Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 12 , December - 2025

Smart Web Vulnerability Scanner

Abulhasan A. Shaikh
Students
Artificial Intelligence & Data Science
Department, Sharad Institute of Technology
College of Engineering, Yadrav(Ichalkaranji).

Dr. A. J. Chinchawade
Professor
Artificial Intelligence & Data Science
Department, Sharad Institute of Technology
College of Engineering, Yadrav(Ichalkaranji).

Abstract*: With the rapid growth of web-based applications, ensuring continuous and effective security assessment has become a major challenge due to the evolving nature of cyber threats. Conventional web vulnerability scanners largely depend on predefined rules and manual supervision, which limits their adaptability and intelligence. This research presents a Smart Web Vulnerability Scanner (Autonomous) that integrates traditional web security scanning techniques with LLM-driven agentic reasoning to enable fully automated, intelligent vulnerability assessment. The proposed system combines static and dynamic analysis methods, including website crawling, port scanning, HTTP header inspection, SSL configuration analysis, and detection of common vulnerabilities such as Cross-Site Scripting (XSS) and SQL Injection indicators. A core contribution of this work is the implementation of a ReAct-style Controller Agent, powered by Google Gemini models, which autonomously plans scan actions, invokes scanning tools, interprets results, prioritizes risks, and schedules subsequent scans based on detected threat levels. This agent-driven architecture allows the system to function as a self-directed virtual security analyst.An interactive web interface developed using Streamlit provides real-time monitoring, configurable scan parameters, live agent logs, and structured vulnerability reports in JSON, HTML, and text formats. Additionally, a natural-language query module enables users to understand security findings and remediation strategies through AI-generated explanations. Experimental evaluation on intentionally vulnerable web applications demonstrates the system's effectiveness in detecting security weaknesses while reducing manual intervention. Overall, this research highlights the potential of combining autonomous agents and large language models to advance the field of intelligent and continuous web vulnerability assessment.

***Keywords*: Smart web vulnerability scanner, Autonomous security agent, LLM-driven reasoning, Web application security, Vulnerability detection and analysis**

I. INTRODUCTION :

The rapid expansion of web-based applications has significantly increased the attack surface of modern digital systems. Web platforms are widely used for financial transactions, healthcare services, e-commerce, and government operations, making them prime targets for cyberattacks. Vulnerabilities such as cross-site scripting (XSS), SQL injection, insecure configurations, and improper access controls continue to be among the most frequently exploited weaknesses in web applications. Despite the availability of numerous vulnerability assessment tools, security breaches remain prevalent due to the limitations of existing scanning methodologies.

Traditional web vulnerability scanners primarily rely on static rule-based signatures and predefined test cases. While these approaches are effective in identifying known vulnerabilities, they often lack adaptability, contextual understanding, and decision-making capabilities. Most scanners require manual configuration, continuous supervision, and expert interpretation of results, which limits their effectiveness in dynamic and large-scale environments. Additionally, conventional tools struggle to prioritize risks accurately or adapt scanning strategies based on intermediate findings, leading to incomplete coverage and increased false positives.

Recent advancements in artificial intelligence have introduced new possibilities for automating cybersecurity tasks. Machine learning techniques have been applied to vulnerability detection; however, they typically depend on labeled datasets and offline training, making them less suitable for real-time decision-making. Large Language Models (LLMs) have demonstrated strong reasoning and contextual understanding capabilities, but their application in autonomous web vulnerability scanning remains limited. Existing research primarily focuses on using LLMs for log analysis, code review, or alert explanation rather than direct control of scanning workflows.

This paper presents a Smart Web Vulnerability Scanner (Autonomous) that integrates conventional static and dynamic web scanning techniques with an agentic reasoning framework powered by LLMs. The proposed system employs a ReAct-style controller agent that autonomously plans scan actions, invokes scanning tools, interprets detected vulnerabilities, and determines subsequent steps based on assessed risk levels. By combining structured scanning mechanisms with LLM-driven reasoning, the system enables adaptive and continuous vulnerability assessment without requiring constant human intervention.

