International Journal of Creative and Open Research in Engineering and Management
ISSN: 3108-1754 (Online)
Volume 02 Issue 11 November-2026 | Impact Factor: 3.5



Intelligent Orchestration for Automated Attack Surface Management: A Comprehensive Survey of AI Agents, Adaptive Security Assessment, and Vulnerability Discovery

Anuj Sayankar
Department of Information Technology
Prof. Ram Meghe Institute of Technology and Research, Badnera-Amravati
Amravati, Maharashtra, India
anujsayankar@gmail.com

Parth Barad
Department of Information Technology
Prof. Ram Meghe Institute of Technology and Research, Badnera-Amravati
Amravati, Maharashtra, India
parthbarad27@gmail.com

Rashmi Varma
Department of Information Technology
Prof. Ram Meghe Institute of Technology and Research, Badnera-Amravati
Amravati, Maharashtra, India
rashmivarma282@gmail.com

Mohd Kashaf
Department of Information Technology
Prof. Ram Meghe Institute of Technology and Research, Badnera-Amravati
Amravati, Maharashtra, India
kashafmohammad705@gmail.com

Dr. S. A. Chorey
Department of Information Technology
Prof. Ram Meghe Institute of Technology and Research, Badnera-Amravati
Amravati, Maharashtra, India
sonikachorey@gmail.com

How to Cite this Article:
Sayankar, A., Barad, P., Varma, R. & Kashaf, M. (2026). Intelligent Orchestration for Automated Attack Surface Management: A Comprehensive Survey of AI Agents, Adaptive Security Assessment, and Vulnerability Discovery. International Journal of Creative and Open Research in Engineering and Management, 02(11), 1-9. https://doi.org/10.55041/ijcope.v2i11.118

License:
This article is published under the terms of the Creative Commons Attribution 4.0 International License (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.
© The Author(s). Published by International Journal of Creative and Open Research in Engineering and Management.







https://doi.org/10.55041/ijcope.v2i11.118

***Abstract*—Attack surface management (ASM) is moving from periodic, tool-centric scanning towards continuous, intelligence-driven assessment. This survey reviews 31 curated sources (2021–2026) covering autonomous penetration testing, LLM-based security agents, multi-agent architectures, reinforcement-learning (RL) planners, security-tool orchestration, attack-path analysis, vulnerability validation, and evaluation benchmarks. Each source was coded against a common schema (AI technique, agent architecture, event-driven support, adaptive tool selection, fallback, validation, attack-chain coverage, testbed, metrics, findings, limitations). We present a literature matrix, comparative tables, a taxonomy, and a gap analysis. Within the reviewed corpus, the literature is strong on agent reasoning and planning but documents few complete implementations of event-driven orchestration, quality-aware tool fallback, and systematic false-positive-reducing validation within a single evaluated framework. Evaluation is also fragmented across simulations, CTF-style benchmarks, and small vulnerable testbeds. FoundFirst AI, the authors' AI-powered ASM project, is used only as practical motivation. This survey reports no new experimental results.**

***Keywords*—attack surface management, penetration testing, large language models, multi-agent systems, security orchestration, vulnerability validation, reinforcement learning, event-driven architecture.**

I. INTRODUCTION

An organisation's attack surface—its internet-facing assets, services, and exposed configurations—changes continuously as infrastructure is deployed, migrated, and retired. Operational ASM therefore requires persistent discovery, scanning, prioritisation, and verification rather than occasional assessments [20]. In practice, the underlying tools (network scanners, web scanners, exploitation frameworks) produce heterogeneous outputs, and stitching them together remains largely manual or relies on fixed scripts [3], [11].

Recent research applies AI to this problem on several fronts: RL agents that learn attack-path policies [7]–[9], [22]–[26], LLM-driven penetration-testing assistants and agents [1], [4], [6], multi-agent LLM frameworks with retrieval-augmented generation (RAG) and shared knowledge [2], [3], [10], and LLM-based validation of vulnerability alerts [19]. A recent tutorial and survey covers LLM agents for vulnerability discovery [5], and a review covers RL-based penetration testing [7]. The ASM-specific, orchestration-centred view—how tools are chosen, coordinated, recovered from failure, and how findings are validated—is less consolidated.

Practical motivation. FoundFirst AI, the authors' AI-powered ASM project, is being designed around event-driven orchestration, adaptive tool selection, fallback, and validated reporting. It is used here only to motivate the questions below; no results for it are reported and it is not the subject of this survey.

