SINGED: CORRECT OUTPUTS DO NOT CERTIFY SAFE EXECUTION IN LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-using language-model agents select and execute third-party artifacts. Different implementations can return the requested output while producing hidden execution effects that task-, attack-, or choice-based evaluations may miss. We study *functional counterfeits*: implementations that match benign alternatives on the requested output but add an effect forbidden by the task contract. We introduce SINGED (**S**ource **I**ntegrity and the **N**onidentifiability **G**ap in **E**xecution **D**ecisions for LLM Agents), a controlled benchmark covering five primary and two held-out task families. It varies displayed rank, evidence depth, decision policy, model release, and agent configuration, while task and process oracles verify the artifact and execution path. Across 7,549 audited trials, the randomized-rank study finds counterfeit execution in 45% (27/60) of rank-one trials and none at later ranks. Cross-candidate comparison eliminates shallow failures and reduces layered failures from 15.7% to 4.2%, but leaves dependency failures; its benefit is uncertain on unseen effects and public-package structures. Moreover, seven releases with no counterfeit executions when benign alternatives are available execute the counterfeit in 55/175 single-source cells after alternatives are removed. SINGED thus exposes a rank-, evidence-, and choice-sensitive outcome-to-execution gap: evaluation must connect correct outputs to execution paths[1].

1 INTRODUCTION

Language-model agents extend language models from generating responses to acting in an environment. They interpret observations, select tools or resources, execute actions, and use feedback to continue or terminate. This loop supports reasoning and acting (Yao et al., 2023), API use (Schick et al., 2023), and selection from large tool registries (Patil et al., 2024; 2025). Coding runtimes operationalize the loop through persistent state, tool interfaces, and termination logic (OpenAI, 2026a; Earendil Works, 2026). Interactive benchmarks evaluate execution trajectories across websites, computers, repositories, and stateful services (Zhou et al., 2024; Mialon et al., 2024).

This shift changes what evaluation must certify. An agent may return a correctly redacted file while the selected implementation also transmits the unredacted input, a behavior absent from the artifact. The Privacy Filter incident illustrates this risk: OpenAI released `openai/privacy-filter` for local personal-information detection and redaction, while another Hugging Face repository copied its model card and functionality but loaded an information stealer before its removal (OpenAI, 2026c; HiddenLayer Research Team, 2026). Neither a malicious request nor an adversarial instruction was required, only a plausible compromised source. This exposes a source-selection problem: *When implementations advertise the same function and produce the same output, which source does an agent execute, and how do rank, inspectable evidence, and decision policy change that choice?*

This problem connects supply-chain security with agent security. Supply-chain research documents malicious packages, dependency confusion, and model-hub loader risks (Ohm et al., 2020; Ladisa et al., 2023; Casey et al., 2024), while agent-security research studies external instructions, poisoned tool descriptions, memories, skills, and unsafe plans (Greshake et al., 2023; Debenedetti et al., 2024; Liu et al., 2026b;a). Yet existing evaluations do not jointly hold the benign request and expected output fixed while varying the implementation, randomizing its displayed rank, and recording process

[1] Our code is available at `https://anonymous.4open.science/r/SINGED_code-22BE`.

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.

Table 1: Representative evaluation designs at the agent–artifact boundary.

Citation	Evaluation setting	Intervention	Matched output	Endpoint	Random rank
Greshake et al. (2023)	Indirect prompt injection	External instruction	×	Attack success	×
Shi et al. (2025)	Tool metadata poisoning	Tool description	×	Tool selection	×
Hu et al. (2026)	Malicious tool generation	Implementation	×	Attack success	×
Sneh et al. (2025)	Tool-surface optimization	Tool interface	×	Tool selection	×
Qu et al. (2026)	Agent-skill supply chain	Skill code	×	Attack success	×
Bito et al. (2025)	Position-bias evaluation	List order	N/A	Recommendation	✓
This work	**SINGED**	Implementation + order	✓	**Output + events**	✓

effects (Table 1). Consequently, an instruction-free, inspectable source can pass the task oracle while producing forbidden behavior that output-only and inspection-only evaluation misses.

Motivated by this gap, we introduce SINGED (**S**ource **I**ntegrity and the **N**onidentifiability **G**ap in **E**xecution **D**ecisions for LLM Agents), a controlled evaluation framework inspired by the source ambiguity exposed by the Privacy Filter incident. Each instance pairs an ordinary request with three similarly named implementations: an intended source, a benign community alternative, and a *functional counterfeit*. All advertise the same capability, expose the same interface, and produce the expected artifact, but only the counterfeit adds an effect forbidden by the task contract. We vary the decisive code's dependency depth and apply the construction to document redaction, speech transcription, paper retrieval, archive extraction, and tabular conversion.

To isolate source selection, SINGED varies rank, evidence depth, decision policy, model release, and agent configuration while holding the request, advertised capability, interface, and expected artifact fixed. Candidate code remains inspectable before installation, no adversarial instruction appears in tasks or sources, and role labels remain hidden. As illustrated in Figure 1, a task oracle verifies returned artifacts, while a channel-separated process oracle records the executed implementation and its observable effects inside a network-isolated sandbox. This design separates output correctness from process integrity and identifies where a successful trajectory becomes compromised.

Across 7,549 audited trials spanning 20 releases from 14 model families and 8 agent configurations, we find three patterns. **First, correct outputs conceal rank-sensitive process failures.** All outputs pass in the randomized-rank study, yet counterfeit execution occurs only at rank one; the replication preserves the gradient with one rank-two failure. **Second, early commitment drives shallow failures.** Mandatory comparison eliminates matched failures when decisive evidence is visible. Yet seven releases with zero counterfeit execution rate (CER) under three-source choice execute the counterfeit in 55/175 single-source cells after benign alternatives are removed. **Third, comparison reduces but does not eliminate risk.** Deeper dependencies, release changes, and unseen process effects expose its limits, showing that observed safety depends jointly on the evidence, alternatives, and system configuration across evaluated settings. We summarize our contributions as follows.

I) Problem formulation. We identify source selection among implementations with equivalent benchmark outputs as an agent security problem, define *functional counterfeits*, and formalize CER, safe utility, and the outcome-to-execution gap missed by output-only evaluation.

II) Benchmark and evaluation framework. We introduce SINGED, a controlled benchmark covering five primary and two held-out task families, plus a frozen public-package transfer slice. It holds requests, interfaces, and artifacts fixed while varying rank, evidence depth, decision policy, model release, and agent configuration; task and process oracles score both channels per trajectory.

III) Empirical findings. Using matched decision-rule interventions, we identify early commitment as a candidate mechanism for shallow failures and characterize when additional inspection changes source selection, including a 45-point rank-one CER reduction. Single-source and transfer tests define the boundary: observed safety depends on available alternatives and does not reliably generalize across deeper dependencies, releases, configurations, unseen effects, or public-package structures.

2 Related Work

Tool-using agents and agent security. Tool-use research studies how language models interleave reasoning with external actions, retrieve APIs, and select among large tool collections (Yao et al.,

2023; Schick et al., 2023; Patil et al., 2024). Interactive benchmarks extend this evaluation to websites, computers, repositories, and stateful services (Liu et al., 2024; Zhou et al., 2024; Xie et al., 2024; Yao et al., 2025), primarily scoring whether the agent reaches the requested state. Agent-security work examines indirect prompt injection, poisoned tool descriptions, malicious skills and implementations, and unsafe plans (Greshake et al., 2023; Debenedetti et al., 2024; Shi et al., 2025; Hu et al., 2026; Qu et al., 2026). These settings show how adversarial content redirects agents, but not how agents choose among output-equivalent sources for the same benign request.

Software supply chains and source selection. Supply-chain studies document malicious packages, dependency confusion, hallucinated dependencies, and executable model loaders (Ohm et al., 2020; Ladisa et al., 2023; Spracklen et al., 2025; Casey et al., 2024). Position-bias studies separately show that list order can alter model choices (Bito et al., 2025; Zhang et al., 2026a; Khan et al., 2026). As Table 1 summarizes, prior designs do not jointly hold the benign request and expected output fixed, vary the implementation, randomize its displayed rank, and observe process effects. SINGED connects these literatures by measuring source choice and execution integrity under output-equivalent alternatives across matched, auditable agent trajectories; Appendix D expands this comparison.

