

การพัฒนารอบการทำงานอัตโนมัติบนแพลตฟอร์ม n8n สำหรับการทดสอบเจาะระบบ โดยใช้แบบจำลองภาษาขนาดใหญ่และหลักฐานเชิงโครงสร้าง

อรรณพ หมั่นสกุล*

*คณะวิศวกรรมศาสตร์และเทคโนโลยี สถาบันการจัดการปัญญาภิวัฒน์, นนทบุรี, ประเทศไทย

*อีเมลผู้ประพันธ์บรรณกิจ: annopmon@pim.ac.th

รับต้นฉบับ: 24 พฤษภาคม 2569; รับบทความฉบับแก้ไข: 20 กรกฎาคม 2569; ตอบรับบทความ: 20 กรกฎาคม 2569

เผยแพร่ออนไลน์: 27 สิงหาคม 2569

บทคัดย่อ

งานวิจัยนี้มีวัตถุประสงค์เพื่อพัฒนาและประเมินกรอบการทำงานอัตโนมัติแบบขับเคลื่อนด้วยหลักฐานสำหรับการทดสอบเจาะระบบ โดยใช้ประยุกต์ใช้แบบจำลองภาษาขนาดใหญ่ร่วมกับกลไกการตรวจสอบภาวะหลอนของโมเดล เพื่อลดข้อจำกัดของการทดสอบเจาะระบบแบบเดิมที่ต้องอาศัยผู้เชี่ยวชาญสูง ใช้เครื่องมือหลายประเภทแยกจากกัน และมีความไม่สม่ำเสมอของผลลัพธ์ระหว่างรอบการทดสอบ กรอบการทำงานที่พัฒนาขึ้นใช้แพลตฟอร์ม n8n เป็นศูนย์กลางในการประสานกระบวนการทำงานอัตโนมัติ ภายใต้สภาพแวดล้อม Docker container stack โดยจำแนกเป้าหมายการทดสอบออกเป็น 4 ประเภท ได้แก่ ระบบบริการเครือข่าย เว็บไซต์แอปพลิเคชันสมัยใหม่ ระบบ API Security และระบบที่เกี่ยวข้องกับการรั่วไหลของข้อมูลลับหรือการตั้งค่าที่ไม่ปลอดภัย จากนั้นระบบจะเลือกกลยุทธ์การทดสอบให้เหมาะสมกับแต่ละประเภทของเป้าหมาย รวบรวมผลลัพธ์จากเครื่องมือให้อยู่ในรูปแบบหลักฐานเชิงโครงสร้าง และจัดเก็บในฐานข้อมูล PostgreSQL ก่อนนำเข้าสู่กระบวนการวิเคราะห์ด้วย GPT-4o ผ่านการออกแบบคำสั่งนำเข้าและการกำหนดผลลัพธ์แบบมีโครงสร้าง โดยไม่มีการปรับแต่งโมเดลเฉพาะงาน ทั้งนี้ ได้พัฒนากลไกตรวจสอบภาวะหลอนของโมเดลแบบอิงหลักฐาน เพื่อตรวจสอบว่าข้อค้นพบทุกประเด็นต้องมีหลักฐานจากเครื่องมือหรือบันทึกการทำงานรองรับก่อนนำเข้าสู่รายงานขั้นสุดท้าย การทดลองดำเนินการกับระบบเป้าหมายที่มีช่องโหว่โดยเจตนา 4 ระบบ ได้แก่ Metasploitable3, OWASP Juice Shop, OWASP crAPI และ OWASP WrongSecrets โดยทดสอบซ้ำระบบละ 6 รอบ รวมทั้งสิ้น 24 รอบการทดลอง ผลการทดลองแสดงให้เห็นว่า กรอบการทำงานสามารถรวบรวมหลักฐานเชิงโครงสร้างได้อย่างสม่ำเสมอ โดยมีค่าความสม่ำเสมอของหลักฐานเชิงโครงสร้างสูงกว่าร้อยละ 97 ทุกเป้าหมาย มีอัตราความแม่นยำในการตรวจสอบข้อค้นพบจากแบบจำลองภาษาขนาดใหญ่อยู่ระหว่างร้อยละ 83.02 ถึง 90.32 และมีคะแนนความสามารถในการตรวจสอบย้อนกลับเฉลี่ยอยู่ระหว่างร้อยละ 89.83 ถึง 96.83 ผลลัพธ์ดังกล่าวสะท้อนให้เห็นว่า การผสานกระบวนการทำงานอัตโนมัติ การจัดการหลักฐานเชิงโครงสร้าง และการตรวจสอบผลลัพธ์ของแบบจำลองภาษาขนาดใหญ่แบบอิงหลักฐาน สามารถเพิ่มความน่าเชื่อถือ ความสม่ำเสมอ และความสามารถในการตรวจสอบย้อนกลับของการทดสอบเจาะระบบอัตโนมัติได้อย่างมีประสิทธิภาพ

คำสำคัญ: การทดสอบเจาะระบบอัตโนมัติ การลดภาวะหลอนของโมเดล แบบจำลองภาษาขนาดใหญ่ การวิเคราะห์ช่องโหว่ การทำงานอัตโนมัติแบบเวิร์กโฟลว์

Development of an n8n-Based Automated Framework for Penetration Testing Using Large Language Models and Structured Evidence

Annop Monsakul*

**Faculty of Engineering and Technology, Panyapiwat Institute of Management, Nonthaburi, Thailand*

*Corresponding author e-mail: annopmon@pim.ac.th

Received: 24 May 2026; Revised: 20 July 2026; Accepted: 20 July 2026

Published online: 27 August 2026

Abstract

This research aims to develop and evaluate an evidence-driven automated framework for penetration testing by integrating large language models with hallucination-aware verification. The proposed framework addresses key limitations of traditional penetration testing, including heavy reliance on expert analysts, fragmented use of multiple security tools, heterogeneous tool outputs, and inconsistent results across repeated testing cycles. The framework employs n8n as the central workflow orchestration platform within a Docker container stack. It classifies assessment targets into four categories: network service systems, modern web applications, API security systems, and systems involving secret leakage or insecure configuration. Based on the target category, the framework selects an appropriate testing strategy, collects outputs from relevant security assessment modules, transforms them into structured evidence, and stores them in a PostgreSQL database. GPT-4o is then used through prompt engineering and structured output, without model fine-tuning, to support vulnerability interpretation, evidence correlation, and risk prioritization. To reduce the risk of unsupported or hallucinated findings, the framework incorporates a hallucination-aware verification mechanism requiring every reported finding to be supported by tool-generated evidence or execution logs before inclusion in the final report. The experiment was conducted using four deliberately vulnerable systems: Metasploitable3, OWASP Juice Shop, OWASP crAPI, and OWASP WrongSecrets. Each target was tested six times, resulting in 24 campaign runs. The experimental results indicate that the proposed framework consistently collected structured evidence, achieving structured-evidence consistency above 97% across all targets. The verification pass rate of LLM-generated findings ranged from 83.02% to 90.32%, while the average traceability score ranged from 89.83% to 96.83%. These results suggest that combining workflow automation, structured evidence management, and evidence-based verification of large language model outputs can improve the reliability, consistency, and traceability of automated penetration testing.

Keywords: Automated penetration testing, Hallucination mitigation, Large language models, Vulnerability analysis, Workflow automation

บทนำ

ปัจจุบันระบบสารสนเทศขององค์กรมีความซับซ้อนและเชื่อมโยงกันมากขึ้น ทั้งในระดับบริการเครือข่าย เว็บแอปพลิเคชัน ระบบ API และองค์ประกอบที่เกี่ยวข้องกับข้อมูลลับหรือการตั้งค่าระบบ ส่งผลให้การประเมินช่องโหว่และการทดสอบเจาะระบบไม่สามารถใช้แนวทางเดียวกันกับทุกระบบได้อย่างมีประสิทธิภาพ [1], [2], [3], [4] เนื่องจากระบบเป้าหมายแต่ละประเภทมีลักษณะของช่องโหว่ วิธีการตรวจสอบ และหลักฐานที่ใช้ประกอบการวิเคราะห์แตกต่างกัน หากใช้กระบวนการทดสอบแบบตายตัว อาจทำให้การตรวจสอบไม่ครอบคลุม ไม่สอดคล้องกับบริบทของระบบ และยากต่อการจัดลำดับความเสี่ยงอย่างถูกต้อง

ในบริบทของการตรวจสอบบริการเครือข่าย เครื่องมือประเภท network scanning เช่น Nmap และ Metasploit ถูกใช้เพื่อค้นหาพอร์ตที่เปิดอยู่ ระบุ service banner ตรวจสอบเวอร์ชันของบริการ และสนับสนุนการประเมินช่องโหว่ระดับระบบ [5], [6], [7] สำหรับเว็บแอปพลิเคชัน เครื่องมือและแนวทางตาม OWASP Web Security Testing Guide และ OWASP ZAP ถูกใช้เพื่อตรวจสอบพฤติกรรมของ HTTP request/response, route, header, cookie, form และช่องโหว่ที่เกี่ยวข้องกับเว็บแอปพลิเคชัน [2], [3], [8], [9] ในส่วนของระบบ API การทดสอบจำเป็นต้องพิจารณา endpoint, HTTP method, status code, authorization behavior และข้อกำหนดตาม OWASP API Security Top 10 เพื่อประเมินช่องโหว่ที่เกิดจากการควบคุมสิทธิ์หรือการออกแบบ API ที่ไม่ปลอดภัย [1], [10] นอกจากนี้ ระบบที่เกี่ยวข้องกับข้อมูลลับหรือการตั้งค่าที่ไม่ปลอดภัย เช่น OWASP WrongSecrets ใช้สำหรับประเมินกรณี secret leakage, token, credential และ configuration ที่อาจนำไปสู่ความเสี่ยงด้านความมั่นคงปลอดภัย [11]

