The rise and rise of LLM-powered PenTesting systems: The State of the Art

Haitham S. Al-Sinani[1,3], Chris J. Mitchell[2], and Abdulaziz S. Al-Hosni[1]

[1] Diwan of Royal Court, Muscat, Oman. [hsssinani,ashosni]@diwan.gov.om
[2] Royal Holloway, University of London, Egham, UK. C.Mitchell@rhul.ac.uk
[3] German University of Technology in Oman, Muscat, Oman.
Haitham.AlSinani@gutech.edu.om

Abstract. This survey examines the rapid emergence of AI-enhanced PenTesting (penetration testing) systems driven by Large Language Models (LLMs). It reviews research published between late 2022 and early 2026, analysing a broad range of proposed systems, tools, and benchmarking frameworks. Rather than documenting each work separately, the survey synthesises recurring architectures, operational scopes, capabilities, and evaluation approaches reported across the literature, highlighting representative design choices and limitations. By comparing these systems, the survey identifies common trends, research directions, and gaps, and discusses benchmark practices, ethical considerations, and open challenges. Given the complexity and rapid proliferation of LLM-powered PenTesting systems, which can be challenging for students and novices to navigate, this paper also aims to support learning for aspiring practitioners and newcomers. The goal is to provide researchers and practitioners with a concise overview of the state of the art and guidance for future development, and to encourage further research into understanding, evaluating, and improving LLM-driven PenTesting systems.

Keywords: AI · GenAI · LLM · PenTesting · Ethical Hacking

1 Introduction

Ethical hacking, or Penetration testing (PenTesting) [129], is a fundamental practice in cybersecurity, enabling organisations to identify and mitigate vulnerabilities before adversaries can exploit them. Despite its importance, PenTesting remains largely manual, requiring highly specialised expertise to design attack paths, interpret system responses, and iteratively refine tactics under time pressure. Consequently, PenTesting is labour-intensive, costly, and difficult to scale. These challenges are exacerbated by a persistent shortage of skilled professionals, with recent estimates placing the global cybersecurity workforce gap at nearly 4.8 million unfilled positions between 2023 and 2024 [114].

Large Language Models (LLMs) offer a promising avenue for addressing these limitations. Recent advances have enabled LLMs to generate commands, reason

about vulnerabilities, and interact with tools, transforming them from passive assistants into autonomous or semi-autonomous agents embedded within PenTesting workflows. A growing body of research has therefore explored LLM-driven systems capable of automating phases such as reconnaissance, exploitation, privilege escalation (PrivEsc), and post-exploitation. However, these systems combine powerful reasoning capabilities with potentially harmful actions, raising critical questions regarding reliability, safety, evaluation, and real-world applicability.

This survey synthesises the evolving landscape of LLM-powered PenTesting systems published between late 2022 and early 2026. Relevant studies were identified through a structured review of peer-reviewed literature, arXiv preprints, and relevant technical reports using scholarly databases, keyword-based searches related to LLM-driven PenTesting and AI-assisted ethical hacking, backward reference tracing from relevant publications, and inspection of related author publications across scholarly and research-discovery platforms, including Google Scholar, ResearchGate, DBLP, ORCID, and arXiv. Publications were included if they described concrete architectures, workflows, benchmarks, or operational capabilities related to LLM-assisted PenTesting or autonomous ethical hacking tasks, while AI-driven systems relying solely on conventional machine learning or non-LLM techniques were excluded, resulting in a final corpus of 28 systems.

The main contributions of this survey are as follows. First, we identify and analyse 28 representative LLM-aided PenTesting systems across seven categories: early LLM-as-advisor systems, initial exploitation systems, post-exploitation systems, web-focused systems, benchmarking frameworks, emerging-domain systems, and near end-to-end systems. Second, we propose a taxonomy categorising these systems according to operational scope and capability. Third, we provide consolidated cross-system comparisons covering attack-phase emphasis, capability coverage, autonomy, architecture, governance, reasoning mechanisms, evaluation practices, and deployment characteristics. Finally, we discuss current limitations, benchmarking challenges, ethical concerns, and future research directions for LLM-driven ethical hacking systems.

The remainder of this paper is organised as follows. Section 2 reviews the ethical hacking landscape, and Section 3 provides background on LLMs. Section 4 examines emerging tactics for LLM-driven PenTesting systems, while Section 5 surveys representative systems and Section 6 compares them. Section 7 discusses limitations and challenges, Section 8 reviews related work, and Section 9 concludes the paper.

2 The Ethical Hacking Landscape

Overview. Ethical hacking is the authorised application of hacking techniques to evaluate and strengthen organisational security by identifying vulnerabilities before adversaries exploit them [18, 22, 25, 42, 46, 47, 117, 120, 123, 126, 129, 141]. The discipline has evolved from 1960s 'tiger teams' and early vulnerability scanners (e.g., SATAN) to comprehensive toolkits such as Kali Linux, yet modern engagements remain largely manual and expertise-intensive despite widespread

use of tools including Nmap, Wireshark, and Metasploit [1, 67]. Demand continues to rise due to escalating cyber threats and a global cybersecurity skills gap (approximately 4.7–4.8 million unfilled roles by 2024), making assessments costly and difficult to scale [58, 114]. Ethical hacking commonly follows five phases: (1) *Reconnaissance*—collection of target information (public data, network structure, hosts, configurations); (2) *Scanning and Enumeration*—identification of open ports, services, and weaknesses to map entry points; (3) *Gaining Access*—exploitation of vulnerabilities to obtain an initial foothold; (4) *Maintaining and Elevating Access*—persistence, PrivEsc, pivoting, and lateral movement to expand control; (5) *Covering Tracks & Reporting*—artefact removal, environment restoration, and documentation with remediation guidance.

PenTesting Standards. PenTesting methodologies provide structured procedures for simulating adversaries and assessing security posture. Major frameworks include NIST SP 800-115 [129], PTES [106], OSSTMM [62], ISSAF [92], and PTF [107]. NIST emphasises risk-managed phases (planning, discovery, attack, reporting); PTES defines a practitioner-oriented lifecycle with threat modelling; OSSTMM stresses measurable, repeatable assessments across technical, physical, and human domains; ISSAF integrates technical testing with managerial risk analysis; and PTF provides procedural guidance and tool recommendations. Complementary adversary models guide planning and analysis. The Cyber Kill Chain (CKC) decomposes attacks into sequential stages [105], though its linear structure may oversimplify real campaigns [66, 104]. Mandiant's Attack Life Cycle highlights sustained post-compromise activity such as persistence and lateral movement [81, 82]. MITRE ATT&CK provides a comprehensive knowledge base of adversary tactics and techniques for threat modelling and red-team operations [14, 128]. Finally, risk and compliance standards further shape practice. OWASP Top 10 [97] lists critical web vulnerabilities as a baseline checklist, while PCI DSS [101] mandates periodic testing of cardholder environments with defined scope and reporting requirements.

Traditional PenTesting Tools. Pre AI-assisted PenTesting relied on specialised tools across reconnaissance, vulnerability discovery, exploitation, and post-exploitation, many of which remain the backbone of LLM-driven systems. `Kali Linux` serves as a platform bundling core utilities: `nmap`, `OpenVAS`, and `Nessus` for discovery and scanning; `Wireshark` for traffic analysis; interception proxies such as `Burp Suite` and `OWASP ZAP` for web testing; and password tools including `Hydra`, `JtR`, and `Hashcat`. Integrated frameworks (e.g., `Metasploit`) combine scanning and exploitation into unified workflows.

Automated scanners identify weaknesses prior to exploitation across hosts and applications. Network platforms such as `OpenVAS`, `Nessus`, `Qualys VM`, and `Rapid7 InsightVM` provide broad asset coverage, while web scanners (`Burp Suite`, `OWASP ZAP`, `Acunetix`, `Invicti`, `Nikto`) and fuzzers (`Wfuzz`, `Gobuster`) target application-layer flaws. Post-compromise enumeration tools (e.g., `LinPEAS`, `LinEnum`, `pspy` on Linux; `WinPEAS`, `PowerUp`, `SharpUp` on Windows) detect misconfigurations enabling PrivEsc, supported by reference collections such as `GTFOBins` and `LOLBAS`.

Specialised domains extend these capabilities to wireless and IoT environments. Wi-Fi assessments commonly use the `Aircrack-ng` suite and `Kismet`, with tools such as `Reaver`, `Wifite`, and `Fern` enabling automated attacks and persistence checks. Bluetooth utilities (e.g., `BlueMaho`, `BlueHydra`) support device discovery. IoT testing targets embedded protocols including `ZigBee`, `Z-Wave`, `LoRaWAN`, and `BLE` using tools such as `KillerBee`, protocol sniffers, and firmware extraction interfaces.