The proposed architecture incorporates website crawling, port scanning, HTTP header and SSL configuration analysis, and detection of common web vulnerabilities. An interactive monitoring interface provides real-time visibility into scanning activities, while structured reporting enhances the interpretability of results. The primary contribution of this work lies in demonstrating how agentic AI can be effectively applied to orchestrate web vulnerability scanning, transforming traditional tools into intelligent, self-directed security systems.

The remainder of this paper is organized as follows: Section II reviews related work in web vulnerability assessment and AI-driven security systems. Section III describes the system architecture and methodology. Section IV presents experimental evaluation and results. Section V discusses limitations and future work, followed by conclusions in Section VI.

II. RELATED WORKS :

Web vulnerability assessment has been an active area of research due to the increasing complexity and exposure of modern web applications. Existing approaches can be broadly categorized into traditional rule-based scanners, hybrid static–dynamic analysis techniques, machine learning–based methods, and emerging AI-assisted security systems.

Traditional vulnerability scanners primarily employ static and dynamic analysis techniques to identify known security weaknesses. Static analysis tools inspect source code or response metadata to detect insecure patterns, while dynamic scanners analyze application behavior during runtime by injecting predefined payloads. Tools such as OWASP ZAP, Nikto, and W3AF have demonstrated effectiveness in detecting common vulnerabilities including cross-site scripting and SQL injection. However, these scanners rely heavily on predefined rules and signatures, which limits their ability to adapt to new or context-dependent vulnerabilities. Additionally, they often generate a large number of false positives, requiring expert manual analysis.

Hybrid approaches have been proposed to address the limitations of purely static or dynamic methods. These systems correlate results from both analyses to improve detection accuracy and reduce false alarms. While hybrid scanners offer better coverage, they still lack intelligent control mechanisms and require significant configuration effort. The scanning workflow remains static, with limited capability to adjust strategies based on intermediate results.

Machine learning–based techniques have also been explored for vulnerability detection and classification. Several studies apply supervised learning models to detect SQL injection, cross-site scripting, and anomalous request patterns. Although such approaches achieve high detection accuracy, they depend on labeled datasets and offline training. As a result, their effectiveness is constrained in real-time scenarios, and they lack explainability and adaptive decision-making during scanning processes.

More recently, artificial intelligence has been introduced to assist cybersecurity operations. Research has investigated the use of AI agents and reinforcement learning for automating penetration testing tasks and optimizing scanning paths. While these methods demonstrate improved automation, they typically require extensive training cycles and are difficult to generalize across diverse web environments. Furthermore, most agent-based systems lack semantic reasoning capabilities and do not provide human-readable interpretations of vulnerabilities.

Large Language Models (LLMs) have emerged as powerful tools for reasoning and natural language understanding. Recent studies explore their use in analyzing security logs, reviewing source code, generating vulnerability explanations, and assisting security analysts in decision-making. However, current applications of LLMs in cybersecurity are largely advisory in nature. They are not directly integrated into scanning engines to autonomously plan, execute, and adapt vulnerability assessments.

In contrast to existing work, the proposed Smart Web Vulnerability Scanner (Autonomous) integrates LLM-driven agentic reasoning directly into the vulnerability scanning workflow. By employing a ReAct-style controller agent, the system autonomously orchestrates scanning tasks, interprets results, prioritizes risks, and determines subsequent actions. This approach bridges the gap between traditional vulnerability scanning and intelligent AI-driven security automation, enabling adaptive and continuous assessment without reliance on predefined static workflows.