การประยุกต์ใช้แบบจำลองภาษาขนาดใหญ่ หรือ Large Language Model (LLM) ในงานด้านความมั่นคงปลอดภัยไซเบอร์มีศักยภาพในการช่วยตีความผลการสแกน วิเคราะห์บริบท จัดระดับความรุนแรง เชื่อมโยงข้อค้นพบ และสร้างรายงานในรูปแบบที่มีโครงสร้าง [12], [13], [14], [15], [16], [17], [18], [19], [20] อย่างไรก็ตาม การใช้ LLM ในบริบทการทดสอบเจาะระบบยังมีความเสี่ยงจากภาวะหลอนของโมเดล (hallucination) ซึ่งอาจทำให้เกิดข้อสรุปที่ไม่มีหลักฐานรองรับ [14], [15], [21], [22], [23], [24] เช่น การระบุช่องโหว่ที่ไม่ปรากฏจริง การเสนอ exploit ที่ไม่สัมพันธ์กับข้อมูลจากระบบ หรือการให้ข้อเสนอแนะที่เกินกว่าหลักฐานที่ตรวจพบ ดังนั้น การนำ LLM มาใช้ในงานด้านนี้จำเป็นต้องมีกลไกควบคุมคุณภาพของผลลัพธ์ โดยให้ข้อค้นพบทุกประเด็นสามารถตรวจสอบย้อนกลับไปยังหลักฐานจากเครื่องมือหรือบันทึกการทำงานได้

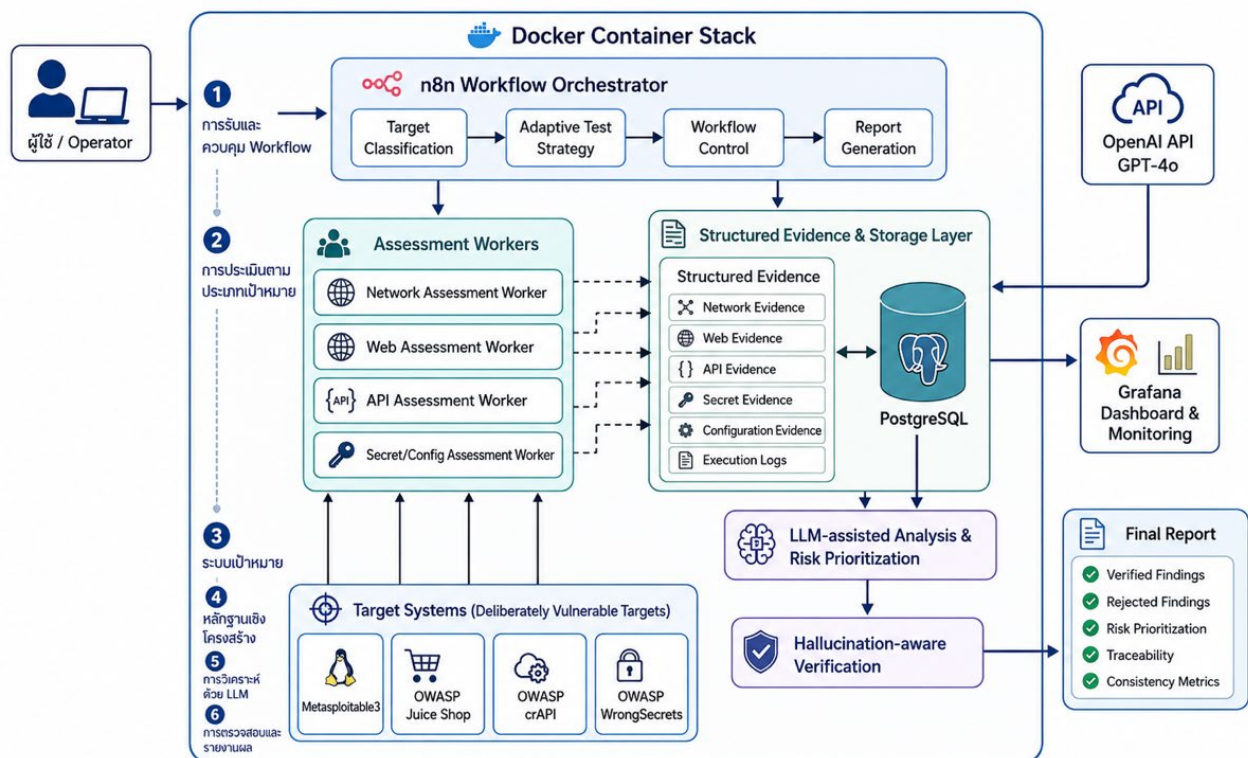
งานวิจัยนี้จึงมุ่งพัฒนารอบการทำงานอัตโนมัติแบบขับเคลื่อนด้วยหลักฐานสำหรับการทดสอบเจาะระบบ โดยใช้แพลตฟอร์ม n8n เป็นศูนย์กลางในการประสานกระบวนการทำงาน ภายใต้สภาพแวดล้อม Docker container stack กรอบการทำงานดังกล่าวจำแนกเป้าหมายออกเป็น 4 ประเภท ได้แก่ ระบบบริการเครือข่าย เว็บแอปพลิเคชัน ระบบ API Security และระบบที่เกี่ยวข้องกับข้อมูลลับรั่วไหล/การตั้งค่าที่ไม่ปลอดภัย (secret/misconfiguration) จากนั้นเลือกกลยุทธ์การทดสอบให้เหมาะสมกับแต่ละประเภท รวบรวมผลลัพธ์ให้อยู่ในรูปแบบหลักฐานเชิงโครงสร้าง วิเคราะห์ด้วย GPT-4o [21], [25], [26] และตรวจสอบข้อค้นพบด้วยกลไกตรวจสอบภาวะหลอนของโมเดลแบบอิงหลักฐานก่อนนำเข้าสู่รายงานขั้นสุดท้าย

จากปัญหาและแนวทางดังกล่าว งานวิจัยนี้กำหนดคำถามวิจัยหลักว่า กรอบการทำงานอัตโนมัติแบบขับเคลื่อนด้วยหลักฐานที่ใช้ LLM ร่วมกับกลไกตรวจสอบภาวะหลอนของโมเดลแบบอิงหลักฐาน สามารถเพิ่มความสม่ำเสมอ ความน่าเชื่อถือ และความสามารถในการตรวจสอบย้อนกลับของการทดสอบเจาะระบบอัตโนมัติได้หรือไม่ โดยมีสมมติฐานว่า หากระบบสามารถจำแนกประเภทเป้าหมาย เลือกวิธีทดสอบที่เหมาะสม จัดเก็บหลักฐานเชิงโครงสร้าง และบังคับให้ข้อค้นพบจาก LLM ต้องมีหลักฐานรองรับก่อนนำเข้าสู่รายงานขั้นสุดท้าย จะช่วยลดความเสี่ยงจากภาวะหลอนของโมเดล และทำให้ผลลัพธ์ของการประเมินช่องโหว่มีความสม่ำเสมอและตรวจสอบได้มากขึ้น

ความสำคัญของงานวิจัยนี้ประกอบด้วยกรอบการออกแบบกรอบการทำงานอัตโนมัติบนแพลตฟอร์ม n8n สำหรับการทดสอบเจาะระบบแบบขับเคลื่อนด้วยหลักฐาน การพัฒนาการจัดการหลักฐานเชิงโครงสร้างและการเชื่อมโยงรหัสหลักฐานเพื่อเพิ่มความสามารถในการตรวจสอบย้อนกลับของข้อค้นพบ และการประยุกต์ใช้กลไกตรวจสอบภาวะหลอนของโมเดลแบบอิงหลักฐานเพื่อคัดกรองข้อค้นพบจากแบบจำลองภาษาขนาดใหญ่ก่อนนำเข้าสู่รายงานขั้นสุดท้าย

ระเบียบวิธีวิจัย

การวิจัยนี้เป็นการวิจัยเชิงทดลองและพัฒนาเชิงระบบ โดยมีวัตถุประสงค์เพื่อออกแบบ พัฒนา และประเมินกรอบการทำงานอัตโนมัติสำหรับการทดสอบเจาะระบบแบบขับเคลื่อนด้วยหลักฐาน โดยประยุกต์ใช้แบบจำลองภาษาขนาดใหญ่ร่วมกับกลไกการตรวจสอบภาวะหลอนของโมเดล ภายใต้สภาพแวดล้อมห้องปฏิบัติการที่ควบคุมได้ การดำเนินการวิจัยมุ่งเน้นให้ระบบสามารถจำแนกประเภทของเป้าหมาย เลือกกลยุทธ์การทดสอบที่เหมาะสม รวบรวมผลลัพธ์จากเครื่องมือให้อยู่ในรูปแบบหลักฐานเชิงโครงสร้าง วิเคราะห์และจัดลำดับความเสี่ยงด้วย LLM และตรวจสอบความถูกต้องของข้อค้นพบก่อนนำเข้าสู่รายงานขั้นสุดท้าย ทั้งนี้ โครงสร้างกรอบแนวคิดของระบบประกอบด้วย 6 ชั้นหลัก ได้แก่ Target classification, Adaptive test strategy, Structured evidence collection, LLM-assisted analysis and risk prioritization, Hallucination-aware verification และ Reporting & consistency evaluation ตามรูปที่ 1



รูปที่ 1: สถาปัตยกรรมของกรอบการทำงานอัตโนมัติแบบขับเคลื่อนด้วยหลักฐานสำหรับการทดสอบเจาะระบบ