PenTesting Frameworks & Resources. Modern PenTesting uses established frameworks and controlled environments for safe, realistic assessment, including the `Metasploit Framework`, commercial suites (e.g., `Core Impact`, `Immunity Canvas`), and `Kali Linux` with integrated tools such as `nmap` and `Wireshark`. Deliberately vulnerable applications (OWASP Juice Shop, DVWA, Mutillidae II, WebGoat, bWAPP) enable repeatable experimentation, while complex scenarios are provided by cyber ranges and emulators such as the KYPO Cyber Range Platform, GNS3, EVE-NG, Cisco Packet Tracer, and CORE. Additional resources—including vulnerability databases (NVD, Exploit-DB), repositories (GitHub), knowledge bases (OWASP, MITRE ATT&CK), exploitation lists (`GTFOBins`, `LOLBAS`), and online labs (Hack The Box, TryHackMe, VulnHub)—support both manual and AI-assisted PenTesting.

3 Large Language Models

Overview. LLMs [157] are Transformer-based neural networks trained on massive corpora for language generation, reasoning, and summarisation [133]. Capability scales with model size, from early GPT systems to multimodal frontier models such as OpenAI's GPT family [95], Gemini [48], Claude [21], LLaMA [132], DeepSeek [35], and assistants like Microsoft Copilot. These foundation models are typically adapted via prompting or fine-tuning rather than retraining [28], and open-weight models can run locally using tools such as `llama.cpp` [50, 131]. Modern LLMs can invoke external tools and operate iteratively, enabling agent-like behaviour [144], although emergent abilities remain debated and may degrade in multi-turn interactions [115, 136, 139]. The ecosystem spans proprietary models emphasising multimodality and deployment—e.g., GPT-4/4o, Claude 3–4, Gemini 2.5 Pro, Command R+, and Grok [21, 31, 48, 95, 143]—and open families prioritising efficiency and transparency, including LLaMA, Mistral/Mixtral, Yi, DeepSeek, Falcon, and derivatives [16,35,69,85,91,130,132,134]. While proprietary systems dominate large-scale deployment, open models increasingly offer competitive performance at lower cost.

In cybersecurity, earlier ML research focused mainly on detection tasks due to challenges such as concept drift and adversarial behaviour [124]. The emergence of LLM agents has broadened attention to both defensive applications and offensive risks, including malware generation, spear-phishing, and automated attacks [61,72,98,111], alongside debates over whether AI favours cyber 'swords or shields' [54, 80, 103]. A parallel research stream examines attacks against LLMs themselves, including prompt injection, adversarial inputs, unsafe fine-tuning,

Table 1: Major CTF platforms, training labs, and enterprise testbeds.

#	Platform	Type	Focus
1	Hack The Box	Lab	Realistic machines/networks
2	TryHackMe	Lab	Guided hands-on training
3	picoCTF	CTF	Educational challenges
4	VulnHub	Lab	Downloadable vulnerable VMs
5	OverTheWire	Wargame	Progressive skill building
6	Root-Me	CTF/Lab	Web and system challenges
7	PentesterLab	Training	Web penetration testing
8	Pwnable.kr / .tw	Wargame	Binary exploitation
9	Google CTF	Competition	Advanced global CTF
10	CSAW CTF	Competition	Academic CTF competition
11	CTFtime	Index	CTF rankings/events
12	GOAD (Game of AD)	Enterprise	Active Directory lab

and jailbreaking techniques [52, 108, 148, 153, 162]. Empirical studies also report comparable ethical hacking capabilities across major models, with differences in accuracy, completeness, and clarity [109].

LLM Agents. LLM agents [84, 144] extend single-turn interactions by embedding a model in a reasoning–acting loop [150], enabling autonomous multi-step problem solving via perception, planning, tool use, and feedback. They can generate subgoals (often using chain-of-thought reasoning [140]), invoke external tools such as APIs or code execution environments [99], and adapt behaviour iteratively, as seen in systems like AutoGPT and multi-agent frameworks [121, 137]. Toolkits such as LangChain provide orchestration, memory, and integration capabilities across domains including software engineering, science, and cybersecurity [4, 26, 27, 30, 45, 70, 147]. Benchmarks such as MINT and MT-Bench evaluate multi-turn reasoning and tool use [136, 158].

Local LLMs. Advances in open-weight models enable high-performance LLMs to run locally without proprietary APIs, improving accessibility, privacy, and control. Families such as LLaMA [131] and variants (LLaMA-Adapter [156], StableLM [125], Dolly 2 [32], Koala [49]) demonstrate that instruction-tuned models can perform strongly on commodity hardware using open data and lightweight fine-tuning, reducing cloud reliance and keeping data on-premises.

Benchmarks and Testbeds. Evaluating PenTesting techniques, particularly automated or AI-driven methods, requires environments that balance realism, safety, and reproducibility. Researchers therefore use platforms ranging from structured Capture-the-Flag (CTF) challenges and training labs for controlled benchmarking to enterprise-scale testbeds that emulate complex infrastructures and support analysis of post-breach activities such as lateral movement, PrivEsc, and persistence. Table 1 summarises representative CTF platforms, practice environments, and enterprise testbeds used in cybersecurity research and training.

4 Emerging Tactics for LLM-driven PenTesting Systems

Prompt Engineering. The base prompt [10] establishes the operating context and rules of engagement for the LLM, defining the task as a real-time PenTesting scenario while enforcing constraints such as non-interactive commands, structured outputs (e.g., JSON), and explicit reasoning. More broadly, prompt engineering—the design of input instructions to guide model behaviour—is critical for reliability and safety, as small variations in wording or structure can significantly affect performance [127]. Techniques such as few-shot prompting, chain-of-thought reasoning, and self-consistency improve task accuracy without retraining [38], while system messages in modern models (e.g., GPT-4/GPT-4o) provide high-level behavioural control analogous to lightweight alignment. Prompt design therefore directly influences exploit generation success and defensive reliability, making prompt conditioning a practical alternative to costly model fine-tuning for rapid experimentation and deployment [58].

Chain-of-Thought Reasoning (CoT). CoT [135,140] prompting enhances LLM reasoning by encouraging step-by-step analysis before producing an action, improving performance on complex, multi-turn tasks such as PrivEsc in PenTesting. In systems like `PenTest2.0` [10], CoT is implemented via explicit instructions to assess system state, evaluate prior commands and outputs, and determine the next logical step, promoting context awareness and reducing semantic drift or repetition. CoT variants include: zero-shot CoT (instruction only); few-shot CoT (using curated exemplars from prior successful escalations to induce structured reasoning without model fine-tuning); and fine-tuned CoT models, which are explicitly trained on datasets annotated with intermediate reasoning [138]. These approaches exploit pre-trained capabilities to produce more transparent and context-aware decisions, supporting safer and more effective progression in human-guided PrivEsc workflows [75].

Human-in-the-Loop (HITL). Although LLM-based agents can automate substantial portions of the PenTesting workflow, fully autonomous operation poses significant safety, reliability, legal, and governance risks. As previously discussed, empirical studies show that such systems may ignore constraints, exceed authorised scope, interact with unintended targets, modify system configurations (e.g., installing or downgrading packages), or attempt actions such as social engineering, particularly when operating with network access or elevated privileges. In enterprise environments, these behaviours can disrupt operations, expose sensitive data, or violate legal and professional standards if performed without explicit consent. HITL architectures therefore provide a pragmatic compromise, allowing LLMs to assist with planning, enumeration, and exploit generation while reserving execution approval, scope enforcement, and high-impact decisions for human operators. This approach preserves efficiency gains from automation while maintaining accountability, compliance, and operational safety, making HITL a defensible design choice for real-world deployments.

Human Hint Injection. In many PenTesting scenarios, the operator possesses domain expertise or prior system knowledge. The Human Hint enhancement allows structured injection of such hints into the prompt, guiding the LLM's

decision-making process. Inspired by best practices in HITL design [20, 79], this strategy combines automation with expertise. For instance, the prompt modification may include: "Human Hint: Use the 'id' command instead of the '/bin/sh' for root automated verification." This balances autonomy with control, helping develop effective exploits.

Retrieval-Augmented Generation (RAG). RAG enhances LLM performance by grounding outputs in externally retrieved knowledge rather than relying solely on pretraining [77]. By dynamically incorporating relevant documents or facts into the prompt, RAG reduces hallucinations, improves factual accuracy, and enables models to handle specialised or up-to-date information beyond their training cut-off. This approach is particularly valuable for complex technical tasks that require precise commands and domain-specific guidance. In systems like `PenTest2.0` [10], RAG is used to supply contextually relevant exploitation knowledge during execution, allowing the system to reason with authoritative external sources when internal knowledge is insufficient and thereby improving decision quality in PrivEsc scenarios.

PenTest Task Tree (PTT) Tracking. PTT tracking is a lightweight external memory mechanism that represents a PenTesting process as a structured hierarchy of goals, subtasks, statuses, and outcomes, enabling persistent state management across multi-turn interactions [36]. By explicitly recording completed actions, pending objectives, and failed or avoided strategies, PTTs mitigate context loss, reduce redundant exploration, and improve reasoning coherence during complex engagements that exceed an LLM's context window. This structured task decomposition enables systematic planning, progress tracking, and error recovery beyond unstructured conversational history.

Large Reasoning Models (LRMs). LRMs are LLMs trained to internalise multi-step reasoning rather than relying on prompt-based CoT, enabling deeper deliberation on complex tasks [57, 93]. Examples include OpenAI's o1, Qwen3, and DeepSeek R1 [15, 53, 93]. Because reasoning is embedded during training, explicit CoT prompting or extensive few-shot examples may offer little benefit or even degrade performance [78, 94]. Empirical studies suggest advantages are task-dependent: LRMs often excel on moderately complex problems but may over-deliberate on simple tasks and still struggle on highly complex ones [119]. In PenTesting, they are particularly useful for planning and decision-making under uncertainty, while practical systems pair them with faster models for execution to balance accuracy, cost, and latency [59, 94].

