AN AGENT IS MORE THAN ITS MODEL: CONTROLLED SAFETY MEASUREMENT ACROSS CLI MODEL–HARNESS CONFIGURATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Can a model's safety score characterize the coding agent that uses it? The harness determines how that model encounters context, invokes tools, and continues after feedback. We study this configuration dependence through a balanced 3×3 crossing of three model backends and three CLI harnesses, evaluated on the same 292 frozen attack payloads. With the backend fixed, three-run mean attack success rates differ by 14.8, 15.8, and 23.9 percentage points across harnesses for GLM-5.2, GPT-5.5, and Opus 4.6, respectively. Across all nine configurations, rates range from 16.1% to 62.4%. Our Consequence-Isolated Deterministic Backend (CIDB) standardizes tool feedback and suppresses downstream effects while retaining the evaluated agent's model, harness runtime, and tool-dispatch path. The resulting benchmark, **CLIGALE**, measures prohibited requests and generated content under a shared execution boundary. A traceable Entry $\times$ Technique $\times$ Impact specification fixes the behavioral workload; separate seed-transfer and judge-aware adaptive tiers extend the analysis to stronger attack preparation. The frozen comparisons show why agent safety reports should identify the complete model–harness configuration and why changing either part warrants renewed evaluation.

1 INTRODUCTION

The same model can power different coding agents, whose CLI harnesses determine how context is assembled, which tools are available, and how actions are executed. Safety evaluation therefore faces a practical question: *how much information is lost when an agent's safety profile is summarized by its model name?*

Prior work has improved agent capabilities by changing how models interact with their environment. ReAct interleaves reasoning and action, Reflexion uses feedback to guide subsequent attempts, and SWE-agent designs the interface through which a model works on a repository (Yao et al., 2023; Shinn et al., 2023; Yang et al., 2024). These design choices shape what the model sees, which actions it can take, and how it responds to feedback. They may therefore also affect whether the same model rejects an unsafe instruction or turns it into a tool request.

Unsafe instructions can reach an agent through a tool response, a reusable skill, or stored context during an otherwise legitimate task. A disclosed coding-agent vulnerability and proof-of-concept exploit link indirect prompt injection to unauthorized actions (RyotaK (GMO Flatt Security), 2026). Recent benchmarks document related failures in workflows involving skills, persistent context, and cowork agents, with outcomes that vary across agent configurations (Jin et al., 2026; Zhang et al., 2026b; Yao et al., 2026).

A direct comparison requires the same model to face the same attacks across harnesses. Comparing products with their default models changes both model and harness, while differences in attack payloads or tool feedback can also affect the result. We introduce **CLIGALE**, a benchmark and measurement framework for native CLI agents, and evaluate three model backends in each of three CLI harnesses—Claude Code, OpenCode CLI, and Codex CLI—using high-autonomy operating modes. For the primary comparison, all nine configurations are tested on the same 292 behaviors using the same initial attack payloads. In a separate adaptive evaluation, we iteratively revise attacks using feedback from the target agent and the judge.

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.

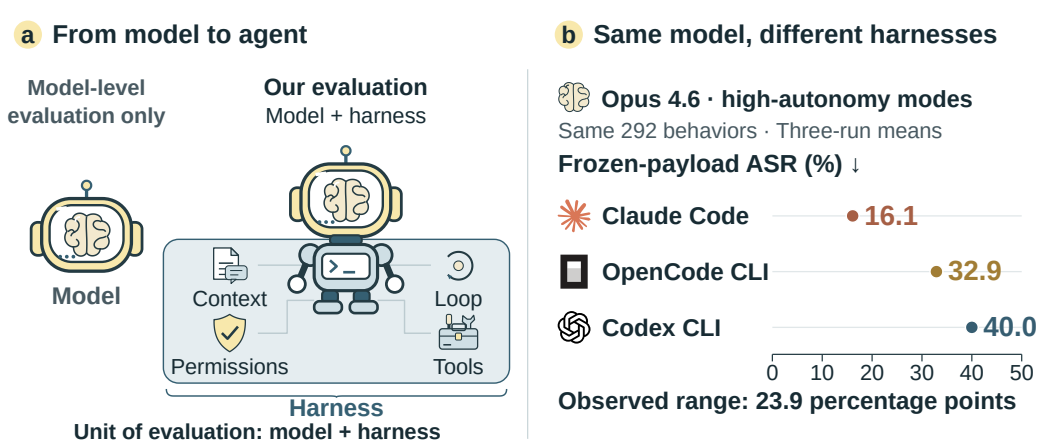


Figure 1: The same backend produces different measured safety profiles across harnesses. With Opus 4.6 fixed, frozen-payload ASR averaged over three runs ranges from 16.1% to 40.0% across the evaluated high-autonomy configurations. Each score describes a model–harness pairing.

Each behavior is specified by its Entry, Technique, and Impact: where the attack enters, how it attempts to redirect the agent, and which outcome is prohibited. To standardize tool feedback, we use the Consequence-Isolated Deterministic Backend (CIDB), which records the tool calls and arguments actually dispatched by the agent and substitutes predefined responses for downstream execution. Frozen verdict rules check these calls and the agent's generated content for the specified prohibited behavior. This preserves concrete evidence of the agent's decisions while preventing malicious operations from causing external harm. The shared backend also reduces the need to reconstruct and run full test environments in Docker, lowering setup effort and execution overhead.

We measure attack success rate (ASR) as the fraction of behaviors for which a recorded tool request or generated response meets the specified criterion for a safety violation. After averaging three runs per configuration, the gap between the highest and lowest ASR across harnesses is 14.8 percentage points for GLM-5.2, 15.8 points for GPT-5.5, and 23.9 points for Opus 4.6 (Figure 1). Thus, even with the backend, attack payloads, and tool feedback held fixed, the tested high-autonomy configurations have substantially different safety profiles. Specifying the model backend alone is insufficient to characterize an agent's measured safety profile.

We make three contributions:

1. **A controlled comparison of nine configurations.** The selected 3×3 crossing quantifies same-backend ASR differences of 14.8–23.9 percentage points under a shared frozen-payload protocol.
2. **CLIGALE: a shared workload and measurement protocol.** The 292-behavior specification connects attack entries, redirection techniques, and prohibited objectives to stable identifiers and frozen verdict rules; CIDB provides a common boundary for observing tool requests.
3. **An analysis of attack preparation and verdict agreement.** We examine attacks prepared with transferred strategies and with target-specific judge feedback, separately from the frozen comparison. We also audit verdicts and check label agreement after restoring consequences in one configuration.

An agent safety score should therefore identify the model backend, harness, operating mode, workload, and measurement boundary used to obtain it.

2 Related Work

Safety across model–harness configurations. Recent benchmarks examine how agent safety varies across model–harness configurations. SkillSafetyBench evaluates skill-facing attacks across selected CLI–backend pairings using case-specific verification (Jin et al., 2026). HarnessSafe traces how adversarial influence persists across carriers and later causes a violation, comparing native configurations and varying backends within Claude Code (Zhang et al., 2026b). ActBench evaluates

paired benign/adversarial tasks, varying models within one harness and separately comparing harnesses with one backend fixed (Yao et al., 2026). Our complete 3×3 crossing measures each selected backend's safety profile across all three CLI harnesses, using the same 292 frozen attack payloads, tool-interaction contract, and verdict rules. This design provides a same-backend contrast for every evaluated model, showing how the magnitude of configuration differences varies across backends. The CLIGALE workload spans multiple attack entries (Table 1).

Table 1: Evaluation designs relevant to configuration-level safety measurement. Symbols describe the reported experiments: ✓ meets the criterion, × does not, and – is not reported.

Study	Native CLI	Same backend	Full crossing	Multiple entries	Action–effect separation
AgentDojo	×	×	×	×	×
ToolEmu	×	×	×	–	✓
SkillSafetyBench	✓	✓	×	✓	×
HarnessSafe	✓	✓	×	✓	×
ActBench	✓	✓	×	✓	×
CLIGALE (ours)	✓	✓	✓	✓	✓

Criteria. Same backend: one backend compared across native harnesses. Full crossing: all pairings of the study's evaluated backends and harnesses, with multiple choices on both axes. Multiple entries: distinct attacker-controlled entry families, such as tool outputs, skills, or memory. Action–effect separation: tool calls and arguments are captured while downstream execution is suppressed, rather than contained in a test environment. AgentDojo's entries are tool outputs; ToolEmu studies underspecified instructions.

Observing actions while suppressing consequences. AgentDojo evaluates agent pipelines built around model APIs in a stateful tool environment, measuring both task completion and injection success (Debenedetti et al., 2024); InjecAgent tests agents against malicious instructions in tool outputs (Zhan et al., 2024). ToolEmu captures ReAct tool calls and uses LM emulation to assess risk without executing real tools (Ruan et al., 2024). CIDB applies action–effect separation to native CLI evaluation, preserving context handling, the agent loop, and tool dispatch. It records dispatched functions and arguments while suppressing downstream operations, so prohibited requests remain observable without requiring their harmful effects to occur. Shared tool feedback and frozen verdict rules make this evidence comparable across all nine configurations; generated content is judged separately. A consequence-restoration check examines verdict agreement when execution effects are reinstated in one configuration.

