A Critical Review of LLM Agents for Automated Penetration Testing: Benchmark Realism and Evidence Grading

Yinhang Zhou[1*],Weihua Xu[1]

[1]Guangzhou Nanyang Polytechnic College, Guangzhou 510900, China
[*]Corresponding author: 827379713@qq.com

Abstract: Penetration testing has long resisted full automation because it requires contextual reasoning, adaptive tool use, and experience-driven decision making across the Penetration Testing Execution Standard (PTES) lifecycle. Recent advances in large language models (LLMs), agentic reasoning, and multi-tool invocation have shifted the field from rule-based scripts and attack-graph reinforcement-learning planners toward interactive agents capable of planning, reflection, and environment interaction. The accompanying literature, however, suffers from inconsistent evaluation protocols, opaque scaffolding, and indiscriminate mixing of peer-reviewed papers with vendor self-reports, rendering headline claims difficult to compare. This work presents a critical narrative review with structured evidence charting of AI-assisted automated penetration testing between January 2023 and May 2026, retaining DeepExploit and AutoPentest-DRL as historical anchors. We introduce a dual-axis framework that annotates every finding by environment realism (R1–R4, from single-challenge CTFs to live enterprise networks) and source authority (A–D, from peer review to vendor self-report). Frontier agents reach 22–44% on R1 single-challenge CTFs (D-CIPHER), 79.17% on R2 AutoPenBench subtasks with a domain-tuned 32B model (xOffense), and 71.4% on the AISI 95-task expert tier (GPT-5.5), yet only 20–30% end-to-end completion on the R3 32-step TLO cyber range. On R4 live enterprise networks, the ARTEMIS multi-agent framework placed second overall against ten human professionals on an 8,000-host university network, submitting nine valid vulnerabilities at an 82% acceptance rate. Robust autonomy remains weak in long-horizon exploitation, Active Directory lateral movement, and defender-present environments. We argue that automated penetration testing is best framed as a systems-engineering problem rather than a single-model race, and we distinguish engineering-tractable Type A capability gaps from planning-bottlenecked Type B failures that require base-model reasoning improvements or reinforcement-learning post-training.

Keywords: automated penetration testing; large language models; AI agents; benchmark realism; evidence stratification

I Introduction

A. Background and Motivation

Penetration testing is an engineering practice for assessing a system's resistance to attack. It spans pre-engagement interactions, information gathering, vulnerability analysis, exploitation, privilege escalation, lateral movement, and reporting, and has long depended on contextual decision making by testers across mainstream security testing standards and attack knowledge bases [1]–[4]. Early automation relied on rules, signatures, and attack graphs; reinforcement-learning approaches such as DeepExploit and AutoPentest-DRL introduced state modeling, but training cost and limited transferability prevented them from scaling [2], [5]–[6].

With the emergence of LLMs, frontier models combined with reasoning paradigms

(CoT, ReAct, Reflexion) and multi-agent frameworks (AutoGen) have endowed agents with planning, reflection, and multi-tool invocation, giving rise to systems such as PentestGPT, CAI, and ARTEMIS [7]–[15]. The accompanying growth of the literature, however, has been marked by inconsistent evaluation protocols, opaque scaffolding, and mixed citation of peer-reviewed papers, official reports, and preprints, making cross-system comparison difficult and creating an urgent need for a critical review centered on evidence grades and evaluation realism [15]–[21].

B. Differences from Existing Surveys

Existing surveys on LLM security mostly focus on model vulnerabilities, malware, or code security, whereas existing surveys on automated penetration testing center on tool inventories or CTF performance [16]–[17], [21]–[24]. This work differs along three dimensions: it treats penetration testing as an end-to-end agentic workflow rather than a set of isolated tasks [7], [15], [18], [25]; it differentiates external validity using benchmark realism along the R1–R4 scale [15]–[20]; and it explicitly stratifies peer-reviewed papers, official evaluations, preprints, and vendor self-reports [24], [26]–[27].

C. Research Questions

This work is organized around four research questions. RQ1: the evolutionary trajectory of AI-assisted automated penetration testing from rules and reinforcement learning to LLM agents. RQ2: the capability maturity and remaining gaps of existing systems across the PTES stages. RQ3: differences in external validity among evaluations on CTFs, containerized benchmarks, cyber ranges, and real enterprise networks. RQ4: the safety governance and defensive perspectives on LLM-based penetration testing agents.

D. Contributions

The main contributions of this work are threefold. (1) We establish a dual-axis analytical framework of “environment realism (R1–R4) × evidence grade (A–D)” so that every conclusion is traceable to its supporting evidence strength (§II.E, §IV) [15]–[20]. (2) We jointly leverage the ARTEMIS R4 live-university-network human-AI comparison [15] and the Type A / Type B failure dichotomy of Deng et al. (2026) [21] to construct a coherent failure-mode analysis (§IV.D, §V.F). (3) We place Cybench, NYU CTF Bench, AutoPenBench, CAIBench, the AISI and CAISI evaluations, and the ARTEMIS real-network study on a unified R1→R4 realism axis, providing a cross-benchmark quantitative comparison table (§IV.E) [15]–[20], [22].

II Review Design and Methodology

A. Type and Positioning of the Review

This work is positioned as a critical narrative review combined with structured evidence charting. The research objects in this field are highly heterogeneous and iterate on a monthly cadence, making strict adherence to PRISMA difficult to match; the dual-axis framework of R1–R4 realism × A–D evidence grade is the principal methodological contribution of this work beyond PRISMA, ensuring that each conclusion is traceable to its evidence strength.

B. Search Sources and Time Window

The search window spans 2023-01-01 through 2026-05-11. Early foundational studies such as DeepExploit and AutoPentest-DRL are retained as exceptions in order to preserve the evolutionary trajectory of the field [1]–[2], [5]–[6].

Search sources include Web of Science, Scopus, IEEE Xplore, the ACM Digital Library, ScienceDirect, SpringerLink, arXiv, OpenReview, USENIX, and the ACL Anthology, together with NIST, the UK AI Security Institute (AISI), MITRE, OWASP, OpenAI system cards, and official project repositories.

For methodological transparency, the principal database queries used to construct

the candidate pool are reported below. This disclosure documents the search procedure but does not constitute a claim of full systematic-review compliance; the work remains a critical narrative review (see §II.A).

Web of Science: TS=((“penetration test*” OR pentest* OR “offensive security” OR “red team*” OR “adversary emulation”) AND (“large language model*” OR LLM* OR “language model agent*” OR “AI agent*” OR agentic OR “reinforcement learning”) AND (automat* OR autonom* OR benchmark* OR evaluat* OR exploit* OR reconnaissance)) AND PY=(2023–2026).

Scopus: TITLE-ABS-KEY((“penetration test*” OR pentest* OR “offensive security” OR “red team*” OR “adversary emulation”) AND (“large language model*” OR llm* OR “language model agent*” OR “AI agent*” OR agentic OR “reinforcement learning”) AND (automat* OR autonom* OR benchmark* OR evaluat* OR exploit* OR reconnaissance)) AND PUBYEAR > 2022.

IEEE Xplore: (“All Metadata”:”penetration testing” OR “All Metadata”:pentest OR “All Metadata”:”offensive security” OR “All Metadata”:”red teaming”) AND (“All Metadata”:”large language model” OR “All Metadata”:LLM OR “All Metadata”:”AI agent” OR “All Metadata”:”reinforcement learning”) AND (“All Metadata”:automation OR “All Metadata”:autonomous OR “All Metadata”:benchmark OR “All Metadata”:evaluation).

