https://doi.org/10.51885/3107-2755_JISC_2026_2_5
SRSTI 81.93.29

ARTIFICIAL INTELLIGENCE IN AUTOMATED PENETRATION TESTING: A CROSS-BRANCH STRUCTURED REVIEW OF LLM-AGENT AND REINFORCEMENT-LEARNING EVIDENCE

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ В АВТОМАТИЗИРОВАННОМ ТЕСТИРОВАНИИ НА ПРОНИКНОВЕНИЕ: МЕЖВЕТВЕВОЙ СТРУКТУРИРОВАННЫЙ ОБЗОР ДАННЫХ ОБ LLM-АГЕНТАХ И ОБУЧЕНИИ С ПОДКРЕПЛЕНИЕМ

ЕНУГЕ АВТОМАТТАНДЫРЫЛҒАН ТЕСТІЛЕУДЕГІ ЖАСАНДЫ ИНТЕЛЛЕКТ: LLM-АГЕНТТЕР МЕН НЫҒАЙТЫП ОҚЫТУ ДЕРЕКТЕРІНІҢ САЛААРАЛЫҚ ҚҰРЫЛЫМДАЛҒАН ШОЛУЫ

R.A. Yessentayev [1*], **Y.Y.Begimbayeva** [2]
[1] Astana IT University, Astana, Kazakhstan
[2] Energo University, Almaty, Kazakhstan
** Corresponding author: Rauan Yessentayev, e-mail: 255316@astanait.edu.kz*

Keywords:
automated penetration testing, large language model agents, reinforcement learning, attack path planning, benchmarking, reproducibility, evidence quality, structured literature review.

ABSTRACT

Research on automated penetration testing has divided into two communities that rarely cite one another: large language model (LLM) agents that orchestrate real security tools, and reinforcement learning (RL) methods that optimise attack paths over formal network models. Recent reviews cover one branch or the other; none synthesises both, so the field lacks a basis for comparing their evidence. This article presents a structured cross-branch review of sixty-two studies published between 2019 and 2026, assembled from a curated seed corpus, audited and extended by documented supplementary retrieval and bidirectional snowballing on three recent reviews. Every primary study is coded against a seven-criterion evidence rubric covering aim, testbed, metrics, baselines, external validation, reproducibility, and safety reporting, and the complete coding, with the basis of each judgement, is printed in the article. Three findings emerge from this corpus. The two branches fail in mirror-image ways: LLM agents act in realistic environments but cannot verify their own state, whereas RL agents reason verifiably over environments that abstract away the conditions making penetration testing difficult. Reported success rates tend to separate into two bands according to who constructed the evaluation, a pattern also reported for a larger independent corpus, although metrics and test conditions differ. External validation is the weakest criterion: in the sample, four of forty-five non-benchmark primary studies test a live target, and three of those four originate from a single research group. Within this corpus, the evidence supports automated penetration testing as decision support rather than validated autonomy.

Түйінді сөздер:
енуге автоматтандырылған тестілеу, үлкен тілдік модельге негізделген агенттер, нығайтып

ТҮЙІНДЕМЕ

Енуге автоматтандырылған тестілеу зерттеулері бір-біріне сирек сілтеме жасайтын екі бағытқа бөлінді: үлкен тілдік модельдерге (LLM) негізделген агенттер және желінің формалды модельдеріндегі нығайтып оқыту (RL) әдістері. Қолда бар шолулар олардың біреуін ғана қамтиды. Мақалада 2019–2026 жылдардағы алпыс екі жұмыстың

оқыту, шабуыл жолын жоспарлау, бенчмаркинг, қайталанымдылық, дәлелдер сапасы, құрылымдалған әдебиет шолуы.

құрылымдалған салааралық шолуы ұсынылған; корпус бастапқы дереккөздерді тексеру, құжатталған қосымша іздеу және қар кесегі әдісі арқылы жинақталды. Әрбір бастапқы зерттеу жеті өлшемді рубрика бойынша кодталды; толық кодтау кестесі пайымдардың негіздерімен жарияланады. Іріктемеде екі бағыт айналы түрде сәтсіздікке ұшырайды: LLM-агенттер шынайы ортада әрекет етеді, бірақ өз күйін тексермейді, ал RL-агенттер міндеттің негізгі шарттарын дерексіздендіретін орталар туралы тексерілетін түрде пайымдайды. Табыстылық көрсеткіштері, әдетте, бағалау ортасын кім құрғанына қарай екі топқа бөлінеді, дегенмен метрикалар мен сынақ жағдайлары әртүрлі; бұл неғұрлым кең тәуелсіз корпуста да байқалған. Бенчмарктерден тыс қырық бес бастапқы жұмыстың төртеуі шынайы нысанда тексерілген, үшеуі бір топқа тиесілі. Корпус деректері енуге автоматтандырылған тестілеуді расталған автономия емес, шешім қабылдауды қолдау құралы ретінде негіздейді.

Ключевые слова:
автоматизированное тестирование на проникновение, агенты на основе больших языковых моделей, обучение с подкреплением, планирование пути атаки, бенчмаркинг, воспроизводимость, качество доказательств, структурированный обзор литературы.

АННОТАЦИЯ

Исследования автоматизированного тестирования на проникновение разделились на два слабо связанных направления: агенты на основе больших языковых моделей (LLM) и методы обучения с подкреплением (RL) на формальных моделях сети. Существующие обзоры охватывают лишь одно из них. Представлен структурированный межветвевой обзор шестидесяти двух работ 2019–2026 годов, собранных аудитом исходного набора, документированным поиском и методом снежного кома. Каждое первичное исследование закодировано по семи критериям; полная таблица кодирования опубликована с основаниями суждений. В выборке направления отказывают зеркально: LLM-агенты действуют в реалистичных средах, но не верифицируют своё состояние, а RL-агенты верифицируемо рассуждают о средах, абстрагирующих ключевые условия задачи. Показатели успешности обычно делятся на две группы в зависимости от того, кто построил оценочную среду, хотя метрики и условия испытаний различаются; то же отмечено в более широком независимом корпусе. Живую цель тестируют четыре из сорока пяти работ вне бенчмарков, три — одной группы. Данные корпуса обосновывают автоматизированное тестирование как поддержку решений, а не проверенную автономию.

INTRODUCTION

Penetration testing is a structured security assessment in which testers identify vulnerabilities, reason about their exploitability, and validate whether individual weaknesses can be chained into genuine security impact. The question of how far this workflow can be automated has produced a large and fast-moving literature, and that literature has organised itself into two branches that developed largely independently of one another.

The first branch places a large language model (LLM) around conventional security tooling as a planner, an interpreter of tool output, or a coordinator of sub-agents. Deng et al. (2024) established structured state tracking, and later systems added direct tool invocation, multi-agent decomposition, retrieval, memory, and curriculum scheduling (Kong et al., 2025; W. G. Li et al., 2026; Nakatani, 2025; Shen et al., 2025; Wu et al., 2025). The second branch formalises attack progress as sequential decision making over network states or attack graphs and applies reinforcement learning (RL) to learn or optimise policies (Ghanem & Chen, 2020; Hu et al., 2020; Schwartz & Kurniawati, 2019; Tran et al., 2022), later adding hierarchical and cascaded

architectures, behaviour diversity, reward machines, and attack-graph structure (Gangupantulu et al., 2021; Ghanem et al., 2023; Y. Li et al., 2024; H. Liu et al., 2024; Y. Yang & Liu, 2022).

The two branches address different halves of the same problem, yet the surveys describing them are branch-local. Peng et al. (2026) systematise the LLM branch with a unified empirical comparison; He et al. (2026) analyse eighty-one papers on penetration-testing agents; J. Liu et al. (2026) review the RL branch; and Chen et al. (2024) survey penetration path planning. Earlier work covers artificial intelligence in penetration testing and vulnerability assessment (McKinnel et al., 2019), network penetration testing (Alhamed & Rahman, 2023), and the mapping of artificial intelligence onto testing activities (Alwabisi, 2025). None covers both branches: He et al. (2026) retain only two non-LLM papers, as baselines, and J. Liu et al. (2026) do not examine LLM agent architectures.

This article is therefore not another inventory of methods. Its contribution is the comparison those reviews cannot make: a cross-branch synthesis showing that the failure modes of the two branches are complementary; an explicit evidence rubric whose complete per-study coding is printed in the article; evidence on how strongly reported results depend on who built the evaluation environment; and a retrieval protocol reported with its date and query families so that the corpus can be regenerated.

The study addresses five research questions: which artificial-intelligence techniques are applied in automated penetration testing, and in what proportions among primary studies (RQ1); how LLMs are used to automate penetration-testing workflows (RQ2); how RL supports attack-path planning and autonomous testing (RQ3); what methodological and technical limitations remain, and how they differ between the branches (RQ4); and what a cross-branch evaluation protocol would have to measure (RQ5).

LITERATURE REVIEW

Automated penetration testing covers the partial or full automation of reconnaissance, vulnerability identification, exploit selection, privilege escalation, lateral movement, and reporting. The task is difficult because testers must reason over uncertain state, changing topology, exploit preconditions, and facts discovered during the engagement, and the two branches diverge on how to handle this.