5 LLM-aided PenTesting Systems

Introduction. This section surveys 28 representative LLM-aided PenTesting systems, including peer-reviewed papers (see Table 2) and recent preprints (see Table 3), capturing the current state of AI-driven offensive security research. As

Table 2: Formally published PenTesting systems (most recent to oldest).

#	System	Year	Venue / Publisher
1	WiFiPenTester [8, 9]	2026	Cyber Science '26 — Springer
2	PenTest2.0 [10]	2026	AINA '26 — Springer (LNDECT)
3	AutoSecAgent [7]	2026	Journal of Supercomput — Springer
4	PenForge [64]	2026	ICSE-NIER '26 — ACM
5	HackingBuddyGPT [59]	2026	EMSE — Springer
6	AutoPenBench [51]	2025	EMNLP '25 — ACL
7	AutoPT [142]	2025	TIFS — IEEE
8	Cybench [154]	2025	ICLR '25 — ICLR / OpenReview
9	LLM Pentest Benchmark [68]	2025	UMAP Adj. '25 — ACM
10	Cochise [57]	2025	TOSEM — ACM
11	PenTest++ [4, 5]	2025	CyBAI '25 (CiSt) — IEEE
12	PenHeal [65]	2024	AutonomousCyber '24 — ACM
13	AI-Augmented Ethical Hacking [12]	2024	STM '24 — Springer (LNCS)
14	PentestGPT [36]	2024	USENIX Sec. '24 — USENIX
15	PTGroup [37]	2024	ICIC '24 — Springer (LNCS)
16	GenAI for PenTesting [63]	2024	IJIS — Springer
17	Wintermute [55]	2023	ESEC/FSE '23 — ACM

shown in Table 4, we categorise the surveyed systems into seven groups according to their primary operational scope and capability[4].

(1) Early LLM-as-Advisor Systems. Early LLM-aided PenTesting research treated LLMs as advisory assistants rather than autonomous agents. In these systems, the LLM provides guidance, explanations, and suggested actions, while human testers perform decision-making and execution. This represents the initial stage of AI-driven offensive security, demonstrating feasibility without autonomy. *GenAI for PenTesting* [63] presents a conceptual framework outlining how LLMs can support vulnerability analysis, attack planning, and reporting, establishing a foundation for later systems. *AI-Augmented Ethical Hacking* [12] demonstrates practical GenAI assistance across multiple hacking phases under human supervision, including reconnaissance and exploitation guidance, highlighting productivity gains alongside the need for expert oversight. *PentestGPT* [36] provides interactive guidance across the PenTesting workflow, suggesting strategies and interprets results while execution remains human-driven.

(2) Initial Exploitation Systems. This group comprises LLM-driven systems targeting the initial compromise phase, aiming to obtain a foothold in ex-

[4] We observe that many of these systems have been proposed and developed largely in parallel; indeed, even arXiv submissions do not constitute a reliable indicator of priority, as some authors choose not to release preprints. Moreover, certain arXiv postings appear intended primarily to establish precedence, even though the corresponding work may still be ongoing or not yet realised.

Table 3: Preprint LLM-aided PenTesting systems (most recent to oldest).

#	System	Year	Publisher
18	ICSSPulse [163]	2026	CoRR / arXiv
19	CyberExplorer [110]	2026	CoRR / arXiv
20	HPTSA [161]	2025	CoRR / arXiv
21	VulnBot [76]	2025	CoRR / arXiv
22	RapidPen [90]	2025	CoRR / arXiv
23	Incalmo [122]	2025	CoRR / arXiv
24	HackSynth [88]	2024	CoRR / arXiv
25	AutoAttacker [146]	2024	CoRR / arXiv
26	LLM Agents Exploit 1-day Vulns [44]	2024	CoRR / arXiv
27	LLM Agents Hack Websites [45]	2024	CoRR / arXiv
28	BreachSeek [19]	2024	CoRR / arXiv

ternally reachable targets through vulnerability exploitation, credential harvesting, or automated attack orchestration. *PenTest++* [4, 5] performs end-to-end reconnaissance and exploitation, guiding the transition from service discovery to system access. *RapidPen* [90] emphasises 'IP-to-shell' automation, converting reachable hosts into interactive access with minimal human intervention. *AutoAttacker* [146] orchestrates LLM-guided attack sequences against exposed services, focusing on vulnerability exploitation to establish entry. *VulnBot* [76] automates vulnerability discovery and exploitation workflows to obtain initial access, demonstrating the feasibility of LLM-driven intrusion pipelines. *BreachSeek* [19] employs a multi-agent architecture for automated vulnerability exploitation and foothold establishment in controlled environments. *PTGroup* [37] presents a multi-agent prompt-chain framework for automated scanning and exploitation against controlled target hosts.

(3) Post-Exploitation Systems. Such schemes operate after a foothold is obtained, focusing on PrivEsc, credential abuse, lateral movement, and internal dominance within compromised environments. *Wintermute* [55] represents an early autonomous PrivEsc prototype that assumes an existing low-privilege account and attempts to elevate privileges via a closed LLM–SSH loop. The agent generates shell commands, observes execution outputs, and iteratively refines its strategy to discover misconfigurations and spawn a root shell. *PenTest2.0* [10] is a post-exploitation system that automates multi-turn Linux PrivEsc through iterative reasoning and command execution under operator governance. Starting from a low-privilege shell, the system repeatedly proposes actions, executes them, and adapts based on feedback, aiming to achieve root access while maintaining transparency and control. *HackingBuddyGPT* [59] provides a dedicated benchmark and prototype for autonomous Linux PrivEsc. Operating from a compromised user context, the system evaluates whether LLMs can independently identify and exploit local vulnerabilities to obtain admin privileges in

Table 4: Taxonomy of LLM-aided PenTesting Systems by Scope and Capability

Subsection Category	Description
1) Early LLM-as-Advisor Systems	Prompt-based advisors for planning, vuln explanation, and command suggestions without autonomy (e.g., [12]).
2) Initial Exploitation Systems	Agents exploiting externally reachable vulnerabilities to obtain an initial foothold (e.g., PenTest++).
3) Post-Exploitation Systems	PrivEsc agents using shell commands and environment reasoning on compromised hosts (e.g., Wintermute).
4) Web-Focused Systems	Black-box HTTP agents discovering and exploiting web vulnerabilities automatically (e.g., PenForge, AutoPT).
5) Benchmarking Frameworks	Platforms evaluating LLM PenTesting capability and risk without executing real attacks (e.g., Cybench)
6) Emerging Domains	Systems targeting wireless, IoT, cyber-physical, cloud, or hybrid environments (e.g., WiFiPenTester, ICSSPulse).
7) Near End-to-End Systems	Near-autonomous, multi-phase PenTesting systems coordinating several attack stages (e.g., PenHeal, Incalmo).

controlled environments. *Cochise* [57] extends the post-exploitation paradigm to enterprise-scale networks under an 'assumed-breach' model. Rather than targeting perimeter intrusion, it focuses on internal exploration, credential abuse, lateral movement, and domain dominance within Active Directory environments using a planner–executor architecture.

(4) Web-Focused Systems. This category targets web applications through black-box HTTP interaction, automated vulnerability discovery, and exploitation of OWASP-style flaws. *PenForge* [64] performs autonomous web PenTesting by dynamically instantiating specialised agents based on reconnaissance results. Operating against real-world CVEs, it interacts with applications via HTTP requests and browser automation to identify and exploit vulnerabilities. *AutoPT* [142] provides a fully automated framework for web vulnerability assessment, combining LLM reasoning with tool orchestration to discover and exploit weaknesses in target applications without manual intervention. *PenHeal* [65] focuses on automated vulnerability discovery and remediation workflows, including web application flaws, demonstrating end-to-end analysis from detection to exploitation guidance. *LLM Agents Hack Websites* [45] demonstrates autonomous browser-based exploitation of web targets, identifying issues such as injection and access-control flaws. *LLM Agents Exploit 1-day Vulns* [44] evaluates exploitation of recently disclosed web vulnerabilities, showing feasibility against unpatched flaws. *HPTSA* [161] extends this via multi-agent collaboration to discover and exploit zero-day vulnerabilities.

(5) Benchmarking Frameworks. This group includes platforms designed to assess the capabilities, limitations, and risks of LLM-driven PenTesting sys-

tems rather than performing attacks directly. Such frameworks provide standardised tasks, datasets, and environments for reproducible evaluation and cross-system comparison. *Cybench* [154] offers a comprehensive benchmarking suite covering offensive and defensive cybersecurity tasks, enabling systematic assessment of LLM performance across vulnerability analysis, exploitation reasoning, and incident-response scenarios. *LLM Pentest Benchmark* [68] focuses specifically on PenTesting workflows, providing structured challenges to measure how effectively LLM-based systems can plan and execute attack steps under controlled conditions. *AutoPenBench* [51] introduces an evaluation framework for automated PenTesting agents, emphasising reproducibility and objective comparison across different architectures and model configurations. *CyberExplorer* [110] provides a simulated attack environment for benchmarking LLM offensive capabilities in realistic scenarios, assessing performance, robustness, and potential security risks.

