VERIIR-AGENT: RISK-AWARE AND AUDITABLE PLANNING FOR LLM-BASED INCIDENT RESPONSE

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models can generate remediation recommendations for cybersecurity incident response, but natural-language outputs are difficult to verify field by field and to audit trade-offs among recovery benefits, execution risks, and business impacts. This paper proposes VERIIR-AGENT, which decomposes response planning into three separately evaluable layers: generation, selection, and verification quality. For generation, typed action records replace free-form responses, followed by deterministic model-independent checks of evidence, commands, policies, completeness, and attack paths. For selection, a multi-objective ranking method considers recovery benefits, evidence support, execution risks, business impacts, and rollback capability over a fixed candidate pool. For verification, a conditional framework decomposes safety guarantees into "coverage gaps" and "false passes on covered properties," corresponding to checker scope and precision. Experiments use 2,219 incident-response cases while holding the model, test cases, retrieval context, and standard decoding parameters constant, comparing free-form generation, prompt-based structuring, and schema-constrained decoding. Schema constraints increase schema validity from 46.2% to 98.0% and completeness from 61.8% to 90.8%, while action-type matching increases only from 72.7% to 75.1%. On 4,438 candidate actions, risk-aware ranking reduces unsafe selection from 17.6% to 10.9% and high-impact selection from 43.8% to 35.7%, while increasing rollback coverage from 21.3% to 31.8%; recovery relevance decreases from 83.1% to 80.2% as the associated cost. These results show that structural compliance, selection preferences, and verification guarantees should be evaluated separately rather than collapsed into a single "reliability" metric, and none should be equated with successful real-world execution or production-grade safety.

1 INTRODUCTION

Cybersecurity incident response requires analysts to select containment, investigation, eradication, hardening, and recovery actions under incomplete and continuously evolving evidence. Traditional playbooks can accumulate experience, but they rely on manual maintenance and are difficult to adapt promptly to new attack methods and asset states (Woods et al., 2023; Schlette et al., 2024; Nelson et al., 2025). Large language models (LLMs) expand the coverage of candidate recommendations and are therefore increasingly being used for incident assessment, response planning, and security operations assistance (Hays & White, 2024; Lin et al., 2025; Kramer et al., 2025; Castro et al., 2025); among these approaches, a lightweight-model-based "fine-tuning–retrieval–look-ahead planning" pipeline has shown potential for reducing hallucinations (Hammar et al., 2026). However, a seemingly reasonable natural-language recommendation may still omit the target host, cite vulnerabilities unrelated to the incident, provide unparsable commands, or recommend high-impact operations without a rollback plan. Treating such outputs directly as executable actions not only hinders machine verification but also obscures the risk trade-offs involved in the decision-making process.

Existing work typically evaluates response recommendations using aggregate metrics such as similarity to reference answers, task completion, or recovery time. Such evaluations answer the question of "whether the model's recommendations resemble the reference actions," but do not distinguish among three different issues: whether the output satisfies processable representation requirements;

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

what is actually selected from the available candidates; and which risk properties are covered by the verifier. Structured prompting, post-processing repair, candidate reranking, and rule-based checking are also often used together, making the source of metric improvements unclear. A checker's "pass" should likewise not be misinterpreted as a safety fact: it is meaningful only under the declared checking scope and precision assumptions.

To address this gap, this paper reorganizes the methodology and evidence of VERIIR-AGENT. The system first organizes the incident context into a graph context with evidence provenance, and then generates multiple typed actions; an independent deterministic checker attaches findings to the actions and blocks disqualifying errors; subsequently, a risk-aware ranker performs multi-objective ranking among the accepted candidates. Figure 1 presents the overall workflow. The core contribution of this paper is not to claim that a single end-to-end score can establish safety, but rather to distinguish representation constraints on the generation side, preference modeling on the selection side, and verification quality on the theoretical side, and then connect these three components.

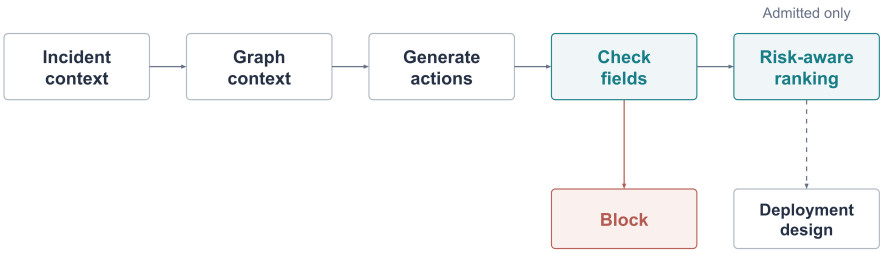


Figure 1: Overview of VERIIR-AGENT. Incident-centered graph context supplies traceable evidence for generation; typed actions enable field-level checks; blocking findings govern admission, while the risk-aware objective ranks admitted candidates only. The approval, audit, and isolated-execution components belong to the deployment design; the experiments in this paper focus on generation constraints, fixed-pool ranking, and verification-quality analysis.

This paper makes three contributions corresponding to the system layers.

1. **Representation and verification layer.** We represent free-form recommendations as typed actions that can be checked field by field, distinguish schema compliance, field completeness, command parsability, and action-type matching, and separate the respective effects of "prompting the model to adopt a structure" and "enforcing the schema at the decoding stage" through controlled comparisons.

2. **Selection layer.** We compare recovery-prioritized ranking with multi-objective risk-aware ranking on exactly the same candidate set, characterize the trade-offs among recovery benefit, evidence support, execution risk, business impact, and rollback capability, and isolate the ranking effect through candidate consistency checks.

3. **Verification quality theory layer.** We decompose the upper bound on the probability that an accepted action remains unsafe into a configuration coverage gap and a false-pass term for covered properties. This decomposition shows that merely increasing the number of candidates cannot reduce the safety gap; coverage expansion and checker improvement are two distinct paths that can be advanced separately.

The experiments use a fixed evaluation set of 2,219 cases across all three output-control conditions. For ranking, each case contains two shared candidate actions, yielding 4,438 candidates in total. This design isolates the effects of output control and ranking: schema constraints primarily affect representational compliance, whereas risk-aware ranking changes the risk–recovery trade-off over an unchanged candidate pool. Historical prediction exports are analyzed separately as an audit of task-mixing and extraction effects, and are not used to attribute rule-based findings from non-action tasks to action generation.

