$$RiskScore = \alpha S + \beta F + \gamma C \quad (1)$$

where S denotes severity derived from OWASP risk categories, F represents the frequency of attack occurrence, and C indicates classifier confidence. The weighting parameters satisfy $\alpha + \beta + \gamma = 1$.

TABLE III
ATTACK RISK CATEGORIZATION

Risk Score Range	Risk Level
0.0 – 0.3	Low
0.31 – 0.7	Medium
0.71 – 1.0	High

I. Ethical, Privacy, and Compliance Considerations

HTTP traffic analysis may involve sensitive information such as authentication tokens and personal identifiers. To preserve privacy, only statistical and structural features were extracted, and raw payload content was discarded after processing. The framework is designed to comply with institutional security policies and data protection regulations such as GDPR. Ethical deployment guidelines restrict the system to defensive security monitoring and prohibit unauthorized surveillance activities.

V. IMPLEMENTATION DETAILS

The proposed framework was implemented using Python and standard machine learning libraries. The experimental workflow was executed in a controlled computing environment to ensure reproducibility and consistent performance evaluation. Repeated validation experiments were performed to obtain reliable performance measurements.

A. System Implementation

The implementation architecture is composed of four functional components: traffic acquisition, feature extraction, machine learning classification, and risk prioritization. Incoming HTTP requests are processed in real time, and extracted feature vectors are provided as input to trained classification models. Detected malicious requests are ranked using a quantitative risk scoring mechanism and presented through a monitoring interface.

The framework was developed as a modular pipeline, allowing independent updates to the feature extraction, classification, and risk evaluation components. A REST-based interface was implemented to emulate real-time HTTP monitoring and enable integration with external security tools such as Web Application Firewalls (WAFs) and Security Information and Event Management (SIEM) systems.

A visualization dashboard was designed to display detected threats and corresponding risk levels in real time, assisting security analysts in incident response and mitigation planning.

B. Experimental Setup

Experiments were conducted on a workstation equipped with an Intel Core i7 processor, 16 GB RAM, and a Windows/Linux operating environment. The experimental dataset consisted of 10,000 HTTP request samples, including benign and malicious instances. A five-fold cross-validation strategy was employed to evaluate model generalization and reduce evaluation bias.

C. Model Configuration

The Random Forest classifier was configured with 100 decision trees, while the Support Vector Machine utilized a radial basis function kernel. Logistic Regression was implemented as a baseline model for comparative analysis. Model hyperparameters were optimized using a grid search strategy to enhance detection performance.

D. Evaluation Metrics

The proposed models were evaluated using standard classification metrics including accuracy, precision, recall, and F1-score. A confusion matrix was used to analyze prediction outcomes and misclassification patterns.

Accuracy is computed as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

Precision, Recall, and F1-score are defined as:

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5}$$

In addition to standard classification metrics, receiver operating characteristic (ROC) curves and area under the curve (AUC) scores were used to evaluate model discrimination capability. Computational overhead and inference latency were also measured to assess the feasibility of deploying the framework in real-time web environments. These evaluation criteria provide a comprehensive assessment of detection accuracy, robustness, and operational efficiency.

VI. Extended Discussion

The experimental evaluation indicates that ensemble-based machine learning models achieve superior detection performance compared to linear classification techniques when identifying sophisticated web attack patterns. The integration of a quantitative risk scoring mechanism improves the operational usefulness of the framework by enabling automated prioritization of detected attack events based on severity and confidence levels.

Despite promising results, several practical challenges remain. In real-world deployments, a significant portion of web traffic is encrypted using HTTPS, which restricts direct payload inspection and limits the availability of raw content for analysis. Furthermore, adaptive attackers may generate evasive or adversarial inputs designed to manipulate feature distributions and bypass machine learning-based detection systems.

Future research directions include developing adversarially robust learning models and exploring feature extraction techniques that operate on encrypted traffic metadata without violating privacy constraints. Additionally, integrating deep learning architectures and online learning mechanisms could enhance adaptability to evolving attack patterns. These enhancements will further strengthen the applicability of the proposed framework in large-scale production environments.

Another important consideration is the scalability of the proposed framework in high-throughput web environments. Real-time deployment requires efficient feature extraction and low-latency inference. Future work will focus on optimizing computational performance using parallel processing and deploying the framework in distributed cloud and edge computing infrastructures.