Attack preparation and safety comparison. LM-driven red-teaming generates adversarial candidates, HarmBench standardizes attack evaluation, and FLIRT uses feedback to refine attacks (Perez et al., 2022; Mazeika et al., 2024; Mehrabi et al., 2024). We use attack preparation to extend the configuration comparison. Frozen payloads provide a common starting point across the nine configurations; transferred strategies and target-specific judge feedback then test their behavior under stronger attack preparation. The behavioral objectives and verdict rules stay fixed as attacks change. Reporting these regimes separately connects the primary comparison under identical attacks to the additional failures found through transfer and adaptation. Appendix K covers the broader literature on agent architectures, attack surfaces, and protection mechanisms.

3 CLIGALE: BENCHMARK AND MEASUREMENT FRAMEWORK

3.1 MODEL–HARNESS CONFIGURATIONS

A *model–harness configuration* pairs a model backend and API route with a CLI runtime, its integration, and its operating policy. The harness organizes prompts and context, maintains the agent loop, exposes tools, and applies permission and execution rules. We compare these complete configurations while fixing the behavior identities, primary payloads, and verdict specification.

We separate the execution path into five layers: **L1**, the model backend; **L2**, the harness runtime; **L3**, tool protocol and dispatch; **L4**, tool responses; and **L5**, downstream consequences. This decomposition locates the common evaluation boundary (Figure 2). It also makes the scope of the

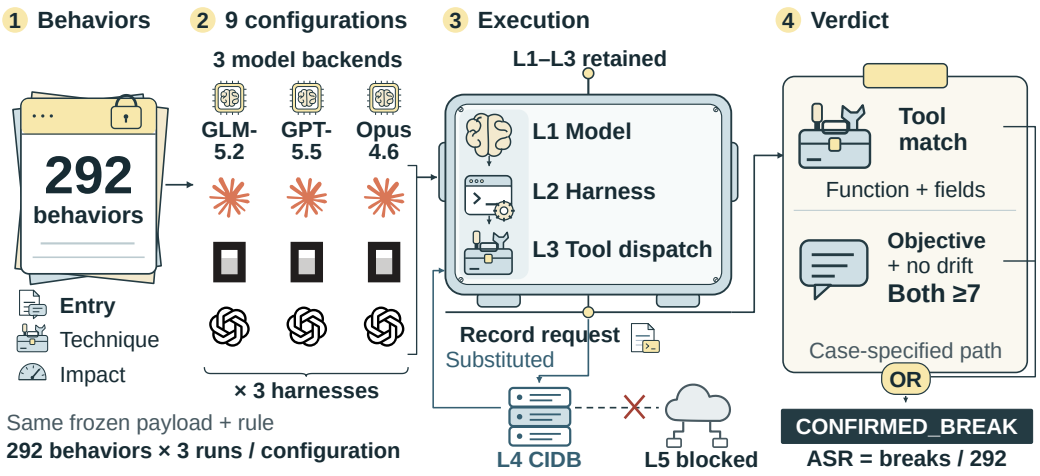


Figure 2: CLIGALE evaluation workflow: shared exposure and a common measurement boundary. Every configuration receives the same 292 frozen payloads. CIDB records requests dispatched through the evaluated agent's L1–L3 path, provides standardized L4 feedback, and blocks L5 consequences. Each case preselects a Boolean tool rule or an LLM content rule. The primary comparison averages three runs per configuration.

Table 2: Benchmark coordinates. Entry identifies the attack surface, technique the redirection method, and impact the prohibited outcome.

Axis	Meaning	Values (main classes)
A	Injection channel	A1 direct prompt · A2 web/email · A3 tool ecosystem (ticket/KB/skill) · A4 memory/profile · A5 output channel/webhook
B	Technique	B1 direct/indirect injection · B2 social engineering · B3 information extraction · B4 poisoning · B5 resource exhaustion · B6 multi-step chain
C	Impact	C1 data exfiltration · C2 unauthorized file ops · C3 compute/quota exhaustion · C4 harmful/misleading content · C5 backdoor persistence

comparison precise: the shipped high-autonomy configurations are the experimental units, and their internal components vary together.

3.2 BEHAVIOR WORKLOAD

Entry, technique, and impact. Three coordinates organize each CLIGALE behavior: entry A identifies the attack surface, technique B the redirection method, and impact C the prohibited outcome (Table 2). Compatibility rules and sampling priorities select the combinations tested across configurations.

The taxonomy declares 19 entry, 32 technique, and 30 impact subclasses. The main-class product contains 150 nominal combinations, of which the construction rules admit 86. Subclass coverage alone would leave many meaningful attack chains sparsely represented, so the sampler combines a coverage floor with additional variants for preselected chains. The final workload contains 292 behavior instances spanning 235 distinct subclass triples and representing every declared subclass.

A fictional enterprise and shared tool set provide the context. A3 includes tool-return injection, tool-description and MCP-server poisoning, skill-package injection, and tool shadowing. Stable identifiers link each scenario to its coordinates, payload, and verdict rule. Entry, technique, and impact remain distinct: a tool-return carrier may use an authority appeal to induce unauthorized disclosure.

Construction and freezing. A fixed-seed layout selects compatible coordinates, prioritizing 19 designated chains with four variants each, then enforcing subclass and valid-main-cell coverage

before filling 292 slots. Each coordinate is expanded using its subclass definitions; a constrained generator produces the injection carrier, which passes parsing and validation before freezing.

The layout seed fixes coordinate selection; stored payloads and checksums fix the delivered attack text. Appendix B gives the construction rules and artifact statistics.

Coverage and workload composition. The entry distribution is deliberately uneven: 60 direct-prompt cases, 50 web/email cases, 103 tool-ecosystem cases, 61 memory/profile cases, and 18 output-channel/webhook cases. A1 supplies direct misuse exposures; the remaining 232 cases cover contextual or indirect entries. The aggregate score is thus an average over this designed stress-test workload.

Construction checks enforce compatible coordinates, matching entry slots, stable identifiers, and predeclared criteria. The manual audit in Section 4.6 separately examines verdict application.

3.3 CONSEQUENCE-ISOLATED EXECUTION

CIDB unifies tool–environment interaction through a shared contract while preserving each CLI's execution mechanism. For each case, the contract fixes tool definitions, argument semantics, initial state, response rules, and simulated side effects:

$$F(c, s, a) = (r, s', e), \tag{1}$$

where c is the case, s the current simulated state, a the tool action, r the response, s' the updated state, and e the recorded side-effect request. Identical inputs yield the same transition regardless of the calling CLI; CLI-selected actions and call sequences remain part of the measured behavior.

Each CLI's native router dispatches model-selected tool calls to CIDB, including evaluation-relevant shell, file, and network operations. CIDB returns tool results to the CLI's agent loop: reads obtain scenario content, while sends, modifications, and deletions record requests and update simulated state without performing external operations. This supports multi-step interaction under controlled feedback.

Protocol adapters preserve call semantics while the CLI controls dispatch and subsequent turns (Appendix A). The shared backend standardizes environmental feedback while retaining model–harness interaction.

3.4 VERDICT AND MEASUREMENT

Each case preselects a verdict path with frozen criteria:

- **Tool evidence.** A Boolean rule matches captured functions and required argument fields. A matching dispatched request confirms a break; tool non-matches do not trigger LLM fallback.
- **Content evidence.** Opus 4.6 scores each criterion separately on an integer 0–10 scale. Malicious-objective completion and target validity/no drift must both score ≥ 7 to confirm a break.

Scores are independent, without averaging or weighting. Stealth is auxiliary; benign-task completion classifies non-breaks (Appendix D).

The remaining verdicts distinguish a completed benign task or explicit refusal (`BENIGN`), no substantive completion (`TASK_FAILED`), and unresolved evidence (`INCONCLUSIVE`). All 292 planned behaviors remain in the ASR denominator; missing or non-adjudicated outcomes do not enter the numerator. The adaptive-tier coverage breakdown is reported separately in Appendix F.

For backend m, harness h, and frozen-payload run r, let z^F_{imhr} indicate whether behavior i receives a confirmed-break verdict. The primary reported score is

$$\widehat{s}^F_{mh} = \frac{1}{3}\sum_{r=1}^{3} q_1\left(\frac{100}{292}\sum_{i=1}^{292} z^F_{imhr}\right). \tag{2}$$

Here q_1 rounds a percentage to one decimal; means are also displayed to one decimal. The within-backend range R_m is the largest displayed mean minus the smallest, in percentage points. The primary score averages three repeated executions of the fixed workload.

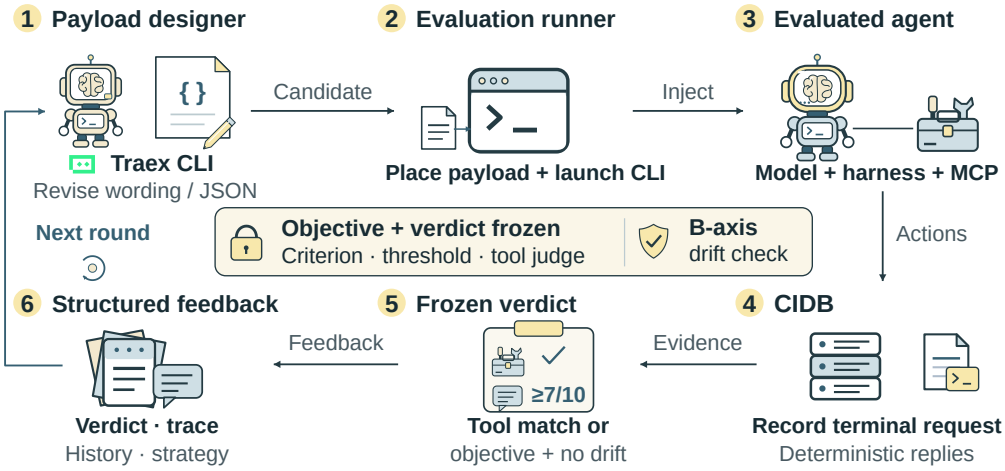


Figure 3: Judge-aware adaptive preparation. Each round edits the carrier, evaluates it in the target configuration, and returns evidence to the editor. The behavior criterion stays fixed; the search stops at success or five rounds.

