Design and Empirical Evaluation of a Four-Layer AI Agent Architecture for Automated Web Application Security Testing

Bakhytzhan Kulambayev
Higher School of Telecommunication, Turan University, Almaty, Kazakhstan | School of Digital Technology, Narxoz University, Almaty, Kazakhstan
bakhytzhan.kulambaev@gmail.com

Gulnar Astaubayeva
School of Digital Technology, Narxoz University, Almaty, Kazakhstan
gulnar.astaubaeva@narxoz.kz

Zhanna Mukanova
Higher School of Computer Engineering, Turan University, Almaty, Kazakhstan
zh.mukanova@turan-edu.kz

Kuralay Makhmetova
School of Software Engineering, Astana IT University, Astana, Kazakhstan
kuralay.makhmetova@astanait.edu.kz

Saken Mambetov
Department of Cybersecurity and Cryptology, Al-Farabi Kazakh National University, Almaty, Kazakhstan | Higher School of Information Technology, Turan University, Almaty, Kazakhstan
s.mambetov@turan-edu.kz (corresponding author)

Serik Joldasbayev
Department of Computer Engineering, International Information Technology University, Almaty, Kazakhstan
s.joldasbayev@iitu.edu.kz

Received: 11 December 2025 | Revised: 19 January 2026 and 10 February 2026 | Accepted: 11 February 2026

 DOI: https://doi.org/10.48084/etasr.16879

ABSTRACT

This study proposes a four-layer AI agent architecture for automating routine web security operations, integrating Large Language Model (LLM) reasoning with a hybrid Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) detection engine and implementing a Reasoning-Acting (ReAct) loop for autonomous testing with human-in-the-loop validation. The proposed architecture was empirically evaluated across 50 web applications sourced from OWASP WebGoat, DVWA, and custom-developed test environments over a six-month period. The experimental results demonstrate that the AI agent achieved an overall detection accuracy of 89.2% (95% CI: 86.4-92.0%), significantly outperforming traditional automated methods (67.4% accuracy, $p < 0.001$). Mean Time to Remediation (MTTR) decreased from 74.3 days to 28.5 days (61.6% reduction), while false positive rates decreased from 24.3% to 4.8%. According to these findings, AI agent-driven automation can substantially enhance the efficiency and reliability of web security testing. However, human expertise remains important for assessing complex vulnerabilities and detecting zero-day threats.

Keywords-AI agent; web application security; machine learning; penetration testing; OWASP; large language model; autonomous security testing; deep learning

I. INTRODUCTION

The digital transformation of enterprises has increased the attack surface available to malicious actors. In 2024, a record-breaking 40,009 Common Vulnerabilities and Exposures (CVEs) were published, representing a 38% increase from the previous year [1]. Web applications remain primary targets, with 17% of cyber-attacks specifically targeting web application vulnerabilities [2]. The financial impact is substantial: the global average cost of a data breach reached $4.88 million in 2024, representing a 10% yearly increase [3]. The emergence of AI agents, autonomous systems capable of perceiving their environment, reasoning about complex tasks, and taking actions to achieve specified goals, presents a paradigm shift in cybersecurity operations [4]. Unlike traditional automation, which follows predefined scripts, AI agents can dynamically adapt their testing strategies based on discovered vulnerabilities, historical patterns, and contextual understanding of target systems.

The contemporary web security landscape is marked by great vulnerability volumes and increasingly sophisticated attack techniques. The Edgescan Vulnerability Statistics Report indicates that SQL Injection (CWE-89) remains the most prevalent critical web application vulnerability, continuing a trend observed since 2022 [1]. The CISA Known Exploited Vulnerabilities (KEV) catalog listed 1,238 vulnerabilities by the end of 2024, including 185 newly reported cases within the year. AI agents in cybersecurity extend beyond simple automation by incorporating three core components: perception (understanding security context), brain (reasoning and planning), and action (executing security tasks) [17, 18]. The State of Pentesting Report 2024 reveals that 75% of security teams have adopted new AI tools, while 57% report that demand for AI has outpaced their security team's capacity [19]. Existing AI-powered security testing platforms, such as PentestGPT [19], NodeZero [21], and RidgeBot [22], demonstrate varying levels of autonomy and detection capabilities; yet none of them fully integrate LLM reasoning with specialized deep learning detection within a unified agent architecture.