This survey addresses eight research questions: RQ1, how has automated ASM evolved; RQ2, how are AI, ML, and LLMs applied to security assessment; RQ3, how do autonomous and multi-agent systems perform security tasks; RQ4, how are tools selected, coordinated, and executed adaptively; RQ5, how do event-driven architectures support scalability and reliability; RQ6, how are vulnerabilities validated and false positives reduced; RQ7, what limitations and gaps remain; RQ8, which future directions are promising. Contributions are: (i) a schema-based synthesis of 31 sources; (ii) comparative tables and a taxonomy; (iii) an evidence-linked gap analysis. The paper follows the section order given in the contents of Sections II–XVI.

II. SURVEY METHODOLOGY

Scope and corpus. The survey covers ASM and reconnaissance, AI/ML for security assessment, LLM-based and multi-agent security systems, tool orchestration, attack-path analysis, vulnerability validation, and evaluation benchmarks. The authors assembled a curated corpus of 31 sources, grouped into ten areas (autonomous penetration testing, LLM/AI security agents, AI planning and decision making, multi-agent security, tool orchestration, attack paths and vulnerability chaining, memory/RAG/knowledge graphs, automated vulnerability detection and validation, ASM and continuous monitoring, and AI-assisted reporting and risk prioritisation). Sources are drawn from journals (e.g., Computers & Security, Expert Systems with Applications, Journal of Information Security and Applications), conference proceedings (USENIX Security, IEEE S&P, ACM ASIACCS, ICLR, IEEE NetSoft), and two benchmark reports.



Selection criteria. Sources were retained if they (a) addressed autonomous or AI-assisted security assessment, orchestration, validation, or attack-path reasoning, (b) were recent (most are 2023–2026) or foundational, and (c) reported an architecture, benchmark, or systematic review. Foundational works include attack-graph analysis [13] and early RL post-exploitation [26].

Extraction and synthesis. Each source was coded in a fixed schema: problem; approach; AI technique; agent architecture; event-driven support; adaptive selection; fallback; validation; attack-chain coverage; dataset/testbed; metrics; main result; limitation; and relevance to ASM orchestration. Capabilities were coded Yes, Partial, or No based on what each paper explicitly describes. Absence of a feature in a paper is reported as "not documented," not as proof that it does not exist. Quantitative results are quoted as reported by the original authors and are not re-computed.

Limitations of the method. The corpus is curated rather than exhaustive, no formal PRISMA flow counts are claimed, one source is a 2026 preprint-style report [21], and several are benchmark reports [28], [29] rather than peer-reviewed articles. Claims of "no study" are therefore restricted to the reviewed corpus.

III. EVOLUTION OF AUTOMATED CYBERSECURITY ASSESSMENT

The corpus suggests five overlapping stages (RQ1). Stage 1 is tool-based scanning, later integrated in orchestration platforms; an example is a security-orchestration platform combining scanners with swarm-based coordination, evaluated on 485 web applications [11]. Stage 2 is graph-based reasoning: attack graphs represent malicious events and their dependencies, but face size, metric, and cycle-handling challenges [13], and knowledge graphs combined with state machines can generate penetration paths in specialised domains [12]. Stage 3 is RL-based autonomous testing, starting with post-exploitation agents [26] and progressing to hierarchical, imitation, hindsight-replay, episodic-memory, and continual-learning formulations [8], [9], [23]–[25], with realistic training environments [22] and a review [7].

Stage 4 is LLM-driven penetration testing, where PentestGPT introduced a task-tree structure to cope with context limits [1] and PentestAgent split work into reconnaissance, analysis, and exploitation with RAG [4]. Stage 5 is multi-agent, knowledge-fused systems that add event-driven scheduling [2], dynamic knowledge graphs with a unified tool protocol [3], and RL-plus-RAG specialisation [10]. Surveys now consolidate this progression [5], [7], [30].

IV. ATTACK SURFACE MANAGEMENT AND RECONNAISSANCE

ASM extends vulnerability scanning with asset discovery, scoping, monitoring, and prioritisation. The ASM state-of-the-art study decomposes ASM into common tasks and concludes that effective ASM needs persistent orchestration and wider asset visibility beyond network mapping, while noting limited intelligent autonomy and partial support for tool coordination [20]. Table I contrasts traditional and automated ASM, drawing on [11], [13], [20]. Table II then summarises reconnaissance and vulnerability-discovery techniques found in the corpus.