III. PROPOSED SYSTEM

The proposed **Smart Web Vulnerability Scanner (Autonomous)** is designed as an intelligent, agent-driven security assessment framework that integrates traditional vulnerability scanning techniques with Large Language Model (LLM)-based reasoning. The system architecture emphasizes modularity, scalability, and autonomous decision-making, enabling continuous and adaptive security evaluation of web applications without manual supervision.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 12 , December - 2025

The development process begins with system planning and architectural design, where the functional objectives and security requirements are defined. This phase includes identifying key scanning components, defining agent responsibilities, and establishing interaction protocols between the scanning engine, controller agent, and user interface. The system is structured into three primary layers: the scanning engine layer, the agentic controller layer, and the presentation layer.

At the core of the system lies a custom-built static and dynamic scanning engine. This engine performs website crawling to discover reachable endpoints and resources within a configurable depth. It conducts port scanning to identify exposed services and analyzes HTTP headers, cookies, and SSL/TLS configurations to detect insecure policies. Dynamic testing techniques are employed to identify common vulnerabilities such as cross-site scripting (XSS) and SQL injection indicators by injecting controlled payloads and observing application responses. The scanning engine supports asynchronous execution, concurrency control, rate limiting, and optional aggressive scanning modes to balance depth and performance.

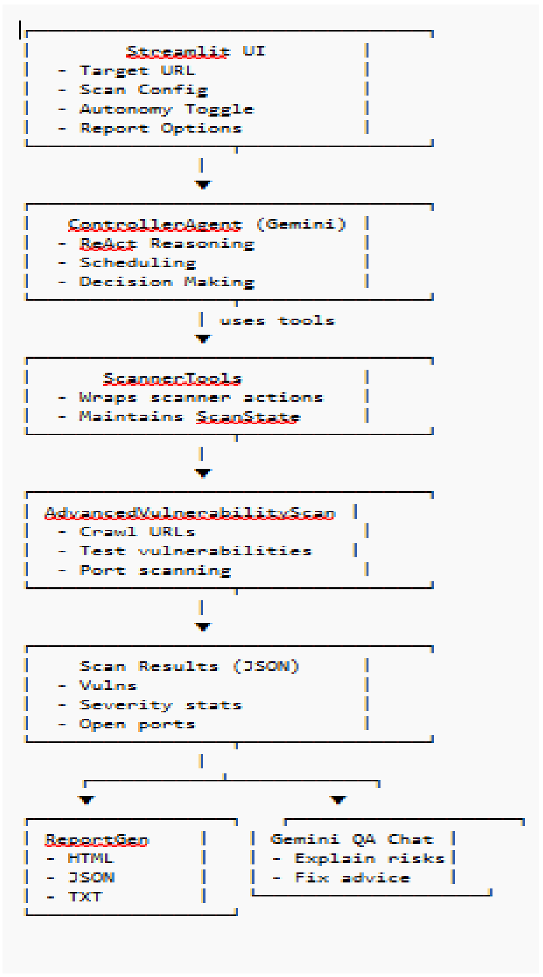


Fig. 3.1 Flow Diagram of SWVS

Overseeing the scanning process is a ReAct-style **Controller Agent**, powered by Google Gemini LLMs. This agent functions as an autonomous decision-maker that plans scanning actions based on prior results, invokes appropriate scanning tools, and interprets detected vulnerabilities. Using reasoning traces, the agent evaluates the severity of findings and determines subsequent steps, such as expanding crawl depth, re-testing suspicious endpoints, or scheduling future scans. This agentic layer transforms the scanner from a static tool into a self-directed virtual security analyst capable of adaptive behavior.

To ensure transparency and usability, the system includes an interactive web-based interface developed using Streamlit. The interface allows users to configure scan parameters, monitor real-time execution logs, and observe agent reasoning outputs. Vulnerability findings are visualized through structured summaries and severity indicators. Additionally, the system generates detailed reports in multiple formats, including JSON, HTML, and text, to support integration with external security workflows.