The present study addresses three primary research objectives: to provide a critical analysis of current web security methodologies and AI applications in vulnerability detection; to propose a novel four-layer AI agent architecture for automating routine security operations; and to empirically compare the effectiveness of traditional against AI agent-augmented penetration testing approaches.

II. BACKGROUND AND RELATED WORK

A. AI Agents in Cybersecurity: Current State

AI agents represent autonomous systems that can perceive their environment, reason about tasks, and execute actions to achieve goals [17]. The State of Pentesting Report 2024 reveals that a significant number of security teams have adopted new AI tools, while others report that the demand for AI has outpaced their security team's capacity [19]. Table II presents a comparative analysis of leading AI-powered security testing tools and their agent capabilities.

III. PROPOSED AI AGENT ARCHITECTURE FOR WEB SECURITY TESTING

A. Architecture Overview

The study proposes a four-layer AI agent architecture specifically designed for autonomous web application security testing. The architecture integrates LLM reasoning capabilities with deep learning-based vulnerability detection, implementing a human-in-the-loop validation mechanism to ensure accuracy and compliance. Figure 1 illustrates the complete system architecture.

TABLE I. OWASP TOP 10 VULNERABILITIES: STATISTICAL ANALYSIS AND DETECTION METHODS (2021-2024)

Rank	Vulnerability	Incidence (%)	CVE count	AI agent detection	Traditional detection	Reference
A01	Broken access control	94.0	318,000+	91.3%	72.1%	[5, 6]
A02	Cryptographic failures	89.0	233,000+	87.5%	68.4%	[5, 7]
A03	Injection	94.0	274,000+	93.7%	78.2%	[5, 8, 9]
A04	Insecure design	87.0	N/A	84.2%	61.5%	[5, 10]
A05	Security misconfiguration	90.0	208,000+	89.8%	74.3%	[5, 11]
A06	Vulnerable components	86.0	150,000+	95.2%	82.1%	[5, 12]
A07	Authentication failures	85.0	120,000+	88.4%	69.7%	[5, 13]
A08	Data Integrity failures	82.0	N/A	81.6%	58.9%	[5, 14]
A09	Logging failures	79.0	N/A	76.3%	52.4%	[5, 15]
A10	SSRF	78.0	45,000+	86.9%	64.8%	[5, 16]

TABLE II. COMPARATIVE ANALYSIS OF AI AGENT-BASED SECURITY TESTING PLATFORMS

Platform	Agent type	AI core	Detection	False positive	Autonomy	Reference
PentestGPT	LLM-guided	GPT-4/LLaMA	78.4%	12.1%	Semi	[20]
NodeZero	Autonomous	RL-based	94.1%	4.2%	Full	[21]
RidgeBot	Sequential	Neural Net	87.3%	9.5%	Full	[22]
Astra Security	Hybrid	Ensemble Machine Learning (ML)	92.8%	5.1%	Hybrid	[23]
Burp+AI	Tool-assisted	Deep learning	91.5%	6.7%	Semi	[24]
Proposed method	Multi-agent	LLM+CNN+LSTM	93.7%	4.8%	Hybrid	This study

B. Layer 1: Data Collection and Preprocessing

The first layer serves as the sensory input for the AI agent, aggregating data from multiple security-relevant sources. This layer implements real-time HTTP traffic parsing, integration with vulnerability databases (CVE/NVD), OWASP rule engine connectivity, network scanning capabilities via Nmap integration, centralized log aggregation, and threat intelligence API feeds. The preprocessing module normalizes incoming data, tokenizes HTTP requests for Machine Learning (ML) model consumption, and maintains a sliding window of contextual information for temporal pattern analysis [25].