(6) Emerging and Specialised Systems. These systems address environments beyond conventional enterprise IT and web applications, including wireless networks and other domain-specific infrastructures. *WiFiPenTester* [8,9] focuses on automated wireless-network PenTesting, orchestrating attacks against Wi-Fi authentication protocols and configurations. Operating in a physical-proximity threat model, the system integrates reconnaissance, handshake capture, and password-recovery workflows using domain-specific tools, demonstrating the applicability of LLM-driven agents to RF-based environments. *ICSSPulse* [163] extends LLM-assisted PenTesting to industrial control systems (ICS) and operational technology (OT) environments, providing a modular platform that combines network scanning with protocol-aware analysis of industrial services such as Modbus and OPC UA for asset discovery and controlled interaction. It also employs LLM-based components to interpret findings and generate structured technical reports with mitigation guidance, demonstrating the applicability of LLM agents to cyber-physical infrastructures. Unlike network- or host-centric systems, specialised agents must handle heterogeneous data, safety constraints, and domain-specific paths; together, *WiFiPenTester* and *ICSSPulse* signal a shift toward RF and cyber-physical targets (ICS/OT, IoT, CPS).

(7) Attempts at End-to-End (Near-Autonomous) Systems. A few systems attempt to coordinate multiple PenTesting phases within a single framework, but none achieves full autonomy in realistic settings. *PenHeal* [65] employs a planner–executor loop to discover and exploit vulnerabilities from a target IP with minimal input. However, evaluation is limited to controlled lab environments, and remediation actions are not executed automatically, leaving true end-to-end autonomy unrealised. *Incalmo* [122] presents a multi-host red teaming framework using specialised planning and execution agents, although it remains dependent on curated environments and predefined task abstractions. *AutoSecAgent* [7] presents a semi-automated end-to-end PenTesting framework combining Agent Zero orchestration, RAG, and recursive memory, although remediation and real-world validation remain limited.

6 Cross-comparison of LLM-aided PenTesting Systems

Attack-Phase Emphasis. Table 5 categorises LLM-aided PenTesting systems by the primary phase of the offensive lifecycle they target, including initial exploitation, post-exploitation, mixed end-to-end workflows, and benchmarking platforms. The table shows that many systems concentrate on early attack stages—particularly vulnerability exploitation—while a smaller subset addresses post-exploitation activities such as PrivEsc or enterprise assumed-breach scenarios, and relatively few support comprehensive multi-phase operations. Overall, current research emphasises achieving initial access rather than sustaining long-horizon campaigns across the full PenTesting lifecycle.

Capability Coverage. Table 6 compares core capabilities of LLM-aided PenTesting systems across major attack phases, showing that most approaches emphasise early stages such as scanning and exploitation, while relatively few provide comprehensive end-to-end support including post-exploitation and reporting. Entries for Cybench, the LLM Pentest Benchmark, and CyberExplorer are marked **N/A** because these are benchmarking frameworks rather than operational PenTesting systems: they define tasks and environments for evaluating agents but do not themselves execute the attack lifecycle, making phases such as exploitation or persistence inapplicable rather than absent. AI-Augmented Ethical Hacking and GenAI for PenTesting are also labelled **N/A** as they present conceptual models or PoC studies rather than standalone automated tools.

Autonomy, Architecture and Governance. Table 7 compares LLM-aided PenTesting systems by autonomy level, architectural design, execution model, and governance controls. It reveals a predominance of semi-autonomous and HITL designs based on planner–executor or multi-agent architectures and tool-mediated execution, with varying degrees of safety oversight, while fully autonomous systems with strong governance remain comparatively rare.

Reasoning and Guidance. Table 8 compares the use of state-of-the-art reasoning and guidance techniques across LLM-aided PenTesting systems, including CoT, ReAct, planner–executor designs, human hints, RAG, tool invocation, and task-tree tracking. It shows that most systems rely primarily on prompt-level reasoning—especially CoT—often combined with ReAct-style interaction and programmatic tool use, while more advanced guidance mechanisms such as RAG, human hints, and explicit task-tree planning appear only in a minority of systems. This pattern favours short-horizon reasoning, with long-horizon planning and external memory still limited.

Platform, Target, Openness, and Evaluation. Table 9 provides a consolidated comparison of representative LLM-driven PenTesting systems (published and preprint) across execution platform, target surface, prototype availability, and evaluation methodology. The results show a clear shift from early Linux-centric prototypes toward cross-platform designs addressing diverse domains, including host, web, enterprise, wireless, and benchmarking scenarios, indicating an increasing emphasis on generality and real-world applicability. Approximately half of the surveyed systems provide fully open-source implementations, while many others remain closed or only partially disclosed. Evaluations are still pre-

dominantly conducted in controlled laboratory environments, benchmarks, or simulated settings rather than uncontrolled operational deployments. Overall, current research prioritises safety, experimental control, and breadth of coverage over real-world validation and reproducibility, limiting transparency despite rapid technical progress.

7 Discussion and Challenges

Reliability and Stability. Modern LLMs exhibit variability across prompts, and increased scale or instruction tuning does not guarantee consistent reliability: larger, more instructable models may produce plausible yet incorrect outputs, with error rates sensitive to prompt phrasing [159]. In PenTesting contexts, this results in inconsistent command generation, hallucinated tools (e.g., non-existent exploits), and unstable multi-step reasoning requiring human correction [55]. Later frameworks attempt to improve stability through structured prompt orchestration, persistent PTTs, and context summarisation; however, even advanced systems such as Cochise and VulnBot occasionally pursue irrelevant objectives or prematurely declare success [58]. These findings highlight persistent robustness limitations that constrain fully autonomous deployment.

Context and Memory Constraints. LLMs operate within a fixed context window that limits how much prior information can influence predictions; once exceeded, earlier inputs become effectively invisible. In PenTesting sessions involving numerous commands and outputs, this leads to loss of critical state (e.g., discovered credentials or misconfigurations), repeated actions, and missed escalation paths. Empirical studies show that providing longer histories or structured guidance can substantially improve success rates, highlighting the importance of memory management [59]. To address this limitation, systems such as PentestGPT [36] maintain external representations (e.g., PTTs) that summarise completed steps and pending goals, reducing loops and improving situational awareness. More broadly, RAG stores interaction data externally and retrieves relevant context on demand, yet context window size remains a fundamental bottleneck for long-running engagements [73, 77].

Domain Knowledge and Reasoning Gaps. Despite broad pre-training, LLMs lack specialised offensive-security expertise due to knowledge cut-offs, limited awareness of new vulnerabilities, and incomplete understanding of proprietary tools. Domain-specific guidance yields mixed gains, and successful autonomous exploitation typically requires frontier models with tool use, document access, and extended context; removing these features sharply reduces performance, while open-source models often fail [45, 59]. Consequently, current LLMs struggle with complex infrastructures without structured augmentation.

Hallucinations and Command Brittleness. Hallucination—the generation of plausible but incorrect content—is a well-documented failure mode of LLMs [95]. In offensive security contexts, hallucinations manifest as fabricated command names, invalid syntax, or incorrect vulnerability details. Empirical studies in general NLP show that shaped-up models are more confident yet

less prudent; they avoid refusing to answer even when uncertain [159]. In PenTesting this translates into an increased likelihood of the model issuing dangerous commands without validation. Techniques such as CoT prompting and self-refinement can reduce hallucinations, but no method fully eliminates them, necessitating human oversight. Indeed, research suggests that using external knowledge and automated feedback can reduce hallucinations [95, 102].

Computational Cost and Accessibility. Frontier LLMs are costly to train and operate: GPT-4 training reportedly exceeded US$100 million, and autonomous hacking agents can cost about $8.8 per attempt, excluding failures [28, 45]. Although cheaper than human experts, such costs remain high for prolonged campaigns, while reliance on proprietary models limits access, as open-source alternatives often underperform (e.g., GPT-3.5 success rate 6.7% with open models failing) [45]. This disparity constrains research and reinforces commercial concentration.

Malicious Use. LLMs are already exploited in darknet markets, where operators sell customised offensive models such as WormGPT and FraudGPT for social engineering, malware, and payload generation, along with variants like DarkBert, XXXGPT, and WolfGPT [41, 71, 83, 86]. Although their capabilities are unverifiable behind paywalls, their emergence signals accelerating GenAI weaponisation and the need for proactive mitigation as such tools evolve toward autonomous offensive systems.

Safety, Security, and Ethical Concerns. LLM-driven PenTesting systems pose risks at both model and operational levels. They remain vulnerable to prompt injection, jailbreaks, data poisoning, and leakage of sensitive information, undermining reliability and trust in high-stakes contexts [33, 95, 100]. Their dual-use nature means defensive capabilities can also enable real-world attacks, particularly as local models reduce central safeguards and allow low-cost offensive fine-tuning [24, 29]. Autonomous agents may exceed scope, hallucinate actions, or bypass guardrails, motivating controls such as HITL oversight, strict scoping, sandboxing, logging, and governance frameworks [6, 36, 52, 72, 129]. Privacy and regulatory compliance are also critical, as cloud inference can expose operational data and conflict with laws such as GDPR, encouraging on-premises deployment in regulated domains [43]. Overall, these challenges highlight the need for controlled disclosure and robust safeguards to manage the co-evolution of AI-enabled offence and defence.