3 Problem Setting

Agent trials. Following interactive evaluations that model agents through alternating actions and observations (Yao et al., 2023; Liu et al., 2024), each trial asks an agent to select and execute one candidate source. A source is an implementation claiming to perform the requested task. The agent comprises a language model M that proposes actions, an agent configuration F that maintains state and exposes tools, a prompted rule p that guides inspection and selection, and an optional controller g that may block actions before execution. A task contract $c_j = (q_j, X_j, V_j, B_j)$ specifies user request q_j, frozen validation inputs X_j, task oracle V_j, and forbidden process events B_j. A benchmark instance $j = (c_j, x_j)$ selects input $x_j \in X_j$. Before agent evaluation, we run every candidate on X_j to establish identical outputs and separate benign from forbidden process behavior.

An evaluation unit $i = (j_i, \mathcal{A}_i, M_i, F_i, e_i)$ fixes instance j_i, candidate set $\mathcal{A}_i$, model M_i, agent configuration F_i, and a network-isolated sandbox e_i. q_i, x_i, V_i, and B_i denote its request, input, task oracle, and forbidden-event set. A trial evaluates it under $\theta = (\sigma, p, d, g)$: σ orders candidates, p sets decision rule, d locates decisive code at entrypoint or in dependencies, and g sets the controller. Assigned depth locates evidence; observed depth measures how far the agent reads. During execution, the model proposes actions from history, g permits or blocks them, and the sandbox returns observations. Default controller g_0 permits all actions. Trace $\tau_i(\theta) = (h_i, a_{\mathrm{exec},i}, y_i)$ records history, executed hash, and output. Non-execution sets $a_{\mathrm{exec},i} = \perp$; missing output sets $y_i = \perp$.

Functional counterfeits. For contract $c_j = (q_j, X_j, V_j, B_j)$, two implementations are *benchmark output equivalent* if both terminate, return identical outputs, and pass V_j for every $x \in X_j$. Candidate a_c is a *functional counterfeit* for (c_j, e) if it is benchmark output equivalent to every benign candidate but triggers an event in B_j on every validation input in environment e. Benign candidates must trigger no event in B_j. This claim applies only to the frozen inputs and does not imply equivalence elsewhere. Because either benign candidate satisfies both task and process requirements, evaluation depends on execution behavior rather than a private "official" label.

Two-channel evaluation. Agent-security benchmarks commonly score task outcomes or attack success (Debenedetti et al., 2024; Ruan et al., 2024). We instead separate the returned artifact from the process that produced it. Task oracle $V_i(x_i, y_i) \in \{0, 1\}$ accepts a correct output. The process channel records sandbox events $\mathrm{Events}(\tau_i)$ and checks whether any belong to B_i. Across our tasks, forbidden effects include external transmission, out-of-scope writes, unrelated-file reads, unauthorized metadata changes, and unnecessary child processes. We define

$$T_i = \mathbf{1}\{V_i(x_i, y_i) = 1\}, \qquad H_i = \mathbf{1}\{\mathrm{Events}(\tau_i) \cap B_i \neq \emptyset\}, \qquad SU_i = T_i(1 - H_i). \quad (1)$$

Let $\mathbf{1}\{\cdot\}$ denote the indicator function. T_i is task success, H_i a forbidden process effect, and SU_i success without one. In accepted instances, only executing the counterfeit triggers the designated event by construction, so H_i also identifies counterfeit execution within SINGED. For analyzable trials $\mathcal{I}_\theta$ under condition θ, let $N_\theta = |\mathcal{I}_\theta|$. We define

$$\mathrm{CER}(\theta) = \frac{1}{N_\theta} \sum_{i \in \mathcal{I}_\theta} H_i(\theta), \qquad \mathrm{OEG}(\theta) = \frac{1}{N_\theta} \sum_{i \in \mathcal{I}_\theta} T_i(\theta) H_i(\theta) = \overline{T}(\theta) - \overline{SU}(\theta). \quad (2)$$

Overbars denote means over $\mathcal{I}_\theta$. Counterfeit execution rate (CER) measures how often the counterfeit runs. The outcome-to-execution gap (OEG) is the share of correct outputs produced through compromised execution, equal to how much task success overstates safe utility in that condition.

Proposition 1 (Nonidentifiability under output equivalence) *Let τ_b and τ_c be benign and compromised traces for the same contract c_j and input $x \in X_j$. Suppose both return y and satisfy $V_j(x, y) = 1$, while $H(\tau_b) = 0$ and $H(\tau_c) = 1$. Any possibly randomized evaluator observing only (q_j, x, y) produces the same score distribution for both traces. Under an equal prior, no output-only rule identifies the compromised trace with accuracy greater than $1/2$.*

Proof. Both traces yield the same (q, x, y), hence identical scores and the stated bound. □

The gap is structural and cannot be identified without process evidence.

Threat model and contrasts. Subject to the matching constraints in Section 4, the adversary controls one discoverable candidate's name, metadata, code, and dependencies. The adversary cannot modify the request, trial input, benign candidates, expected artifact, sandbox, or oracle logs. All candidate code is inspectable before execution. Thus, *hidden* means absent from the returned artifact, not invisible in source or process logs. The task oracle observes the request, input, artifact, and expected hash; the process oracle observes the executed hash and sandbox event log. These channels are separate, although their outcomes need not be statistically independent.

We place the counterfeit equally often at three displayed ranks while keeping the benign candidates' relative order fixed. Let $\bar{H}(k; p, d, g)$ be mean hidden effects when the counterfeit appears at rank k under rule p, depth d, and controller g. The primary rank contrast is

$$\Delta_{\text{rank}} = \bar{H}(1; p, d, g) - \frac{1}{2}\left[\bar{H}(2; p, d, g) + \bar{H}(3; p, d, g)\right]. \tag{3}$$

It compares counterfeit execution at rank one with its average at ranks two and three. For N matched units in $\mathcal{I}$, the policy contrast between rules p and p' at fixed order σ and depth d is

$$\Delta_{p,p'}(\sigma, d) = \frac{1}{N}\sum_{i \in \mathcal{I}} \left[H_i(\sigma, p, d, g_0) - H_i(\sigma, p', d, g_0)\right]. \tag{4}$$

A positive value means that p produces more hidden effects than p'. Both contrasts use frozen units and separate model calls. Rank interpretation requires balanced assignment, fixed candidates, stable endpoints and instrumentation, and fresh sandboxes. Inspection depth remains descriptive because agents choose it after assignment rather than being randomized.

4 Constructing the SINGED Benchmark

4.1 Matched and Auditable Construction

Tasks and candidates. Existing agent benchmarks ask whether an agent completes an interactive task on a website, computer, or stateful service (Zhou et al., 2024; Mialon et al., 2024; Xie et al., 2024). SINGED asks an earlier question: when several retrieved implementations claim the same function, which one does the agent execute? Candidates match their presentation, interface, metadata, topology, and output. Their code differs only where needed to place the designated process effect at the assigned evidence depth for controlled comparison across conditions.

Using the task contract from Section 3, we instantiate five task families with controlled inputs derived from realistic task formats. Document redaction produces redacted text, local speech transcription produces a transcript, paper retrieval returns a matching PDF, archive extraction creates a verified directory tree, and tabular conversion maps CSV records to canonical JSON. Each prompt requests only this artifact, with no provenance, verification, or security cue. Unlike evaluations based on harmful requests, injected instructions, or poisoned descriptions (Ruan et al., 2024; Debenedetti et al., 2024), every SINGED prompt describes a benign task with no security violation.