1. การออกแบบภาพรวมของการวิจัย

การดำเนินการวิจัยแบ่งออกเป็น 6 ขั้นตอนหลัก ได้แก่ 1) การออกแบบสถาปัตยกรรมระบบและกรอบการทำงานอัตโนมัติแบบขับเคลื่อนด้วยหลักฐาน 2) การจัดเตรียมสภาพแวดล้อมการทดลองภายใต้ Docker container stack 3) การพัฒนาระบบการทำงานอัตโนมัติบนแพลตฟอร์ม n8n สำหรับควบคุมกระบวนการทดสอบ 4) การพัฒนาโมดูลรวบรวมหลักฐานเชิงโครงสร้าง การวิเคราะห์ด้วยแบบจำลองภาษาขนาดใหญ่ และกลไกตรวจสอบภาวะหลอนของโมเดลแบบอิงหลักฐาน 5) การทดลองซ้ำกับระบบเป้าหมาย 4 ประเภท รวม 24 รอบการทดลอง และ 6) การวิเคราะห์ผลลัพธ์ในด้านความสม่ำเสมอ ความน่าเชื่อถือ และความสามารถในการตรวจสอบย้อนกลับ ทั้งนี้ ภาพรวมของสถาปัตยกรรมระบบแสดงในรูปที่ 1 และขั้นตอนการดำเนินการวิจัยแสดงในรูปที่ 3 เพื่อให้เห็นลำดับการทำงานของกรอบการวิจัยอย่างเป็นระบบ

2. สถาปัตยกรรมของระบบที่พัฒนา

สถาปัตยกรรมของระบบถูกออกแบบให้ทำงานภายใต้ Docker container stack เพื่อให้สามารถควบคุมสภาพแวดล้อม แยกองค์ประกอบของระบบออกจากกัน และทำซ้ำการทดลองได้อย่างเป็นระบบ องค์ประกอบหลักของระบบประกอบด้วย n8n ทำหน้าที่เป็น Workflow orchestrator PostgreSQL ทำหน้าที่เป็นฐานข้อมูลกลางสำหรับจัดเก็บหลักฐานและผลการทดลอง Assessment workers เช่น Nmap, ZAP, Metasploit, Custom script สำหรับประเมินช่องโหว่ตามประเภทของเป้าหมาย Grafana สำหรับติดตามสถานะการทดลอง และ GPT-4o ผ่าน OpenAI API สำหรับสนับสนุนการวิเคราะห์ข้อค้นพบด้วย LLM [21], [25], [26], [27] ตามตารางที่ 1

ตารางที่ 1: องค์ประกอบของสภาพแวดล้อมการทดลอง

องค์ประกอบ	บทบาทในระบบ	เครื่องมือ/เทคโนโลยี
1. Workflow Orchestrator	ควบคุมลำดับการทำงานทั้งหมด	n8n
2. Evidence Database	จัดเก็บหลักฐาน ผลการทดลอง บันทึกการทำงาน	PostgreSQL
3. Network Worker	ประเมินบริการเครือข่าย	Custom worker
4. Web Worker	ประเมินเว็บแอปพลิเคชัน	Custom worker/Web scanner
5. API Worker	ประเมิน API Security	Custom worker
6. Secret/Config Worker	ตรวจจับ secret และ misconfiguration	Custom worker
7. LLM Analysis	วิเคราะห์และจัดลำดับความเสี่ยง	GPT-4o
8. Monitoring	ติดตามสถานะการทดลอง	Grafana

กระบวนการทำงานที่พัฒนามาน n8n ทำหน้าที่เป็นศูนย์กลางในการประสานงานกระบวนการทดสอบเจาะระบบและการวิเคราะห์ช่องโหว่ โดยแบ่งโครงสร้างการทำงานออกเป็น 8 ขั้นตอน ได้แก่ Trigger & target input, Target classification, Adaptive test strategy, Assessment worker invocation, Structured evidence collection, LLM- assisted analysis & risk prioritization, Hallucination-aware verification และ Reporting & consistency evaluation การออกแบบในลักษณะนี้ช่วยให้การทดสอบแต่ละรอบมีลำดับขั้นตอนชัดเจน สามารถตรวจสอบผลลัพธ์ในแต่ละขั้นได้ และสามารถขยายหรือปรับเปลี่ยนโมดูลในอนาคตได้โดยไม่กระทบต่อสถาปัตยกรรมหลักของระบบ

สำหรับ Secret/Config assessment worker งานวิจัยนี้ใช้โมดูลประเมินแบบ custom worker เพื่อค้นหาร่องรอยของข้อมูลลับและการตั้งค่าที่ไม่ปลอดภัยในระบบเป้าหมาย โดยกระบวนการตรวจสอบประกอบด้วยการค้นหารูปแบบข้อความที่มีความเสี่ยง เช่น token, credential, API key, password, private key, environment variable และ configuration ที่เปิดเผยข้อมูลสำคัญหรือมีค่าที่ไม่ปลอดภัย โมดูลดังกล่าวใช้แนวทาง rule-based pattern matching ร่วมกับ keyword matching และ configuration checklist เพื่อระบุหลักฐานเบื้องต้นจากระบบเป้าหมาย OWASP WrongSecrets

ผลลัพธ์ที่ได้จาก Secret/Config assessment worker จะถูกแปลงเป็นหลักฐานเชิงโครงสร้างก่อนจัดเก็บในฐานข้อมูล PostgreSQL โดยแต่ละรายการประกอบด้วย campaign ID, target ID, evidence type, source worker, timestamp, raw observation, matched pattern, severity hint และ reference location เช่น ชื่อไฟล์ ตำแหน่งของค่า configuration หรือข้อความบันทึกการทำงานที่เกี่ยวข้อง การออกแบบดังกล่าวช่วยให้ข้อค้นพบด้านข้อมูลลับรั่วไหลและการตั้งค่าที่ไม่ปลอดภัยสามารถตรวจสอบย้อนกลับไปยังหลักฐานต้นทางได้อย่างชัดเจน

ทั้งนี้ งานวิจัยนี้ไม่ได้มุ่งเปรียบเทียบประสิทธิภาพของเครื่องมือตรวจจับ secret โดยตรง แต่ใช้ Secret/Config assessment worker เป็นส่วนหนึ่งของกรอบการทำงานเพื่อทดสอบความสามารถในการรวบรวม แปลง และตรวจสอบหลักฐานเชิงโครงสร้างในบริบทของ secret leakage และ misconfiguration

3. ระบบเป้าหมายในการทดลอง

การทดลองใช้ระบบเป้าหมายที่มีช่องโหว่โดยเจตนา จำนวน 4 ระบบ เพื่อให้ครอบคลุมบริบทของการประเมินช่องโหว่ที่แตกต่างกัน ได้แก่ ระบบบริการเครือข่าย เว็บแอปพลิเคชันสมัยใหม่ ระบบ API Security และระบบที่เกี่ยวข้องกับการรั่วไหลของข้อมูลลับหรือการตั้งค่าที่ไม่ปลอดภัย การเลือกเป้าหมายทั้ง 4 ประเภทนี้มีวัตถุประสงค์เพื่อประเมินว่ากรอบการทำงานที่พัฒนาขึ้นสามารถจำแนกประเภทเป้าหมาย เลือกลยุทธ์การทดสอบ และรวบรวมหลักฐานเชิงโครงสร้างได้เหมาะสมกับลักษณะของช่องโหว่แต่ละประเภทหรือไม่ โดยรายละเอียดของระบบเป้าหมาย ประเภทเป้าหมาย และจุดเน้นการทดสอบ แสดงดังตารางที่ 2

ตารางที่ 2: ระบบเป้าหมายและประเภทของการประเมินช่องโหว่

ระบบเป้าหมาย	ประเภทเป้าหมาย	จุดเน้นการทดสอบ
1. Metasploitable3	Host/Network Service	พอร์ต บริการ service banner และช่องโหว่ระดับระบบ
2. OWASP Juice Shop	Modern Web Application	HTTP request/response, routes, headers, forms
3. OWASP crAPI	API Security	endpoint, method, status code, authorization behavior
4. OWASP WrongSecrets	Secret/Misconfiguration	secret leakage, token, credential, configuration

4. ตัวแปรในการทดลอง

การวิจัยนี้กำหนดตัวแปรในการทดลองเพื่อใช้ประเมินประสิทธิภาพ ความสม่ำเสมอ และความสามารถในการตรวจสอบย้อนกลับของกรอบการทำงานที่พัฒนาขึ้น โดยแบ่งเป็นตัวแปรต้น ตัวแปรตาม และตัวแปรควบคุม ดังตารางที่ 3