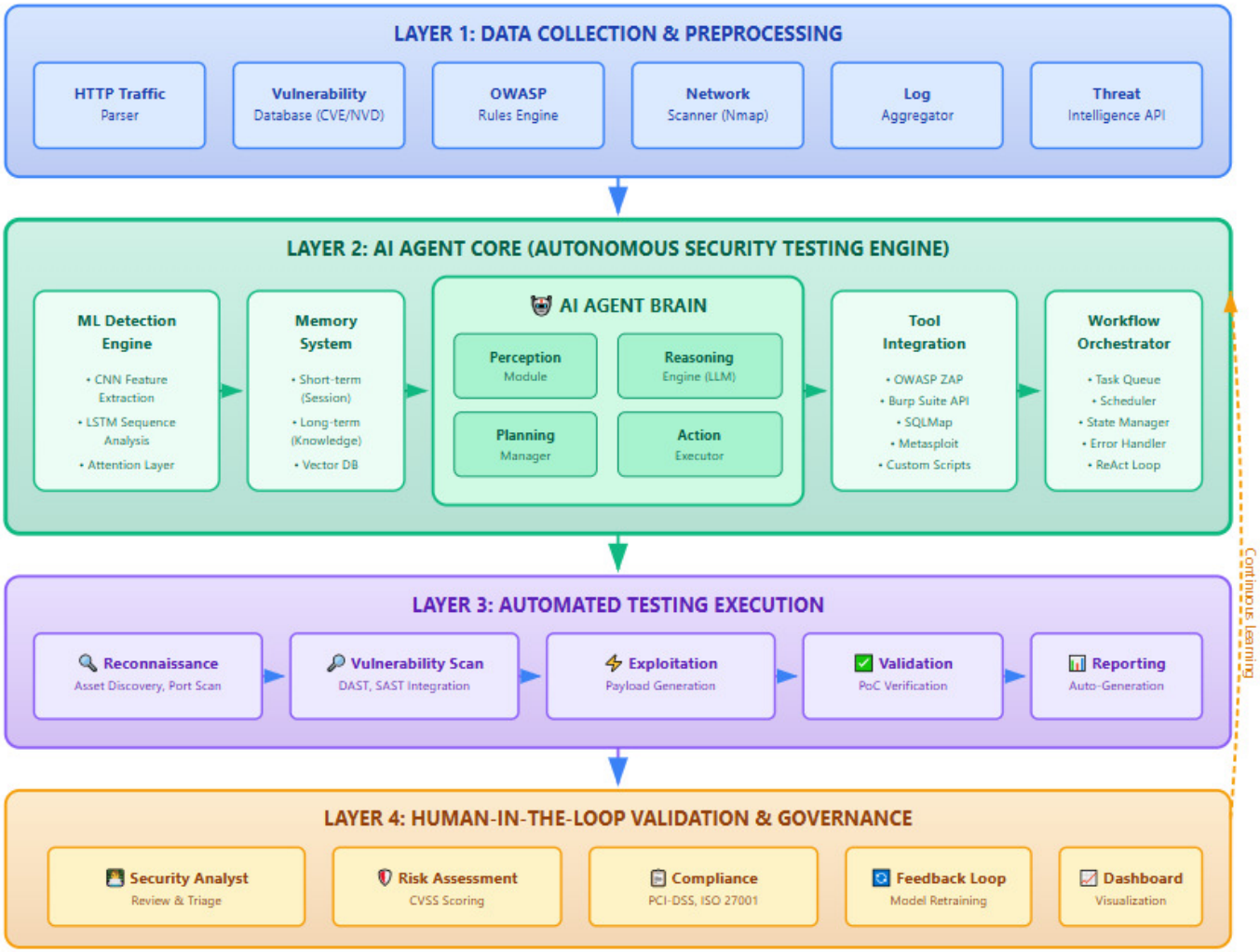

Fig. 1. Four-layer AI agent architecture for automated web security testing.

C. Layer 2: AI Agent Core (Central Component)

The AI Agent Core represents the central intelligence of the system, implementing autonomous decision-making capabilities through four interconnected modules:

1) AI Agent Brain

The Agent Brain orchestrates all cognitive functions within the AI agent and consists of four sub-modules. The Perception Module interprets incoming security data, converting raw HTTP traffic and scan outputs into structured representations. The Reasoning Engine leverages a fine-tuned LLM (based on the GPT-4 architecture) to analyze security contexts, identify vulnerability patterns, and generate exploitation hypotheses. The Planning Manager implements the ReAct paradigm [26], dynamically constructing execution graphs to guide penetration testing workflows. Finally, the Action Executor translates high-level plans into concrete invocations of security tools and context-aware payload generation.