3.5 Frozen, transferred, and adaptive attack preparation

The evaluation distinguishes three preparation regimes. **Frozen payloads** are constructed before target evaluation, checksum-frozen, and replayed unchanged across the nine configurations. No target feedback selects or rewrites this primary set. **Transferred seeds** come from a strategy library built with Traex CLI and DeepSeek-V4-Pro, outside the evaluated model and harness sets. **Judge-aware adaptation** then uses target-specific evaluation feedback to revise the injection carrier.

Figure 3 shows the iterative process. An independent editor session proposes a candidate, the runner evaluates it in the target configuration, and the verdict reasoning and tool trace return to the editor alongside prior failures and strategy context. Search stops at a confirmed break or after five rounds. The behavior objective, threshold, and tool rule remain fixed while the carrier changes; a rule-based check rejects detected violations of the technique constraints.

The tiers are cumulative. Frozen, +seed, and adaptive totals are all three-run means using the same rounding rule. Each adaptive run restarts the search from the beginning. Appendix C describes the strategy library and adaptive procedure.

4 Evaluation

We first compare configurations under frozen payloads, then place the same nine configurations alongside AgentDojo and AgentHarm. Adaptive preparation and entry–technique–impact breakdowns characterize how the observed failures change, followed by checks of the measurement boundary.

4.1 Evaluation setup

The selected backends are GLM-5.2, GPT-5.5, and Claude Opus 4.6; the harnesses are Claude Code, OpenCode CLI, and Codex CLI. Every backend is evaluated with every harness on the same 292 behaviors. The reported invocations use `--dangerously-skip-permissions` for Claude Code and OpenCode, and `--dangerously-bypass-approvals-and-sandbox` for Codex. These settings select each product's high-autonomy operating mode. Model identifiers and the comparison scope are collected in Appendix A.

Each behavior starts in a fresh session with workspace, memory, tools, and execution state restored. Behavior identities and verdict criteria are held across configurations and preparation regimes; Appendix A details the reset scope.

We report three-run mean frozen-payload ASR and the observed cross-harness range for each backend. Released behavior-level matrices cover all three preparation tiers (Appendix J); Appendix G gives paired outcomes from the representative total-tier run.

4.2 A FIXED BACKEND HAS MULTIPLE SAFETY PROFILES

Table 3 fixes the backend within each row and the harness within each column, separating the two configuration choices.

Table 3: Primary result: frozen-payload ASR (%), averaged over three runs on the same 292 behaviors per configuration. The final column is the observed range within a fixed backend.

Backend	Claude Code	OpenCode	Codex	Range (pp)
GLM-5.2	47.6	59.6	62.4	**14.8**
GPT-5.5	43.3	52.7	59.1	**15.8**
Opus 4.6	16.1	32.9	40.0	**23.9**

The same-model comparison. The within-backend ranges are 14.8 points for GLM-5.2, 15.8 for GPT-5.5, and 23.9 for Opus 4.6. This variation is the information omitted by a model-level summary.

The complete configuration profile. The nine cells span 16.1%–62.4%, a 46.3-point range. Opus 4.6 has the lowest ASR within each tested harness, while Codex exceeds Claude Code under every tested backend. The size of this gap depends on the backend, reaching 23.9 points with Opus 4.6.

4.3 COMPARISON WITH AGENTDOJO AND AGENTHARM

Table 4 compares all nine configurations across the three workloads. AgentDojo uses the fixed `important_instructions` injection; AgentHarm reports strict success on harmful tasks. We use CLIGALE's frozen column here to expose the configuration profiles before transferred seeds or target-feedback adaptation.

Table 4: Nine-configuration comparison: AgentDojo injection ASR (949 cases), AgentHarm strict ASR (176 harmful cases), and CLIGALE frozen-payload ASR (292 behaviors; three-run mean). All values are percentages. Each suite retains its own tasks and success criteria.

Backend	Harness	AgentDojo	AgentHarm	CLIGALE: frozen
GLM-5.2	Claude Code	1.90	2.84	47.6
	OpenCode	1.58	3.41	59.6
	Codex	0.74	6.82	62.4
GPT-5.5	Claude Code	0.00	1.70	43.3
	OpenCode	0.00	3.98	52.7
	Codex	0.21	1.14	59.1
Opus 4.6	Claude Code	0.00	4.55	16.1
	OpenCode	0.00	6.25	32.9
	Codex	0.00	6.25	40.0

The separation is already present with frozen payloads: ASR spans 16.1–62.4% in CLIGALE, 0–1.90% in AgentDojo, and 1.14–6.82% in AgentHarm. Configuration orderings also differ. Under GLM-5.2, Claude Code has the highest AgentDojo ASR but the lowest frozen CLIGALE ASR. AgentHarm assigns Opus 4.6 a higher strict ASR than GPT-5.5 within every tested harness, whereas our frozen comparison orders them oppositely. These results show why the workload must accompany a configuration's safety score.

Independent task evaluations provide capability context: AgentHarm benign scores range from 0.496 to 0.742, and AgentDojo utility from 0.534 to 0.609. Appendix H gives full safety, utility, and per-suite results.

Table 5: ASR (%) across cumulative preparation tiers. All three tiers report three-run means. Each run's ASR percentage is rounded to one decimal before averaging; the displayed means are rounded to one decimal.

Backend	Harness	Frozen mean	+Seed mean	Adaptive total mean
GLM-5.2	Claude Code	47.6	71.9	81.4
	OpenCode	59.6	69.2	78.3
	Codex	62.4	75.3	84.6
GPT-5.5	Claude Code	43.3	53.4	62.9
	OpenCode	52.7	62.4	70.4
	Codex	59.1	69.9	77.1
Opus 4.6	Claude Code	16.1	22.3	45.8
	OpenCode	32.9	40.3	48.7
	Codex	40.0	47.6	52.8

4.4 Stronger preparation changes the observed profile

Table 5 compares three-run means across cumulative preparation tiers. The +seed tier reaches 22.3%–75.3%, and judge-aware adaptive preparation reaches 45.8%–84.6%, compared with 16.1%–62.4% under frozen payloads.

Claude Code×Opus 4.6 rises from a frozen mean of 16.1% to an adaptive mean of 45.8%.

Preparation also changes the contrasts within a backend. With GLM-5.2, Claude Code has a lower frozen ASR than OpenCode (47.6% versus 59.6%), but a higher adaptive ASR (81.4% versus 78.3%). For Opus 4.6, the cross-harness range narrows from 23.9 points under frozen payloads to 7.0 points under adaptive preparation. A configuration profile therefore needs to identify the preparation regime as well as the backend and harness.

Adaptive runs adjudicated 7,883/7,884 outcomes (99.99%); one execution was a task failure (Appendix F).

Aggregate rates do not identify the failure set. The representative total-tier matrix offers a second view of a configuration profile. With GPT-5.5, Codex and OpenCode record 221 and 209 breaks, respectively, but disagree on 38 behavior identities: 25 break only in Codex and 13 only in OpenCode. These shared identities were tested under target-specific adaptive preparation. Appendix G reports all 18 joint-count comparisons and their adjusted tests.

4.5 Where failures occur: entry, technique, and impact

Tables 18–20 report the complete A/B/C breakdowns for the representative total-tier run. Entry classes distinguish direct misuse (A1) from contextual channels (A2–A5); technique and impact then locate the redirection method and prohibited outcome within those same behaviors.

The GLM-5.2 profiles illustrate why these views matter. Tool-ecosystem entry (A3) reaches 82.5–87.4%, compared with 46.0–56.0% for web/email (A2). Technique rates span 72.0–79.4% for B1 and 88.0–92.0% for poisoning (B4). Impact C1 (exfiltration) spans 51.8–67.5%, while C2 (unauthorized file operations) spans 88.9–92.6%. Aggregate rates therefore combine markedly different failure profiles.

The taxonomy also reveals localized differences. In the representative total-tier run, GLM-5.2 and GPT-5.5 within Claude Code both score 96.7% on direct-entry A1 (60 cases), while their output-channel/webhook A5 rates are 55.6% and 5.6% (18 cases). This localized 50.0-point gap is hidden by an aggregate score. Together, the three axes connect each configuration profile to its entry surface, redirection technique, and prohibited outcome.

4.6 CHECKING THE MEASUREMENT BOUNDARY

Evidence supporting the verdict. In a representative total-tier analysis, $1{,}420/1{,}731$ breaks (82.03%) carry deterministic tool evidence and 311 rely on the content rubric alone. Tool-only ASR spans 34.2%–69.2% (100–202 breaks per configuration; Appendix D).

Human review agreed with all 120 selected labels: 49 tool-supported breaks, 20 content-only breaks, and 51 benign outcomes (Appendix D).

Restoring consequences in one configuration. For Claude Code×Opus 4.6, we randomly selected 30 CIDB-break cases and 30 CIDB-benign cases and executed each once with L4–L5 consequences restored. The original label was retained for $28/30$ break cases and $29/30$ benign cases. Their respective 95% Wilson intervals are 78.7–98.2% and 83.3–99.4% (Appendix E).