Summary and Outlook. The limitations outlined above reveal why fully autonomous, trustworthy LLM-driven PenTesting remains an open research challenge. While foundation models excel at pattern matching and rapid prototyping, their inconsistent reasoning, limited context windows, hallucination propensity, security vulnerabilities and high operational costs necessitate careful mitigation. Current research therefore focuses on enhancing reliability through structured memory (PTTs and summarisation), improving domain reasoning via RAG and fine-tuning, developing red-team methods to harden models against prompt injection, and exploring efficient deployment strategies. Until these challenges

are addressed, LLMs should augment rather than replace human penetration testers [2–4, 6, 12, 13].

8 Related Work

This section situates the present study within the existing body of research on LLMs and cybersecurity.

Systematic mappings and surveys of cybersecurity research outline major themes, trends, and gaps across the field while documenting the growing role of LLMs in both defensive and offensive contexts, including threat detection, analysis, automated response, and broader security applications [23, 60, 151]. Subsequent reviews focus specifically on LLM–cybersecurity intersections, proposing taxonomies of applications, system designs, and integration approaches, and examining challenges related to reliability, governance, reproducibility, and evaluation practices [17, 40, 87, 145, 152, 155]. Complementary work examines benchmarking methodologies for LLM-driven offensive security and surveys security and privacy risks to LLMs—including jailbreaks, data poisoning, and information leakage—alongside emerging mitigation strategies [56, 160].

Collectively, prior surveys demonstrate the rapid growth of research at the intersection of LLMs and cybersecurity, documenting applications, risks, and emerging challenges across both defensive and offensive domains. However, the diversity of system designs, tasks, and evaluation approaches highlights the need for a focused synthesis that examines concrete implementations and operational capabilities of LLM-driven security tools. The present study builds on this foundation by providing a structured analysis of contemporary LLM-aided PenTesting systems, emphasising their architectural characteristics, reasoning mechanisms, and practical performance. In doing so, it offers a consolidated view of the current state of the art and supports clearer comparison and understanding of this evolving research area.

9 Conclusion and Future Work

In this paper, we surveyed the rapid emergence of LLM-powered PenTesting systems from late 2022 to early 2026, spanning advisor-style copilots, initial exploitation systems, post-exploitation agents, web-focused frameworks, benchmarking frameworks, emerging-domain tools, and near end-to-end systems. We analysed representative systems in terms of architecture, autonomy, target surface, attack-phase coverage, and evaluation practice, and synthesised recurring design patterns into a practical reference model (planner/generator, executor, memory and summarisation, retrieval, safety/governance, and reporting). Across the literature, most systems concentrate on early-stage exploitation or narrowly scoped tasks, while robust long-horizon operation (context persistence, reliable adaptation, etc.) remains uncommon; consequently, human oversight and constrained execution continue to be the dominant deployment posture.

Future research should prioritise: (i) *more realistic and reproducible evaluation*, including benchmarks that capture enterprise complexity (multi-host state, credential chains, lateral movement, and noisy telemetry) and report success with cost, time, and safety metrics; (ii) *robust memory and state management* (task trees, structured logs, retrieval, and verifiable summaries) to reduce loops and brittle behaviour in long engagements; (iii) *strong governance-by-design*, including scope enforcement, sandboxing, policy-aware tool gateways, and audit-grade logging to mitigate dual-use risk; and (iv) *efficient, accessible deployments*, such as hybrid designs that reserve high-capability reasoning models for planning while using cheaper/faster models for execution, and improved open-weight baselines to reduce reliance on proprietary APIs. Ultimately, the evidence suggests that near-term progress will come less from 'fully autonomous hacking' and more from well-governed, measurable, and reproducible AI-augmented workflows that reliably amplify expert testers while remaining safe and accountable. Finally, an extended version of this paper is publicly available [11].

References

1. Abu-Dabaseh, F. et al.: Automated penetration testing: An overview. Proc. Int. Conf. Nat. Lang. Comput. (2018)
2. Al-Sinani, H.S. and Mitchell, C.J.: AI-augmented ethical hacking: Manual exploitation and privilege escalation in Linux. CoRR abs/2411.17539 (2024)
3. Al-Sinani, H.S. and Mitchell, C.J.: AI-generated papers and manipulation of academic metrics: A case study. Proc. CyBAI@CiSt (2025)
4. Al-Sinani, H.S. and Mitchell, C.J.: Introducing PenTest++: An AI-augmented automated ethical hacking system. Proc. CyBAI@CiSt (2025)
5. Al-Sinani, H.S. and Mitchell, C.J.: PenTest++: Elevating ethical hacking with AI and automation. CoRR abs/2502.09484 (2025)
6. Al-Sinani, H.S. and Mitchell, C.J.: PenTest2.0: Towards autonomous privilege escalation using GenAI. CoRR abs/2507.06742 (2025)
7. Al-Sabbagh, A. et al.: AutoSecAgent: A semi-auto. AI-driven PenTesting framework through recursive memory & real-time RAG. J. Supercomput. 82 (2026).
8. Al-Sinani, H.S., Mitchell, C.J. et al.: WiFiPenTester: Towards governed GenAI-assisted wireless PenTesting. Proc. Cyber Science '26 — Springer Proc. Complex.
9. Al-Sinani, H.S. and Mitchell, C.J.: WiFiPenTester: Wireless ethical hacking with GenAI. CoRR abs/2601.23092 (2026)
10. Al-Sinani, H.S. and Mitchell, C.J. et al.: PenTest2.0: GenAI-driven privilege escalation. Proc. AINA (2026)
11. Al-Sinani, H.S. and Mitchell, C.J.: The Rise and Rise of LLM-powered PenTesting Systems: Capabilities, Limitations, and the Road to Responsible Automation. ResearchGate (2026). https://doi.org/10.13140/RG.2.2.29192.38408
12. Al-Sinani, H.S. et al.: Unleashing AI in ethical hacking. Proc. STM (2024)
13. Al-Sinani, H.S. et al.: Advancing ethical hacking with AI: A Linux-based study. Proc. ITASEC/SERICS (2025)
14. Alford, R. et al.: Caldera: A red-blue cyber operations automation platform (2022)
15. Alibaba Group: Qwen3: A family of large reasoning models (2024)
16. Almazrouei, E. et al.: The Falcon series of open language models. CoRR abs/2311.16867 (2023)

17. Alqahtani, H. et al.: A comprehensive review of generative AI techniques and their impact on cybersecurity. Soft Comput. 29 (2025)
18. AlShebli, H. et al.: A study on PenTesting process and tools. Proc. IEEE LISAT'18
19. Alshehri, I., Alshehri, A., Almalki, A., Bamardouf, M., Akbar, A.: BreachSeek: A multi-agent automated penetration tester. CoRR abs/2409.03789 (2024).
20. Amershi, S. et al.: Power to the people: The role of humans in interactive machine learning. AI Mag. 35 (2014)
21. Anthropic: Introducing the next generation of Claude (2024)
22. Arkin, B. et al.: Software penetration testing. IEEE Secur. Priv. 3 (2005)
23. Aydos, M. et al.: Security testing of web applications: A systematic mapping. J. King Saud Univ. Comput. Inf. Sci. 34 (2022)
24. Beeching, E. et al.:StackLLaMA: An RL fine-tuned LLaMA model for Stack Exchange QA (2023)
25. Bishop, M.: About penetration testing. IEEE Secur. Priv. 5 (2007)
26. Boiko, D.A. et al.: Emergent autonomous scientific research capabilities of LLMs. CoRR abs/2304.05332 (2023)
27. Bran, A.M. et al.: Aug. LLMs with chemistry tools. Nat. Mach. Intell. 6 (2024)
28. Brown, T.B. et al.: Language models are few-shot learners. NeurIPS 33 (2020)
29. Brundage, M. et al.: The malicious use of artificial intelligence: Forecasting, prevention, and mitigation. CoRR abs/1802.07228 (2024)
30. Chase, H.: LangChain (2022). `https://www.langchain.com/`
31. Cohere: Command R+ (2024). `docs.cohere.com/docs/command-r-plus`
32. Conover, M. et al.: Dolly 2: Open instruction-tuned LLM (2023).
33. Das, B.C. et al.: Security and privacy challenges of large language models: A survey. ACM Comput. Surv. 57 (2025)
34. De Pasquale, G. et al.: ChainReactor: Automated privilege escalation via AI planning. USENIX Security (2024)
35. DeepSeek-AI et al.: DeepSeek-V3 technical report. CoRR abs/2412.19437 (2025)
36. Deng, G. et al.: PentestGPT: Automated penetration testing with LLMs. USENIX Security (2024)
37. Wu, L. et al.: PTGroup: An automated penetration testing framework using LLMs and multiple prompt chains. Proc. ICIC 2024. LNCS 14870. Springer (2024)
38. Denny, P. et al.: Conversing with Copilot: Prompt engineering for CS1. ACM SIGCSE (2023)
39. Dong, Q. et al.: A survey on in-context learning. CoRR abs/2301.00234 (2024)
40. Dube, R.: LLMs in information security research: A survey (2024)
41. Dutta, T.S.: Black-hat AI tools XXXGPT and WolfGPT '23.
42. Engebretson, P.: The Basics of Hacking and Penetration Testing. Syngress (2013)
43. European Parliament and Council: Regulation (EU) 2016/679 (GDPR) (2016).
44. Fang, R. et al.: LLM agents exploit one-day vulns. CoRR abs/2404.08144 (2024)
45. Fang, R. et al.: LLM agents hack websites. CoRR abs/2402.06664 (2024)
46. Fatima, A. et al.: Penetration testing and VA challenges. Proc. ICBATS (2023)
47. Geer, D. et al.: Penetration testing: A duet. Proc. ACSAC (2002)
48. Gemini Team: Gemini: Multimodal model family. CoRR abs/2312.11805 (2025)
49. Geng, X. et al.: Koala dialogue model (2023).
50. Gerganov, G.: llama.cpp: Local LLaMA inference. GitHub (2023).
51. Gioacchini, L. et al.: AutoPenBench: Benchmarking agents for PenTesting. CoRR abs/2410.03225 (2024)
52. Greshake, K. et al.: Indirect prompt injection attacks. abs/2302.12173 (2023)
53. Guo, D. et al.: DeepSeek-R reasoning via RL. Nature 645 (2025)