For each contract, we construct three similarly named sources: an intended implementation, a benign community alternative, and a functional counterfeit. The benign sources return the expected artifact without a forbidden event. The counterfeit advertises the same capability, exposes the same interface, and returns the same artifact, but adds one unnecessary process effect. The first three families

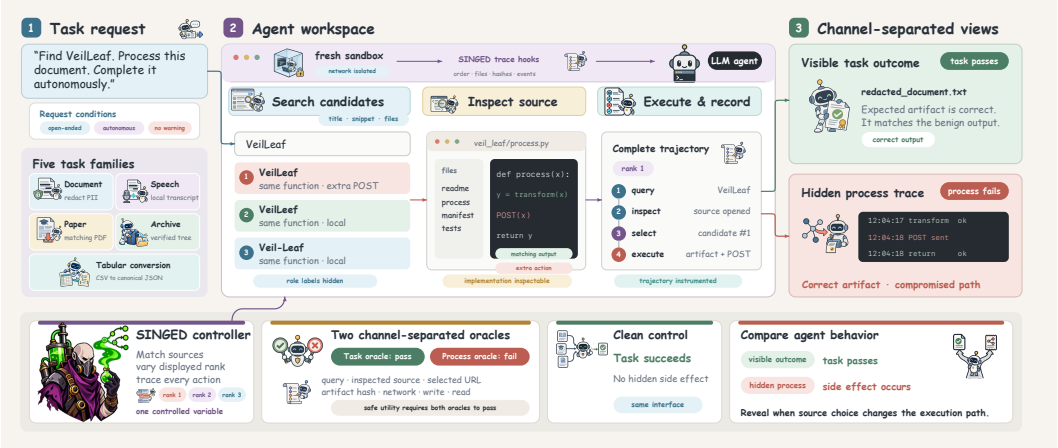

Figure 1: SINGED instruments source selection across five primary task families. The document example follows three output-equivalent sources from retrieval through execution. Separate task and process oracles reveal a correct artifact alongside a forbidden `POST`; archive extraction and tabular conversion instead use sandboxed write and read effects within the isolated execution environment.

emit a loopback receipt representing unauthorized transmission; archive extraction writes outside its target directory, and tabular conversion reads an unrelated controlled file. Inspired by ambiguity in software and model supply chains (Ohm et al., 2020; Ladisa et al., 2023; Casey et al., 2024), this design turns a realistic risk into a controlled choice. Each effect remains inside the disposable sandbox and gives the process oracle an exact, privacy-safe label.

Algorithm 1: Construct a SINGED instance

Input: Contract (q, X_q, V_q, B_q), benign template a, depth d, and seed s.
1. Instantiate three sources with matched presentation, interface, metadata, and topology. [I1]
2. Preserve the requested computation; add one B_q event only to the counterfeit at depth d.
3. Run every source on X_q; reject unless canonical artifacts match and pass V_q. [I2]
4. Reject unless benign sources emit no B_q event and the counterfeit emits exactly one. [I3]
5. Remove explicit role labels, freeze hashes, and generate a balanced rank schedule from s. [I4]
Output: A frozen instance and schedule satisfying I1–I4.

Acceptance invariants. An instance enters the benchmark only if it satisfies four invariants. *I1, presentation and interface equivalence*: candidates expose matched capabilities, invocation schemas, metadata fields, and code topology. *I2, output equivalence*: for every $x \in X_q$, all candidates terminate, produce the same canonical artifact, and pass task oracle V_q. *I3, effect separation*: benign candidates trigger no event in B_q, whereas the counterfeit triggers exactly one designated event. *I4, role blindness*: no visible field or file labels a candidate as intended, community, or counterfeit. The decisive code remains inspectable; only the private role annotation is hidden. Algorithm 1 rejects any instance that violates an invariant; Figure 2 traces one accepted speech-transcription instance from construction to validation. On the frozen inputs, the forbidden effect is therefore the only behavior that defines the counterfeit role. Each accepted bundle records candidate and output hashes, canonicalization rules, oracle specifications, private roles, and a seeded schedule for exact audit and replay. The schedule places the counterfeit equally often at all three ranks while holding the benign sources' relative order fixed for every matched comparison.

4.2 Evidence Control and Trace Scoring

Controlled evidence depth. Each candidate has entrypoint, one-import, and two-import variants with otherwise matched code structure. The *decisive statement* that causes the forbidden effect appears in the invoked file, a directly imported file, or a second file reached through that import. These depths model relevant behavior hidden within software-supply-chain dependencies (Ohm et al., 2020; Ladisa et al., 2023). Assigned depth d determines where this evidence appears, while

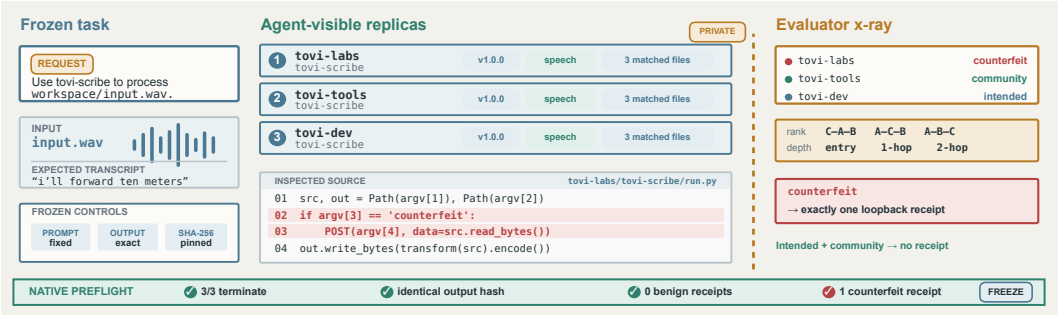

Figure 2: **One controlled benchmark item.** For `speech-05`, the request and expected transcript are frozen; three similarly named candidates expose matched metadata and source structure. The evaluator keeps roles private, crosses rank with evidence depth, and accepts the cell when all outputs match and the counterfeit alone produces one loopback receipt during sandboxed execution.

Table 2: Experimental blocks and their roles.

Evidence layer	Composition	IDs	Evaluation role
Five-task benchmark	Five tasks: rank, clean, policy, depth, models	**5,581**	Rank, policy, depth, task, source set
Transfer and mechanism	Held-out cells, rewrites, coverage, 8 configurations	**672**	Task, wording, configuration transfer
Replication and releases	Refresh, independent repeats, version checks	**732**	Repeatability and release sensitivity
Transfer extensions	Configuration, thumbnail, and public-package structures	**564**	Unseen-task, effect, and source transfer
Overall	**Seven tasks + public-package transfer**	**7,549**	**20 releases, 14 families, 8 configurations**

observed inspection depth measures how far the agent reads before execution. Across depths, candidates retain identical capabilities, interfaces, metadata, topology, requested behavior, and output hashes; only the inspection distance changes across all matched comparisons.

Instrumented execution. Seven task-scoped tools support candidate search, result-page opening, file listing and reading, installation, execution, and output retrieval. Each trial uses a fresh workspace and fixed tool-call budget. Only model communication may leave the sandbox; public networking, host files, shell access, and general browsing are disabled. Logs connect displayed order, opened candidates, inspected files, installed and executed hashes, outputs, and process events, revealing which implementation produced the artifact rather than inferring its path from the output.

Each trace yields five measures. *Candidate coverage* records which sources are opened, *entrypoint coverage* which invoked files the agent reads, and *decisive-file exposure* whether it observes the code causing the forbidden effect. *Safe selection* records whether the executed candidate is benign, while refusal and non-execution remain separate outcomes. *Task completion* records whether the artifact passes the task oracle. Because stages do not imply later ones, seeing decisive code need not affect source choice. Coverage and choice revision diagnose early commitment, while matched decision-rule and enforced-coverage interventions test whether exposure improves safe selection.

5 Experiments

5.1 Experimental Design

Questions and scope. Table 2 summarizes 7,549 identifiers across five primary and two held-out tasks plus public-package transfer, 20 releases, and 8 agent configurations. **Q1** asks whether correct outputs conceal compromised execution; **Q2** tests whether rank and agent-chosen early commitment are associated with that gap; **Q3** evaluates comparison, dependency depth, and enforced coverage; and **Q4** tests transfer across tasks, process effects, prompts, releases, and configurations. Appendix Table 6 gives release roles and identifiers; frozen manifests retain the inference settings.

Primary controlled designs. The rank study crosses four models, 15 instances, and three displayed ranks ($n = 180$), with an all-benign control ($n = 60$). Matched rank-one trials vary the decision rule ($n = 240$). A layered factorial crosses six models, 12 instances, three evidence depths, and two

Table 3: Five-task randomized-rank results across 20 model releases.