2 Related Work

Incident response automation spans machine-readable playbooks, optimization, reinforcement learning, simulation, and LLM-assisted planning. CACAO standardizes workflows (OASIS Open, 2023), while methods evaluated in environments such as CAGE-2 (CAGE, 2022) depend on maintained playbooks or high-fidelity models. The closest work to ours is Hammar et al. (Hammar et al., 2026), which combines instruction tuning, threat-intelligence retrieval, and look-ahead planning, selects actions by expected recovery time, and derives a hallucination bound under specific assumptions. However, it uses free-form actions, model self-consistency for filtering, and a single recovery objective. VERIIR-AGENT instead uses typed actions, deterministic model-independent checks, and multi-objective ranking over recovery, risk, impact, and rollback, while treating verification separately through coverage and checker precision. RAG, GraphRAG, and ontology-based retrieval improve grounding (Ayala & Bechard, 2024; Edge et al., 2024; Zhao et al., 2024), but do not guarantee complete fields, valid commands, or policy compliance; schema constraints, external rules, authorization, and human oversight provide complementary safeguards (Li et al., 2026; Syros et al., 2025). Likewise, self-consistency and hallucination detectors (Manakul et al., 2023; Sriramanan et al., 2024) do not establish coverage of all safety-relevant properties, and retrieval may introduce indirect prompt injection (Debenedetti et al., 2024). We therefore distinguish coverage gaps from false passes on covered properties rather than treating checker pass rates as overall safety, and do not estimate a numerical safety bound from the current data (Abbasi-Yadkori et al., 2024).

3 Problem Definition and Design Goals

3.1 Incident Context and Response Actions

Given an incident context I, the system outputs an ordered response plan $\rho = (a_1, \ldots, a_n)$. The controlled ranking experiments in this paper evaluate only the first selected action for each case, while the deployment design can handle plan prefixes with dependencies. I consists of available observations such as alerts, logs, IOCs, vulnerabilities, assets, services, and attack stages. To ensure that generation and checking are grounded in the same facts, the system organizes the normalized incident object and its two-hop neighborhood into an incident-centric knowledge subgraph K_I, and provides the summary together with the original instruction to the model.

A candidate action is represented as

$$a = \langle \text{phase}, \text{type}, \text{target}, \text{command}, \text{pre}, \text{effect}, \text{evidence}, \text{risk}, \text{rollback} \rangle, \tag{1}$$

that is, nine fields corresponding to the response phase, controlled action type, target, optional command or API call, preconditions, expected effect, supporting evidence, operational risk description, and rollback plan. Compared with free text, this representation does not require the checker to infer the target or command from explanatory paragraphs; missing fields are also visible and auditable states.

We distinguish four types of response-related failures: unsupported actions, invalid actions, unsafe actions, and ineffective actions. These categories may overlap. The rules implemented in this paper cover only part of these definitions: for example, command parsing does not prove successful execution in the target environment, and the presence of a vulnerability identifier in the current context does not prove that the target version is affected. Therefore, field records serve as the interface for checking rather than as a substitute for factual correctness.

3.2 Three-Layer Objectives and Evaluation Questions

This paper decomposes the evaluation into three distinct questions that cannot substitute for one another:

- **Representation question:** When the model, test set, retrieval context, and standard decoding parameters are held constant, how do prompt-based structuring and schema constraints respectively affect the schema validity rate, completeness, and other action-related metrics of the raw outputs?

- **Selection question:** When the candidate actions are exactly the same, how does risk-aware ranking change the unsafe labels, high-impact labels, rollback coverage, and recovery relevance of the selected actions?
- **Verification question:** Under what assumptions can the acceptance rules exclude covered hazardous properties; when these assumptions do not fully hold, what actionable components constitute the residual risk?

The answers to these three questions cannot substitute for one another. Schema validity does not imply appropriate selection; a lower-risk selection cannot, in turn, prove that the checker has not missed any issues; similarly, similarity to a reference answer does not prove that the recommendation is executable.

4 Method: From Typed Generation to Risk-Aware Decision Making

4.1 Evidence Conditioning and Typed Action Generation

For each case, the incident instruction is parsed into a structured object and represented as an incident-centered graph containing IOCs, CVEs, assets, services, attack stages, ATT&CK techniques, mitigations, and security rules. The model is conditioned on a textual summary of the incident node's two-hop neighborhood, with ATT&CK techniques, mitigations, and rules drawn from fixed local catalogs. The prototype does not learn graph representations or assume real-time synchronization with an enterprise knowledge base. Action generation uses a LoRA-adapted DeepSeek-R1-Distill-Qwen-14B under three controlled output settings: *free-form generation*, *prompt-based structuring*, and *schema-constrained decoding*. Schema validity is evaluated directly on unrepaired raw outputs, so improvements reflect constraints applied during generation rather than post-processing repair. A normalization layer subsequently maps supported response formats to typed action records for downstream verification while preserving the original output. Historical prediction exports contain only `Action` and `Explanation`, with additional typed fields introduced during post-processing; we therefore use these records only for post-extraction audit and not to attribute effects to schema-constrained generation.

4.2 Field-Level Deterministic Verification and Acceptance

For each normalized action in Equation equation 1, the prototype performs five deterministic checks: (1) CVE plausibility and context membership; (2) command parsing and executable whitelisting; (3) high-risk policies and rollback requirements; (4) required-field completeness; and (5) consistency with ATT&CK techniques or attack stages. Each finding records its category, severity, and explanation. Malformed CVEs, unparsable commands, prohibited operations, and destructive actions without rollback are treated as errors and block the corresponding action, whereas missing evidence, unknown executables, and graph mismatches are warnings and do not automatically block it. A response is accepted only if no action is blocked, while all accepted actions, blocked actions, and findings are retained. These checks provide rule-based findings rather than independent safety guarantees: for example, CVE membership does not verify affected versions, and command parsing does not establish execution outcomes.

4.3 Multi-Objective Risk-Aware Ranking over a Fixed Candidate Pool

Acceptance does not determine the priority among candidates. For each accepted action, we define

$$S_{\text{risk}}(a) = \alpha G_{\text{rec}}(a) + \beta E(a) - \gamma R_{\text{exec}}(a) - \delta B(a) + \eta R_b(a). \tag{2}$$

where G_{rec} denotes the expected recovery benefit, E denotes evidence support, R_{exec} denotes execution risk, B denotes business impact, and R_b denotes rollback availability and quality; all terms are normalized to $[0, 1]$. The coefficients express organizational preferences regarding recovery speed, impact, and reversibility. Blocking findings take precedence over this score, and the ranking process cannot re-admit rejected actions.