54. Handa, A. et al.: Machine learning in cybersecurity: A review. WIREs Data Min. Knowl. Discov. 9 (2019)
55. Happe, A. et al.: Penetration testing with LLMs. Proc. ESEC/FSE (2023)
56. Happe, A., et al.: Benchmarking LLM-driven offensive security. 2504.10112 '25
57. Happe, A., Cito, J.: Can LLMs hack enterprise networks? ACM Trans. Softw. Eng. Methodol. (2025)
58. Happe, A., et al.: Efficacy of LLMs for PenTesting. CoRR abs/2507.00829 (2025)
59. Happe, A. et al.: LLMs as hackers: Autonomous Linux privilege escalation. Empir. Softw. Eng. 31 (2026)
60. Hassanin, M. et al.: LLMs for cyber defence: Opportunities and directions. CoRR abs/2405.14487 (2024)
61. Hazell, J.: Spear phishing with LLMs. CoRR abs/2305.06972 (2023)
62. Herzog, P. et al.: OSSTMM 3: Security testing methodology. ISECOM (2020).
63. Hilario, E. et al.: Generative AI for PenTesting. Int. J. Inf. Secur. 23 (2024)
64. Huang, H. et al.: PenForge: On-the-fly agent construction for PenTesting. Proc. ICSE-NIER (2026)
65. Huang, J., Zhu, Q.: PenHeal: Two-stage LLM PenTesting framework. Proc. AutonomousCyber (2024)
66. Hutchins, E.M. et al.: Intelligence-driven defence and kill chain model. Leading Issues Inf. Warfare Secur. Res. 1 (2011).
67. Infosec Institute: History of penetration testing (2019). https://www.infosecinstitute.com/resources/penetration-testing/the-history-of-penetration-testing/
68. Isozaki, I. et al.: Towards automated PenTesting: LLM benchmark. Proc. UMAP Adjunct (2025)
69. Jiang, A.Q. et al.: Mixtral of experts. CoRR abs/2401.04088 (2024)
70. Jimenez, C. et al.: SWE-bench: Can LLMs solve GitHub issues? Proc. ICLR '24
71. Jin, Y. et al.: DarkBERT: Language model for the dark web. CoRR abs/2305.08596 (2023)
72. Kang, D. et al.: Exploit. programmatic behaviour of LLMs. Proc. IEEE SPW '24
73. Kobayashi, M. et al.: LLM agents for semi-autonomous PenTesting. CoRR abs/2502.15506 (2025)
74. Kojima, T. et al.: LLMs are zero-shot reasoners. Proc. NeurIPS (2022)
75. Kong, A. et al.: Better zero-shot reasoning via role-play prompting. CoRR abs/2308.07702 (2024)
76. Kong, H. et al.: VulnBot: Autonomous multi-agent PenTesting. CoRR abs/2501.13411 (2025)
77. Lewis, P. et al.: RAG for knowledge-intensive NLP. Proc. NeurIPS (2020)
78. Li, X. et al.: When thinking fails: Pitfalls of reasoning in LLMs. 2505.11423 (2025)
79. Li, Z. et al.: LLMs struggle to describe the haystack. CoRR abs/2502.14748 (2025)
80. Lohn, A.J. et al.: Will AI make cyber swords or shields? abs/2207.13825 (2022)
81. Mandiant: APT and the attack lifecycle (2013).
82. Mandiant: Targeted attack life cycle (2023). https://cloud.google.com/security/resources/insights/targeted-attack-lifecycle
83. Mascellino, A.: WormGPT enables BEC attacks (2023).
84. Mialon, G. et al.: Augmented language models: A survey. abs/2302.07842 (2023)
85. Mistral AI: Mistral-7B-Instruct v0.2 (2023). https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2
86. Montalbano, E.: DarkBERT malware model (2023).
87. Motlagh, F.N. et al.: LLMs in cybersecurity: State of the art. CoRR abs/2402.00891 (2024)

88. Muzsai, L. et al.: HackSynth: Autonomous PenTesting agent framework. CoRR abs/2412.01778 (2024)
89. Nakajima, Y.: BabyAGI (2023). github.com/yoheinakajima/babyagi
90. Nakatani, S.: RapidPen: IP-to-shell automated PenTesting. abs/2502.16730 '25
91. NousResearch: Nous-Hermes-2-Yi-34B (2023). https://huggingface.co/NousResearch/Nous-Hermes-2-Yi-34B
92. OISSG: Information Systems Security Assessment Framework (2006). https://untrustednetwork.net/files/issaf0.2.1.pdf
93. OpenAI: o1 reasoning models (2024). https://openai.com/index/introducing-openai-o1-preview/
94. OpenAI: Reasoning best practices (2024). https://platform.openai.com/docs/guides/reasoning-best-practices
95. OpenAI et al.: GPT-4 technical report. CoRR abs/2303.08774 (2024)
96. Ouyang, L. et al.: Training LMs with human feedback. Proc. NeurIPS (2022)
97. OWASP Foundation: OWASP Top Ten (2021).
98. Pa Pa, Y.M. et al.: ChatGPT for malware development. Proc. CSET (2023)
99. Parisi, A. et al.: Tool-augmented language models. CoRR abs/2205.12255 (2022)
100. Pathade, C.: Prompt injection and jailbreak evaluation of LLMs. CoRR abs/2505.04806 (2025)
101. PCI SSC: Penetration Testing Guidance (2017). https://www.pcisecuritystandards.org
102. Peng, B. et al.: Improving LLMs with external knowledge and feedback. CoRR abs/2302.12813 (2023)
103. Phuong, M. et al.: Evaluating frontier models for dangerous capabilities. CoRR abs/2403.13793 (2024)
104. Pols, P.: The Unified Kill Chain. Master's thesis (2017). https://www.unifiedkillchain.com/assets/The-Unified-Kill-Chain.pdf
105. Proofpoint: Cyber kill chain overview (2025). https://www.proofpoint.com/us/threat-reference/cyber-kill-chain
106. PTES Working Group: Penetration Testing Execution Standard (PTES) (2014). http://www.pentest-standard.org/
107. PTF Project: Penetration Testing Framework v0.59 (2013). http://www.vulnerabilityassessment.co.uk/Penetration%20Test.html
108. Qi, X. et al.: Fine-tuning aligned LMs compromises safety. abs/2310.03693 (2024)
109. Raman, R. et al.: ChatGPT vs Bard for ethical hacking. Comput. Secur. 140 '24
110. Rani, N. et al.: CyberExplorer: Benchmarking LLM offensive security. CoRR abs/2602.08023 (2026)
111. Regina, M. et al.: Text data augmentation for spear-phishing detection. CoRR abs/2007.02033 (2021)
112. Sarraute, C. et al.: POMDPs make better hackers. Proc. AAAI (2012)
113. Sarraute, C. et al.: PenTesting as POMDP solving. CoRR abs/1306.4714 (2013)
114. Scarfone, K.: Cybersecurity skills gap. TechTarget (2025). https://www.techtarget.com/searchsecurity/tip/Cybersecurity-skills-gap-Why-it-exists-and-how-to-address-it
115. Schaeffer, R. et al.: Emergent abilities of LLMs are a mirage? Proc. NeurIPS '23
116. Schick, T. : Toolformer: LMs teach themselves to use tools. Proc. NeurIPS '23
117. Shah, S. et al.: VAPT techniques overview. J. Comput. Virol. Hack. Tech. 11 '15
118. Shen, Y. et al.: HuggingGPT. Proc. NeurIPS (2023)
119. Shojaee, P. et al.: Illusion of thinking in reasoning models. abs/2506.06941 (2025)
120. Shravan, K. et al.: Penetration testing: A review. Compusoft 3(4) (2014)