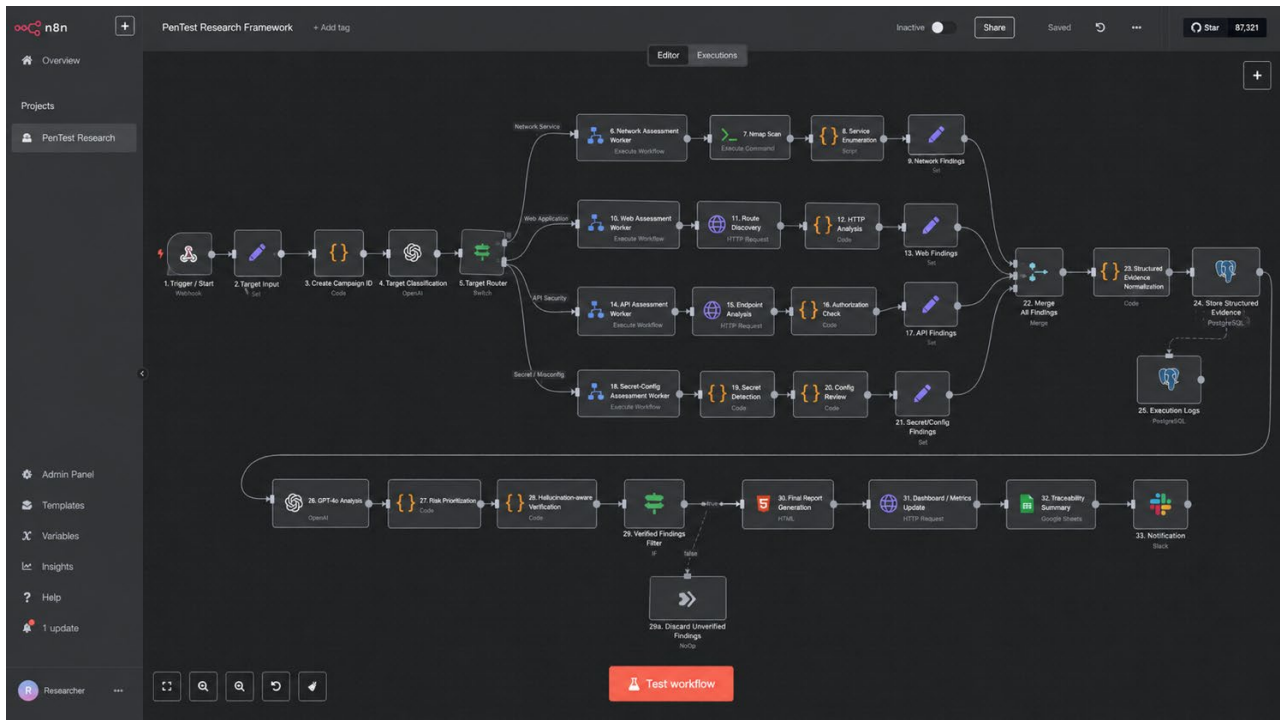
ตารางที่ 3: ตัวแปรในการทดลอง

ประเภทตัวแปร	รายละเอียดโดยสรุป
ตัวแปรต้น	ประเภทของระบบเป้าหมาย ได้แก่ Metasploitable3, OWASP Juice Shop, OWASP crAPI และ OWASP WrongSecrets; รอบการทดสอบซ้ำ 6 รอบต่อเป้าหมาย; และกลยุทธ์การทดสอบแบบ adaptive ตามประเภทของเป้าหมาย
ตัวแปรตาม	จำนวนหลักฐานเชิงโครงสร้าง, จำนวนข้อค้นพบจาก LLM, จำนวนข้อค้นพบที่ผ่านการตรวจสอบ, จำนวนข้อค้นพบที่ถูกปฏิเสธ, ค่าความสามารถในการตรวจสอบย้อนกลับ, ระยะเวลาในการดำเนินการ และค่าความสม่ำเสมอของผลลัพธ์
ตัวแปรควบคุม	workflow version, prompt template, structured output schema, LLM model, verification rules, configuration ของระบบเป้าหมาย และสภาพแวดล้อม Docker container stack

5. การพัฒนากระบวนการทำงานบนแพลตฟอร์ม n8n

กระบวนการทำงานที่พัฒนาขึ้นบนแพลตฟอร์ม n8n ทำหน้าที่เป็นศูนย์กลางในการควบคุมและประสานกระบวนการทดสอบเจาะระบบอัตโนมัติ [25] ตั้งแต่การรับข้อมูลเป้าหมาย การจำแนกประเภทของระบบ การเลือกโมดูลประเมินช่องโหว่ที่เหมาะสม การรวบรวมหลักฐานเชิงโครงสร้าง การวิเคราะห์ด้วยแบบจำลองภาษาขนาดใหญ่ การตรวจสอบความน่าเชื่อถือของข้อค้นพบ ไปจนถึงการบันทึกผลและจัดทำรายงานขั้นสุดท้าย ทั้งนี้ ภาพรวมของกระบวนการทำงานอัตโนมัติ (workflow automation) แสดงดังรูปที่ 2 โดยสามารถสรุปลำดับการทำงานหลักได้ 5 ขั้นตอน ดังนี้

1. รับข้อมูลเป้าหมายและสร้าง campaign ID เพื่อใช้ติดตามผลการทดลองในแต่ละรอบ
2. จำแนกประเภทของเป้าหมายและเลือก assessment worker ให้เหมาะสมกับลักษณะของระบบ
3. รวบรวมผลลัพธ์จาก assessment worker และแปลงให้อยู่ในรูปแบบหลักฐานเชิงโครงสร้าง
4. วิเคราะห์หลักฐานด้วย GPT-4o และตรวจสอบข้อค้นพบด้วยกลไกตรวจสอบภาวะหลอนของโมเดลแบบอิงหลักฐาน
5. บันทึกผลลงฐานข้อมูล สร้างรายงานขั้นสุดท้าย และคำนวณตัวชี้วัดสำหรับการประเมินผลการทดลอง



รูปที่ 2: ตัวอย่าง กระบวนการทำงานอัตโนมัติบนแพลตฟอร์ม n8n สำหรับกรอบการทดสอบเจาะระบบแบบอัตโนมัติ

หลังจาก Target classification ระบบจะใช้กฎแบบ rule-based เพื่อเลือก assessment worker ให้สอดคล้องกับลักษณะของเป้าหมาย หากเป้าหมายมีลักษณะเป็นบริการเครือข่าย ระบบจะเรียกใช้ Network assessment worker หากพบลักษณะของเว็บแอปพลิเคชัน ระบบจะเรียกใช้ Web assessment worker หากพบ endpoint หรือพฤติกรรมของ API ระบบจะเรียกใช้ API assessment worker และหากพบร่องรอยของข้อมูลลับหรือการตั้งค่าที่ไม่ปลอดภัย ระบบจะเรียกใช้ Secret/Config assessment worker ในกรณีที่เป้าหมายมีหลายลักษณะร่วมกัน ระบบสามารถเรียกใช้ worker มากกว่าหนึ่งตัว และบันทึก source worker ไว้ในหลักฐานแต่ละรายการเพื่อใช้ตรวจสอบย้อนกลับ ตามตารางที่ 4

ตารางที่ 4: กลยุทธ์การเลือกโมดูลประเมินแบบปรับตามประเภทเป้าหมาย

ประเภทเป้าหมาย	เงื่อนไขที่ใช้พิจารณา	โมดูลที่ถูกเรียกใช้	หลักฐานที่รวบรวม
Host/ Network Service	พบ IP address, open port, service banner หรือ network service	Network Assessment Worker	port, protocol, service banner, version, vulnerability hint
Modern Web Application	พบ HTTP/HTTPS endpoint, route, form, header หรือ web response	Web Assessment Worker	request/response, route, header, cookie, form, alert
API Security	พบ API endpoint, HTTP method, JSON response หรือ authorization behavior	API Assessment Worker	endpoint, method, status code, response behavior, authorization evidence
Secret/Misconfiguration	พบ token, credential, API key, configuration หรือ secret pattern	Secret/Config Assessment Worker	secret pattern, file/config reference, matched value, configuration evidence
Mixed Target	พบคุณลักษณะมากกว่า 1 ประเภท	เรียกหลาย worker ตามลำดับ	หลักฐานจากหลาย worker พร้อม source worker

โดยรหัสเทียมในอัลกอริทึมที่ 1 แสดงลำดับการทำงานหลักของกรอบการทำงานที่นำเสนอ โดยเริ่มจากการสร้างรหัสประจำรอบการทดสอบ การจำแนกประเภทเป้าหมาย และการเลือกโมดูลประเมินช่องโหว่ให้เหมาะสมกับลักษณะของระบบเป้าหมาย จากนั้นระบบจะแปลงผลลัพธ์ดิบจากโมดูลประเมินให้อยู่ในรูปแบบหลักฐานเชิงโครงสร้าง และส่งต่อไปยังแบบจำลองภาษาขนาดใหญ่เพื่อช่วย

วิเคราะห์ข้อค้นพบ ข้อค้นพบแต่ละรายการจะต้องผ่านการตรวจสอบด้านการรองรับด้วยหลักฐาน ความสามารถในการตรวจสอบย้อนกลับ และการควบคุมขอบเขตของข้อกล่าวอ้างก่อนนำเข้าสู่รายงานขั้นสุดท้ายดังนี้

อัลกอริทึมที่ 1 กระบวนการทดสอบเจาะระบบอัตโนมัติแบบขับเคลื่อนด้วยหลักฐาน

ข้อมูลนำเข้า: ข้อมูลเป้าหมาย (T), ค่ากำหนดของกระบวนการทำงาน (C), และกฎการตรวจสอบ (R)

ผลลัพธ์: ข้อค้นพบที่ผ่านการตรวจสอบ (F_v), ข้อค้นพบที่ไม่ผ่านการตรวจสอบ (F_r), และรายงานขั้นสุดท้าย

1. สร้างรหัสประจำรอบการทดสอบ (CID) สำหรับเป้าหมาย (T)
2. จำแนกประเภทของเป้าหมาย (T_c) ตามคุณลักษณะของระบบเป้าหมาย
3. เลือกโมดูลประเมินช่องโหว่ (W) ตามกลยุทธ์การทดสอบแบบปรับตามประเภทเป้าหมาย
4. เรียกใช้โมดูลประเมินช่องโหว่ (W) และรวบรวมผลลัพธ์ดิบ (D_r)
5. แปลงผลลัพธ์ดิบ (D_r) ให้อยู่ในรูปแบบหลักฐานเชิงโครงสร้าง (E_s)
6. จัดเก็บหลักฐานเชิงโครงสร้าง (E_s) ลงในฐานข้อมูลหลักฐาน
7. ส่งหลักฐานเชิงโครงสร้าง (E_s) ไปยังแบบจำลองภาษาขนาดใหญ่เพื่อวิเคราะห์แบบอิงหลักฐาน
8. สร้างข้อค้นพบเบื้องต้น (F_c) พร้อมรหัสอ้างอิงหลักฐานของแต่ละข้อค้นพบ
9. ตรวจสอบข้อค้นพบแต่ละรายการ ($f \in F_c$) ด้วยกฎด้านการรองรับด้วยหลักฐาน การตรวจสอบย้อนกลับ และการควบคุมขอบเขตข้อกล่าวอ้าง
10. หากข้อค้นพบ (f) ผ่านกฎการตรวจสอบทั้งหมด ให้เพิ่ม (f) ลงใน (F_v)
11. หากข้อค้นพบ (f) ไม่ผ่านกฎการตรวจสอบ ให้เพิ่ม (f) ลงใน (F_r)
12. สร้างรายงานขั้นสุดท้ายโดยใช้เฉพาะข้อค้นพบที่ผ่านการตรวจสอบ (F_v)
13. คำนวณตัวชี้วัดการประเมินผลและจัดเก็บผลลัพธ์ของรอบการทดสอบ