To isolate the causal effect of the selection layer, we compare the baseline of "ranking solely by recovery relevance" with Equation equation 2 without regenerating candidates. For each case, the candidate IDs must be exactly identical under the two ranking methods, and all candidates are included in both rankings. We report candidate-set consistency and ranking coverage as experimental integrity checks, separately from the final selection change rate. The deployment design can additionally handle action dependencies, approval hierarchies, and evidence snapshots; this paper does not treat these capabilities as validated automated execution.

4.4 Coverage Decomposition of Verification Guarantees

Let A_u denote the event that an issued action is unsupported, invalid, or policy-violating; C_k that verifier k covers a required property; FN_k that a covered violation is incorrectly passed; and C_{gap} that at least one required property is not covered by any configured verifier. Acceptance requires all mandatory verifiers to pass.

Proposition 1 (Conditional Exclusion within Coverage). If every mandatory verifier is reliable within its declared scope and all mandatory checks must pass, then no issued action can violate a covered mandatory property. Otherwise, a verifier covering the violated property would have to both detect the violation and return a pass, contradicting the reliability assumption. This result is conditional and does not imply complete coverage or perfect verifier reliability in the current implementation.

With imperfect verifiers, the union bound yields

$$\Pr(A_u) \leq \Pr(C_{\text{gap}}) + \sum_k \Pr(FN_k \cap C_k). \tag{3}$$

The first term captures uncovered properties, while the second captures false passes within covered properties. Expanding verification scope, e.g., to versions, target states, or dependencies, primarily reduces the former; improving parsers, fact verifiers, and policy classifiers primarily reduces the latter. Increasing the candidate pool alone reduces neither term and may instead increase verification and planning cost. Because the current experiments do not measure all coverage gaps or false-pass probabilities, we do not report a numerical safety upper bound.

5 Experimental Setup

5.1 Data, Models, and Evidence Sources

The controlled experiments use 2,219 action-generation cases and, for ranking, two fixed candidates per case (4,438 candidates in total). Two earlier exports containing 6,813 mixed-task records each are retained only for source auditing. Because their recorded model paths differ, they are not used for causal attribution of graph-retrieval effects.

The existing model configuration uses DeepSeek-R1-Distill-Qwen-14B with a LoRA rank of 64, a scaling factor of 128, dropout of 0.05, a learning rate of 1.5×10^{-4}, a batch size of 1, one training epoch, and a random seed of 99125. Earlier data processing used a deterministic 80/20 split, with sampling disabled during held-out inference and a maximum generation length of 6,000 tokens. These historical configurations do not replace the complete manifest required for the newer controlled runs; this paper reports only descriptive results based on the provided aggregated data.

5.2 Controlled Protocol and Metrics

The structure experiment fixes the checkpoint, case IDs, retrieval context, and standard decoding parameters, and varies only the output control mechanism; therefore, the formatting instructions introduced by prompt-based structuring and schema constraints are allowed to differ, while the incident task and evidence remain unchanged. We report the schema validity rate of the raw outputs, completeness of required fields, action-type matching, and command validity; command validity represents only the level of checking achieved by the actual evaluator and does not imply real-world execution.

The ranking experiment fixes all candidate actions and compares only the recovery-prioritized and risk-aware objectives. We report the unsafe selection rate, high-impact selection rate, rollback coverage, and recovery relevance of the first selected action. A six-level sensitivity analysis is conducted on the execution-risk coefficient: $\alpha, \beta, \delta, \eta$ are held constant, while $\gamma_m = m\gamma$, where $m \in \{0.5, 0.75, 1, 1.25, 1.5, 2\}$. All nominal weights are fixed prior to evaluation and are not tuned on the test set; only the execution-risk coefficient is varied in the sensitivity analysis.

All reported values are descriptive comparisons under a fixed and controlled experimental setting, and percentage-point differences are used to quantify the observed effects of the ranking objective. Accordingly, the analysis focuses on within-pool behavioral differences rather than statistical generalization across repeated runs. The "unsafe" and "high-impact" measures are evaluator-defined labels and are used to characterize relative selection behavior, not to estimate observed harm in deployment.

6 Experimental Results

6.1 Schema Constraints Primarily Improve Representational Compliance

Figure 2 separates the effects of prompt-level structuring and decoding-level enforcement. The schema validity rates of free-form generation, prompt-based structuring, and schema-constrained decoding are 46.2%, 64.7%, and 98.0%, respectively. Thus, prompt-based structuring improves over free-form generation by 18.5 percentage points, while schema constraints provide a further improvement of 33.3 percentage points; the total improvement relative to free-form generation is 51.8 percentage points. Completeness is 61.8%, 73.5%, and 90.8%, respectively, with schema constraints improving over free-form generation by 29.0 percentage points.

On the other hand, action-type matching increases only from 72.7% to 74.0% and 75.1%, while command validity increases from 80.9% to 82.0% and 83.2%. Relative to free-form generation, schema constraints improve these two metrics by only 2.4 and 2.3 percentage points, respectively. The contrast between these results indicates that constraining outputs to a parseable and field-complete representation space primarily addresses interface compliance; neither the factual grounding of actions nor successful execution can be established by this alone.

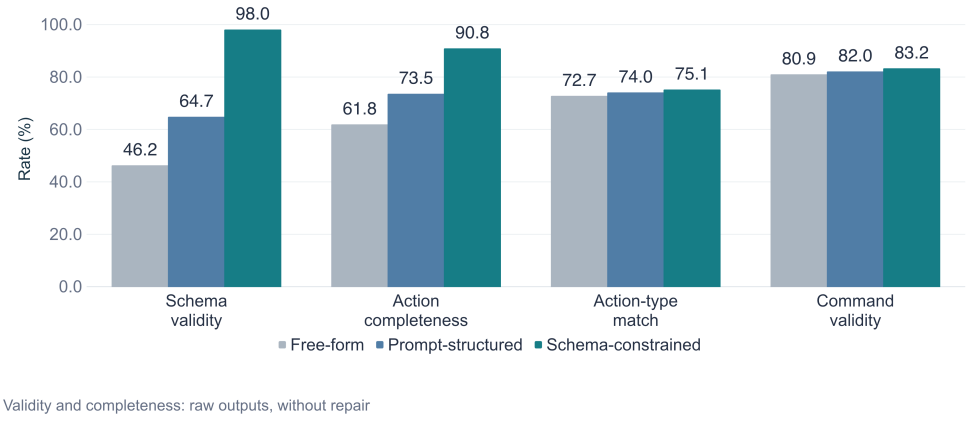


Figure 2: Output-control comparison on the same 2,219 cases. The three conditions fix the checkpoint, test cases, retrieval context, and ordinary decoding parameters; schema validity is assessed on raw, unrepaired output.