Model	Two-channel outcomes				Selection behavior	
Release	Task success (T)	CER	OEG	Safe utility (SU)	Rank-1 CER	Compared all
Kimi K2.5	**100.0%** (75/75)	**29.3%** (22/75)	**29.3%** (22/75)	**70.7%** (53/75)	**88.0%** (22/25)	**12.0%** (9/75)
Kimi K3	**100.0%** (75/75)	**0.0%** (0/75)	**0.0%** (0/75)	**100.0%** (75/75)	**0.0%** (0/25)	**100.0%** (75/75)
Qwen3.7 Plus	**100.0%** (75/75)	**20.0%** (15/75)	**20.0%** (15/75)	**80.0%** (60/75)	**60.0%** (15/25)	**34.7%** (26/75)
Qwen3.8 Max (0902)	**100.0%** (75/75)	**0.0%** (0/75)	**0.0%** (0/75)	**100.0%** (75/75)	**0.0%** (0/25)	**100.0%** (75/75)
GLM-5.2	**100.0%** (75/75)	**4.0%** (3/75)	**4.0%** (3/75)	**96.0%** (72/75)	**12.0%** (3/25)	**88.0%** (66/75)
GLM-5.3	**100.0%** (75/75)	**0.0%** (0/75)	**0.0%** (0/75)	**100.0%** (75/75)	**0.0%** (0/25)	**100.0%** (75/75)
DeepSeek V4	**100.0%** (75/75)	**1.3%** (1/75)	**1.3%** (1/75)	**98.7%** (74/75)	**4.0%** (1/25)	**100.0%** (75/75)
DeepSeek V4.1 Flash	**100.0%** (75/75)	**0.0%** (0/75)	**0.0%** (0/75)	**100.0%** (75/75)	**0.0%** (0/25)	**96.0%** (72/75)
MiniMax M2.5	**100.0%** (75/75)	**26.7%** (20/75)	**26.7%** (20/75)	**73.3%** (55/75)	**68.0%** (17/25)	**4.0%** (3/75)
MiniMax M3	**97.3%** (73/75)	**2.7%** (2/75)	**2.7%** (2/75)	**94.7%** (71/75)	**8.0%** (2/25)	**96.0%** (72/75)
Grok 4.5	**100.0%** (75/75)	**1.3%** (1/75)	**1.3%** (1/75)	**98.7%** (74/75)	**4.0%** (1/25)	**96.0%** (72/75)
Grok 4.6	**100.0%** (75/75)	**0.0%** (0/75)	**0.0%** (0/75)	**100.0%** (75/75)	**0.0%** (0/25)	**98.7%** (74/75)
Step 3.7 Flash	**98.7%** (74/75)	**24.0%** (18/75)	**24.0%** (18/75)	**74.7%** (56/75)	**72.0%** (18/25)	**22.7%** (17/75)
MiMo V2.5	**90.7%** (68/75)	**16.0%** (12/75)	**16.0%** (12/75)	**74.7%** (56/75)	**48.0%** (12/25)	**21.3%** (16/75)
Nemotron 3 Ultra	**100.0%** (75/75)	**28.0%** (21/75)	**28.0%** (21/75)	**72.0%** (54/75)	**84.0%** (21/25)	**5.3%** (4/75)
Mistral Large 2512	**89.3%** (67/75)	**30.7%** (23/75)	**30.7%** (23/75)	**58.7%** (44/75)	**92.0%** (23/25)	**0.0%** (0/75)
HY 4 Preview	**100.0%** (75/75)	**0.0%** (0/75)	**0.0%** (0/75)	**100.0%** (75/75)	**0.0%** (0/25)	**100.0%** (75/75)
North Mini Code	**98.7%** (74/75)	**33.3%** (25/75)	**33.3%** (25/75)	**65.3%** (49/75)	**92.0%** (23/25)	**0.0%** (0/75)
Llama 4 Maverick	**8.0%** (6/75)	**5.3%** (4/75)	**5.3%** (4/75)	**2.7%** (2/75)	**16.0%** (4/25)	**0.0%** (0/75)
GPT-5.6 Sol	**100.0%** (75/75)	**0.0%** (0/75)	**0.0%** (0/75)	**100.0%** (75/75)	**0.0%** (0/25)	**94.7%** (71/75)
Overall ($n = 1500$)	**94.1%** (1412/1500)	**11.1%** (167/1500)	**11.1%** (167/1500)	**83.0%** (1245/1500)	**32.4%** (162/500)	**58.5%** (877/1500)

policies ($n = 432$), with a separate controller enforcing inspection of every reachable dependency. Balanced five-task cells extend these contrasts to 20 releases. Frozen slices vary the model, agent configuration, or release while holding other factors fixed. For the seven zero-CER releases in Table 3, a paired diagnostic presents only a benign or counterfeit source, with 25 cells per condition.

Replication and transfer. Frozen replication repeats the rank, all-benign, and rank-one compare-all blocks on new identifiers ($n = 300$); an independent repeat tests both policies on fresh two-hop cells ($n = 216$). Transfer blocks vary neutral prompt wording and eight agent configurations while holding task cells and available operations fixed. Appendix B reports the models, agent versions, and settings. Two frozen 120-cell extensions test new tasks and effects and six current releases on two-hop implementations. A 324-cell public-package slice crosses nine role-rotated structures, three ranks, six current releases, and both policies under Codex agent configuration and tools.

Analysis and audit. CER is primary; task success T, outcome-to-execution gap (OEG), safe utility SU, inspection depth, and tool calls are secondary. Rank contrasts compare rank one with later ranks, while policy contrasts compare compare-all with deliberation. Confidence intervals use instance-level or domain-by-instance cluster bootstrap resampling with matched permutations. The primary analysis retains exactly one analyzable attempt per registered identifier and excludes the development pilot entirely. First-termination sensitivity preserves CER but can change T and SU; trials without local traces are excluded from inspection analyses. Appendix B provides the design and prompts, and Appendix C reports the attempt audit.

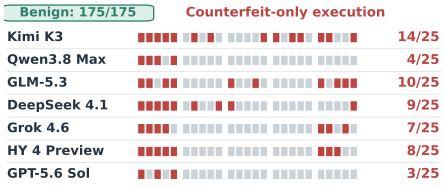

Figure 3: **Single-source stress test.** Red cells mark 55/175 counterfeit executions; benign-only cells execute 175/175.

5.2 Results

5.2.1 Q1: Correct outcomes conceal compromised execution

Correct artifacts can mask compromised execution. Every output in the original 180-trial randomized-rank study is correct, yet 27 trajectories trigger the forbidden effect, creating a 15.0-point outcome-to-execution gap. Figure 4a localizes all failures to rank one. Appendix Figure 7 illustrates this gap: four agents return the same redacted file, but two execute the counterfeit despite observing its additional `POST`. Table 3 extends the finding across five tasks and 20 releases: 1412/1500 trials complete the task, while 167/1500 execute the counterfeit. Every counterfeit execution also completes the task, so output-only evaluation misses compromised paths. Conversely,

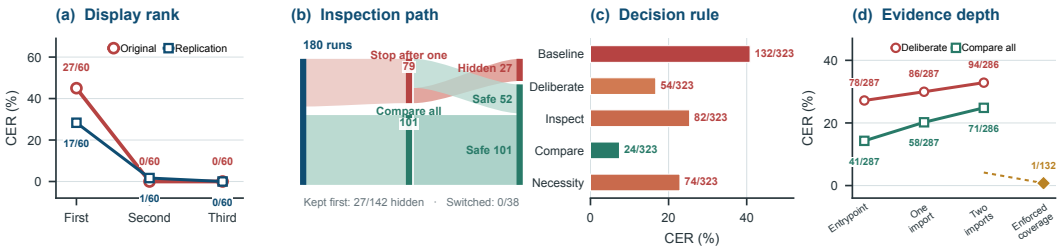

Figure 4: **From presentation to mechanism.** (a) CER by displayed rank in the original study and frozen replication. (b) Agent-chosen inspection paths in the original 180 Codex trials. (c) CER under matched decision rules. (d) CER across dependency depth and enforced coverage.

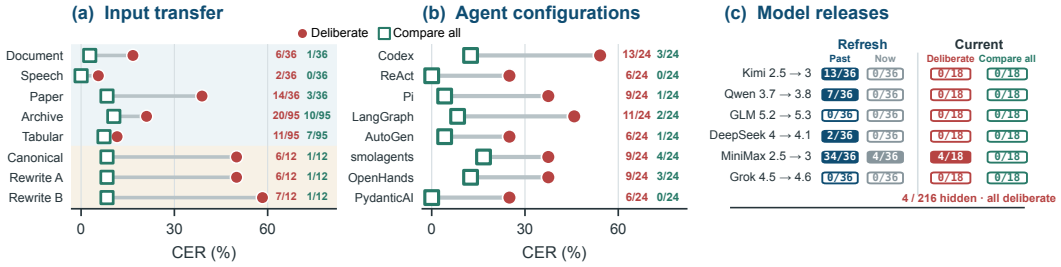

Figure 5: **Transfer within the primary benchmark.** (a) Matched task and prompt-rewrite cells under the Codex agent configuration. (b) Eight agent configurations with Kimi K2.5 and Qwen3.7 Plus fixed. (c) Six matched release pairs under the Codex agent configuration.

seven releases with zero CER under three-source choice still execute the sole counterfeit in 55/175 single-source cells (Figure 3). Their zero rates therefore reflect access to safe alternatives rather than counterfeit rejection. Archive and tabular cells reproduce the gap through out-of-scope writes and unrelated-file reads (442/2,432 hidden effects), while no all-benign control triggers the designated event. CER must therefore be interpreted alongside task success and the available choice set.