ACM Digital Library: [[All: “penetration testing”] OR [All: pentest] OR [All: “offensive security”] OR [All: “red teaming”]] AND [[All: “large language model”] OR [All: LLM] OR [All: “AI agent”] OR [All: agentic] OR [All: “reinforcement learning”]] AND [[All: automation] OR [All: autonomous] OR [All: benchmark] OR [All: evaluation]].

arXiv / OpenReview: (“penetration testing” OR pentest OR “offensive security”) AND (“large language model” OR LLM OR agent OR “reinforcement learning”).

Targeted official-source queries: site:aisi.gov.uk cyber evaluation model penetration testing; site:nist.gov CAISI cybersecurity evaluation; site:mitre.org ATT&CK CALDERA; site:openai.com system card cybersecurity; site:owasp.org LLM applications 2025.

C. Inclusion and Exclusion Criteria

Included studies must directly address automated penetration testing, agent-based offensive security, cyber-range exercises, or CTF solving; report a traceable methodology and quantitative results; originate from peer-reviewed venues, formal conferences, official reports, system cards, preprints, or open-source projects of methodological value; and be mappable to the PTES lifecycle, MITRE ATT&CK, or the benchmark-realism framework.

Excluded: marketing material or self-promotion without methodological detail; studies that discuss general LLM safety risks without addressing penetration testing or cyber evaluation; early preprints that overlap substantially with a formal version; and materials without identifiable source, version, or date.

D. Screening, Deduplication, and Data Charting

Records were deduplicated by DOI, title, author, and year; disputed items were independently re-checked at the title-abstract and full-text stages. Data were charted along the following dimensions: system / benchmark, publication type, PTES stage, ATT&CK mapping, R-level, tool coupling, human baseline, quantitative metrics, cost, A–D grade, and source [24], [26]–[27].

E. Evidence Grades and Environment Realism

To avoid conflating CTF scores, official evaluations, preprints, and author self-reports as equivalent evidence, this work adopts a four-

level evidence coding scheme A–D. The grade is determined along three core dimensions: methodological transparency (whether task definitions, evaluation protocols, scaffolding, and token budgets are fully disclosed); independent reproducibility (whether data, code, and run cost are public); and source authority (peer review > official evaluation > high-transparency preprint > vendor self-report / README). The specific rules are as follows [15], [19]–[20], [24], [26]–[27]:

Table I. Evidence-Grade Coding (A–D).

Grade	Definition	Usage in this paper
A	Peer-reviewed papers or formal benchmark papers; methodology and results sufficiently transparent	Can support major conclusions
B	Authoritative official evaluations, system cards, or high-transparency preprints	Can support major conclusions, with limitations noted
C	Preprints or open-source project papers with sufficient methodology but no independent verification	Used as supplementary evidence
D	Author self-evaluation, vendor blog, README, or methodologically opaque materials	Used only as background; cannot support strong conclusions

Benchmark realism is partitioned into four progressively higher external-validity levels, R1 through R4. The classification rests on four criteria: whether the task is single-step or multi-step; whether it involves a single host or multiple hosts; whether it includes organizational assets, identities, logs, EDR, and defenders; and whether the evaluation is conducted in a real production network against human professionals. The specific rules are as follows:

Table II. Environment-Realism Coding (R1–R4).

Code	Definition	Example
R1	Static knowledge items, single-challenge CTFs, isolated challenges	Security Q&A; single-challenge Web/Misc/Pwn
R2	Containerized multi-step tasks; single-host ranges; controlled benchmarks	AutoPenBench; local HTB subsets; Cybench sub-tasks
R3	Multi-host cyber range; simulated enterprise network; no active defender	AISI TLO / Cooling Tower 32-step tasks
R4	Real enterprise / campus / production network, or head-to-head against human professionals	ARTEMIS live-university-network evaluation; commercial bug bounty

All quantitative findings in the text are jointly annotated as “R-level × evidence grade.” For example, “(R2, A)” indicates peer-reviewed evidence from a containerized multi-step task that can support a major conclusion, whereas “(R4, B)” indicates a high-transparency preprint from a real production environment that can support a major conclusion but with explicit independent-reproducibility limitations.

F. Limitations of This Review

As a critical narrative review, this work has three classes of limitations that should be made explicit. First, dual independent screening and protocol pre-registration were not performed, so selection bias is possible; this is mitigated by exposing the evidence strength behind each conclusion through the R1–R4 × A–D dual-axis

annotation. Second, the snapshot as of 2026-05-11 cannot reflect work published thereafter, so quantitative results should be updated periodically. Third, vendor system cards and government evaluations lack anonymous peer review; readers should be mindful of vendor conflicts of interest and of the absence of active defenders in the evaluation networks, both of which limit external generalizability (see the limitations discussion at the end of §IV.D) [24], [26]–[27].

III Technical Evolution of AI-Assisted Automated Penetration Testing

A. Evolution Timeline

We divide the evolution of AI-assisted automated penetration testing into four generationally distinct phases. Table 3 presents the timeline organized by year, representative systems, reasoning paradigm, and attainable upper bound on environment realism, which facilitates cross-phase comparison.

Table III. Evolution Timeline of AI-Assisted Penetration Testing (stratified by R1–R4 realism ceiling).

Phase	Time window	Representative systems	Reasoning paradigm	Tool invocation	Realism ceiling
I Rule-based scripts	≤2017	Metasploit, AutoSploit, Sn1per, BeEF	Hard-coded payloads; static CVE matching	Shell + RPC + module composition	R1–R2
II Reinforcement learning and attack graphs	2017–2022	DeepExploit, AutoPentest-DRL, HA-DRL	DQN / A3C / PPO + MulVAL attack graph	Metasploit RPC + discrete action space	R2
III LLM single-agent	2023–2024	PentestGPT, HackingBuddyGPT, AutoAttacker, CIPHER, Nebula	ReAct + CoT + human-in-the-loop prompt engineering	Function Calling + Bash channel	R2
IV Multi-agent and specialized models	2024–present	CAI, PentestAgent, D-CIPHER, VulnBot, Aurora, Pentest-R1, xOffense, ARTEMIS	Plan-and-Solve [47], ToT [11], Reflexion [10]; multi-agent coordination; neuro-symbolic integration; RL post-training	MCP + sub-agent spawning + RAG + CALDERA integration	R3–R4

The transition from rule-based scripts to LLM agents has been driven jointly by three forces: leaps in model reasoning capability (GPT-4o, Claude 3.5 Sonnet, DeepSeek-R1, GPT-5.5), the engineering maturation of agent middleware (LangChain, AutoGen, CrewAI, and others), and the maturation of evaluation paradigms (NYU CTF Bench, Cybench, AutoPenBench, the AISI suite, and others) [13]–[14].

B. Layered Architecture of AI Penetration Agents (8 Pillars)

Mainstream multi-agent penetration systems have converged in engineering practice on a common “eight-layer architecture,” which the CAI official documentation calls the “8 Pillars Architecture” [8]. Table 4 presents this layered structure together with the responsibilities of each layer and representative implementations.

Table IV. Layered Architecture of AI Penetration Agents (8 Pillars).