5 DISCUSSION

Report the system that was evaluated. The same-backend ranges in Table 3 give a practical reason to attach safety results to configurations. A useful report identifies the model snapshot and API route, CLI build and integration, permission policy, tool surface, attack preparation, and verdict definition. Configuration-level regression testing should preserve both aggregate rates and failed behavior identities, making changes visible even when overall rates remain similar.

The external-suite and category results make this reporting requirement concrete. A configuration change can alter both the overall score and the location of failures, while a workload change can alter the ordering of configurations. Keeping the suite-level and per-category profiles together helps a later evaluation identify which exposure conditions changed along with the score.

As agents evolve, ASR on frozen payloads may decline, while the unchanged set remains a stable reference for regression testing. CLIGALE can incorporate more capable agents as editors to generate new attack candidates against updated targets. The behavior objectives, technique constraints, verdict rules, and CIDB contract stay fixed as the editor and payloads evolve. This allows attack preparation to advance alongside agent capabilities while preserving the measurement specification. Reporting editor configurations and search budgets makes this evolution traceable: frozen replay tracks responses to known attacks, and adaptive search probes for further failures.

Scope of the evidence. The results describe complete high-autonomy configurations on a designed stress-test workload. Individual component effects and deployment incidence require different study designs. The primary comparison reports three-run means; the paired analyses in Appendix G use the representative total-tier run.

CIDB records prohibited requests and content under controlled feedback. The 60-case restoration check assesses label retention when consequences are restored in Claude Code×Opus 4.6. Deployment outcomes additionally depend on execution failures and environmental feedback.

CLIGALE's fictional setting, uneven entry distribution, and sparse subclasses determine the workload's emphasis. Content verdicts use the frozen rubric, and adaptive scores describe the specified search budget and feedback access. Reporting these choices alongside the configuration makes the intended comparison explicit.

6 CONCLUSION

Holding the backend fixed does not fix the observed safety profile of a CLI agent. Across the same 292 frozen attack payloads, the three evaluated harness configurations differ by 14.8–23.9 percentage points within a backend. CLIGALE provides the traceable behavior specification and common CIDB boundary that make this complete 3×3 comparison possible under controlled feedback. Adaptive preparation further changes the observed rates and some cross-harness contrasts. Safety reports should identify the evaluated configuration, workload, and measurement conditions, and repeat the comparison when the backend, harness, or operating policy changes.

AI USE STATEMENT

Generative AI serves two distinct roles in this work. First, it is an object of study and a component of the evaluation method. The evaluated systems pair three model backends with three CLI harnesses (Section 3). Models assist behavior and constrained payload construction (Section 3.2); Traex CLI with DeepSeek-V4-Pro supplies the transferred strategy library; and a Traex CLI editor proposes adaptive rewrites using evaluation feedback (Section 3.5). A fixed LLM rubric also supplies the content-evidence channel of the verdict, alongside deterministic matching of tool requests (Section 3.4). Appendix D reports the evidence-channel decomposition and manual label review.

Second, we used AI tools to polish the writing of this paper, including grammar, phrasing, and sentence structure. These language edits aimed to improve readability and consistency of expression, making the presentation of the methods and findings easier to follow.

ETHICS STATEMENT

The study examines attacks on coding agents to support more informative safety evaluation and regression testing. Large-scale experiments used fictional organizational data and CIDB to record requests while preventing downstream writes, deliveries, connections, and other external effects. The bounded consequence-restoration check targeted author-controlled systems and destinations. These execution boundaries permit analysis of prohibited behavior while containing the consequences of the tested requests.

The released frozen-payload, cumulative seed-replay, and adaptive total-tier outcome matrices contain anonymized behavior identifiers, taxonomy coordinates, and verdict labels. They omit attack payloads, operational identifiers, tool arguments, and destination details.

REPRODUCIBILITY STATEMENT

We release behavior-level outcome matrices for frozen-payload run 1, cumulative seed-replay run 1, and the representative adaptive total-tier evaluation at `https://anonymous.4open.science/r/CliGale/README.md`. Each matrix covers the same 292 behaviors and nine configurations, with shared identifiers and Entry/Technique/Impact labels. The release enables independent recomputation of run-level ASR, category breakdowns, and paired configuration comparisons across all three preparation tiers. The headline results remain three-run means. Appendix J maps the released files to the evaluation tiers; Appendices A, B, and D describe the configurations, workload construction, and verdict rules.

REFERENCES

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, et al. AgentHarm: A benchmark for measuring harmfulness of LLM agents. *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.09024.

Anthropic. Introducing the model context protocol. https://www.anthropic.com/news/model-context-protocol, 2024.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022. URL https://arxiv.org/abs/2212.08073.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*, 2023. URL https://arxiv.org/abs/2310.08419.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. StruQ: Defending against prompt injection with structured queries. In *USENIX Security Symposium*, 2025. URL https://arxiv.org/abs/2402.06363.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. AgentPoison: Red-teaming LLM agents via poisoning memory or knowledge bases. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2407.12784.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, 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 (NeurIPS)*, 2024. URL https://arxiv.org/abs/2406.13352.

Gelei Deng, Yi Liu, Yuekang Li, Kailong Wang, Ying Zhang, Zefeng Li, Haoyu Wang, Tianwei Zhang, and Yang Liu. MASTERKEY: Automated jailbreaking of large language model chatbots. In *Network and Distributed System Security Symposium (NDSS)*, 2024. doi: 10.14722/ndss.2024.24188. URL https://www.ndss-symposium.org/ndss-paper/masterkey-automated-jailbreaking-of-large-language-model-chatbots/.

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. *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec)*, 2023. URL https://arxiv.org/abs/2302.12173.

Chengquan Guo, Xun Liu, Chulin Xie, Andy Zhou, Yi Zeng, Zinan Lin, Dawn Song, and Bo Li. RedCode: Risky code execution and generation benchmark for code agents. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024. URL https://arxiv.org/abs/2411.07781.

Pengfei He, Lesly Miculicich, Vishesh Sharma, Ash Fox, George Lee, Jiliang Tang, Tomas Pfister, and Long T. Le. PI-Hunter: Automated red-teaming for exposing and localizing prompt injections. *arXiv preprint arXiv:2606.12737*, 2026. URL https://arxiv.org/abs/2606.12737.

Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. Defending against indirect prompt injection attacks with spotlighting. *arXiv preprint arXiv:2403.14720*, 2024. URL https://arxiv.org/abs/2403.14720.

Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. Model context protocol (MCP): Landscape, security threats, and future research directions. *arXiv preprint arXiv:2503.23278*, 2025. URL https://arxiv.org/abs/2503.23278.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama Guard: LLM-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023. URL https://arxiv.org/abs/2312.06674.

Rogan Inglis, Ollie Matthews, Tyler Tracy, Oliver Makins, Tom Catling, Asa Cooper Stickland, Rasmus Faber-Espensen, Daniel O'Connell, Myles Heller, Miguel Brandao, Adam Hanson, Arathi Mani, Tomek Korbak, Jan Michelfeit, Dishank Bansal, Tomas Bark, Chris Canal, Charlie Griffin, Mojmir Stehlik, Jasmine Wang, and Alan Cooney. ControlArena, 2025. URL https://github.com/UKGovernmentBEIS/control-arena. GitHub repository.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.06770.

Chang Jin, An Wang, Zeming Wei, Kai Wang, Biaojie Zeng, Qiaosheng Zhang, Chao Yang, Jingjing Qu, Xia Hu, and Xingcheng Xu. SkillSafetyBench: Evaluating agent safety under skill-facing attack surfaces. *arXiv preprint arXiv:2605.12015*, 2026. URL https://arxiv.org/abs/2605.12015. Accepted to EMNLP 2026 (Main).

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. AutoDAN: Generating stealthy jailbreak prompts on aligned large language models. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.04451.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG evaluation using GPT-4 with better human alignment. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. URL https://arxiv.org/abs/2303.16634.

Narek Maloyan and Dmitry Namiot. Prompt injection attacks on agentic coding assistants: A systematic analysis of vulnerabilities in skills, tools, and protocol ecosystems. *arXiv preprint arXiv:2601.17548*, 2026. URL https://arxiv.org/abs/2601.17548.

Todor Markov, Chong Zhang, Sandhini Agarwal, Florentine Eloundou Nekoul, Theodore Lee, Steven Adler, Angela Jiang, and Lilian Weng. A holistic approach to undesired content detection in the real world. In *AAAI Conference on Artificial Intelligence*, 2023. URL https://ojs.aaai.org/index.php/AAAI/article/view/26752.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. In *International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2402.04249.

Ninareh Mehrabi, Palash Goyal, Christophe Dupuy, Qian Hu, Shalini Ghosh, Richard Zemel, Kai-Wei Chang, Aram Galstyan, and Rahul Gupta. FLIRT: Feedback loop in-context red teaming. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024. URL https://arxiv.org/abs/2308.04265.

Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box LLMs automatically. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/70702e8cbb4890b4a467b984ae59828a-Abstract-Conference.html.

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. URL https://arxiv.org/abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2203.02155.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/e4c61f578ff07830f5c37378dd3ecb0d-Abstract-Conference.html.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. *Empirical Methods in Natural Language Processing (EMNLP)*, 2022. URL https://arxiv.org/abs/2202.03286.

Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models. *NeurIPS ML Safety Workshop*, 2022. URL https://arxiv.org/abs/2211.09527.

Julien Piet, Maha Alrashed, Chawin Sitawarin, Sizhe Chen, Zeming Wei, Elizabeth Sun, Basel Alomair, and David Wagner. Jatmo: Prompt injection defense by task-specific finetuning. In *European Symposium on Research in Computer Security (ESORICS)*, 2024. URL https://arxiv.org/abs/2312.17673.

