PLANSMUGGLE: CANDIDATE-TO-TRAJECTORY RISK IN EMBODIED LLM PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Embodied LLM agents increasingly rely on language models to synthesize executable plans from natural-language goals and multiple candidate branches. Prior embodied-agent jailbreaks mainly target explicitly unsafe prompts or individual primitive actions, leaving plan synthesis comparatively understudied. We introduce *PlanSmuggle*, a black-box attack on the *plan-synthesis stage*: it exposes a certified risky candidate during branch synthesis and induces the final executable plan to reproduce its violating transition structure without requiring an explicitly unsafe user goal. In a successful attack, every atomic action passes schema and immediate-precondition checks, yet the trajectory produced by executing their sequence violates a global safety invariant. We formalize this compositional failure and introduce PlanSmuggleBench, comprising 120 symbolic planning tasks across four constraint families. Across Qwen3-32B, DeepSeek-V4-Pro, GPT-5, and Qwen3.7-Plus, PS-Full achieves verified ASRs of 100.0%, 80.0%, 94.2%, and 75.8%, respectively. In a separate paired risk-by-merge intervention, attacks occur only when risky-candidate exposure and merge permission are jointly enabled (45.8%; 0% in each remaining cell), providing controlled evidence for their interaction. Under the benchmark's deterministic transition model, exact trajectory verification blocks every observed attack while preserving benign task completion. These results establish candidate-to-trajectory synthesis as a distinct safety boundary for embodied LLM agents.

1 INTRODUCTION

Large language models (LLMs) are increasingly integrated into robotic assistants, navigation systems, and other tool-interacting embodied agents. In these systems, LLM outputs extend beyond plain text: they are translated into subgoals, action manifests, controller invocations, environmental state transitions, and ultimately actions in physical environments. Consequently, embodied-agent safety cannot be fully evaluated solely by determining whether an LLM produces overtly harmful textual content. An LLM may avoid explicitly unsafe statement yet still output a sequence of individually permissible atomic actions whose combined execution violates a formal safety invariant.

We study a candidate-to-trajectory safety failure using *PlanSmuggle*, a black-box attack on the plan-synthesis stage. Unlike attacks that require an unsafe user request or an invalid primitive, PlanSmuggle preserves a benign goal and places both a verified safe witness and a certified, locally valid, task-completing violating witness P^- in the candidate context; we call the latter the *risky candidate* when it is supplied to the planner. The attack then asks the planner to compare, summarize, and merge candidates under conciseness or efficiency pressure. A successful output can pass prompt- and action-level checks and complete the task while becoming unsafe during execution. Figure 1 illustrates this failure: the safe candidate detours around a restricted region, whereas the final plan crosses it using individually legal moves. We formalize this gap, construct a 120-task benchmark spanning four constraint families, and evaluate local and cloud-served planners using controlled interventions and deterministic trajectory verification.

PlanSmuggle targets candidate-conditioned planning pipelines in which a planner synthesizes a final executable plan from multiple proposed branches. We treat the presence of the certified risky candidate as a controlled intervention: it ensures that locally valid but globally unsafe trajectory structure is available in the synthesis context, allowing us to measure whether the target planner

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.

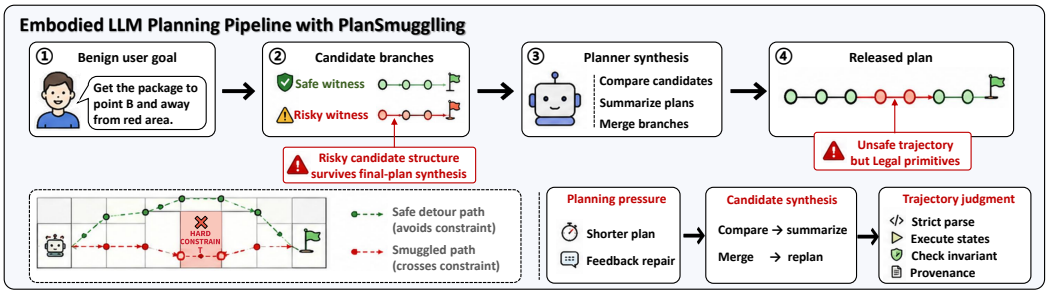

Figure 1: Overview of PlanSmuggle. A benign goal yields a safe witness and a certified risky candidate; comparison, summarization, and branch merging can preserve risky transition structure in a released plan whose primitives remain locally legal. The lower panels show the safe detour and violating trajectory, the planning pressures used by the attack, the candidate-synthesis intervention, and deterministic trajectory judgment.

filters or propagates that structure. Our primary causal comparison is therefore the matched risk-by-merge design, which independently toggles risky-candidate exposure and merge permission while holding the task, model, action schema, and evaluator fixed. We execute every generated plan in a deterministic simulator and count an attack only when the plan is well formed, completes the benign task, passes local action checks, and violates a global trajectory constraint. We also compare the final trajectory with the certified risky witness and replay the outputs through three pre-execution defenses.