Layer	Responsibility	Typical implementations / components	Representative systems
1 User / Operator	Task assignment, guardrail approval, results review	CLI / TUI / Web UI / API	CAI, PentestGPT, BoxPwnr
2 Planning	Generate the macro kill chain (Recon → Exploit → Pivot → Loot)	Plan-and-Solve [47], Tree-of-Thought [11], Aurora classical planning [40]	PentestAgent, Aurora, D-CIPHER

续表2

Layer	Responsibility	Typical implementations / components	Representative systems
3 Reasoning	Decision model + ReAct loop + Reflexion reflection	GPT-5.x, Claude Mythos, DeepSeek-R1, alias1, Pentest-R1	All
4 Multi-Agent (collaboration)	Role partitioning: Recon / Exploit / Privesc / Reporter / Critic / Verifier	Message bus / MCP / blackboard / sub-agent spawn	PentestAgent, D-CIPHER, ARTEMIS, CAI
5 Tool / Action	Structured function calls and pseudo-terminal actions	Function Calling (nmap, sqlmap, ffuf, msfconsole, pwntools); pseudo-terminal (bash, python, msfconsole)	All
6 Memory	Short-term conversation buffer; long-term episodic memory; RAG knowledge retrieval	Conversation buffer + Summarizer; HackTricks/CVE RAG; notes.md / findings.json	AutoAttacker, CAI, PentestAgent
7 Safety & Audit	Input-side prompt-injection detection; output-side dangerous-command interception; full trace logging	Input Guardrails; Output Guardrails; MLflow / SQLite / OpenTelemetry	CAI, ARTEMIS, CALDERA + mcp
8 Environment	Isolation of execution environment and target ingress	Docker / Kali / Vagrant VM / cyber range / real target	All

The MITRE caldera-mcp plugin, PentestAgent's spawn_mcp_agent, and ARTEMIS's on-demand sub-agents are concrete instantiations of this eight-layer structure at the collaboration and tool layers. Throughout this paper, "alias1" refers to the security-tuned reasoning model line released by the CAI team [8], [29]; it serves as the Reasoning-layer component in several CAI-based evaluations, including the Dragos OT CTF results cited in §V.C.1 and Tables 7 and 9.

IV Evaluation Ecosystem and Benchmark Realism

A. CTFs and Single-Challenge Tasks (R1)

CTF tasks are reproducible, well-bounded, simple to score, and well-suited to rapid iteration, and they have been the principal vehicle for early evaluation of LLM offensive capabilities. NYU CTF Bench, Cybench, certain subsets of CAIBench, and many HackTheBox/TryHackMe sub-tasks all fall in this category. Existing evidence shows that LLMs have made marked progress in Web, Misc, Crypto, script analysis, and vulnerability explanation, but remain clearly limited in Pwn, reverse engineering, kernel exploitation, and tasks that require complex binary debugging [16]–[17], [22].

The limitations of CTFs are also clear. They typically lack realistic asset management, business context, network noise, defender responses, and compliance boundaries, and the task objectives are usually designed to be validated by a single flag. CTF scores therefore cannot be extrapolated directly to real-world enterprise penetration capability [15]–[17].

B. Containerized Benchmarks and Multi-Step Ranges (R2)

Works such as AutoPenBench and PentestEval have pushed evaluation from single-challenge tasks toward multi-step workflows [18], [28]. Containerized benchmarks allow controlled environments, reproducible experiments, and the recording of success rate, runtime, cost, and degree of human intervention. They are closer to real penetration workflows than single-challenge CTFs, yet they typically still lack the identity systems, domain trusts, log monitoring, EDR, traffic noise, and business constraints found in organizational networks. Containerized benchmarks are therefore best suited to evaluating tool use, local planning, error recovery, and task completion rates, and

should not be treated as proof of full red-team capability.

The AWD (Attack with Defense) format requires teams to attack and defend simultaneously under identical initial conditions, with flags rotated periodically; this places extremely high demands on the real-time responsiveness and continual tactical adjustment of LLM agents. The Attack & Defense category of CAIBench simulates exactly this scenario, with SOTA models reporting success rates between 20% and 40% on such adversarial CTFs, well below top human performance (R2, C) [17]. CAI reports a patching success rate above the industry baseline in this setting, achieved by introducing a Retester Agent and a Patch Verification Agent that run patching in parallel with attack (R2, D) [8], [29]. The core methodological value of AWD and the CAIBench A&D category is that they evaluate "attack" and "defense" jointly within a single benchmark, partially redressing the absence of defender feedback in pure CTF evaluations.

From a purple/blue-team perspective, Mantis (2024) [45] demonstrates a defensive prompt-injection approach: by exploiting the susceptibility of LLMs to adversarial input, Mantis injects prompts into banners, error pages, or logs returned to an attacker so that, on parsing these "honeypot prompts," the attacker's LLM agent is induced to execute harmless or self-defeating actions. This is an early instance of AI-vs-AI defense and foreshadows a new form of attack–defense competition between AI agents (R2, C). Mechanisms in the Mantis family [45] impose a direct security requirement on automated penetration agents: every tool output originating from a target must pass through a prompt-injection detection layer before entering the LLM context.

C. Cyber Ranges and Real-Network Evaluation (R3–R4)

AISI's long-horizon cyber ranges and TLO/Cooling Tower style tasks represent a higher-realism direction in evaluation [19]–[20], [30]. They place models in multi-step, multi-host, long-horizon goal-directed environments that emphasize task planning, state persistence, and failure recovery. The AISI "The Last Ones (TLO)" task is a 32-step enterprise-network intrusion simulation spanning four subnets and roughly twenty hosts, modeling the full attack chain from an unauthenticated starting point to a protected database; AISI estimates that human experts require about 20 hours to complete the scope. Across ten attempts, GPT-5.5 completed the chain end-to-end twice and Claude Mythos Preview three times, whereas GPT-5.4 and Opus 4.7 each achieved 0/10 (R3, B). A separate seven-step industrial-control-system simulation, "Cooling Tower," has not yet been completed by any model, with the bottleneck appearing in upstream IT stages rather than in the ICS portion itself.

AISI observed in its scaling-curve analysis that attack success rate grows monotonically with the reasoning-token budget without exhibiting a saturation knee, suggesting that the long-horizon planning bottleneck is engineering- and compute-related rather than fundamental — yet R3 evaluations still generally lack active defenders, real-time alerting, and operational noise from enterprise IT.

D. ARTEMIS: R4 Live-University-Network Human-AI Comparison

Lin et al. [15] placed AI penetration agents in the active research network of a large university (approximately 8,000 hosts across 12 subnets) and evaluated them head-to-head against 10 human professional penetration testers and 6 publicly available AI agents, while proposing a multi-agent framework called ARTEMIS (R4, B).

This study provides a critical calibration point for capability evaluation in real-world business settings.

ARTEMIS introduces three key innovations: dynamic just-in-time prompt synthesis, the spawning of independent sub-agents per suspected attack surface, and a three-tier triage by exploitability, impact, and duplication prior to submission. Under a unified scoring protocol, ARTEMIS ranked second overall, surpassing 9 of 10 human professionals, with 9 valid vulnerabilities submitted at an 82% acceptance rate; the comparator agents Codex and CyAgent performed below most human testers. Lin et al. note that AI agents enjoy advantages in systematic enumeration and parallel exploitation, with some configurations costing approximately USD 18 per hour, compared with a market rate of about USD 60 per hour for human professional testers.

This evaluation advances PTES-stage capability assessment from "simulation competence" to "production competence," and corroborates the AISI Expert-tier characterization of GPT-5.5 as "approaching a junior penetration tester" [19]. The sub-agent partitioning sidesteps the long-horizon planning difficulty of a single agent and offers an engineering-complementary path to the Type B failure modes discussed in §V.F. Limitations must also be stated explicitly: the target network contained no active defender or EDR; vulnerability submissions were judged by the host organization; training and deployment details were not fully disclosed; and the human baseline was limited to 10 testers — for these reasons we grade the evidence as B rather than A.

E. Cross-Benchmark Quantitative Results (Ordered R1→R4)

Table 5 lists 10 representative quantitative results spanning the four realism levels, ordered from R1 to R4. Each row is annotated with metric category and sample size; values across different metric categories (Solve rate / Pass rate / Completion rate / Hint-augmented / End-to-end attempts / Real-network multi-metric) are not directly comparable.