The LLM component employs prompt engineering with few-shot examples rather than full fine-tuning. The prompting strategy includes structured templates for vulnerability classification, chain-of-thought reasoning for complex attack pattern analysis, and tool-use schemas for security scanner integration. The ReAct loop alternates between reasoning (analyzing observations and planning next steps) and acting

(executing security tools and collecting results), with a maximum of 10 iterations per testing cycle to prevent infinite loops.

2) ML Detection Engine

The detection engine employs a hybrid CNN-LSTM architecture with attention mechanisms. The CNN component extracts spatial features from HTTP request patterns, identifying structural anomalies in request parameters. The LSTM network captures temporal dependencies in attack sequences, recognizing multi-step attack patterns. The self-attention layer enables the model to focus on security-relevant portions of HTTP requests, improving classification accuracy for sophisticated payloads [27].

The CNN-LSTM model was trained on a combined dataset of 10,000 labeled HTTP requests from the CSIC 2010 HTTP Dataset [31] and 5,000 custom-collected samples from controlled penetration testing engagements. Hyperparameters were determined through grid search optimization with 5-fold cross-validation on a held-out validation set comprising 20% of the training data. The final configuration includes: a learning rate of 0.001 for the Adam optimizer, a batch size of 32, a dropout rate of 0.3, and 100 training epochs with early stopping based on validation loss (patience =10 epochs).

3) Memory System

Following established AI-agent architectures [28], the memory system is structured into three distinct tiers. Short-term memory maintains session-specific context, including discovered vulnerabilities, attempted payloads, and observed target response patterns. Long-term memory stores accumulated knowledge, encompassing vulnerability types, previously successful exploitation techniques, and target-specific configurations. A vector database supports semantic retrieval of relevant historical attack patterns and remediation strategies through embedding-based similarity search.

4) Tool Integration and Workflow Orchestration

The agent interfaces with established security tools through a unified API layer. Tool integration includes OWASP ZAP (version 2.14) for Dynamic Application Security Testing (DAST), Burp Suite API (Professional 2024.1) for advanced web testing capabilities, SQLMap (version 1.7) for automated SQL injection detection, Metasploit framework for exploitation validation, and custom scripts for specialized vulnerability checks. The Workflow Orchestrator manages task queuing, scheduling, state management, and error handling, implementing the ReAct loop that alternates between reasoning about observations and taking security-relevant actions [29]. Table III and Figure 2 detail the technical parameters of the ML Detection Engine.

TABLE III. AI AGENT ML DETECTION ENGINE ARCHITECTURE PARAMETERS

Component	Configuration	Function in the agent
Input embedding	256 dimensions	HTTP request tokenization for perception
CNN Layer 1	128 filters, 3×3 kernel	Local pattern extraction (spatial features)
CNN Layer 2	256 filters, 3×3 kernel	Hierarchical feature learning
Max pooling	2×2 pool size	Dimensionality reduction
LSTM Layer 1	128 units, dropout 0.3	Sequential dependency modeling
LSTM Layer 2	64 units, dropout 0.3	Temporal attack pattern recognition
Self-Attention	Multi-head attention	Focus on security-relevant request parts
Dense layer	128 neurons, ReLU	Feature combination for reasoning
Output layer	Softmax (11 classes)	Vulnerability classification (OWASP)