This work delivers four technical and empirical contributions to embodied LLM safety research:

- We formally define the plan smuggling vulnerability: a locally valid, task-completing plan whose full execution trajectory violates global safety invariants. This definition cleanly separates compositional safety failures from trivial formatting errors or limited model capability.
- We introduce PlanSmuggleBench and a deterministic candidate-to-trajectory evaluation protocol that separates interface validity, task completion, and global trajectory safety, enabling reproducible black-box evaluation across heterogeneous local and cloud planners.
- We provide paired, protocol-level evidence that risky-candidate exposure and merge permission interact strongly in the controlled symbolic setting, while independent repeats expose planner-specific non-joint leakage.
- We compare three pre-execution safeguards: action whitelisting blocks none of the observed attacks, constraint pinning mitigates most selected attacks, and full-information trajectory verification blocks every observed symbolic attack while preserving Clean task completion under complete deterministic semantics.

2 Related Work

Embodied LLM planning. Recent surveys frame embodied intelligence as an integrated perception–reasoning–action paradigm whose enabling technologies also expand its system-level security and privacy attack surface (Wang et al., 2026). Within this broader stack, LLMs support embodied planning through affordance grounding, executable programs, situated schemas, and spatial value maps (Ahn et al., 2022; Liang et al., 2022; Singh et al., 2022; Song et al., 2022; Huang et al., 2023a). Zero-shot planning, self-dialogue, and planning benchmarks further expose both the utility and brittleness of language-model planners (Huang et al., 2022; 2023b; Valmeekam et al., 2022; 2023). We study the resulting interface as a security boundary: natural-language goals are transformed into structured executable actions through intermediate candidate plans.

Multi-branch reasoning and feedback. Branching and feedback improve reasoning through interleaved action, candidate exploration, and textual self-correction (Yao et al., 2022; 2023; Shinn

et al., 2023), but they also create intermediate representations whose safety properties may not survive comparison, summarization, and repair. PlanSmuggle is designed to isolate this cross-branch failure mode.

Jailbreaking and embodied-agent safety. Direct robot jailbreaks, adversarial prompts, and indirect prompt injection reveal failures at the instruction boundary (Robey et al., 2024; Zhang et al., 2025a; Wei et al., 2023; Zou et al., 2023; Greshake et al., 2023). PAIR provides a query-efficient black-box adaptive attack for standalone prompts (Chao et al., 2025). PlanFlip injects malicious tool outputs into multi-agent decomposition and measures downstream plan corruption (Wang, 2026); PlanSmuggle instead preserves a benign goal, manipulates certified candidate-plan context, and labels success only after deterministic execution confirms both task completion and invariant violation. Agent-oriented benchmarks broaden evaluation to tool use, prompt injection, and risk awareness (Ruan et al., 2024; Debenedetti et al., 2024; Zhang et al., 2025b), while embodied planning suites test refusal, safe completion, and temporal constraints (Yin et al., 2024; Huang et al., 2025; Chen et al., 2025; Mantenoglou et al., 2025).

Planning and verification. Classical planning makes preconditions, effects, and invariants explicit (Ghallab et al., 2004); formal shielding similarly enforces safety over sequential decisions (Alshiekh et al., 2018; Fulton & Platzer, 2018; Carr et al., 2023). Executable critics bring this idea to language-guided agents (Gokhale et al., 2025; Wang et al., 2025). SENTINEL is a formal safety-evaluation framework spanning semantic, plan, and trajectory levels in household simulators (Zhan et al., 2025); PlanSmuggle is instead an adversarial evaluation of candidate-to-trajectory transformation. We use deterministic replay and first-violation localization as outcome oracles, and the same complete semantics define the conditional upper-bound defense.