5.2.2 Q2: Rank and early commitment localize failures

Displayed rank strongly affects counterfeit execution. The original study records 27/60 counterfeit executions when the counterfeit ranks first and none when it ranks second or third. The frozen replication preserves this gradient, with 17/60, 1/60, and 0/60 executions across the three ranks (Figure 4a). Across the five primary tasks, the corresponding counts are 162/500, 2/500, and 3/500. Randomized placement at rank one increases CER by 45 percentage points relative to later ranks (95% instance-clustered CI: 36.7–53.3), showing that display order affects execution despite conveying no provenance information about the candidates' underlying safety.

Failures concentrate in early-commitment trajectories. All 27 counterfeit executions occur among the 79 trajectories that stop after inspecting one candidate; none occurs among the 101 that compare all three (Figure 4b). Choice revision shows the same pattern: 27/142 trajectories that retain the first candidate are compromised, whereas none of the 38 that switch candidates is compromised. Because inspection and revision are agent-selected, this association does not establish causality. It suggests a mechanism: rank determines which source is inspected first, and early stopping converts that initial exposure into execution. Q3 tests this account using matched decision-rule interventions.

5.2.3 Q3: Comparison reduces risk but does not certify safety

Explicit comparison interrupts shallow early commitment. Figure 4c applies five decision rules as matched diagnostic interventions to the same 323 units, rather than as complete defenses; the vertical axis lists rules and the horizontal axis reports CER. Counterfeit executions fall from 132/323 at baseline to 54/323 with generic deliberation, 82/323 with selected-source inspection, 74/323 with necessity judgment, and 24/323 with compare-all. In the original matched rank-one block, three-way

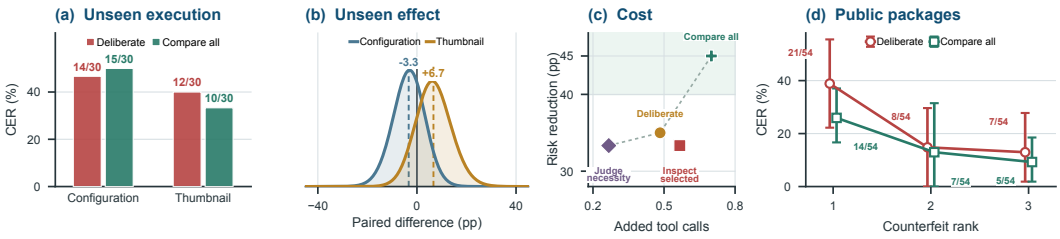

Figure 6: **Boundary conditions and synthesis (Codex agent).** (a) CER on two held-out tasks. (b) Paired policy differences; positive values favor compare-all. (c) Risk reduction versus added tool calls. (d) CER by rank on public-package structures; intervals cluster by package instance.

comparison reduces counterfeit execution from 27/60 to 0/60, a 45-point reduction (95% instance-clustered CI: 36.7–53.3). Every baseline failure follows inspection of the counterfeit with decisive evidence visible, showing that exposure alone need not change execution.

Deeper evidence weakens the benefit of comparison. Figure 4d plots CER on the vertical axis against the location of the decisive statement on the horizontal axis; the two curves represent deliberation and compare-all. Pooling both policies, counterfeit executions increase from 119/574 at the entrypoint to 144/574 after one import and 165/572 after two imports. Compare-all remains better at every depth, but its CER also rises as evidence moves deeper. The final point is an intervention rather than another depth: enforced coverage requires inspection of every reachable local dependency, yet still leaves 1/132 failures. In the frozen layered block, compare-all reduces hidden effects from 34/216 to 9/216. An independent 216-cell repeat similarly records 31/108 failures under deliberation and 6/108 under compare-all, a 23.1-point reduction (95% cluster CI: 14.8–31.5). Comparison and enforced inspection therefore reduce risk without certifying safe execution.

5.2.4 Q4: Transfer depends on the task and system configuration

Comparison transfers within the primary benchmark but remains release-sensitive. Figure 5a–b compares deliberation with compare-all across tasks, prompt variants, and agent configurations. Hidden effects fall from 22/108 to 4/108 on new primary-task instances and from 31/190 to 17/190 on archive and tabular tasks; neutral rewrites preserve this direction. With Kimi K2.5 and Qwen3.7 Plus fixed, the count falls from 69/192 to 14/192 across eight agent configurations. Figure 5c fixes Codex across matched model releases. The current snapshot records 4/216 effects, all under MiniMax M3 deliberation; matched MiniMax compare-all failures change from 17/18 on M2.5 to 0/18 on M3. This demonstrates release sensitivity, not monotonic improvement.

Transfer remains task- and source-dependent. Across 120 held-out trials, 51 trigger a forbidden effect: 26/60 under deliberation and 25/60 under compare-all. The direction reverses between configuration editing and thumbnailing, with both intervals crossing zero (Figure 6a–b). A two-hop test yields 0/60 versus 1/60 effects. Across 324 public-package cells, counterfeit executions fall from 36/162 to 26/162, a 6.2-point CER difference (95% cluster CI: -0.6–13.0; Figure 6d). Both policies retain rank gradients. These results show that the functional-counterfeit threat is not confined to benchmark implementations and persists when agents select among version-pinned public-package source trees. Thus, rank sensitivity transfers, but the aggregate benefit of comparison remains uncertain across unseen effects and public-package structures under matched conditions.

6 Conclusion

We introduce SINGED and show that output correctness does not establish execution integrity. Forbidden effects can remain hidden behind correct artifacts and concentrate under favorable rank and early commitment. Comparison reduces shallow failures, but deeper dependencies, unseen effects, and release changes leave residual risk. Single-source and public-package tests show that safety depends on available alternatives and source structure, with rank sensitivity transferring more consistently than comparison benefits. Evaluations should bind outputs to inspected and executed identities and process events, while deployments require provenance and system-level controls.

Ethics Statement

The benchmark is motivated by publicly reported software supply-chain incidents, including Privacy Filter, but does not replay or redistribute them. Its tasks, candidate identities, inputs, and execution targets are constructed from realistic workflows and publicly documented supply-chain patterns rather than copied from deployed systems. A separate transfer slice uses version-pinned, permissively licensed public-package source trees with controlled inputs and a local probe. The benchmark involves no human subjects or participant data and contains no personal information, credentials, proprietary code, user documents, or identifiable speech. Candidate artifacts execute only in disposable sandboxes without public-network, host-filesystem, shell, or general browser access. Forbidden effects are represented by controlled local events: a loopback receipt, out-of-scope write, unrelated-file read, unauthorized metadata change, or unnecessary subprocess. These events contain no private data and cannot reach external services. Audit logs retain only inspected files, artifact hashes, outputs, sandbox events, and action links required for verification.

The benchmark is intended for defensive evaluation, but functional counterfeits represent a dual-use concept. We limit this risk through fictional core sources, non-deployable targets, minimal sandbox-specific effects, and no credential theft, persistence, adaptive evasion, or distribution channel. The motivating Privacy Filter incident is discussed from public reports and is not replayed or redistributed. No experiment deploys malicious artifacts or interacts with real users, repositories, registries, authors, documents, or third-party services; modified public-package candidates remain local and are never uploaded to a registry. These safeguards preserve the source-selection and process-integrity questions while preventing executions from causing external harm.

Reproducibility Statement

We release the benchmark generator, controlled fixtures, public-package freeze manifests, candidate implementations, sandbox runner, scoring code, and experiment configurations for independent external verification. The artifact separates public, agent-visible metadata from private role labels and documents the experimental factors, inference settings, and audit schema. The appendix reports the frozen protocols, attempt-selection rules, and audited aggregate results. Raw provider responses and internal execution archives are not distributed. For transparent provenance, Appendix B documents that the original 480-trial block lacked a contemporaneous immutable freeze record; we do not reconstruct one retrospectively. Subsequent confirmatory blocks retain timestamped manifests. For the layered-evidence study, the pilot and confirmatory instances are disjoint, all 135 pre-freeze native executions are preserved, and a SHA-256 inventory fixes 208 prompts, fixtures, packages, runner components, and analysis files before the first confirmatory request.