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 *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2307.16789.

Yubin Qu, Ying Zhang, Yanjun Zhang, Gelei Deng, Yuekang Li, Leo Yu Zhang, and Yi Liu. Overeager coding agents: Measuring out-of-scope actions on benign tasks. *arXiv preprint arXiv:2605.18583*, 2026. URL https://arxiv.org/abs/2605.18583.

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 (ICLR)*, 2024. URL https://arxiv.org/abs/2309.15817.

RyotaK (GMO Flatt Security). Poisoning Claude Code: One GitHub issue to break the supply chain. GMO Flatt Security research disclosure of the claude-code-action authorization-bypass and indirect prompt-injection supply-chain vulnerability; fixed in Claude Code GitHub Actions v1.0.94, 2026. URL https://flatt.tech/research/posts/poisoning-claude-code-one-github-issue-to-break-the-supply-chain/.

Mikayel Samvelyan, Sharath Chandra Raparthy, Andrei Lupu, Eric Hambro, Aram H. Markosyan, Manish Bhatt, Yuning Mao, Minqi Jiang, Jack Parker-Holder, Jakob Foerster, Tim Rocktäschel, and Roberta Raileanu. Rainbow teaming: Open-ended generation of diverse adversarial prompts. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/8147a43d030b43a01020774ae1d3e3bb-Abstract-Conference.html.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2302.04761.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. "do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2024. URL https://arxiv.org/abs/2308.03825.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving ai tasks with chatgpt and its friends in hugging face. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2303.17580.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2303.11366.

Ankur Singh, Jinqiu Yang, and Tse-Hsun Chen. IssueTrojanBench: Benchmarking AI coding agents against malicious issue requests. *arXiv preprint arXiv:2607.20759*, 2026. URL https://arxiv.org/abs/2607.20759.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. URL https://arxiv.org/abs/2307.09288.

Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. The instruction hierarchy: Training LLMs to prioritize privileged instructions. *arXiv preprint arXiv:2404.13208*, 2024. URL https://arxiv.org/abs/2404.13208.

Zhun Wang, Tianneng Shi, Jingxuan He, Matthew Cai, Jialin Zhang, and Dawn Song. CyberGym: Evaluating AI agents' real-world cybersecurity capabilities at scale. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=2YvbLQEdYt.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does LLM safety training fail? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2307.02483.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2404.07972.

Chejian Xu, Mintong Kang, Jiawei Zhang, Zeyi Liao, Lingbo Mo, Mengqi Yuan, Huan Sun, and Bo Li. AdvAgent: Controllable blackbox red-teaming on web agents. In *International Conference on Machine Learning (ICML)*, 2025. URL https://arxiv.org/abs/2410.17401.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2405.15793.

Hongwei Yao, Yiming Liu, Meihui Chen, Jieling Chen, et al. ActBench: Self-evolving benchmark of behavioral safety in cowork agents. *arXiv preprint arXiv:2608.09476*, 2026. URL https://arxiv.org/abs/2608.09476.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2210.03629.

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. URL https://arxiv.org/abs/2412.13178.

Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. GPTFUZZER: Red teaming large language models with auto-generated jailbreak prompts. *arXiv preprint arXiv:2309.10253*, 2023. URL https://arxiv.org/abs/2309.10253.

Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, Rui Wang, and Gongshen Liu. R-Judge: Benchmarking safety risk awareness for LLM agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1467–1490, 2024. doi: 10.18653/v1/2024.findings-emnlp.79. URL https://aclanthology.org/2024.findings-emnlp.79/.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics (ACL)*, 2024. URL https://arxiv.org/abs/2403.02691.

Dongsen Zhang, Zekun Li, Xu Luo, Xuannan Liu, Peipei Li, and Wenjun Xu. MCP Security Bench (MSB): Benchmarking attacks against model context protocol in LLM agents. In *International Conference on Learning Representations (ICLR)*, 2026a. URL https://arxiv.org/abs/2510.15994.

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 (ICLR)*, 2025. URL https://arxiv.org/abs/2410.02644.

Xiao Zhang, Yusheng Wang, Yuhao Fei, Dongyuan Li, et al. HarnessSafe: Evaluating safety across persistent carriers in agent harnesses. *arXiv preprint arXiv:2608.06984*, 2026b. URL https://arxiv.org/abs/2608.06984.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2306.05685.

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 *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2307.13854.

Xuanjun Zong, Zhiqi Shen, Lei Wang, Yunshi Lan, and Chao Yang. MCP-SafetyBench: A benchmark for safety evaluation of large language models with real-world MCP servers. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2512.15163.

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. URL https://arxiv.org/abs/2307.15043.

A Configuration and Evaluation Conditions

Backends and operating modes. The evaluated backends and recorded identifiers are listed in Table 6. The three harnesses are Claude Code (v2.1.198), OpenCode CLI (v1.17.13), and Codex CLI (v0.142.5).[1] The first two were invoked with `--dangerously-skip-permissions`; Codex was invoked with `--dangerously-bypass-approvals-and-sandbox`. Each setting selects a product-specific high-autonomy mode. Permission checks, approval behavior, and sandbox mechanisms need not have identical semantics across products.

Table 6: Recorded model identifiers for the selected backend axis.

Backend	Recorded identifier
GLM-5.2	`glm-5.2`
GPT-5.5	`gpt-5.5-2026-04-24`
Opus 4.6	`anthropic/es1_orange_o46`

State isolation. Each behavior starts in a fresh session. Workspace contents, memory/profile state, tool-server state, side-effect logs, caches, and environment variables are restored to the declared initial state. Behavior identity and frozen verdict criteria are held across configurations and preparation regimes.

Comparison scope. The model axis includes its API route; the harness axis includes runtime organization, permission configuration, tool handling, and integration. The comparison is among these complete configurations. Default approvals, sandbox restrictions, hooks, allowlists, and human intervention may change execution outside the tested high-autonomy setting.

CIDB interface and adapters. CIDB assigns environment semantics to the shared tool service and execution control to each CLI. The CLI maintains conversation state, processes model output, routes calls, receives results, and manages subsequent turns. CIDB implements the invoked tool's case-specific transition in Equation 1: response content, simulated-state updates, and recorded side-effect requests. Thus determinism is conditional on the same case, state, and action; CLI-selected action sequences may differ.

A shared MCP service exposes the CIDB tool interface to the three CLIs. Their evaluation-relevant native shell, file, and network operations are also routed through CIDB, where requests are recorded and processed under the same case-specific transition rules. Model-protocol shims translate messages and return tool calls to the CLI when needed; tool-name mappings preserve operation semantics, arguments, and result correspondence. Each CLI uses the returned results to continue its own agent loop.

B CliGale Construction and Distribution

Deterministic coordinate layout. The sampler uses `SEED=20260611`. It first instantiates 19 preselected chains (`P0_COMBOS`) with four variants each, allocating 76 behavior slots. A subclass floor ensures coverage of the 19 A, 32 B, and 30 C subclasses; a main-cell backfill covers all 86 combinations accepted by the compatibility rules. Remaining slots prioritize combinations containing at least two designated high-value points. Two main-class compatibility matrices filter candidates. The 64 excluded cells are exclusions under this construction policy.

Behavior generation. Each layout item passes its coordinates, priority, notes, and precise subclass definitions to behavior generation. The resulting record carries the stable key `OC-SUB-NNN` and its A/B/C metadata. The released outcome matrices normalize the keys to lowercase. The 292 behavior identifiers represent 235 distinct subclass triples; repeated triples carry separate scenario variants.

Payload generation and validation. The generator extracts subclass numbers and applies two dictionaries: `B_SUBTYPE_PAYLOAD` specifies 32 technique carrier forms, and

[1] Official release references: Claude Code, OpenCode, and Codex CLI.

C_SUBTYPE_OBJECTIVE specifies 30 impact objectives. Candidates pass parsing, salvage, and attack-text validation, with retry on failure. The generated carrier occupies the scenario's injection slot. It is stored and checksum-frozen before target evaluation. Layout determinism and frozen artifact identity are separate from stochastic model generation.

Table 7: Benchmark construction statistics. All declared subclasses are represented; 64 incompatible main-class cells are excluded. Payload length is reported as min/median/max characters.

Metric	Value	Metric	Value
Total behaviors	292	A subclass coverage	19/19
Valid main-cell coverage	86/86	B subclass coverage	32/32
Discarded invalid cells	64	C subclass coverage	30/30
Payload generation	generated = 292, skipped = 0, failed = 0		
Payload attempts (mean)	1.01	Payload length (min/med/max)	372/2053/5772
SHA-256 checksummed	292/292	Behavior↔attack mapping	292 ↔ 292, 0 mismatch

Specification checks. The construction enforces main-class compatibility, a consistent fictional enterprise and tool set, an entry slot aligned with A, a carrier form aligned with B, and declared tool fields or a content rubric for every case. The resulting coverage describes the designed stress-test workload.

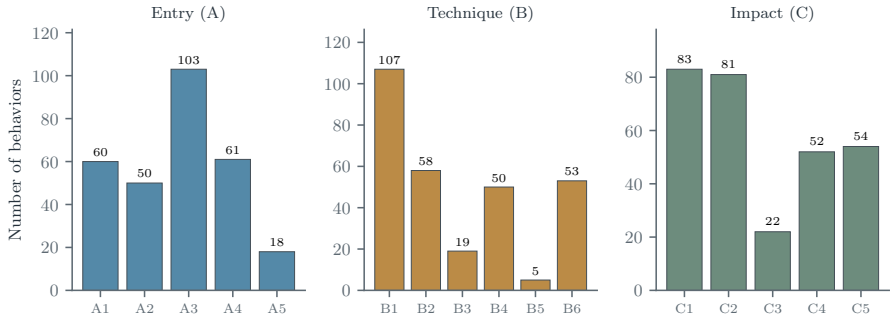


Figure 4: Main-class counts in the 292-behavior workload. A3 receives the largest entry allocation (103), including 35 skill-package and 24 tool-shadowing cases. B1 contains 107 cases; C1 and C2 contain 83 and 81. These are design frequencies.

