International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 2, Mar-Apr-2026, ISSN (Online): 2395-566X

Automating the Penetration Testing Flow using Model Context Protocol (MCP) and AI-Orchestrated Security Agents

Navipriyaa M[1], Pooja Ponrani D[1], Prathiba Devi V S[1], Mr. Prasannavenkatesan K[2]
Corresponding author: Mr. Prasannavenkatesan K
[1] School of Computer Science and Engineering, Kumaraguru College Of Technology ,Coimbatore, Tamil Nadu, India
[2] Assistant Professor I, School of Computer Science and Engineering, Kumaraguru College Of Technology ,Coimbatore , Tamil Nadu, India

Abstract- — Penetration testing plays a vital role in identifying security weaknesses in modern computing systems. With the rapid growth of distributed architectures, cloud-native applications, and microservices, traditional penetration testing approaches have become increasingly complex and time-consuming. Although automated tools are widely used, they typically function in isolation and require significant human expertise to coordinate multi-stage attack scenarios This paper presents an enhanced AI-driven penetration testing framework that leverages the Model Context Protocol (MCP) for structured communication and orchestration among multiple intelligent agents. The proposed system integrates reconnaissance, vulnerability assessment, exploitation, privilege escalation, and reporting into a cohesive pipeline. Unlike traditional systems, the framework incorporates contextual reasoning, adaptive decision-making, and dynamic exploit chaining using an AI Planner. Additionally, the system constructs real-time attack graphs and computes risk scores based on vulnerability severity, exploit confidence, and attack depth. Experimental results demonstrate significant improvements in automation efficiency, reduction in manual effort, and higher success rates in identifying complex exploit chains. The proposed framework represents a shift from static automation toward intelligent, adaptive penetration testing systems.

Keywords Automated Penetration Testing, Model Context Protocol, Attack Graphs, AI Security Agents, Exploit Chaining, Cybersecurity Automation.

I. INTRODUCTION

Penetration testing has evolved from a manual, expertise-driven activity into a semi-automated process supported by specialized tools. However, with the rapid adoption of cloud-native architectures, containerized deployments, and microservices-based systems, the attack surface has increased significantly. These environments are highly dynamic, making it difficult for traditional penetration testing approaches to keep pace.

Modern applications expose multiple entry points such as APIs, web interfaces, and distributed services, each of which can introduce vulnerabilities. As a result, security assessments must not only identify individual weaknesses but also understand how these vulnerabilities can be chained together to form complex attack paths.

Although tools like Nmap, Metasploit, SQLMap, and Nikto are effective in their respective domains, they operate in silos and lack coordination. This creates a dependency on human testers to interpret outputs, correlate findings, and decide the next steps.

Recent advancements in Artificial Intelligence, especially Large Language Models (LLMs), have opened new opportunities for automating complex workflows. These models can analyze outputs, maintain context, and make decisions dynamically. When combined with structured communication frameworks like Model Context Protocol (MCP), they enable seamless interaction between tools and intelligent agents.

This research aims to bridge the gap between isolated automation and intelligent orchestration by proposing an AI-driven penetration testing framework that is adaptive, context-aware, and capable of executing multi-stage attack scenarios autonomously.

In addition to the growing complexity of modern systems, another major challenge in penetration testing is the continuous evolution of attack techniques. Attackers increasingly use automated scripts, AI-assisted tools, and sophisticated exploit

chains that can bypass traditional defences. This makes it essential for security testing mechanisms to evolve beyond static and rule-based approaches.

Furthermore, the adoption of DevOps and continuous deployment practices has significantly reduced the time available for manual security assessments. Applications are updated frequently, and vulnerabilities can be introduced at any stage of the development lifecycle. As a result, there is a pressing need for continuous and automated penetration testing solutions that can integrate seamlessly into modern development pipelines.