TABLE V. . Cross-Benchmark Quantitative Results for LLM Penetration Systems (ordered R1→R4; values within a given metric-category column are comparable, but values across columns are not directly comparable due to differing benchmark definitions, sample sizes, and scoring protocols; E.L. = evidence level).

Realism	System / model	Benchmark	Metric category (not comparable across categories)	Value (with sample size / uncertainty)	E.L.	Source
R1	D-CIPHER multi-agent	NYU CTF Bench	Solve rate	22.0%	A	[31]
R1	D-CIPHER multi-agent	HTB subset	Solve rate	44.0%	A	[31]
R1	Claude 3.5 Sonnet (no subtasks)	Cybench	Solve rate	~17.5%	A	[16]
R1	GPT-4o single-agent	AutoPenBench	Solve rate	21.21%	A	[18]
R1	GPT-5.5 / Mythos / GPT-5.4 / Opus 4.7	AISI Expert tier (95 tasks)	Task pass rate (n=95)	71.4 / 68.6 / 52.4 / 48.6% (±SEM see [19])	B	[19]
R2	xOffense (Qwen3-32B + CoT fine-tuned)	AutoPenBench subtasks	Subtask completion rate	79.17%	C	[32]
R2	GPT-4-Turbo + guidance + Reflexion	HackingBuddyGPT Linux privesc	Hint-augmented success	83% vs 33% baseline	A	[33]
R3	GPT-5.5	AISI TLO 32-step	End-to-end attempts (n=10)	2 / 10 attempts	B	[19], [30]

续表2

Realism	System / model	Benchmark	Metric category (not comparable across categories)	Value (with sample size / uncertainty)	E.L.	Source
R3	Claude Mythos Preview	AISI TLO 32-step	End-to-end attempts (n=10)	3 / 10 attempts	B	[19], [30]
R4	ARTEMIS	8,000-host / 12-subnet university network	Real-network rank / valid vulns / acceptance	2nd place / 9 valid vulns / 82% acceptance	B	[15]

Table 5 supports the following three judgments, each bound to its corresponding evidence grade.

First, on standard academic CTF single-challenge tasks (R1), top multi-agent systems cluster in the 20%–45% range — the 22%–44% achieved by D-CIPHER across three benchmarks and the 17.5% of Claude 3.5 Sonnet on Cybench fall within the same band (A-level evidence).

Second, the marginal benefit of domain fine-tuning or extended-chain reasoning on a narrow domain substantially exceeds the benefit of switching to a larger base model — xOffense (Qwen3-32B) achieves 79.17% on AutoPenBench sub-tasks, far above the 21.21% of a single GPT-4o agent; adding guidance and Reflexion to HackingBuddyGPT raises success rate from 33% to 83%. This judgment is jointly supported by A- and C-level evidence.

Third, solve rate and reproducibility decay in tandem as realism increases, but the metrics involved are not directly comparable — the AISI Expert-tier 95-task pass rate ranges from 48.6% to 71.4% across the four frontier models evaluated (±SEM as reported in [19]), whereas the same set of models reach the end of the TLO 32-step chain in only 2/10 and 3/10 attempts (n = 10, absolute success rather than a rate) [19], [30]; ARTEMIS ranks near the top of the real-university-network evaluation but only at second place, not by an overwhelming margin. CAIBench also reports a structural gap of roughly 70% knowledge performance versus 20%–40% adversarial performance [17], which mirrors the same decay pattern.

F. Comparability Issues in Benchmark Design

Cross-study comparison must address the following variables: task realism, tool permissions, internet access, allowance of human hints, token budget, model version, agent scaffold, retry budget on failure, cost, scoring rubric, and human verification procedure. If these variables are not controlled or reported, direct model ranking is not warranted [15]–[20]. Empirical results from CAIBench show that the same model paired with different agentic frameworks can produce up to 2.6× variance in A&D CTF performance, indicating that scaffolding affects real performance more than the model itself [17]. Merves et al. [34] further confirm this through a factorial experiment on 10 frontier models across 200 NYU CTF tasks, where a Kali Linux tool stack yields a +9.5 percentage-point improvement in solve rate over Ubuntu, while auto-prompting and category hints actually hurt performance in tool-complete environments. Real-CVE benchmarks such as CVE-Bench [35] push external validity into the R2–R3 range.

Statements of the form “Model A outperforms Model B” should therefore be replaced by more conservative, conditional formulations: under a specific benchmark, a specific set of tool permissions, a specific token budget, and a specific scaffold, a given system reports a higher success rate; cross-environment generalization remains to be established.

V Evidence Synthesis along the PTES Lifecycle

A. PTES × Capability Maturity × Evidence-Grade Matrix

Table 6 presents an assessment of AI automation capability across the seven PTES stages, jointly annotated with a three-level qualitative maturity scale (mature / supportive / weak) and the A/B/C/D evidence grade, with the upper bound on environment realism made explicit for each conclusion.

Table VI. PTES Stage × AI Capability Maturity × Evidence-Grade Matrix.

PTES stage	Representative AI capability	Maturity	Supporting realism	Evidence grade	Main bottleneck
Pre-engagement Interactions	Scope confirmation, compliance checks, template generation	Weak (human-led)	R1	C	Legal / semantic boundaries; requires human communication
Information Gathering	Nmap / Shodan / OSINT parsing; cross-tool summarization	Mature	R1–R4	A	Long-output truncation; over-interpretation of weak signals
Threat Modeling	Asset / threat modeling; risk-matrix generation	Supportive	R2	C	Lacks internal organizational context
Vulnerability Analysis	Interpretation of scanner output; CVE association; PoC reading	Supportive, trending toward mature	R1–R3	A	False-positive identification still relies on humans; CVE recall dominates
Exploitation	Payload generation; Web exploitation; command-chain construction	Supportive	R1–R2	A	Pwn-class tasks and bypassing ASLR / CFG remain weak; poor state persistence
Post-Exploitation – privilege escalation	Linux / Windows local privilege escalation	Supportive (mature when well-bounded)	R2	A	Windows / AD privesc weaker than Linux
Post-Exploitation – lateral movement	AD enumeration; ACL abuse; Kerberos / NTLM	Weak	R3–R4	B	State management and BloodHound-graph reasoning insufficient
Post-Exploitation – persistence	Selection of stealthy mechanisms (WMI / COM / IFEO etc.)	Weak	R2–R3	C	Lacks defender awareness and OPSEC assessment
Reporting	Markdown / PDF report generation; CVSS scoring	Mature	R1–R4	A	Unverified conclusions and exaggerated remediation advice

Maturity definitions are as follows: “mature” denotes that frontier systems approach or match the human professional baseline across multiple independent evaluations (A/B-level evidence); “supportive” denotes a stable benefit in controlled settings but cross-environment generalization that is not yet sufficiently established; “weak” denotes performance materially below human level in most public evaluations, or evidence too sparse to support a judgment. Pre-engagement interactions, compliance boundaries, and dual-use risks rely heavily on human decision making — see §VII.A.

B. Information Gathering and Reconnaissance

Reconnaissance is one of the most stable application stages for LLM agents. Models can interpret outputs from Nmap, Masscan, HTTP fingerprinting, DNS, certificates, directory scans, and open-source intelligence, and can organize fragmented findings into hypotheses for subsequent steps. The advantage stems from language understanding and cross-tool summarization rather than from replacing the scanners themselves [7], [25], [36]. AutoAttacker uses a Summarizer module to compress each

round of shell output, effectively mitigating context-window pressure [36].