6.2 Risk-Aware Ranking Changes the Risk–Recovery Balance

On the shared candidate pool, risk-aware ranking reduces the unsafe selection rate from 17.6% to 10.9%, a decrease of 6.7 percentage points; the high-impact selection rate decreases from 43.8% to

35.7%, a reduction of 8.1 percentage points. Rollback coverage increases from 21.3% to 31.8%, an improvement of 10.5 percentage points. At the same time, recovery relevance decreases from 83.1% to 80.2%, a decline of 2.9 percentage points (Figure 3). The latter is a relevance label rather than a measured recovery time.

Candidate ID consistency and ranking coverage are both 100%, indicating that the two objectives operate on the same candidates; 83.7% of cases select the same action, while 16.3% result in a changed selection. This control supports attributing the above differences to ranking preferences within a fixed candidate set; however, it cannot yet be concluded from these results whether the quality of the candidate pool itself improves accordingly or whether each changed selection is superior in a real-world environment.

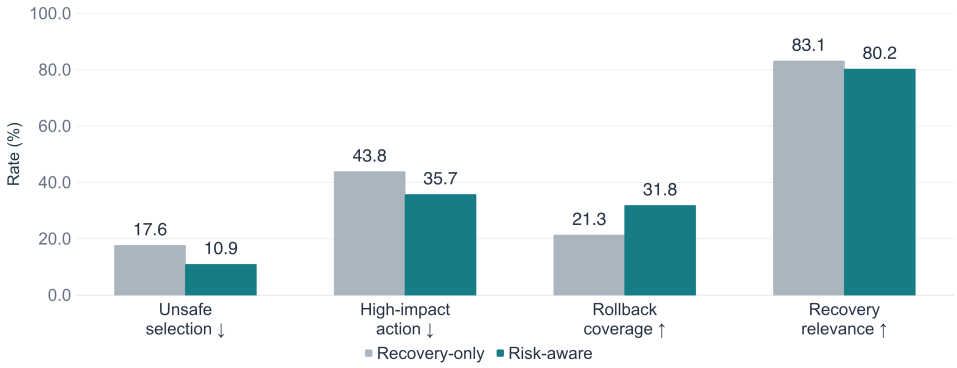


Figure 3: Recovery-only versus risk-aware ranking on the same candidate pool. Risk-aware ranking lowers unsafe and high-impact selection and raises rollback coverage at the cost of recovery relevance.

6.3 Directional Sensitivity to the Risk Weight

Figure 4 presents the results when only the execution-risk penalty is varied. As m increases from 0.5 to 2, the unsafe selection rates are 14.2, 12.4, 10.9, 10.3, 9.8, and 9.3%, respectively; the recovery relevance rates are 82.4, 81.4, 80.2, 79.5, 78.8, and 76.9%, respectively. Across the six reported discrete settings, increasing the risk penalty is consistently associated with a lower unsafe selection rate and lower recovery relevance.

This curve reflects only the directional trade-off at the tested points; behavior between points, robustness under other weight combinations, the optimal multiplier, and uncertainty across runs are all beyond its scope of proof. The value of this analysis lies in showing that multi-objective selection is not a single-point result: once organizational preferences change, the system needs to make explicit which objectives are being traded against one another.

6.4 Comparison between Text-RAG and KG-RAG and Audit of Task Mixing

The aggregated results for Text-RAG and KG-RAG show that the action-type matching rate increases from 77.21% to 79.64%, an improvement of 2.44 percentage points; the mean semantic similarity increases from 82.15% to 82.91%, an improvement of 0.76 percentage points; while evidence accuracy decreases from 60.81% to 60.44%, a reduction of 0.37 percentage points. However, further auditing of the paired experimental records shows that, although the two groups share consistent sample identifiers and reference answers, the 6,813 records simultaneously contain 2,219 action-generation tasks, 2,280 next-state prediction tasks, and 2,314 incident-identification tasks, and the stored model paths are not identical. Because the existing records lack the original evaluation outputs required for task-wise realignment, it is not possible to further determine the specific contribution of each task to the aggregated metrics above. Therefore, these aggregate results can be used to describe the overall differences between Text-RAG and KG-RAG on the mixed-task set,

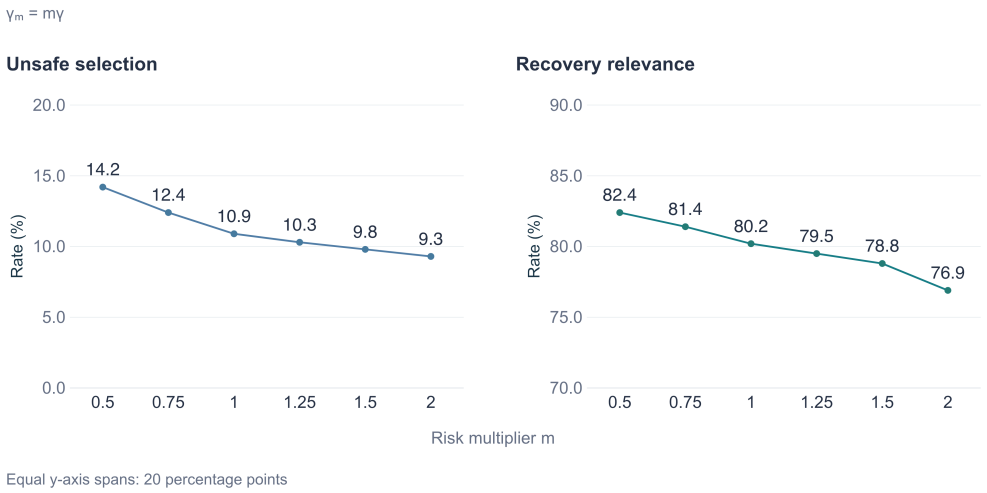


Figure 4: Sensitivity to the execution-risk coefficient $\gamma_m = m\gamma$. Other weights and the candidate pool are fixed; the dashed line marks $m = 1$. The two panels use different vertical ranges with equal spans.

but cannot be directly interpreted as an independent improvement in action-generation capability brought by KG-RAG.

Further analysis of the post-processing acceptance results reveals the boundary between rule checking and action extraction. Across all 6,813 records, Text-RAG accepts 5,710 records (83.81%) and blocks 1,103 records (16.19%); KG-RAG accepts 5,617 records (82.45%) and blocks 1,196 records (17.55%). All blocked records contain stored error-level findings, while all accepted records contain no stored errors; therefore, these results primarily reflect consistency between the acceptance mechanism and the predefined severity rules, rather than constituting empirical evidence of safety on their own. Within the 2,219 action-generation records, Text-RAG and KG-RAG accept 1,854 (83.55%) and 1,826 (82.29%) records, respectively. Among the 365 blocked records under Text-RAG, one is associated with a command syntax error and 364 with policy errors, whereas all 393 blocked records under KG-RAG are triggered by policy errors. At the same time, a large number of accepted actions still carry warning-level findings, indicating that "passing the acceptance rules" is not equivalent to "having no potential issues."in Table 1.