The RL branch treats the problem as sequential decision making, an approach originally formulated in planning terms (Sarraute et al., 2013), and makes it tractable by committing to a state and action representation supplied by a simulator or an attack graph. The LLM branch instead keeps the environment real and delegates representation to the model's own context, which is flexible but unverifiable.

Within the LLM branch, four workflow roles recur: task decomposition, command or exploit suggestion, tool-output interpretation, and state or report summarisation. Happe and Cito (2023) show that a general-purpose model can act as a penetration-testing sparring partner, with an ethical framing that shaped subsequent work. Deng et al. (2024) formalise interactive reasoning across engagement stages with an explicit task tree; Shen et al. (2025) add online retrieval of vulnerability records; J. Xu et al. (2024) organise attack-chain generation around the MITRE ATT&CK taxonomy; Nakatani (2025) targets longer address-to-shell workflows; and W. G. Li et al. (2026) introduce a rectification module for hallucinated commands with command-specific memory.

Multi-agent decomposition is the dominant architecture among the sampled studies of this branch. Kong et al. (2025) organise collaboration around a task graph with a check-and-reflect loop, Wu et al. (2025) add curriculum-guided scheduling, and other systems explore different coordination topologies (Alshehri et al., 2024; Challita & Parrend, 2026; Geng et al., 2025; Mayoral-Vilches et al., 2025). W. Wang et al. (2026) fuse a dynamic knowledge graph into web testing; Mai

et al. (2025) address long-context forgetting through memory activation; L. Wang et al. (2025) couple LLMs to classical planners; and Luong et al. (2025) show that a fine-tuned open-weight model can compete with frontier models. Muzsai et al. (2024) pair an agent with its own evaluation harness, and Zheng and Zhu (2026) evaluate against an active defender, whereas almost every other evaluation in the corpus assumes that defences are absent.

Within the RL branch, learning is used primarily to select actions over a modelled environment. Schwartz and Kurniawati (2019) formulate autonomous penetration testing as an RL problem, Ghanem and Chen (2020) study efficient network testing under simulation, and capture-the-flag formulations weigh model-free learning against injected prior knowledge (Zennaro & Erdődi, 2020). Deep RL adds neural policies (Hu et al., 2020; S. Zhou et al., 2021), hierarchical approaches decompose large networks (Ghanem et al., 2023; H. Liu et al., 2024), cascaded agents structure large action spaces (Tran et al., 2022), and path-design work targets attack-path selection directly (Yi & Liu, 2023).

Several studies connect learning to explicit security structure: planning under incomplete information (T.-Y. Zhou et al., 2019), reward machines that encode domain knowledge (Y. Li et al., 2024), attack-graph formulations for crown-jewels analysis and exfiltration-path discovery (Cody et al., 2022; Gangupantulu et al., 2021), and a layered reference model relating the two (Cody, 2022). Behaviour-diverse RL seeks a set of distinct successful strategies rather than a single optimum (Y. Yang & Liu, 2022), and a unified modelling framework attempts to reconcile these formulations (Y. Wang et al., 2025).

The credibility of the RL branch rests on its environments: simulators derived from network attack models, emulator hybrids (Janisch et al., 2023), emulated gyms (L. Li et al., 2021), and accelerated implementations (Simon et al., 2026) define the state and action spaces within which results are reported. Work on generalisation studies the simulation-to-real gap through domain randomisation and meta-learning (S. Zhou et al., 2025).

Hybrid approaches respond to the limitations of both branches. Al-Sabbagh et al. (2026) combine semi-automation with recursive memory and real-time retrieval, treating context loss as a measurable error class; Dai et al. (2025) add self-reflection over a knowledge base; Karagiannis et al. (2024) move from simulation toward validation; and Moreno et al. (2025) combine learned decision support with a recommender architecture. De Pasquale et al. (2024) show that the non-learning end of the spectrum remains competitive, discovering privilege-escalation chains on live cloud instances through classical planning.

Benchmarks form the fourth group. Capture-the-flag suites (Shao et al., 2024; Zhang et al., 2025) provide scale and automatic verification; suites grounded in real vulnerabilities (Gioacchini et al., 2024; Zhu et al., 2025) trade scale for realism; stage-level designs (R. Yang et al., 2025) localise failures; and meta-benchmarks (Sanz-Gómez et al., 2025) and real-world suites (Mai et al., 2025) push toward operational conditions, while Happe and Cito (2025) audit testbed, metric, and experiment design. Progress nevertheless remains method-specific: no benchmark in this corpus evaluates a learned policy and an LLM agent on the same task under the same budget.

Broad reviews of LLMs in cybersecurity (Motlagh et al., 2025; H. Xu et al., 2025) and of vulnerability detection and repair (X. Zhou et al., 2025) are retained as background rather than as primary studies, because detecting or repairing a vulnerability is not the same activity as autonomously exploiting one.

Table 1 positions this review against the eight prior reviews identified above; because three competing reviews appeared while this study was being prepared, the difference is stated explicitly.

Table 1. Coverage of this review compared with prior reviews

Review	LLM	RL	Hybrid	Bench.	Repro.	Safety	N(corpus)
McKinnel et al. (2019)	N	P	N	N	P	N	n/r
Alhamed & Rahman (2023)	N	P	N	N	N	P	39
Chen et al. (2024)	N	Y	P	N	N	N	n/r
Alwabisi (2025)	P	P	N	N	N	N	57
H. Xu et al. (2025)	P	N	N	P	N	P	185
Peng et al. (2026)	Y	N	P	Y	Y	P	15
He et al. (2026)	Y	P	P	Y	P	P	81
J. Liu et al. (2026)	N	Y	P	P	N	N	n/r
This review	Y	Y	Y	Y	Y	Y	62

Note: compiled by the authors. LLM = LLM agents, RL = RL and attack-path planning, Bench. = benchmarking, Repro. = reproducibility, Safety = safety reporting; Y = covered, P = partially covered, N = not covered. N(corpus) is the primary-study count as stated by each review; n/r means the review states no single such count in its abstract or accessible full text. McKinnel et al. (2019) report 663 records retrieved and 214 unique after de-duplication, but no final included count.

Two clarifications follow. First, the corpus is smaller than the eighty-one records of He et al. (2026) and larger than the fifteen frameworks of Peng et al. (2026) by design: the study does not enumerate the LLM branch, which is already enumerated, but places a representative sample of it alongside the RL branch under one rubric. Second, the dimensions on which full coverage is claimed here, reproducibility and safety reporting as coded per-study variables, are those that prior reviews report only in aggregate or not at all.

MATERIALS AND METHODS

The material is a corpus of published research on artificial intelligence in automated penetration testing. The methods are those of a structured literature review: documented retrieval and screening, data extraction, a per-study quality rubric, and an explicit rule separating source-level claims from review-level inference.

Corpus construction. The corpus was assembled in two stages whose auditability differs. The first stage began from thirty-five records held in a curated spreadsheet. Their original database exports, hit counts, duplicate-removal log, and upstream exclusions were not preserved and are not reconstructed retrospectively; instead, each record was re-verified on 20 September 2026 against the publisher record, index entry, or repository page for title, authorship, year, venue, and persistent identifier, and re-screened against the eligibility criteria below. All thirty-five were retained, and two entries citing a superseded preprint were updated to the peer-reviewed version.

In the second stage, the seed was extended on the same date through four query families, issued against openly accessible academic search interfaces and publisher and repository pages, and, as the principal mechanism, through backward and forward snowballing on the three most recent reviews (He et al., 2026; J. Liu et al., 2026; Peng et al., 2026). Q1 combined automated or autonomous penetration testing with artificial intelligence, large language model, or agent; Q2 combined penetration or attack path planning with reinforcement learning or attack graph; Q3 combined penetration testing or offensive security with benchmark, evaluation, or testbed, and with large language model or agent; and Q4 combined penetration testing with retrieval-augmented generation, memory, reward machine, or recommender system.

Snowballing proved the higher-yield step. Two of the three reviews enumerate their corpora in full, eighty-one records in He et al. (2026) and fifteen in Peng et al. (2026), so their reference lists form a published frame that any reader can re-screen. Screening them recovered nearly all of the benchmark and simulation-environment literature missing from the seed; the second stage added twenty-seven records, giving a final corpus of sixty-two (Figure 1).

