AdaptiveFuzz: LLM-Based Adaptive Fuzzing for Vulnerability Discovery

Vigneshwar Sundararajan
vigneshwar_ks@ucf.edu
University of Central Florida
Orlando, Florida, USA

ABSTRACT

The increasing complexity of modern software systems necessitates advanced methodologies for vulnerability discovery beyond conventional penetration testing and static fuzzing techniques. This project introduces a Model Context Protocol (MCP)-based adaptive fuzzing framework, integrating Large Language Models (LLMs) to automate security assessments dynamically. The system employs an MCP Coordination Server to orchestrate interactions between an LLM-driven decision engine, terminal access tools, and web search utilities. The Exploit Discovery Recipe Repository structures penetration testing methodologies, while CVE documentation and data sources assist in identifying known vulnerabilities. The framework continuously loops through state tracking, tool execution, and data-driven refinement, autonomously adapting test strategies based on real-time feedback. By leveraging MCP, the approach ensures contextual awareness, tool optimization, and intelligent fuzzing. The final output includes a comprehensive vulnerability report, detailing discovered weaknesses and attack vectors. This framework enhances efficiency, accuracy, and scalability in automated security testing, bridging the gap between human expertise and AI-driven automation.

1 INTRODUCTION

Software security remains a critical challenge as traditional vulnerability discovery methods often fail to detect sophisticated attack vectors. Conventional penetration testing and static fuzzing rely on predefined input patterns, making them ineffective against complex vulnerabilities that emerge under dynamic conditions. As modern software architectures evolve, security assessments demand adaptive methodologies capable of emulating human-like attack strategies and adjusting to real-time scenarios. However, integrating intelligent automation into penetration testing remains a challenge due to the lack of structured coordination between decision-making models, security tools, and vulnerability databases.

To address this limitation, we propose a Model Context Protocol (MCP)-based adaptive fuzzing framework, leveraging Large Language Models (LLMs) for intelligent input generation and automated security testing. The MCP Coordination Server acts as the central controller, managing interactions between the LLM-driven decision engine, terminal access tools, and external vulnerability sources. The Exploit Discovery Recipe Repository structures penetration testing workflows, while CVE databases provide real-time insights into known security flaws. The state manager continuously tracks progress, ensuring that testing strategies evolve dynamically. Through real-time feedback loops, the system adapts its fuzzing methodologies, automating security assessments with minimal manual intervention.

Evaluation Metrics: To assess the effectiveness of the proposed framework, multiple evaluation metrics can be employed. Vulnerability Detection Rate (VDR) will measure the number of unique vulnerabilities discovered compared to traditional methods. False Positive Rate (FPR) will evaluate the accuracy of the detected vulnerabilities, ensuring minimal false alerts. Execution Efficiency (EE) will track the time required to identify security flaws, highlighting the framework's performance in real-world scenarios. Additionally, Adaptive Learning Improvement (ALI) will quantify how well the system refines its fuzzing strategies over iterative test cycles. These metrics will provide a comprehensive assessment of the system's reliability, accuracy, and efficiency in automated security testing.

2 PROBLEM STATEMENT

Traditional vulnerability discovery techniques, such as manual penetration testing and conventional fuzzing tools, struggle to identify complex security flaws that arise in unpredictable and evolving software environments. These methods heavily rely on predefined input patterns, limiting their effectiveness against adaptive or deeply embedded vulnerabilities. Furthermore, static fuzzing approaches lack the ability to contextually refine attack strategies based on real-time feedback, leading to inefficiencies and overlooked security gaps. The absence of intelligent decision-making in automated security assessments necessitates a dynamic and learning-based approach to vulnerability discovery. To address this, an LLM-driven adaptive fuzzing framework, integrated with Model Context Protocol (MCP), is proposed to enhance penetration testing efficiency. This framework combines LLM-powered intelligent input generation, real-time vulnerability tracking, and automated tool coordination, ensuring a scalable, adaptive, and intelligent approach to security testing in modern software infrastructures.

3 RELATED WORK

(1) **Model Context Protocol (MCP) in Security Automation:** MCP provides a structured way to integrate AI-driven decision-making with security tools. Existing works, such as MCP_Security [1], demonstrate how MCP improves tool coordination, which our framework extends for adaptive fuzzing.

(2) **LLMs in Automated Security Analysis:** LLMs have been applied in security tasks like malware detection and exploit generation, but their use in real-time fuzzing remains limited. Our approach integrates LLM-based test input generation with MCP-driven execution to enhance security automation [2].

(3) **AI-Powered Fuzzing Techniques:** While traditional fuzzers like AFL and LibFuzzer focus on static mutations, recent AI-enhanced fuzzing methods improve adaptability. Our framework advances this by combining LLM-driven fuzzing strategies with real-time feedback loops, optimizing test case generation [3].