Table 1: Recorded admission for the action-generation subset (2,219 cases per configuration). The last row is conditional on admission; the entries are stored rule decisions, not task accuracy or operational safety.

Outcome	Text-RAG	KG-RAG
Admitted	1,854 (83.55%)	1,826 (82.29%)
Blocked	365 (16.45%)	393 (17.71%)
Admitted with warnings	1,699	1,699
Admitted without findings	155	127
Warning rate among admitted	91.64%	93.04%

Further task-level auditing shows that command errors aggregated across tasks primarily originate from non-action-generation tasks. Among the 663 Text-RAG records containing command syntax errors, 661 are from incident-identification tasks; for KG-RAG, the corresponding number is 731 out of 734. Certain natural-language expressions, such as "restore service" or other narrative fragments, may be identified as commands during extraction and subsequently trigger error judgments based on quotation-related or other syntactic rules. Therefore, command-error statistics on the mixed-task set largely reflect false positives produced when the action extractor is applied to non-action requests, and cannot be directly interpreted as the command-defect rate of action generation itself.

In contrast to the mixed-task audit above, Figure 5 summarizes the integrity checks for the fixed-candidate-pool ranking experiment, which are used to verify whether the two ranking methods are compared over exactly the same candidate set. The aggregate results on the mixed-task set and the corresponding task-level audit details are provided in the appendix.

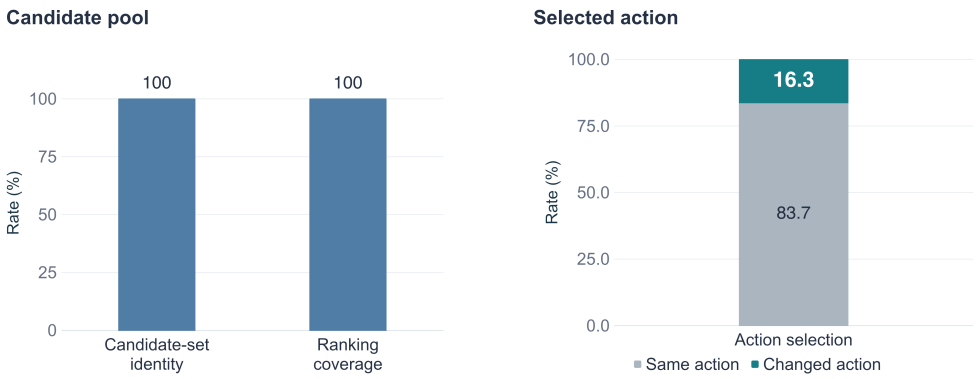


Figure 5: Consistency checks for fixed-pool ranking: candidate-set identity and ranking coverage are both 100%; the two rankers select different actions in 16.3% of cases.

7 Discussion

The three layers of VeriIR-Agent should be interpreted separately. Schema constraints mainly improve validity and completeness because they restrict output structure, but have limited effect on semantic properties such as action-type matching or command validity. Likewise, multi-objective ranking improves risk, impact, and rollback outcomes by reordering a fixed candidate pool rather than generating better candidates. This differs from prior work focused mainly on generation, retrieval grounding, or single-objective recovery ranking (Hammar et al., 2026; Ayala & Bechard, 2024; Edge et al., 2024; Zhao et al., 2024), and suggests that structural compliance, candidate quality, selection, and verification should be evaluated independently rather than collapsed into a single reliability score. Future work should therefore focus on semantic candidate quality, broader verification coverage, and more precise checkers. Our conclusions are limited by aggregated existing data, without repeated-run variance, cross-model or cross-dataset evaluation, or adversarial testing. Moreover, evaluator-defined command validity, rollback, and recovery relevance do not guarantee successful execution, reversibility, or recovery time. Real-world deployment would require sandboxed command validation, versioned evidence and policies, explicit postconditions and rollback outcomes, and authorization checks. Equation equation 3 should thus be viewed as a conditional framework for reducing coverage gaps and false passes, not as an empirical safety certificate.

8 Conclusion

This paper presents VeriIR-Agent, which separates LLM-based incident response into structured generation, risk-aware selection, and verification-quality analysis. Controlled experiments show that schema constraints primarily improve structural compliance—increasing completeness by 29.0 percentage points while improving action-type matching by only 2.4 points—whereas risk-aware ranking reduces unsafe selection by 6.7 points and increases rollback coverage by 10.5 points on a fixed candidate pool, with a 2.9-point decrease in recovery relevance. Together, these results show that representation quality, selection preferences, and verification guarantees capture different aspects of system behavior and should be evaluated separately. The verification decomposition further distinguishes uncovered properties from false passes within covered properties, corresponding to improvements in checker scope and precision. Future work will extend these findings across models, datasets, and adversarial settings, and evaluate command execution and rollback outcomes in isolated environments.

REFERENCES

Y. Abbasi-Yadkori et al. Mitigating LLM hallucinations via conformal abstention, 2024. arXiv preprint arXiv:2405.01563.

O. Ayala and P. Bechard. Reducing hallucination in structured outputs via retrieval-augmented generation, 2024. URL https://aclanthology.org/2024.naacl-industry.19/. in NAACL Industry Track, pp. 228–238.

CAGE. TTCP CAGE challenge 2, 2022. URL https://github.com/cage-challenge/cage-challenge-2. in AAAI Workshop on Artificial Intelligence for Cyber Security.

S. R. Castro, R. Campbell, N. Lau, O. Villalobos, J. Duan, and A. A. Cardenas. Large language models are autonomous cyber defenders, 2025. arXiv preprint arXiv:2505.04843.

E. Debenedetti, J. Zhang, M. Balunović, L. Beurer-Kellner, M. Fischer, and F. Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/97091a5177d8dc64b1da8bf3e1f6fb54-Abstract-Datasets_and_Benchmarks_Track.html. in Advances in Neural Information Processing Systems, vol. 37.

D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, and J. Larson. From local to global: A graph RAG approach to query-focused summarization, 2024. arXiv preprint arXiv:2404.16130.

K. Hammar, T. Alpcan, and E. C. Lupu. Incident response planning using a lightweight large language model with reduced hallucination, 2026. in Network and Distributed System Security Symposium (NDSS).