6. การจัดการหลักฐานเชิงโครงสร้างและการวิเคราะห์ด้วย LLM

ผลลัพธ์จาก assessment worker แต่ละประเภทจะถูกแปลงจากข้อมูลดิบให้อยู่ในรูปแบบ หลักฐานเชิงโครงสร้างก่อนเข้าสู่กระบวนการวิเคราะห์ โดยแต่ละรายการหลักฐานประกอบด้วยข้อมูลสำคัญ เช่น campaign ID, target ID, evidence type, source worker, timestamp, raw observation และ reference สำหรับการตรวจสอบย้อนกลับ หลักฐานทั้งหมดจะถูกจัดเก็บในฐานข้อมูล PostgreSQL เพื่อใช้เป็นแหล่งข้อมูลกลางสำหรับการวิเคราะห์ การตรวจสอบความถูกต้องของข้อค้นพบ การสร้างรายงาน และการคำนวณตัวชี้วัดของผลการทดลอง แนวทางนี้ช่วยลดปัญหาความไม่เป็นมาตรฐานของผลลัพธ์จากเครื่องมือหลายประเภท และทำให้สามารถเชื่อมโยงข้อค้นพบกลับไปยังหลักฐานต้นทางได้อย่างเป็นระบบ

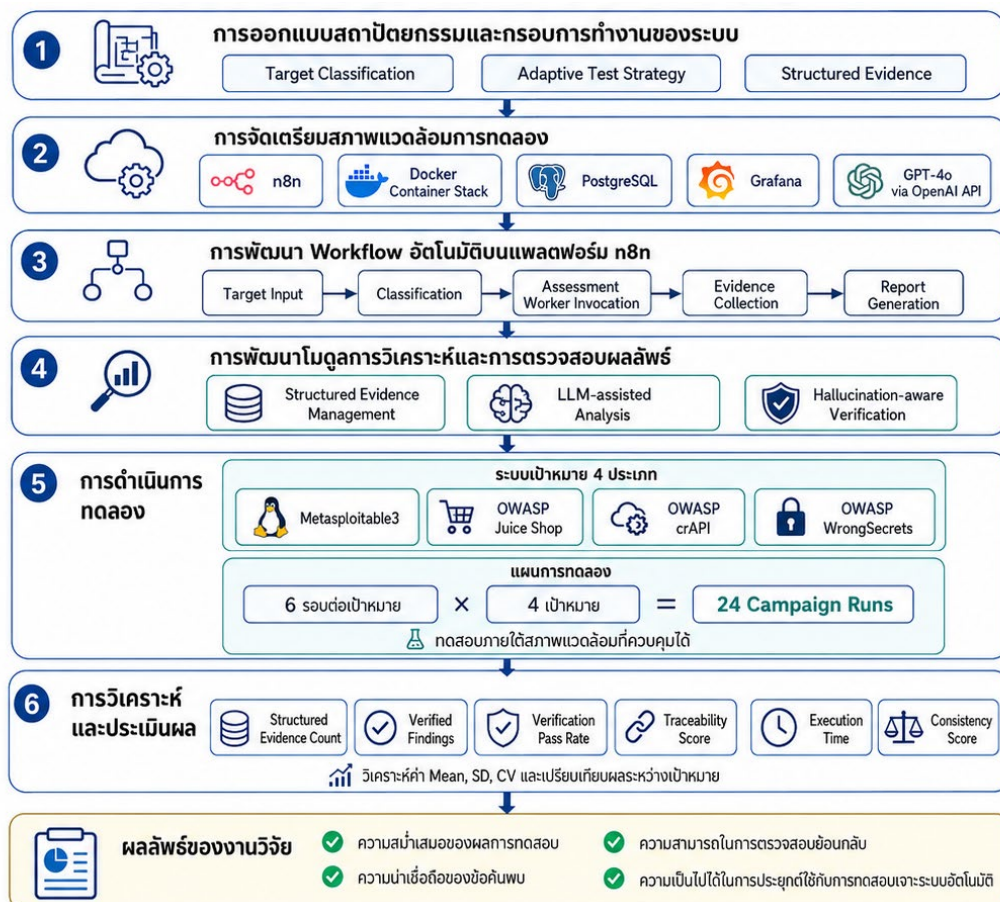
หลังจากจัดเก็บหลักฐานเชิงโครงสร้างแล้ว ระบบจะส่งข้อมูลดังกล่าวไปยัง GPT-4o ผ่าน OpenAI API เพื่อสนับสนุนการวิเคราะห์ช่องโหว่ การเชื่อมโยงหลักฐาน การจัดลำดับความเสี่ยง และการสร้างข้อค้นพบในรูปแบบที่สามารถตรวจสอบต่อได้ [12], [21], งานวิจัยนี้ใช้วิธีการออกแบบคำสังนำเข้า (prompt engineering) และผลลัพธ์แบบมีโครงสร้าง (structured output) โดยไม่มีการปรับแต่งโมเดลเฉพาะงาน (fine-tune model) เพื่อควบคุมให้ LLM วิเคราะห์เฉพาะหลักฐานที่ได้รับ และต้องระบุ evidence ID ที่สนับสนุนข้อค้นพบแต่ละรายการ วิธีดังกล่าวทำให้ LLM ทำหน้าที่เป็นเครื่องมือช่วยตีความและสังเคราะห์หลักฐาน แทนการสร้างข้อสรุปโดยไม่มีข้อมูลรองรับ ซึ่งเป็นเงื่อนไขสำคัญต่อการลดความเสี่ยงจากภาวะหลอนของโมเดล และเพิ่มความน่าเชื่อถือของผลการทดสอบเจาะระบบอัตโนมัติ

7. กลไกตรวจสอบภาวะหลอนของโมเดลแบบอิงหลักฐาน

เพื่อควบคุมความน่าเชื่อถือของผลลัพธ์จาก LLM งานวิจัยนี้พัฒนากลไกตรวจสอบภาวะหลอนของโมเดลแบบอิงหลักฐาน สำหรับตรวจสอบข้อค้นพบก่อนนำเข้าสู่รายงานขั้นสุดท้าย โดยกลไกดังกล่าวพิจารณาข้อค้นพบจาก LLM ใน 4 ด้าน ได้แก่ Evidence support, Traceability, Claim control และ Report readiness เพื่อให้มั่นใจว่าข้อค้นพบทุกประเด็นมี evidence ID รองรับ สามารถตรวจสอบย้อนกลับไปยังหลักฐานต้นทางได้ ไม่กล่าวอ้างเกินกว่าหลักฐานที่ตรวจพบ และมีข้อมูลเพียงพอสำหรับการจัดทำรายงาน ทั้งนี้ ข้อค้นพบที่ผ่านเกณฑ์จะถูกจัดเป็นข้อค้นพบที่ผ่านการตรวจสอบ (verified findings) ส่วนข้อค้นพบที่ไม่ผ่านเกณฑ์จะถูกจัดเป็นข้อค้นพบที่ไม่ผ่านการตรวจสอบ (rejected findings) และไม่นำเข้าสู่รายงานขั้นสุดท้าย [14], [15], [22], [24]

8. ขั้นตอนการทดลอง

การทดลองดำเนินการกับระบบเป้าหมายที่มีช่องโหว่โดยเจตนา 4 ระบบ โดยทดสอบซ้ำระบบละ 6 รอบ รวมทั้งสิ้น 24 รอบการทดลอง ในแต่ละรอบ ระบบจะรับข้อมูลเป้าหมายและสร้าง campaign ID จากนั้นจำแนกประเภทของเป้าหมาย เลือก assessment worker ที่เหมาะสม รวบรวมและแปลงผลลัพธ์เป็นหลักฐานเชิงโครงสร้าง วิเคราะห์หลักฐานด้วย GPT-4o ตรวจสอบข้อค้นพบด้วยกลไกตรวจสอบภาวะหลอนของโมเดลแบบอิงหลักฐาน บันทึกผลการทดลองลงฐานข้อมูล PostgreSQL และคำนวณตัวชี้วัดด้านความสม่ำเสมอ ความน่าเชื่อถือ และความสามารถในการตรวจสอบย้อนกลับ ทั้งนี้ ในการทดลองซ้ำแต่ละรอบมีการควบคุม workflow version, prompt template, structured output schema, LLM model, verification rules และ configuration ของระบบเป้าหมายให้คงที่ เพื่อให้สามารถเปรียบเทียบผลลัพธ์ระหว่างรอบการทดลองได้อย่างเป็นระบบ ดังรูปที่ 3