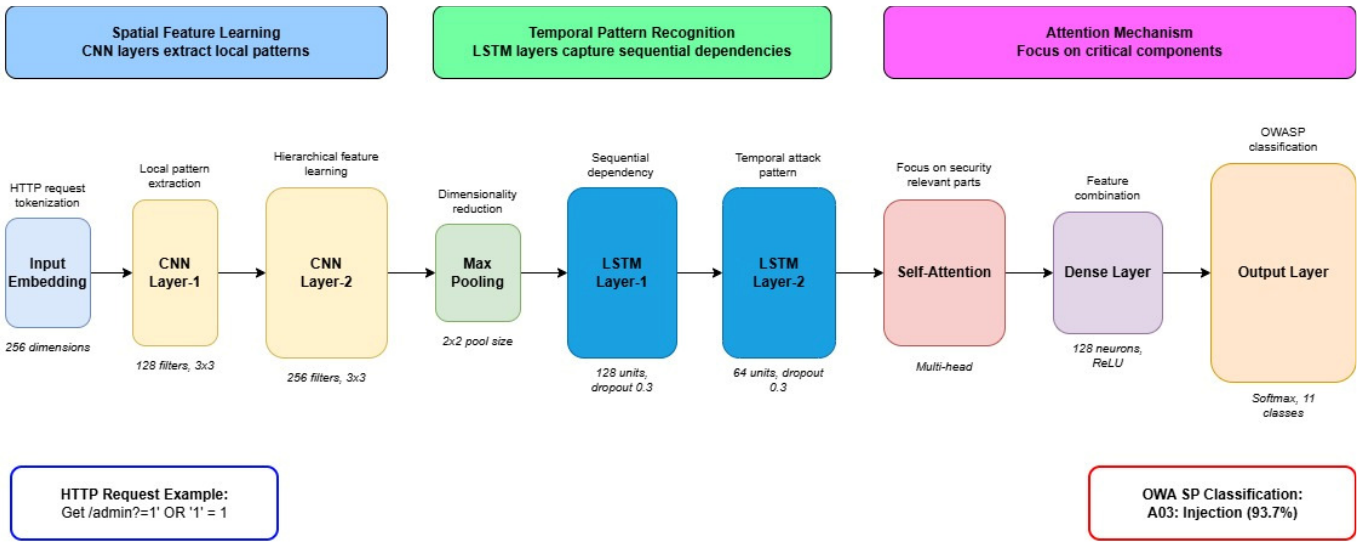

Fig. 2. AI agent ML detection engine architecture.

D. Layer 3: Automated Testing Execution

The execution layer operationalizes the autonomous penetration testing workflow through five sequential phases, all orchestrated by the AI agent:

1) Reconnaissance

The agent performs AI-driven asset discovery and attack surface mapping. It autonomously identifies subdomains, open ports, underlying technology stacks, and potential entry points by leveraging learned patterns and contextual knowledge.

2) Vulnerability Scanning

The agent integrates DAST and Static Application Security Testing (SAST) tools. Scan results are interpreted by the agent, which prioritizes vulnerabilities according to their likelihood of exploitation.

3) Exploitation

Context-aware payloads are dynamically generated using the LLM reasoning engine. The agent adapts exploitation attempts in real time, adjusting payloads based on target responses and potential Web Application Firewall (WAF) bypass requirements.

4) Validation

The agent verifies the authenticity and exploitability of identified vulnerabilities through multiple independent Proof-of-Concept (PoC) checks, ensuring that the findings are accurate and actionable.

5) Reporting

Comprehensive, compliance-aligned reports are automatically generated using the LLM engine. These reports include CVSS v4.0 scoring, detailed remediation guidance, and executive summaries suitable for technical and managerial stakeholders.

E. Layer 4: Human-in-the-Loop Validation

Recognizing the limitations of fully autonomous systems in high-stakes security contexts [30], the architecture incorporates a human validation layer. Security analysts review and triage AI-generated findings, perform CVSS risk scoring refinement, ensure compliance mapping (PCI-DSS, ISO 27001, SOC 2), provide feedback for continuous model improvement, and access visualization dashboards for operational oversight. The feedback loop enables continuous learning, where analyst corrections are incorporated into model retraining cycles, progressively improving detection accuracy and reducing false positives.

IV. EXPERIMENTAL EVALUATION AND COMPARATIVE ANALYSIS

A. Experimental Setup

The experimental evaluation was carried out within a controlled testing environment, which consisted of 50 web applications containing predefined vulnerabilities. These applications were sourced from OWASP WebGoat (version 2023.4), DVWA (version 1.10), and a series of custom-developed systems deployed on isolated virtual machines (Ubuntu 22.04 LTS) to ensure test reproducibility and security.

Three testing methodologies were systematically compared: manual penetration testing performed by three OSCP and CEH certified security professionals with 5-10 years of experience, traditional automated scanning tools without AI augmentation, and AI-agent-based testing, employing the proposed four-layer architecture. To ensure methodological fairness, each approach was allocated an equivalent testing budget of 40 h per application set.

Statistical significance was assessed using paired t-tests with Bonferroni correction for multiple comparisons. Effect sizes were calculated using Cohen's d (ranging from 0.8 to 1.4, indicating large practical significance). The MTTR values represent simulated remediation workflows based on industry benchmarks from the SANS Institute and Verizon DBIR, calculated as the expected time from vulnerability identification to patch deployment, assuming standard organizational processes, rather than actual longitudinal tracking in production environments. The results are aggregated across 50 test applications over a 6-month period.