S. Hays and J. White. Employing LLMs for incident response planning and review, 2024. arXiv preprint arXiv:2403.01271.

D. Kramer, L. Rosique, A. Narotam, E. Bursztein, P. G. Kelley, K. Thomas, and A. Woodruff. Integrating large language models into security incident response, 2025. in Symposium on Usable Privacy and Security (SOUPS).

E. Li, T. Mallick, E. Rose, W. Robertson, A. Oprea, and C. Nita-Rotaru. ACE: A security architecture for LLM-integrated app systems, 2026. in Network and Distributed System Security Symposium (NDSS).

X. Lin et al. IRCopilot: Automated incident response with large language models, 2025. arXiv preprint arXiv:2505.20945.

P. Manakul, A. Liusie, and M. Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models, 2023. in Proceedings of EMNLP, pp. 9004–9017.

A. Nelson, S. Rekhi, M. Souppaya, and K. Scarfone. Incident response recommendations and considerations for cybersecurity risk management: A CSF 2.0 community profile, 2025. National Institute of Standards and Technology, Tech. Rep. NIST SP 800-61 Rev. 3, Apr.

OASIS Open. CACAO security playbooks version 2.0, 2023. URL https://docs.oasis-open.org/cacao/security-playbooks/v2.0/. OASIS Open, Tech. Rep.

D. Schlette, P. Empl, M. Caselli, T. Schreck, and G. Pernul. Do you play it by the books? a study on incident response playbooks and influencing factors, 2024. in IEEE Symposium on Security and Privacy, pp. 3625–3643.

G. Sriramanan et al. LLM-Check: Investigating detection of hallucinations in large language models, 2024. in Advances in Neural Information Processing Systems, vol. 37, pp. 34 188–34 216.

G. Syros, A. Suri, C. Nita-Rotaru, and A. Oprea. SAGA: A security architecture for governing AI agentic systems, 2025. arXiv preprint arXiv:2504.21034.

D. W. Woods, R. Böhme, J. Wolff, and D. Schwarcz. Lessons lost: Incident response in the age of cyber insurance and breach attorneys, 2023. URL https://www.usenix.org/conference/usenixsecurity23/presentation/woods. in 32nd USENIX Security Symposium, pp. 2259–2273.

C. Zhao, G. Agrawal, T. Kumarage, Z. Tan, Y. Deng, Y.-C. Chen, and H. Liu. Ontology-aware RAG for improved question-answering in cybersecurity education, 2024. arXiv preprint arXiv:2412.14191.

A Data, Action Schema, and Experimental Protocol

A.1 Data Composition and Graph Context

The experimental data contain a total of 6,813 records, including 2,219 recovery action-generation records, 2,280 next-state prediction records, and 2,314 incident-identification records. Figure A1 presents the overall distributions of reference action categories and ATT&CK tactic annotations. Among the reference action categories, evidence acquisition (2,649 cases), monitoring (1,862 cases), and containment (1,121 cases) are the most frequent. ATT&CK annotations are counted in a multi-label manner, with Initial Access (3,924 occurrences), Execution (3,355 occurrences), and Exfiltration (2,924 occurrences) being the most frequent tactics; an additional 2,434 records contain no normalized ATT&CK tactic annotations.

Reference action categories and ATT&CK tactics use different counting schemes: the former assigns one reference category to each record, whereas the latter allows a single record to correspond to multiple tactic labels. Therefore, the two types of counts should not be directly compared with the total number of cases. These distributions characterize the task composition and security semantics of the dataset; however, because the data simultaneously contain action generation, state prediction, and incident identification tasks, the corresponding aggregate statistics are not used to independently attribute model performance changes on any single task.

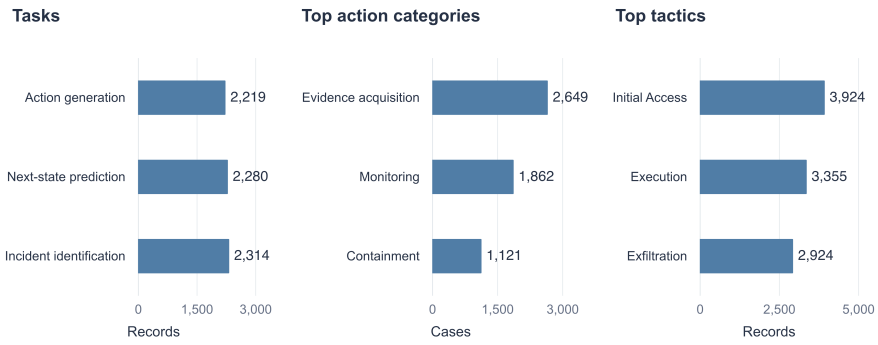


Figure A1: Composition of the 6,813 paired references, retained from the earlier summary. ATT&CK tactic counts are multi-label counts of explicit annotations.

The graph context is constructed following an "incident object–local graph–textual summary" pipeline. The incident instruction is first converted into a structured object; the constructor then connects the incident with IOCs, CVEs, assets, services, attack stages, techniques, mitigation measures, and rules; finally, the two-hop neighborhood of the incident node is extracted. The evaluation records contain 11,666 unique IOCs, 6,813 incident nodes, 1,923 asset identifiers, and 274 CVE identifiers. The local catalogs provide 14 mitigation measures, 13 attack-stage labels, 12 ATT&CK technique identifiers, 10 service labels, and 7 security rules. Each retrieval context contains an average of 45.0 nodes and 44.0 edges. These statistics describe the contexts constructed on a per-case basis rather than the scale of an external production knowledge base. This design allows the generator and checker to refer to the same case context, but it does not imply that the system has access to factually complete or continuously updated enterprise knowledge. Figure A2 presents the graph construction scale and relation patterns.

A.2 Action Schema and Run Records

Table A1 defines the core action schema adopted in this paper. Each action is represented by a set of structured fields that provide a unified interface for subsequent verification, ranking, and approval. The field requirements are not identical across different action types; for example, some actions may not include a specific command. Therefore, field completeness is evaluated according to the corresponding action type and its predefined evaluation requirements. For fields defined as optional,



Entities

Entity type	Count
IOC	11,666
Incident	6,813
Asset	1,923
CVE	274

Catalogs

Catalog type	Count
Mitigations	14
Attack-stage labels	13
ATT&CK technique IDs	12
Service labels	10
Security rules	7

Figure A2: Construction and scale of the incident-centered graph context. Entity counts are unique within type across contexts; graph-size statistics are per case.

the system still checks whether they satisfy the corresponding validation conditions based on the semantics of the action.