Another limitation of traditional systems is the lack of context awareness. For instance, identifying a vulnerability without understanding its relationship to other system components may lead to incomplete risk assessment. In real-world scenarios, attackers exploit combinations of vulnerabilities rather than isolated weaknesses.

The proposed approach addresses these challenges by incorporating contextual reasoning, enabling the system to remember previous actions, analyze dependencies, and make informed decisions. This significantly enhances the ability to identify complex, multi-stage attack paths.

II. RELATED WORKS

A. Traditional Penetration Testing Tools

Traditional tools form the backbone of penetration testing but are limited in scope when used independently. For instance, Nmap is effective for network discovery and port scanning, while Metasploit provides a powerful framework for exploitation. SQLMap specializes in detecting and exploiting SQL injection vulnerabilities.

However, these tools lack interoperability and do not provide automated decision-making capabilities. The effectiveness of these tools largely depends on the expertise of the tester, who must manually interpret results and chain together different stages of an attack. This limitation reduces scalability and increases the time required for comprehensive assessments.

Another limitation of traditional tools is their inability to provide feedback-driven execution. Once a scan or exploit is performed, the results are not automatically used to refine subsequent actions. This leads to inefficiencies, as testers must manually analyze outputs and decide the next steps.

Moreover, these tools do not support collaborative execution, where multiple tools can share information in real time. This lack of integration results in fragmented workflows and reduced overall effectiveness.

B. Attack Graph Techniques

Attack graphs are used to model potential paths that an attacker can take to compromise a system. They represent vulnerabilities as nodes and attack transitions as edges, providing a structured view of possible attack scenarios.

Despite their usefulness, traditional attack graphs are static in nature and do not adapt to real-time changes in the environment. They also fail to validate whether the theoretical attack paths are practically exploitable. This disconnect limits their applicability in real-world penetration testing scenarios.
In addition, many attack graph models suffer from state explosion problems, where the number of possible attack paths grows exponentially in large systems. This makes it difficult to analyze and prioritize attack paths efficiently.

Another limitation is the absence of real-time validation mechanisms, which means that generated attack paths may not always be feasible in practice. This reduces their usefulness in active penetration testing scenarios.

C. AI Applications in Cybersecurity

Artificial Intelligence has been widely adopted in cybersecurity for tasks such as intrusion detection, anomaly detection, malware classification, and threat intelligence analysis. Machine learning models can process large volumes of data and identify patterns that are difficult for humans to detect.

However, most AI applications focus on detection rather than active engagement. The use of AI in penetration testing, particularly for orchestrating multi-stage attack workflows, is still in its early stages. Existing systems often lack the ability to dynamically adapt their strategies based on intermediate results.

Despite advancements, AI models in cybersecurity often face challenges related to data quality and interpretability. Models trained on limited datasets may not generalize well to new environments.

Additionally, most AI systems lack interactive capabilities, meaning they cannot actively engage with systems to test vulnerabilities. This highlights the need for AI systems that can both analyze and act.

D. Identified Research Gap

From the analysis of existing approaches, several gaps are evident:

Lack of coordination between different penetration testing tools Absence of real-time adaptive decision-making Limited capability for exploit chaining Insufficient use of contextual memory across stages These challenges highlight the need for a unified framework that combines AI intelligence with structured orchestration. The proposed system addresses these issues by integrating MCP with AI-driven agents to enable intelligent, end-to-end automation.

III. PROPOSED METHODOLOGY

A. System Architecture

The proposed framework consists of multiple interconnected components:

Target Interface → Recon Agent → Scan Agent → Exploit Agent → Privilege Escalation Agent → AI Planner → Attack Graph Engine → Report Generator

Each agent performs a specific function, while the AI Planner acts as the central decision-making unit. MCP enables communication between these components using structured data formats.

The architecture of the proposed system is designed to support modularity, scalability, and intelligent decision-making. Each agent in the system is responsible for a specific stage of the penetration testing lifecycle, ensuring separation of concerns and efficient execution.