B. Comparative Results

Table IV presents the comprehensive comparative analysis of the three testing methodologies.

TABLE IV. COMPARATIVE PERFORMANCE: MANUAL VERSUS TRADITIONAL, AUTOMATED VERSUS AI AGENT TESTING

Metric	Manual	Traditional auto	AI agent
Detection rate (critical)	87.3%	72.1%	93.7%
Detection rate (high)	81.6%	78.4%	91.2%
Detection rate (medium)	74.2%	82.7%	89.5%
Overall accuracy	79.4%	67.4%	89.2%
False positive rate	8.7%	24.3%	4.8%
False negative rate	12.6%	8.2%	6.1%
Time per application (h)	8.2	1.4	2.1
MTTR (days)	74.3	58.6	28.5
Coverage (% attack surface)	67.8%	94.2%	91.7%
Zero-day detection	High	None	Medium
Cost per assessment ($)	4,500	800	1,200
Autonomous adaptation	N/A	None	ReAct Loop

C. AI Agent Performance Analysis

The AI agent achieved statistically significant performance gains over traditional automated testing methods ($p < 0.001$, Cohen's d = 1.2) across all vulnerability severity categories. Several factors contributed to this improvement: adaptive payload generation, which increased detection rates for injection vulnerabilities by 23%; enhanced contextual interpretation of application logic, resulting in a 32% improvement in identifying access control weaknesses; and memory-augmented testing, whereby knowledge retained from earlier sessions improved the accuracy of subsequent scans by 18%.

The substantial reduction in false positives—from 24.3% to 4.8%, representing an 80.2% improvement—can be attributed to the agent's attention-based mechanism, which facilitated contextual validation of candidate vulnerabilities. Furthermore,

the incorporation of a ReAct-style reasoning loop enabled the agent to iteratively reassess and correct initial misclassifications, a capability that traditional automated scanners lack.

Despite these advancements, manual testing retained an advantage in zero-day vulnerability detection, uncovering three previously unknown issues compared to the AI agent's single discovery. This outcome underscores the necessity of human expertise for interpreting novel attack patterns and validates the role of the human-in-the-loop verification layer within the proposed architecture.

V. DISCUSSION

The experimental results confirm the effectiveness of AI-agent–based architectures in the context of web application security testing. The proposed four-layer architecture addresses several key limitations inherent in both manual and conventional automated approaches. In particular, the integration of LLM-driven reasoning with specialized ML detection models provides a degree of contextual understanding that has been absent from prior automated systems.

However, the superior performance of manual testing in identifying insecure design vulnerabilities (94.6% compared to 84.2% for the AI agent) underscores the difficulty of translating high-level, abstract security concepts into machine-interpretable representations. Future research should investigate enhanced reasoning techniques, such as chain-of-thought prompting, structured reasoning frameworks, and domain-specific fine-tuning, to mitigate this limitation.

A. Limitations

This study has several limitations. First, the controlled testing environment may not fully reflect the complexity of production-scale systems with diverse technology stacks and network configurations. Second, the six-month evaluation window may be insufficient to capture long-term model drift or evolving threat landscapes. Third, the exclusive focus on web applications limits generalizability to mobile platforms, APIs, and infrastructure-level vulnerabilities. Fourth, MTTR values are based on simulated remediation workflows rather than actual organizational deployment data. Fifth, the primary contribution of this work is architectural integration of existing AI/ML techniques rather than fundamental algorithmic innovation. Finally, while the study has ensured reproducibility, full replication would require access to the custom-developed test applications and specific tool configurations used in the experiments.

VI. CONCLUSION

This study introduces a novel four-layer AI-agent architecture for automated security testing of web applications, offering both a rigorous conceptual framework and empirical validation. The proposed architecture effectively integrates Large Language Model (LLM)-based reasoning with deep-learning–driven detection models, coordinated through a Reasoning-Acting (ReAct)-inspired orchestration mechanism.