The limitations are that models tend to over-interpret weak signals — misclassifying generic banners, default pages, or error pages as specific frameworks — and may overlook biases introduced by scanner noise, WAFs, CDNs, proxies, and network segmentation. Reconnaissance is therefore well suited to a human-in-the-loop setup in which the model generates candidate hypotheses and a human confirms priorities [7], [15], [25].

C. Exploitation and Attack-Chain Construction

Exploitation is the stage with the largest evaluation discrepancies. For public exploits, standard Web vulnerabilities, weak passwords, simple injections, and known service misconfigurations, LLMs can effectively generate strategies and command sequences. The 2024 series of works by Fang et al. demonstrates that LLM agents can autonomously hack websites, exploit one-day vulnerabilities, and, through multi-agent collaboration, address zero-day real-world vulnerabilities [37]–[39]. For binary exploitation, kernel vulnerabilities, complex authentication bypass, and business-logic vulnerabilities, current systems remain unstable [16]–[18], [28]. The core difficulty of attack-chain construction is state persistence: a single penetration may involve multiple entry points, ephemeral credentials, sessions, proxy chains, privilege boundaries, and failure paths. Without structured memory and plan validation, models tend to repeat ineffective attempts, forget already-obtained information, or continue under erroneous assumptions [10], [19], [31], [40].

Discussing "exploitation" in aggregate masks substantial internal heterogeneity. Table 7 subdivides exploitation into four categories — Web, binary (Pwn), OT/ICS, and AD lateral movement — and presents representative evidence, maturity, and bottlenecks for each.

Table VII. Capability Gaps across Four Exploitation Task Categories.

Exploitation category	Representative evidence	Maturity	Realism	Evidence grade	Bottleneck
Web exploitation	xOffense (Qwen3-32B + CoT) on AutoPenBench sub-tasks 79.17%; PentestAgent black-box Web penetration	Supportive, trending toward mature	R1–R2	A/C	Complex business-logic vulnerabilities and multi-step chained exploitation
Binary / Pwn	Cybench GPT-4o Pwn sub-task success rate < 15%	Weak	R1	A	ROP-gadget search; precise stack-layout control; ASLR bypass; formal reasoning
OT / ICS	CAI alias1 Dragos OT CTF 32/34; AISI Cooling Tower 7-step end-to-end 0/N	Supportive (CTF) / Weak (cyber range)	R1–R3	B/D	ICS protocol parsing; safety interlocks and physical-state reasoning; upstream IT stages often fail first
AD lateral movement	ARTEMIS 2nd place on real university network; AISI TLO 32-step 2–3/10	Weak, trending toward supportive	R3–R4	B	BloodHound ACL-graph reasoning; Kerberos / NTLM; ADCS; GPO; state management

Table 7 reveals a fact easily obscured by aggregated claims: the apparent success rate of AI on Web exploitation and OT CTFs may differ by an order of magnitude from its success rate on Pwn and real Active Directory lateral movement. Any assertion about "AI penetration capability" should therefore specify the exploitation category, on pain of a "leveling" misinterpretation.

D. Privilege Escalation, Lateral Movement, and Post-Exploitation

Linux privilege escalation, Windows privilege escalation, Active Directory lateral movement, Kerberos/NTLM, ADCS, ACL abuse, GPO, credential dumping, and persistence remain the stages with the most pronounced capability gaps. HackingBuddyGPT and related studies show that, on well-bounded Linux privilege-escalation tasks, LLM agents can raise success rates: GPT-4-Turbo achieves a 33% success rate without guidance, rising to 83% with high-level hints and a reflection mechanism, matching and in some configurations exceeding the 75% baseline of human professional pentesters (R2, A) [33]. By contrast, Llama3 reaches only 0%–33% and GPT-3.5-Turbo only 16%–50%, indicating that privilege-escalation capability varies substantially across base-model families.

Lateral movement and post-exploitation in a complete Active Directory enterprise domain still demand complex state management, privilege reasoning, and tight security-boundary control. AutoAttacker demonstrates a RAG-augmented multi-stage attack chain for lateral movement inside Metasploitable II, but the authors acknowledge that the environment is too simplified to represent a real AD domain [36]. The AISI TLO evaluation (2026) is the first to place lateral movement in a realistic enterprise AD forest context, requiring the model to autonomously complete an attack chain across four subnets and roughly twenty hosts — GPT-5.5 completes 2/10 and Mythos Preview 3/10, materially above the 0/10 of GPT-5.4 and Opus 4.7 [19]. Single-host privilege-escalation success rates therefore cannot be extrapolated to end-to-end capability in an enterprise domain. Future benchmarks should report stage-level metrics for local privilege escalation, domain enumeration, path discovery, credential abuse, lateral movement, and goal attainment separately [15], [18], [28].

E. Vulnerability Identification and Report Generation

LLMs are of clear value in vulnerability explanation, CVE association, classification of misconfigurations, PoC reading, and code-snippet review. They can convert scanner output into actionable verification steps and help testers understand vulnerability preconditions and exploitation limitations. PentestGPT, AutoAttacker, and PentestAgent have all demonstrated this capability [7], [25], [36]. The vulnerability-identification stage is, however, prone to hallucination: models may fabricate CVE identifiers, confuse affected version ranges, or generate nonexistent parameters or incorrect exploitation conditions. High-risk conclusions must be verified against official advisories, patch notes, source code, hands-on validation, or authoritative databases [23], [41]–[42].

Report writing is currently one of the most readily deployable LLM applications. Models can organize tool outputs, evidentiary screenshots, reproduction steps, impact analysis, and remediation recommendations into a structured report. On the industrial vulnerability-disclosure side, HackerOne's Hai agent (including the Hai Triage component) [46] integrates a comparable LLM triad — exploitability verification, noise deduplication, and remediation closure — into a real bug-bounty workflow. The principal risks are that reports may contain unverified conclusions, exaggerated impact, or mis-prescribed remediation. Report generation should therefore be bound to evidence chains, command logs, screenshots, timestamps, and human review [7], [23], [25], [41]–[42].

F. Failure-Mode Taxonomy: Type A and Type B

Deng et al. (arXiv:2602.17622, 2026) systematically re-tested five mainstream LLM-based penetration implementations and proposed a dual failure-mode taxonomy: Type A (capability gaps) and Type B (planning / state management)

(R1–R3, B) [21]. Type A failures arise from incomplete tool sets, inappropriate scaffold choices, or insufficient context management, and can be repaired directly by engineering improvements. Type B failures stem from fundamental limitations in long-horizon planning and state management; they depend on improved reasoning ability in the base model or on dedicated reinforcement-learning post-training, and the marginal benefit of increasing token budget or rewording prompts decays rapidly.

Empirical results show that Type A dominates on simple benchmarks; as benchmark difficulty rises, the Type B share grows rapidly and becomes the overwhelming cause of failure on HTB-level and AD lateral-movement tasks. Table 8 maps mainstream mitigation techniques to Type A and Type B and indicates the applicable scope of each.

Table VIII. Type A / Type B Failure Modes × Mitigation Techniques.