รูปที่ 3: ขั้นตอนการดำเนินการวิจัยและการทดลอง 24 รอบการทดลอง

9. ตัวชี้วัดที่ใช้ในการประเมินผล

ตัวชี้วัดที่ใช้ในการประเมินกรอบการทำงานประกอบด้วย ผลการทดลองถูกวิเคราะห์ในมิติของจำนวนหลักฐานเชิงโครงสร้าง จำนวนข้อค้นพบจาก LLM จำนวนข้อค้นพบที่ผ่านและไม่ผ่านการตรวจสอบ อัตราการผ่านการตรวจสอบ ความสามารถในการตรวจสอบย้อนกลับ เวลาในการดำเนินการ และความสม่ำเสมอของผลลัพธ์ ดังแสดงในตารางที่ 5 โดยตัวชี้วัดเชิงอัตราและค่าความสม่ำเสมอคำนวณตามสมการ (1)–(4)

$$\text{Verification Pass Rate} = \frac{\text{Verified Findings}}{\text{LLM Findings}} \times 100 \quad (1)$$

$$\text{Rejection Rate} = \frac{\text{Rejected Findings}}{\text{LLM Findings}} \times 100 \quad (2)$$

$$\text{CV}(\%) = \frac{\text{SD}}{\text{Mean}} \times 100 \quad (3)$$

$$\text{Consistency Score} = 100 - \text{CV} \quad (4)$$

ตารางที่ 5: ตัวชี้วัดที่ใช้ในการประเมินกรอบการทำงาน

ตัวชี้วัด	ความหมาย	วิธีคำนวณ
Structured Evidence Count	จำนวนหลักฐานที่รวบรวมได้	นับจำนวน evidence records
LLM Findings	จำนวนข้อค้นพบที่ LLM สร้าง	นับจำนวน findings
Verified Findings	จำนวนข้อค้นพบที่ผ่านการตรวจสอบ	นับ findings ที่ผ่าน verification
Rejected Findings	จำนวนข้อค้นพบที่ไม่ผ่านการตรวจสอบ	นับ findings ที่ถูก rejected
Verification Pass Rate	อัตราข้อค้นพบที่ผ่านการตรวจสอบ	สมการ (1)
Rejection Rate	อัตราข้อค้นพบที่ถูกปฏิเสธ	สมการ (2)
Traceability Score	ความสามารถในการตรวจสอบย้อนกลับ	คะแนนจาก evidence linkage
Execution Time	เวลาดำเนินการต่อ campaign	เวลาสิ้นสุด-เวลาเริ่มต้น
Consistency Score	ความสม่ำเสมอของผลลัพธ์	สมการ (4)

ผลการวิจัยและการอภิปรายผล

การวิจัยนี้ประเมินกรอบการทำงานอัตโนมัติแบบขับเคลื่อนด้วยหลักฐานสำหรับการทดสอบเจาะระบบ โดยทดลองกับระบบเป้าหมายที่มีช่องโหว่โดยเจตนา 4 ประเภท ได้แก่ Metasploitable3, OWASP Juice Shop, OWASP crAPI และ OWASP WrongSecrets โดยทดสอบซ้ำระบบละ 6 รอบ รวมทั้งสิ้น 24 รอบการทดลอง ผลการทดลองถูกวิเคราะห์ในมิติของความสามารถในการตรวจสอบย้อนกลับจำนวนหลักฐานเชิงโครงสร้าง จำนวนข้อค้นพบจาก LLM จำนวนข้อค้นพบที่ผ่านและไม่ผ่านการตรวจสอบ อัตราการผ่านการตรวจสอบ เวลาในการดำเนินการ และความสม่ำเสมอของผลลัพธ์

1. ผลการทดลองสรุปตามระบบเป้าหมาย

ตารางที่ 6 แสดงผลการทดลองสรุปตามระบบเป้าหมาย โดยรวมตัวชี้วัดหลัก ได้แก่ จำนวนหลักฐานเชิงโครงสร้างเฉลี่ย จำนวนข้อค้นพบจาก LLM จำนวนข้อค้นพบที่ผ่านการตรวจสอบ จำนวนข้อค้นพบที่ถูกปฏิเสธ อัตราการผ่านการตรวจสอบ ค่า ความสามารถในการตรวจสอบย้อนกลับเฉลี่ย และเวลาเฉลี่ยในการดำเนินการทดลอง

จากตารางที่ 6 พบว่า OWASP Juice Shop มีจำนวนหลักฐานเชิงโครงสร้างเฉลี่ยสูงสุด เท่ากับ 136.83 รายการ และใช้เวลาเฉลี่ยในการดำเนินการสูงสุด เท่ากับ 632.83 วินาที สะท้อนให้เห็นว่าเว็บแอปพลิเคชันมีองค์ประกอบที่ต้องตรวจสอบหลากหลาย เช่น เส้นทางการใช้งาน HTTP request/response headers cookies และ forms จึงสร้างหลักฐานจำนวนมากและใช้เวลาประเมินสูงกว่าเป้าหมายอื่น

ตารางที่ 6: ผลการทดลองสรุปตามระบบเป้าหมาย

ระบบเป้าหมาย	Average Evidence	LLM Findings	Verified	Rejected	Pass Rate (%)	Traceability (%)	Mean Time (s)
Metasploitable3	89.17	81	69	12	85.19	94.83	442.17
OWASP Juice Shop	136.83	125	107	18	85.60	91.83	632.83
OWASP crAPI	115.33	106	88	18	83.02	89.83	562.33
OWASP WrongSecrets	70.00	62	56	6	90.32	96.83	373.17
รวม/เฉลี่ย	102.83	374	320	54	85.56	93.33	502.63

ในด้านคุณภาพของข้อค้นพบ พบว่า ระบบสร้างข้อค้นพบจาก LLM ทั้งหมด 374 รายการ โดยผ่านการตรวจสอบ 320 รายการ และถูกปฏิเสธ 54 รายการ คิดเป็นอัตราการผ่านการตรวจสอบเฉลี่ยร้อยละ 85.56 และอัตราการปฏิเสธร้อยละ 14.44 ผลลัพธ์นี้สะท้อนว่ากลไกตรวจสอบภาวะหลอนของโมเดลแบบอิงหลักฐาน มีบทบาทสำคัญในการคัดกรองข้อค้นพบที่ไม่มีหลักฐานรองรับก่อนนำเข้าสู่รายงานขั้นสุดท้าย

เมื่อพิจารณารายเป้าหมาย พบว่า OWASP WrongSecrets มีอัตราการผ่านการตรวจสอบสูงสุด เท่ากับร้อยละ 90.32 และมีค่าความสามารถในการตรวจสอบย้อนกลับเฉลี่ยสูงสุด เท่ากับร้อยละ 96.83 เนื่องจากหลักฐานด้าน secret leakage และ misconfiguration มักมีลักษณะชัดเจน สามารถเชื่อมโยงกลับไปยัง token, credential, file หรือ configuration ที่ตรวจพบได้โดยตรง ขณะที่ OWASP crAPI มีอัตราการผ่านการตรวจสอบต่ำสุด เท่ากับร้อยละ 83.02 และมีค่าความสามารถในการตรวจสอบย้อนกลับต่ำสุด เท่ากับร้อยละ 89.83 สะท้อนว่าการวิเคราะห์หลักฐานด้าน API Security มีความซับซ้อนมากกว่า เนื่องจากต้องตีความพฤติกรรมของ endpoint, HTTP method, response behavior และ authorization behavior ร่วมกัน

2. ความสม่ำเสมอของผลลัพธ์

ตารางที่ 7 แสดงค่าความสม่ำเสมอของผลลัพธ์ใน 4 มิติ ได้แก่ ความสม่ำเสมอของจำนวนหลักฐานเชิงโครงสร้าง ความสม่ำเสมอของข้อค้นพบที่ผ่านการตรวจสอบ ความสม่ำเสมอของค่าความสามารถในการตรวจสอบย้อนกลับ และความสม่ำเสมอของเวลาในการดำเนินการ

ตารางที่ 7: ค่าความสม่ำเสมอของผลลัพธ์

ระบบเป้าหมาย	Evidence Consistency (%)	Verified Finding Consistency (%)	Traceability Consistency (%)	Execution Time Consistency (%)
Metasploitable3	98.35	95.24	99.21	98.32
OWASP Juice Shop	97.67	95.78	99.18	97.94
OWASP crAPI	98.13	96.48	99.16	98.39
OWASP WrongSecrets	97.98	94.47	99.22	98.55
เฉลี่ย	98.03	95.49	99.19	98.30

จากตารางที่ 7 พบว่า กรอบการทำงานที่พัฒนาขึ้นสามารถให้ผลลัพธ์ที่มีความสม่ำเสมอสูงในทุกมิติ โดยค่า Evidence consistency เฉลี่ยอยู่ที่ร้อยละ 98.03 แสดงให้เห็นว่าระบบสามารถรวบรวมหลักฐานเชิงโครงสร้างได้อย่างมีเสถียรภาพในแต่ละรอบการทดลอง โดยค่า Verified finding consistency เฉลี่ยอยู่ที่ร้อยละ 95.49 แสดงให้เห็นว่าจำนวนข้อค้นพบที่ผ่านการตรวจสอบมีความคงที่ในระดับดี แม้การวิเคราะห์จะอาศัย LLM ซึ่งอาจมีความไม่แน่นอนในการสร้างผลลัพธ์บางส่วน