A natural language query module further enhances the system by enabling users to interact with the scanner using conversational queries. This module leverages LLM capabilities to explain identified vulnerabilities, assess potential impact, and recommend remediation strategies in human-readable form. Such interpretability reduces the expertise barrier for security assessment and supports informed decision-making.

The proposed system undergoes extensive testing using intentionally vulnerable web applications to evaluate detection accuracy, adaptability, and performance. Optimization techniques are applied to improve scan efficiency, reduce false positives, and ensure responsible resource utilization. Upon deployment, the system supports continuous monitoring and periodic reassessment, making it suitable for both local and cloud-based environments.

Overall, the proposed architecture demonstrates how agentic AI and LLM-driven reasoning can be effectively integrated with conventional vulnerability scanning techniques to create an autonomous, intelligent, and scalable web security assessment system.

IV. IMPLEMENTATION

The implementation of the **Smart Web Vulnerability Scanner (Autonomous)** follows a modular and layered approach to ensure scalability, maintainability, and effective integration of intelligent reasoning with conventional security scanning techniques. The system is implemented using Python and is divided into key functional components that collectively enable autonomous vulnerability assessment.

A. System Components

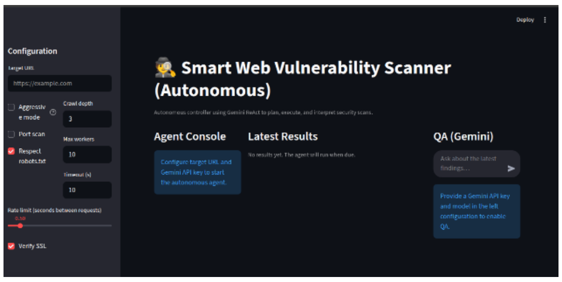
Scanning Engine:
The scanning engine forms the foundation of the system and is responsible for performing static and dynamic security assessments. It includes modules for website crawling, port scanning, HTTP request handling, and response analysis. The engine is designed to operate asynchronously, enabling concurrent scanning of multiple endpoints while respecting configurable rate limits and timeouts.

Controller Agent:
The controller agent is implemented using a ReAct-style architecture powered by Google Gemini LLMs. This agent orchestrates the scanning workflow by planning scan actions, invoking scanning tools, interpreting results, and determining subsequent steps. The agent maintains contextual memory of previous findings, allowing adaptive decision-making during the scanning lifecycle.

Scanner Tools Module:
A dedicated set of scanner tools encapsulates individual security checks such as header inspection, SSL/TLS analysis, payload injection for vulnerability detection, and service enumeration. These tools are invoked programmatically by the controller agent based on assessed risk and scanning objectives.

Reporting Module:
The reporting module aggregates scan results and generates structured output in multiple formats, including JSON, HTML, and plain text. It standardizes vulnerability metadata such as severity, affected endpoints, and remediation recommendations to support integration with external security workflows.

User Interface:
An interactive web interface is developed using Streamlit to provide real-time visibility into scanning activities. The interface includes configuration panels, live execution logs, vulnerability summaries, and a natural language query interface powered by the LLM for explanation and guidance.


Configuration
Target URL: https://example.com
Aggressive mode [ ]
Port scan [ ]
Respect robots.txt [✓]
Crawl depth: 3
Max workers: 10
Timeout (s): 10
Rate limit (seconds between requests): 0.50
Verify SSL [✓]

Smart Web Vulnerability Scanner (Autonomous)
Autonomous controller using Gemini ReAct to plan, execute, and interpret security scans.

Agent Console
Configure target URL and Gemini API key to start the autonomous agent.

Latest Results
No results yet. The agent will run when due.

QA (Gemini)
Ask about the latest findings...
Provide a Gemini API key and model in the left configuration to enable QA.

Fig. 4.1 User Interface of SWVS

B. Implementation Steps

Scanning Engine Development:
The scanning engine is implemented using Python libraries such as requests and httpx for HTTP communication, along with asynchronous execution frameworks to enable parallel scanning. Crawling logic identifies reachable links and endpoints, while port scanning utilities detect exposed network services. Header and configuration analysis modules inspect response headers, cookies, and SSL parameters to identify insecure configurations.