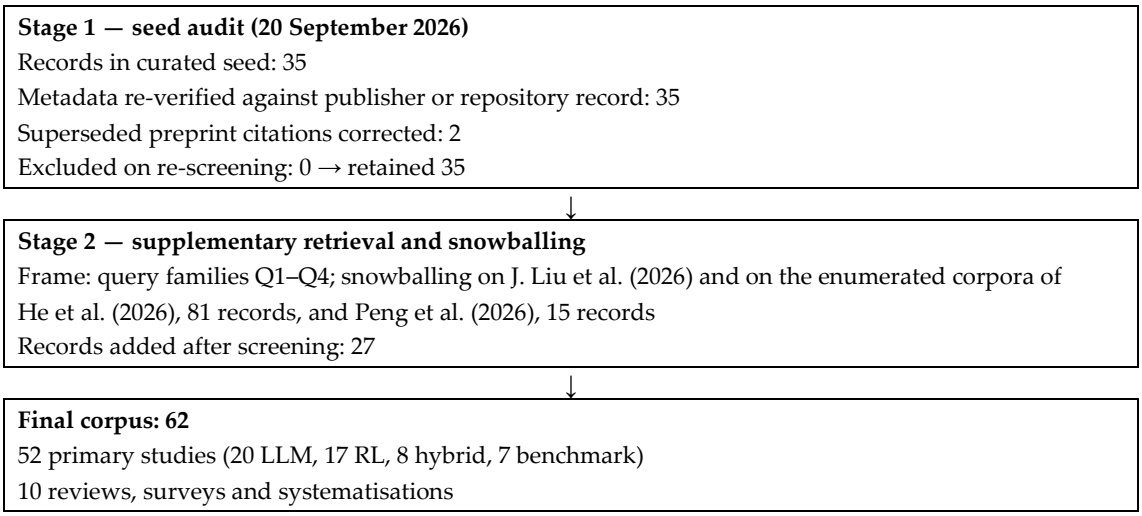


Figure 1. Corpus assembly

Note: compiled by the authors. Only recorded quantities are shown: per-database hit counts and duplicate tallies were not recorded (see Materials and Methods), and provenance is reported per assembly stage.

Eligibility criteria. A study was included if it proposed, evaluated, or benchmarked an artificial-intelligence method for penetration testing or attack-path planning; if it provided a benchmark, testbed, or simulation environment for such methods; or if it was a review directly addressing these topics. Excluded were general cybersecurity work with no automated-penetration-testing component, work with no artificial-intelligence component, opinion and blog sources, and records whose bibliographic metadata could not be verified.

One screening decision is a scope choice rather than an eligibility judgement and accounts for most non-inclusions. Because He et al. (2026) already enumerate the LLM branch, that branch was sampled for architectural and evaluative representativeness, covering each architectural generation, the systems most often used as baselines, and the studies validated outside a laboratory, rather than enumerated. Records not selected under this rule are eligible but not sampled: they are neither coded nor listed in this article, and the per-study coding table in the Results and Discussion section contains only the fifty-two included primary studies. The distinction between ineligible and eligible-but-not-sampled records is therefore stated at the level of the rule, and the records it concerns remain identifiable in the corpus of He et al. (2026).

What this protocol is, and is not. The retrieval is reproducible in that the queries, the date, the snowballing frames, and the screening rules are specified. Provenance is reported per assembly stage (thirty-five seed and twenty-seven second-stage records; Figure 1) rather than per record, and records screened out at the second stage, including eligible-but-not-sampled records, are not listed individually. It is not a database-exhaustive systematic search: no exports from subscription indexes were obtained, so neither per-database hit counts nor a duplicate-removal tally exist. The work is therefore presented as a structured rather than a systematic literature review, and the prevalence figures below are properties of this corpus, not estimates for the field.

Data extraction and classification. For each study the following were extracted: title, identifier, year, venue, authors, method family, evaluation environment, contribution, reported

metrics, reproducibility artefacts, evidence of real-world validation, and limitations or safety discussion. Each study was assigned one dominant analytical role (LLM agent, RL and attack-path planning, hybrid or knowledge-guided, benchmark or environment, or review), with secondary roles recorded qualitatively; studies that span roles, such as Wu et al. (2025) and Al-Sabbagh et al. (2026), are counted once and discussed in several places.

Proportions are computed over primary studies only, so that secondary literature is not counted as if it applied a technique, and percentages are rounded to whole numbers because, at a corpus size near fifty, one reclassification moves a category by about two percentage points.

Quality assessment. Each primary study was coded on seven criteria at three levels (explicit, partial or unclear, and no reported evidence): C1, whether the problem and intended contribution are stated clearly; C2, whether the environment, dataset, network, or attack graph is described; C3, whether success, cost, accuracy, path, or other evaluation metrics are defined; C4, whether a baseline, ablation, or meaningful comparison is reported; C5, whether there is real-world, enterprise, or otherwise realistic validation beyond a narrow simulation; C6, whether code, prompts, configurations, models, or testbed details suffice for replication; and C7, whether limitations, authorisation, unsafe actions, hallucination, or other risks are discussed explicitly. No composite score is computed, because the families report incommensurable artefacts; the criteria qualify the strength and transferability of claims rather than rank papers.

Coding basis, recorded per study. Because coding strength depends on how much of each paper could be read, the coding table records the basis of every row. Basis A (nine studies) means that substantive full text was consulted; basis B (thirty-five) means abstract, bibliographic metadata, and any public artefact link; and basis C (eight) means abstract together with a tabulation of the study in a published survey. Criteria C1 to C6 are codeable on any basis, and the record element behind each code differs by criterion: the stated aim (C1); the named testbed, network, or attack graph (C2); the metrics and comparisons reported in the abstract or, for basis C, in the survey tabulation (C3, C4); the stated evaluation setting (C5); and the artefact actually available at the linked repository (C6). Coding without the full text can miss evidence reported only in the body of a paper, so the likelier error on bases B and C is a downgrade rather than an upgrade, although an abstract that overstates the realism of an evaluation could inflate C5.

The nine studies read in full are a purposive rather than random selection of anchor studies for the main lines of work: the foundational RL formulation (Schwartz & Kurniawati, 2019), a widely used LLM baseline (Deng et al., 2024), the shared benchmark on which several systems report results (Gioacchini et al., 2024), a reference simulation-emulation environment (Janisch et al., 2023), a reference model linking RL and attack graphs (Cody, 2022), an RL method for behaviour diversity (Y. Yang & Liu, 2022), the live Active Directory evaluation (Happe & Cito, 2026), and two of the most recent systems (Al-Sabbagh et al., 2026; W. G. Li et al., 2026). Together they span every method family and the years 2019 to 2026. Reading all fifty-two studies in full was not feasible for a single rater, so the other forty-three were coded on bases B and C. The full-text subset also provides the reference against which abstract-level coding is compared in the Results and Discussion section.

Criterion C7 is different: whether a paper discusses limitations, authorisation, or misuse cannot be established from an abstract, so C7 was coded only on basis A and is marked not assessed for the other forty-three studies rather than coding silence as absence. The C7 row of the aggregate table therefore rests on eight of forty-five studies and supports no quantitative claim.

All coding was performed by a single rater, and no inter-rater statistic is reported, so the rubric is a structured reading of the corpus rather than a measurement instrument of established reliability. What is offered instead is inspectability: every judgement and its basis are printed, so a reader can see what a cell rests on and recode it.

Synthesis rules. A citation supports a source-level statement only when the cited study directly supports it; where a conclusion combines evidence across papers, the text marks it as review-level inference. This matters most in the cross-branch synthesis and in the proposals for future work, which are the authors' own.

Scope of inference. Two levels of conclusion are distinguished. Sample-level findings, namely the composition and temporal profile of the corpus, the proportions in Table 2, and every aggregate derived from the per-study coding, describe the sixty-two analysed records only and are not estimates for the field. A statement about the field is made only where a sample-level pattern is corroborated by a source covering a larger or independent frame, such as the eighty-one-paper corpus of He et al. (2026) or the unified benchmark of Peng et al. (2026), and that source is then cited. Interpretations beyond both, including the mirror-image account of the two branches, are review-level inferences presented as hypotheses for testing rather than as measured properties of the literature.

RESULTS AND DISCUSSION

Corpus composition. Among the fifty-two primary studies in the corpus (Table 2), LLM agent systems form the largest group (20 studies, 38 per cent), followed by RL and attack-path planning (17, 33 per cent), hybrid or knowledge-guided systems (8, 15 per cent), and benchmarks and simulation environments (7, 13 per cent); the ten reviews are excluded from these proportions. This answers RQ1 for the analysed corpus. Because the LLM branch was sampled rather than enumerated, these proportions describe the corpus and are not estimates of the relative size of the two branches in the field.

Table 2. Corpus composition and publication years

Category	n	%	Dominant evidence weakness
LLM agent	20	38	Unverifiable internal state; safety rarely assessable
RL / attack-path planning	17	33	Simulation-bound evaluation
Hybrid / knowledge-guided	8	15	Early-stage validation
Benchmark / environment	7	13	Fragmented, method-specific
Primary subtotal	52	—	
Review / survey	10	—	Branch-local scope
Total corpus	62	—	
By year	—	—	2019: 3 ǀ 2020: 3 ǀ 2021: 3 ǀ 2022: 4 ǀ 2023: 5 ǀ 2024: 11 ǀ 2025: 22 ǀ 2026: 11

Note: compiled by the authors. Percentages are over the 52 primary studies of this corpus and do not sum to 100 because of rounding; — = not applicable. The 2026 figure covers a partial year, to 20 September 2026.