The AI Planner acts as the central controller, analyzing outputs from each agent and determining the next course of action. This centralized intelligence allows the system to adapt dynamically rather than following a fixed sequence.

The use of MCP ensures standardized communication between components, enabling seamless data exchange and maintaining contextual continuity throughout the workflow.

Another important aspect of the architecture is its ability to support extensibility. New tools and agents can be integrated into the system without affecting existing components. This ensures that the framework remains adaptable to future advancements in cybersecurity tools.

The modular nature of the system also enables independent updates and maintenance, reducing system downtime and improving reliability.

B. Workflow Explanation

The system follows a sequential yet adaptive workflow:

Step 1: Reconnaissance
Initial information gathering is performed using scanning tools to identify open ports, services, and system details.

Step 2: Vulnerability Scanning
Detected services are analyzed for known vulnerabilities using specialized scanning tools.

Step 3: Exploitation
The system attempts to exploit identified vulnerabilities using appropriate modules.

Step 4: Privilege Escalation
If access is obtained, further techniques are applied to gain higher privileges.

Step 5: Attack Graph Construction
A dynamic attack graph is generated to represent relationships between vulnerabilities.

Step 6: Risk Assessment
Each attack path is evaluated based on multiple factors to determine its severity.

Step 7: Report Generation
Detailed reports are generated including vulnerabilities, exploit paths, and mitigation strategies.

The workflow is not strictly linear but adaptive in nature. Based on intermediate results, the AI Planner can revisit previous steps, skip unnecessary actions, or prioritize high-risk vulnerabilities.

- During reconnaissance, the system gathers both passive and active intelligence.
- In vulnerability scanning, multiple tools may be used in parallel to improve coverage.
- Exploitation is guided by AI-based prioritization rather than random attempts.
- Privilege escalation involves both automated scripts and heuristic-based techniques.

This adaptability significantly improves efficiency and success rates.

Each stage of the workflow is enhanced with feedback loops, allowing the system to refine its actions based on previous outcomes. For example, if an exploit attempt fails, the system can analyze the reason for failure and attempt alternative strategies.

Additionally, the workflow incorporates decision checkpoints, where the AI Planner evaluates multiple possible actions and selects the most effective one based on risk and probability.

C. MCP-Based Orchestration
MCP acts as the backbone of the system by enabling:

- Standardized communication protocols
- Context persistence across agents
- Decoupling of tools and decision logic

Each interaction follows a structured format containing:

- Input parameters
- Execution results
- Metadata (confidence, timestamps, status)

This structured communication ensures that the AI Planner can interpret results accurately and make informed decisions.

Another key advantage of MCP is its ability to ensure data consistency and integrity across all agents. By standardizing communication formats, the system minimizes errors caused by incompatible data representations.

Furthermore, MCP enables asynchronous communication, allowing agents to operate independently while still maintaining coordination.

D. Attack Graph Modeling
The attack graph is represented as:

$$G = (V, E)$$

Where:
V represents vulnerabilities
E represents exploit transitions

Each node corresponds to a potential weakness, while edges represent possible attack paths.

The exploit chain score is calculated based on probability and impact, enabling prioritization of high-risk attack paths.

Attack graphs provide a visual and analytical representation of how vulnerabilities are interconnected. In this system, the graph is dynamically updated as new vulnerabilities are discovered.

Each node represents a vulnerability with associated attributes such as severity, exploitability, and confidence level. Edges represent possible transitions between vulnerabilities, forming a chain of attacks.

The dynamic nature of the graph allows the system to:

- Identify critical attack paths in real time
- Prioritize high-impact vulnerabilities
- Avoid redundant or low-probability attack attempts

This enhances both efficiency and accuracy in penetration testing.

The system also incorporates weight-based prioritization, where each edge in the graph is assigned a cost representing the effort required to exploit a vulnerability. This allows the AI Planner to identify optimal attack paths using cost minimization strategies.