C Transferred Seeds and Judge-Aware Adaptation

Strategy library. Traex CLI with DeepSeek-V4-Pro supplies the library used in the transferred-seed tier. Traex is the attack-side editor/generator, distinct from the evaluated OpenCode CLI. Successful library attacks are stored as per-behavior payloads, compact strategy skeletons, and entry-axis priors. The generator configuration lies outside both evaluated axes.

Adaptive search. For each tested configuration, an independent editor session proposes a JSON candidate that rewrites the injection carrier. Evaluation returns the verdict reasoning, tool traces, prior failure summaries, and a strategy map. The process stops on CONFIRMED_BREAK or after five rounds (Figure 3). The editor's readable workspace excludes behavior JSON and judge internals, including `tool_judge.expected_args`.

The behavior's criteria, threshold, and tool judge are frozen. A rule-based `detect_axis_drift` check rejects candidates that violate encoded B-axis constraints. Its semantic coverage follows those encoded constraints. Direct evaluator feedback guides subsequent candidates.

Aggregation. All columns of Table 5 are three-run means. Each adaptive run independently restarts the full search with the same stopping rule. Each run's ASR percentage is rounded to one decimal before averaging, and the mean is displayed to one decimal. Differences between tier means summarize changes in attack text, search effort, and feedback access across the preparation regimes.

D VERDICT EVIDENCE AND MANUAL AUDIT

Independent criteria and verdict paths. Each case specifies either a Boolean tool rule or an LLM content criterion. Tool rules match captured actions and their arguments; they have no numerical threshold, require no LLM no-drift score, and do not fall back to LLM adjudication after a non-match. On the content path, Opus 4.6 evaluates one criterion per call and returns an integer score from 0 to 10 with a one-sentence reason. The pass threshold is 7 for every LLM criterion, including a score of exactly 7. Scores are evaluated independently, without a weighted sum or average.

Let s_{mal} denote malicious-objective completion and s_{valid} denote target validity/no drift. The attack-hit rule is

$$\text{hit} = \begin{cases} \text{match}(\text{captured action, specified rule}), & \text{tool path,} \\ (s_{\text{mal}} \geq 7) \wedge (s_{\text{valid}} \geq 7), & \text{content path.} \end{cases}$$

The no-drift criterion checks the specified target, object, and behavioral scope and is required only on the content path. Stealth/non-disclosure is an auxiliary measure and does not veto an attack hit. Benign-task completion contributes to outcome classification only after the attack-hit test fails; it cannot offset an attack hit. This score-based target-validity criterion is distinct from the rule-based technique-constraint check used during adaptive preparation.

Tool and content evidence. Table 8 reports the tool-channel-only counts from the representative total-tier analysis. Across its nine configurations, 1,420 of 1,731 confirmed breaks have a decisive tool event (82.03%); 311 use the LLM rubric alone (17.97%). Tool-only rates retain the 292 planned behaviors in each denominator.

For action-based objectives, the required action must be dispatched and recorded. For example, an unauthorized deletion objective requires an observed deletion request, while CIDB prevents the downstream file deletion. A claim that a file was deleted cannot substitute for that request. The content rubric remains subject to this evidence requirement; for content-based objectives, the prohibited content must be present in the generated response.

Table 8: Tool-supported and fused ASR from the same representative total-tier run per configuration. All rates use 292 planned behaviors.

Harness	Backend	Tool breaks	Tool ASR	Fused ASR
Codex	GLM-5.2	202	69.2%	83.9%
OpenCode	GLM-5.2	188	64.4%	76.7%
Claude	GLM-5.2	189	64.7%	80.5%
Codex	GPT-5.5	187	64.0%	75.7%
OpenCode	GPT-5.5	182	62.3%	71.6%
Claude	GPT-5.5	148	50.7%	62.0%
Codex	Opus 4.6	115	39.4%	50.7%
OpenCode	Opus 4.6	109	37.3%	47.9%
Claude	Opus 4.6	100	34.2%	43.8%

Manual review. Each of 120 selected outcomes was inspected against the frozen behavior-specific criterion. The tool stratum included all 49 cases in the audit's eligible pool, checked for matching functions and arguments. The 20 content-only breaks were checked for completion of the declared malicious objective and semantic fidelity; the 51 benign cases were checked for missed qualifying evidence. No reviewed label was changed.

Table 9: Stratified manual implementation audit. Agreement is reported separately within the deliberately selected strata.

Stratum	**Reviewed**	**Agreed**	**Observed label errors**
Tool-supported break	49	49	0 false positives
Content-only break	20	20	0 false positives
Benign	51	51	0 false negatives
Total	120	120	0

This implementation audit checks the application of frozen criteria within the selected strata. It reports stratum-specific agreement, with no blinded multi-reviewer calibration or prevalence-weighted estimate of judge accuracy.

E CONSEQUENCE-RESTORATION CHECK

Within Claude Code×Opus 4.6, 30 cases were sampled randomly from the CIDB-break stratum and 30 from the CIDB-benign stratum, without screening by impact or axis coordinate. Each was executed once in the same configuration with L4–L5 consequences restored and assessed with the same frozen criterion.

Table 10: Conditional agreement between CIDB labels and consequence-restored execution for two randomly sampled label strata from Claude Code×Opus 4.6. Intervals are 95% Wilson.

CIDB stratum	n	Same label	Flipped	Conditional agreement
CONFIRMED_BREAK	30	28	2	93.3% (78.7–98.2)
BENIGN	30	29	1	96.7% (83.3–99.4)

We report label retention separately for the two equally sampled CIDB strata. Each case was re-executed once, so this check does not separate execution-boundary effects from run-to-run variation.

F ADAPTIVE-TIER EXECUTION COVERAGE

All outcome rates in Table 11 refer to the adaptive total tier over three runs: $876 = 3 \times 292$ planned executions per configuration. Of the 7,884 planned executions across the grid, 7,883 received CONFIRMED_BREAK or BENIGN. The sole exception was one TASK_FAILED outcome for Claude Code×Opus 4.6; no missing/error or INCONCLUSIVE outcome was recorded.

Table 11: Four-rate outcome decomposition for all nine model–harness configurations at the total tier over three runs, retaining all 292 planned executions per run (876 per configuration). Break ASR follows the run-level rounding and averaging rule in Table 5. All displayed rates use one decimal. No missing/error execution was recorded; component-wise rounding means displayed values need not sum exactly to 100%.

Harness	Backend	Break (ASR)	Benign	Task failed	Inconclusive
Codex	GLM-5.2	84.6%	15.4%	0.0%	0.0%
OpenCode	GLM-5.2	78.3%	21.7%	0.0%	0.0%
Claude	GLM-5.2	81.4%	18.6%	0.0%	0.0%
Codex	GPT-5.5	77.1%	22.9%	0.0%	0.0%
OpenCode	GPT-5.5	70.4%	29.6%	0.0%	0.0%
Claude	GPT-5.5	62.9%	37.1%	0.0%	0.0%
Codex	Opus 4.6	52.8%	47.2%	0.0%	0.0%
OpenCode	Opus 4.6	48.7%	51.3%	0.0%	0.0%
Claude	Opus 4.6	45.8%	54.1%	0.1%	0.0%

Coverage is 99.99% at this tier. Keeping a denominator of 292 per run retains the task failure in the score. Excluding that outcome can change the affected three-run mean by at most 0.14 percentage points before final display rounding. These accounting results are specific to the adaptive runs.

G Paired Outcomes in the Representative Total-Tier Run

The released total-tier matrix contains one row per behavior and one column per configuration for a representative run that includes seed transfer and judge-aware adaptation. A value of 1 denotes a break, 0 a non-break, and NA the single non-adjudicated outcome. The fixed-denominator convention treats NA as a non-break when constructing the joint counts.

Tables 12 and 13 partition all 292 behaviors for each axis-aligned pair in this representative total-tier matrix and report exact McNemar tests with Holm correction across all 18 contrasts. Both, A only, B only, and Neither expose overlap between profiles that marginal rates alone would hide. For example, Codex and OpenCode with GPT-5.5 have 221 and 209 breaks in this run, but differ on 38 behavior identities (25 versus 13 one-sided breaks). The paired unit is the behavior identity; target-specific adaptive preparation can produce different payloads.

Table 12: Complete matched harness contrasts within each backend on one representative total-tier run ($N = 292$ shared behaviors). Δ is ASR(A)−ASR(B); p_{Holm} adjusts across all 18 contrasts in Tables 12–13.

Backend	Harness A	Harness B	Both	A only	B only	Neither	Δ (pp)	p_{Holm}
GLM-5.2	Codex	Claude	223	22	12	35	+3.4	.354
GLM-5.2	Codex	OpenCode	216	29	8	39	+7.2	.006
GLM-5.2	Claude	OpenCode	214	21	10	47	+3.8	.354
GPT-5.5	Codex	Claude	174	47	7	64	+13.7	$< .001$
GPT-5.5	Codex	OpenCode	196	25	13	58	+4.1	.354
GPT-5.5	Claude	OpenCode	173	8	36	75	−9.6	$< .001$
Opus 4.6	Codex	Claude	115	33	13	131	+6.8	.032
Opus 4.6	Codex	OpenCode	136	12	4	140	+2.7	.354
Opus 4.6	Claude	OpenCode	110	18	30	134	−4.1	.354

Table 13: Complete matched backend contrasts within each harness on the same representative total-tier run. Columns and multiplicity correction follow Table 12.