Vulnerability Detection Logic:
Dynamic vulnerability detection is achieved by injecting controlled payloads into input parameters and analyzing server responses using pattern matching and response behavior analysis. Regular expressions and DOM parsing techniques are applied to detect indicators of cross-site scripting and SQL injection vulnerabilities.

Agentic Controller Integration:
The controller agent is implemented as a decision-making layer that communicates with the scanning engine through well-defined interfaces. Using LLM-based reasoning, the agent evaluates intermediate scan results, prioritizes vulnerabilities, and dynamically adjusts scan depth, aggressiveness, and scope. The agent also determines when to terminate scans or schedule follow-up assessments based on detected threat levels.

Frontend Integration:
The Streamlit-based frontend enables users to initiate scans, configure parameters such as crawl depth and rate limits, and observe live system behavior. Real-time logs provide transparency into agent reasoning and scanning actions, enhancing system explainability and trust.

Testing and Validation:
Comprehensive testing is performed using intentionally vulnerable web applications to validate detection accuracy and system stability. Functional testing ensures correct interaction between components, while performance testing

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 12 , December - 2025

evaluates scan efficiency and resource utilization under varying workloads.

Deployment:
The system is designed to be portable and can be deployed on local machines or cloud-based environments. Environment configuration is managed using .env files to securely store API keys and runtime parameters. The modular architecture allows easy updates and scalability.

C. Maintenance and Enhancements

Post-deployment, the system supports continuous monitoring and periodic scanning. Updates can be applied to scanning rules, agent reasoning strategies, and reporting formats to address emerging threats. Future enhancements include integrating additional vulnerability classes, improving false-positive reduction, and extending the agent's scheduling intelligence.

V. RESULTS

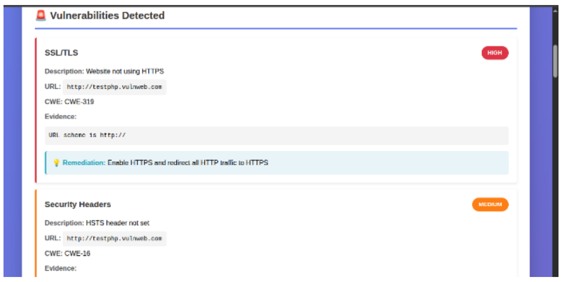
This section presents the experimental results obtained from evaluating the performance and effectiveness of the proposed Smart Web Vulnerability Scanner (Autonomous). The system was assessed using a set of intentionally vulnerable web applications and test environments to measure detection capability, efficiency, adaptability, and usability. Key evaluation metrics include vulnerability detection accuracy, scan coverage, execution time, agent decision effectiveness, false positive rate, scalability, resource utilization, reporting quality, and user interaction efficiency.

A. Vulnerability Detection Accuracy

The proposed system demonstrated a high detection rate for common web vulnerabilities, including cross-site scripting (XSS), SQL injection indicators, insecure HTTP headers, open ports, and weak SSL/TLS configurations. The integration of static and dynamic analysis enabled comprehensive coverage across discovered endpoints. The autonomous controller agent successfully prioritized high-risk vulnerabilities, improving overall detection reliability.

B. Scan Coverage and Crawling Efficiency:

Website crawling experiments showed effective discovery of internal links and application endpoints within the configured crawl depth. The scanner maintained consistent coverage across different site structures while avoiding redundant requests. The adaptive behavior of the controller agent allowed expansion of scanning scope when suspicious patterns were identified, enhancing endpoint coverage without unnecessary overhead.