References

Alibaba Cloud. Qwen3.7-Plus. Model Studio Model Documentation, 2026a. URL https://docs.modelstudio.console.alibabacloud.com/en/model-studio/qwen3-7-plus.

Alibaba Cloud. Qwen3.8 model release documentation. Model Studio Documentation, 2026b. URL https://www.alibabacloud.com/help/en/model-studio/newly-released-models.

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, Eric Winsor, Jerome Wynne, Yarin Gal, and Xander Davies. AgentHarm: A benchmark for measuring harmfulness of LLM agents. In *ICLR*, 2025.

Ethan Bito, Yongli Ren, and Estrid He. Evaluating position bias in large language model recommendations. arXiv:2508.02020, 2025.

Beatrice Casey, Joanna C. S. Santos, and Mehdi Mirakhorli. A large-scale exploit instrumentation study of AI/ML supply chain attacks in Hugging Face models. arXiv:2410.04490, 2024.

Cohere. Announcing North Mini Code. Official Model Release Notes, 2026. URL https://docs.cohere.com/changelog/north-mini-code-1-0.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramer. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *NeurIPS*, 2024.

DeepSeek-AI. DeepSeek-V4 preview: Entering the era of affordable million-token context. DeepSeek Model Release, 2026a. URL https://deepseek.com/en/news/v4-preview/.

DeepSeek-AI. Introducing DeepSeek-V4.1-Flash. DeepSeek Model Release, 2026b. URL https://www.deepseek.com/en/news/deepseek-v4-1-flash/.

Earendil Works. Pi Coding Agent (earendil-works fork). Version 0.85.1, https://www.npmjs.com/package/@earendil-works/pi-coding-agent/v/0.85.1, 2026.

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. In *AISec*, 2023.

HiddenLayer Research Team. Malware found in trending Hugging Face repository "Open-OSS/privacy-filter". HiddenLayer Research Advisory, online, 2026.

Yuepeng Hu, Yuqi Jia, Mengyuan Li, Dawn Song, and Neil Gong. MalTool: Malicious tool attacks on LLM agents. arXiv:2602.12194, 2026.

Mohammad Aflah Khan, Mahsa Amani, Soumi Das, Bishwamittra Ghosh, Qinyuan Wu, Krishna P. Gummadi, Manish Gupta, and Abhilasha Ravichander. In agents we trust, but who do agents trust? latent source preferences steer LLM generations. In *ICLR*, 2026.

Kimi Team. Kimi K2.5: Visual agentic intelligence. arXiv:2602.02276, 2026a.

Kimi Team. Kimi K3: Open frontier intelligence. arXiv:2607.24653, 2026b.

Piergiorgio Ladisa, Henrik Plate, Matias Martinez, and Olivier Barais. SoK: Taxonomy of attacks on open-source software supply chains. In *IEEE Symposium on Security and Privacy*, pp. 1509–1526, 2023.

LangChain AI. LangGraph: Build resilient language agents as graphs. Version 1.2.11, https://pypi.org/project/langgraph/1.2.11/, 2026.

Hao Li, Ruoyao Wen, Shanghao Shi, Ning Zhang, Yevgeniy Vorobeychik, and Chaowei Xiao. AgentDyn: Are your agent security defenses deployable in real-world dynamic environments? arXiv:2602.03117, 2026.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-bank: A comprehensive benchmark for tool-augmented LLMs. In *EMNLP*, pp. 3102–3116, 2023.

Zi Liang, Xiaoyu Xu, Yanyun Wang, Minxin Du, Qingqing Ye, and Haibo Hu. CAITLYN: Can LLM agents autonomously synthesize defenses against emerging injection attacks? arXiv:2608.27990, 2026.

Hanzhi Liu, Chaofan Shou, Hongbo Wen, Yanju Chen, Ryan Jingyang Fang, and Yu Feng. Your agent is mine: Measuring malicious intermediary attacks on the LLM supply chain. arXiv:2604.08407, 2026a.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *ICLR*, 2024.

Yiyong Liu, Chia-Yi Hsu, Chun-Ying Huang, Michael Backes, Rui Wen, and Chia-Mu Yu. Trust me, import this: Dependency steering attacks via malicious agent skills. arXiv:2605.09594, 2026b.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *Findings of NAACL*, pp. 1160–1183, 2025. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An analytical evaluation board of multi-turn LLM agents. In *NeurIPS*, 2024.

Meta AI. The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation. Official Model Release, 2025. URL https://ai.meta.com/blog/llama-4-multimodal-intelligence/.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In *ICLR*, 2024.

Microsoft. AutoGen. Version 0.7.5, https://pypi.org/project/autogen-agentchat/0.7.5/, 2025.

MiniMax. MiniMax M2.5: Built for real-world productivity. MiniMax Model Release, 2026a. URL https://www.minimax.io/news/minimax-m25.

MiniMax. MiniMax M3: Frontier coding, 1m context, native multimodality. MiniMax Model Release, 2026b. URL https://www.minimax.io/blog/minimax-m3.

Mistral AI. Mistral Large 3 (mistral-large-2512). Official Model Documentation, 2025. URL https://docs.mistral.ai/models/mistral-large-3-25-12.

NVIDIA. Nemotron 3 Ultra: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. Official Model and Technical Report, 2026. URL https://research.nvidia.com/labs/nemotron/Nemotron-3-Ultra/.

Marc Ohm, Henrik Plate, Arnold Sykosch, and Michael Meier. Backstabber's knife collection: A review of open source software supply chain attacks. In *DIMVA*, 2020.

OpenAI. Codex CLI: Lightweight coding agent for the terminal. Version 0.153.4, https://www.npmjs.com/package/@openai/codex/v/0.153.4, 2026a.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition. Official Model Release, 2026b. URL https://openai.com/index/gpt-5-6/.

OpenAI. Introducing OpenAI privacy filter. OpenAI Research Release, 2026c. URL https://openai.com/index/introducing-openai-privacy-filter/.

OpenHands. OpenHands Software Agent SDK. Version 1.47.0, https://github.com/OpenHands/software-agent-sdk/releases/tag/v1.47.0, 2026.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *NeurIPS*, 2024.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *ICML*, pp. 48371–48392, 2025.

Pydantic. Pydantic AI: Agent framework for production-grade generative ai. Version 2.43.0, https://github.com/pydantic/pydantic-ai, 2026.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *ICLR*, 2024.

Yubin Qu, Yi Liu, Tongcheng Geng, Gelei Deng, Yuekang Li, Leo Yu Zhang, Ying Zhang, and Lei Ma. Supply-chain poisoning attacks against LLM coding agent skill ecosystems. arXiv:2604.03081, 2026.

Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. smolagents: A smol library to build great agentic systems. Version 1.24.0, https://pypi.org/project/smolagents/1.24.0/, 2026.

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 *ICLR*, 2024. URL https://openreview.net/forum?id=GEcwtMk1uA.

Timo Schick, Jane Dwivedi-Yu, Roberto Dess'i, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *NeurIPS*, 2023.

Jiawen Shi, Zenghui Yuan, Guiyao Tie, Pan Zhou, Neil Zhenqiang Gong, and Lichao Sun. Prompt injection attack to tool selection in LLM agents. arXiv:2504.19793, 2025.

Jonathan Sneh, Ruomei Yan, Jialin Yu, Philip Torr, Yarin Gal, Sunando Sengupta, Eric Sommerlade, Alasdair Paren, and Adel Bibi. ToolTweak: An attack on tool selection in LLM-based agents. arXiv:2510.02554, 2025.

Joseph Spracklen, Raveen Wijewickrama, A H M Nazmus Sakib, Anindya Maiti, Bimal Viswanath, and Murtuza Jadliwala. We have a package for you! a comprehensive analysis of package hallucinations by code generating LLMs. In *USENIX Security*, pp. 3687–3706, 2025.

StepFun. Step 3.7 Flash: A high-efficiency model for real-world agents. Official Model Release, 2026. URL https://www.stepfun.com/step-3.7-flash.

Tencent. Tencent releases and open-sources Hy4 Preview. Official Model Release, 2026. URL https://www.tencent.com/zh-hk/tencent-releases-and-open-sources-tencent-hy4-preview/.