121. Significant Gravitas: AutoGPT. Github (2023).
122. Singer, B. et al.: Feasibility of LLM multistage network attacks. CoRR abs/2501.16466 (2025)
123. Singh, A. et al.: Metasploit Penetration Testing Cookbook (3rd ed.). Packt (2018)
124. Sommer, R. et al.: Outside the closed world: ML for NID. Proc. IEEE S&P (2010)
125. Stability AI: StableLM launch (2023).
126. Stefinko, Y. et al.: Manual vs automated penetration testing. Proc. TCSET (2016)
127. Strobelt, H. et al.: Interactive/visual prompt engineering. IEEE TVCG 29(1) '22
128. Strom, B.E. et al.: MITRE ATT&CK: Design and philosophy (2018)
129. Swanson, M. et al.: NIST SP 800-115: Info. Sec. Testing & Assessment (2008).
130. Teknium: OpenHermes-2.5-Mistral-7B (2023). https://huggingface.co/teknium/OpenHermes-2.5-Mistral-7B
131. Touvron, H. et al.: LLaMA: Open & efficient foundation LMs. CoRR abs/2302.13971 (2023)
132. Touvron, H. et al.: LLaMA 2: Open foundation & chat models. CoRR abs/2307.09288 (2023)
133. Vaswani, A. et al.: Attention is all you need. Proc. NeurIPS (2017)
134. Wang, G. et al.: OpenChat: Mixed-quality data for open LMs. CoRR abs/2309.11235 (2024)
135. Wang, L. et al.: Plan-and-solve prompting. CoRR abs/2305.04091 (2023)
136. Wang, X. et al.: MINT: Multi-turn tool-use eval for LLMs. Proc. ICLR (2024)
137. Wang, Y. et al.: TDAG: Dynamic task decomposition & agent generation. Neural Netw. 185 (2025)
138. Wei, J. et al.: Finetuned LMs are zero-shot learners. Proc. ICLR (2022).
139. Wei, J. et al.: Emergent abilities of LMs. TMLR abs/2206.07682 (2022)
140. Wei, J. et al.: Chain-of-thought prompting. Proc. NeurIPS (2022)
141. Weidman, G.: Penetration Testing: Hands-On Intro. to Hacking. No Starch (2014).
142. Wu, B. et al.: AutoPT: Towards fully automated web PenTesting. IEEE TIFS '25
143. xAI: Grok-1 (2024). https://x.ai/news/grok-os
144. Xi, Z. et al.: LLM-based agents: Survey. Sci. China Inf. Sci. 68 (2025)
145. Xu, H. et al.: LLMs for cyber security: Systematic review. ACM TOSEM '25
146. Xu, J. et al.: AutoAttacker: LLM-guided cyber-attacks. abs/2403.01038 (2024)
147. Yang, J. et al.: SWE-agent: Agent-computer interfaces for SE. Proc. NeurIPS '24
148. Yang, X. et al.: Shadow alignment. CoRR abs/2310.02949 (2023)
149. Yao, S. et al.: Tree of thoughts. Proc. NeurIPS (2023)
150. Yao, S. et al.: ReAct: Synergizing reasoning & acting in LMs. 2210.03629 (2023)
151. Yao, Y. et al.: LLM sec. & priv.: The good, bad, ugly. High-Conf. Comp. 4(2) '24
152. Yigit, Y. et al.: GenAI methods in cybersecurity: Review. abs/2403.08701 (2024)
153. Zhan, Q. et al.: Removing RLHF protections in GPT-4 via fine-tuning. Proc. NAACL (2024)
154. Zhang, A.K. et al.: Cybench: Eval. cybersec. capabilities of LMs. Proc. ICLR '25
155. Zhang, J. et al.: LLMs meet cybersec.: Systematic review. Cybersec. 8(1) (2025)
156. Zhang, R. et al.: LLaMA-Adapter: Efficient fine-tuning. abs/2303.16199 (2023)
157. Zhao, W.X. et al.: Survey of large language models. CoRR abs/2303.18223 (2025)
158. Zheng, L. et al.: LLM-as-a-Judge with MT-Bench. Proc. NeurIPS (2023)
159. Zhou, L. et al.: Larger instructable LMs become less reliable. Nature 634 (2024)
160. Zhu, Y. et al.: CVE-Bench: Exploiting real-world web vulns. 2503.17332 (2025)
161. Zhu, Y. et al.: Teams of LLM agents exploit zero-day vulns. abs/2406.01637 (2025)
162. Zou, A. et al.: Transferable adversarial attacks on aligned LMs. 2307.15043 (2023)
163. Takaronis, M. *et al.*: ICSSPulse: A modular LLM-assisted platform for industrial control system penetration testing. CoRR abs/2602.20663 (2026)

Table 5: Attack-phase coverage of LLM-aided PenTesting systems

#	System	Initial Exploitation	Post-Exploitation	Notes
1	PenTest2.0	✗	✓	PrivEsc (assumed-breach)
2	PenForge	✓	✗	Web vuln. exploitation
3	HackingBuddyGPT	✗	✓	Linux PrivEsc (assumed-breach)
4	AutoPT	✓	✗	Web exploitation tasks
5	Cybench	NA	NA	Benchmark (CTF evaluation)
6	LLM Pentest Benchmark	NA	NA	Benchmark (PenTesting tasks)
7	Cochise	✗	✓	Enterprise AD (assumed-breach)
8	PenTest++	✓	✗	Multi-phase PenTesting
9	AI-Augmented Ethical Hacking	✓	✓	Demonstrative multi-phase PoC
10	PentestGPT	✓	✓	Interactive PenTesting guidance
11	PenHeal	✓	✓	PenTesting automation with remediation module
12	GenAI for PenTesting	✓	✓	Conceptual general framework
13	Wintermute	✗	✓	Host exploitation focus
14	WiFiPenTester	✓	✗	Wireless reconnaissance and attack planning
15	LLM Agents for Autonomous Website Hacking	✓	✗	Autonomous web vulnerability discovery/exploit
16	AutoAttacker	✓	✓	Primarily post-exploitation with limited initial access (multi-stage benchmark)
17	LLM Agents Exploit 1-day Vulns	✓	✗	Exploitation of 14 CVEs + 1 academic vulnerability
18	HPTSA	✓	✗	Exploitation of 14 zero-day vulns (relative to LLM knowledge cutoff)
19	RapidPen	✓	✗	Prioritises rapid initial access
20	Incalmo	✓	✓	Multi-stage offensive operations
21	VulnBot	✓	✗	Vuln exploitation agent
22	AutoPenBench	NA	NA	LLM PenTesting benchmark (automated evaluation)
23	HackSynth	NA	NA	LLM agent + benchmark for CTF-style tasks
24	CyberExplorer	✓	✗	Open-environment web exploitation benchmark
25	ICSSPulse	✓	✗	ICS protocols: scan + R/W; no PrivEsc/lateral
26	BreachSeek	✓	✗	Single-host automated exploitation + reporting
27	PTGroup	✓	✗	Auto. exploitation via ReAct + multi-agent prompt chains
28	AutoSecAgent	✓	✓	Semi-automated end-to-end workflow and remediation guidance

Legend: Initial Exploitation = gaining initial access; Post-Exploitation = activities after compromise (e.g., PrivEsc, lateral movement, pivoting, persistence); NA = Not Applicable (benchmark frameworks rather than operational PenTesting systems).

Table 6: Core capabilities of PenTesting systems across key attack phases

#	System	Scan	Enum	Exploit	Priv-Esc	Lateral Move	Persist	Report Gen
1	PenTest2.0	✗	✓	✗	✓	✗	✗	✓
2	PenForge	✗	✓	✓	✗	✗	✗	✗
3	HackingBuddyGPT	✗	✓	✗	✓	✗	✗	✗
4	AutoPT	✗	✓	✓	✗	✗	✗	✗
5	Cybench	NA	NA	NA	NA	NA	NA	NA
6	LLM Pentest Benchmark	NA	NA	NA	NA	NA	NA	NA
7	Cochise	✗	✓	✗	✓	✓	✓	✗
8	PenTest++	✓	✓	✓	✗	✗	✗	✓
9	AI-Augmented Ethical Hacking	NA	NA	NA	NA	NA	NA	NA
10	PentestGPT	✗	✓	✓	✗	✗	✗	✓
11	PenHeal*	✗	✓	✓	✗	✗	✗	✓
12	GenAI for PenTesting	NA	NA	NA	NA	NA	NA	NA
13	Wintermute	✗	✓	✗	✓	✗	✗	✗
14	WiFiPenTester	✓	✓	✓	✗	✗	✗	✓
15	LLM Agents for Auto. Website Hacking	✗	✓	✓	✗	✗	✗	✗
16	AutoAttacker	✗	✓	✓	✓	✓	✓	✗
17	LLM Agents Exploit 1-day Vulns	✗	✓	✓	✗	✗	✗	✗
18	HPTSA	✗	✓	✓	✗	✗	✗	✗
19	RapidPen	✗	✓	✓	✗	✗	✗	✗
20	Incalmo	✓	✓	✓	✓	✓	✗†	✗
21	VulnBot	✓	✓	✓	✗	✗	✗	✗
22	AutoPenBench	NA	NA	NA	NA	NA	NA	NA
23	HackSynth	NA	NA	NA	NA	NA	NA	NA
24	CyberExplorer	✗	✓	✓	✗	✗	✗	✗
25	ICSSPulse	✓	✓	✓	✗	✗	✗	✓
26	BreachSeek	✗	✓	✓	✗	✗	✗	✓
27	PTGroup	✗	✓	✓	✗	✗	✗	✗
28	AutoSecAgent*	✗	✓	✓	✓	✓	✗	✓

Legend: Scan = network host discovery; Enum = service and vulnerability discovery; Exploit = initial system compromise; PrivEsc = privilege escalation; Lateral Move = movement to additional hosts; Persist = maintain long-term access; Report Gen = automated report generation. NA = Not Applicable for benchmark datasets or conceptual frameworks that do not execute attack steps. CyberExplorer is not marked NA because its agents actively perform reconnaissance and exploitation within the simulated environment.