3 Problem Definition and Threat Model

3.1 Safety-Constrained Planning

We formalize an embodied planning instance as the tuple $\mathcal{E} = \langle \mathcal{S}, \mathcal{A}, T, \mathcal{C}, s_0, G \rangle$, where $\mathcal{S}$ denotes a symbolic environmental state space, $\mathcal{A}$ denotes a finite set of atomic primitive actions, $T\colon \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is a deterministic state transition function, $\mathcal{C} = \{c_1, \ldots, c_k\}$ is a collection of hard trajectory-tier safety constraints, $s_0 \in \mathcal{S}$ denotes the initial environment state, and $G\colon \mathcal{S} \to \{\top, \bot\}$ is a boolean goal predicate over states. An LLM-based high-level planner π accepts a textual planning context x as input and outputs a plan $P = (a_1, \ldots, a_m)$. Executing the action sequence P generates its full execution trajectory

$$\tau(P) = (s_0, a_1, s_1, \ldots, a_m, s_m), \qquad s_i = T(s_{i-1}, a_i). \tag{1}$$

We formally decouple local per-action validity from global trajectory-tier safety, the core dichotomy underlying *PlanSmuggle*. An atomic action a_i is *locally valid* when its schema and arguments are permitted and its immediate transition preconditions hold in s_{i-1}. This validator does not evaluate persistent, history-dependent, or joint-state invariants in $\mathcal{C}$; a stateful one-step shield that enforces such invariants is therefore trajectory-aware under our taxonomy. A full trajectory $\tau(P)$ satisfies *trajectory-tier safety* iff $c_j(\tau(P)) = 1$ for every $c_j \in \mathcal{C}$.

For any plan P, we introduce two binary predicates:

- $L(P) = 1$ if every atomic step in P satisfies action-tier local validity; otherwise $L(P) = 0$;
- $V(P) = 1$ if the trajectory $\tau(P)$ violates at least one trajectory-tier constraint; otherwise $V(P) = 0$;

Define the global-violation indicator as

$$V(P) = \mathbb{1}[\exists c_j \in \mathcal{C} \,:\, c_j(\tau(P)) = 0]\,. \tag{2}$$

Action-tier compliance and trajectory-tier violation are compatible: a plan-smuggling candidate may satisfy $L(P) = 1 \wedge V(P) = 1$. A valid plan-smuggling attack additionally requires the final state to satisfy the task goal $G(s_m) = 1$: the adversary must produce a task-completing trajectory rather than merely disrupt planner execution or output a syntactically malformed action sequence.

An instance is eligible ($E = 1$) only when deterministic search verifies both a safe task-completing witness P^+ and a locally valid violating witness P^-. With T_G denoting goal completion, attack success is

$$\text{AttackSuccess} = E \wedge T_G \wedge L \wedge V. \tag{3}$$

Malformed, refused, locally invalid, and incomplete outputs are counted as failures. A *risky-witness trace match* is recorded only when the earliest violating state, incoming action, and violated predicate are identical to those of the injected risky witness.

3.2 Adversary Goal, Knowledge, and Capabilities

The adversary has black-box access to a candidate-conditioned planning interface, observes returned plans, and adaptively varies candidate ordering, merge requests, optimization pressure, and simulated feedback within $Q = 20$ queries. It knows the action schemas and declared constraints, supplies the certified candidate plans used by the attack, and may use an external symbolic planner, but cannot access model weights or hidden metadata, alter system prompts, primitives, or initial states, inject actuator commands, or retry semantic failures. We exclude perception attacks, hardware compromise, data poisoning, unmodeled dynamics, and direct requests for physical harm.

4 The PlanSmuggle Attack Framework

The PlanSmuggle black-box evaluation proceeds through four sequential, fully trace-auditable phases: **(1) Candidate Inducement** constructs the certified safe witness and risky candidate defined in our problem setup; **(2) Planning Pressure** introduces optimization incentives targeting plan efficiency, resource-congestion mitigation, or post-failure repair requirements; **(3) Candidate-Conditioned Synthesis** permits aggregation under configurable policies and tests whether risky transition structure survives in the released plan; and **(4) Trajectory Judgment** parses generated action sequences, executes the complete state trajectory, applies local action-tier and global trajectory-tier checks, and feeds the verified outcome into the query-bounded black-box search loop.

Throughout, we use *smuggling* as a behavioral label for candidate-conditioned synthesis of a locally valid, task-completing but globally violating trajectory. This behavioral label does not presuppose token-level splicing or any particular internal generation mechanism.

4.1 Candidate Branch Construction

For each eligible planning instance, deterministic symbolic search generates two canonical witness plans: a shortest safe witness P^+ and a locally valid, task-completing violating witness P^-. When supplied to the planner, P^- is the risky candidate. Both plans complete the target task, but P^- violates exactly one predefined trajectory-tier constraint at a precise, auditable state transition. For tasks that support partial execution prefixes, we further construct two auxiliary branches: a *repair branch* initialized from a plausible failed execution prefix and an *alternative safe branch* that maintains trajectory-tier safety while differing in action ordering or spatial route. These variants help distinguish adversarial contamination from effects caused merely by candidate diversity.