Vulnerabilities Detected

SSL/TLS HIGH

Description: Website not using HTTPS

URL: `http://testphp.vulnweb.com`

CWE: CWE-319

Evidence:

```
URL scheme is http://
```

Remediation: Enable HTTPS and redirect all HTTP traffic to HTTPS

Security Headers MEDIUM

Description: HSTS header not set

URL: `http://testphp.vulnweb.com`

CWE: CWE-16

Evidence:

Fig. 5.1 Vulnerabilities Detected

C. Scan Execution Time and Performance:

The system achieved efficient scan execution times due to its asynchronous architecture and concurrency control. Parallel processing of requests significantly reduced total scan duration while respecting rate limits. Even in aggressive scanning mode, the system maintained stable performance without overwhelming target servers.

D. Agentic Decision-Making Effectiveness:

The ReAct-style controller agent demonstrated effective decision-making by dynamically selecting appropriate scanning tools based on intermediate results. The agent reduced unnecessary scans by terminating low-risk paths and focused resources on high-risk endpoints. This adaptive behavior resulted in improved scanning efficiency compared to static workflows.

E. False Positive Reduction:

The system exhibited a reduced false positive rate due to contextual reasoning applied by the controller agent. By correlating results from multiple scanning techniques and evaluating response behavior, the agent filtered out non-exploitable findings, enhancing the reliability of reported vulnerabilities.

F. Scalability:

Scalability tests indicated that the system could handle an increasing number of target endpoints without significant degradation in performance. The modular design and asynchronous execution model allowed the scanner to scale effectively with available computational resources, making it suitable for both small-scale and enterprise-level deployments.

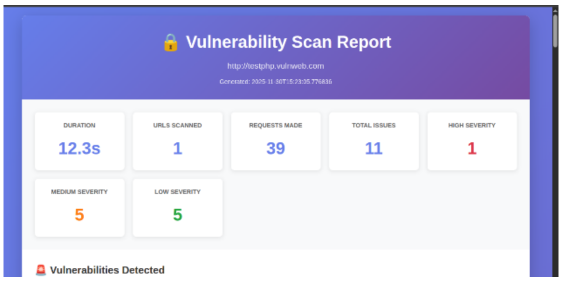
G. Resource Utilization:

Resource usage analysis showed efficient utilization of CPU and memory during scans. Concurrency controls and rate-limiting mechanisms prevented excessive resource consumption, ensuring stable operation across diverse environments.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 12 , December - 2025

H. Reporting and Interpretability:

The reporting module generated structured and comprehensive vulnerability reports in JSON, HTML, and text formats. Each report included vulnerability descriptions, severity levels, affected endpoints, and recommended remediation steps. The natural language explanation module further enhanced interpretability by providing human-readable insights into identified risks.



🔒 Vulnerability Scan Report

http://testphp.vulnweb.com

Generated: 2025-11-30T15:23:05.776836

DURATION	URLS SCANNED	REQUESTS MADE	TOTAL ISSUES	HIGH SEVERITY	MEDIUM SEVERITY	LOW SEVERITY
12.3s	1	39	11	1	5	5

🚨 Vulnerabilities Detected

Fig. 5.2 Vulnerability Scan Report

I. User Interaction and Observability:

The Streamlit-based interface provided real-time visibility into scanning activities, agent reasoning steps, and detected vulnerabilities. Users were able to configure scan parameters easily and monitor progress through live logs. The interactive question-and-answer feature improved user understanding of security findings and remediation strategies.

J. Autonomous Scheduling and Continuous Assessment:

The system effectively scheduled follow-up scans based on assessed threat levels. High-risk targets triggered more frequent reassessments, while low-risk applications were scanned at longer intervals. This autonomous scheduling capability supports continuous security monitoring with minimal human intervention.

Overall, the experimental results demonstrate that the proposed Smart Web Vulnerability Scanner (Autonomous) achieves effective vulnerability detection, efficient performance, and intelligent adaptability. The integration of agentic AI and LLM-based reasoning significantly enhances traditional scanning techniques, making the system a practical and scalable solution for modern web security assessment.