Within the corpus, output is roughly flat at three to five studies per year through 2023 and then rises sharply, peaking at twenty-two in 2025. The RL branch supplies almost all studies before 2023, whereas later growth is dominated by LLM agents and, from 2025, by the benchmarks built to evaluate them. That the benchmarks arrived after the systems they measure plausibly explains, as a review-level inference rather than a measured relation, why so many systems report results on suites they constructed themselves.

Branch findings. Addressing RQ2, the LLM branch has solved execution autonomy and is now bounded by verification. Peng et al. (2026) find hallucinated success indicators in a majority of the open-source frameworks they evaluate under a unified benchmark, with agents mistaking encoded strings for genuine flags and terminating early, and most chained exploitation attempts

stalled at incomplete discovery or failed combination. Because this persisted across backbone models, it reflects current architectures rather than any one model: an agent that can run any command but cannot determine whether it achieved anything is not partially autonomous but unverified. In the analysed sample, the studies of this branch that leave the laboratory all come from one group (Happe & Cito, 2023, 2026; Happe et al., 2026), including a live Active Directory testbed (Happe & Cito, 2026) and a comparison with human testers (Happe et al., 2026).

Addressing RQ3, the RL branch has solved verifiability and is bounded by realism. A policy that provably optimises an objective over a simplified network model tells us about the model, and transfer to operational conditions is usually asserted rather than demonstrated. Within this corpus, work on domain randomisation and meta-learning (S. Zhou et al., 2025) is the clearest attempt to measure transfer. The hybrid studies indicate that hybridisation responds to this structural mismatch, and the evidence supports individual components (memory and retrieval, reward machines, attack graphs, more realistic validation) rather than any validated architecture combining them.

Benchmarking and the benchmark effect. Table 3 groups reported results by who constructed the evaluation environment. The central point concerns the population of results compiled here rather than any single system: systems measured on an externally built, shared benchmark cluster at or below about 25 per cent, whereas systems measured on environments built by their own authors cluster far higher. He et al. (2026) observe the same separation in a larger sample, roughly 15 to 24 per cent for general-purpose systems on a shared benchmark against 74 to 92 per cent on purpose-built suites. The separation is therefore unlikely to be an artefact of the present selection alone, and it is this independent corroboration, rather than the entries of Table 3, that supports reading it as a field-level pattern.

The comparison is nonetheless ordinal rather than quantitative. The figures in Table 3 measure different things (stage scores, task and sub-task completion, success and exploitation rates, a relative gain, counts) under different conditions, including the number and type of targets, human assistance, prior knowledge of the vulnerability, and the model and fine-tuning used. No pooled estimate or ranking is computed; the two-band reading rests on the direction of the difference between shared and self-built evaluations, not on the size of any gap. Even on one benchmark the conditions differ: on AutoPenBench, autonomous agents reach 21 and 24.2 per cent, the human-assisted variant 64 per cent, and xOffense 72.72 per cent after domain fine-tuning (Gioacchini et al., 2024; He et al., 2026; Luong et al., 2025), the one shared-benchmark result outside the lower band, so the bands describe typical reported outcomes rather than a fixed ceiling.

Table 3. Reported results against the environment that produced them

System or benchmark	Environment	Metric	Reported outcome	Source
PentestEval	346 stage-level tasks	Stage score; end-to-end success	Mean stage score 0.41; end-to-end below 6%	R. Yang et al. (2025)
CVE-Bench	40 real web vulnerabilities	Exploit success rate	Best agent about 12.5% (one-day setting)	Zhu et al. (2025)
AutoPenBench	33 containerised tasks	Task success rate	21% autonomous, 64% human-assisted	Gioacchini et al. (2024)
Pentest-R1	AutoPenBench	Task completion	24.2% task completion	He et al. (2026)
xOffense	AutoPenBench	Task completion	72.72% after domain fine-tuning	Luong et al. (2025)

System or benchmark	Environment	Metric	Reported outcome	Source
PentestGPT	13 targets (own)	Sub-task completion (relative gain)	+58.6% sub-task completion vs. direct model use	Deng et al. (2024)
PentestAgent	67 targets (own)	Success rate	74.2% success	Shen et al. (2025)
APT-Agent	7 services (own)	End-to-end exploitation rate	84.29% vs. 48.57% / 18.57% baselines	W. G. Li et al. (2026)
Incalmo	40 scenarios (own)	Environments (count)	37 of 40 environments	He et al. (2026)
Happe & Cito	Live Active Directory	Qualitative outcome	Autonomous account compromise	Happe & Cito (2026)
ChainReactor	Live cloud instances	Chains found (count)	16 privilege-escalation chains rediscovered	De Pasquale et al. (2024)

Note: compiled by the authors from the cited sources. Entries are quoted as reported and are not mutually comparable; that is the point of the table. Metric gives what the reported figure measures; the metrics differ in unit (stages, tasks, sub-tasks, targets, environments) and some figures are relative gains or counts. "Own" marks an evaluation built by the authors of the system evaluated. The Pentest-R1 and Incalmo entries are taken from He et al. (2026); neither system is part of the coded corpus.

A headline success rate therefore carries information about the provenance of the evaluation before it carries information about the system: two systems reporting 25 and 80 per cent cannot be ranked without knowing who built each testbed and what each figure measures, which most of the corpus does not report.

How fragile these numbers are is shown by one case. A recent survey states that a widely cited agent scores 21 per cent on a shared benchmark against 74.2 per cent on its own suite (He et al., 2026), but the 21 per cent figure could not be traced to any primary evaluation of that agent and coincides with the success rate the benchmark reports for its own baseline agent (Gioacchini et al., 2024), so the pairing was excluded from Table 3. This single documented case, not a measured rate, shows how a comparison that is easy to restate and hard to trace can circulate through the secondary literature unchecked.

Two further observations follow. First, difficulty is unevenly distributed within benchmarks, so an aggregate rate partly measures the share of easy tasks. Second, in the corpus, human assistance changes outcomes more than most architectural innovations, and this is the one contrast measured with the environment held constant: 21 per cent for the autonomous agent against 64 per cent for the human-assisted variant on the same thirty-three tasks (Gioacchini et al., 2024). A field claiming progress toward autonomy should treat that forty-three-point gap as a primary metric.

Evidence quality across the branches. Table 4 aggregates the per-study coding for the forty-five non-benchmark primary studies. In this sample both branches rely on controlled settings, for different reasons: LLM studies restrict tool access and task scope to keep engagements tractable and safe, whereas RL studies need formalised state and action spaces before learning is possible.

Table 4. Evidence strength by method family

Rubric dimension	LLM	RL	Hybrid	n assessed
C2 Testbed described	High	High	High	45 / 45
C3 Metrics defined	High	High	High	45 / 45
C4 Baseline or ablation	High	High	Medium	45 / 45
C5 External validation	Low	Low	Low	45 / 45
C6 Reproducibility artefacts	Medium	Medium	Medium	45 / 45
C7 Limitations and safety	High	Medium	—	8 / 45

Note: compiled by the authors from the per-study coding given in full in Table 5. High = Y (explicit evidence) in more than half of the studies assessed, Low = N (no reported evidence) in more than half, Medium = partial or mixed evidence, with neither Y nor N in more than half; — = fewer than two studies assessed. C1 is omitted because every primary study is coded Y on it. Counts exclude the seven benchmark papers, for which several criteria do not apply. C7 is codeable only from full text and was assessed for 3 LLM, 4 RL and 1 hybrid study; those labels are too thinly sampled to compare.

Robustness to the coding basis. Because thirty-seven of the forty-five non-benchmark studies were coded without the full text, the aggregates were recomputed separately for the eight basis-A studies and the thirty-seven basis-B/C studies. C2, C3, and C4 are coded Y for 88 per cent of basis-A studies and for 89, 81, and 70 per cent of basis-B/C studies, so abstract-level coding does not inflate these criteria. Explicit external validation is rare on both bases (C5 = Y for one of eight and three of thirty-seven), although full-text reading moves some studies from no to partial evidence (three of eight coded P, against six of thirty-seven); abstract coding therefore probably overstates the share of studies with no realistic validation, but not the rarity of explicit validation. Explicit reproducibility evidence (C6 = Y) is a minority on both bases (two of eight; seven of thirty-seven). Under the legend of Table 4 the labels for C2 to C4 and C6 are identical on both bases; only C5 would move from Low to Medium in the eight-study full-text subset. Because that subset is small and purposive rather than random, the comparison is indicative rather than conclusive.

Three tensions follow from this coding and together answer RQ4. First, capability and verifiability do not advance together: longer agent workflows broaden task coverage, while reliability work shows that plausible actions may be wrong or ungrounded (W. G. Li et al., 2026; Peng et al., 2026). Second, formal optimisation trades against environmental realism: RL offers well-defined states, actions, and objectives, but its policies inherit the simplifications of the network model (Ghanem et al., 2023; S. Zhou et al., 2025), so the branch that can prove things about its agent proves them only about a simplified world.