Failure mode	Typical manifestations	Representative mitigations	Applicable scope	Evidence grade
Type A (capability gap)	Tool-call errors; command hallucination; CVE misrecall; parsing failures due to inappropriate scaffold	Function-Calling schemas; dry-run; tool-response fact-checking agent; RAG (HackTricks / CVE / ExploitDB)	Largest gains on R1–R2 single-challenge and containerized tasks; residual Type B persists at R3+	A
Type A (capability gap)	Context-window overflow; long-output truncation; loss of key findings	Summarizer (AutoAttacker); contextual memory (CAI); sub-agent context partitioning (ARTEMIS, PentestAgent)	Substantially mitigates engineering bottlenecks in R2–R3 multi-step tasks; does not address long-horizon causal consistency	A/B
Type B (planning / state management)	Forgetting the initial objective after step 30; losing already-collected credentials; cycling through completed actions; continuing under wrong assumptions	Reflexion (short-horizon mitigation); two-stage RL post-training (Pentest-R1); neuro-symbolic classical-planning constraints (Aurora)	Reflexion yields nonlinear gains on R2 privesc; RL post-training and causally-consistent planning are the central directions for R3–R4 long-horizon tasks	A/C
Type B (planning / state management)	Failure to discover AD lateral-movement paths; confusion across multi-entry-point session switching	Structured asset / credential / session / attack-path graphs; engineering workaround via ARTEMIS-style on-demand sub-agent spawning	Engineering workarounds yield localized gains at R4 but cannot substitute for improvements in base-model reasoning depth	B/C

The engineering implication of this framework is clear: Reflexion and RAG primarily address Type A, whereas two-stage RL in the style of Pentest-R1 and causally-consistent planning in the style of Aurora are the central paths to addressing Type B. By spawning sub-agents on demand, ARTEMIS structurally sidesteps the long-horizon planning difficulty of a single agent — this is an engineering workaround rather than a root-cause fix, which explains why, despite surpassing 9 of 10 humans on the R4 real-network evaluation, it still ranks second rather than overwhelmingly first.

F. Representative Systems and Evidence Matrix

This section presents the mainstream AI penetration / red-team systems jointly along three axes: realism, evidence grade, and the conclusions they can support. Table 9 selects 10 representative systems spanning academic peer review, official evaluations, representative preprints, and the open-source ecosystem.

Table VIII. Type A / Type B Failure Modes × Mitigation Techniques.

System / study	Type	Main scenario	Realism	Evidence grade	Conclusions supported	Limitations
PentestGPT [7]	Peer-reviewed system paper (USENIX Sec 2024)	HTB / CTF interactive penetration	R2	A	LLM dialog framework contributes positively to stage guidance and task decomposition	Early versions relied on humans to execute commands
Cybench [16]	Benchmark paper (ICLR 2025 Oral)	Expert-grade containerized CTF tasks	R1–R2	A	Scaffolding and tool-permission settings materially affect model safety and capability evaluation	Tasks subject to public-writeup contamination risk
AutoPenBench [18]	Benchmark paper (EMNLP 2025 Industry)	Containerized multi-step penetration tasks	R2	A	Meaningful evaluation of mainstream agents in task-assist mode	Lacks AD / EDR / real-network scenarios
PentestAgent [25]	Conference paper (ACM ASIA CCS 2025)	Multi-agent Web penetration	R2	A	Multi-agent orchestration and RAG assist stage decomposition and vulnerability verification	Scenario coverage skews to Web
D-CIPHER [31]	Red-team preprint (NYU)	Containerized CTF	R1–R2	C	Achieves 22%–44% solve rate across NYU CTF / Cybench / HTB subsets	Limited to containerized CTFs; Type B failures pronounced on long-horizon tasks
Pentest-R1 [43]	Preprint	End-to-end HTB	R2	C	Two-stage RL (SFT + GRPO / DPO) improves handling of Type B tasks	High training cost; cross-environment generalization unconfirmed
CAI / alias series [8], [29]	Preprint + open-source framework	CTF / OT CTF / bug bounty	R1–R2	C–D	Suitable as an open-source baseline for ecosystem and tool exploration; alias1 reaches 32/34 on the Dragos OT CTF	Engineering-heavy system design; lacks rigorous peer review
BoxPwnr	Open-source individual project	Continuous evaluation on HTB / THM / PortSwigger	R2	D	A rare open-source agent that runs at scale and continuously on public CTF ranges with public traces	Research / experimental in nature; lacks peer review and aggregated solve rate
ARTEMIS [15]	Real-scale evaluation preprint	8,000-host university network / 12 weeks	R4	B	Multi-agent system submits 9 valid vulnerabilities at an 82% acceptance rate in a real large-scale network, with detailed human-AI comparison evidence	Training / evaluation details not fully disclosed; no EDR / defender
AISI GPT-5.5 / TLO evaluation [19]	Official government evaluation	Expert-tier evaluation + long-horizon cyber range	R1–R3	B	Frontier models have short-horizon offensive capability, but end-to-end long-horizon stability is only 20%–30%	Limited evaluation sample size and human-baseline scale

Table 9 shows that the stronger evidence concentrates in R1–R2 environments: reconnaissance, vulnerability analysis, local exploitation, and report generation all receive A-level support. A-level evidence in R3–R4 environments remains scarce; current R4 conclusions rely almost entirely on a single B-level study, ARTEMIS. This work therefore does not adopt summary-level evaluations — any cross-system comparison should be confined to the "R-level × evidence grade × supported conclusion" triple [15], [19]–[20].

VII Risks, Governance, and Defensive Perspectives

A. Dual Use and Authorization Boundaries

Automated penetration testing is inherently dual-use: LLM agents lower the threshold for attack-chain orchestration and tool use, and may also be misused for unauthorized scanning, exploitation, social-engineering assistance, and malicious automation. Research and deployment must therefore make authorization boundaries, target scope, log retention, human approval gates, data minimization, and accountability explicit [3], [23], [41]–[42]. OpenAI, Anthropic, AISI, and NIST have all placed "autonomous cyber attack" on the frontier-model risk-assessment checklist; the AISI report shows that GPT-5.5's cost per successful attack on expert-tier tasks has fallen by several multiples relative to its predecessor, and, taken together with the capability extrapolation of AutoAttacker, this indicates that LLM advances are turning "expert-level long-horizon attacks" into "low-skill, large-scale, routine operations" [19], [36].

B. Hallucination, Excessive Agency, and Prompt Injection

The core risks of LLM-based penetration agents include command hallucination, CVE hallucination, incorrect remediation advice, excessive agency, prompt injection, and context poisoning. Adversaries or target systems can induce agents to take inappropriate actions via web pages, banners, logs, or returned data (see the Mantis [45] defensive prompt injection discussed in §IV.B.1). Countermeasures include instruction hierarchies, minimal tool privileges, approval gates for high-risk commands, output validation, sandboxed execution, and replayable audit trails [23]. The CAI Guardrails framework provides a practical reference across multiple pipelines: input-side prompt-injection detection, output-side dangerous-command interception, tool-level permission checks, and post-decoding re-validation of encoded payloads [8], [29].

C. The Defender's Perspective

Defenders should not focus solely on whether AI agents can compromise a target; they should also focus on how to detect, attribute, and constrain LLM-driven attacks. Observable signals include anomalous tool-call cadence, automated scanning patterns, repeated failed commands, prompt-like payloads, traces of LLM-generated reports or comments, cross-host lateral-movement paths, and anomalous API calls; CALDERA, ATT&CK, SIEM/EDR, honeypots, and purple-team exercises can jointly form a defender-side evaluation framework [1]–[2]. Mantis (§IV.B.1) [45] demonstrates an early form of AI-vs-AI defense — exploiting the LLM's susceptibility to adversarial input, the defender actively injects prompts on the scanned side to backfire on the attacker's agent.

D. Purple-Team Coordination and Human-AI Collaboration

The current industry consensus is to retain a Human-in-the-Loop (HITL) deployment posture. CAI explicitly declares itself a "semi-autonomous operation" on the grounds that "fully autonomous cybersecurity systems remain immature on complex tasks" [29]. Representative HITL practices include pre-command confirmation, review at key decision points, halt-before-critical-vulnerability gates, kill switches, and operator overrides. The AISI finding that GPT-5.5's safety mitigations were broken by a six-hour universal jailbreak further reinforces the necessity of HITL and layered Guardrails in production environments [19].