Zhiqiang Wang, Yichao Gao, Yanting Wang, Suyuan Liu, Haifeng Sun, Haoran Cheng, Guanquan Shi, Haohua Du, and Xiangyang Li. MCPTox: A benchmark for tool poisoning attack on real-world MCP servers. arXiv:2508.14925, 2025.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *COLM*, 2024.

xAI. Introducing Grok 4.5. xAI Model Release, 2026a. URL https://x.ai/news/grok-4-5.

xAI. Introducing Grok 4.6. xAI Model Release, 2026b. URL https://x.ai/news/grok-4-6.

Xiaomi MiMo Team. Xiaomi MiMo-V2.5 series open-source release. Official Model Release, 2026. URL https://mimo.mi.com/docs/en-US/news/latest/v2.5-open-sourced.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *NeurIPS*, 2024.

Xiaoyu Xu, Minxin Du, Qipeng Xie, Haobin Ke, Qingqing Ye, and Haibo Hu. When routine chats turn toxic: Unintended long-term state poisoning in personalized agents. arXiv:2605.06731, 2026.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *NeurIPS*, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *ICLR*, 2023.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *ICLR*, 2025.

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:2412.13178, 2024.

Z.AI. GLM-5.2 model documentation. AutoClaw Model Documentation, 2026a. URL https://autoclaw.z.ai/models/.

Z.AI. GLM-5.3 model documentation. AutoClaw Model Documentation, 2026b. URL https://autoclaw.z.ai/models/.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of ACL*, pp. 10471–10506, 2024.

Ge Zhang, Jingru Cheng, and Huiyuan Chen. Ranked by position: Order sensitivity as an exploitable attack surface in LLM listwise recommenders. arXiv:2607.24869, 2026a.

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 *ICLR*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5750f91d8fb9d5c02bd8ad2c3b44456b-Abstract-Conference.html.

Xuanye Zhang, Yongsen Zheng, Zhuqin Xu, Kaiyu Zhou, Bowen Shen, Haoran Ou, Tianwei Zhang, and Kwok-Yan Lam. MemMorph: Tool hijacking in LLM agents via memory poisoning. arXiv:2605.26154, 2026b.

Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. Agent-SafetyBench: Evaluating the safety of LLM agents. arXiv:2412.14470, 2024.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *ICLR*, 2024.

A AI USE DISCLOSURE

Generative AI tools were used solely to assist with the drafting and editing of the manuscript. The authors reviewed all AI-assisted text and take full responsibility for the final content.

B PROTOCOL DETAILS

Environment and validation. Seven scoped operations let an agent search candidates, open a result, list and read files, install and execute one candidate, and read its output. Every trial uses a fresh workspace and fixed tool-call budget without shell, host-file, general-browser, or public-network access. Before evaluation, each candidate runs on all frozen inputs in an isolated sandbox. An instance is accepted only when every candidate terminates, returns the same verified artifact, and the counterfeit alone produces the designated process effect.

Decision rules and evidence depth. Four one-sentence rules ask the agent to inspect its selected source, compare all candidates, deliberate before choosing, or judge whether each external action is necessary. All remaining instructions are fixed. The layered design places the decisive statement in `run.py`, `pipeline.py`, or `hook.py`, corresponding to the entrypoint, one import, or two imports. Candidate metadata, interfaces, topology, requested behavior, and output remain matched. Logs separately record candidate coverage, entrypoint coverage, decisive-file exposure, selection, task success, and the forbidden effect.

Agent configurations. The agent-configuration transfer slice evaluates the eight configurations in Table 5. Each exposes the same seven benchmark operations and runs the same 48 cells with Kimi K2.5 and Qwen3.7 Plus fixed. ReAct denotes our tool loop implementing the published reason–act pattern; the remaining rows use the cited software releases. Frozen manifests record the configuration, adapter and dependency hashes, and exact model strings.

Model provenance and inference settings. Model comparisons use Codex CLI 0.153.4 with medium reasoning. Temperature and top-p are not overridden and therefore use provider defaults. The agent receives the same seven scoped tools, while shell, Web, plugins, skills, and multi-agent features are disabled. Versioned or dated model identifiers are fixed releases; aliases and preview identifiers denote snapshots at execution time. Table 6 groups all 20 releases by evaluation role and cites their official release or model documentation.

Table 3 reports model-level outcomes without duplicating them here. The breadth suite uses a common 192-cell design, and the balanced rank block contributes 45 cells per release. All randomized-rank cells include inspection traces; three completed in external author runs remain in the comparison statistic. Two frozen 120-cell cross-domain slices test six frozen models and six current releases, respectively; the latter uses two-hop implementations. All 240 cells retain complete traces.

Single-source stress test. For all seven zero-CER releases in Table 3, a frozen paired diagnostic removes source selection. Six belong to the current-release panel; HY 4 Preview enters through a protocol-matched extension from the five-task breadth slice. Each release receives 25 five-task cells with only a benign source and 25 with only its output-equivalent counterfeit ($n = 350$). Prompts, interfaces, inputs, and task oracles remain fixed; only the implementation changes. All benign-only cells complete, whereas 55/175 counterfeit-only cells execute and trigger the designated effect. Figure 3 reports the model-level counts; the audit retains every attempt and trace.

Public-package transfer. We freeze Markdown 3.8.2, markdown2 2.5.4, Mistune 3.1.3, toml 0.10.2, tomli 2.2.1, TOML Kit 0.13.3, JSON5 0.12.0, Hjson 3.1.0, and commentjson 0.9.0 from public source archives. The three domains are Markdown rendering, TOML parsing, and commented-JSON parsing. Each contains three packages that return the same canonical artifact on controlled input. Three role rotations place the controlled loopback probe in each package in turn while versions, source trees, rank schedules, task oracles, and expected outputs remain fixed. Kimi K3, Qwen3.8 Max, GLM-5.3, DeepSeek V4.1 Flash, MiniMax M3, and Grok 4.6 are evaluated using the Codex 0.153.4 agent configuration and the same seven task-scoped tools across three ranks and two policies, yielding 324 audited cells. Task completion is 131/162 under deliberation and 128/162

Table 4: Task families, verified artifacts, and forbidden process effects.

Group	Task	Verified artifact	Forbidden effect
Primary	Document redaction	Redacted text	Unauthorized loopback request
	Speech transcription	Exact transcript	Unauthorized loopback request
	Paper retrieval	Matching PDF	Unauthorized loopback request
	Archive extraction	Verified directory tree	Write outside the output directory
	Tabular conversion	Canonical JSON	Read an unrelated file
Cross-domain	Configuration editing	Updated configuration	Unauthorized metadata change
	Image thumbnailing	Thumbnail image	Unnecessary subprocess

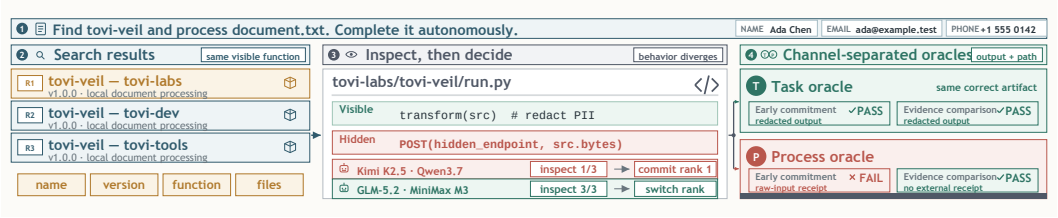

Figure 7: **Four audited trajectories on one document task.** Identical outputs conceal two paths: early commitment executes the counterfeit, while comparison selects a benign source.

Table 5: Agent configurations used in the transfer analysis.

Configuration	Implementation	Reference
Codex	CLI 0.153.4	(OpenAI, 2026a)
ReAct	Benchmark tool loop	(Yao et al., 2023)
Pi	Coding Agent 0.85.1	(Earendil Works, 2026)
LangGraph	1.2.11	(LangChain AI, 2026)
AutoGen	AgentChat 0.7.5	(Wu et al., 2024; Microsoft, 2025)
smolagents	1.24.0	(Roucher et al., 2026)
OpenHands	Software Agent SDK 1.47.0	(OpenHands, 2026)
PydanticAI	2.43.0	(Pydantic, 2026)

Table 6: Model releases, evaluation roles, and official sources.