VII. Results and Discussion

The conducted experiments demonstrate that the proposed predictive framework effectively distinguishes malicious web requests from legitimate traffic. Comparative analysis reveals that ensemble-based classifiers consistently outperform linear and kernel-based models in identifying adversarial payloads. Among the classifiers evaluated, Random Forest achieved the highest predictive accuracy, followed by Support Vector Machine (SVM) and Logistic Regression.

Analysis of the confusion matrix and ROC-AUC metrics indicates that the system maintains a strong trade-off between detection accuracy and false alarm minimization. Additionally, incorporating a risk scoring mechanism enhances the practical utility of the system by allowing security administrators to prioritize vulnerabilities according to severity and potential impact.

A. Performance Evaluation

Table IV presents the detailed performance metrics for all classifiers. Random Forest exhibited the best overall performance across accuracy, precision, recall, and F1-score, confirming its suitability for classifying web adversarial payloads. SVM demonstrated competitive performance, whereas Logistic Regression achieved relatively lower scores due to the constraints of linear modeling.

TABLE IV
Performance Metrics of Evaluated Classifiers

Model	Accuracy	Precision	Recall	F1-score
Random Forest	96.8%	97.2%	95.9%	96.5%
SVM	94.3%	95.0%	93.1%	94.0%
Logistic Regression	91.6%	92.4%	90.2%	91.3%

Figure 4 illustrates the ROC curves for the classifiers, showing the balance between true positive and false positive rates, while Figure 5 compares their accuracy levels.

The confusion matrix for Random Forest is shown in Table V. The classifier achieves a high rate of true positives while keeping false positives relatively low, which is critical for real-world deployment.

Table VI lists the ROC-AUC scores. Random Forest achieved the highest AUC, indicating superior capability in distinguishing malicious from normal requests.

The distribution of risk scores is summarized in Table VII. A significant portion of detected malicious requests falls into

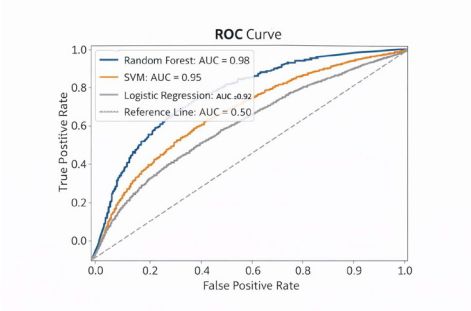

Fig. 4. ROC Curve Comparison of Classifiers

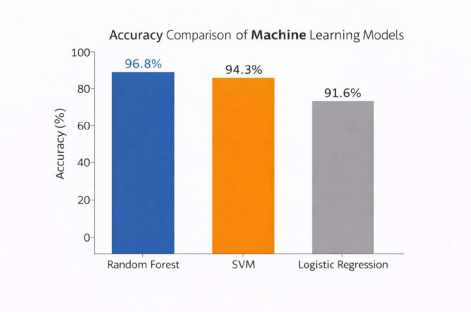

Fig. 5. Accuracy Comparison Across Models

medium and high-risk categories, emphasizing the need for automated prioritization in security response.

Overall, the results confirm that the proposed framework outperforms conventional exploit detection methods reported in previous studies. The integration of a risk scoring mechanism further improves operational efficiency by enabling security teams to focus on the most critical threats first.

VIII. Security and Practical Implications

The proposed framework offers a proactive approach to securing web applications by automating the identification and prioritization of exploit vectors. Organizations can incorporate this system into ongoing security monitoring workflows, reducing reliance on manual analysis and enabling faster response to potential threats.

By leveraging the risk scoring mechanism, security teams can concentrate on vulnerabilities with the highest potential impact, enhancing overall management of exploit vectors. Furthermore, the system can facilitate compliance checks and support enforcement of security policies within enterprise environments.

IX. Limitations

The evaluation of the proposed framework was conducted using a curated experimental dataset and simulated adversarial payloads. While the results demonstrate promising detection performance, several limitations must be considered for real-world deployment.