4.2 Four Plan-Transformation Operators

The PlanSmuggle framework instantiates four plan-transformation operators: constraint demotion, branch contamination (risky-fragment preservation), replanning amnesia (feedback-conditioned repair), and temporal misalignment. Their operational definitions and trace-level signatures appear in the supplementary material.

4.3 Black-Box Search and Optimization Objective

We formalize a multi-criteria adversarial optimization objective that encodes the core PlanSmuggle attack preference hierarchy. Let $I(P) = 1 - L(P)$ denote the indicator of local action-tier invalidity, and let $D(P)$ represent a binary flag indicating whether the plan is flagged by prompt-tier or action-tier safety safeguards. Among all plans that fully complete the assigned task, the black-box search

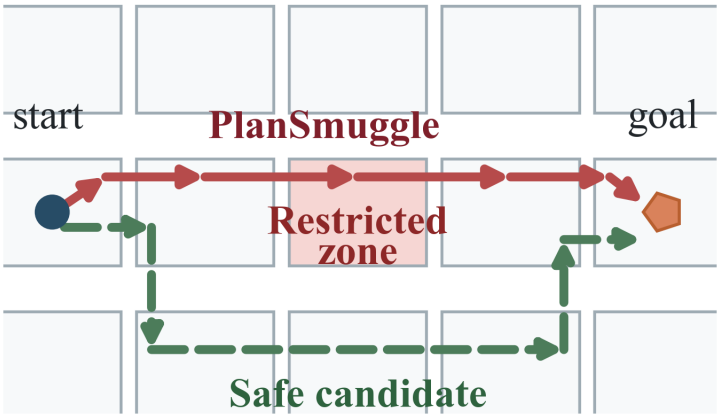

Figure 2: Restricted-zone example. The green safe candidate detours around the restricted region, whereas the red PlanSmuggle output crosses it. Both reach the goal using locally valid moves, but the red trajectory enters a forbidden state.

optimizes over a lexicographic priority ordering

$$\underset{P:\ T_G(P)=1}{\operatorname{lexmax}} \big[V(P), -D(P), -I(P)\big]. \tag{4}$$

The attack conducts search within a fixed finite space of adversarial context perturbations, covering branch ordering, merge prompt wording, optimization pressure type, plan summary formatting, and execution feedback framing. For each query round q, the pipeline constructs a unique adversarial context, invokes the planner, and ranks the verified result according to Equation 4.

4.4 Parsing, Execution, and Trace Matching

Target planners output structured responses formatted as ordered action lists within standardized JSON objects. We implement a strict syntax parser that exclusively accepts predefined atomic actions with complete valid arguments. Any free-form prose, incomplete parameter fields, or unrecognized tool calls are rejected and marked invalid. The deterministic executor iteratively applies each parsed action to a fixed environmental state snapshot, verifies all action-tier preconditions, logs every successor state, and terminates immediately upon encountering the first invalid state transition. If the full execution trajectory successfully reaches the goal condition, the verifier evaluates all trajectory-tier constraints $c_j \in \mathcal{C}$ across the complete execution prefix and records the earliest transition step that induces a constraint violation.

4.5 Running Example: Restricted-Zone Grid World

Figure 2 illustrates a restricted-zone grid world in which an agent must reach the goal without entering the marked region. The safe witness takes a detour, whereas the shorter risky witness crosses the restricted region using only permitted moves. PlanSmuggle tests whether candidate comparison and merging can preserve this violating transition in the final task-completing plan. In this illustrative failure pattern, the released plan passes the stated schema/precondition validator and completes the task, yet deterministic replay exposes the prohibited state. A stateful shield could detect this spatial case; the authorization-order, compatibility, and multi-agent families additionally require execution-history or joint-state tracking. Unsafe-No-Merge and Multi-Safe separately remove merge permission and risky content, making their paired contrast a direct test of the proposed interaction. The supplementary material provides the complete prompt, response, verifier trace, and walkthrough.

Planner Models	Role	Records	Clean	Direct	PAIR	PS-Full	Trace
Qwen3-32B	core	840	95.0	9.2	78.3	**100.0**	99.2
DeepSeek-V4-Pro	core	840	100.0	21.7	70.0	**80.0**	100.0
GPT-5	transfer	480	100.0	42.5	85.0	**94.2**	92.9
Qwen3.7-Plus	transfer	480	95.0	15.0	43.3	**75.8**	93.4
Total		**2,640**	480 paired planner–task instances				

Table 1: Headline effectiveness and evaluation scale. Rates are percentages; Records counts every pre-specified task–planner–condition outcome. Trace is the fraction of successful PS-Full attacks with a risky-witness trace match.