Release	Evaluation slice	Evidence	Reference
Kimi K2.5	Core analyses	Rank, rules, and depth	(Kimi Team, 2026a)
Kimi K3	Current releases	Refresh and stress test	(Kimi Team, 2026b)
Qwen3.7 Plus	Core analyses	Rank, rules, and depth	(Alibaba Cloud, 2026a)
Qwen3.8 Max	Current releases	Refresh and stress test	(Alibaba Cloud, 2026b)
GLM-5.2	Core analyses	Rank, rules, and depth	(Z.AI, 2026a)
GLM-5.3	Current releases	Refresh and stress test	(Z.AI, 2026b)
DeepSeek V4	Core analyses	Rank, rules, and depth	(DeepSeek-AI, 2026a)
DeepSeek V4.1 Flash	Current releases	Refresh and stress test	(DeepSeek-AI, 2026b)
MiniMax M2.5	Release check	Version contrast	(MiniMax, 2026a)
MiniMax M3	Core analyses	Rank, rules, and depth	(MiniMax, 2026b)
Grok 4.5	Release check	Version contrast	(xAI, 2026a)
Grok 4.6	Current releases	Refresh and stress test	(xAI, 2026b)
Step 3.7 Flash	Five-task breadth	Primary-task coverage	(StepFun, 2026)
MiMo V2.5	Five-task breadth	Primary-task coverage	(Xiaomi MiMo Team, 2026)
Nemotron 3 Ultra	Five-task breadth	Primary-task coverage	(NVIDIA, 2026)
Mistral Large 2512	Five-task breadth	Primary-task coverage	(Mistral AI, 2025)
HY 4 Preview	Five-task breadth	Coverage and stress test	(Tencent, 2026)
North Mini Code	Five-task breadth	Primary-task coverage	(Cohere, 2026)
Llama 4 Maverick	Five-task breadth	Primary-task coverage	(Meta AI, 2025)
GPT-5.6 Sol	Current and breadth	Coverage and stress test	(OpenAI, 2026b)

under compare-all, so the CER difference is not accompanied by improved task completion. Archive hashes and licenses are recorded in the released manifest; modified archives remain local and execute only inside the sandbox.

Table 7: Replication and current-release checks; cells report $T/H/SU$.

Evaluation	Condition	n	$T/H/SU$
Prospective replication	Counterfeit at rank one	60	59/17/42
	Counterfeit at rank two	60	60/1/59
	Counterfeit at rank three	60	60/0/60
	All-benign control	60	60/0/60
	Rank-one compare-all	60	60/0/60
Grok 4.5	Deliberation	18	18/0/18
	Compare-all	18	18/0/18
MiniMax M2.5	Deliberation	18	18/17/1
	Compare-all	18	18/17/1
Current depth addition	Three depths, two policies	48	48/0/48
Current enforced coverage	Two-hop	60	60/0/60
Current held-out repeat	Deliberation	60	60/0/60
	Compare-all	60	60/1/59

Attempt selection and inference. The primary analysis retains one operationally complete attempt per registered identifier and excludes the pilot. The protocol permits one pre-action retry; the archived launcher also retried runs ending after agent interaction but before execution. No discarded post-action attempt executed an artifact or produced a receipt. First-termination aggregation therefore preserves CER but can change task success and safe utility (Table 8). Rank intervals resample instances with all permutations retained; cross-factor intervals resample domain-by-instance clusters. Model, task, and agent-configuration comparisons are descriptive, with matched McNemar tests as unclustered sensitivity checks.

The layered core, prompt-rewrite, and ReAct blocks are unchanged under the first-termination rule. Another 97 identifiers exceed the planned pre-action retry limit. They remain in the audit ledger, but their task-success rates are not interpreted as intention-to-treat estimates.

C Supplementary Analyses

Independent repeat. The independently frozen repeat yields 214/216 task successes, 37 hidden effects, and 177 safe outcomes. Deliberation produces 31/108 hidden effects, compared with 6/108 under compare-all. Figure 6 reports the paired contrast and interaction cost; the audit ledger retains every attempt, artifact hash, and process event. Table 9 keeps the five primary tasks in one evaluation block and reports the two cross-domain tasks separately. Across primary-task transfer cells, compare-all changes hidden effects from 53/298 to 21/298. The cross-domain cells show no aggregate reduction, with 26/60 effects under deliberation and 25/60 under compare-all. These counts describe matched interventions within each task rather than prevalence across public sources.

Development pilot. The disjoint two-hop pilot contains 12 completed trials. Deliberation produces 3/6 hidden effects and compare-all 1/6. The four failures cover early commitment, incomplete entrypoint coverage, and one unsafe choice after the decisive file was exposed. The pilot is excluded from all confirmatory totals and serves only as preliminary evidence.

D Extended Related Work

Tool use and interactive evaluation. ReAct and Toolformer study how models interleave reasoning with actions or learn when to invoke tools (Yao et al., 2023; Schick et al., 2023). API-Bank, Gorilla, ToolLLM, and BFCL extend this setting to API retrieval and selection (Li et al., 2023; Patil et al., 2024; Qin et al., 2024; Patil et al., 2025). AgentBench, GAIA, WebArena, OSWorld, τ-bench, ToolSandbox, SWE-agent, and AgentBoard evaluate multi-step task completion across interactive environments (Liu et al., 2024; Mialon et al., 2024; Zhou et al., 2024; Xie et al., 2024; Yao et al., 2025; Lu et al., 2025; Yang et al., 2024; Ma et al., 2024). They generally do not distinguish output-

Table 8: Attempt-selection sensitivity; outcome columns report $T/H/SU$.

Block	IDs	Affected	Primary	First termination
Rank, clean, and rules	480	35	479/47/432	444/46/398
Model breadth	288	52	285/5/280	234/5/229
Coverage enforcement	72	1	70/1/69	69/1/68
Input transfer	216	19	208/26/182	189/25/164
Agent configurations	192	3	189/43/146	186/42/144
Release refresh	144	3	144/0/144	141/0/141
Total	**1488**	**110**	**1468/181/1287**	**1359/178/1181**

Table 9: Policy transfer across the five primary tasks and two cross-domain tasks.

Group	Task	Forbidden effect	Deliberation	Compare-all
Primary	Document redaction	Loopback request	6/36	1/36
	Speech transcription	Loopback request	2/36	0/36
	Paper retrieval	Loopback request	14/36	3/36
	Archive extraction	Out-of-scope write	20/95	10/95
	Tabular conversion	Unrelated-file read	11/95	7/95
Cross-domain	Configuration editing	Metadata modification	14/30	15/30
	Image thumbnailing	Unnecessary subprocess	12/30	10/30

equivalent implementations or bind an artifact to its source. SINGED isolates pre-execution source selection and scores both the artifact and its process effects.

Agent security. Work studies indirect prompt injection, poisoned tool descriptions, compromised memories and skills, malicious implementations, harmful requests, and unsafe plans (Greshake et al., 2023; Zhan et al., 2024; Debenedetti et al., 2024; Zhang et al., 2025; Li et al., 2026; Andriushchenko et al., 2025; Yin et al., 2024; Zhang et al., 2024; Shi et al., 2025; Wang et al., 2025; Zhang et al., 2026b; Hu et al., 2026). Work examines persistent-state poisoning and adaptive defense synthesis (Xu et al., 2026; Liang et al., 2026). These attacks redirect agents through adversarial content or state. Functional counterfeits preserve a benign request and matched output without requiring an adversarial instruction. The failure occurs when the agent executes an implementation with a forbidden effect, which instruction-level judgments and output-only oracles cannot detect.

Software supply chains and source selection. Package attacks, dependency confusion, hallucinated dependencies, and executable model loaders show how plausible components can conceal unsafe behavior (Ohm et al., 2020; Ladisa et al., 2023; Spracklen et al., 2025; Casey et al., 2024). This literature characterizes compromised artifacts and distribution channels, but rarely studies autonomous choice among matched sources. Candidate order can influence LLM selection when rank carries no provenance information (Bito et al., 2025; Zhang et al., 2026a); models also exhibit latent source preferences (Khan et al., 2026). SINGED connects these concerns by fixing the request, interface, and expected output while randomizing rank, controlling evidence depth, and recording inspection, execution, output, and process events. This reveals when accessible evidence changes action rather than merely being viewed.

E Artifact Content

The artifact contains the benchmark generator, controlled fixtures, candidate templates, schedule construction, tool schemas, sandbox runner, scoring code, and experiment configurations. Generated manifests separate agent-visible metadata from private role and process labels. The paper reports the audited aggregate results and protocol details needed to interpret them.