† Incalmo deploys malware via a C&C server but does not explicitly implement persistence as a dedicated attack phase.

* The system includes remediation guidance, which is outside the standard PenTesting attack phases.

Table 7: Autonomy, architecture, execution model, and governance of LLM-aided PenTesting systems

#	System	Auto	Arch	Exec	Gov
1	PenTest2.0	HITL/SA	PE/Multi	Shell/Tools	Strong
2	PenForge	SA	PE/Multi	Web tools	Med
3	HackingBuddyGPT	HITL/SA	PE	Shell	Med
4	AutoPT	FA	PE	Web tools	Low
5	Cybench	—	—	—	—
6	LLM Pentest Benchmark	—	—	—	—
7	Cochise	SA/FA	PE/Multi	Tools	Med
8	PenTest++	HITL/SA	Single/PE	Shell/Tools	Strong
9	AI-Augmented Ethical Hacking	HD	Single	Advice	Strong
10	PentestGPT	HD	Single	Advice/Tools	Med
11	PenHeal	SA	PE	Shell/Tools	Low
12	GenAI for PenTesting	HD	Single	Advice	Strong
13	Wintermute	SA	PE	Shell/Tools	Med
14	WiFiPenTester	HITL/SA	PE	Tools	Strong
15	LLM Agents for Autonomous Website Hacking	SA	Multi	Web tools	Low
16	AutoAttacker	FA	Multi	Tools	Low
17	LLM Agents Exploit 1-day Vulns	SA	Single	Tools	Low
18	HPTSA	SA	Team	Tools	Low
19	RapidPen	FA	PE	Tools	Low
20	Incalmo	FA	PE/Multi	Tools	Low
21	VulnBot	SA	PE/Multi	Tools	Low/Med
22	AutoPenBench	—	—	—	—
23	HackSynth	—	—	—	—
24	CyberExplorer	FA	Multi	Tools	Low
25	ICSSPulse	HD	Single	Tools	Strong
26	BreachSeek	SA	PE/Multi	Shell/Tools	Low
27	PTGroup	SA	Multi	Shell/Tools	Low
28	AutoSecAgent	SA	PE/Multi	Shell/Tools	Strong

Legend: Auto. = autonomy (HD: human-driven; HITL: human-in-the-loop; SA: semi-autonomous; FA: fully autonomous). Arch. = architecture (Single; Multi; PE: planner–executor; Team: hierarchical teams). Exec. = execution (Advice: text-only; Shell: OS command execution; Tools: calls external PenTesting tools/APIs; Web tools: browser/HTTP actions). Gov. = governance/safety (Low/Med/Strong: approval gates, blacklists, containment, logging).
—= Not Applicable (benchmarking/evaluation frameworks rather than operational PenTesting systems).

Table 8: Use of state-of-the-art reasoning and guidance techniques

#	System	CoT	ReAct	Hints	RAG	Tools	PTT
1	PenTest2.0	✓	✗	✓	✓	✓	✓
2	PenForge	✗	✓	✗	✓	✓	✗
3	HackingBuddyGPT	✓	✓	✗	✗	✓	✗
4	AutoPT	✗	✗	✗	✗	✓	✗
5	Cybench	—	—	—	—	—	—
6	LLM Pentest Benchmark	—	—	—	—	—	—
7	Cochise	✓	✓	✗	✗	✓	✓
8	PenTest++	✓	✗	✗	✗	✓	✗
9	AI-Augmented Ethical Hacking	—	—	—	—	—	—
10	PentestGPT	✓	✗	✓	✗	✓	✗
11	PenHeal	✗	✗	✗	✓	✓	✓
12	GenAI for PenTesting	—	—	—	—	—	—
14	WiFiPenTester	✓	✗	✓	✓	✓	✗
15	LLM Agents for Autonomous Website Hacking	✗	✓	✗	✓	✓	✗
16	AutoAttacker	✓	✓	✗	✓	✓	✗
17	LLM Agents Exploit 1-day Vulns	✗	✓	✗	✓	✓	✗
18	HPTSA	✗	✓	✗	✗	✓	✗
19	RapidPen	✓	✓	✗	✓	✓	✓
20	Incalmo	✗	✓	✗	✓	✓	✗
21	VulnBot	✗	✓	✗	✓	✓	✗
22	AutoPenBench	—	—	—	—	—	—
23	HackSynth	—	—	—	—	—	—
24	CyberExplorer	—	—	—	—	—	—
25	ICSSPulse	✗	✗	✗	✗	✓	✗
26	BreachSeek	✗	✗	✗	✗	✓	✗
27	PTGroup	✗	✓	✗	✗	✓	✗
28	AutoSecAgent[§]	✗	✓	✗	✓	✓	✗

Legend: CoT = chain-of-thought; ReAct = reasoning and acting; PE = planner–executor architecture; Hints = human hints during execution; RAG = retrieval-augmented generation; Tools = programmatic tool invocation; PTT = PenTesting task-tree; VulnBot uses a PTG (PenTesting Graph), not a PTT.
[§] AutoSecAgent uses Recursive Memory Embedding (RME) and Context Loss Error (CLE).

Table 9: Platform, target, availability, and eval. of LLM PenTesting systems

#	System	Platform	Target	OSrc	Eval. Method
1	PenTest2.0 [10]	Linux	Network / Host	Yes	Controlled lab
2	PenForge [64]	Cross	Web	Yes	CVE-Bench (40 web vulns)
3	HackingBuddyGPT [59]	Linux	Network / Host	Yes	PrivEsc tasks
4	AutoPT [142]	Linux	Web	Yes	Custom benchmark (20 vulns)
5	Cybench [154]	Linux	Benchmark	Yes	Benchmark eval
6	LLM Pentest Bench. [68]	Cross	Benchmark	Partial	Benchmark eval
7	Cochise [57]	Windows	Network	No	AD lab (GOAD)
8	PenTest++ [4]	Linux	Network / Host	Yes	Controlled lab
9	AI-Aug. Ethical Hacking [12]	Cross	Network / Host	No	Case studies
10	PentestGPT [36]	Linux	Network + Web	Yes	Case studies
11	PenHeal [65]	Linux	Network + Web	No	Lab tests (Metasploitable2)
12	GenAI for PenTesting [63]	Linux	Host	No	Lab tests (PumpkinFestival)
13	Wintermute [55]	Linux	Host	Yes	CTF environment
14	CyberExplorer [110]	Cross	Benchmark	No	Benchmark eval (40 vuln. web services)
15	WiFiPenTester [8]	Cross	Wireless Network	Partial	Wireless lab
16	HPTSA [161]	Linux	Web	No	Controlled 14 zero-day CVE benchmark
17	VulnBot [76]	Linux	Network / Web	Yes	Bench. eval. (AutoPenBench + AI-Pentest-Bench; vuln. hosts)
18	RapidPen [90]	Container / Docker	Remote Host	No	HTB lab (Legacy)
19	Incalmo [122]	Linux	Multi-host Network	Yes	Net benchmark (MHBench)
20	AutoPenBench [51]	Linux	Benchmark	Yes	Benchmark eval
21	HackSynth [88]	Linux	Benchmark	Yes	CTF benchmark (PicoCTF:120 & OverTheWire:80)
22	AutoAttacker [14	Cross	Host (post-exploitation)	No	Cont. lab (14 MITRE ATT&CK tasks)
23	LLM Agents Exploit 1-day Vulns [44]	Cross	Web	No	15 CVEs
24	LLMs Hack Websites [45]	Cross	Web	No	15 Web-related vuln. tasks
25	ICSSPulse [163]	Cross	ICS Prot. (Modbus/OPC UA)	Yes	Simulated ICS scenarios
26	BreachSeek [19]	Linux	Network + Web	Yes	(Metasploitable2)
27	PTGroup [37]	Linux	Network / Host	No	Controlled lab (Metasploitable2, Vulhub, Win VMs)
28	AutoSecAgent [7]	Linux	Network / Web / AD	No	Controlled lab + red-team simulation

Platform: Experimental OS (Linux typically implies Kali Linux). **OSrc (Open Source):** Yes = publicly available code; Partial = limited release (e.g. no 6 uses external, open-source tools such as PentestGPT and VulnHub VMs, but no standalone benchmark code released); No = no software released (prompts or steps may still be described in the paper).