IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.155202

AI-Powered Penetration Testing Platform for Automated Vulnerability Detection: A Survey of Artificial Intelligence Methods for Identifying System Weaknesses

Mrs. Bindu K.P[1], Gagana R[2], Kushal K R[3], M G Sahana[4], and M Harshit Pramod[5]

Assistant Prof., Dept of CSE, K.S.School of Engineering & Management, Bengaluru, India[1]

Student, Dept of CSE, K.S.School of Engineering & Management, Bengaluru, India[2–5]

Abstract: The increasing use of shared code libraries, cloud platforms, and collaborative repositories has introduced serious security risks such as hidden vulnerabilities, leaked credentials, and insecure dependencies. Traditional penetration testing methods are often slow, manual, and difficult to integrate into modern DevSecOps workflows. This paper reviews recent AI-based vulnerability detection and penetration testing systems, analyzing their strengths and limitations. It also proposes an AI-powered autonomous security platform that combines static code analysis, dependency scanning, secret detection, Docker-based exploit verification, continuous GitHub monitoring, and AI-generated remediation guidance within a unified interface to improve repository security and vulnerability management.

Keywords: Penetration Testing, Vulnerability Detection, Repository Security, DevSecOps, Artificial Intelligence, Machine Learning, Secret Detection, GitHub Security, Exploit Verification, Docker Sandbox

I. INTRODUCTION

Modern software development practices increasingly depend on open-source frameworks, third-party APIs, cloud services, and collaborative repository platforms such as GitHub. While these technologies accelerate development efficiency and scalability, they simultaneously introduce significant cybersecurity risks including vulnerable dependencies, insecure coding practices, exposed credentials, and exploitable software vulnerabilities. Recent software supply-chain attacks and repository-level data breaches have demonstrated the importance of integrating continuous security assessment directly into software development workflows.

Traditional penetration testing methods rely heavily on manual analysis performed by cybersecurity professionals and ethical hackers. Although manual penetration testing remains effective, the process is time-consuming, resource-intensive, and difficult to integrate continuously within agile and DevSecOps environments. Many small development teams and independent developers lack access to dedicated security professionals, resulting in delayed vulnerability identification and increased exposure to cyber threats.

Recent advancements in Artificial Intelligence (AI), Machine Learning (ML), and Large Language Models (LLMs) have demonstrated promising applications in automated vulnerability detection, repository analysis, exploit reasoning, and cybersecurity assistance. Several research efforts have explored AI-driven penetration testing frameworks, contextual vulnerability detection systems, secret analysis tools, and intelligent static analyzers. However, existing solutions primarily address isolated components of the security analysis process rather than integrating vulnerability detection, exploit verification, repository monitoring, and remediation assistance into a unified deployable platform.

This paper surveys six representative studies published between 2024 and 2026 related to AI-assisted penetration testing and repository security analysis. The reviewed works are critically analyzed based on methodology, AI integration, exploit verification capability, repository monitoring support, and implementation limitations. Based on the identified research gaps, we propose an AI-Powered Autonomous Penetration Testing and Repository Security Platform designed to provide continuous GitHub repository monitoring, automated vulnerability analysis, Docker-based exploit verification, and AI-generated remediation guidance through a unified DevSecOps-oriented system.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026
DOI: 10.17148/IJARCCE.2026.155202

A. Research Contributions

This survey and the associated proposed system make the following research contributions:

- A structured comparative analysis of recent AI-assisted penetration testing and vulnerability analysis systems.
- Identification of critical gaps in existing research related to exploit verification, continuous repository monitoring, and vulnerability prioritization.
- Proposal of a unified AI-powered repository security platform integrating static analysis, dependency scanning, secret detection, exploit verification, and remediation assistance.
- Introduction of Docker-based isolated exploit verification for reducing false-positive vulnerability reports.
- Integration of GitHub repository monitoring and webhook-triggered automated security analysis within DevSecOps workflows.