TABLE I
TRADITIONAL VERSUS AUTOMATED ASM

Aspect	Traditional	Automated / continuous
Discovery	Periodic, scope-bound scans	Continuous asset discovery [20]
Tooling	Separate tools, manual correlation	Orchestrated pipelines [11], [20]
Reasoning	Analyst judgement	Graph, RL, or LLM planning [3], [9], [12]
Validation	Manual triage	Partial automation [4], [19]
Open issue	Latency, coverage gaps	Autonomy, validation, benchmarks [5], [20]

TABLE II
RECONNAISSANCE AND DISCOVERY TECHNIQUES

Technique	Sources	Observation
Scanner orchestration	[11], [20]	Integrates tools; non-LLM
LLM recon agents	[3], [4]	Recon feeds planning; knowledge graph or database
RL path planning	[8], [9], [23]–[26]	Simulated or emulated targets
Knowledge / attack graphs	[12], [13]	Strong on chains; data-dependent
CTI-to-exploit mapping	[14], [17]	Depends on public exploit data
Static alert triage	[19]	Source-code domain only
RAG assistants	[15], [16], [21]	Assistance, limited autonomy

V. AI/ML IN CYBERSECURITY

RL formulates penetration testing as sequential decision making. Reported work addresses training efficiency and generalisation: episodic memory with 80-host simulated scenarios [9]; hindsight experience replay [23]; imitation learning from expert pairs, tested on Metasploitable 2 and simulated networks [24]; hierarchical decomposition with expert priors [25]; and continual learning over 40 Vulhub environments, reporting over 60% retention of earlier knowledge [8]. PenGym shows that agents trained in realistic emulated ranges transfer better than those trained only in simulation in its largest scenario [22]. Explainability work notes that reward curves can hide behavioural collapse [31], and a comparison of environments finds no single environment satisfies all requirements [30].

LLMs and RAG supply planning, language understanding, and knowledge. They are used for autonomous testing [1], [4], local fine-tuned agents [6], RAG assistants and benchmarks [15], [16], and reproduction-step generation from CTI [17]. However, evaluations of LLM capability are sobering: models were found unreliable for standalone vulnerability detection [18], agents solve professional CTF tasks only unevenly [27], and exploit-generation success was reported at 25–50% in one benchmark [29] with limited autonomous offence in another [28]. Table III summarises AI families (RQ2).

TABLE III
AI-BASED APPROACHES

Family	Role	Strength	Limitation
RL / DRL [7]–[9], [22]–[26]	Action and path policy	Learns sequences	Simulation, retraining cost
LLM agents [1], [4], [6]	Planning, command generation	Flexible reasoning	Hallucination, context limits
LLM + RAG [2], [10], [15], [16]	Grounded knowledge	Fresher knowledge	Retrieval quality
Graphs [12], [13]	Path and chain reasoning	Explicit chains	Scalability, data completeness
LLM + static analysis [19]	Alert validation	False-positive filtering	Source-code only



VI. LLM-BASED AUTONOMOUS SECURITY AGENTS

PentestGPT, evaluated on 13 HackTheBox/VulnHub targets with 182 sub-tasks, reported 80% task completion versus 35% for GPT-3.5 and 47% for GPT-4 baselines, and identifies context loss, hallucination, and weak exploit-code generation as limits [1]. PentestAgent evaluated 67 VulHub targets and 11 HackTheBox machines and reports 74.2% success with GPT-4, but states that chaining several vulnerabilities (for example SSRF to an internal application to root) is out of scope [4]. Hackphyr fine-tunes a 7B model with a CoALA-style architecture and reports above 85% win rates in simple NetSecGame scenarios and 23–50% in complex ones, with repetitive service-discovery loops under uncertainty [6]. A tutorial-survey of 155 sources organises LLM vulnerability-discovery agents into four phases (mission scoping, exploration and tool selection, hypothesis and verification, payload refinement) and notes the absence of a shared reference architecture and consistent benchmarking [5].

Across these works (RQ3), LLM agents are effective on bounded benchmarks but remain sensitive to model choice, memory design, and human intervention. Benchmarks such as Cybench (40 professional CTF tasks) and CyberSecEval 2 and 3 provide capability evidence but are CTF-centric or benchmark-level rather than operational ASM [27]–[29].

VII. MULTI-AGENT CYBERSECURITY SYSTEMS

Multi-agent designs assign specialised roles. AutoSecAgent combines a fine-tuned 14B model, recursive memory, and RAG with event-driven task scheduling over Redis Pub/Sub, and reports 92.4% vulnerability detection rate and 58.1% exploit success rate in controlled labs and a red-team simulation [2]. PTFusion uses a MasterAgent for strategy and Recon/Attack agents via a unified tool protocol, with a dynamic knowledge graph; it reports 100% penetration success on six VulnHub environments but evaluates a limited scope [3]. AutoWAFuzzer combines LLM generation, RL, and RAG for WAF testing and is domain-specific [10]. RAVEN uses explorer, analyst, and reporter roles for memory-corruption analysis and reports an average quality score of 54.21% on 105 samples [21]. Table IV compares these architectures.