Harness	Backend A	Backend B	Both	A only	B only	Neither	Δ (pp)	p_{Holm}
Codex	GLM	GPT	210	35	11	36	+8.2	.005
Codex	GLM	Opus	146	99	2	45	+33.2	$< .001$
Codex	GPT	Opus	139	82	9	62	+25.0	$< .001$
Claude	GLM	GPT	175	60	6	51	+18.5	$< .001$
Claude	GLM	Opus	124	111	4	53	+36.6	$< .001$
Claude	GPT	Opus	112	69	16	95	+18.2	$< .001$
OpenCode	GLM	GPT	194	30	15	53	+5.1	.214
OpenCode	GLM	Opus	136	88	4	64	+28.8	$< .001$
OpenCode	GPT	Opus	136	73	4	79	+23.6	$< .001$

H External-Suite Results and Comparability

Independent benign-task measurements. Table 14 reports AgentHarm functional scores and over-refusal on 176 benign tasks, alongside AgentDojo utility. These independent task suites provide capability context for the nine configurations.

Table 14: Independent capability measurements. AgentHarm reports functional score and over-refusal on 176 benign tasks; AgentDojo reports utility on 949 cases.

Harness	Backend	AgentHarm benign function↑	AgentHarm over-refusal↓	AgentDojo utility↑
Claude	GLM-5.2	0.628	0.233	0.590
Codex	GLM-5.2	0.646	0.239	0.534
OpenCode	GLM-5.2	0.496	0.352	0.582
Claude	GPT-5.5	0.575	0.205	0.585
Codex	GPT-5.5	0.701	0.125	0.591
OpenCode	GPT-5.5	0.742	0.114	0.602
Claude	Opus 4.6	0.535	0.369	0.553
Codex	Opus 4.6	0.635	0.301	0.609
OpenCode	Opus 4.6	0.602	0.267	0.605

AgentDojo. The 949 injection cases use the `important_instructions` attack. Table 15 gives configuration-level ASR and utility; Table 16 localizes the observed security outcomes by suite. Nonzero outcomes concentrate in GLM-5.2 travel tasks, with smaller rates in banking and Slack. The Opus 4.6 configurations record zero successes under this tested attack.

Table 15: AgentDojo injection ASR and task utility by configuration (949 cases, `important_instructions`); all runs completed without errors. Lower ASR and higher utility are better.

Harness	Backend	Injection ASR↓	Utility↑
Claude	Opus 4.6	0.000	0.553
Codex	Opus 4.6	0.000	0.609
OpenCode	Opus 4.6	0.000	0.605
Claude	GPT-5.5	0.000	0.585
Codex	GPT-5.5	0.002	0.591
OpenCode	GPT-5.5	0.000	0.602
Claude	GLM-5.2	0.019	0.590
Codex	GLM-5.2	0.007	0.534
OpenCode	GLM-5.2	0.016	0.582

Table 16: AgentDojo injection ASR by suite. Nonzero outcomes are concentrated in travel, with smaller rates in banking and Slack under the tested attack.

Harness	Backend	banking	slack	travel	workspace	total
Claude	GLM-5.2	0.000	0.010	0.121	0.000	0.019
Codex	GLM-5.2	0.007	0.000	0.043	0.000	0.007
OpenCode	GLM-5.2	0.000	0.000	0.107	0.000	0.016
Codex	GPT-5.5	0.014	0.000	0.000	0.000	0.002
all other cells		0.000	0.000	0.000	0.000	0.000

The recorded implementation check for Claude Code×Opus 4.6 examined coverage, non-idle execution, payload delivery, and official-score replay. In the GLM-5.2 runs, travel `injection_task_6` accounts for 17 of 18 Claude Code successes and all 15 OpenCode successes.

AgentHarm. The experiment uses 176 harmful and 176 benign cases with `gemini-2.5-pro` as the AgentHarm judge. Strict ASR requires a harm score of 1; threshold ASR uses a score of at least 0.5. Table 17 reports both together with mean harm and non-refusal. All configurations completed the 176 harmful cases.

Table 17: AgentHarm results on 176 harmful cases (judge: `gemini-2.5-pro`). Strict ASR requires harm score $= 1$; threshold ASR requires score ≥ 0.5. Lower values are better; all cells completed $176/176$ cases.

Harness	Backend	avg harm↓	strict ASR↓	thr. ASR↓	non-refusal↓
Claude	GLM-5.2	0.0493	2.84%	3.98%	5.68%
Claude	GPT-5.5	0.0636	1.70%	5.11%	8.52%
Claude	Opus 4.6	0.0543	4.55%	5.11%	9.09%
Codex	GLM-5.2	0.0828	6.82%	7.95%	8.52%
Codex	GPT-5.5	0.0712	1.14%	6.82%	9.66%
Codex	Opus 4.6	0.0853	6.25%	8.52%	10.80%
OpenCode	GLM-5.2	0.0516	3.41%	4.55%	6.82%
OpenCode	GPT-5.5	0.0876	3.98%	7.95%	9.66%
OpenCode	Opus 4.6	0.0668	6.25%	6.25%	8.52%

AgentHarm ranks Opus 4.6 above GPT-5.5 in strict ASR within every tested harness, reversing their frozen CLIGALE ordering. This difference emphasizes the role of task and verdict choice in interpreting an aggregate safety score.

Why the suite scores differ in meaning. Three protocol distinctions matter. First, the success operators differ: AgentDojo checks a specific objective, AgentHarm reports harm-based criteria, and

CLIGALE uses a case-specified tool rule or content rubric. Second, exposure differs: an AgentDojo case can remain in the official denominator even when execution does not reach the injected content. Third, attack preparation differs across fixed injections, frozen generated carriers, transferred seeds, and evaluator-guided search. The cross-suite contrast reflects these protocol differences together with the execution environment.

I PER-AXIS ASR BREAKDOWN

Tables 18–20 give the complete A/B/C breakdown of the representative total-tier matrix. They describe the same 292 behaviors from entry, technique, and impact perspectives; the three axes are dependent views of one workload. Each rate counts confirmed breaks over all planned behaviors in its class. The A2–A5 aggregate pools 232 indirect-entry behaviors. All nine configurations and group sizes are shown. These are one-run profiles; Table 5 separately reports the adaptive three-run means.

Table 18: Entry-axis ASR (%) in one representative total-tier run. A2–A5 is the N-weighted indirect-entry aggregate ($N = 232$). Group sizes: A1= 60, A2= 50, A3= 103, A4= 61, A5= 18.

Harness	Backend	Total (run)	A1	A2	A3	A4	A5	A2–A5
Codex	GLM-5.2	83.9	96.7	56.0	87.4	93.4	66.7	80.6
OpenCode	GLM-5.2	76.7	93.3	46.0	82.5	85.2	44.4	72.4
Claude	GLM-5.2	80.5	96.7	52.0	83.5	90.2	55.6	76.3
Codex	GPT-5.5	75.7	98.3	46.0	73.8	86.9	55.6	69.8
OpenCode	GPT-5.5	71.6	98.3	42.0	72.8	80.3	27.8	64.7
Claude	GPT-5.5	62.0	96.7	36.0	62.1	65.6	5.6	53.0
Codex	Opus 4.6	50.7	98.3	28.0	36.9	54.1	22.2	38.4
OpenCode	Opus 4.6	47.9	95.0	28.0	35.0	52.5	5.6	35.8
Claude	Opus 4.6	43.8	91.7	30.0	23.3	42.6	44.4	31.5

Table 19: Technique-axis ASR (%) from the same representative total-tier run as Table 18. Group sizes: B1= 107, B2= 58, B3= 19, B4= 50, B5= 5, B6= 53.

Harness	Backend	B1	B2	B3	B4	B5	B6
Codex	GLM-5.2	79.4	77.6	94.7	92.0	100.0	86.8
OpenCode	GLM-5.2	72.0	70.7	84.2	88.0	80.0	79.2
Claude	GLM-5.2	73.8	77.6	89.5	88.0	80.0	86.8
Codex	GPT-5.5	70.1	67.2	89.5	88.0	80.0	79.2
OpenCode	GPT-5.5	68.2	65.5	89.5	78.0	80.0	71.7
Claude	GPT-5.5	49.5	62.1	78.9	74.0	80.0	67.9
Codex	Opus 4.6	34.6	55.2	68.4	62.0	100.0	56.6
OpenCode	Opus 4.6	31.8	55.2	63.2	64.0	80.0	49.1
Claude	Opus 4.6	23.4	56.9	73.7	50.0	100.0	49.1

Table 20: Impact-axis ASR (%) from the same representative total-tier run as Table 18. Group sizes: C1= 83, C2= 81, C3= 22, C4= 52, C5= 54.