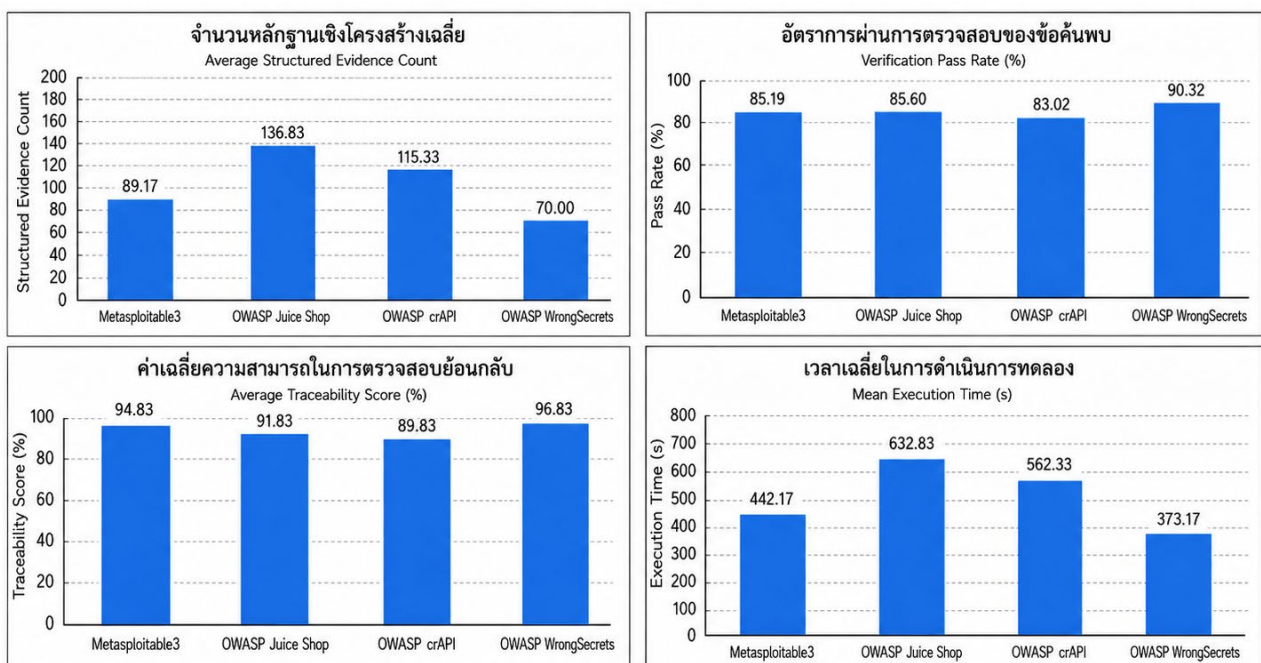
นอกจากนี้ ค่า Traceability consistency เฉลี่ยสูงถึงร้อยละ 99.19 แสดงให้เห็นว่าการออกแบบฐานข้อมูลหลักฐานเชิงโครงสร้าง และการเชื่อมโยง evidence ID สามารถสนับสนุนการตรวจสอบย้อนกลับได้อย่างมีประสิทธิภาพ ส่วนค่า Execution time consistency เฉลี่ยอยู่ที่ร้อยละ 98.30 แสดงว่าการทำงานของกระบวนการทำงานอัตโนมัติภายใต้สภาพแวดล้อม Docker container stack มีเวลาการดำเนินการที่ค่อนข้างคงที่ระหว่างรอบการทดลอง

โดยรวม ผลการทดลองสนับสนุนสมมติฐานของงานวิจัยว่า การผสาน workflow automation, structured evidence, LLM-assisted analysis และ hallucination-aware verification สามารถเพิ่มความน่าเชื่อถือ ความสม่ำเสมอ และความสามารถในการตรวจสอบย้อนกลับของการทดสอบเจาะระบบอัตโนมัติได้อย่างเป็นระบบ

เมื่อเปรียบเทียบกับงานวิจัยที่เกี่ยวข้อง เช่น PentestGPT และงานด้าน LLM agents สำหรับการทดสอบเจาะระบบ ซึ่งมุ่งใช้แบบจำลองภาษาขนาดใหญ่เพื่อช่วยวางแผน วิเคราะห์ และสนับสนุนกระบวนการทดสอบเจาะระบบ [12], [13] งานวิจัยนี้มีจุดแตกต่างสำคัญคือไม่ได้ใช้แบบจำลองภาษาขนาดใหญ่สร้างข้อสรุปโดยอิสระ แต่กำหนดให้ข้อค้นพบทุกประเด็นต้องเชื่อมโยงกับหลักฐานเชิงโครงสร้างและรหัสหลักฐานก่อนนำเข้าสู่รายงานขั้นสุดท้าย นอกจากนี้ เมื่อพิจารณาร่วมกับงานด้านการประเมินความสามารถและความเสี่ยงของแบบจำลองภาษาขนาดใหญ่ในบริบทความมั่นคงปลอดภัยไซเบอร์ เช่น CyberSecEval, SecBench และ CyberMetric [14], [16], [17] งานวิจัยนี้มุ่งเน้นการออกแบบกรอบการทำงานเชิงปฏิบัติที่ผสานการจัดการหลักฐานเชิงโครงสร้าง การวิเคราะห์ด้วยแบบจำลองภาษาขนาดใหญ่ และกลไกตรวจสอบภาวะหลอนของโมเดลแบบอิงหลักฐาน ผลการทดลองแสดงให้เห็นว่ากรอบการทำงานที่นำเสนอสามารถให้คำอธิบายการผ่านการตรวจสอบของข้อค้นพบเฉลี่ยร้อยละ 85.56 คะแนนความสามารถในการตรวจสอบย้อนกลับเฉลี่ยร้อยละ 93.33 และค่าความสม่ำเสมอของหลักฐานเชิงโครงสร้างเฉลี่ยร้อยละ 98.03 ซึ่งสะท้อนถึงความสามารถในการเพิ่มความน่าเชื่อถือ ความสม่ำเสมอ และการตรวจสอบย้อนกลับของการทดสอบเจาะระบบอัตโนมัติ

3. สรุปภาพรวมผลการทดลอง

กรอบการทำงานอัตโนมัติแบบขับเคลื่อนด้วยหลักฐานสามารถให้ผลลัพธ์ที่มีความสม่ำเสมอ น่าเชื่อถือ และตรวจสอบย้อนกลับได้ดี โดยแต่ละระบบเป้าหมายมีลักษณะผลลัพธ์แตกต่างกันตามประเภทของช่องโหว่และความซับซ้อนของระบบ ตามรูปที่ 4



รูปที่ 4: กราฟภาพรวมผลการทดลองทั้ง 4 ตัวชี้วัด

จากรูปที่ 4 พบว่า OWASP Juice Shop มีจำนวนหลักฐานเชิงโครงสร้างเฉลี่ยและเวลาเฉลี่ยในการดำเนินการทดลองสูงสุด สะท้อนความซับซ้อนของเว็บแอปพลิเคชันที่มีองค์ประกอบต้องตรวจสอบหลายประเภท [5] เช่น route, request/response, header, cookie และ form ขณะที่ OWASP WrongSecrets มีอัตราการผ่านการตรวจสอบของข้อค้นพบและความสามารถในการตรวจสอบย้อนกลับสูงสุด เนื่องจากหลักฐานด้านข้อมูลลับรั่วไหล/การตั้งค่าที่ไม่ปลอดภัย (secret/misconfiguration) มีความชัดเจน และสามารถตรวจสอบย้อนกลับไปยังตำแหน่งของไฟล์ token credential หรือ configuration ได้โดยตรง ส่วน OWASP crAPI มีอัตราการผ่านการตรวจสอบและค่าความสามารถในการตรวจสอบย้อนกลับต่ำสุด สะท้อนว่าการตีความหลักฐานด้าน API Security มีความซับซ้อนมากกว่า เนื่องจากต้องพิจารณาพฤติกรรมของ endpoint, HTTP method, response behavior และ authorization behavior [6] ร่วมกัน ผลดังกล่าวแสดงให้เห็นว่ากรอบการทำงานที่พัฒนาขึ้นสามารถรองรับการประเมินช่องโหว่ที่แตกต่างกันตามบริบทของระบบเป้าหมาย และช่วยให้การใช้ LLM อยู่ภายใต้หลักฐานที่ตรวจสอบได้อย่างเป็นระบบ

4. การแสดงผลผ่าน Dashboard

นอกจากการสรุปผลการทดลองในรูปแบบตารางและกราฟแล้ว ระบบยังรองรับการแสดงผลผ่าน Grafana dashboard สำหรับติดตามผลการทดลองและตัวชี้วัดหลักของกรอบการทำงาน [27] เช่น จำนวนรอบการทดลอง จำนวนข้อค้นพบจาก LLM [12], [14], [21], [22] จำนวนข้อค้นพบที่ผ่านและไม่ผ่านการตรวจสอบ อัตราการผ่านการตรวจสอบ ค่าความสามารถในการตรวจสอบย้อนกลับ และเวลาในการดำเนินการทดลอง ดังแสดงในรูปที่ 5 ทั้งนี้ dashboard มีบทบาทเป็นเครื่องมือสนับสนุนการติดตามและตรวจสอบสถานะของระบบในภาพรวม ขณะที่การอภิปรายผลเชิงวิจัยอ้างอิงจากตารางผลลัพธ์และรูปที่ 5 เป็นหลัก เพื่อลดความซ้ำซ้อนของข้อมูลและทำให้บทความมีความกระชับ



รูปที่ 5: การแสดงผลสรุปผ่าน Grafana dashboard สำหรับกรอบการทดสอบเจาะระบบแบบอัตโนมัติ

สรุปผล

งานวิจัยนี้ได้นำเสนอกรอบการทำงานอัตโนมัติแบบขับเคลื่อนด้วยหลักฐานสำหรับการทดสอบเจาะระบบ โดยใช้แพลตฟอร์ม n8n เป็นศูนย์กลางในการประสานกระบวนการทำงานภายใต้สภาพแวดล้อม Docker container stack ระบบที่พัฒนาขึ้นสามารถจำแนกประเภทของเป้าหมาย เลือกกลยุทธ์การทดสอบที่เหมาะสม รวบรวมผลลัพธ์จาก assessment worker ให้อยู่ในรูปแบบหลักฐานเชิงโครงสร้าง วิเคราะห์ด้วยแบบจำลองภาษาขนาดใหญ่ และตรวจสอบข้อค้นพบก่อนนำเข้าสู่รายงานขั้นสุดท้ายด้วยกลไกตรวจสอบภาวะหลอนของโมเดลแบบอิงหลักฐาน