TABLE IV
AUTONOMOUS AND MULTI-AGENT ARCHITECTURES (AS REPORTED BY THE ORIGINAL AUTHORS)

System	Architecture	Model / method	Testbed	Reported result
PentestGPT [1]	Tripartite; task tree	LLM (GPT family)	13 HTB/VulnHub targets, 182 sub-tasks	80% completion vs 35% / 47% baselines
AutoSecAgent [2]	Multi-agent; event-driven scheduling	Fine-tuned 14B LLM, RAG, memory	Virtual lab; red-team simulation	92.4% VDR; 58.1% ESR
PTFusion [3]	Master / Recon / Attack agents	LLM, dynamic KG, tool protocol	6 VulnHub environments	100% PSR
PentestAgent [4]	Four-agent pipeline	LLM, RAG	67 VulHub + 11 HTB targets	74.2% (GPT-4)
Hackphyr [6]	Single agent (ReAct+, CoALA)	Fine-tuned 7B LLM	NetSecGame	>85% simple; 23–50% complex
AutoWAFuzzer [10]	Multi-agent; LLM + RL + RAG	RL policy, RAG	ModSecurity, Naxsi, WAF-Brain, CloudGuard	Improved bypass rate
RAVEN [21]	Explorer / analyst / reporter	LLM + RAG	105 samples, 15 CWE types	54.21% avg quality

VIII. INTELLIGENT SECURITY TOOL ORCHESTRATION

Tool selection and execution (RQ4) is handled in three ways in the corpus. First, platform-level orchestration integrates scanners and coordinates them, for instance via swarm-based coordination [11]; ASM studies stress persistent orchestration [20]. Second, learned policies select attack actions [8], [9], [23]–[26]. Third, LLM planners choose tools and commands, either via structured task trees [1], knowledge-graph-guided planning with a unified tool protocol [3], or priority-based scheduling [2]. A recurring problem is heterogeneous tool output, which causes context decay and imprecise commands [3].

Fallback and recovery are the weakest dimension. AutoSecAgent reassigns failed agents' tasks [2]; PentestAgent retries and debugs exploits [4]; PTFusion, PentestGPT, and Hackphyr document no explicit fallback [1], [3], [6]. None of the reviewed records describes a general mechanism that detects a failed or low-quality tool result, switches to an alternative tool, and compares or merges results. Table V summarises the approaches.

TABLE V
TOOL-SELECTION AND ORCHESTRATION APPROACHES

Approach	Selection mechanism	Coordination	Fallback / recovery	Gap
OCVA platform [11]	Swarm-based	Orchestrated scanners	None documented	Non-LLM, web-centric
AutoSecAgent [2]	Priority-based tasks	Redis Pub/Sub events	Task reassignment	No tool-quality fallback
PTFusion [3]	MasterAgent with DKG	Unified tool protocol	None documented	No result comparison
PentestAgent [4]	Planning Agent	Structured pipeline	Exploit retry / debug	No tool-level fallback
ASM study [20]	Pipeline tasks	Persistent orchestration	Partial	Limited autonomy
RL agents [8], [9], [23]–[26]	Learned policy	Single agent	None documented	Simulation-centric
AutoWAFuzzer [10]	RL policy + RAG	Four components	None documented	WAF domain only

IX. EVENT-DRIVEN AND DISTRIBUTED ARCHITECTURES

Direct evidence for event-driven security assessment (RQ5) is thin. AutoSecAgent is the only reviewed system explicitly using event-driven scheduling with a message bus, and it reports resource and scaling metrics including evaluation across up to 20 hosts [2]. The ASM study identifies persistent orchestration as necessary [20]. PTFusion reacts to updated knowledge but is not described as an event-bus architecture [3]; PentestAgent, Hackphyr, and the RL systems use sequential or loop-based control [4], [6], [8], [9]. Consequently, claims that event-driven designs improve scalability and reliability for ASM are, within this corpus, supported mainly by one system and require independent benchmarking. As design rationale (not reported evidence), publish/subscribe decoupling is expected to allow



asynchronous workers, retries, and phase transitions triggered by new findings; this remains to be evaluated.

X. VULNERABILITY VALIDATION AND REPORTING

Validation (RQ6) is mostly partial. PentestAgent checks that exploits are functional [4], and AutoSecAgent relies on tool-confirmable exploit success [2]; neither is a full finding-validation pipeline covering reproducibility, evidence, and impact. For source code, a hybrid of multiple SAST tools, code property graphs, and an LLM reports 96.43% precision and 87.10% F1 on SARD and 85.20% F1 on Devign/FFmpeg [19]. Vuln2Action generates reproduction steps by mapping CVE/NVD, CWE, CAPEC, and Exploit-DB records, but depends on public exploit data [17]. Risk-based work defines exploit-chain and exploitation-risk scores from vulnerability and threat data [14]. RAVEN evaluates reports and evidence quality, but only for memory-corruption code [21]. SecLLMHolmes shows that LLM judgement on code vulnerabilities is unreliable without safeguards [18].