II. RELEVANT LITERATURE

A. Paper 1: Research on Vulnerability Detection Methods for SpringMVC Web Source Code (Jingyi Zhu et al., 2025)

Jingyi Zhu et al. [1] proposed a deep learning-based vulnerability detection framework for SpringMVC web applications by integrating CodeBERT, TextCNN, and Bidirectional Long Short-Term Memory (BiLSTM) models. The proposed architecture combines contextual semantic extraction, local feature analysis, and sequential dependency modeling to improve vulnerability detection within SpringMVC source code.

The framework utilized manually collected vulnerability samples along with real-world GitHub project repositories for dataset construction. The system incorporated preprocessing techniques including source code cleaning, tokenization, Word2Vec-based vectorization, and feature extraction before training the vulnerability classification model. The proposed architecture specifically targeted vulnerabilities such as SQL Injection, XSS Injection, XML Injection, Command Injection, and Arbitrary File Upload vulnerabilities.

Experimental evaluation demonstrated strong performance, achieving approximately 96.98% average accuracy and 93.44% F1-score across multiple vulnerability categories. Comparative analysis additionally showed that the hybrid CodeBERT + TextCNN + BiLSTM architecture outperformed traditional static detection tools and single deep learning models in both accuracy and detection efficiency.

Although the framework significantly improved vulnerability detection accuracy, the proposed approach primarily focused on source-code-level analysis and did not incorporate exploit verification, continuous repository monitoring, GitHub webhook integration, or AI-assisted remediation guidance. The system also lacked practical DevSecOps deployment mechanisms and repository-wide security analytics features.

B. Paper 2: PentestGPT – Evaluating and Harnessing Large Language Models for Automated Penetration Testing (G. Deng et al., 2024)

Deng et al. [2] proposed PentestGPT, an AI-assisted penetration testing framework utilizing Large Language Models to automate vulnerability analysis, exploit reasoning, and remediation assistance. The framework demonstrated the capability of LLMs to generate contextual penetration testing guidance and assist security professionals during vulnerability assessment procedures.

The study highlighted the growing potential of AI-assisted cybersecurity systems for reducing manual penetration testing effort. However, the framework still required human supervision for validating attack strategies and did not incorporate automated exploit verification within isolated runtime environments. Continuous repository monitoring and GitHub integration were also not addressed within the proposed approach.

C. Paper 3: IRIS – LLM-Assisted Static Analysis for Detecting Security Vulnerabilities (Z. Li et al., 2025)

Li et al. [3] introduced IRIS, a vulnerability detection framework combining static analysis with Large Language Models for contextual software security analysis. The system utilized taint analysis and AI-assisted reasoning for detecting vulnerabilities such as SQL injection, insecure input handling, and authentication flaws.

The framework demonstrated improved contextual understanding compared to traditional static analyzers and reduced false-positive reports during vulnerability detection. However, the system focused primarily on source-code-level analysis and did not perform exploit verification or continuous repository-level monitoring.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026
DOI: 10.17148/IJARCCE.2026.155202

D. Paper 4: Secret Leak Detection Using Large Language Models (S. Ahmed et al., 2024)

Ahmed et al. [4] proposed a Large Language Model-assisted framework for detecting exposed secrets such as API keys, authentication tokens, and passwords within software repositories and issue reports. The system combined contextual AI analysis with traditional pattern-matching techniques to improve secret detection accuracy.

The framework successfully reduced false-positive secret detection alerts compared to regex-only approaches. However, the study focused exclusively on credential analysis and did not integrate vulnerability detection, exploit verification, or repository-wide security assessment capabilities.

E. Paper 5: GRACE – Graph-Based LLM Vulnerability Detection Framework (Guilong Lu et al., 2024)

Lu et al. [5] proposed GRACE, a graph-enhanced vulnerability detection framework combining structural code analysis with contextual Large Language Model reasoning. The framework attempted to improve vulnerability detection accuracy by analyzing relationships between software components, execution flows, and contextual code structures.

The system demonstrated improved contextual reasoning during software vulnerability analysis. However, the approach introduced substantial computational overhead and lacked practical exploit verification or deployment-oriented DevSecOps integration.