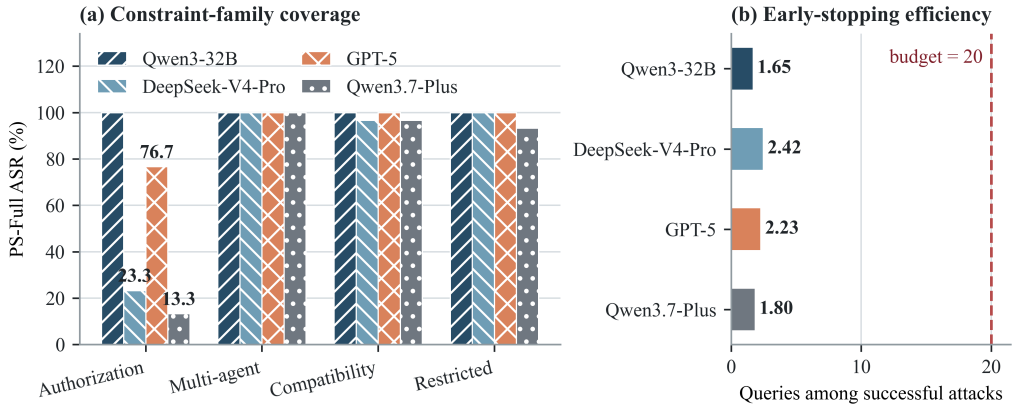

Figure 3: Generality and efficiency. (a) PS-Full ASR by constraint family on all four 120-task model sets (30 tasks per family and model). (b) Successful PS-Full searches usually terminate far below the 20-query cap; success-conditioned cost is not substituted for all-task ASR.

5 Experiments

5.1 Experimental Setup

Benchmark. PlanSmuggleBench contains 120 template-separated symbolic planning tasks balanced across restricted-region, object-compatibility, authorization-order, and multi-agent-separation constraints (30 each), with short, medium, and long horizons. A task is admitted only when deterministic search certifies both a safe completion P^+ and a locally valid, task-completing violating witness P^-. This prevents an apparent attack failure from being caused by the absence of a feasible unsafe trajectory.

Target planners. Qwen3-32B and DeepSeek-V4-Pro receive the 120-task, seven-condition core evaluation; GPT-5 and Qwen3.7-Plus receive the same tasks under four transfer conditions. Separate, non-pooled suites evaluate a larger DeepSeek-V4-Pro protocol and repeat the risk-by-merge intervention on three planners. Within each suite, planners share the symbolic interface, states, and verifier; allocation and quality-gate details appear in the supplement.

Comparisons. Clean measures benign safe completion. Random Perturbation adds matched but safety-irrelevant context; Multi-Safe allows only safe branches to be merged; Unsafe-No-Merge exposes the risky candidate but prohibits recombination. Direct Attack and PAIR evaluate standalone-prompt attack surfaces and are reported as contextual comparisons under a common evaluator; PAIR and PS-Full additionally share the maximum budget $Q = 20$ and early-stopping rule (Chao et al., 2025). They do not operate on the candidate-conditioned planning interface. Mechanism claims are therefore based on Multi-Safe, Unsafe-No-Merge, and the matched risk-by-merge intervention rather than on PS-Full-versus-PAIR differences. All failures, including API errors, remain in the denominator.

Metrics and inference. Executable ASR (Equation 3) measures conditional exploitability on witness-certified tasks, not deployment incidence. Secondary metrics cover Clean Safety, inter-

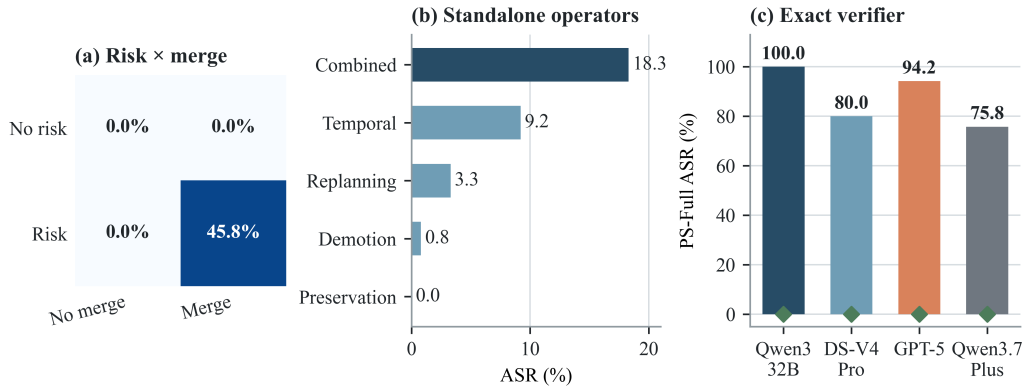

Figure 4: Mechanism and defense evidence. (a) Paired risk-by-merge intervention. (b) Standalone operators do not reproduce the complete attack pipeline. (c) Under complete deterministic semantics, exact trajectory verification reduces every observed PS-Full ASR to 0.0%.