Third, and most importantly, the branches fail in mirror image, a review-level inference from the coded sample rather than a measured property of the field, and one that branch-local reviews cannot make. The LLM branch operates in real environments it cannot reason about verifiably; the RL branch reasons verifiably about environments that are not real. Each branch's principal weakness is the other's principal strength, which explains why hybridisation keeps reappearing and why neither branch's benchmarks can diagnose the other's failure mode. Because the branches are evaluated in different environments and with different metrics (goal attainment, reward, or steps in simulation for RL; task or sub-task completion on real tools for LLM agents), the comparison rests on the rubric criteria and on the properties summarised in Figure 2, not on reported performance figures.

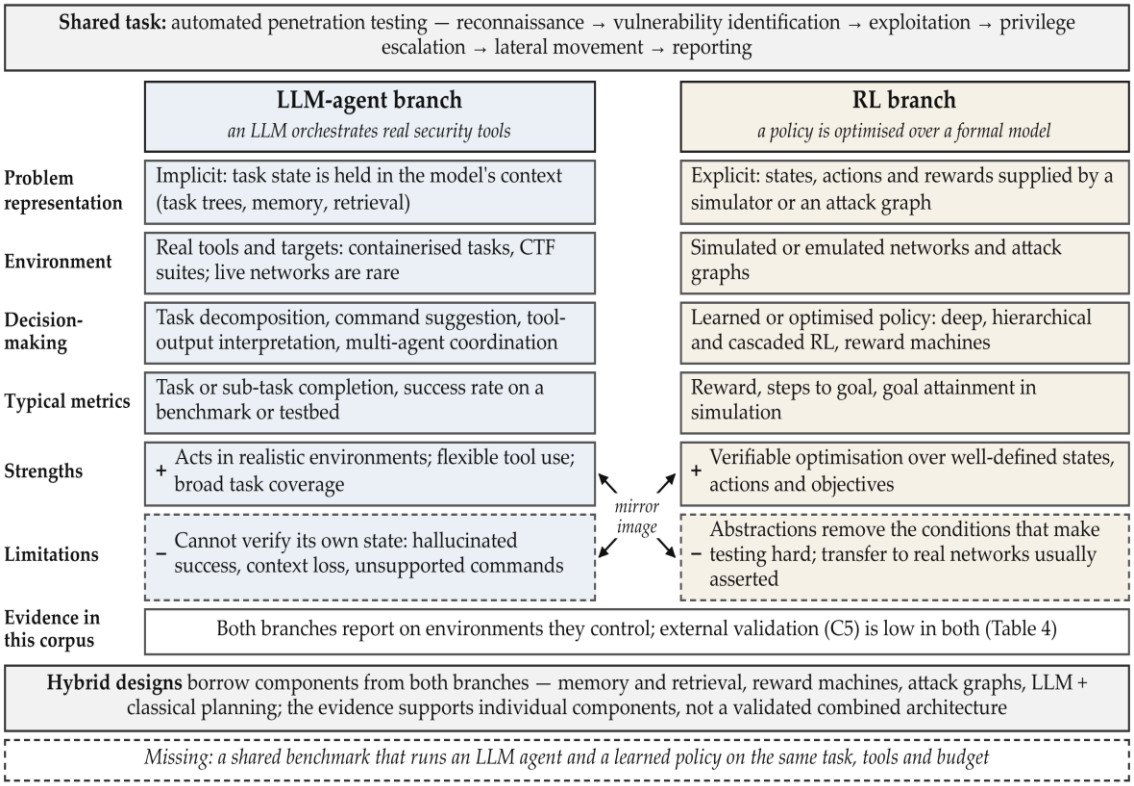


Figure 2. Comparative scheme of the LLM-agent and RL branches

Note: compiled by the authors from the reviewed studies; the evidence row summarises Table 4.

The mirror-image synthesis also resolves an apparent contradiction: papers can report improved task completion and still provide weak evidence for deployable autonomy, because a system may improve inside a controlled testbed while external validity, reproducibility, and safety remain unresolved.

External validation is the criterion on which the corpus performs worst, and the concentration is more striking than the count. Of the forty-five non-benchmark primary studies in this corpus, four report evaluation against a live or enterprise target; three come from a single research group (Happe & Cito, 2023, 2026; Happe et al., 2026), and the fourth is a classical planner rather than a learned agent (De Pasquale et al., 2024). Within this corpus, nearly all evidence on how these systems behave outside a laboratory therefore rests on one group's work and one non-learning system. Whether the same concentration holds for the field cannot be established from a sampled corpus and is offered as a hypothesis for an exhaustive enumeration.

Safety and authorisation reporting cannot be judged from an abstract, and only nine studies, eight of them outside the benchmark group, were read in full. Within that sample the pattern follows the branch division. The LLM and hybrid studies read in full discuss limitations and, in three cases, misuse or authorisation explicitly (Al-Sabbagh et al., 2026; Deng et al., 2024; Happe & Cito, 2026), whereas none of the four RL studies read in full discusses authorisation, responsible use, or misuse, and one has no limitations section (Cody, 2022; Janisch et al., 2023; Schwartz & Kurniawati, 2019; Y. Yang & Liu, 2022). No generalisation is drawn from four studies, but the asymmetry is what the mirror-image thesis predicts: agents that only ever act on a simulator never force the question of what they are authorised to do.

The problem is methodological before it is ethical: an agent that hallucinates success produces, in an authorised engagement, a report asserting a compromise that did not occur, and in an unconstrained one acts outside the agreed perimeter without any record that it did so.

Table 5 gives the complete per-study coding on which Table 4 and the observations above rest, reproduced in full so that every aggregate can be recomputed and any cell recoded. It is restricted to the fifty-two included primary studies: as stated in the Materials and Methods section, provenance is reported per assembly stage (Figure 1), and eligible-but-not-sampled records are not listed.

Table 5. Per-study quality coding, all 52 primary studies

Study	Fam.	Bas.	C1–C7	Study	Fam.	Bas.	C1–C7
hackingBuddyGPT (2023)	LLM	B	YPPPYP—	Improved DQN (2021)	RL	B	YYYYNP—
BreachSeek (2024)	LLM	C	YPPPNP—	Layered ref. model (2022)	RL	A	YPPPNNP
PentestGPT (2024)	LLM	A	YYYYPYY	Cascaded RL (2022)	RL	B	YYYYNP—
HackSynth (2024)	LLM	B	YYYYNY—	Behaviour-diverse (2022)	RL	A	YYYYNPN
AutoAttacker (2024)	LLM	C	YYYYNP—	Hierarchical RL (2023)	RL	B	YYYYPP—
LLM+PDDL planner (2025)	LLM	B	YYYYNP—	NASimEmu (2023)	RL	A	YYYYPYP
Controller (2025)	LLM	C	YYYYNP—	Yi & Liu (2023)	RL	B	YYYYNP—
VulnBot (2025)	LLM	B	YYYYNY—	H. Liu et al. (2024)	RL	B	YYYYNP—
xOffense (2025)	LLM	C	YYYYNY—	Unified framework (2025)	RL	B	YYPPNP—
TermiAgent (2025)	LLM	B	YYYYPY—	Mind the gap (2025)	RL	B	YYYYPP—
CAI (2025)	LLM	B	YPPPPY—	NASimJax (2026)	RL	B	YYYYNY—
RapidPen (2025)	LLM	B	YYYPNP—	Crown jewels (2021)	Hyb.	B	YYYPNP—
PentestAgent (2025)	LLM	B	YYYYNP—	Exfiltration paths (2022)	Hyb.	B	YYYPNP—
CurriculumPT (2025)	LLM	B	YYYYNP—	Shennina (2024)	Hyb.	B	YYPPPP—
RedTeamLLM (2026)	LLM	C	YPPPNP—	Reward machine (2024)	Hyb.	B	YYYYNP—
PTFusion (2026)	LLM	C	YYYYNP—	ChainReactor (2024)	Hyb.	B	YYYYYY—
AD assumed breach (2026)	LLM	A	YYYYYPY	RefPentester (2025)	Hyb.	C	YYYYNP—
LLMs as hackers (2026)	LLM	B	YYYYYP—	RL + recommender (2025)	Hyb.	B	YYYPNP—
APT-Agent (2026)	LLM	A	YYYYNPP	AutoSecAgent (2026)	Hyb.	A	YYYYPNY
ZERO-APT (2026)	LLM	B	YYYYNP—	AutoPenBench (2024)	Bench.	A	YYYYPYY
Schwartz & Kurniawati (2019)	RL	A	YYYYNPP	NYU CTF Bench (2024)	Bench.	B	YYYYNY—
NIG-AP (2019)	RL	B	YYYYNP—	Benchmarking audit (2025)	Bench.	B	YYYPPY—
Ghanem & Chen (2020)	RL	B	YYYYNP—	CAIBench (2025)	Bench.	B	YYYYNY—
Hu et al. (2020)	RL	B	YYYYNP—	PentestEval (2025)	Bench.	B	YYYYNP—
Zennaro & Erdődi (2020)	RL	B	YYYYNP—	Cybench (2025)	Bench.	B	YYYYNY—
CyGIL (2021)	RL	C	YYPPPN—	CVE-Bench (2025)	Bench.	B	YYYYPY—

Note: compiled by the authors. Fam. = method family: LLM = LLM agent, RL = RL and attack-path planning, Hyb. = hybrid or knowledge-guided, Bench. = benchmark or environment. Bas. = basis of the coding: A = full text consulted, B = abstract, metadata and artefact links, C = abstract plus a tabulation in a published survey. The C1–C7 column gives the seven codes in order, one character each: Y = explicit evidence, P = partial or unclear, N = none reported, — = not assessed. C7 is codeable only on basis A and is shown as not assessed elsewhere. Abbreviations in study labels: AD = Active Directory, CTF = capture the flag, DQN = deep Q-network, PDDL = Planning Domain Definition Language. The table lists included primary studies only; it reads down the left half and continues in the right half.