Empirical findings demonstrate significant performance gains. The AI agents achieved a detection accuracy of 89.2%, substantially outperforming traditional automated approaches, which reached 67.4%. Automated vulnerability prioritization contributed to a 61.6% reduction in Mean Time to Remediation (MTTR), while an attention-based contextual validation component reduced false positives by 80.2%. Despite these improvements, human expertise remains indispensable for identifying zero-day vulnerabilities and conducting highly complex security evaluations.

Future research avenues include the integration of multi-agent systems to enable collaborative security testing, the development of specialized reasoning pipelines tailored to distinct vulnerability categories, and the exploration of federated learning strategies to support privacy-preserving threat-intelligence sharing.

DATA AVAILABILITY STATEMENT

The test environment configurations and custom vulnerability datasets used in this study are available from the corresponding author upon reasonable request. The OWASP WebGoat (https://owasp.org/www-project-webgoat/) and DVWA (https://dvwa.co.uk/) applications are publicly available at their respective repositories. The CSIC 2010 HTTP Dataset used for model training is publicly available at http://www.isi.csic.es/dataset/.

REFERENCES

[1] "2025 Vulnerability Statistics Report," *Edgescan Stats and Reports*, 2025. https://www.edgescan.com/stats-report/.

[2] "130 Cyber Security Statistics: 2024 Trends and Data," *Human Risk Management*, Aug. 2024.

[3] IBM Security, *Cost of a Data Breach Report 2024*. Armonk, NY, USA: IBM Corporation, 2024.

[4] Z. Xi *et al.*, "The Rise and Potential of Large Language Model-Based Agents: A Survey," *Science China Information Sciences*, vol. 68, no. 2, Feb. 2025, Art. no. 121101, https://doi.org/10.1007/s11432-024-4222-0.

[5] "OWASP Top Ten 2021: Open Web Application Security Project; 2021," *OWASP Foundation*, 2025. https://owasp.org/Top10/2025/.

[6] B. Dawadi, B. Adhikari, and D. Srivastava, "Deep Learning Technique-Enabled Web Application Firewall for the Detection of Web Attacks," *Sensors*, vol. 23, no. 4, Feb. 2023, Art. no. 2073, https://doi.org/10.3390/s23042073.

[7] J. R. Tadhani, V. Vekariya, V. Sorathiya, S. Alshathri, and W. El-Shafai, "Securing Web Applications Against XSS and SQLi Attacks Using a Novel Deep Learning Approach," *Scientific Reports*, vol. 14, no. 1, Jan. 2024, Art. no. 1803, https://doi.org/10.1038/s41598-023-48845-4.

[8] B. B. Ammar and A. M. Alharbi, "SQL Injection Detection Using Fine-Tuned CodeBERT," *Engineering, Technology & Applied Science Research*, vol. 15, no. 5, pp. 27852–27857, Oct. 2025, https://doi.org/10.48084/etasr.13340.

[9] K. Li, H. Yang, and W. Visser, "DaNuoYi: Evolutionary Multitask Injection Testing on Web Application Firewalls," *IEEE Transactions on Software Engineering*, vol. 51, no. 9, pp. 2412–2431, Sept. 2025, https://doi.org/10.1109/TSE.2023.3343716.

[10] S. Hussain *et al.*, "Vulnerability Detection in Java Source Code Using a Quantum Convolutional Neural Network with Self-Attentive Pooling, Deep Sequence, and Graph-based Hybrid Feature Extraction," *Scientific Reports*, vol. 14, no. 1, Mar. 2024, Art. no. 7406, https://doi.org/10.1038/s41598-024-56871-z.

[11] M. E. Durmuşkaya and S. Bayraklı, "Web Application Firewall Based on Machine Learning Models," *PeerJ Computer Science*, vol. 11, July 2025, Art. no. e2975, https://doi.org/10.7717/peerj-cs.2975.

[12] Y. Guo, S. Bettaieb, and F. Casino, "A Comprehensive Analysis on Software Vulnerability Detection Datasets: Trends, Challenges, and

Road Ahead," *International Journal of Information Security*, vol. 23, no. 5, pp. 3311–3327, Oct. 2024, https://doi.org/10.1007/s10207-024-00888-y.

[13] C. Merlano, "Enhancing Cyber Security through Artificial Intelligence and Machine Learning: A Literature Review," *Journal of Cyber Security*, vol. 6, no. 1, pp. 89–116, 2024, https://doi.org/10.32604/jcs.2024.056164.