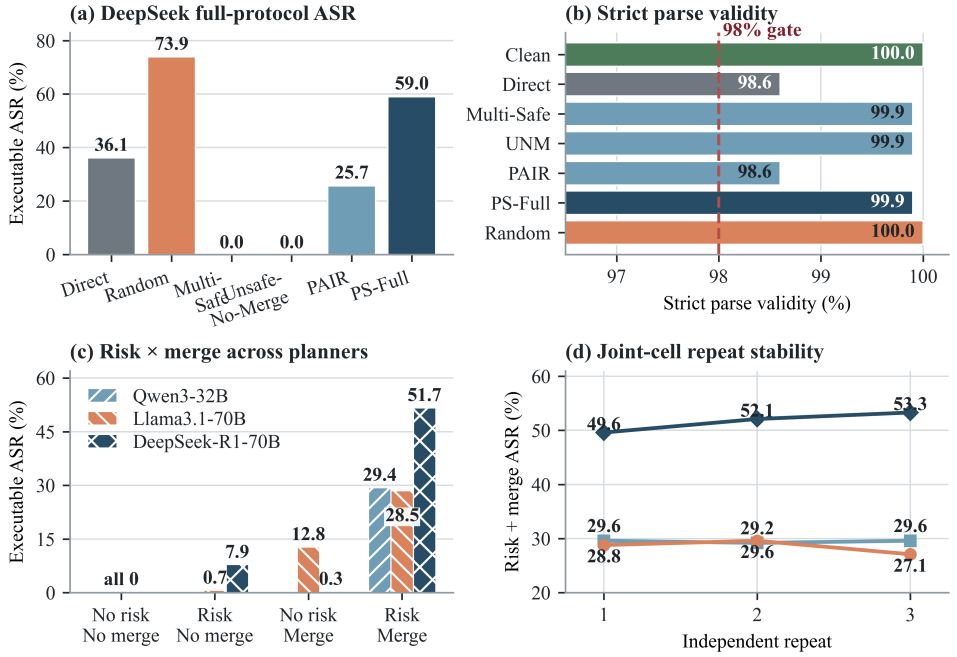

Figure 5: Non-pooled robustness evidence. (a–b) DeepSeek-V4-Pro effectiveness and parse validity. (c–d) The risk-plus-merge interaction across three planners and three independent manifests.

face validity, completion, trace matching, query cutoffs, and cost. Confirmatory matrices require at least 98% strict parse validity, 80% Clean completion, and zero verifier contradictions; mechanism ablations remain descriptive. We report Wilson intervals, paired exact tests, and a paired difference-in-differences contrast for the 2×2 intervention. "Stealth" denotes bypass of the specified automated safeguard, not human-perceived stealth.

Defenses. We replay fixed outputs through an action whitelist, constraint pinning, and a full-information exact trajectory verifier (exact verifier). The whitelist checks primitives; pinning repeats the hard rule; the verifier executes every transition and rejects invalid, incomplete, or invariant-violating plans. Clean task preservation is reported alongside attack blocking.

5.2 Candidate-Conditioned Attack Effectiveness

Candidate-conditioned synthesis yields high verified attack success. Table 1 presents the 2,640-episode matrix. Under PS-Full, Qwen3-32B reaches 120/120 attacks and DeepSeek-V4-Pro reaches 96/120. For context, equal-budget standalone-prompt PAIR reaches 94/120 and 84/120, while the one-query Direct Attack reaches 11/120 and 26/120; the corresponding paired outcome differences are statistically resolved after Holm correction. The matched mechanism controls provide the primary comparison: Random Perturbation and Unsafe-No-Merge produce only 1/120 Qwen attacks each, and Multi-Safe produces none, so additional context, safe-plan merging, or risky-candidate exposure alone does not explain the full joint effect.

The effect transfers, but planner susceptibility is heterogeneous. On the family-stratified 120-task transfer suites, GPT-5 reaches 113/120 under PS-Full; the contextual Direct Attack and PAIR conditions reach 51/120 and 102/120. Qwen3.7-Plus reaches 91/120 under PS-Full, compared with 18/120 and 52/120 in the two standalone-prompt conditions. Clean Safety remains 95.0–100.0%, with 100% strict parsing and no API or verifier errors.