Threats to validity. Regarding selection bias, the corpus originates from a curated seed whose retrieval log was not preserved; the dated, documented second stage mitigates this, but the seed's original exclusions cannot be recovered and no exhaustive coverage of either branch is

claimed (for exhaustive enumeration of the LLM branch, see He et al., 2026). Table 6 sets out each limitation of the selection procedure with its likely effect on the findings. Regarding publication bias, twenty-four of the fifty-two primary studies are preprints and positive prototype results are more visible than negative ones; the reliability literature (Happe & Cito, 2025; Peng et al., 2026) partly counteracts this and is deliberately over-weighted in the synthesis. Regarding coding validity, the coding was single-rater, full text was read only for a purposive subset of nine studies, and the basis varies per study, but Table 5 makes every judgement inspectable. Regarding temporal validity, preprints, models, and benchmarks change after publication; metadata were verified against a persistent identifier, repository page, or publisher record on the stated date; records were updated to a peer-reviewed version where one was identified, and bibliography entries were completed where the cited record omitted authors or publication details.

Table 6. Limitations of the selection procedure and their possible influence on the findings

Limitation	Possible influence on the findings
Curated seed (35 of 62 records) without a preserved retrieval log	Seed selection cannot be audited and may favour prominent work; per-study codes are unaffected, but the proportions and temporal profile in Table 2 are not generalisable.
LLM branch sampled (20 primary studies) rather than enumerated	The LLM share in Table 2 understates the branch. The rule retained studies validated outside a laboratory, so external validation is, if anything, overstated and the finding that it is rare is conservative.
Snowballing frames dominated by LLM-oriented reviews (He et al., 2026; Peng et al., 2026)	All seven benchmark papers evaluate LLM agents, so the benchmark effect (Table 3) concerns LLM evaluation only.
No exports from subscription indexes such as Scopus or Web of Science	Papers neither openly indexed nor preprinted may be missing, more likely in the RL branch, which is less often preprinted here (7 of 17 studies, against 11 of 20 LLM studies).
Preprints (24 of 52 primary studies); positive results more visible	Capabilities and success rates are likely biased upward, which strengthens the conclusion that end-to-end autonomy is not validated; preprint versions may also change.
English-language queries and frames only	Russian- and Kazakh-language work, including regional studies, is excluded, with an effect of unknown direction; the conclusions apply to the English-language literature, and regional databases are a next step.
Single cut-off date (20 September 2026)	Later work, including new live-target evaluations, is not reflected; the external-validation picture (C5) is the most likely to change.
Commercial platforms without published evaluations	Only research prototypes are covered; whether they surpass, match, or fall short of deployed products cannot be established.
Note: compiled by the authors.	

CONCLUSION

This structured review examined sixty-two studies on artificial intelligence in automated penetration testing across the LLM and RL branches that prior reviews treat separately. Within this corpus, the strongest pattern is that such testing is better supported as decision support and workflow automation than as validated end-to-end autonomy. The objectives were met: the corpus was classified and quantified over primary studies (RQ1); the roles of LLMs (RQ2) and of RL (RQ3) were characterised; the limitations were derived from an explicit per-study coding (RQ4); and the requirements for a cross-branch evaluation protocol were specified (RQ5).

The scientific novelty lies in the cross-branch comparison and its central interpretive claim, derived from the coded sample: the branches fail in mirror image. LLM agents act in real environments they cannot reason about verifiably, and RL agents reason verifiably about environments that are not real, so neither branch's benchmarks can diagnose the other's failure mode, which explains the repeated rediscovery of hybrid designs and the absence of any shared evaluation. A second, methodological contribution is a per-study evidence coding with the basis of each judgement recorded, which allows the aggregate claims to be checked rather than accepted.

The practical significance is threefold. For practitioners, a reported success rate carries information about who built the evaluation before it carries information about the system, a pattern observed in this corpus and corroborated in a larger one (He et al., 2026), although the figures differ in metric and test conditions; procurement and assessment should therefore require the provenance of any quoted benchmark figure. For researchers, external validation is the weakest link in the analysed evidence base: in the sample, four of the forty-five non-benchmark primary studies test a live target, and three of those four come from one group. For evaluation designers, the human-assistance gap measured with the environment held constant is a more informative indicator of progress toward autonomy than any aggregate success rate.

The conclusions are bounded and fall into two groups. Sample-level findings, namely the composition of the corpus, the per-criterion coding, the concentration of external validation in one group, and the asymmetry in safety reporting, describe the sixty-two analysed records only; their sensitivity to the coding basis is examined in the Results and Discussion section, and the likely effect of each selection limitation is set out in Table 6. Field-level conclusions are limited to two patterns corroborated beyond the sample: the dependence of reported success rates on who built the evaluation (He et al., 2026), which holds as a direction of difference rather than a quantified effect because the figures differ in metric and test conditions, and the verification failures of the LLM branch under a unified benchmark (Peng et al., 2026). The mirror-image account is a review-level interpretation offered for testing. The review does not establish that one model family is superior or support exhaustive prevalence claims, and because the coding was single-rater and the safety criterion rests on eight full-text readings outside the benchmark group, the safety asymmetry is a hypothesis for confirmation rather than an established proportion.

Five directions for further research follow; they are the authors' proposals rather than claims of the cited work. First, cross-branch evaluation: a shared task set on which a learned policy and an LLM agent run under identical environment, tool, and budget constraints, combining stage-level decomposition (R. Yang et al., 2025) and milestone-based partial credit (Gioacchini et al., 2024) with an environment expressible both as a formal state space and as a live target. Second, verification as a system component: hallucination and state loss should be measured, and agents should report a verification rate alongside a success rate. Third, environment realism: RL work should test simulation-to-real transfer and report environment assumptions, action-space construction, and reward design. Fourth, hybrid architectures assessed componentwise rather than assumed superior. Fifth, safety inside the evaluation protocol: authorisation assumptions, constrained testbeds, action logging, and an unsafe-action rate reported as a standard metric alongside capability.

CONFLICT OF INTEREST: The authors declare no conflict of interest.

FUNDING: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

INSTITUTIONAL REVIEW BOARD STATEMENT: Not applicable. This study reviews previously published literature; it involved no human participants or animals, and no penetration testing was performed by the authors.

INFORMED CONSENT STATEMENT: Not applicable. The study did not involve human participants.

DATA AVAILABILITY STATEMENT: All data underlying this review are contained in the article. Table 5 gives the complete per-study coding of all 52 primary studies, Figure 1 the stage-level provenance, and the Materials and Methods section the retrieval protocol, date, and query families. No new experimental data were generated.

ACKNOWLEDGEMENTS: The authors thank colleagues for methodological discussions, and the anonymous reviewers for comments that improved the manuscript.

STATEMENT ON THE USE OF ARTIFICIAL INTELLIGENCE TECHNOLOGIES: During the preparation and revision of this manuscript, the authors used the generative artificial-intelligence tool Claude Opus 5 (Anthropic) for editorial support (Claude Opus 5, 2026), namely language editing and checking the consistency of terminology and abbreviations. The selection, screening, and coding of the studies, the analysis, and the conclusions are the authors' own. The authors reviewed and edited all output of the tool and take full responsibility for the content of the publication.