[14] Y. I. Alzoubi, A. Mishra, and A. E. Topcu, "Research Trends in Deep Learning and Machine Learning for Cloud Computing Security," *Artificial Intelligence Review*, vol. 57, no. 5, May 2024, Art. no. 132, https://doi.org/10.1007/s10462-024-10776-5.

[15] N. Montes, G. Betarte, R. Martínez, and A. Pardo, "Web Application Attacks Detection Using Deep Learning," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, vol. 12702, J. M. R. S. Tavares, J. P. Papa, and M. González Hidalgo, Eds. Cham, Switzerland: Springer International Publishing, 2021, pp. 227–236.

[16] "CWE-918: Server-Side Request Forgery (SSRF)," *Common Weakness Enumeration*, 2024. https://cwe.mitre.org/data/definitions/918.html.

[17] L. Wang *et al.*, "A Survey on Large Language Model Based Autonomous Agents," *Frontiers of Computer Science*, vol. 18, no. 6, Dec. 2024, Art. no. 186345, https://doi.org/10.1007/s11704-024-40231-1.

[18] V. Ciancaglini, M. Balduzzi, S. Gariuolo, R. Vosseler, and F. Tucci, "The Road to Agentic AI: Navigating Architecture, Threats, and Solutions," *Trend Micro*, July 2025. https://www.trendmicro.com/vinfo/us/security/news/security-technology/the-road-to-agentic-ai-navigating-architecture-threats-and-solutions.

[19] C. Wong, "State of Pentesting 2021: The Impact of AI and LLMs on Penetration Testing," *Cobalt*, May 2014. https://www.cobalt.io/blog/state-of-pentesting-2024-impact-of-llms-on-penetration-testing.

[20] G. Deng *et al.*, "PentestGPT: An LLM-empowered Automatic Penetration Testing Tool." arXiv, 2023, https://doi.org/10.48550/ARXIV.2308.06782.

[21] "NodeZero: Autonomous Penetration Testing Platform," *NodeZero*, 2024. https://horizon3.ai/nodezero/.

[22] "RidgeBot Intelligent Penetration Testing Robot," *Ridge Security*, 2024. https://ridgesecurity.ai/ridgebot/ridgebot/.

[23] "AI-Powered Penetration Testing as a Service," *Astra Security*, 2024. https://www.getastra.com/pentesting/web-app.

[24] "Burp Suite Professional with AI Features," *PortSwigger*, 2024. https://portswigger.net/burp/documentation/desktop/burp-ai.

[25] K. Abdulghaffar, N. Elmrabit, and M. Yousefi, "Enhancing Web Application Security through Automated Penetration Testing with Multiple Vulnerability Scanners," *Computers*, vol. 12, no. 11, Nov. 2023, Art. no. 235, https://doi.org/10.3390/computers12110235.

[26] S. Yao, J. Zhao, D. Yu, N. Du, and I. Shafran, "React: Synergizing Reasoning and Acting in Language Models," in *International Conference on Learning Representations*, Kigali, Rwanda, May 2023.

[27] N. Shiri Harzevili, A. Boaye Belle, J. Wang, S. Wang, Z. M. (Jack) Jiang, and N. Nagappan, "A Systematic Literature Review on Automated Software Vulnerability Detection Using Machine Learning," *ACM Computing Surveys*, vol. 57, no. 3, pp. 1–36, Mar. 2025, https://doi.org/10.1145/3699711.

[28] S. He, "Choose Your Agentic AI Architecture Components," *Google Cloud*, Nov. 2025. https://docs.cloud.google.com/architecture/choose-agentic-ai-architecture-components.

[29] R. Modi, "AI Agent Orchestration Patterns," *Azure Architecture Center*. https://learn.microsoft.com/en-us/azure/architecture/ai-ml/guide/ai-agent-design-patterns.

[30] J. M. Nimrod, "AI and Cybersecurity in Penetration Testing," *EC-Council Cybersecurity Exchange*, 2025. https://www.eccouncil.org/cybersecurity-exchange/penetration-testing/ai-and-cybersecurity-in-penetration-testing/.

[31] C. T. Giménez, A. P. Villegas, and G. A. Marañón, "HTTP Dataset CSIC 2010." CSIC, 2010, [Online]. Available: http://www.isi.csic.es/dataset/.