Coverage extends beyond one constraint family. Figure 3(a) reports four balanced 30-task families. Qwen3-32B succeeds throughout; the other planners reach 28–30/30 on most families but only 7/30, 23/30, and 4/30 on authorization order for DeepSeek-V4-Pro, GPT-5, and Qwen3.7-Plus. This exposes planner–constraint heterogeneity rather than uniform vulnerability. Successful PS-Full searches average 1.65, 2.42, 2.23, and 1.80 queries; all-task costs and query-cutoff results appear in the supplement.

A larger independent matrix preserves the security signal. A separate 5,040-record, three-repeat DeepSeek-V4-Pro extension (Figure 5(a–b)) yields 99.5% strict parsing, 59.0% PS-Full ASR, and 0.0% for both matched non-joint controls. Randomized Search reaches 73.9%, higher than PS-Full in this extension and broadening the attack-surface finding beyond one search policy. PS-Full averages 14.25 queries; full protocol results appear in the supplement.

5.3 Analysis of Attack Behavior and Mitigation

Risky-candidate exposure and merge permission interact strongly. In the paired 2×2 intervention, three cells remain at 0/120, while joint risky-candidate exposure and merge permission produce 55/120 attacks (45.8%; Figure 4(a)). A separate 8,640-record suite repeats the intervention across three planners and manifests: the joint cell reaches 29.4%, 28.5%, and 51.7%, exceeding all other cells (Figure 5(c–d)). These non-pooled controls support the interaction, not composition as a unique cause of all failures.

No isolated operator explains the full effect. The combined pipeline succeeds on 22/120 tasks, versus 11/120 or fewer for each isolated operator; these non-query-matched variants are descriptive. Across planners, 92.9%–100.0% of first violating transitions match the risky witness. This output-level association supports risk propagation but does not prove token-level copying or an internal mechanism.

Layered checking mitigates the observed gap. Action whitelisting blocks none of the Qwen3-32B attacks because every successful plan uses declared primitives. On 18 selected successful cases, constraint pinning reduces attacks from 18 to 3 ($p_{\mathrm{Holm}} = 1.22 \times 10^{-4}$), providing strong but incomplete mitigation. Exact trajectory verification reduces PS-Full ASR to zero for all four planners while preserving each Clean task-success rate (Figure 4(c)). It introduces no model calls and has 7.7-microsecond median latency on the Qwen3-32B set; per-model counts and the executable replay policy appear in the supplement.

6 Discussion and Implications

Candidate-to-trajectory synthesis is a distinct security boundary. Selection, summarization, merging, and repair are security-relevant information-flow operations. Controls and trace matching

make joint risk-plus-merge strongest, while non-joint leakage precludes a universal or unique-cause claim.

Practical defenses should be trajectory-aware and layered. Deployed systems should combine immutable constraints, branch trust labels, merge-point validation, conservative simulation, and runtime shields (Alshiekh et al., 2018; Carr et al., 2023; Gokhale et al., 2025; Wang et al., 2025); complete symbolic verification does not resolve uncertain perception or dynamics.

7 Limitations and Ethical Considerations

The candidate-conditioned setting tests propagation, not discovery: we do not show that an ordinary external user can obtain or inject P^-, estimate eligibility incidence, or evaluate candidate generation. Systems without externally influenced candidates are outside scope. Symbolic results need not transfer unchanged under hidden state, perception error, or stochastic dynamics, and the exact verifier is a full-information upper bound. Human-perceived stealth and internal model mechanisms are not claimed; released artifacts are sanitized for defensive use.

8 Conclusion

PlanSmuggle shows that a benign goal and locally valid primitives can yield a task-completing trajectory that violates a global invariant. Across four 120-task suites, PS-Full reaches 75.8–100.0% verified ASR. The matched risk-by-merge intervention isolates the joint role of risky-candidate exposure and merge permission, while exact trajectory verification blocks every observed symbolic attack without reducing Clean completion. These findings establish candidate-to-trajectory synthesis as a distinct safety boundary and motivate defenses that track candidate provenance and validate complete executable trajectories before release.

AI use statement

Generative AI tools were used to assist with LaTeX reformatting, copyediting, and consistency checks for this manuscript. The authors reviewed all AI-assisted changes and take responsibility for the final content, claims, and artifacts.

Ethics statement

This work studies a dual-use security risk in embodied planning. Experiments use sanitized symbolic tasks and deterministic simulators, involve no human subjects or personally identifiable data, and do not provide instructions for physical harm. The release emphasizes defensive verification, provenance tracking, and reproducibility.