Thus, validation methods that work well exist in narrow domains (source code, known exploits), whereas live, multi-tool ASM findings lack a validated, evidence-grounded reporting stage; false-positive reduction is demonstrated chiefly in [19].

XI. COMPARATIVE ANALYSIS

Table VI codes all 31 sources on seven capabilities (Y = yes, P = partial or domain-limited, N = not documented). Table VII lists testbeds, headline results, and limitations for selected sources. Three patterns emerge. (1) AI use is widespread, but only a few sources combine multi-agent coordination with event-driven support and adaptive selection [2], [3]. (2) No source is coded Yes for fallback; most are No. (3) Attack-chain coverage is strong in RL and graph work [7], [9], [12], [13], [22]–[26] but absent or out of scope in several LLM agents [4], [10]. Results are not directly comparable because metrics and testbeds differ (task completion, PSR, VDR/ESR, win rate, F1, quality score).

TABLE VI
LITERATURE MATRIX: CAPABILITY COVERAGE (Y = YES, P = PARTIAL, N = NOT DOCUMENTED)

Ref	Venue, year	Focus	AI	Agent	Event	Adaptive	Fallback	Valid.	Chain
[1]	USENIX Sec. 2024	LLM pentesting, task tree	Y	P	N	N	N	P	P
[2]	J. Supercomput. 2026	Multi-agent pentest, RAG, memory	Y	Y	Y	Y	P	P	Y
[3]	Inf. Fusion 2026	Web pentest, KG fusion, tool protocol	Y	Y	P	Y	N	P	P
[4]	ASIACCS 2025	Four-agent pentest with RAG	Y	Y	N	Y	P	P	N
[5]	IEEE Access 2026	Survey: LLM agents, vulnerability discovery	Y	Y	P	Y	P	Y	P
[6]	ESWA 2026	Local fine-tuned SLM cyber agent	Y	P	N	Y	N	P	Y
[7]	ESWA 2026	Review: RL pentesting	Y	Y	N	Y	N	P	Y
[8]	ESWA 2025	Continual RL pentesting	Y	Y	N	Y	N	P	P
[9]	CMES 2024	RL path planning, episodic memory	Y	Y	N	Y	N	N	Y
[10]	ESWA 2026	WAF testing: multi-agent, RL, RAG	Y	Y	P	Y	N	P	N
[11]	Comput. Secur. 2022	Orchestration platform, swarm	P	N	P	Y	N	P	N
[12]	JKSU-CIS 2026	KG path generation (vehicles)	P	N	P	Y	N	Y	Y
[13]	Comput. Secur. 2023	Review: attack graphs	P	N	N	P	N	P	Y
[14]	Appl. Sci. 2023	Exploitation risk assessment	N	N	N	P	N	P	Y
[15]	Procedia CS 2026	RAG-LLM pentest chatbot	Y	P	N	P	N	P	N
[16]	ISWA 2026	RAG-LLM benchmarking	Y	P	N	P	N	P	P
[17]	JISA 2026	LLM reproduction steps, exploit mapping	Y	N	N	P	N	P	P
[18]	IEEE S&P 2024	LLM vulnerability reasoning benchmark	Y	N	N	N	N	P	N
[19]	JSS 2026	SAST alert validation with LLM	Y	N	N	P	N	Y	P
[20]	IEEE NetSoft 2025	ASM state of the art	N	N	P	P	P	P	N
[21]	Preprint 2026	RAG multi-agent memory-corruption analysis	Y	Y	N	P	N	Y	P
[22]	Comput. Secur. 2025	PenGym RL training environment	Y	Y	N	Y	N	P	Y
[23]	Comput. Secur. 2025	HER-PT	Y	Y	N	Y	N	P	Y
[24]	Comput. Secur. 2023	GAIL-PT	Y	Y	N	Y	N	P	Y
[25]	Comput. Secur. 2023	Hierarchical DRL with expert priors	Y	Y	N	Y	N	P	Y
[26]	Comput. Secur. 2021	DRL post-exploitation	Y	Y	N	Y	N	P	Y
[27]	ICLR 2025	Cybench: LM-agent CTF benchmark	Y	Y	N	P	N	P	P
[28]	arXiv 2024	CyberSecEval 3	Y	P	N	P	N	P	P
[29]	Meta 2024	CyberSecEval 2	Y	N	N	N	N	P	P
[30]	IoT 2026	Review: AI autonomous pentesting	Y	Y	N	Y	N	P	P
[31]	JISA 2026	Explainable RL cyber agents	Y	Y	N	Y	N	P	Y