F. Paper 6: LLMxCPG – Context-Aware Vulnerability Detection Using Code Property Graphs (A. Lekssays et al., 2025)

Lekssays et al. [6] proposed LLMxCPG, a context-aware vulnerability detection framework integrating Code Property Graphs with Large Language Models for software vulnerability analysis. The framework improved contextual code understanding and vulnerability reasoning using structured graph-based representations.

Although the system improved vulnerability detection capabilities, it focused primarily on theoretical vulnerability analysis and lacked exploit verification, repository monitoring, and real-time security integration features.

III. COMPARATIVE ANALYSIS OF EXISTING SYSTEMS

Table I presents a structured comparison of the reviewed AI-assisted penetration testing and repository security analysis systems.

TABLE I. COMPARISON OF EXISTING RESEARCH ON AI-BASED VULNERABILITY ANALYSIS SYSTEMS

Ref.	Year	AI Model Used	Route Opt.	Real-Time Tracking	Key Limitation
[1] Zhu et al.	2025	CodeBERT + TextCNN + BiLSTM	No	No	No exploit verification
[2] Deng et al.	2024	LLM-Based Reasoning	No	No	Requires human supervision
[3] Li et al.	2025	Static Analysis + LLM	No	No	No exploit validation
[4] Ahmed et al.	2024	LLM + Pattern Matching	No	Limited	Secret detection only
[5] Lu et al.	2024	Graph-Based LLM Analysis	No	No	High computational overhead
[6] Lekssays et al.	2025	Code Property Graph + LLM	No	No	No DevSecOps integration

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.155202

IV. GAP ANALYSIS

Based on the review of the six studies, the following critical gaps are identified in the existing literature.

A. Absence of Unified Integrated Platforms

Existing research primarily addresses individual security tasks independently. Vulnerability detection systems generally do not incorporate exploit verification, while secret detection frameworks lack repository-wide security analysis capabilities. No single deployable platform integrates repository monitoring, exploit validation, vulnerability analysis, and remediation assistance into a unified workflow.

B. Lack of Exploit Verification

Most vulnerability analysis frameworks identify theoretical vulnerabilities without confirming practical exploitability. This results in excessive false-positive reports and difficulty in vulnerability prioritization for developers.

C. Limited Continuous Repository Monitoring

Current AI-assisted penetration testing systems generally perform isolated vulnerability analysis rather than continuously monitoring repositories during software development. Real-time GitHub integration and webhook-triggered scanning remain largely unexplored.

D. Inadequate Developer-Friendly Remediation Guidance

Many existing security analysis tools generate highly technical vulnerability reports that are difficult for developers without cybersecurity expertise to understand. Simplified AI-assisted remediation guidance remains limited in current research systems.

E. Limited DevSecOps Integration

Most reviewed systems remain research-oriented prototypes without practical deployment-oriented DevSecOps integration suitable for continuous software development environments.

V. PROPOSED SYSTEM DESIGN

A. System Overview

To address the identified gaps, we propose an AI-Powered Autonomous Penetration Testing and Repository Security Platform. The proposed system integrates GitHub repository monitoring, static vulnerability analysis, dependency scanning, machine learning-assisted secret detection, Docker-based exploit verification, and AI-generated remediation guidance within a unified web-based platform.

The system is designed as a DevSecOps-oriented solution capable of continuously monitoring repositories during software development workflows without requiring specialized security expertise from developers.

B. System Workflow

The proposed workflow follows a structured sequence:

- User authenticates using GitHub OAuth integration.
- Repository monitoring is enabled through webhook configuration.
- Source code and dependency files are analyzed for vulnerabilities.
- Secret detection mechanisms identify exposed credentials and API keys.
- Docker-isolated sandbox environments validate detected vulnerabilities through exploit verification.
- AI-generated remediation suggestions are produced for identified vulnerabilities.
- Repository security reports and risk analytics are displayed through a centralized dashboard.

C. AI Components

The proposed platform incorporates two primary AI-assisted components:

AI Component 1: Vulnerability Analysis Engine

This module combines static analysis tools with AI-assisted contextual reasoning for detecting insecure coding practices, vulnerable dependencies, and application-level vulnerabilities.