Table A1: Core fields of the typed action and their role in the decision chain.

Field	Meaning and use
phase, type	Position in the response lifecycle and controlled action class
target, command	Asset or interface reference, for parsing and allow-list checks
pre, effect	Stated preconditions and intended state change
evidence	Links to logs, IOCs, vulnerabilities, or graph paths
risk, rollback	Support policy checks and multi-objective ranking

To enable reproducible interpretation of a comparison, each run should record at least the case split, checkpoint, prompt and schema versions, evidence context and retrieval budget, number of candidates, random seed, verifier rules, risk weights, and approval policy. If actual execution is evaluated, the target image, environment version, resource constraints, observed side effects, and rollback outcome should also be recorded. Figure A3 illustrates the sequence from evidence freezing to audit export in the deployment design; execution isolation and approval, however, are not directly validated by the aggregated experiments in this paper.

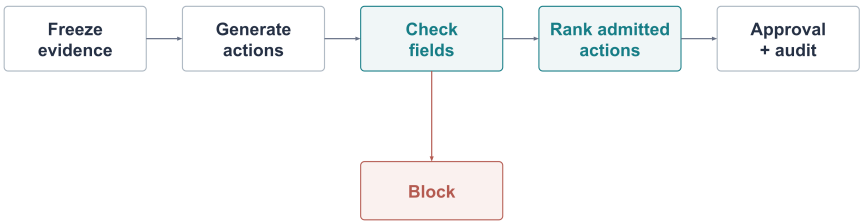


Figure A3: Verification and decision flow in the deployment design. Blocking failures cannot be overridden by generation or ranking; nonblocking warnings remain subject to audit and later policy handling.

B Verification Guarantees and Deployment Boundaries

B.1 Derivation of the Coverage Upper Bound

This section provides the derivation of Equation equation 3. If an issued action is unsafe, then at least one of two types of events must occur: a required property is not covered by any configured

verifier, namely C_{gap}; or at least one covered required property is incorrectly allowed to pass by the corresponding verifier, namely some $FN_k \cap C_k$. Therefore,

$$A_u \subseteq C_{\text{gap}} \cup \bigcup_k (FN_k \cap C_k),$$

and Equation equation 3 follows immediately from the union bound. This upper bound does not require the individual terms to be independent and may therefore be relatively loose in practice; its purpose is not to provide an exact numerical value in the absence of measurements, but to avoid implicitly treating unobserved coverage gaps as zero.

When all mandatory verifiers are reliable within their coverage scopes, the second term becomes zero, and Proposition 1 shows that the acceptance rule can exclude violations within the covered scope. If a critical property is treated as a warning rather than a blocking condition, or if a check is not performed because of missing information, it must be reconsidered whether that property belongs to the "mandatory coverage" set. Representing unknown states through abstention, escalation to human review, or restricted approval provides practical ways to make this logic operational in engineering settings.

B.2 Deployment Workflow and Execution Boundaries

Figure B1 describes the offline preparation and online planning stages of the system design. In the offline stage, local catalogs, the adapted model, and policy rules can be prepared; in the online stage, the normalized incident and two-hop graph context are provided to the generator, followed by field-level checking and multi-objective ranking. The versioned enterprise knowledge base, cross-department authorization, and complete approval chain shown in the figure are production-oriented extensions, and their usability cannot be inferred from the offline recommendation evaluation conducted in this paper.

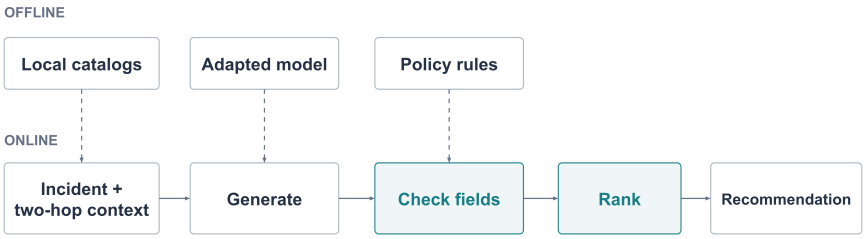


Figure B1: Offline preparation and online planning in the system design. The evaluated path ends at recommendations, findings, and ranking; enterprise knowledge synchronization and execution approval require separate validation.

To support the traceability of response decisions, this paper organizes the deployment-oriented planning process into several sequential stages. First, the case configuration, evidence context, and policy version used for the current run are determined; candidate actions are then generated and subjected to the necessary field-level checks; candidates that pass these checks proceed to multi-objective ranking, with corresponding approval requirements determined according to action risk and impact; finally, the configuration, checking findings, ranking results, and audit information associated with the current decision are retained. If the incident context changes, the system can rerun the planning process based on the updated evidence while preserving the relationships between different context versions. Figure B2 illustrates the logical ordering of these stages, with the purpose of showing the processing relationship from evidence preparation to response decision-making rather than indicating that this paper has evaluated execution processes or recovery time in a real-world environment.

B.3 Relational Schema and Verifier Coordination

The incident-centric graph connects "what is observed" with "what is recommended." Figure B4 illustrates the local relational schema: IOCs, CVEs, assets, services, and attack stages provide trace-

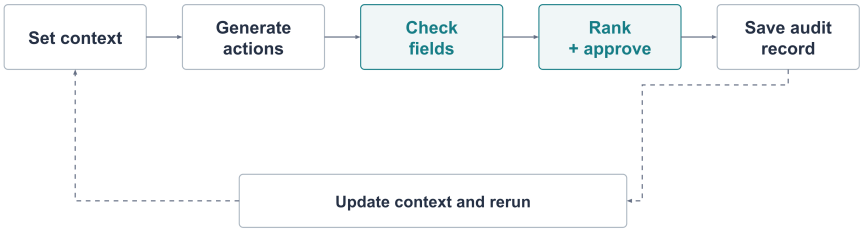


Figure B2: Illustrative response timeline. Validation and risk assessment precede any operational execution; the body reports recommendations and ranking, not measured recovery duration along this timeline.

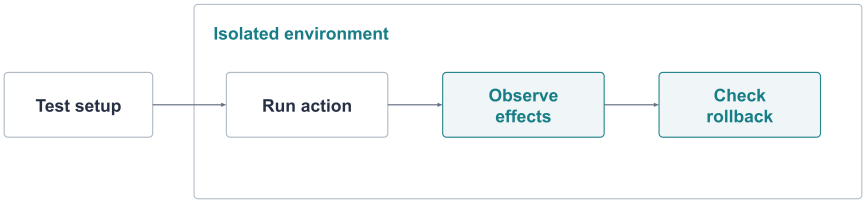


Figure B3: Design boundary of the isolated evaluation environment. Disposable replicas enable execution-based checks, but this paper does not report production-equivalent execution or rollback success.

able anchors for action targets, evidence, or preconditions, while ATT&CK techniques, mitigation measures, and rules provide references for consistency and policy checking. This relational structure does not replace asset version information, source timestamps, or enterprise authorization states; these missing elements are precisely where coverage gaps may arise.