REFERENCES

Alhamed, M., & Rahman, M. M. H. (2023). A systematic literature review on penetration testing in networks: Future research directions. *Applied Sciences, 13*(12), Article 6986. https://doi.org/10.3390/app13126986

Al-Sabbagh, A., Abdo, E., Samrouth, K., Alkhatib, A. A. A., & Elkhodr, M. (2026). AutoSecAgent: A semi-automated AI-driven penetration testing framework through recursive memory and real-time RAG. *The Journal of Supercomputing, 82*(5), Article 318. https://doi.org/10.1007/s11227-026-08439-z

Alshehri, I., Alshehri, A., Almalki, A., Bamardouf, M., & Akbar, A. (2024). *BreachSeek: A multi-agent automated penetration tester* [Preprint]. arXiv. https://arxiv.org/abs/2409.03789

Alwabisi, S. O. (2025). *AI in penetration testing: A systematic mapping study* [Preprint]. TechRxiv. https://doi.org/10.36227/techrxiv.175099664.46246512/v1

Challita, B., & Parrend, P. (2026). RedTeamLLM: An agentic AI framework for offensive security. In E. Mercier-Laurent, G. Kayakutlu, I. Chomiak-Orsa, L. Przysucha, & D. Verdejo (Eds.), *Artificial intelligence for sustainability and global security* (IFIP Advances in Information and Communication Technology, Vol. 780, pp. 337–354). Springer. https://doi.org/10.1007/978-3-032-21184-2_26

Chen, Z., Kang, F., Xiong, X., & Shu, H. (2024). A survey on penetration path planning in automated penetration testing. *Applied Sciences, 14*(18), Article 8355. https://doi.org/10.3390/app14188355

Claude Opus 5. (2026, September 20). Editorial support for a literature-review manuscript [Large language model]. Anthropic. https://claude.ai

Cody, T. (2022). *A layered reference model for penetration testing with reinforcement learning and attack graphs* [Preprint]. arXiv. https://arxiv.org/abs/2206.06934

Cody, T., Rahman, A., Redino, C., Huang, L., Clark, R., Kakkar, A., Kushwaha, D., Park, P., Beling, P., & Bowen, E. (2022). *Discovering exfiltration paths using reinforcement learning with attack graphs* [Preprint]. arXiv. https://arxiv.org/abs/2201.12416

Dai, H., Li, Y., Yan, J., & Zhang, Z. (2025). RefPentester: A knowledge-informed self-reflective penetration testing framework based on large language models. In *Proceedings of the 22nd Annual International Conference on Privacy, Security and Trust* (pp. 1–8). IEEE. https://arxiv.org/abs/2505.07089

Deng, G., Liu, Y., Mayoral-Vilches, V., Liu, P., Li, Y., Xu, Y., Zhang, T., Liu, Y., Pinzger, M., & Rass, S. (2024). PentestGPT: Evaluating and harnessing large language models for

automated penetration testing. In *Proceedings of the 33rd USENIX Security Symposium* (pp. 847–864). USENIX Association.

De Pasquale, G., Grishchenko, I., Iesari, R., Pizarro, G., Cavallaro, L., Kruegel, C., & Vigna, G. (2024). ChainReactor: Automated privilege escalation chain discovery via AI planning. In *Proceedings of the 33rd USENIX Security Symposium* (pp. 5913–5929). USENIX Association.

Gangupantulu, R., Cody, T., Rahman, A., Redino, C., Clark, R., & Park, P. (2021). Crown jewels analysis using reinforcement learning with attack graphs. In *Proceedings of the IEEE Symposium Series on Computational Intelligence* (pp. 1–6). IEEE. https://arxiv.org/abs/2108.09358

Geng, X., An, N., Xu, B., Yang, X., Jiang, B., Liu, B., & Liu, J. (2025). Controller makes pentesting better: An improved multi-agent automated penetration testing framework. In *Proceedings of the IEEE 24th International Conference on Trust, Security and Privacy in Computing and Communications* (pp. 845–852). IEEE. https://doi.org/10.1109/TrustCom66490.2025.00098

Ghanem, M. C., & Chen, T. M. (2020). Reinforcement learning for efficient network penetration testing. *Information, 11*(1), Article 6. https://doi.org/10.3390/info11010006

Ghanem, M. C., Chen, T. M., & Nepomuceno, E. G. (2023). Hierarchical reinforcement learning for efficient and effective automated penetration testing of large networks. *Journal of Intelligent Information Systems, 60*(2), 281–303. https://doi.org/10.1007/s10844-022-00738-0

Gioacchini, L., Mellia, M., Drago, I., Delsanto, A., Siracusano, G., & Bifulco, R. (2024). *AutoPenBench: Benchmarking generative agents for penetration testing* [Preprint]. arXiv. https://arxiv.org/abs/2410.03225

Happe, A., & Cito, J. (2023). Getting pwn'd by AI: Penetration testing with large language models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 2082–2086). https://doi.org/10.1145/3611643.3613083

Happe, A., & Cito, J. (2025). *Benchmarking practices in LLM-driven offensive security: Testbeds, metrics, and experiment design* [Preprint]. arXiv. https://arxiv.org/abs/2504.10112

Happe, A., & Cito, J. (2026). Can LLMs hack enterprise networks? Autonomous assumed breach penetration-testing Active Directory networks. *ACM Transactions on Software Engineering and Methodology, 35*(6), 1–54. https://doi.org/10.1145/3766895

Happe, A., Kaplan, A., & Cito, J. (2026). LLMs as hackers: Autonomous Linux privilege escalation attacks. *Empirical Software Engineering, 31*, Article 70. https://doi.org/10.1007/s10664-025-10758-3

He, Z., Dong, J., Li, Z., Chen, T., Deng, G., Luo, F., Ji, J., Cao, Y., & Luo, X. (2026). *A survey of LLM-driven penetration testing: Taxonomy, co-evolution, and open challenges* [Preprint]. arXiv. https://arxiv.org/abs/2607.02605

Hu, Z., Beuran, R., & Tan, Y. (2020). Automated penetration testing using deep reinforcement learning. In *Proceedings of the IEEE European Symposium on Security and Privacy Workshops* (pp. 2–10). IEEE. https://doi.org/10.1109/EuroSPW51379.2020.00010

Janisch, J., Pevný, T., & Lisý, V. (2023). *NASimEmu: Network attack simulator & emulator for training agents generalizing to novel scenarios* [Preprint]. arXiv. https://arxiv.org/abs/2305.17246

Karagiannis, S., Fusco, C., Agathos, L., Mallouli, W., Casola, V., Ntantogian, C., & Magkos, E. (2024). AI-powered penetration testing using Shennina: From simulation to validation. In *Proceedings of the 19th International Conference on Availability, Reliability and Security* (pp. 1–7). https://doi.org/10.1145/3664476.3670452

Kong, H., Hu, D., Ge, J., Li, L., Li, T., & Wu, B. (2025). *VulnBot: Autonomous penetration testing for a multi-agent collaborative framework* [Preprint]. arXiv. https://arxiv.org/abs/2501.13411

Li, L., Fayad, R., & Taylor, A. (2021). CyGIL: A cyber gym for training autonomous agents over emulated network systems. In *Proceedings of the 1st International Workshop on Adaptive Cyber Defense*. https://arxiv.org/abs/2109.03331

Li, W. G., Abuadbba, A., Moore, K., & Kim, D. D. (2026). *APT-Agent: Automated penetration testing using large language models* [Preprint]. arXiv. https://arxiv.org/abs/2605.24949

Li, Y., Dai, H., & Yan, J. (2024). *Knowledge-informed auto-penetration testing based on reinforcement learning with reward machine* [Preprint]. arXiv. https://arxiv.org/abs/2405.15908

Liu, H., Liu, C., Wu, X., Qu, Y., & Liu, H. (2024). An automated penetration testing framework based on hierarchical reinforcement learning. *Electronics, 13*(21), Article 4311. https://doi.org/10.3390/electronics13214311

Liu, J., Zhang, Y., Zhou, S., Yang, J., Lu, Y., & Zhong, X. (2026). Autonomous penetration testing using reinforcement learning: A review and perspectives. *Expert Systems with Applications, 300*, Article 130219. https://doi.org/10.1016/j.eswa.2025.130219

Luong, P. D., Bao, L. T. G., Tam, N. V. K., Khoa, D. H. N., Quyen, N. H., Pham, V.-H., & Duy, P. T. (2025). *xOffense: An AI-driven autonomous penetration testing framework with offensive knowledge-enhanced LLMs and multi-agent systems* [Preprint]. arXiv. https://arxiv.org/abs/2509.13021v1

Mai, W., Hong, G., Chen, J., Dai, J., Liu, Q., Pan, X., & Zhang, Y. (2025). *Shell or nothing: Real-world benchmarks and memory-activated agents for automated penetration testing* [Preprint]. arXiv. https://arxiv.org/abs/2509.09207

Mayoral-Vilches, V., Navarrete-Lozano, L. J., Sanz-Gómez, M., Espejo, L. S., Crespo-Álvarez, M., Oca-Gonzalez, F., Balassone, F., Glera-Picón, A., Ayucar-Carbajo, U., & Ruiz-Alcalde, J. A. (2025). *CAI: An open, bug bounty-ready cybersecurity AI* [Preprint]. arXiv. https://arxiv.org/abs/2504.06017

McKinnel, D. R., Dargahi, T., Dehghantanha, A., & Choo, K.-K. R. (2019). A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment. *Computers & Electrical Engineering, 75*, 175–188. https://doi.org/10.1016/j.compeleceng.2019.02.022

Moreno, A. C., Hernandez-Suarez, A., Sanchez-Perez, G., Toscano-Medina, L. K., Perez-Meana, H., Portillo-Portillo, J., Olivares-Mercado, J., & García Villalba, L. J. (2025). Analysis of autonomous penetration testing through reinforcement learning and recommender systems. *Sensors, 25*(1), Article 211. https://doi.org/10.3390/s25010211

Motlagh, F., Hajizadeh, M., Majd, M., Najafi, P., Cheng, F., & Meinel, C. (2025). Large language models in cybersecurity: State-of-the-art. In *Proceedings of the 11th International Conference on Information Systems Security and Privacy* (pp. 98–110). https://doi.org/10.5220/0013377600003899