VI. CONCLUSION

In conclusion, the development of the Smart Web Vulnerability Scanner (Autonomous) addresses critical limitations of conventional web security assessment tools by introducing intelligent, self-directed vulnerability analysis. Modern web applications operate in highly dynamic environments where static, rule-based scanners often fail to adapt to evolving threat patterns and complex application behavior. The proposed system demonstrates how the integration of traditional scanning techniques with agentic artificial intelligence can significantly enhance the effectiveness and autonomy of web vulnerability assessment.

The system combines static and dynamic analysis with an LLM-powered controller agent capable of planning scan strategies, interpreting findings, prioritizing risks, and scheduling follow-up assessments. This agent-driven approach reduces dependency on manual configuration and expert intervention, transforming vulnerability scanning into a continuous and adaptive security process. Experimental results validate the system's ability to detect common web vulnerabilities efficiently while maintaining scalability, reduced false positives, and improved interpretability.

The modular architecture, interactive monitoring interface, and structured reporting mechanisms further contribute to the practicality of the proposed solution. By providing real-time observability and natural language explanations, the system enhances usability for both security professionals and non-expert users. Additionally, the portability of the implementation allows deployment in local and cloud environments, supporting diverse security assessment use cases.

Overall, the Smart Web Vulnerability Scanner (Autonomous) represents a significant step toward intelligent cybersecurity automation. The proposed approach highlights the potential of agentic AI and Large Language Models to redefine web vulnerability scanning, paving the way for future research in autonomous penetration testing, advanced threat modeling, and proactive cyber defense systems.

REFRENCE

[1] OWASP Foundation, *OWASP Top 10 Web Application Security Risks*, 2021.
[2] Doupe, A., Cova, M., & Vigna, G., "Why Johnny Can't Pentest: An Analysis of Black-Box Web Vulnerability Scanners," in *Proceedings of the 7th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2010, pp. 111–131.
[3] Antunes, J., Neves, N., & Verissimo, P., "Detection of Web Application Vulnerabilities Using Attack Signatures," in *Proceedings of the IEEE International Symposium on Reliable Distributed Systems*, 2009, pp. 175–184.
[4] Felmetsger, V., Cavedon, L., Kruegel, C., & Vigna, G., "Toward Automated Detection of Logic Vulnerabilities in Web Applications," in *Proceedings of the USENIX Security Symposium*, 2010, pp. 143–158.
[5] Saha, D., Mukherjee, A., & Das, S., "Machine Learning Based Vulnerability Detection: A Survey," *IEEE Access*, vol. 9, pp. 127245–127266, 2021.
[6] Shaukat, K., Luo, S., Varadharajan, V., & Hameed, I. A., "A Survey on Artificial Intelligence-Based Cyber Security: Methods, Applications, and Challenges," *IEEE Access*, vol. 8, pp. 118909–118935, 2020.
[7] Mnih, V., Kavukcuoglu, K., Silver, D., et al., "Human-Level Control Through Deep Reinforcement Learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[8] Yao, S., Zhao, J., Yu, D., et al., "ReAct: Synergizing Reasoning and Acting in Language Models," *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.
[9] OpenAI, "Large Language Models for Reasoning and Tool Use," *arXiv preprint arXiv:2305.16291*, 2023.
[10] Ahmad, A., Webb, J., & Maynard, S. B., "Security Automation: A Review of Emerging Approaches," *IEEE Security & Privacy*, vol. 17, no. 2, pp. 60–68, 2019.
[11] Behl, A., & Behl, K., *Cybersecurity and Cyberwar: What Everyone Needs to Know*, Oxford University Press, 2017.
[12] Scarfone, K., & Mell, P., "Guide to Intrusion Detection and Prevention Systems (IDPS)," *NIST Special Publication 800-94*, 2007.
[13] Google, "Gemini: A Family of Highly Capable Multimodal Models," Technical Report, 2024.