Additionally, the attack graph can be used for post-analysis, helping security teams understand how vulnerabilities are interconnected and how attacks propagate through the system.

E. Risk Scoring Mechanism
The risk associated with a vulnerability is computed using multiple parameters:

- CVSS score
- Exploit reliability
- Depth of attack chain

This multi-factor evaluation provides a more realistic assessment compared to traditional severity scoring.

Traditional risk scoring methods rely heavily on static metrics such as CVSS scores. However, these do not always reflect real-world exploitability.

The proposed system introduces a multi-factor risk scoring model that considers:

- Severity of the vulnerability
- Likelihood of successful exploitation
- Position within the attack chain
- Potential impact on the system

By combining these factors, the system produces a more realistic and actionable risk assessment, helping organizations prioritize remediation efforts effectively.

The risk scoring model also considers environment-specific factors, such as system configuration and network exposure. This ensures that the calculated risk reflects real-world conditions rather than generic severity levels.

The inclusion of multiple parameters allows for fine-grained risk assessment, enabling better prioritization of mitigation efforts.

F. Exploit Confidence Estimation
Exploit success is not always guaranteed. Therefore, the system estimates confidence levels using:

- Tool feedback
- Service compatibility
- Historical exploit performance

This allows the AI Planner to prioritize more reliable attack strategies.

The confidence estimation mechanism also incorporates dynamic updates, where confidence scores are adjusted based on real-time feedback from tool execution.

This adaptive approach improves the accuracy of predictions and helps the system focus on high-probability attack paths.

IV. EXPERIMENTAL RESULTS AND EVALUATION

The experimental evaluation demonstrates that the proposed system significantly outperforms traditional approaches in terms of automation and efficiency.

The system was tested in a controlled environment with multiple vulnerable machines and real-world applications. Results show that the AI-driven framework can identify complex multi-stage attack paths that are often missed in manual testing.

Additionally, the system reduces execution time by automating decision-making and minimizing redundant actions. The ability to dynamically adapt to different environments further enhances its effectiveness.

A. Setup
The system was evaluated in a controlled environment using:

- Kali Linux testing platform
- Multiple vulnerable virtual machines
- Web applications such as OWASP Juice Shop
- Over 100 automated testing runs

B. Performance Analysis
The proposed system was compared with manual testing and traditional automated tools.

Key observations include:

- Significant reduction in execution time
- Improved detection of multi-stage attack paths
- Higher level of automation
- Reduced dependency on manual intervention

In addition to performance improvements, the system demonstrates consistency in results across multiple test runs. This indicates that the AI-driven approach is reliable and not dependent on random factors.

The evaluation also highlights the system's ability to adapt to different types of targets, including web applications and virtual machines with varying configurations.

Furthermore, the system reduces the likelihood of false positives, as vulnerabilities are validated through actual exploitation attempts rather than solely relying on scan results.

V. DISCUSSION

The results indicate that integrating AI with penetration testing workflows can significantly improve efficiency and effectiveness. The system demonstrates the ability to adapt to different environments and dynamically adjust its strategy based on intermediate findings.

Unlike traditional approaches, the proposed framework not only identifies vulnerabilities but also evaluates their real-world exploitability. The use of attack graphs enhances understanding of complex attack scenarios.

However, certain limitations exist:

- Dependence on tool compatibility
- Requirement for structured inputs
- Potential ethical concerns in misuse

These challenges highlight the need for controlled deployment and further refinement.

Another important observation is that the integration of AI enables the system to simulate attacker behavior more realistically. Instead of following predefined rules, the system dynamically adjusts its strategy based on system responses.

The use of attack graphs provides a holistic view of system security, allowing security professionals to understand not only individual vulnerabilities but also their combined impact.

However, the reliance on AI introduces challenges related to explainability, as decision-making processes may not always be transparent. Addressing this issue is essential for building trust in AI-driven systems.

Additionally, ethical concerns must be carefully managed to ensure that such systems are used only in authorized environments.