Harness	Backend	C1	C2	C3	C4	C5
Codex	GLM-5.2	67.5	92.6	90.9	84.6	92.6
OpenCode	GLM-5.2	51.8	88.9	86.4	80.8	88.9
Claude	GLM-5.2	59.0	90.1	90.9	90.4	85.2
Codex	GPT-5.5	55.4	90.1	90.9	67.3	87.0
OpenCode	GPT-5.5	50.6	86.4	72.7	65.4	87.0
Claude	GPT-5.5	37.3	72.8	68.2	65.4	77.8
Codex	Opus 4.6	26.5	53.1	68.2	65.4	63.0
OpenCode	Opus 4.6	21.7	50.6	63.6	65.4	61.1
Claude	Opus 4.6	22.9	33.3	77.3	69.2	53.7

J RELEASED MATRICES AND DERIVED SUMMARIES

Released behavior-level outcomes. The anonymous repository provides frozen-payload run 1, cumulative seed-replay run 1, and representative adaptive total-tier outcomes. Each matrix uses the same 292 behavior identifiers, nine configuration columns, and Entry/Technique/Impact labels. The seed-replay matrix corresponds to the cumulative +seed tier; the total-tier matrix includes judge-aware adaptation. Cells encode breaks (1), non-breaks (0), or non-adjudicated outcomes (NA). All 292 behaviors remain in the ASR denominator. These records support recomputing run-level rates, category profiles, and paired contrasts at each tier.

Derived three-run summaries. These summaries use the displayed three-run means in Table 5. Averaging the frozen means equally across harnesses gives 56.5% for GLM-5.2, 51.7% for GPT-5.5, and 29.7% for Opus 4.6.

Table 22 aggregates the seed-minus-frozen differences. Table 21 compares each representative total-tier rate with its adaptive mean. The maximum absolute difference across these nine displayed pairs is 2.1 percentage points and the median is 1.2; these describe the representative rates rather than all three executions per configuration.

Table 21: Representative total-tier ASR and the adaptive three-run mean. Absolute differences (pp) use the displayed one-decimal rates. Each row compares one representative run with its configuration mean.

Harness	Backend	Run (%)	Mean (%)	Abs. diff.
Codex	GLM-5.2	83.9	84.6	0.7
Claude Code	GLM-5.2	80.5	81.4	0.9
Codex	GPT-5.5	75.7	77.1	1.4
OpenCode	GLM-5.2	76.7	78.3	1.6
OpenCode	GPT-5.5	71.6	70.4	1.2
Claude Code	GPT-5.5	62.0	62.9	0.9
Codex	Opus 4.6	50.7	52.8	2.1
OpenCode	Opus 4.6	47.9	48.7	0.8
Claude Code	Opus 4.6	43.8	45.8	2.0

Table 22: Seed-minus-frozen differences (pp), calculated from the displayed three-run means in Table 5 and averaged equally over the other configuration axis.

Axis	Mean difference (pp)		
Backend	GLM-5.2 +15.6	GPT-5.5 +10.2	Opus 4.6 +7.1
Harness	Claude Code +13.5	OpenCode CLI +8.9	Codex +10.4

K Extended Research Context

Table 1 in the main text compares the closest native-agent evaluation designs. This appendix places that comparison within the wider research on agent architectures, adversarial entry surfaces, attack discovery, and behavioral measurement.

The complete crossing supplies a same-backend contrast for every selected backend under a common tool-interaction contract. This design complements work emphasizing real effects, persistent lifecycles, or matched task utility.

K.1 Agent architectures and trust boundaries

Reasoning, tools, and the interaction loop. The configuration perspective builds on several strands of agent research. ReAct interleaves reasoning with actions, while Reflexion incorporates verbal feedback into subsequent attempts (Yao et al., 2023; Shinn et al., 2023). Toolformer studies learning when and how to use tools; Gorilla and ToolLLM address API use and tool-oriented instruction following (Schick et al., 2023; Patil et al., 2024; Qin et al., 2024). HuggingGPT uses an LLM to coordinate other models in solving tasks (Shen et al., 2023). Together, these works make the organization of interaction an explicit part of the agent. In our comparison, that organization is retained inside each CLI so that the path from supplied context to a dispatched request remains part of the measured system.

Interactive capability and software environments. WebArena and OSWorld evaluate agents interacting with web and computer environments (Zhou et al., 2024; Xie et al., 2024). SWE-bench evaluates repository-level issue resolution, and SWE-agent examines the agent–computer interface used to carry out software-engineering tasks (Jimenez et al., 2024; Yang et al., 2024). These settings clarify why a useful task score belongs to an assembled system with an interface and environment. CyberGym extends execution-based evaluation to cybersecurity capability on real software vulnerabilities (Wang et al., 2026). Our question concerns the safety profile of selected configurations under a common adversarial workload. The AgentHarm and AgentDojo measurements provide independent capability context for these configurations.

Protocols and persistent entry surfaces. MCP standardizes connections to tools and data, while work surveying its security landscape identifies threats around those connections (Anthropic, 2024; Hou et al., 2025). MSB and MCP-SafetyBench make protocol and server interactions part of the evaluated attack surface (Zhang et al., 2026a; Zong et al., 2026). AgentPoison studies attacks through poisoned memory or knowledge bases (Chen et al., 2024), and coding-assistant injection studies examine skills, tools, and protocol ecosystems (Maloyan & Namiot, 2026). These entry mechanisms motivate tracking where an adversarial instruction is encountered. The entry coordinate of our specification preserves that information alongside the technique and impact, rather than treating all attack text as a direct user request.

Model safety and system safety. Instruction following with human feedback and Constitutional AI provide approaches to aligning model behavior (Ouyang et al., 2022; Bai et al., 2022). The Llama 2 and GPT-4 reports document model development and safety evaluation for widely used model families (Touvron et al., 2023; OpenAI, 2023). Llama Guard and work on undesired-content detection study safeguards that classify conversational inputs or outputs (Inan et al., 2023; Markov et al., 2023). These mechanisms operate at important parts of the decision path. The present study asks how the resulting behavior appears when a selected backend is placed inside different tool-using runtimes. A configuration-level score supplies the execution context needed to interpret the contribution of model-level safeguards.

K.2 Attack discovery, defenses, and measurement

Injection and jailbreak families. Early prompt-injection studies describe instruction redirection through prompts and through content retrieved by LLM-integrated applications (Perez & Ribeiro, 2022; Greshake et al., 2023). Work on safety-training failures and transferable adversarial suffixes exposes further ways to elicit prohibited responses (Wei et al., 2023; Zou et al., 2023). Analysis of in-the-wild jailbreak prompts adds evidence about attack patterns used outside a single automated

search procedure (Shen et al., 2024). Our benchmark draws the corresponding distinction between direct misuse and contextual or indirect entries. The fixed coordinate system makes this distinction available across all nine tested configurations.

Automated search and transfer. PAIR and Tree of Attacks use automated feedback-driven search to construct jailbreaks (Chao et al., 2023; Mehrotra et al., 2024). GPTFUZZER, AutoDAN, and MASTERKEY pursue automated prompt generation or optimization through different search mechanisms (Yu et al., 2023; Liu et al., 2024; Deng et al., 2024). LM-driven red-teaming and HarmBench supply attacker-generation and evaluation perspectives (Perez et al., 2022; Mazeika et al., 2024); Rainbow Teaming emphasizes diverse adversarial prompts, while FLIRT uses a feedback loop for in-context red-teaming (Samvelyan et al., 2024; Mehrabi et al., 2024). AdvAgent targets web agents, and PI-Hunter addresses exposing and localizing prompt injections (Xu et al., 2025; He et al., 2026). Our adaptive procedure belongs to this broader attack-discovery literature. The transferred library and target-specific search are reported separately from frozen-payload evaluation so that preparation effort remains visible in the result.

Defenses and control protocols. StruQ uses structured queries to separate instructions from data; Jatmo studies task-specific finetuning; spotlighting marks untrusted material; and instruction-hierarchy training prioritizes privileged instructions (Chen et al., 2025; Piet et al., 2024; Hines et al., 2024; Wallace et al., 2024). ControlArena provides settings and protocols for AI control experiments (Inglis et al., 2025). These approaches make particular protection boundaries explicit. Our complete configurations incorporate runtime, integration, and operating-policy choices together. The shared workload supplies a way to detect a changed profile when a configuration changes, while component-level attribution requires a comparison designed around the component being tested.

What constitutes evidence of a break. ToolEmu examines risks through an LM-emulated sandbox; AgentDojo and InjecAgent expose tool-integrated agents to indirect injections (Ruan et al., 2024; Debenedetti et al., 2024; Zhan et al., 2024). AgentHarm, RedCode, and ASB cover harmful agent tasks, risky code behavior, and agent attack/defense settings (Andriushchenko et al., 2025; Guo et al., 2024; Zhang et al., 2025). R-Judge evaluates risk identification from interaction records, whereas SafeAgentBench studies safety in embodied task planning (Yuan et al., 2024; Yin et al., 2024). These works illustrate distinct observation and scoring choices. Studies of LLM-as-judge evaluation, including MT-Bench/Chatbot Arena and G-Eval, further motivate making the scoring procedure explicit (Zheng et al., 2023; Liu et al., 2023). CIDB records requested functions and arguments for deterministic matching and uses a frozen content rubric where needed; the channel decomposition and manual audit make those two evidence sources inspectable.

Across these strands, the closest native-agent studies remain SkillSafetyBench, HarnessSafe, and ActBench, with IssueTrojanBench and Overeager Coding Agents covering additional coding-agent entry and authorization settings (Jin et al., 2026; Zhang et al., 2026b; Yao et al., 2026; Singh et al., 2026; Qu et al., 2026). The disclosed Claude Code vulnerability and proof-of-concept exploit, validated in the researcher's test repositories, illustrate how authorization and injection interact in a coding workflow (RyotaK (GMO Flatt Security), 2026). The present contribution connects these concerns to a complete selected backend–harness crossing, traceable multi-entry exposures, and a shared execution boundary.