TABLE VII
TESTBEDS, RESULTS, AND LIMITATIONS OF SELECTED SOURCES (AS REPORTED)

Ref	Testbed / dataset	Reported result	Limitation
[1]	13 HTB/VulnHub targets, 182 sub-tasks	80% task completion	Context limits; hallucination; human help useful
[2]	Virtual lab, 10–20 hosts; red-team simulation	92.4% VDR; 58.1% ESR	Tool fragility; prompt injection; limited production validation
[3]	6 VulnHub environments	100% PSR	Limited scope; depends on LLM and context
[4]	67 VulHub, 11 HTB targets	74.2% with GPT-4	No vulnerability chaining; knowledge gaps
[6]	NetSecGame; 1,641 training examples	>85% simple; 23–50% complex	Repetitive loops; simulation only
[8]	40 Vulhub environments	>60% retention; forward transfer	Environment dependence
[9]	Simulated networks up to 80 hosts	Faster, stabler convergence	No finding validation; simulation
[11]	485 web applications	Reported to outperform compared tools	Non-LLM; web-centric
[12]	Vehicle security knowledge graph	36 of 38 historical attacks reproduced (94.7%)	Data completeness; static model



Ref	Testbed / dataset	Reported result	Limitation
[17]	CVE/NVD, CWE, CAPEC, Exploit-DB dataset	Structured exploit flows feasible	Exploit-DB dependency; textual output
[18]	228 code scenarios	LLMs unreliable for standalone detection	Code, not live targets
[19]	SARD, Devign/FFmpeg, CASTLE	96.43% precision, 87.10% F1 (SARD)	Source-code domain only
[21]	105 samples, 15 CWE types	54.21% average quality	Narrow domain
[27]	40 professional CTF tasks	Performance varies substantially	CTF-centric
[29]	MITRE-ATT&CK-based and exploit tests	25–50% exploit success reported	Benchmark-level, not operational

XII. TAXONOMY OF EXISTING APPROACHES

Fig. 1 organises the corpus along five dimensions: reasoning technique, coordination, tool/action selection, resilience, and validation. The figure shows that the reasoning and selection branches are populated by many studies, whereas the resilience branch contains few documented mechanisms. Placement follows Table VI and the discussion in Sections V–X.

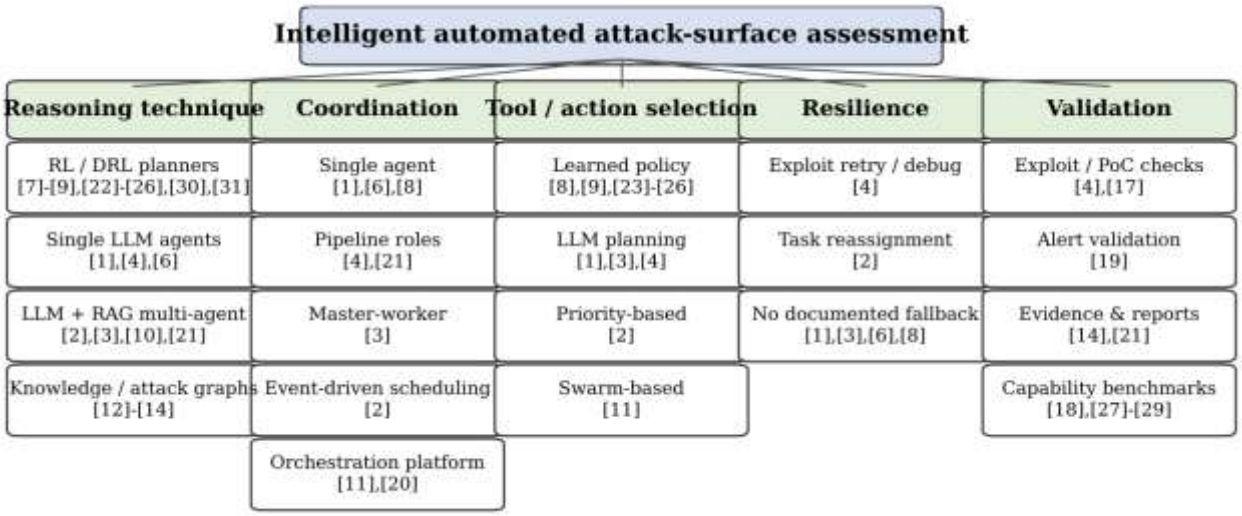


Fig. 1. Taxonomy of AI-based automated security assessment approaches in the reviewed corpus (numbers are reference indices).

XIII. RESEARCH GAPS AND CRITICAL ANALYSIS

Table VIII links each gap to evidence in the corpus (RQ7). The gaps are observations about the reviewed sources, not claims that the problems are unsolved elsewhere.