Muzsai, L., Imolai, D., & Lukács, A. (2024). *HackSynth: LLM agent and evaluation framework for autonomous penetration testing* [Preprint]. arXiv. https://arxiv.org/abs/2412.01778

Nakatani, S. (2025). *RapidPen: Fully automated IP-to-shell penetration testing with LLM-based agents* [Preprint]. arXiv. https://arxiv.org/abs/2502.16730

Peng, J., Li, Z., You, C., Wang, Y., Sun, H., Tian, X., Zhang, S., Liu, J., Zhao, J., Liu, R., Ou, H., Sun, Y., Zhang, J., Jiao, Y., Song, K., Zhang, C., Shi, F., Sun, H., Yan, R., & Huang, C. (2026). *Hackers or hallucinators? A comprehensive analysis of LLM-based automated penetration testing* [Preprint]. arXiv. https://arxiv.org/abs/2604.05719

Sanz-Gómez, M., Mayoral-Vilches, V., Balassone, F., Navarrete-Lozano, L. J., Veas Chavez, C. R. J., & del Mundo de Torres, M. (2025). *Cybersecurity AI Benchmark (CAIBench): A meta-benchmark for evaluating cybersecurity AI agents* [Preprint]. arXiv. https://arxiv.org/abs/2510.24317

Sarraute, C., Buffet, O., & Hoffmann, J. (2013). *Penetration testing == POMDP solving?* [Preprint]. arXiv. https://arxiv.org/abs/1306.4714

Schwartz, J., & Kurniawati, H. (2019). *Autonomous penetration testing using reinforcement learning* [Preprint]. arXiv. https://arxiv.org/abs/1905.05965

Shao, M., Jancheska, S., Udeshi, M., Dolan-Gavitt, B., Xi, H., Milner, K., Chen, B., Yin, M., Garg, S., Krishnamurthy, P., Khorrami, F., Karri, R., & Shafique, M. (2024). NYU CTF Bench: A scalable open-source benchmark dataset for evaluating LLMs in offensive security. In *Advances in Neural Information Processing Systems* (Vol. 37, pp. 57472–57498). https://doi.org/10.52202/079017-1832

Shen, X., Wang, L., Li, Z., Chen, Y., Zhao, W., Sun, D., Wang, J., & Ruan, W. (2025). PentestAgent: Incorporating LLM agents to automated penetration testing. In *Proceedings of the 20th ACM Asia Conference on Computer and Communications Security* (pp. 375–391). https://doi.org/10.1145/3708821.3733882

Simon, R., Carrasquel, J., Mees, W., & Libin, P. (2026). *NASimJax: A GPU-accelerated policy learning framework for penetration testing* [Preprint]. arXiv. https://arxiv.org/abs/2603.19864

Tran, K., Standen, M., Kim, J., Bowman, D., Richer, T., Akella, A., & Lin, C.-T. (2022). Cascaded reinforcement learning agents for large action spaces in autonomous penetration testing. *Applied Sciences, 12*(21), Article 11265. https://doi.org/10.3390/app122111265

Wang, L., Shi, X., Li, Z., Jiang, Y., Tan, S., Jiang, Y., Cheng, J., Chen, W., Shen, X., Li, Z., & Chen, Y. (2025). *Automated penetration testing with LLM agents and classical planning* [Preprint]. arXiv. https://arxiv.org/abs/2512.11143

Wang, W., Gu, H., Wu, Z., Chen, H., Chen, X., & Shi, F. (2026). PTFusion: LLM-driven context-aware knowledge fusion for web penetration testing. *Information Fusion, 127*, Article 103731. https://doi.org/10.1016/j.inffus.2025.103731

Wang, Y., Liu, S., Wang, W., Zhou, C., Zhang, C., Jin, J., & Zhu, C. (2025). *A unified modeling framework for automated penetration testing* [Preprint]. arXiv. https://arxiv.org/abs/2502.11588

Wu, X., Tian, Y., Chen, Y., Ye, P., Cui, X., Jia, J., Li, S., Liu, J., & Niu, W. (2025). CurriculumPT: LLM-based multi-agent autonomous penetration testing with curriculum-guided task scheduling. *Applied Sciences, 15*(16), Article 9096. https://doi.org/10.3390/app15169096

Xu, H., Wang, S., Li, N., Wang, K., Zhao, Y., Chen, K., Yu, T., Liu, Y., & Wang, H. (2025). Large language models for cyber security: A systematic literature review. *ACM Transactions on Software Engineering and Methodology*. https://doi.org/10.1145/3769676

Xu, J., Stokes, J. W., McDonald, G., Bai, X., Marshall, D., Wang, S., Swaminathan, A., & Li, Z. (2024). *AutoAttacker: A large language model guided system to implement automatic cyber-attacks* [Preprint]. arXiv. https://arxiv.org/abs/2403.01038

Yang, R., Cheng, M., Deng, G., Zhang, T., Wang, J., & Xie, X. (2025). *PentestEval: Benchmarking LLM-based penetration testing with modular and stage-level design* [Preprint]. arXiv. https://arxiv.org/abs/2512.14233

Yang, Y., & Liu, X. (2022). *Behaviour-diverse automatic penetration testing: A curiosity-driven multi-objective deep reinforcement learning approach* [Preprint]. arXiv. https://arxiv.org/abs/2202.10630

Yi, J., & Liu, X. (2023). Deep reinforcement learning for intelligent penetration testing path design. *Applied Sciences, 13*(16), Article 9467. https://doi.org/10.3390/app13169467

Zennaro, F. M., & Erdődi, L. (2020). *Modeling penetration testing with reinforcement learning using capture-the-flag challenges and tabular Q-learning* [Preprint]. arXiv. https://arxiv.org/abs/2005.12632

Zhang, A. K., Perry, N., Dulepet, R., Ji, J., Menders, C., Lin, J. W., Jones, E., Hussein, G., Liu, S., Jasper, D., Peetathawatchai, P., Glenn, A., Sivashankar, V., Zamoshchin, D., Glikbarg, L., Askaryar, D., Yang, M., Zhang, T., Alluri, R., … Liang, P. (2025). Cybench: A framework

for evaluating cybersecurity capabilities and risks of language models. In *Proceedings of the International Conference on Learning Representations*. https://arxiv.org/abs/2408.08926

Zheng, A., & Zhu, T. (2026). *ZERO-APT: A closed-loop adversarial framework for LLM-driven automated penetration testing under intelligent defense* [Preprint]. arXiv. https://arxiv.org/abs/2606.05567

Zhou, S., Liu, J., Hou, D., Zhong, X., & Zhang, Y. (2021). Autonomous penetration testing based on improved deep Q-network. *Applied Sciences, 11*(19), Article 8823. https://doi.org/10.3390/app11198823

Zhou, S., Liu, J., Lu, Y., Yang, J., Zhang, Y., & Chen, J. (2025). Mind the gap: Towards generalizable autonomous penetration testing via domain randomization and meta-reinforcement learning. *Frontiers of Information Technology & Electronic Engineering, 26*(12), 2511–2528. https://doi.org/10.1631/fitee.2500100

Zhou, T.-Y., Zang, Y.-C., Zhu, J.-H., & Wang, Q.-X. (2019). NIG-AP: A new method for automated penetration testing. *Frontiers of Information Technology & Electronic Engineering, 20*(9), 1277–1288. https://doi.org/10.1631/fitee.1800532

Zhou, X., Cao, S., Sun, X., & Lo, D. (2025). Large language model for vulnerability detection and repair: Literature review and the road ahead. *ACM Transactions on Software Engineering and Methodology, 34*(5), 1–31. https://doi.org/10.1145/3708522

Zhu, Y., Kellermann, A., Bowman, D., Li, P., Gupta, A., Danda, A., Fang, R., Jensen, C., Ihli, E., Benn, J., Geronimo, J., Dhir, A., Rao, S., Yu, K., Stone, T., & Kang, D. (2025). CVE-Bench: A benchmark for AI agents' ability to exploit real-world web application vulnerabilities. In *Proceedings of the International Conference on Machine Learning*. https://arxiv.org/abs/2503.17332

Авторлар туралы мәліметтер
Информация об авторах
Information about authors



Есентаев Рауан — ақпараттық қауіпсіздік магистратурасының білім алушысы, Astana IT University, Астана, Қазақстан

Есентаев Рауан — магистрант по направлению информационная безопасность, Astana IT University, Астана, Казахстан

Yessentayev Rauan — Master's student in Information Security, Astana IT University, Astana, Kazakhstan

e-mail: 255316@astanait.edu.kz, ORCID: https://orcid.org/0009-0004-9303-9701



Бегімбаева Еңлік — техника ғылымдарының кандидаты, қауымдастырылған профессор, Energo University, Алматы, Қазақстан

Бегимбаева Енлик — кандидат технических наук, ассоциированный профессор, Energo University, Алматы, Казахстан

Begimbayeva Yenlik — Candidate of Technical Sciences, Associate Professor, Energo University, Almaty, Kazakhstan

e-mail: enlikb89@gmail.com, ORCID: https://orcid.org/0000-0002-4907-3345