V. CONCLUSION AND FUTURE WORK

This paper introduces an AI-driven framework for automating penetration testing using MCP-based orchestration. By integrating multiple tools with intelligent decision-making, the system achieves higher automation and improved exploit detection.

The research demonstrates that combining AI reasoning with structured workflows can transform penetration testing into a more efficient and adaptive process.

Future work will focus on:

- Incorporating reinforcement learning for improved decision-making
- Integration with DevSecOps pipelines
- Expanding support for diverse attack vectors
- Enhancing real-time threat simulation capabilities

The proposed framework demonstrates that combining AI with structured orchestration can significantly enhance the effectiveness of penetration testing. By enabling adaptive decision-making and real-time analysis, the system overcomes many limitations of traditional approaches. The integration of multiple components into a unified framework ensures that the entire penetration testing lifecycle is automated while maintaining high accuracy and efficiency. This research highlights the potential of AI-driven systems to transform cybersecurity practices and provides a foundation for future advancements in autonomous security testing.

REFERENCES

1. Z. Ezetta and W.-C. Feng, “PentestMCP: A toolkit for agentic penetration testing,” arXiv Preprint arXiv:2510.03610, Oct. 2025. https://arxiv.org/abs/2510.03610
2. H. Kong, D. Hu, J. Ge, L. Li, T. Li, and B. Wu, “VulnBot: Autonomous penetration testing for a multi-agent collaborative framework,” arXiv preprint, Jan. 2025. https://arxiv.org/abs/2501.13411v1
3. K. Nakano, R. Feyyazi, S. J. Yang, and M. Zuzak, “Guided reasoning in LLM-driven penetration testing using structured attack trees,” arXiv preprint, Sep. 2025. https://arxiv.org/abs/2509.07939v1
4. M. C. Ghanem, T. M. Chen, and E. G. Nepomuceno, “Hierarchical reinforcement learning for efficient and effective automated penetration testing of large networks,” J. Intell. Inf. Syst., vol. 60, pp. 281–303, 2023. https://doi.org/10.1007/s10844-022-00738-0
5. N. V. Hung and N. T. Cong, “Applying reinforcement learning in automated penetration testing,” J. Sci. Technol. Inf. Security, vol. 3, no. 17, pp. 61–77, Apr. 2023. https://doi.org/10.54654/isj.v3i17.876
6. “xOffense: An AI-driven autonomous penetration testing framework with offensive knowledge-enhanced LLMs,” EmergentMind preprint, Sep. 2025. https://www.emergentmind.com/papers/2509.13021v1
7. M. Patil, D. Thakare, A. Bhure, S. Kaundanyapure, and A. Mune, “An AI-based approach for automating penetration testing,” Int. J. Res. Appl. Sci. Eng. Technol., 2024. https://doi.org/10.22214/ijraset.2024.61113
8. K. C. Bannari, S. V. Bannari, A. A. Bannari, and L. S. Bannari, “AI in automation of penetration testing,” J. Cyber Security, Privacy Issues and Challenges, vol. 3, no. 2, pp. 16–22, 2024. https://matjournals.net/engineering/index.php/JCSPIC/article/view/576
9. S. Krishna Jampani, “Revolutionizing penetration testing: AI-powered automation for enterprise security,” Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol., vol. 10, no. 6, pp. 1562–1569, Dec. 2024. https://doi.org/10.32628/CSEIT241061201
10. “An automated approach to web offensive security,” Comput. Commun., vol. 195, pp. 248–261, Nov. 2022. https://doi.org/10.1016/j.comcom.2022.08.018
11. “Penetration testing: Taxonomies, trade-offs, and adaptive strategies,” Comput. Electr. Eng., 2025. https://doi.org/10.1016/j.compeleceng.2025.110686
12. “Autonomous penetration testing using reinforcement learning: A review and perspectives,” Expert Syst. Appl., 2025. https://doi.org/10.1016/j.eswa.2025.130219