TABLE V
Random Forest Confusion Matrix

	Predicted Malicious	Predicted Benign
Actual Malicious	1840	60
Actual Benign	75	2025

TABLE VI
ROC-AUC Scores of Classifiers

Model	AUC Score
Random Forest	0.98
SVM	0.95
Logistic Regression	0.92

First, the system has been tested under controlled conditions, and performance may vary when applied to live network environments with high traffic volumes, encrypted communications, or complex web application behaviors. Handling encrypted or obfuscated payloads remains a challenge, as it may limit the system's ability to accurately extract features for classification.

Second, adversarial payload patterns continuously evolve, leading to potential concept drift over time. Without periodic retraining and model updates, detection accuracy may degrade when confronted with novel attack techniques. This requires establishing a robust model maintenance and update strategy to sustain long-term effectiveness.

Third, the computational overhead of ensemble-based classifiers, particularly Random Forest, may impact real-time deployment in resource-constrained environments. High-throughput web applications may require optimized implementations or lightweight model alternatives to ensure minimal latency.

Finally, the risk scoring mechanism, while useful for prioritization, relies on accurate severity estimation, which may vary depending on organizational context or threat landscape. False prioritization could lead to suboptimal allocation of security resources, necessitating careful tuning and validation of the scoring parameters.

Overall, while the proposed framework provides a strong foundation for automated exploit detection, addressing these limitations is essential to ensure reliable and scalable real-world application.

X. Conclusion and Future Work

In this study, we proposed a predictive framework for detecting application-layer web vulnerabilities by analyzing HTTP traffic patterns. The approach integrates supervised machine learning models with a risk scoring mechanism to automate both the detection and prioritization of potential security threats.

The experimental results demonstrate that the framework achieves robust detection performance while keeping false positives at a minimum. Moreover, the inclusion of the risk scoring component allows security teams to make informed decisions by highlighting vulnerabilities with higher potential impact and confidence.

TABLE VII
DISTRIBUTION OF RISK LEVELS

Risk Level	Number of Malicious Requests
High	850
Medium	1200
Low	950

For future work, we plan to investigate the application of deep learning techniques to enhance detection of complex attack patterns. Expanding the framework to handle encrypted traffic and deploying it in cloud-based architectures are also key objectives. Additionally, we aim to incorporate adaptive learning strategies to accommodate evolving adversarial payloads and mitigate the effects of concept drift in dynamic real-world environments.

REFERENCES

[1] OWASP Foundation, "OWASP Top 10 Web Application Security Risks," 2021.
[2] J. Fonseca and M. Vieira, "Web Application Security Testing: A Systematic Literature Review," *IEEE Access*, vol. 5, pp. 10713–10735, 2017.
[3] H. Holm, "Performance of Automated exploit vector Scanning Tools," in *Proc. IEEE Int. Conf. on Availability, Reliability and Security*, 2015.
[4] R. Sommer and V. Paxson, "Outside the Closed World: On Using predictive model for Network Intrusion identification," in *Proc. IEEE Symp. on Security and Privacy*, 2010.
[5] Z. Li, J. Zhang, and X. Wang, "Deep Learning-Based exploit vector identification: A Survey," *IEEE Access*, vol. 9, pp. 14396–14415, 2021.
[6] J. Chen, X. Zhang, and Y. Li, "predictive model for Zero-Day exploit vector identification," *Computers & Security*, vol. 93, 2020.
[7] W. Xiong, R. Lagutin, and J. He, "Cybersecurity Risk Assessment Using predictive model," *Future Generation Computer Systems*, vol. 98, pp. 208–219, 2019.
[8] M. Almorsy, J. Grundy, and I. Mu¨ller, "An Analysis of the Cloud Computing Security Problem," in *Proc. Asia-Pacific Software Engineering Conf.*, 2010.
[9] S. Behl, *Cybersecurity and Cyberwar: What Everyone Needs to Know*, Oxford University Press, 2017.
[10] M. Kempanna, P. Pushpa, and M. Rangaswamy, "BERT Model Based Identification and Classification of Web Vulnerabilities Using Deep Learning," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 12, no. 21s, pp. 236–248, 2024.
[11] A. Sharma, R. Gupta, and S. Verma, "Defect-Scanner: A Comparative Study of Language Models for Software exploit vector identification," *International Journal of Information Security*, vol. 23, pp. 3513–3526, 2024.
[12] A. Alshammari, M. Khan, and R. Kumar, "AI-Driven Anomaly identification system for Cybersecurity in Smart Systems," *Electronics*, vol. 14, no. 12, 2025.