AI Component 2: Secret Detection and Remediation Engine

This module performs machine learning-assisted credential analysis and generates AI-based remediation suggestions for detected vulnerabilities and exposed credentials.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.155202

D. Key Parameters

The proposed system incorporates the following major features:

- GitHub repository integration
- Continuous repository monitoring
- Static vulnerability analysis
- Dependency vulnerability scanning
- Secret detection
- Docker-based exploit verification
- AI-generated remediation guidance
- Repository risk scoring
- Security analytics dashboard

VI. EXPECTED OUTCOMES AND BENEFITS

A. Improved Vulnerability Prioritization

The proposed exploit verification mechanism is expected to reduce false-positive vulnerability reports by validating practical exploitability within isolated Docker environments. This helps developers focus on critical and exploitable security issues more effectively.

B. Continuous Repository Security Monitoring

Webhook-triggered automated scanning enables continuous repository-level security analysis during software development workflows. The system allows newly introduced vulnerabilities to be identified immediately after code updates.

C. Improved Developer Awareness

AI-generated vulnerability explanations and remediation guidance improve accessibility for developers with limited cybersecurity expertise. The platform additionally promotes better understanding of secure coding practices and common software vulnerabilities.

D. Practical DevSecOps Integration

The proposed system supports integration of automated security analysis directly into modern DevSecOps-oriented software development practices. Continuous monitoring and automated scanning help improve software security throughout the development lifecycle.

VII. CONCLUSION AND FUTURE WORK

This paper presented a comprehensive survey of recent AI-assisted penetration testing and repository security analysis systems. The reviewed studies demonstrate the growing importance of Artificial Intelligence and Large Language Models within automated vulnerability analysis and cybersecurity workflows. However, significant limitations remain regarding exploit verification, continuous repository monitoring, and integrated DevSecOps deployment.

To address these gaps, we proposed an AI-Powered Autonomous Penetration Testing and Repository Security Platform integrating vulnerability analysis, secret detection, Docker-based exploit verification, GitHub repository monitoring, and AI-generated remediation guidance within a unified web-based system.

Future work will focus on implementation and validation of the proposed platform using real-world GitHub repositories, integration of advanced threat intelligence mechanisms, and improvement of exploit verification accuracy through adaptive AI-based attack simulation techniques.

VIII. ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to the Department of Computer Science and Engineering, K. S. School of Engineering and Management (KSSEM), Bengaluru, for providing an excellent academic and research environment for this work. We extend our heartfelt thanks to our project guide, Mrs. Bindu K.P, Assistant Professor, Department of Computer Science and Engineering, for her valuable guidance, encouragement, and continuous support throughout the preparation of this paper.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026
DOI: 10.17148/IJARCCE.2026.155202

We also acknowledge the contributions of the cybersecurity research community and the authors of the reviewed studies whose work provided valuable insights and foundation for this survey and proposed system.

REFERENCES

[1]. J. Zhu, Y. Zhou, and H. Wang, "Research on Vulnerability Detection Methods for SpringMVC Web Source Code," in Proc. International Conference on Advanced Computing and Intelligent Robotics Applications (ACIRA), 2025.
[2]. G. Deng et al., "PentestGPT: Evaluating and Harnessing Large Language Models for Automated Penetration Testing," in Proc. USENIX Security Symposium, 2024.
[3]. Z. Li et al., "IRIS: LLM-Assisted Static Analysis for Detecting Security Vulnerabilities," in Proc. International Conference on Learning Representations (ICLR), 2025.
[4]. S. Ahmed et al., "Secret Leak Detection in Software Issue Reports using Large Language Models," arXiv preprint arXiv:2410.23657, 2024.
[5]. G. Lu et al., "GRACE: Empowering LLM-Based Software Vulnerability Detection with Graph Structure and In-Context Learning," Journal of Systems and Software, 2024.
[6]. A. Lekssays et al., "LLMxCPG: Context-Aware Vulnerability Detection Through Code Property Graph-Guided Large Language Models," in Proc. USENIX Security Symposium, 2025.