TABLE VIII
RESEARCH GAPS AND OPPORTUNITIES

Gap	Evidence in corpus	Opportunity
Fragmented tools and outputs	Heterogeneous outputs cause context decay [3]; ASM needs orchestration [20]	Normalised result schema; unified tool interfaces
Limited intelligent tool selection	Mostly learned policies or LLM planning on fixed toolsets [1], [3], [8]	Cost- and quality-aware tool selection
Weak agent coordination	Few multi-agent systems with explicit messaging [2]; reliability issues [5]	Evaluated coordination protocols
No adaptive fallback	No general Tool A to Tool B mechanism documented [1], [3], [4], [6]	Failure and quality detection with alternative tools
Inconsistent validation / false positives	Validation partial [2], [4]; strong only in code [19]; LLM unreliability [18]	Evidence-based live-finding validation
Incomplete attack-chain coverage	Chaining out of scope [4]; strong in RL/graph [9], [12]	Combine chain reasoning with live agents
Scale, reliability, benchmarks	Small testbeds [3], [4]; CTF-centric [27]; no unified metrics [5]	Shared ASM benchmark with failure injection

Further limitations apply across the corpus: hallucination and context limits [1], [5]; reliance on simulation [6], [9], [22]; prompt-injection and tool-fragility risks [2]; limited production validation [2]; and lack of transparency [5], [31]. Many headline success rates come from small, intentionally vulnerable testbeds and should not be generalised to enterprise ASM.

XIV. FUTURE RESEARCH DIRECTIONS

The following are research proposals, not implemented or validated results. (1) Quality-aware tool selection and fallback: detect failed, empty, or inconsistent results and switch or merge tools, evaluated with injected failures. (2) Event-driven orchestration benchmarks: compare event-bus and sequential pipelines on throughput, latency, and recovery. (3) Evidence-grounded validation: combine reproducibility checks, evidence capture, impact assessment, and false-positive filtering for live findings, extending the approach shown for



source code [19]. (4) Attack-chain synthesis: link agent-discovered findings to attack-path graphs [12], [13]. (5) Shared ASM benchmarks beyond CTFs, with ground truth and common metrics [5], [27]. (6) Safe and explainable autonomy: guardrails against prompt injection, local models for privacy [6], and explainability [31]. FoundFirst AI is being designed along directions (1)–(4); their evaluation is future work.

XV. DISCUSSION

Answering the research questions: RQ1–RQ2, ASM has moved from scanner orchestration to graph, RL, and LLM-based reasoning (Sections III–V). RQ3–RQ4, agents perform well on bounded benchmarks, but tool selection is mostly planner-driven and fallback is rarely documented (Sections VI–VIII). RQ5, event-driven evidence is limited to one explicit system (Section IX). RQ6, validation is domain-limited (Section X). RQ7–RQ8 are addressed in Sections XIII–XIV.

Fig. 2 shows a conceptual reference architecture that integrates the capabilities found separately in the corpus: an orchestrator with a decision engine, an event bus, specialised agents, a shared knowledge layer, validation, attack-path reasoning, and risk prioritisation. It is a synthesis for discussion and motivation, not a validated design, and is not a contribution of any reviewed paper.

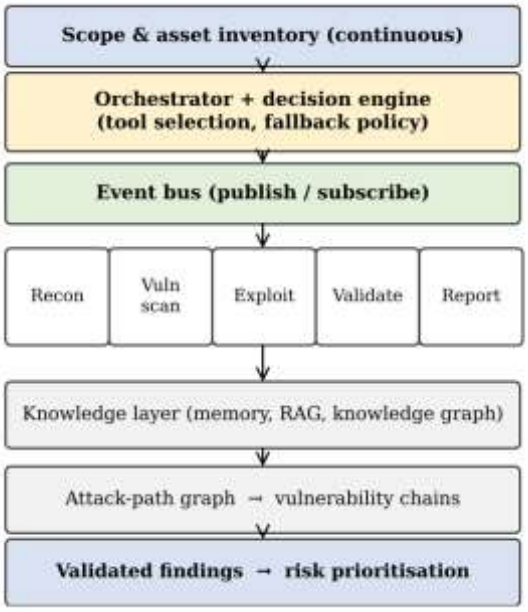


Fig. 2. Conceptual event-driven ASM reference architecture (synthesis for discussion; not experimentally validated).

Threats to validity include corpus curation bias, reliance on authors' reported metrics, rapid change in LLM capabilities, and coding based on what papers explicitly describe.

XVI. CONCLUSION

This survey synthesised 31 sources on AI-driven automated attack surface assessment. The literature offers capable planners and agents, but within the reviewed corpus integrated event-driven orchestration, quality-aware fallback, and live-finding validation are not yet documented together in one evaluated framework. Addressing this with shared benchmarks and failure-injection experiments is a promising direction.