VIII Research Gaps and Future Directions

A. From CTF Scores to External Validity in Real Environments

Future benchmarks should reduce the centrality of single-flag solve rate and shift toward reporting staged task completion, false-positive

rate, cost, failure modes, frequency of human intervention, harm control, and external validity. Public evaluations in R3–R4 environments remain severely insufficient and constitute a focus for follow-up research [15], [19]–[20]. The "meta-benchmark" paradigm advocated by CAIBench is poised to become the mainstream evaluation form — assessment must span knowledge, attack, defense, privacy, robotics, and OT scenarios, rather than CTFs alone [17].

B. Human Baselines and Human-AI Collaboration

Many studies lack a rigorous human baseline, making it impossible to determine whether agent gains come from model capability, tool automation, or evaluation design. Future studies should jointly report performance for experts, junior staff, humans without AI assistance, humans with AI assistance, and fully autonomous agents. The ARTEMIS head-to-head comparison against 10 human professionals on an R4 real network is the most methodologically exemplary case to date [15], [18].

C. Long-Horizon Planning and State Management (Addressing Type B)

Long-horizon failure is the most significant current bottleneck. Future systems will need structured asset graphs, credential graphs, session graphs, attack-path graphs, failure memory, plan validators, and resumable executors. Simply enlarging the context window does not resolve state consistency or causal planning [10], [19], [31], [40]. Two-stage RL post-training in the Pentest-R1 style and causally-consistent planning in the Aurora style are two complementary paths to overcoming Type B failures (§V.F).

D. Cost, Latency, and Sustainability

Frontier models can incur very large token consumption, tool-call counts, and retry costs on long-horizon penetration tasks. Studies should report mean cost, maximum cost, runtime, retry counts on failure, and human-intervention cost; otherwise the practical deployment value is hard to assess [18]–[20]. Notably, the USD 18/hour versus USD 60/hour economic gap reported by ARTEMIS, together with the several-fold reduction in GPT-5.5's cost per successful attack reported by AISI relative to its predecessor, indicates that the economic threshold for at-scale deployment of AI-driven attacks is falling rapidly [15], [19].

E. Specialized Security Models and Auditable Toolchains

More security-specialized models, red-team-specific agents, open-source security LLMs, and in-house penetration assistants (such as CIPHER [44] and the alias1 line) are likely to emerge. Their deployment, however, must be paired with tool-permission control, audit logging, data isolation, handling of sensitive information, and compliance review. Fully autonomous attack systems should not be the default deployment form in most enterprise environments [23], [41]–[42]. The proliferation of the MCP standard will allow agents from different vendors to interconnect like microservices, but it also imports tool supply-chain risk (OWASP LLM08) into red-team infrastructure.

IX Conclusion

Using a critical narrative review combined with structured evidence charting, this work reconstructs the technical trajectory of AI-assisted automated penetration testing from rules, reinforcement learning, and attack graphs to LLM agents and multi-agent systems [5]–[7], [9]–[14], [25], [40]. Current evidence does not support the strong conclusion that "fully autonomous red-teaming is mature": long-horizon state management, enterprise-domain lateral movement, generalization to real networks, the presence of defenders, cost control, and security auditability remain major bottlenecks [15], [19]–[20], [31], [33], [40].

This work reaches four core judgments. (1) Scaffolding, tools, and memory design influence outcomes no less than the model itself, so benchmark realism must become a central evaluation dimension; the R1–R4 × A–D matrix provides an operational analytical framework for this purpose [15]–[19]. (2) ARTEMIS surpasses 9 of 10 humans on the R4 university network but still ranks only second, corroborating the AISI Expert-tier characterization of GPT-5.5 as “approaching a junior penetration tester”; together, the two findings calibrate the realistic position of frontier capability in 2025–2026 [15], [19]. (3) The Type A / Type B failure dichotomy provides a clear boundary — Reflexion and RAG primarily address Type A, whereas Pentest-R1-style RL post-training and Aurora-style causally-consistent planning are the central paths to addressing Type B [21]. (4) Automated penetration testing is a high-risk dual-use technology whose economic threshold for at-scale deployment is dropping rapidly (ARTEMIS USD 18/h versus humans USD 60/h); research and deployment must align with governance frameworks from NIST, OWASP, and MITRE, and must maintain HITL, auditability, and least privilege [1]–[2], [19], [23], [36], [41]–[42].

Future work should prioritize building open, reproducible, and stratifiable R2–R4 evaluation infrastructure that incorporates human baselines, cost, failure modes, and defender response; only when capability expansion and risk control advance in tandem can LLM-based penetration agents move from demonstration systems to responsible security-engineering tools [15], [18]–[20], [23], [28], [41]–[42].

References

[1] MITRE Corporation, "MITRE CALDERA: Automated adversary emulation platform," GitHub repository, 2014. [Online]. Available: https://github.com/mitre/caldera. [Accessed: May 28, 2025].

[2] MITRE Corporation, "ATT&CK: Adversarial Tactics, Techniques, and Common Knowledge," MITRE, ongoing.

[3] National Institute of Standards and Technology, "Technical Guide to Information Security Testing and Assessment," NIST SP 800-115, 2008.

[4] The PTES Team, The Penetration Testing Execution Standard, PTES Documentation, version 1.1, 2016.

[5]I. Takaesu, “Deep Exploit: Fully Automatic Penetration Test Tool Using Reinforcement Learning,” CODE BLUE 2019, Bluebox, Tokyo, Japan, Oct. 2019.

[6] Z. Hu, R. Beuran, and Y. Tan, “Automated Penetration Testing Using Deep Reinforcement Learning,” in 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Genoa, Italy, 2020, pp. 2–10, doi: 10.1109/EuroSPW51379.2020.00010.

[7] G. Deng, Y. Liu, V. Mayoral-Vilches, P. Liu, Y. Li, Y. Xu, T. Zhang, Y. Liu, M. Pinzger, and S. Rass, "PentestGPT: Evaluating and Harnessing Large Language Models for Automated Penetration Testing," in Proc. 33rd USENIX Security Symposium, 2024, pp. 847–864.

[8] V. Mayoral-Vilches, L. J. Navarrete-Lozano, M. Sanz-Gómez, L. Salas Espejo, M. Crespo-Álvarez, F. Oca-Gonzalez, F. Balassone, A. Glera-Picón, U. Ayucar-Carbajo, J. A. Ruiz-Alcalde, S. Rass, M. Pinzger, and E. Gil-Uriarte, “CAI: An Open, Bug Bounty-Ready Cybersecurity AI,” arXiv preprint arXiv:2504.06017, 2025.

[9] S. Yao et al., "ReAct: Synergizing Reasoning and Acting in Language Models," in Proc. ICLR, 2023.

[10]N. Shinn et al., "Reflexion: Language Agents with Verbal Reinforcement Learning," in Proc. NeurIPS, 2023.

[11] S. Yao et al., "Tree of Thoughts: Deliberate Problem Solving with Large Language Models," in Proc. NeurIPS, 2023.

[12] J. Wei et al., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," in Proc. NeurIPS, 2022.

[13] Q. Wu et al., "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation," arXiv preprint arXiv:2308.08155, 2023.

[14] S. Hong et al., "MetaGPT: Meta Programming for a Multi-Agent Collaborative Framework," in Proc. ICLR, 2024.

[15] J. W. Lin et al., "Comparing AI Agents to Cybersecurity Professionals in Real-World Penetration Testing," arXiv preprint arXiv:2512.09882, 2025.