(4) **Cyber Threat Intelligence for Pentesting:** Threat intelligence tools like Shodan [4] and Maigret [5] provide valuable insights into exposed vulnerabilities. By integrating these into MCP, our system automates context-aware penetration testing, leveraging real-time data for targeted security assessments.

4 METHODOLOGY

The proposed LLM-based adaptive fuzzing framework leverages Model Context Protocol (MCP) to facilitate structured communication between AI-driven decision-making, automated penetration testing tools, and external security data sources. The system is designed to dynamically refine its testing approach based on real-time analysis, state tracking, and intelligent tool execution. The architecture consists of two primary components:

MCP Client: Handles LLM integration, terminal interactions, and web-based data retrieval.

MCP Coordination Server: Manages tool selection, execution state tracking, and result analysis to optimize fuzzing efficiency.

4.1 Architecture Overview

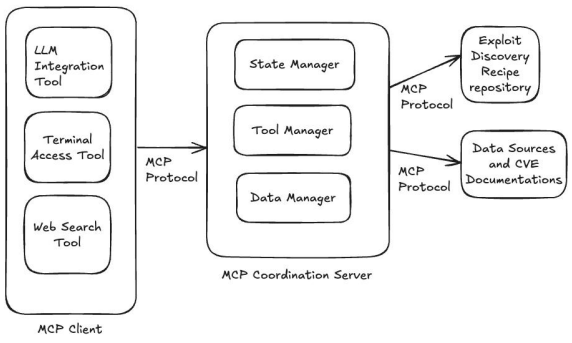

Figure 1: AdaptiveFuzz Architecture

The MCP Client communicates with the MCP Coordination Server through MCP Protocol, ensuring seamless tool execution and state-aware fuzzing operations. The framework integrates various security resources, including:

Exploit Discovery Recipe Repository. Contains methodical pentesting workflows, guiding how fuzzing should be performed (e.g., network scanning, enumeration, privilege escalation).

Data Sources & CVE Documentation. Stores known vulnerabilities (CVEs) and real-world exploit intelligence, helping LLM refine attack strategies dynamically.

4.2 Execution Flow

(1) **Initiation & LLM Query:** The MCP Coordination Server requests input from the LLM Integration Tool, querying the next fuzzing step. The LLM (e.g., Qwen2.5-Coder or Llama) analyzes prior results and suggests the next testing approach.

(2) **Tool Selection & Command Execution:** The Tool Manager selects the appropriate pentesting tool based on LLM recommendations. The Terminal Access Tool executes the command and fetches real-time security data.

(3) **State Tracking & Dynamic Adaptation:** The State Manager continuously updates progress and execution results. The Data Manager stores collected fuzzing outputs for future refinements.

(4) **Feedback Loop & Intelligence Enhancement:** The MCP Client submits output back to the LLM, refining further test strategies. The loop continues until a vulnerability is identified or the system detects an infinite loop scenario, triggering termination and report generation.

(5) **Web Search & External Intelligence:** If the LLM requires external intelligence, the Web Search Tool queries threat intelligence sources (e.g., GitHub, CVE databases, security forums, etc.).

4.3 Output & Reporting

Upon completion, the framework generates a structured vulnerability report, highlighting Identified vulnerabilities & exploit paths, Associated CVEs & threat intelligence, Recommended mitigation strategies. This intelligent fuzzing approach ensures higher vulnerability detection accuracy, automated decision-making, and reduced manual effort, making it scalable for real-world security assessments.

5 PRELIMINARY RESULTS

The core framework setup is complete, integrating Model Context Protocol (MCP) for LLM-driven penetration testing. The system now supports automated fuzzing, tool execution, and real-time vulnerability analysis, ensuring an adaptive approach to security assessments.

Achievements: The LLM setup using Ollama with models like Qwen2.5-Coder has been tested for generating fuzzing inputs. The MCP Coordination Server successfully manages execution, state tracking, and tool orchestration. Security tools are integrated within the MCP framework, allowing dynamic adjustments based on real-time feedback.

Challenges & Refinements: A key challenge was function calling issues in Ollama, which was resolved by shifting to MCP-based execution. The State Manager has been optimized to prevent infinite loops, ensuring efficient fuzzing cycles.

Next Steps: The focus now is on refining execution loops, automating report generation, and testing the framework in real-world pentesting scenarios to validate its effectiveness in vulnerability discovery.

REFERENCES

[1] fr0gger. MCP Security. GitHub repository. Available at https://github.com/fr0gger/MCP_Security.
[2] Model Context Protocol. Introduction to Model Context Protocol. Available at https://modelcontextprotocol.io/introduction.
[3] punkpeye. Awesome MCP Servers. GitHub repository. Available at https://github.com/punkpeye/awesome-mcp-servers/blob/main/README.md.
[4] BurtTheCoder. MCP Shodan. GitHub repository. Available at https://github.com/BurtTheCoder/mcp-shodan.
[5] BurtTheCoder. MCP Maigret. GitHub repository. Available at https://github.com/BurtTheCoder/mcp-maigret.