ACKNOWLEDGMENT

The authors thank the Department of Information Technology, Prof. Ram Meghe Institute of Technology and Research, Badnera-Amravati, for support.

REFERENCES

[1] G. Deng et al., "PentestGPT: Evaluating and harnessing large language models for automated penetration testing," in USENIX Security Symp., 2024.
[2] "AutoSecAgent: A semi-automated AI-driven penetration testing framework through recursive memory and real-time RAG," J. Supercomput., Springer, 2026.
[3] "PTFusion: LLM-driven context-aware knowledge fusion for web penetration testing," Inf. Fusion, Elsevier, 2026.
[4] "PentestAgent: Incorporating LLM agents to automated penetration testing," in Proc. ACM Asia Conf. Comput. Commun. Secur. (ASIACCS), 2025.
[5] "LLM-based intelligent agents for cybersecurity: A tutorial and survey of automated vulnerability discovery," IEEE Access, 2026.
[6] "Building adaptative and transparent cyber agents with local language models," Expert Syst. Appl., Elsevier, 2026.
[7] "Autonomous penetration testing using reinforcement learning: A review and perspectives," Expert Syst. Appl., Elsevier, 2026.
[8] "SCRIPT: A scalable continual reinforcement learning framework for autonomous penetration testing," Expert Syst. Appl., Elsevier, 2025.
[9] "Efficient penetration testing path planning based on reinforcement learning with episodic memory," Comput. Model. Eng. Sci., 2024.
[10] "AutoWAFuzzer: An adaptive framework for web application firewall penetration testing with multi-agent system and RAG-enabled reinforcement learning," Expert Syst. Appl., Elsevier, 2026.
[11] "A proactive approach to assess web application security through the integration of security tools in a security orchestration platform," Comput. Secur., Elsevier, 2022.
[12] "KGB-Path: A knowledge graph-based framework for penetration test path generation in intelligent connected vehicles," J. King Saud Univ. Comput. Inf. Sci., 2026.
[13] "Attack graph analysis: An explanatory guide," Comput. Secur., Elsevier, 2023.
[14] "Vulnerability exploitation risk assessment based on offensive security approach," Appl. Sci., 2023.
[15] "Design and implementation of a RAG-enhanced LLM chatbot for penetration testing tasks," Procedia Comput. Sci., Elsevier, 2026.
[16] "RAG-augmented LLMs for penetration testing: A benchmarking study of open-source LLM models," Intell. Syst. Appl., Elsevier, 2026.
[17] "Vuln2Action: An LLM-based framework for generating vulnerability reproduction steps and mapping exploits," J. Inf. Secur. Appl., Elsevier, 2026.
[18] S. Ullah et al., "LLMs cannot reliably identify and reason about security vulnerabilities (yet?): A comprehensive evaluation, framework, and benchmarks," in Proc. IEEE Symp. Secur. Privacy, 2024.
[19] "Enhancing static vulnerability alert validation using large language models," J. Syst. Softw., Elsevier, 2026.
[20] "Attack surface management: State of the art and operational challenges," in Proc. IEEE NetSoft, 2025.
[21] "RAVEN: Retrieval-augmented vulnerability exploration network for memory corruption analysis in user code and binary programs," 2026 (research paper; venue to be confirmed).
[22] "PenGym: Realistic training environment for reinforcement learning pentesting agents," Comput. Secur., Elsevier, 2025.
[23] "HER-PT: An intelligent penetration testing framework with hindsight experience replay," Comput. Secur., Elsevier, 2025.
[24] "GAIL-PT: An intelligent penetration testing framework with generative adversarial imitation learning," Comput. Secur., Elsevier, 2023.
[25] "A hierarchical deep reinforcement learning model with expert prior knowledge for intelligent penetration testing," Comput. Secur., Elsevier, 2023.
[26] "Automating post-exploitation with deep reinforcement learning," Comput. Secur., Elsevier, 2021.
[27] A. K. Zhang et al., "Cybench: A framework for evaluating cybersecurity capabilities and risks of language models," in Proc. ICLR, 2025.
[28] S. Wan et al., "CYBERSECEVAL 3: Advancing the evaluation of cybersecurity risks and capabilities in large language models," arXiv:2408.01605, 2024.
[29] M. Bhatt et al., "CYBERSECEVAL 2: A wide-ranging cybersecurity evaluation suite for large language models," Meta Research, 2024.
[30] "Autonomous pentesting using artificial intelligence: From the cybersecurity point of view," Internet of Things, Elsevier, 2026.
[31] "Unveiling the black box: A multi-layer framework for explaining reinforcement learning-based cyber agents," J. Inf. Secur. Appl., Elsevier, 2026.