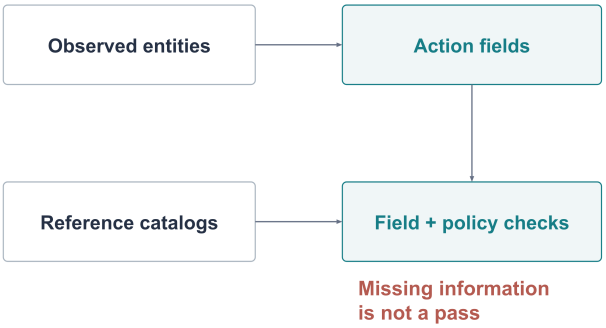


Figure B4: Incident-centered relation pattern. The evaluated prototype instantiates a bounded subset from case instructions and local catalogs; source freshness, version provenance, and enterprise asset mappings remain to be validated.

Verifier coordination follows the sequence of "generation does not adjudicate, verification may block, and ranking cannot override blocking." For example, a command parser can determine whether the syntax is parseable, policy rules can check explicitly prohibited patterns, and a graph-consistency checker can identify mismatches in attack stages or techniques; a pass from each verifier should be accompanied by a declaration of its coverage scope. If the context does not contain the target version, the conclusion of a CVE check should be insufficient information rather than ver-

sion safety; if a command is outside the permitted execution environment, successful parsing should likewise not be elevated to executability. This form of record keeping maps the two types of improvements in Equation equation 3 to concrete engineering actions: enriching information sources and rule coverage to reduce C_{gap}, and calibrating checkers using labeled cases and execution feedback to reduce FN_k.

C Source Audit and Supplementary Statistics

C.1 Data Consistency and Rule-Based Result Audit

The experimental records for Text-RAG and KG-RAG each contain 6,813 unique case identifiers, and the two datasets can be matched one-to-one in terms of case identifiers, reference answers, and saved graph contexts. However, the available records do not provide sufficient information to confirm that the two configurations used exactly the same model checkpoint. Therefore, the corresponding comparison can only be treated as a paired-result analysis and cannot establish strict model-level causal attribution.

Table C1 summarizes the acceptance results after rule-based checking by task type. Because action generation, next-state prediction, and incident identification are all processed through the same action-extraction and post-processing pipeline, command and policy errors aggregated across tasks are influenced by both task type and the extraction process. Therefore, these error counts cannot be directly interpreted as the defect rate of action generation itself.

Under the Text-RAG configuration, 663 command syntax errors and 443 policy errors are recorded, with 3 records containing both types of errors; under the KG-RAG configuration, the corresponding numbers are 734 and 463, with 1 record containing both types. These statistics reflect failures triggered by the predefined rules in the stored records and cannot be used to infer that accepted actions contain no undetected issues or that their false-pass rate is zero.

Table C1: Admission counts by requested task in the earlier exports; the case counts are identical in both configurations.

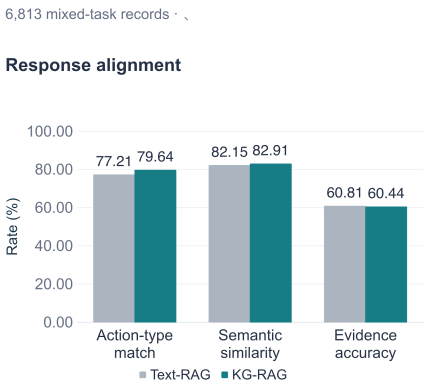
Requested task	Cases	Base admitted	Graph admitted
Recovery-action generation	2,219	1,854	1,826
Next-state prediction	2,280	2,205	2,211
Incident identification	2,314	1,651	1,580

Supplementary statistics further show that different checking metrics use different counting units. Existing aggregate records report 3,052 command-related findings among 4,182 command instances, 3,766 completeness findings among 6,813 action records, and 440 policy errors. In the corresponding paired experimental records, a total of 4,158 command strings were extracted, and 443 actions containing policy errors were identified; no separate completeness-finding records were retained. When counted by command-syntax checking events, a total of 3,041 findings were recorded, including 2,174 warnings and 867 errors, affecting 1,472 records.

It should be noted that the numbers of finding events, command strings, and affected records use different statistical denominators and therefore cannot be treated as equivalent or substituted for one another. In addition, because the existing records do not retain the original evaluator outputs required for completeness assessment, the corresponding completeness statistics cannot be independently reproduced from the current data. Figure C1 summarizes these supplementary statistics to illustrate the different counting schemes and their relationships, rather than treating them as results fully equivalent to the current rule-based audit.

C.2 Supplementary Experimental Results

Table C2 summarizes the main quantitative results reported in the main text for comparing output constraints and risk-aware ranking, including metrics related to structural compliance, action prop-



Summary

Measure and unit	Value
Command findings / instances	3,052 / 4,182
Completeness findings / actions	3,766 / 6,813
Actions with policy errors	440
Extracted command strings	4,158
Actions with policy errors	443
Command finding events	3,041
Warnings / errors	2,174 / 867
Records with command findings	1,472
Completeness findings	Not retained

Figure C1: Earlier response-alignment and diagnostic summary, retained for source reconciliation. In the right panel the command rates use command instances as the denominator and the other rates use actions; these values are not independently reproduced by the current export audit.

erties, and ranking outcomes. All values correspond to the experimental settings reported in the main text and are presented as descriptive statistics.

In addition, candidate-pool consistency and ranking coverage are used to verify whether different ranking methods are compared over the same candidate set. Their purpose is to confirm the integrity of the experimental comparison rather than to measure the performance of the ranking methods themselves. The corresponding results are shown in Figure 5.

Table C2: Key values from the existing controlled experiments (%).

Metric	Free-form	Prompt-structured	Constrained
Schema validity	46.2	64.7	98.0
Completeness	61.8	73.5	90.8
Action-type match	72.7	74.0	75.1
Command validity	80.9	82.0	83.2

Metric	Recovery-only	Risk-aware
Unsafe selection	17.6	10.9
High-impact selection	43.8	35.7
Rollback coverage	21.3	31.8
Recovery relevance	83.1	80.2