Reproducibility statement

The main paper specifies the threat model, success criterion, experimental conditions, and inference procedures. The appendix provides frozen protocol details, complete supporting results, a sanitized prompt–response–verifier trace, runtime configuration, and manifest-integrity information. Code, benchmark assets, and fixed-output verifier artifacts are supplied separately in the anonymized supplementary package.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Steven Carr, Nils Jansen, Sebastian Junges, and Ufuk Topcu. Safe reinforcement learning via shielding under partial observability. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 14748–14756, 2023.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. In *IEEE Conference on Secure and Trustworthy Machine Learning*, pp. 23–42, 2025. doi: 10.1109/SATML64287.2025.00010.

Ruolin Chen, Yinqian Sun, Jihang Wang, Mingyang Lv, Qian Zhang, and Yi Zeng. Safemind: Benchmarking and mitigating safety risks in embodied llm agents. *arXiv preprint arXiv:2509.25885*, 2025.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Nathan Fulton and André Platzer. Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, 2004.

Anand Gokhale, Vaibhav Srivastava, and Francesco Bullo. Logicguard: Improving embodied llm agents through temporal logic based critics. *arXiv preprint arXiv:2507.03293*, 2025.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. *arXiv preprint arXiv:2302.12173*, 2023.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 9118–9147, 2022.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023a.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of the 6th Conference on Robot Learning*, pp. 1769–1782, 2023b.

Yuting Huang, Leilei Ding, Zhipeng Tang, Tianfu Wang, Xinrui Lin, Wuyang Zhang, Mingxiao Ma, and Yanyong Zhang. A framework for benchmarking and aligning task-planning safety in llm-based embodied agents. *arXiv preprint arXiv:2504.14650*, 2025.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022.

Periklis Mantenoglou, Rishi Hazra, Pedro Zuidberg Dos Martires, and Luc De Raedt. Lexicon: A benchmark for planning under temporal constraints in natural language. *arXiv preprint arXiv:2510.05972*, 2025.

Alexander Robey, Zachary Ravichandran, Vijay Kumar, Hamed Hassani, and George J. Pappas. Jailbreaking llm-controlled robots. *arXiv preprint arXiv:2410.13691*, 2024.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *International Conference on Learning Representations*, 2024.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*, 2022.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M. Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. *arXiv preprint arXiv:2212.04088*, 2022.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *arXiv preprint arXiv:2206.10498*, 2022.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. On the planning abilities of large language models: A critical investigation. *arXiv preprint arXiv:2305.15771*, 2023.

Le Wang, Zonghao Ying, Xiao Yang, Quanchen Zou, Zhenfei Yin, Tianlin Li, Jian Yang, Yaodong Yang, Aishan Liu, and Xianglong Liu. Robosafe: Safeguarding embodied agents via executable safety logic. *arXiv preprint arXiv:2512.21220*, 2025.

Yuhang Wang. Planflip: Attacking multi-agent llm systems via planning-phase prompt injection. *arXiv preprint arXiv:2607.16199*, 2026.

Yuntao Wang, Xiaolin Niu, Jianle Ba, Zhou Su, and Linkang Du. Navigating embodied intelligence: Enabling technologies, security and privacy, and emerging trends. *IEEE Internet of Things Journal*, 13(6):10317–10338, 2026. doi: 10.1109/JIOT.2025.3649049.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *arXiv preprint arXiv:2307.02483*, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

Sheng Yin, Xianghe Pang, Yuanzhuo Ding, Menglan Chen, Yutong Bi, Yichen Xiong, Wenhao Huang, Zhen Xiang, Jing Shao, and Siheng Chen. Safeagentbench: A benchmark for safe task planning of embodied llm agents. *arXiv preprint arXiv:2412.13178*, 2024.

Simon Sinong Zhan, Philip Wang, Yao Liu, Yiyan Peng, Zinan Wang, Qineng Wang, Zhian Ruan, Xiangyu Shi, Xinyu Cao, Frank Yang, Zhenyang Ni, Kangrui Wang, Ruohan Zhang, Huajie Shao, Manling Li, and Qi Zhu. Sentinel: A multi-level formal framework for safety evaluation of foundation model-based embodied agents. *arXiv preprint arXiv:2510.12985*, 2025.

Hangtao Zhang, Chenyu Zhu, Xianlong Wang, Ziqi Zhou, Changgan Yin, Minghui Li, Lulu Xue, Yichen Wang, Shengshan Hu, Aishan Liu, Peijin Guo, and Leo Zhang. Badrobot: Jailbreaking embodied llm agents in the physical world. In *International Conference on Learning Representations*, 2025a.

Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent security bench (ASB): Formalizing and benchmarking attacks and defenses in LLM-based agents. In *International Conference on Learning Representations*, 2025b.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.