ผลการทดลองกับระบบเป้าหมายที่มีช่องโหว่โดยเจตนา 4 ประเภท จำนวน 24 รอบการทดลอง แสดงให้เห็นว่า กรอบการทำงานสามารถรวบรวมหลักฐานเชิงโครงสร้างได้อย่างสม่ำเสมอ มีอัตราการผ่านการตรวจสอบของข้อค้นพบจาก LLM อยู่ในระดับสูง สอดคล้องกับแนวทางการใช้แบบจำลองภาษาขนาดใหญ่ในบริบทความมั่นคงปลอดภัยไซเบอร์ [12], [14], [16], [20], [22] และมีค่าความสามารถในการตรวจสอบย้อนกลับที่สะท้อนความเชื่อมโยงของผลลัพธ์กับหลักฐานได้อย่างชัดเจน นอกจากนี้ ค่าความสม่ำเสมอของตัวชี้วัดหลักยังอยู่ในระดับสูง แสดงให้เห็นว่าระบบสามารถให้ผลลัพธ์ที่ค่อนข้างคงที่ภายใต้สภาพแวดล้อมการทดลองที่ควบคุมได้

ประเด็นสำคัญของงานวิจัยนี้คือการเสนอแนวทางการใช้ LLM ในการทดสอบเจาะระบบภายใต้หลักฐานที่ตรวจสอบได้ แทนการให้ LLM สรุปผลโดยอิสระ โดยการผสานหลักฐานเชิงโครงสร้าง รหัสหลักฐาน ความสามารถในการตรวจสอบย้อนกลับ และกลไกตรวจสอบภาวะหลอนของโมเดลแบบอิงหลักฐาน ทำให้ข้อค้นพบที่เข้าสู่รายงานขั้นสุดท้ายมีความน่าเชื่อถือมากขึ้น ลดความเสี่ยงจากการกล่าวอ้างเกินหลักฐาน และสนับสนุนการนำผลการประเมินไปใช้ในบริบทด้านความมั่นคงปลอดภัยไซเบอร์ได้อย่างเป็นระบบ

อย่างไรก็ตาม งานวิจัยนี้ดำเนินการภายใต้สภาพแวดล้อมห้องปฏิบัติการและใช้ระบบเป้าหมายที่มีช่องโหว่ที่กำหนดไว้ จึงยังไม่ครอบคลุมความซับซ้อนของระบบ production จริง งานวิจัยในอนาคตควรขยายการทดลองไปยังสภาพแวดล้อมที่ใกล้เคียงการใช้งานจริงมากขึ้น เปรียบเทียบผลลัพธ์ระหว่าง LLM หลายรุ่น เพิ่มกลไก human-in-the-loop หลังขั้นตอนการตรวจสอบภาวะหลอนของโมเดลแบบอิงหลักฐานและก่อนขั้นตอนการสร้างรายงานขั้นสุดท้าย โดยระบบจะส่งต่อเฉพาะข้อค้นพบที่มีความเสี่ยงสูง ข้อค้นพบที่มีหลักฐานไม่สมบูรณ์ หรือข้อค้นพบที่ระบบจัดให้อยู่ในกลุ่มต้องตรวจสอบเพิ่มเติมไปยังผู้เชี่ยวชาญด้านความมั่นคงปลอดภัยไซเบอร์ เพื่อพิจารณาอนุมัติ ปฏิเสธ หรือปรับแก้ข้อค้นพบดังกล่าวก่อนนำเข้าสู่รายงานขั้นสุดท้าย และระบบจะบันทึกผลการตัดสินใจลงฐานข้อมูลเพื่อสนับสนุนการตรวจสอบย้อนกลับ รวมถึงพัฒนากระบวนการตรวจสอบเชิงหลักฐานให้รองรับมาตรฐานหรือกรอบการประเมินด้านความมั่นคงปลอดภัยไซเบอร์ที่หลากหลายยิ่งขึ้น

REFERENCES

- [1] OWASP API Security Project Team. "OWASP Top 10 API Security Risks – 2023." OWASP.org. Accessed: May 22, 2026. [Online]. Available: <https://owasp.org/API-Security/editions/2023/en/0x11-t10/>
- [2] OWASP Foundation. *OWASP Web Security Testing Guide v4.2*. (2020). Accessed: May 22, 2026. [Online]. Available: <https://owasp.org/www-project-web-security-testing-guide/>
- [3] OWASP Foundation, "OWASP Top Ten Web Application Security Risks," 2025. [Online]. Available: <https://owasp.org/www-project-top-ten/>
- [4] Forum of Incident Response and Security Teams, "Common Vulnerability Scoring System Version 4.0: Specification Document," Document Version 1.2, Nov. 1, 2023. [Online]. Available: <https://www.first.org/cvss/v4-0/cvss-v40-specification.pdf>
- [5] *Metasploitable3*. (2025). GitHub repository. Rapid7, Inc. Accessed: May 22, 2026. [Online]. Available: <https://github.com/rapid7/metasploitable3>
- [6] G. Lyon, *Nmap Network Scanning*. Beaverton, OR, USA: Nmap Software, 2009. Accessed: May 22, 2026. [Online]. Available: <https://nmap.org/book/toc.html>
- [7] *Metasploit Framework*. GitHub repository. Rapid7, Inc. Accessed: May 22, 2026. [Online]. Available: <https://github.com/rapid7/metasploit-framework>



- [8] B. Kimminich et al. *OWASP Juice Shop*. (2026). GitHub repository. (v20.1.1). OWASP. Accessed: Jul. 18, 2026. [Online]. Available: <https://github.com/juice-shop/juice-shop>
- [9] ZAP Dev Team. "Documentation." ZAPROXY.org. Accessed: May 22, 2026. [Online]. Available: <https://www.zaproxy.org/docs/>
- [10] OWASP Foundation. "OWASP crAPI." OWASP.org. Accessed: May 22, 2026. [Online]. Available: <https://owasp.org/www-project-crap/>
- [11] J. Willemssen et al. *OWASP WrongSecrets*. (2026). OWASP. Accessed: May 22, 2026. [Online]. Available: <https://owasp.org/www-project-wrongsecrets/>
- [12] G. Deng et al., "PentestGPT: Evaluating and harnessing large language models for automated penetration testing," in *Proc. 33rd USENIX Secur. Symp.*, Philadelphia, PA, USA, Aug. 2024, pp. 847–864.
- [13] R. Fang, R. Bindu, A. Gupta, Q. Zhan, and D. Kang, "LLM agents can autonomously hack websites," 2024, *arXiv:2402.06664*.
- [14] M. Bhatt et al., "CyberSecEval 2: A wide-ranging cybersecurity evaluation suite for large language models," 2024, *arXiv:2404.13161*.
- [15] S. Wan et al., "CYBERSECEVAL 3: Advancing the evaluation of cybersecurity risks and capabilities in large language models," 2024, *arXiv:2408.01605*.
- [16] P. Jing et al., "SecBench: A comprehensive multi-dimensional benchmarking dataset for LLMs in cybersecurity," 2024, *arXiv:2412.20787*.
- [17] N. Tihanyi, M. A. Ferrag, R. Jain, T. Bisztray, and M. Debbah, "CyberMetric: A benchmark dataset based on retrieval-augmented generation for evaluating LLMs in cybersecurity knowledge," in *Proc. IEEE Int. Conf. Cyber Secur. Resilience (CSR)*, London, U.K., Sep. 2024, pp. 296–302.
- [18] I. David and A. Gervais, "Multi-agent penetration testing AI for the web," 2025, *arXiv:2508.20816*.
- [19] X. Wu et al., "CurriculumPT: LLM-based multi-agent autonomous penetration testing with curriculum-guided task scheduling," *Appl. Sci.*, vol. 15, no. 16, 2025, Art. no. 9096, doi: 10.3390/app15169096.
- [20] N. O. Jaffal, M. Alkhanafseh, and D. Mohaisen, "Large language models in cybersecurity: A survey of applications, vulnerabilities, and defense techniques," *AI*, vol. 6, no. 9, 2025, Art. no. 216, doi: 10.3390/ai6090216.
- [21] OpenAI. "GPT-4o system card." OPENAI.com. Accessed: May 22, 2026. [Online]. Available: <https://openai.com/index/gpt-4o-system-card/>
- [22] *Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile*, NIST AI 600-1, National Institute of Standards and Technology, Gaithersburg, MD, USA, Jul. 2024. [Online]. Available: <https://doi.org/10.6028/NIST.AI.600-1>
- [23] National Institute of Standards and Technology, "The NIST Cybersecurity Framework (CSF) 2.0," Gaithersburg, MD, USA, NIST CSWP 29, Feb. 2024. [Online]. Available: <https://doi.org/10.6028/NIST.CSWP.29>
- [24] OWASP, "OWASP Top 10 for LLM Applications 2025," OWASP Found., Wilmington, DE, USA, Accessed: May 22, 2026. [Online]. Available: <https://genai.owasp.org/resource/owasp-top-10-for-llm-applications-2025/>
- [25] n8n. "AI Workflow Automation Platform." N8N.io. Accessed: May 22, 2026. [Online]. Available: <https://n8n.io/>
- [26] Docker Inc. Docker Compose. (2026). Accessed: May 22, 2026. [Online]. Available: <https://docs.docker.com/compose/>
- [27] Grafana Labs. "Dashboards." GRAFANA.com. Accessed: May 22, 2026. [Online]. Available: <https://grafana.com/docs/grafana/latest/visualizations/dashboards/>