[16] A. K. Zhang et al., "Cybench: A Framework for Evaluating Cybersecurity Capabilities and Risks of Language Models," in Proc. ICLR, 2025.

[17] M. Sanz-Gómez et al., "Cybersecurity AI Benchmark (CAIBench): A Meta-Benchmark for Evaluating Cybersecurity AI Agents," arXiv preprint arXiv:2510.24317, 2025.

[18] L. Gioacchini, A. Delsanto, I. Drago, M. Mellia, G. Siracusano, and R. Bifulco, “AutoPenBench: A Vulnerability Testing Benchmark for Generative Agents,” in Proc. 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track, Suzhou, China: Association for Computational Linguistics, 2025, pp. 1615–1624, doi: 10.18653/v1/2025.emnlp-industry.114.

[19] UK AI Security Institute, "Our Evaluation of OpenAI's GPT-5.5 Cyber Capabilities," AISI Blog/Technical Evaluation, 2026.

[20] Center for AI Standards and Innovation, National Institute of Standards and Technology, “CAISI Evaluation of DeepSeek V4 Pro,” NIST News/Updates, May 1, 2026, updated May 2, 2026.

[21] G. Deng et al., "What Makes a Good LLM Agent for Real-world Penetration Testing?" arXiv preprint arXiv:2602.17622, 2026.

[22] M. Shao et al., "NYU CTF Bench: A Scalable Open-Source Benchmark Dataset for Evaluating LLMs in Offensive Security," in Proc. NeurIPS Datasets and Benchmarks Track, 2024.

[23] OWASP Foundation, "OWASP Top 10 for Large Language Model Applications 2025," OWASP, 2025.

[24] M. J. Page et al., "The PRISMA 2020 Statement: An Updated Guideline for Reporting Systematic Reviews," BMJ, vol. 372, n71, 2021.

[25] X. Shen et al., "PentestAgent: Incorporating LLM Agents to Automated Penetration Testing," in Proc. ACM ASIA CCS, 2025.

[26] A. C. Tricco, E. Lillie, W. Zarin, K. K. O’Brien, H. Colquhoun, D. Levac, D. Moher, M. D. J. Peters, T. Horsley, L. Weeks, S. Hempel, et al., “PRISMA Extension for Scoping Reviews (PRISMA-ScR): Checklist and Explanation,” Annals of Internal Medicine, vol. 169, no. 7, pp. 467–473, 2018, doi: 10.7326/M18-0850.

[27] M. L. Rethlefsen et al., "PRISMA-S: An Extension to the PRISMA Statement for Reporting Literature Searches in Systematic Reviews," Systematic Reviews, vol. 10, article 39, 2021.

[28] R. Yang, M. Cheng, G. Deng, T. Zhang, J. Wang, and X. Xie, “PentestEval: Benchmarking LLM-based Penetration Testing with Modular and Stage-Level Design,” arXiv preprint arXiv:2512.14233, 2025.

[29] V. Mayoral-Vilches, L. J. Navarrete-Lozano, F. Balassone, M. Sanz-Gómez, C. R. J. Veas Chavez, M. del Mundo de Torres, and V. Turiel, “Cybersecurity AI: The World’s Top AI Agent for Security Capture-the-Flag (CTF),” arXiv preprint arXiv:2512.02654, 2025.

[30] L. Folkerts, W. Payne, S. Inman, P. Giavridis, J. Skinner, S. Deverett, J. Aung, E. Zorer, M. Schmatz, M. Ghanem, J. Wilkinson, A. Steer, V. Hong, and J. Wang, “Measuring AI Agents’ Progress on Multi-Step Cyber Attack Scenarios,” arXiv preprint arXiv:2603.11214, 2026.

[31] M. Udeshi et al., "D-CIPHER: Dynamic Collaborative Intelligent Multi-Agent System with Planner and Heterogeneous Executors for Offensive Security," arXiv preprint arXiv:2502.10931, 2025.

[32] P. D. Luong, L. T. G. Bao, N. V. K. Tam, D. H. N. Khoa, N. H. Quyen, V.-H. Pham, and P. T. Duy, “xOffense: An Autonomous Multi-Agent Framework for Penetration Testing with Domain-Adapted Large Language Models,” arXiv preprint arXiv:2509.13021, 2025.

[33] A. Happe, A. Kaplan, and J. Cito, "LLMs as Hackers: Autonomous Linux Privilege Escalation Attacks," Empirical Software Engineering, vol. 31, article 70, 2026.

[34] T. H. Merves, M. H. Conaway, J. M. Escobar, H. T. Otal, and U. Tatar, “Systematic Capability Benchmarking of Frontier Large Language Models for Offensive Cyber Tasks,” arXiv preprint arXiv:2604.17159, 2026.

[35] Y. Zhu, A. Kellermann, D. Bowman, P. Li, A. Gupta, A. Danda, R. Fang, C. Jensen, E. Ihli, J. Benn, J. Geronimo, A. Dhir, S. Rao, K. Yu, T. Stone, and D. Kang, “CVE-Bench: A Benchmark for AI Agents’ Ability to Exploit Real-World Web Application Vulnerabilities,” arXiv preprint arXiv:2503.17332, 2025.

[36] J. Xu, J. W. Stokes, G. McDonald, X. Bai, D. Marshall, S. Wang, A. Swaminathan, and Z. Li, "AutoAttacker: A Large Language Model Guided System to Implement Automatic Cyber-attacks," arXiv preprint arXiv:2403.01038, 2024.

[37] R. Fang, R. Bindu, A. Gupta, Q. Zhan, and D. Kang, “LLM Agents can Autonomously Hack Websites,” arXiv preprint arXiv:2402.06664, 2024.

[38] R. Fang, R. Bindu, A. Gupta, and D. Kang, “LLM Agents can Autonomously Exploit One-day Vulnerabilities,” arXiv preprint arXiv:2404.08144, 2024.

[39] Y. Zhu, A. Kellermann, A. Gupta, P. Li, R. Fang, R. Bindu, and D. Kang, “Teams of LLM Agents can Exploit Zero-Day Vulnerabilities,” arXiv preprint arXiv:2406.01637, 2024. Accepted by EACL 2026.

[40] L. Wang et al., "From Sands to Mansions: Towards Automated Cyberattack Emulation with Classical Planning and Large Language Models," arXiv preprint arXiv:2407.16928, 2024.

[41] National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, 2023.

[42] National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile," NIST AI 600-1, 2024.

[43] H. Kong, D. Hu, J. Ge, L. Li, H. Li, and T. Li, “Pentest-R1: Towards Autonomous Penetration Testing Reasoning Optimized via Two-Stage Reinforcement Learning,” arXiv preprint arXiv:2508.07382, 2025.

[44] D. Pratama et al., "CIPHER: Cybersecurity Intelligent Penetration-Testing Helper for Ethical Researcher," Sensors, vol. 24, no. 21, 2024.

[45] D. Pasquini, E. M. Kornaropoulos, and G. Ateniese, “Hacking Back the AI-Hacker: Prompt Injection as a Defense Against LLM-driven Cyberattacks,” arXiv preprint arXiv:2410.20911, 2024.

[46]HackerOne, “Hai: AI Co-pilot and Hai Triage for Security Teams,” HackerOne Product Documentation, 2024. [Online]. Available: https://www.hackerone.com/ai/hai

[47] L. Wang, W. Xu, Y. Lan, Z. Hu, Y. Lan, R. K.-W. Lee, and E.-P. Lim, “Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models,” in Proc. 61st Annu. Meeting Assoc. Comput. Linguistics (ACL), 2023, pp. 2609–2634.